Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Request Segmentation and Metadata Leakage in Low-Rate GPUPower Telemetry: A Single-Platform Audi

Ali Hamza

akhamza@edu.hse.ru

National Research University Higher School of Economics

Chih-Yi Chiu

National Chung Cheng University

Aziz Ahmad

National Research University Higher School of Economics

Shezad Arian

National Research University Higher School of Economics

Hassan iftikhar

Applied AI Institute

Research Article

Keywords: GPU security, power telemetry, side channels, request segmentation, measurement methodology, model inference

Posted Date: October 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11183309/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Request Segmentation and Metadata Leakage in Low-Rate GPU Power Telemetry: A Single-Platform Audit

Ali Hamza[1,*], Chih-Yi Chiu[2], Aziz Ahmad[1], Shehzad[1], Hassan Iftikhar[3]

[1]HSE University, Moscow, Russia
[2]Department of Computer Science and Information Engineering, National Chung Cheng University, Minhsiung, Chiayi, Taiwan
[3]Applied AI Institute
[*]Corresponding author: akhamza@edu.hse.ru

Abstract

Software-accessible graphics processing unit power telemetry can expose workload metadata, but request boundaries are often easier to obtain in an experiment than in an attack. We audit this distinction on one NVIDIA RTX 2080 Ti using continuous telemetry from two GPT-2 models. A post-hoc reanalysis preserves an archived development, validation, and test split comprising 36 sessions, including ten test sessions with 583 monitored requests. The archived passive detector and classifier achieve class-balanced correct identification of 0.076 over all requests. An oracle-window classifier trained on the identical 144 development events achieves 0.764; using all 1,043 observable development requests raises it to 0.887. The detected score exceeds its 0.052 coverage-adjusted random-label reference. Only 60 requests are matched, all generation requests. The duration and overlap rules mechanically exclude 110 of 117 prefill requests; relaxing duration alone still finds none. A validation-selected candidate-change detector, with its classifier retrained, increases the all-request score to 0.127 but produces more false alarms. Replacing degenerate normalization using development-derived scales yields 0.100 for the validation-selected pipeline. Amplitude-only and timing-only ablations both retain conditional signal. Two new ten-second hardware measurements observe approximately 9.8 power-value changes per second despite more than 1,000 interface calls per second. These results support evaluating detection coverage, false alarms, training supervision, and conditional inference together. They establish limitations of evaluated pipelines on one platform, rather than a general security guarantee.

Keywords: GPU security; power telemetry; side channels; request segmentation; measurement methodology; model inference

1 Introduction

System monitoring interfaces are useful for resource management and can also reveal information across workload boundaries. The NVIDIA Management Library (NVML) exposes device measurements whose interpretation depends on the architecture and query [1]. A process that can read these measurements may observe activity produced by other workloads on the device. Whether it can turn that activity into request-level information depends on identifying the requests as well as distinguishing their features.

These are separate problems. Cropping a trace at victim-provided start and end times supplies a boundary oracle. A classifier trained on such windows measures information available given the oracle. A passive

observer of a continuous trace must additionally detect requests, separate adjacent operations, and distinguish relevant work from background activity. A high score on successfully located requests can coexist with low coverage. Conversely, low coverage does not make the successfully inferred metadata harmless.

This paper evaluates that distinction on a single RTX 2080 Ti and two GPT-2 model sizes. Its contribution is an auditable evaluation protocol for continuous telemetry: a value-change cadence check; common-denominator detected/oracle comparisons; a boundary comparison with identical training-event identities; and complete validation-selected detector pipelines. The primary reanalysis holds the archived detector configuration fixed. Additional analyses are explicitly post-hoc because the test traces had already been examined in an earlier version. No result in this revision is presented as a newly protected test-set evaluation.

The findings require a narrower claim than "low-rate power telemetry does not leak." The archived power-and-time pipeline correctly identifies 44 of 583 requests after matching. Holding its 144 development-event identities fixed, changing from detected to oracle windows raises the all-request score from 0.076 to 0.764; additional oracle training data raise it to 0.887. Amplitude features retain signal, and a complete alternative pipeline reaches 0.127. The practical lesson is to disclose boundary supervision and training support and to report misses and false alarms alongside inference scores.

2 Related work and scope

GPU side channels comprise several distinct observation mechanisms. Rendered Insecure demonstrates attacks using allocation behavior, counters, and timing [2]; its attacker capabilities differ from passive NVML polling. Luo et al. study power analysis of a GPU AES implementation [3]. Maia et al. reconstruct network structure from magnetic measurements at a GPU power cable [4]. Recent work in the target journal further illustrates this diversity: Bepary et al. develop a gate-level electromagnetic leakage model for pre-silicon assessment [9], and Rao et al. survey side-channel and related hardware-security threats across the microelectronics lifecycle [10]. These experiments establish that accelerator activity can expose structure, but their sensors, attacker access, and boundary assumptions differ from software-readable management telemetry.

