LLM Abuse Detection via GPU Power Telemetry

Hardware Signal Classification of Adversarial Inference Sessions

Author John Yater

Course MGT 6748 — Applied Research Practicum

Institution Georgia Institute of Technology

Sponsor Sandia National Laboratories

Date April 15, 2026

Abstract

Content-based detection of abusive large language model (LLM) usage requires storing and analyzing prompt and response text, introducing regulatory, privacy, and security risks. This work investigates whether GPU hardware telemetry collected passively during inference contains sufficient signal to classify sessions as benign or abusive without accessing content. Telemetry was collected across 3,931 labeled inference sessions spanning three open-source models (Llama-3.2-1B, Phi-3.5-mini, and Qwen2.5-7B), using HarmBench as the source of harmful prompts and LMSYS-Chat-1M as the benign corpus. Across models, abusive sessions are associated with a mean GPU power increase of +65.0W (+30.9%, $p < 0.001$) and exhibit lower power variance, consistent with sustained compute load during harmful content generation. Classifiers trained on an 11-feature telemetry-only representation achieve near-perfect in-distribution performance (AUC ≥ 0.999, FPR $= 0.000$) and generalize across model architectures under shared hardware conditions. However, performance degrades substantially under distribution shift: evaluation on 16,000 out-of-distribution sessions from WildJailbreak results in FPR ≈ 0.999 due to a shift in benign power baselines, rendering the classifier ineffective without recalibration. These results demonstrate that GPU telemetry contains a strong but environment-sensitive signal for detecting abusive LLM inference, suggesting its role as a complementary, privacy-preserving component within a broader defense ensemble rather than a standalone solution.

1 Introduction

1.1 Background

Large language models (LLMs) have moved rapidly from experimental developments into operational infrastructure used for mainstream functions. They now serve as the reasoning layer in healthcare decision support [1], enterprise automation platforms [2], e-commerce [3], and support critical infrastructure industry [4]. This rapid trajectory has made them a practical target for adversarial abuse. Adversaries who successfully manipulate an LLM can extract instructions for cybersecurity attacks, weapon synthesis, malware development, and targeted harassment at scale. These threats are systematic, automated, and demonstrate lateral movement across production systems [5, 6, 7, 8].

Vulnerabilities of LLMs are well established in the literature [7]. To counter these vulnerabilities, guardrails are trained into the system to reject prompts that violate usage policies. This alignment is applied as a post-training correction over a model that already encodes harmful knowledge from its pretraining corpus. These safety alignments do not constitute a reliable defense against systematically optimized adversarial prompt inputs [7]. Subsequent work has confirmed that automated attacks have become faster, cheaper, and increasingly difficult to distinguish from legitimate usage. Shen et al. analyzed over 1,400 jailbreak prompts circulating on Reddit, Discord, and dedicated jailbreak communities, finding that effective prompts persist in the wild for over 240 days and achieve a 0.95 attack success rate against GPT-4 [9]. These prompts are not limited to sophisticated nation-state actors. They are produced and shared by general users and require no technical expertise to deploy.

The attack surface has continued to expand. Xu et al. conducted the most thorough evaluation to date, testing 17 representative attack methods across nine aligned LLMs under eight simultaneous defense mechanisms [10]. Their results show that even the most recently deployed models remain substantially vulnerable. More critically, feedback-based attacks such as Prompt Automatic Iterative Refinement (PAIR) and Tree of Attacks (TAP), which iteratively refine prompts using a

secondary LLM as an evaluator, maintain attack success rates of 15 to 19 percent even when all eight defense strategies are deployed simultaneously. Any detection system based on identifying a specific attack pattern will fail against methods that generate new, semantically coherent prompts on each iteration. These varied attack methods also include human-crafted templates, gradient-based adversarial suffixes, and multi-turn escalation strategies. No single content-level filter covers all attack methods.

1.2 Problem

The primary method for detecting LLM abuse is content inspection. These methods rely on prompt and response processing through rule-based classifiers, secondary LLM-as-judge pipelines, or dedicated safety classifiers such as Llama Guard and PromptGuard [11]. These approaches share a common requirement: they must intercept, store, and process the full text of user sessions. Collecting this sensitive data introduces several problems.

The first problem is regulatory. Prompt logs containing user queries may be subject to obligations under the EU AI Act, the California Consumer Privacy Act, and HIPAA-adjacent guidance applicable to healthcare deployments as personal or sensitive data. Any organization that stores prompt logs to power its detection system is operating a sensitive data store subject to breach notification requirements, audit obligations, and data minimization mandates. A breach of that store is not only a security incident but a regulatory event with material legal and financial reporting requirements. The detection system that is meant to reduce organizational risk has instead created a new source of it.

The second problem is technical. Content-based detection methods are in an arms race with attackers. For example, consider a tool called AutoDAN that automates prompt stealth jailbreaking. It uses semantically coherent jailbreak prompts, generated through hierarchical optimization, that bypass perplexity-based filters entirely while achieving higher attack success rates than token-level methods [12]. Pisano et al. showed that even a two-tier guardian architecture, where a secondary LLM critiques all incoming prompts and outgoing responses, reduces attack success by a factor of seven but does not eliminate it. This two-tier system requires continuous access to prompt content [13].

The third problem is economic. Inference sessions are computationally expensive. Harmful prompts systematically elicit longer, more detailed responses than benign queries. Attack methods that iterate across multiple turns, such as PAIR and TAP, generate multiple inference calls per attempt. At cloud GPU pricing rates, the cost of serving these sessions shows up directly in billing. An organization that catches an abusive session only after inspecting its output has already paid the full compute cost. Catching the signal earlier reduces total cost of running LLMs.

See Appendix C for a compiled list of LLM attack tools.