DeepTheft reconstructs neural-network architectures from processor energy measurements through the RAPL interface [5]. It is not an NVML GPU-power experiment. Energon is more directly related: it studies GPU power and thermal observations for transformer architecture inference, including model-family and hyperparameter classification [6]. A recent systematization emphasizes that neural-network side-channel results depend strongly on threat models, platforms, objectives, and analysis stages [7]. Our experiment neither reproduces those attacks nor refutes their reported accuracy. Its novelty is methodological: it treats request localization as a separately measured attack stage, counts its misses on the inference denominator, controls the identities of training events across detected and oracle windows, and reruns classification for validation-selected detectors. We do not claim the first GPU telemetry side channel or a new architecture-extraction attack.

Yang et al. characterize NVIDIA built-in power measurement across more than 70 GPUs, including update frequency, transient response, and architecture-dependent sampling behavior [8]. Their work establishes that query frequency and sensor behavior must be distinguished. Baddour et al. provide a complementary target-journal example: SideLink uses NVLink contention to build covert and application-fingerprinting channels across H200, A100, and V100 systems [11]. Its interconnect-latency observations and multi-GPU threat model differ from our single-device NVML power telemetry, but reinforce that the observable and boundary definition must be stated explicitly. Our two cadence runs are a local platform check, not a new discovery of that distinction or a measurement of internal sensor bandwidth. Unlike the measurement-calibration objective

of Yang et al., the architecture-inference objective of Energon, and the cross-attack taxonomy of the SoK, our specific question is how localization misses, training-event support, and detector implementation affect request-level identification on a common denominator. Characteristics measured on other GPU generations are not assumed to apply to this device.

We study identification of two known model sizes and prediction of generated-token count. We do not recover prompt contents, weights, cryptographic keys, or arbitrary model architectures. Auxiliary defense and cryptographic experiments do not establish the request-segmentation claim. Online Resource 1 records their limitations and distinguishes their evidential scope from the primary corpus.

3 Threat model and measurement protocol

3.1 Observer and victim

The observer reads device-level power, utilization, streaming-multiprocessor clock, and temperature from the tested host's accessible NVML interface. It receives continuous timestamps but no victim event timestamps, model labels, or output-token counts during segmentation. Ground truth is stored separately and is used for supervised development and evaluation. The observer has representative labeled development sessions from the same two models. This is a closed-set identification experiment, not an unknown-model discovery task.

No active contention probe is used in the 36 selected sessions. Nevertheless, the primary detector uses multiple passive telemetry channels. We reserve "power-and-time classifier" for its inference features; the complete pipeline is not power-only. Normalization uses full-trace statistics and smoothing is centered. The detector is therefore an offline passive observer, not a demonstrated causal streaming attack. Access to global telemetry is a deployment assumption established for this host, not a statement about permissions in every cloud or driver configuration.

3.2 Platform, requests, and sessions

The collection manifests identify an NVIDIA GeForce RTX 2080 Ti and driver 595.79. Both GPT-2 Small (124 million parameters) and GPT-2 Large (774 million parameters) are loaded before recording, and both receive a warm-up generation. Consequently, the label "clean cold" means no additional preheating; it does not mean a GPU untouched since boot. Other clean conditions add 10 or 30 seconds of preheating. The collector records at a requested 200 Hz; actual timestamps, rather than the requested rate, determine durations.

Each session schedules 60 events by seeded random selection across the two models, five fixed prompts, and requested generation limits of 0, 16, 32, 64, or 128 tokens. Zero denotes a prefill-only forward pass. The collector processes one prompt at a time (batch size one) with Hugging Face model identifiers `gpt2` and `gpt2-large`. It loads the models through `AutoModelForCausalLM`, without an explicit precision override or autocast. Generation uses deterministic decoding with caching, and the stored target is the actual number of generated tokens, calculated from the returned sequence length. Synchronization surrounds the measured GPU operation. Thus the target interval excludes input tokenization and is not a complete network-service latency.

The session manifests do not record the collection operating-system build, PyTorch/Transformers/CUDA versions, model repository revisions, or actual parameter dtype. Earlier project records describe a Windows 11 host, but do not provide a versioned environment bound to this continuous corpus. We therefore do not substitute current installed versions or earlier dataset settings for the missing records. Online Resource 2 pins the CPU reanalysis environment; it is not a complete reconstruction of the collection environment.

The ten conditions are clean with three preheat settings, CPU contention, three GPU-background-load

levels, open-world distractors, back-to-back requests, and long idle gaps. GPU loads use target duty fractions 0.20, 0.50, and 0.80; these are configured load settings, not measured utilization guarantees. Back-to-back gaps are 20–100 ms and long gaps are 2–5 s. The open-world schedule replaces events with unrelated matrix multiplications with probability 0.30. This limited distractor set does not represent unrestricted production traffic.

The preserved split contains 18 development, eight validation, and ten test sessions, with 1,044, 465, and 583 monitored requests, respectively. Test exposure totals 1,134.85 s (18.91 min). There is one test session per condition. The test set contains 290 Large and 293 Small requests; 466 are generation and 117 are prefill-only. Splitting by session avoids putting neighboring windows from the same trace into both training and testing, but all sessions share a device, workload definitions, and collection software. This is not cross-device generalization.

3.3 New hardware cadence measurements

We repeated tight-loop power polling twice for ten seconds on 27 September 2026: once with no load injected by the audit script and once with a PyTorch matrix-multiplication load. The first run had 11,893 reads and 98 value changes; the second had 10,372 reads and 98 changes. Call rates were 1,189.3 and 1,036.9 Hz, while value-change rates were both approximately 9.80 Hz. Median intervals between changes were 100.67 and 101.29 ms; repeated-read fractions were 99.18% and 99.06%.

The no-injected-load trace ranged from 40.7 to 274.9 W, so it is not a verified idle baseline. The loaded trace ranged from 232.6 to 268.1 W. The measurements establish repeated observations and an approximately 100-ms value-change cadence under these conditions. They do not identify the internal sensor integration window, establish independent samples, or prove that no unobserved refresh returns the same value. API-call frequency must therefore be distinguished from value-change frequency and information bandwidth. Raw polling records are included in Online Resource 2.

4 Evaluation methods

4.1 Detection and event matching

The archived detector forms $s = 0.55z_P + 0.30z_U + 0.15z_C$, where P, U, and C are power, utilization, and clock. For each channel, the baseline is the lowest quartile of the full trace; z subtracts its median and divides by the larger of 1.4826 times its median absolute deviation or 10^{-6}. A three-sample moving average smooths s. Active runs exceed threshold 2, gaps up to 0.1 s are merged, and windows shorter than 0.2 s are discarded. This archived configuration was selected on validation data with mean F1 0.191. We preserve it for the primary reproduction.

Predicted and true intervals are paired greedily by descending intersection over union (IoU), subject to one prediction and one truth interval per pair. The primary minimum IoU is 0.3. Unmatched requests are misses and unmatched predicted windows are false alarms. Detection scores are computed against monitored requests. For classifier dataset construction, the archived matching routine considers all labeled events and then retains monitored matches, preventing a distractor match from becoming a model-label training example. Online Resource 2 contains both the matching code and per-session counts.

Only matched development windows train the detected-window classifier; only matched test windows receive a scored model prediction. This retrospective scoring convention does not supply the attacker with a deployed mechanism to recognize successful matches or reject false alarms. We therefore report detector false

alarms separately and do not call the all-request score a complete open-world attack success rate.

4.2 Features and learning

We use random forests [12] implemented in scikit-learn [13], with 200 trees, maximum depth eight, seed 42, and balanced class weights for classification. Regression uses the same tree count, depth, and seed. Models are trained on development sessions only. Detected and oracle models are fitted separately to their respective window distributions. We report both a full-data oracle and a training-controlled oracle: the latter uses oracle crops for exactly the same 144 development event IDs that train the archived detected-window model. This removes training count, labels, and event identity as explanations for that comparison, while deliberately retaining the test-time coverage difference.

Four feature sets are evaluated. *Time* contains interval duration and sample count. *Amplitude* contains 13 power-distribution and baseline-relative statistics, excluding duration, energy, and transition counts. *Power + time* contains the original 25 power, energy, duration, sample-count, and transition/plateau statistics. *All passive* uses the archived 32-column representation, adding utilization, clock, and temperature summaries; its two optional probe columns are zero for these sessions. The full feature names and extraction code are supplied in Online Resources 1 and 2.

Baselines normally use the preceding 0.5 s. If that interval has no samples, extraction uses full-trace low-power values. Amplitude features are not physically independent of duration: window length changes the mixture of operational phases and the estimates of distribution statistics. The ablation tests the contribution of an explicit feature group, not a causal separation of electrical amplitude and execution time.

There are 144 archived-detector development matches and 60 test matches. Oracle extraction yields 1,043 development and 582 test windows; one request in each split contains no telemetry sample. Such requests remain in the all-request denominator and are counted as misses. The training-controlled oracle uses 144 development crops and all 582 observable test crops. On the 60 common test event IDs, we additionally compare detected and oracle predictions conditionally. Token-count regression is fitted and scored only on generation requests, never prefill-only events.

4.3 Denominators, reference scores, and uncertainty

For C classes, let N_c be the number of monitored requests in class c, M_c the number matched, and K_c the number both matched and correctly classified. We report the class-balanced all-request identification score

$$A_{\text{all}} = \frac{1}{C}\sum_{c=1}^{C}\frac{K_c}{N_c}. \tag{1}$$

This includes misses but does not penalize false-positive windows. Uniformly random labels assigned to the *same detected requests* have expected score

$$A_{\text{random|det}} = \frac{1}{C^2}\sum_{c=1}^{C}\frac{M_c}{N_c}. \tag{2}$$

The familiar binary 0.5 reference applies to conditional classification with both classes represented; it does not apply to an all-request score under sparse coverage. We report the difference $A_{\text{all}} - A_{\text{random|det}}$ using paired resampling. The random-label reference is an expectation under the observed matches, not an independently executed random detector.