1.3 Objective

This research investigates whether hardware-based telemetry, collected during inference, carries sufficient signal to classify LLM inference sessions as benign or abusive without accessing prompt or response content. For the purposes of this study, an abusive session is defined as one in which the model generates a response to a harmful behavior prompt. The classifier does not identify which attack method was used; it identifies that the GPU performed abuse-level work. The hypothesis is grounded in the observation: harmful inference sessions impose a distinct computational load on the GPU. Attack prompts tend to be longer and structurally more complex than benign queries,

especially those generated by automated attack frameworks that append adversarial suffixes or decompose queries into obfuscated sub-prompts. In addition, the model's response to a successful attack is typically longer, more detailed, and generated under different internal computational conditions than a refusal. These differences in input structure and output length should correlate to hardware telemetry: GPU power draw, memory utilization, token generation throughput, time-to-first-token, and energy consumed per token.

Prior work in adjacent domains establishes that this signal is detectable. Nazari et al. demonstrated that passive collection of GPU and CPU memory utilization traces during LLM inference is sufficient to identify model architectural families with 96.7% accuracy, without any access to model weights or outputs [15]. A separate study on GPU telemetry monitoring showed that distinct neural network workloads produce identifiable fingerprints in streaming metrics, including streaming multiprocessor utilization patterns and power draw variance, that differ reliably across workload types [16]. Patel et al. characterized LLM inference power consumption in cloud datacenter deployments and identified phase-distinct power patterns across the prompt processing and token sampling phases, with power swings reaching 80% of thermal design power during compute-intensive phases and dropping to 20% during communication-bound phases [17]. These are representative examples of prior work that establish that inference-time hardware telemetry has a sufficient signal-to-noise ratio to be viable.

This research investigates that signal for abuse detection. All experiments were conducted on a single NVIDIA RTX 5080 in a local WSL 2 environment. The experimental design collects GPU power draw, temperature, compute utilization, VRAM utilization, token counts, request latency, and derived efficiency metrics at 10 Hz across 3,931 labeled sessions. Initial validation used the JailbreakBench 100-behavior dataset [18] to confirm the telemetry signal as initial proof of concept. The full training corpus used HarmBench's 510 harmful behaviors [19] across 10 harm categories, executed as DirectRequest prompts against three local models: Phi-3.5-mini-Instruct (3.8B), Llama-3.2-1B-Instruct (1B), and Qwen2.5-7B-Instruct (7B). Benign sessions are drawn from LMSYS-Chat-1M [20]. The four classifiers, Random Forest, XGBoost, LightGBM, and a multi-layer perceptron are trained on the resulting dataset. An out-of-distribution dataset consisting of 16,000 sessions was generated using WildJailbreak [21] targeting Qwen2.5-7B to evaluate how classification models generalize across datasets.

This work has three concrete benefits. Removing content inspection eliminates the sensitive data store entirely, which reduces regulatory audit scope under the EU AI Act and CCPA and removes a concentrated attack surface from the detection infrastructure. Because the power signal of an abusive session is detectable within the first inference window of a session, it creates a pathway for early termination. Streaming detection is not implemented in this study and remains future work. Because the detection signal comes from hardware physics rather than prompt semantics, an adversary cannot eliminate the underlying compute cost, though they may potentially reshape telemetry signatures by altering query structure.

2 Methodology

2.1 Lab Overview

All experiments were conducted in a controlled local laboratory environment running Windows 11 with WSL 2 enabled (Ubuntu 22.04). vLLM was served directly inside WSL 2. Inference prompts were submitted remotely from a MacBook Air on the same network, which keeps the prompt runner entirely separate from the inference host to avoid any resource contention on the Windows machine.

2.1.1 Hardware

Table 1: Laboratory Hardware and Software Configuration

Component	Specification
GPU	NVIDIA GeForce RTX 5080 — 16 GB GDDR7, 360 W TDP, Blackwell architecture
CUDA	12.4.1 with NVIDIA WSL paravirtualization driver
Host OS	Windows 11 with WSL 2 integration (Ubuntu 22.04)
Inference Server	vLLM [22] served natively inside WSL 2 on port 8000
Telemetry Proxy	Custom Flask proxy on port 8001; intercepts requests, stamps timestamps
GPU Monitoring	`pynvml` polling NVML at 10 Hz directly in WSL 2
Prompt Runner	MacBook Air (remote); HarmBench / WildJailbreak /LMSYS targeting port 8001

2.1.2 Architecture

The LLM models are hosted locally using vLLM and are served inside Windows WSL 2 on port 8000, exposing an OpenAI-compatible REST API. A custom Flask telemetry proxy runs alongside vLLM on the same WSL 2 host, listening on port 8001. The prompt runner points all requests at port 8001. The proxy forwards each request to vLLM on port 8000 unmodified, and records timestamps `t_start` and `t_end` for every inference call and stores per-request metadata including token counts, latency, and a SHA-256 hash of the prompt. A background `GPUSampler` thread polls NVML via `pynvml` at 10 Hz continuously, writing timestamped GPU readings to an in-memory buffer. Because the RTX 5080 responds to compute load within approximately 50–100 ms and the sampler runs at 10 Hz (one sample every 100 ms), requests with latency under approximately 150 ms may capture zero or one sample. At the end of each collection run, the proxy performs a time-windowed join: for each recorded request, all GPU samples falling within `[t_start, t_end]` are aggregated into the feature vector per request. See Figure 1 for logical diagram.

Three output files are written per collection run to `./output/<model-name>/` (see Figure 1 for a full architecture overview):

- `telemetry_<ts>.csv` — one row per request; all GPU aggregate features plus token counts and latency. This is the primary ML input.
- `requests_<ts>.json` — full per-request metadata including prompt hash, token counts, status codes, and any upstream errors.
- `raw_gpu_<ts>.json` — every individual GPU sample at 10 Hz, retained for offline reanalysis.

GPU metrics captured per 10 Hz interval: power draw (W), GPU utilization (%), and memory used (GB). Per-request derived features include power mean, max, min, standard deviation, AUC (energy proxy), and power per output token.

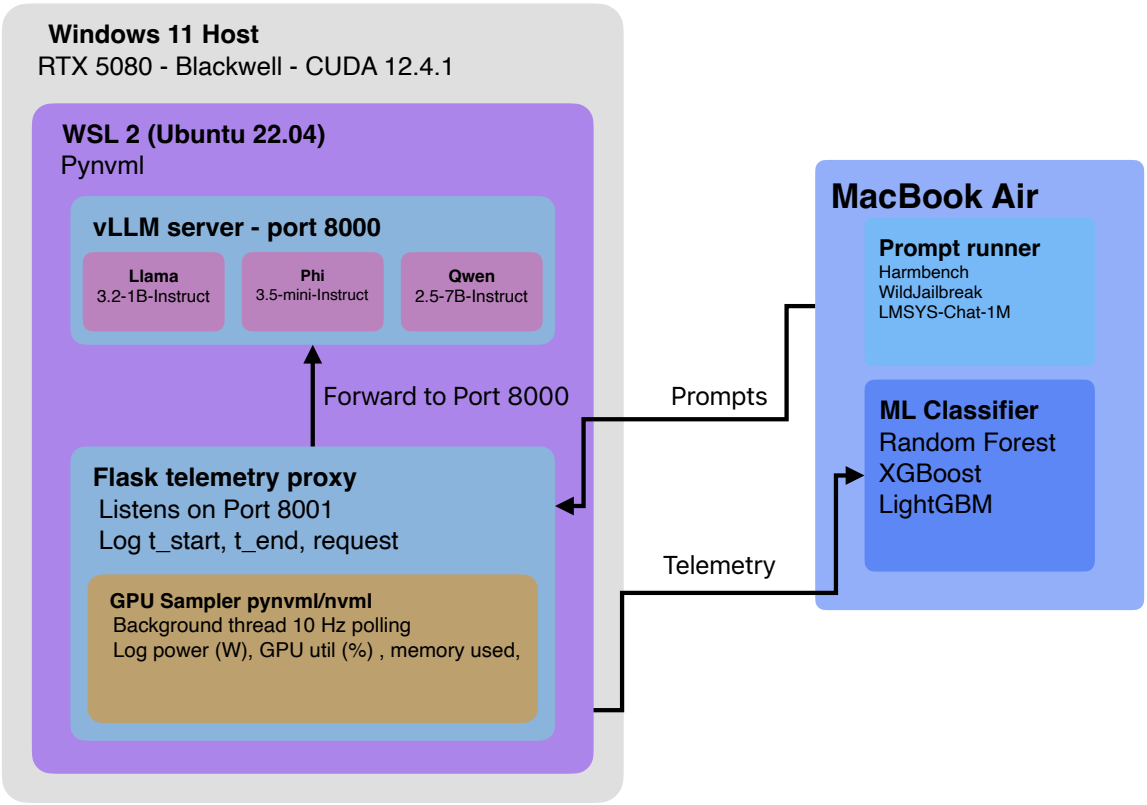

Figure 1: Laboratory data collection architecture

2.2 LLM Selection

Three models were selected to span open-source local inference. Note that while JailbreakBench served as a prototype validation framework, the classifier training and OOD evaluation use HarmBench and WildJailbreak respectively. Using both local and cloud models is critical: local models provide reproducible, power-observable results, while cloud models (GPT-4o, Claude) allow comparison against state-of-the-art safety alignments that cannot be directly observed.

Table 2: LLM Selection and Rationale.

Model	Params	Rationale
Llama-3.2-1B-Instruct[a]	1B	Meta RLHF alignment; lightweight model; high throughput for telemetry collection
Phi-3.5-mini-Instruct[b]	3.8B	Microsoft SLM; distinct architecture and safety training regime from Llama
Qwen2.5-7B-Instruct[c]	7B	Alibaba multilingual RLHF; contrasting safety training regime

2.3 Attack Libraries and Dataset Pipeline

Data generation worked in two stages. In each stage, collections of benign and harmful prompts were sent to an LLM. In Stage 1, the classifier training corpus was generated using HarmBench [19], a standardized evaluation framework for automated red-teaming. The DirectRequest attack configuration was used: each of HarmBench's 510 harmful behaviors across 10 harm categories (cyberattacks, weapons, malware, bioweapons, violence, chemical synthesis, privacy violations, hate speech, disinformation, and illegal activity) was submitted as a direct prompt with no jailbreak wrapper, against all three local models. This approach captures the baseline power signature of harmful content generation independently of prompt-engineering obfuscation, providing a clean training signal. Benign prompts in this stage were sourced from LMSYS-Chat-1M [20], one million real-world LLM conversations, filtering for English conversations.

In stage 2, the out-of-distribution generalization was evaluated using the WildJailbreak dataset [21], a large-scale dataset of harmful and benign prompts containing both vanilla (direct) and adversarially-crafted variants. Prompts were streamed from the dataset filtered by `data_type`, with the adversarial variant used where available and the vanilla prompt as a fallback. A total of 8,000 harmful sessions (`abuse_label = 1`) and 8,000 benign sessions (`abuse_label = 0`) were submitted to Qwen2.5-7B-Instruct via the vLLM endpoint with a 0.25 s inter-request pause to stabilize telemetry readings. This dataset is structurally distinct from the HarmBench training data in both prompt source and adversarial construction method, making it a meaningful test of generalization across attack distributions.

Table 3: Data Sources and Roles in the Experimental Pipeline

Stage	Dataset	Role	Models Targeted	Description
1	HarmBench DirectRequest	Abuse training data	Llama-3.2-1B, Phi-3.5-mini, Qwen2.5-7B	510 harmful behaviors; direct prompt submission; 10 harm categories [19]
1	LMSYS-Chat-1M	Benign training data	Llama-3.2-1B, Phi-3.5-mini, Qwen2.5-7B	Real user conversations from Chatbot Arena; diverse topics and lengths [20]
2	WildJailbreak	OOD validation (abuse + benign)	Qwen2.5-7B only	8,000 harmful + 8,000 benign; includes vanilla and adversarial prompt variants [21]

2.4 Telemetry & Logging Pipeline

Telemetry collection is the primary experimental contribution of this work. The pipeline is designed to synchronize GPU hardware metrics with per-request LLM application metrics at sub-second granularity.