Conditional balanced accuracy is also computed within each eligible test session and averaged. Three detected-window sessions lack both classes, leaving seven eligible sessions. Regression R^2 requires at least two generation requests with nonconstant targets, leaving nine detected-window sessions. Oracle metrics have ten eligible sessions. These session averages differ from pooling all matched requests, so both aggregation rules are identified explicitly.

Intervals are 95% percentile cluster-bootstrap intervals from 5,000 resamples of whole test sessions with seed 20260927. For pooled all-request scores, class counts are summed within each resample before computing the ratio. Models remain fixed; intervals describe variation among recorded test sessions and omit uncertainty from retraining, detector selection, device sampling, and post-hoc analysis choice. With only ten heterogeneous sessions, the intervals are descriptive, not strong population guarantees. Multiple exploratory feature comparisons receive no multiplicity correction.

4.4 Detector and overlap sensitivity

Two extra detector families are selected using validation mean F1 at IoU 0.3. Power hysteresis smooths power over approximately 50 ms, sets a full-trace tenth-percentile baseline, and tests rises of 5, 10, 20, or 40 W, with a release threshold at half the rise and merge gaps of 0.1 or 0.5 s. Minimum duration is 0.05 s. The selected setting uses 5 W and a 0.1-s gap.

The second family is the repository's candidate-change heuristic: it proposes boundaries from changes in power, utilization, and clock and from large changes in a combined level-and-edge score, then retains active segments. It tests spacing controls, activity thresholds, and merge gaps in 16 configurations. Although the legacy function name contains "pelt," this execution path does not call the dynamic-programming PELT routine, and we do not describe it as PELT. Selection details are recorded in the audit JSON. The archived and alternative detectors are scored at IoU 0.1, 0.3, and 0.5. For the primary IoU of 0.3, each selected detector gets a separately fitted classifier trained on its own development matches and scored on its test matches; misses retain their place in the 583-request denominator.

4.5 Normalization and minimum-duration controls

The archived normalization can divide a changing channel by 10^{-6} when its lowest-quartile baseline is constant. We retain that implementation for reproduction and add two controls. For each channel, a reference scale is the median of positive $1.4826\,\mathrm{MAD}$ values across development-session baselines: 8.2811 W, 2.2239 utilization percentage points, and 333.585 MHz. The zero-MAD fallback substitutes this scale only when the session baseline MAD is zero. The scale-floor rule takes the larger of the session scale and the development reference. Both retain per-session full-trace centering and thus remain offline procedures.

Fixed controls change only normalization, keeping threshold 2, merge gap 0.1 s, and minimum duration 0.2 s. A separate 36-configuration grid crosses both rules with thresholds 0.75, 1.25, or 2; minimum durations 0.01, 0.05, or 0.2 s; and gaps 0.1 or 0.5 s. Mean validation F1 against monitored requests selects the configuration, with grid order breaking ties. Selection of the two earlier detector families preserves their archived all-event validation convention; all test detection scores use monitored requests. Finally, duration-only controls retain archived normalization and change its minimum duration to 0.01, 0.05, or 0.1 s. Every control retrains the power-and-time classifier on its own development matches. These controls were motivated by inspection of the reused corpus and are post-hoc, not independent validation.

4.6 Analysis provenance

The archived split and primary configuration precede this reanalysis. The new feature ablations, references, and detector sensitivities were designed after inspection of existing results. Their design is recorded in the analysis-plan file, but this is not preregistration or a fresh lockbox. The hardware cadence measurements are new physical runs; the request-level results are computations on previously collected traces. OpenAI Codex assisted with manuscript restructuring, analysis-code preparation, and execution of the revision audit. This assistance is disclosed separately from human authorship and does not establish independent human validation.

5 Results

5.1 Detection limits the evaluated pipeline

At IoU 0.3, the archived detector matches 60 of 583 monitored requests (pooled coverage 10.29%). Mean session recall is 0.103 and F1 is 0.151 (95% interval 0.104–0.202). All 60 matches are generation requests. None of the 117 prefill-only requests is matched. Thus the conditional inference results describe a selected subset, not a representative sample of every request type.

Table 1 shows the complete class accounting. The power-and-time classifier correctly labels 33 Large and 11 Small requests. Its pooled conditional balanced accuracy is 0.688; averaging eligible session scores gives 0.712 (0.569–0.865). The all-request score is only 0.076 (0.046–0.108), because most requests have no matched prediction. Misses are not a third model class.

Table 1 Power-and-time predictions after archived detection, including unmatched requests

True model	Predicted Large	Predicted Small	Miss	Total
Large	33	3	254	290
Small	13	11	269	293

The same-detection random-label reference is 0.052 (0.035–0.069). The paired difference is 0.024 (0.005–0.043). The pipeline therefore retains a positive identification advantage under this descriptive analysis. Comparing 0.076 with 0.5 would incorrectly label the all-request result as below chance. Figure 1 uses the same denominator throughout and separates the controlled and full-data oracle protocols.

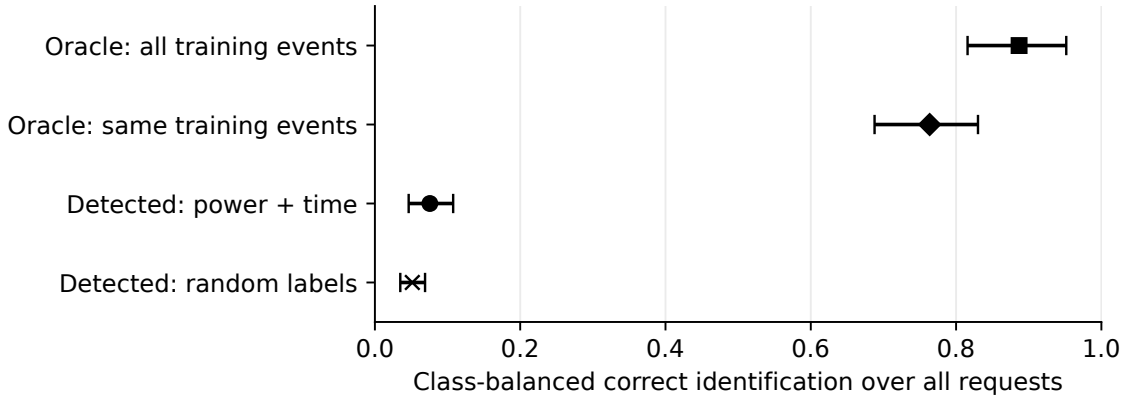


Fig. 1 All-request identification with identical class denominators; horizontal intervals resample test sessions. The controlled oracle and detected models use the same 144 development event IDs; the full oracle uses all observable development events

Mean session false-alarm rate is 488.8 per hour. This is a time-normalized rate from short sessions, not an hour-long observation. The total test exposure is only 18.91 min. A usable attack must handle these windows as well as the correctly matched ones; a high conditional classifier score alone is insufficient evidence of that ability.

5.2 Oracle information and feature ablations

Table 2 compares feature sets with the same all-request denominator. With the same 144 training-event IDs as the detected model, the oracle power-and-time score is 0.764 (0.688–0.830). The full-data oracle reaches 0.887 (0.816–0.952), with conditional session balanced accuracy 0.890 (0.821–0.953). Thus training support explains part, but not most, of the headline gap. Both oracle pipelines are privileged references, not proven mathematical upper bounds on every possible attacker. On only the 60 commonly scored test events, detected and controlled-oracle balanced accuracy is 0.688 and 0.694, respectively; the session-mean paired difference is 0.026 with a 95% interval of -0.046 to 0.115. The large all-request difference is therefore principally coverage in this controlled analysis, rather than better classification of the events already found.

Table 2 All-request identification scores and 95% session-bootstrap intervals; the paired-training oracle uses the same 144 development-event IDs as the detected model

Features	Detected	Oracle, paired train	Oracle, full train
Time (2)	0.060 [0.037, 0.083]	0.742 [0.675, 0.806]	0.839 [0.729, 0.944]
Amplitude (13)	0.072 [0.045, 0.105]	0.750 [0.682, 0.816]	0.794 [0.740, 0.839]
Power + time (25)	0.076 [0.046, 0.108]	0.764 [0.688, 0.830]	0.887 [0.816, 0.952]
All passive (32)	0.072 [0.045, 0.102]	0.769 [0.704, 0.827]	0.880 [0.801, 0.949]

With oracle windows, amplitude-only conditional session balanced accuracy is 0.800 and timing-only accuracy is 0.843; the combination reaches 0.890. With detected windows, the corresponding means are 0.693, 0.591, and 0.712. Figure 2 shows their intervals. These outcomes contradict an interpretation that the observations contain only timing signal. They also do not show that amplitude predicts a model independently of all timing-related workload structure. The feature groups are correlated, and adding passive channels does not consistently improve the classifier.