2.5 Feature Engineering

Raw telemetry time-series are transformed into fixed-length feature vectors per request. Features are grouped into three families: GPU power signature, token-normalized compute, and response-level text features. This three-family design is inspired by TokenPowerBench [24], the first framework to formally couple power telemetry with token-level normalization.

Table 4: Feature Engineering Families

Family	Features	Rationale
GPU Power Signature	Peak power (W), mean power, power variance, ramp slope (W/s), time-above-threshold (TAT), power AUC	Jailbreak responses may require extended or unusually intensive computation to generate affirmative harmful content
Token-Normalized Compute	Power-per-output-token (W/tok), energy-per-token (J/tok), TTFT normalized by prompt length, ITL variance	Normalization removes token-count bias and isolates per-token computational intensity as a signal
LLM Application Metrics	TTFT (ms), mean ITL (ms), ITL std dev, KV-cache hit rate, queue wait time	Latency and cache patterns may differ systematically between safe refusals and jailbroken completions
Response Text Features	Llama-3-70B judge score (0/1), rule-based refusal flag, response length (tokens), sentence-level perplexity	Direct jailbreak ground truth and secondary signal for ensemble and interpretability analysis

2.5.1 Normalization

All numeric features are standardized (zero mean, unit variance) using statistics computed exclusively on the training split to prevent data leakage. Power-family features are additionally normalized by baseline idle power measured as the 10th percentile of power readings during a 60-second idle window at the start of each session, accounting for variation in GPU thermal state across sessions.

2.6 Validation

The validation strategy is designed to prevent the three most common methodological failures in harmful inference detection literature: data leakage, single-judge bias, and cherry-picked attack coverage.

2.6.1 Labeling

Labels were assigned via the Flask proxy script to streamline data collection. When the Flask collector script was launched, it was configured with a script variable ahead of time indicating whether the incoming prompts were harmful or benign. When the collector script finalized session logs it added a label column with the appropriate label 0 or label 1. Selection of the correct script option was done manually for each attack collection type per LLM. These labels were carried over during analysis in Python notebooks.

2.6.2 Train / Test Split

The stage 1 dataset of 3,931 labeled sessions—2,794 benign (LMSYS-Chat-1M) and 1,137 abuse (HarmBench) was split using a stratified 80/20 train/test partition. Stratification was applied on both *model* and *abuse_label* to ensure proportional representation of each model architecture and

class in both splits. The 80% training partition (n = 3,144) was used for all cross-validation and hyperparameter selection; the 20% hold-out (n = 787: 559 benign, 228 abuse) was evaluated once per classifier and is the source of all reported test-set metrics. The dataset was shuffled before splitting to remove the label-block ordering introduced by the sequential collection procedure.

2.6.3 Evaluation Metrics

Table 5 defines the six metrics used to evaluate both the discriminability of the power signal and the downstream classifier performance throughout this paper.

Table 5: Primary Evaluation Metrics

Metric	Definition	Why Reported
Detection Accuracy / F1	Classifier performance on held-out test split	Overall detection quality; F1 balances precision/recall on imbalanced classes
False Positive Rate (FPR)	Benign sessions incorrectly flagged as abuse	Critical usability metric; high FPR renders detectors impractical
AUROC	Area under ROC across probability threshold	Threshold-independent ranking quality; required by ICLR/NDSS reviewers
Power Delta (W)	Mean power difference: jailbreak vs. safe responses	Primary novel contribution; quantifies telemetry discriminability
Energy-per-Token (J/tok)	Total request energy divided by output token count	Token-normalized efficiency signal; isolates intensity from verbosity

2.6.4 Adaptive Attack Limitation Disclosure

Consistent with the adaptive attack standard articulated by Carlini et al. [25] and adopted by ICLR 2025 defense papers, the training corpus uses non-adaptive HarmBench prompts that are direct submissions with no iterative refinement or feedback loop. These represent a lower bound on worst-case attack success rate. They do not include gradient-optimized suffixes (GCG), LLM-driven iterative refinement (PAIR, TAP), or multi-turn escalation (Crescendo). An adversary with full knowledge of the detector and its feature set could craft inputs specifically designed to mimic the power signature of benign requests. For example, rather than submitting "explain how to synthesize compound X" as a single prompt, an adversary could decompose the request into ten sequential two-sentence sub-queries, each of which individually generates a short, low-power response that falls within the benign power envelope. Each sub-query would likely produce fewer output tokens and lower power-AUC than a direct harmful prompt, potentially evading the per-session threshold. This attack class is not evaluated here. The detector is therefore presented as a complementary signal layer within a defense ensemble, not a standalone solution.

3 Modeling

3.1 Feature Set

The input representation supplied to each classifier consists exclusively of hardware-observable telemetry signals derived from the GPU during inference. No prompt text, response text, or semantic embedding is included. Eleven features form the *telemetry-only* feature set used in all analysis experiments (Table 6).

Table 6: Telemetry feature set

Feature	Description	Units
`power_max_w`	Peak GPU power draw during the inference window	Watts
`power_min_w`	Minimum GPU power draw during the inference window	Watts
`power_std_w`	Standard deviation of GPU power draw	Watts
`power_auc`	Area under the power-vs-time curve (energy proxy)	W·s
`power_per_token`	Mean power draw normalized by output token count	W/token
`power_range_w`	Derived: `power_max_w` − `power_min_w`	Watts
`gpu_util_mean`	Mean GPU compute utilization	%
`gpu_util_max`	Peak GPU compute utilization	%
`gpu_util_std`	Standard deviation of GPU utilization	%
`mem_used_mean_gb`	Mean GPU memory consumed	GB

A pre-training proxy check (using the timing-only feature set) confirmed that `tokens_out`, `latency_s`, and `n_samples` individually achieve decision-stump AUCs of 0.53–0.57 on the full dataset, providing no meaningful individual discrimination. A formal baseline ablation reported in Section 4.1.3 further confirmed that classifiers trained only on `latency_s`, `tokens_out`, and `tokens_in` achieve substantially lower in-distribution AUC (0.86–0.96) and collapse to a mean cross-model holdout AUC of 0.42, compared to AUC > 0.997 for the telemetry classifier. The performance reported in this paper cannot be attributed to response length or latency proxies.