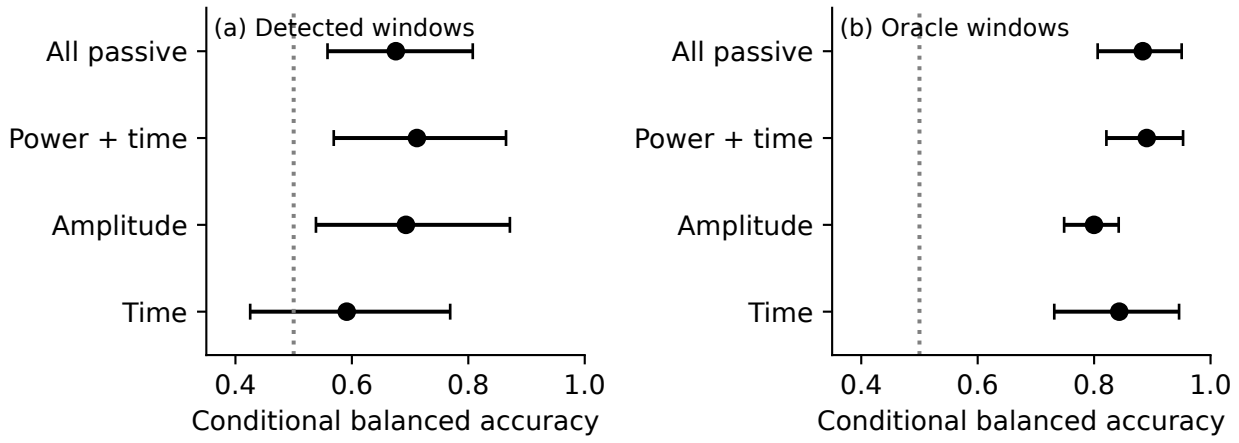


Fig. 2 Conditional classification ablations with 95% session-bootstrap intervals. Detected-window averages use seven eligible sessions; oracle averages use ten. The dotted line is the binary conditional chance reference

For token-count prediction, the power-and-time oracle model has mean session $R^2 = 0.922$ (0.871–0.969). The detected-window model has $R^2 = 0.300$ (-0.006–0.557), conditional on matched generation requests in

nine eligible sessions. Its pooled mean absolute error is 24.85 tokens, compared with 6.58 for oracle windows. Timing-only detected-window regression has mean $R^2 = -0.303$, and amplitude-only has -0.151. Full conditional results appear in Online Resource 1. Missed requests have no token prediction; these regression scores are not all-request leakage measures.

5.3 The result depends on the detector and overlap rule

At IoU 0.3, the selected power-hysteresis detector has mean F1 0.133 and recall 0.100. The candidate-change heuristic reaches F1 0.195 (0.154–0.235) and recall 0.176, exceeding the archived detector's point estimates. Its mean false-alarm rate also rises to 1,280.2 per hour. Table 3 completes the comparison by retraining each power-and-time classifier. Candidate change increases the all-request identification score to 0.127 (0.097–0.159), compared with 0.076 for the archived pipeline. Power hysteresis reaches 0.091 (0.043–0.147). These scores still omit an online mechanism for rejecting false-positive windows.

Table 3 Complete detector/classifier pipelines at IoU 0.3; all-request intervals resample sessions, and false-alarm rates are session means

Detector	Train	Test matches	All-request score	False alarms/h
Archived	144	60	0.076 [0.046, 0.108]	488.8
Power hysteresis	135	59	0.091 [0.043, 0.147]	443.9
Candidate change	140	102	0.127 [0.097, 0.159]	1280.2
Normalized threshold	143	71	0.100 [0.065, 0.138]	565.3

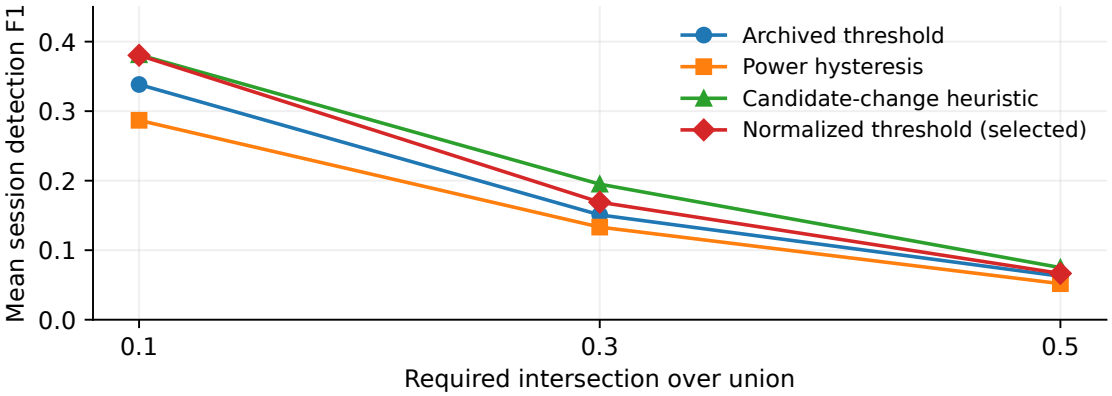


Fig. 3 Detection sensitivity to the overlap requirement. Curves show mean F1 over the ten test sessions; additional detector configurations were selected on validation sessions only

As Fig. 3 shows, reducing the required IoU to 0.1 increases archived F1 to 0.338, while increasing it to 0.5 reduces F1 to 0.063. Candidate-change F1 changes from 0.381 to 0.075 across the same range. Request localization and end-to-end identification are therefore detector-dependent. The study does not establish that these simple detectors exhaust the information in the trace; learned sequence segmentation or a different event definition may perform better.