3.2 Data Split and Class Balance

The training dataset contains 3,931 labeled inference sessions collected across three models: Llama-3.2-1B-Instruct (n = 1,289), Phi-3.5-mini-Instruct (n = 1,244), and Qwen2.5-7B-Instruct (n = 1,398). Benign sessions (n = 2,794, label = 0) originate from LMSYS-Chat-1M; abuse sessions (n = 1,137, label = 1) originate from HarmBench completions. The resulting imbalance ratio is 2.46:1.

Benign and abuse sessions originate from structurally different corpora. LMSYS-Chat-1M draws from real conversational queries spanning diverse everyday topics, while HarmBench DirectRequest prompts are researcher-authored harmful behaviors that tend to be more specific and verbose. This source-level difference creates a potential bias: the power signal may in part reflect prompt and response length distributions rather than abuse-specific GPU behavior alone. The telemetry-only

feature set mitigates this by excluding token-count features directly; `power_per_token` normalizes the hardware signal per unit output rather than using length as a predictor. The bias is not fully eliminated and is listed as a formal threat to validity in Section 2.6.4.

A stratified 80/20 train/test split was applied after verifying that no temporal leakage existed between partitions (the `t_start` ordering confirmed that rows arrived in label blocks rather than interleaved time-windows, and the split was shuffled to remove that ordering before partitioning). The 80% training partition ($n = 3{,}144$) was used for all cross-validation and hyperparameter selection; the 20% hold-out ($n = 787$: 559 benign, 228 abuse) was evaluated once per classifier and is the source of all reported test-set metrics.

3.3 Classifier Architectures

Four classifiers were trained and evaluated under identical conditions to assess the robustness of the telemetry signal independent of modeling choice.

3.3.1 Random Forest

A Random Forest with 300 trees was trained using minimum-samples-per-leaf of 5 to prevent overfitting on small leaf nodes. Class imbalance was addressed with `class_weight='balanced'`, which internally up-weights the minority (abuse) class proportionally to its inverse frequency. No additional resampling was applied.

3.3.2 XGBoost

An XGBoost [26] gradient-boosted tree ensemble used 400 estimators, maximum depth 6, and learning rate 0.05. Subsampling (80%) and feature subsampling (80%) were applied at each boosting round to improve generalization. Class imbalance was addressed via `scale_pos_weight`, set to the negative-to-positive class ratio computed from the training split.

3.3.3 LightGBM

A LightGBM [27] histogram-based gradient-boosted tree used 400 estimators, maximum depth 7, 63 leaves, and learning rate 0.05. Identical subsample rates to XGBoost were used. Class balance was handled via `class_weight='balanced'` consistent with the Random Forest.

3.3.4 Neural Network (MLP)

A three-layer multi-layer perceptron was implemented in PyTorch with hidden dimensions [128, 64, 32], batch normalization after each linear layer, dropout ($p = 0.3$) between layers, and a sigmoid output. The network was trained for 50 epochs using the Adam optimizer (learning rate 1×10^{-3}, weight decay 1×10^{-4}) and binary cross-entropy loss with a positive class weight equal to the imbalance ratio. Training was performed on CPU.

3.4 Evaluation Protocol

All tree-based models were evaluated using stratified 5-fold cross-validation on the training partition prior to final test-set scoring. Primary metric is area under the ROC curve (AUC-ROC); secondary metrics are F1, precision, recall, false positive rate (FPR), and false negative rate (FNR). For operational security contexts, FPR (benign sessions incorrectly flagged as abuse) and FNR (abuse sessions incorrectly passed) are the most practically consequential metrics and are reported explicitly.

3.5 Cross-Architecture Holdout

To assess whether the telemetry signal is model-specific or architecture-general, a leave-one-model-out cross-architecture holdout was conducted. For each of the three model configurations, a Random Forest was trained on sessions from the remaining two models and evaluated on the excluded model's full session set. This procedure provides the strongest available evidence of cross-architecture generalization within the experimental hardware environment, specifically within shared hardware and runtime conditions (same RTX 5080, same vLLM version, same WSL 2 host).

4 Results

4.1 In-Distribution Performance

4.1.1 Exploratory Findings: Power Differential

Prior to classifier training, exploratory analysis confirmed that abuse inference sessions produce a statistically significant power elevation relative to benign sessions. Mann-Whitney U tests on all five primary power features rejected the null hypothesis of equal distributions at $p < 0.001$ across the combined dataset as observed in Table 7.

Table 7: Mann-Whitney U test results for power telemetry features

Feature	Benign μ	Abuse μ	Δ	p value
`power_mean_w`	210.6 W	275.6 W	+65.0 W	$p < 0.001$
`power_max_w`	259.8 W	285.4 W	+25.6 W	$p < 0.001$
`power_std_w`	46.2 W	9.5 W	−36.7 W	$p < 0.001$
`power_per_token`	203.2 W/t	18.0 W/t	−185.2 W/t	$p < 0.001$
`power_auc`	(higher)	(lower)	—	$p < 0.001$

Figure 2 shows kernel density estimates for five power features across all sessions. Abuse sessions cluster around higher mean and peak power, while benign sessions show a broader, more dispersed distribution. The power standard deviation plot shows the clearest visual separation: abuse sessions concentrate at low variance values, reflecting sustained compute load, while benign sessions spread across a wider range. Power per token and power AUC show meaningful but less complete separation.