5.4 Implementation choices account for part of the detection deficit

Five test sessions have zero baseline clock MAD; four subsequently show clock changes. One also has zero baseline utilization MAD. In the idle-gap trace the maximum unsmoothed weighted clock contribution

is 2.475×10^8, compared with 17.89 for power. These scales make the nominal channel weights a poor description of their effective influence.

Validation selects zero-MAD fallback with the original threshold, duration, and gap (F1 0.1732). Its retrained classifier matches 71 test requests and scores 0.100 (0.065–0.138), with 565.3 false alarms per hour. The fixed scale-floor control matches 81 and scores 0.113 (0.076–0.151), with 431.9 false alarms per hour. Its better test point estimate does not make it the validation-selected configuration. These are descriptive differences, not a claim of statistically established superiority. Normalization matters, but neither control eliminates the oracle gap.

The duration rule has a separate mechanical effect. For truth duration d and predicted duration at least m, IoU $\leq d/m$. At $m = 0.2$ s and IoU 0.3, all 110 prefill requests shorter than 60 ms are necessarily unmatchable. The other seven prefill requests also receive no match. Reducing the minimum to 10 ms still finds no prefill requests; it matches 65 generation requests, scores 0.086 (0.051–0.125), and raises false alarms to 1,320.7 per hour. Thus the duration cutoff guarantees many misses, but removing it alone does not solve localization. Online Resource 1 reports every control and duration stratum.

6 Discussion and limitations

6.1 What the comparison establishes

For this dataset and these learning procedures, oracle windows enable much higher all-request identification than the evaluated passive segmentation pipelines. The result remains large when the detected and oracle models use identical training-event identities. On the requests already located by the archived detector, however, oracle cropping changes conditional accuracy little. This combination is direct evidence that coverage, rather than only inference quality or training count, drives the controlled gap. It is not evidence that management telemetry is safe to expose indiscriminately: detectable requests carry signal, and the candidate-change pipeline increases all-request identification.

The approximately 100-ms value-change cadence offers a plausible measurement constraint, particularly for brief prefill operations. It does not establish the cause of every miss. The detector's duration and overlap rules mechanically exclude 110 prefill requests, and zero-MAD normalization can magnify clock or utilization changes. The corresponding controls isolate those implementation changes while retaining other pipeline choices; they do not identify the independent causal effects of telemetry lag, background activity, or merged requests. Shorter minimum windows alone find no prefill requests, and replacing normalization improves point estimates without approaching oracle coverage.

The full-data oracle still changes the number and distribution of training examples, which is why it is reported separately from the controlled oracle. The controlled comparison fixes event identity and count but necessarily changes the feature values produced by the two window types; that is the boundary treatment being tested. It does not simulate a causal online oracle or isolate every source of localization error. A future experiment could train a sequence model directly on continuous labeled traces and evaluate it on a fresh multi-device corpus. We do not infer a universal bound on leakage from the present comparison.

6.2 External validity and statistical limits

Only one consumer GPU, one recorded driver version, two model sizes from one family, and five fixed prompts are studied. Modern serving systems may use other devices, batching, quantization, concurrent models, or restricted telemetry permissions. The study includes configured load changes but not a multi-tenant

production deployment. Both models are resident, and the collector controls the schedule. Results should not be extrapolated to all language-model services.

The ten test sessions provide little within-condition replication. Full-trace normalization, retrospective matching, and the omission of a learned false-alarm rejection stage limit deployment claims. Bootstrap intervals do not account for the history of data reuse. The strongest contribution is the transparent accounting and reproducible within-dataset comparison, not a narrowly estimated population effect. A genuinely new multi-device corpus with a preregistered detector and repeated sessions would be needed for stronger generalization claims.

6.3 Implications for evaluation and mitigation

An evaluation of telemetry leakage should state the observer's permissions, distinguish polling from value-change cadence, describe how windows are obtained, and report class-specific coverage alongside conditional inference. For all-request metrics, the reference score must use the same detection coverage. False alarms require an explicit operating interpretation rather than being hidden by scoring only successful matches.

Restricting telemetry access, reducing precision, or altering workload timing may change leakage, but this audit does not establish an optimal mitigation. Previously explored sanitization policies were trace replays, and previously explored tail-smearing workloads had limited conditions and sparse matches. They do not justify a deployed-defense claim here. Online Resource 1 preserves these caveats rather than treating auxiliary experiments as confirmation of a general security conclusion.