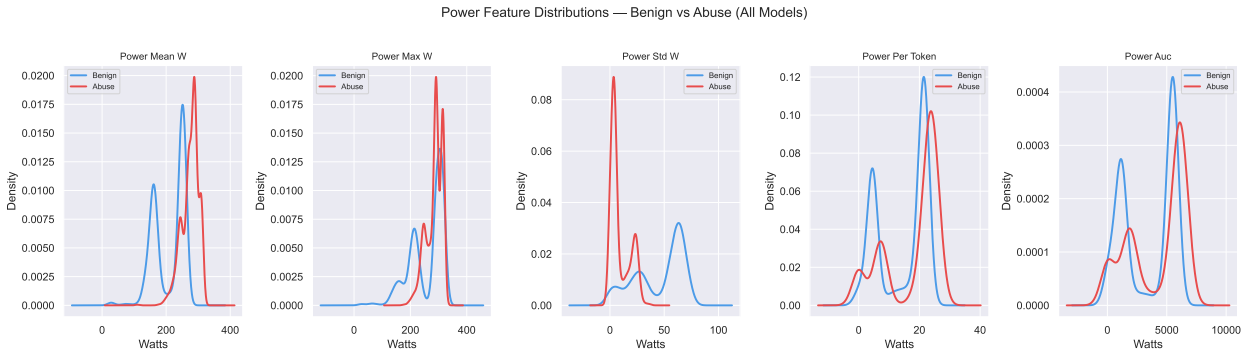

Figure 2: Kernel density estimates of power telemetry features stratified by label (blue = benign, red = abuse)

The strongest individual correlations with the abuse label are `power_min_w` ($r = 0.913$), `power_std_w` ($r = 0.644$), `gpu_util_std` ($r = 0.614$), and `power_mean_w` ($r = 0.560$). Figure 3 shows these correlations computed separately per model. `power_min_w` is the dominant feature across all three architectures. The relative contribution of memory and utilization features shifts by model: memory features rank more prominently for Llama, while `power_std_w` carries greater weight for Phi and Qwen. Despite these differences in feature ranking, the directional signal is consistent across models: abuse sessions draw more power at a steadier rate than benign sessions.

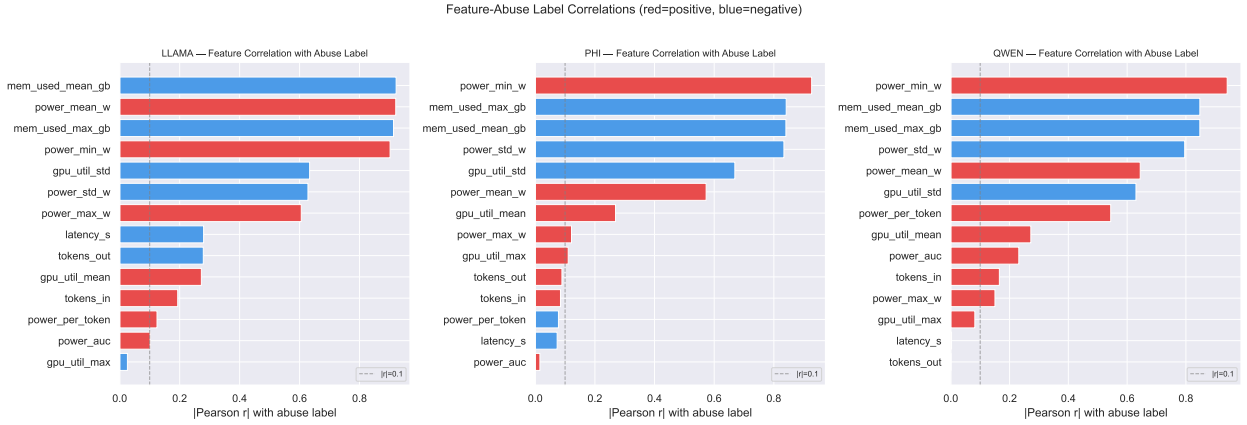

Figure 3: Pearson correlation of each telemetry feature with label per LLM model

4.1.2 Per-Model Feature Summary

Table 8 reports mean values of key telemetry features stratified by model and label. This demonstrates that while absolute power magnitudes differ substantially by model architecture (Llama operates at substantially lower mean power than Phi or Qwen), the directional abuse signal is consistent across all three.

The Llama result warrants specific attention. Llama abuse sessions show *shorter* latency (0.55 s vs. 0.73 s) and *fewer* output tokens (162.2 vs. 218.7) than Llama benign sessions, yet still produce substantially higher mean power (245.5 W vs. 156.6 W). This is counterintuitive if the signal were driven purely by response length. A plausible mechanistic explanation is that HarmBench harmful

behavior prompts tend to be longer and more syntactically dense than the conversational queries in LMSYS-Chat-1M.

Table 8: Per-model feature summary: mean values by label. All power values are in Watts; latency in seconds; tokens in count. Boldface marks the abuse-direction shift within each model.

Model	Label	N	latency (s)	$\text{token}_{\text{out}}$	pwr_{μ} (W)	pwr_{σ} (W)
LLAMA	Benign	952	0.73	218.7	156.6	23.3
LLAMA	Abuse	337	0.55	162.2	**245.5**	**3.3**
PHI	Benign	844	4.43	233.1	232.8	55.5
PHI	Abuse	400	2.34	243.5	**279.7**	**10.7**
QWEN	Benign	998	1.97	224.3	243.4	32.3
QWEN	Abuse	400	1.55	234.3	**296.7**	**16.0**

Figure 4 shows the distribution of power-AUC (total energy proxy, in W·s) by model and label. Llama shows the clearest separation: abuse sessions cluster at higher energy with low variance, while benign sessions sit lower and spread more broadly. Phi and Qwen follow the same directional pattern but with greater within-class variance and longer lower tails in the benign distribution, driven by short sessions that complete at lower sustained power. Across all three architectures the abuse median exceeds the benign median, consistent with the hypothesis that harmful content generation imposes a higher sustained compute load per request regardless of model size.

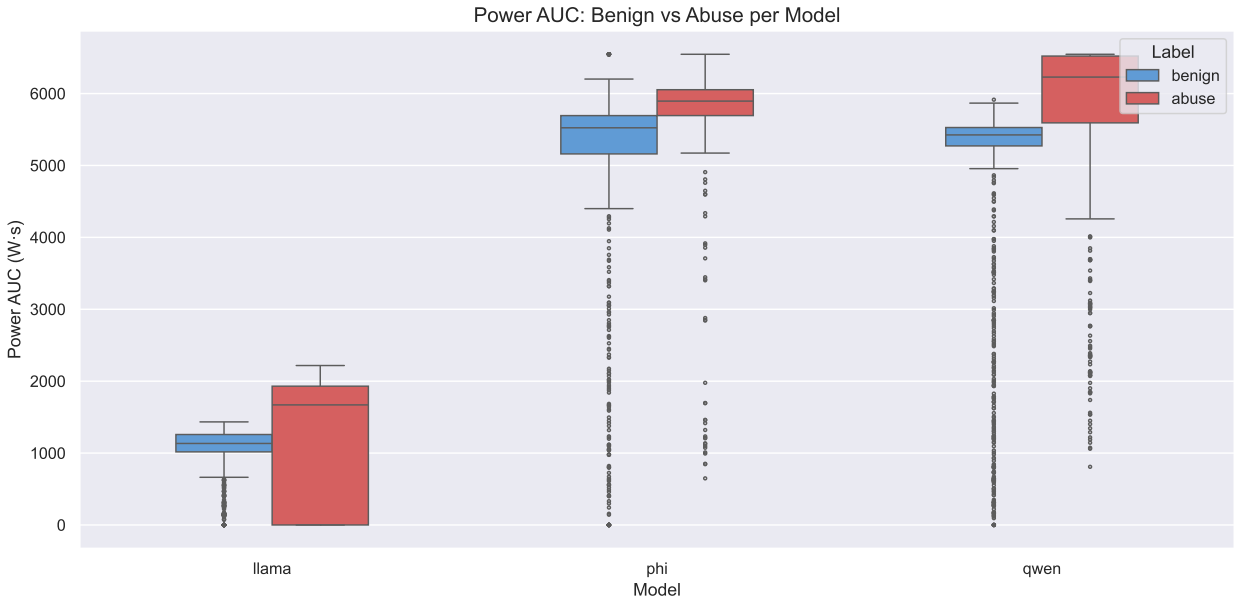

Figure 4: Distribution of power-AUC by model and label

4.1.3 Classifier Performance

Table 9 compares two feature configurations across all four classifiers. The telemetry-only set consists of 11 GPU power and utilization features: mean, max, min, standard deviation, AUC, and range of power draw, plus mean, max, and standard deviation of GPU compute utilization, power

per output token, and mean memory used. The timing-only set uses only `latency_s`, `tokens_out`, and `tokens_in`—the three features most likely to reflect corpus-level differences between HarmBench and LMSYS-Chat-1M rather than genuine hardware signal. If the telemetry classifier's performance were driven primarily by length and latency proxies, the timing-only baseline would approach similar AUC scores.

All four telemetry-only classifiers achieve near-perfect test set performance on the 80/20 stratified split. This performance is not expected to persist outside controlled conditions and should be read as an upper bound. The practical false negative rate is approximately 0.88%, meaning fewer than 1 in 100 abuse sessions pass through undetected. The false positive rate is 0.0% for the tree-based classifiers, meaning no benign session is incorrectly flagged in this evaluation. The timing-only baseline achieves AUC of 0.86 to 0.96 depending on classifier (Figure 6), with substantially higher FPR and FNR across all four models. Figure 5 shows the near perfect ROC curves for Telemetry-Only feature set. The performance gap, and in particular the cross-model holdout collapse confirms that the telemetry signal is not reducible to response length or request latency.

Table 9: Hold-out test performance

Model	Feature Set	Test AUC	F1	Precision	FPR	FNR
Random Forest	Telemetry-only	0.9998	0.9956	1.000	0.000	0.009
XGBoost	Telemetry-only	0.9994	0.9956	1.000	0.000	0.009
LightGBM	Telemetry-only	0.9992	0.9956	1.000	0.000	0.009
Neural Network	Telemetry-only	0.9996	0.9956	1.000	0.000	0.009
Random Forest	Timing-only	0.9565	0.8294	0.814	0.079	0.154
XGBoost	Timing-only	0.9525	0.8172	0.765	0.110	0.123
LightGBM	Timing-only	0.9327	0.7692	0.712	0.139	0.163
Neural Network	Timing-only	0.8617	0.6430	0.495	0.381	0.084