7 Conclusion

On the tested RTX 2080 Ti, evaluated passive detectors miss most monitored requests, while successfully located windows retain metadata signal. With identical training-event identities, all-request class-balanced identification is 0.076 after archived detection and 0.764 with oracle boundaries; the full-data oracle reaches 0.887. A retrained candidate-change pipeline improves the detected score to 0.127 at the cost of more false alarms. Normalization controls also improve point estimates, while shorter duration cutoffs alone fail to recover prefill requests. Both amplitude and timing contribute to inference. Low observed value-change cadence and weak attack performance are therefore insufficient grounds for claiming absence of leakage. Request localization, training supervision, inference, and false alarms must be evaluated together.

Statements and Declarations

Funding. No funding was received for this work.

Competing interests. The authors declare no known competing financial interests or personal relationships that could have appeared to influence this work.

Author contributions. Ali Hamza: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, original draft, visualization. Chih-Yi Chiu: supervision, methodology, review and editing. Aziz Ahmad: methodology, software, validation, review and editing. Shehzad: software, visualization, investigation. Hassan Iftikhar: formal analysis, review and editing.

Data and code availability. The selected continuous traces, ground truth, split manifest, audit outputs, cadence records, and analysis scripts are assembled in Online Resource 2. No public repository or permanent identifier is claimed.

Ethics. Experiments use the authors' hardware and workloads, as stated in the original study. No third-party account or system is targeted.

Supplementary material. Online Resource 1: detailed results, feature inventory, and evidence-scope audit (`ESM_1.pdf`). Online Resource 2: local reproducibility archive of primary data, analysis code, and machine-readable outputs (`ESM_2.zip`).

References

[1] NVIDIA (2026) NVML API Reference Guide: Device Queries. https://docs.nvidia.com/deploy/nvml-api/api/group__nvmlDeviceQueries.html. Accessed 27 September 2026

[2] Naghibijouybari H, Neupane A, Qian Z, Abu-Ghazaleh N (2018) Rendered Insecure: GPU Side Channel Attacks are Practical. In: ACM SIGSAC Conference on Computer and Communications Security, pp 2139–2153. https://www.cs.ucr.edu/~nael/pubs/ccs18.pdf

[3] Luo C, Fei Y, Zhang L, Ding AA, Luo P, Mukherjee S, Kaeli D (2018) Power Analysis Attack of an AES GPU Implementation. Journal of Hardware and Systems Security 2:69–82. https://doi.org/10.1007/s41635-018-0032-7

[4] Maia HT, Xiao C, Li D, Grinspun E, Zheng C (2022) Can One Hear the Shape of a Neural Network? Snooping the GPU via Magnetic Side Channel. In: 31st USENIX Security Symposium. https://www.usenix.org/conference/usenixsecurity22/presentation/maia

[5] Gao Y, Qiu H, Zhang Z, Wang B, Ma H, Abuadbba A, Xue M, Fu A, Nepal S (2024) DeepTheft: Stealing DNN Model Architectures through Power Side Channel. In: IEEE Symposium on Security and Privacy, pp 3311–3326. https://doi.org/10.1109/SP54263.2024.00250

[6] Chaudhuri A, Shukla S, Bhattacharya S, Mukhopadhyay D (2025) Energon: Unveiling Transformers from GPU Power and Thermal Side-Channels. In: IEEE/ACM International Conference on Computer-Aided Design. Accepted manuscript: https://doi.org/10.48550/arXiv.2508.01768

[7] Horváth P, Lauret D, Liu Z, Batina L (2024) SoK: Neural Network Extraction Through Physical Side Channels. In: 33rd USENIX Security Symposium, pp 3403–3422. https://www.usenix.org/conference/usenixsecurity24/presentation/horvath

[8] Yang Z, Adámek K, Armour W (2024) Accurate and Convenient Energy Measurements for GPUs: A Detailed Study of NVIDIA GPU's Built-In Power Sensor. In: SC24: International Conference for High Performance Computing, Networking, Storage and Analysis. https://doi.org/10.1109/SC41406.2024.00028

[9] Bepary MK, Zhang T, Zhou J, Rahman F, Farahmandi F, Tehranipoor M (2025) Towards Efficient Gate-Level Electromagnetic Side-Channel Leakage Modeling and Vulnerability Assessment. Journal of Hardware and Systems Security 9:24–37. https://doi.org/10.1007/s41635-025-00161-1

[10] Prabhakara Rao S, Balaji SRA, Rahman F et al (2025) Secure and Trustworthy Microelectronics: Vulnerabilities, Solutions, and Trends. Journal of Hardware and Systems Security 9:163–189. https://doi.org/10.1007/s41635-025-00165-x

[11] Baddour I, Banerjee DS, Sanadhya SK (2026) SideLink: Exposing NVLink to Covert- and Side-Channel Attacks. Journal of Hardware and Systems Security 10:8. https://doi.org/10.1007/s41635-026-00175-3

[12] Breiman L (2001) Random Forests. Machine Learning 45:5–32. https://doi.org/10.1023/A:1010933404324

[13] Pedregosa F et al (2011) Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12:2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- ESM1.pdf
- ESM2.zip