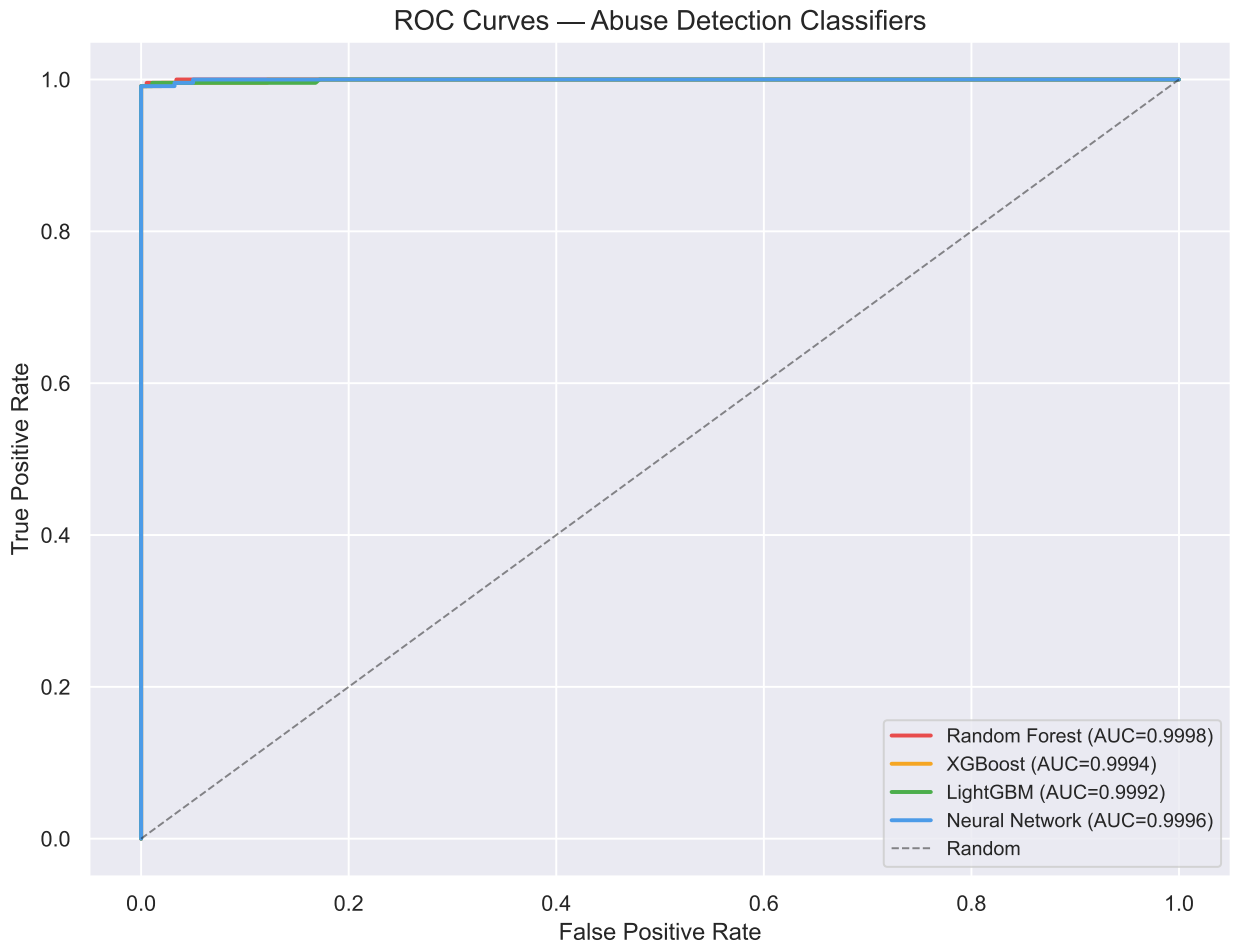

Figure 5: ROC curves for all **telemetry-only** classifiers

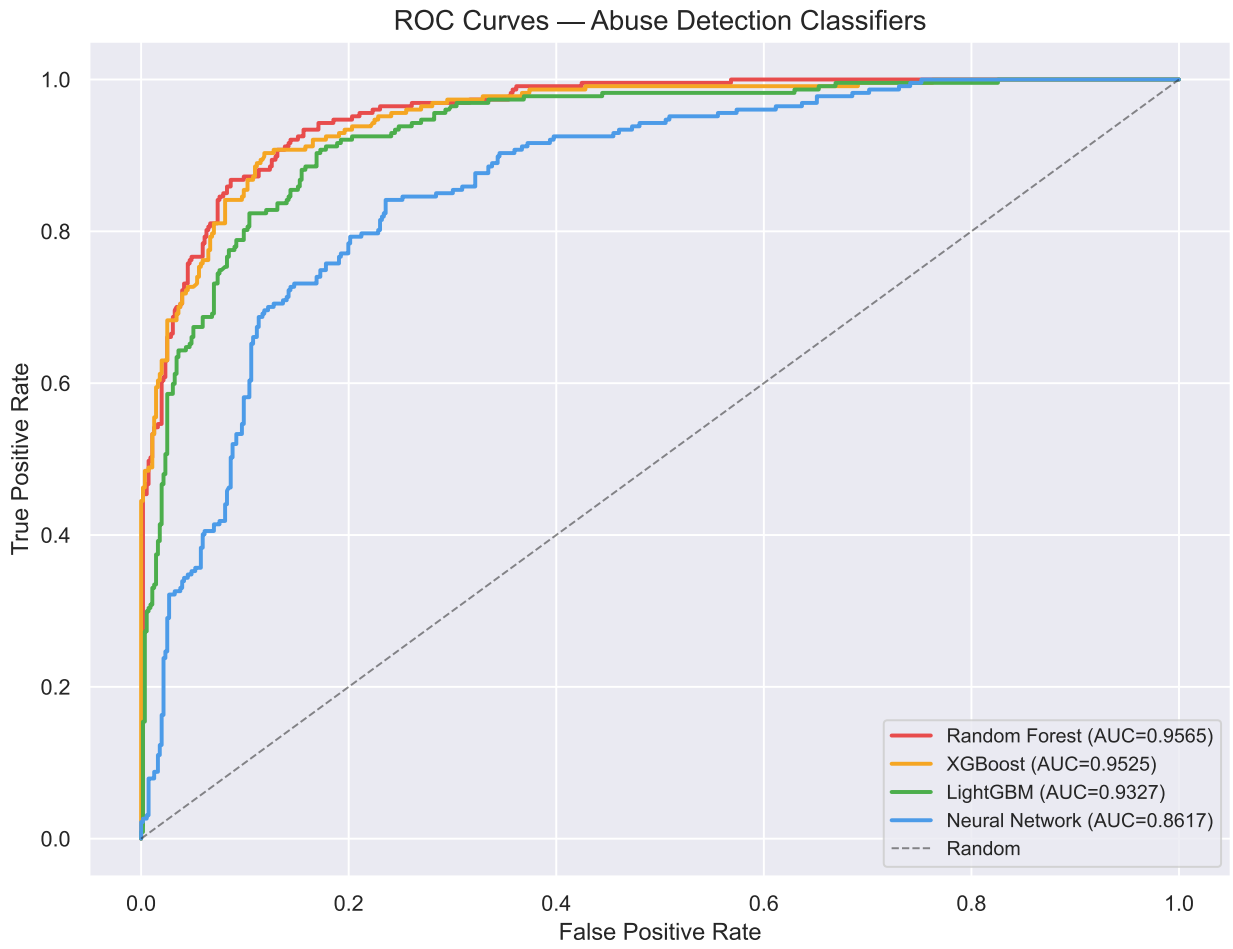

Figure 6: ROC curves for the **timing-only** feature set

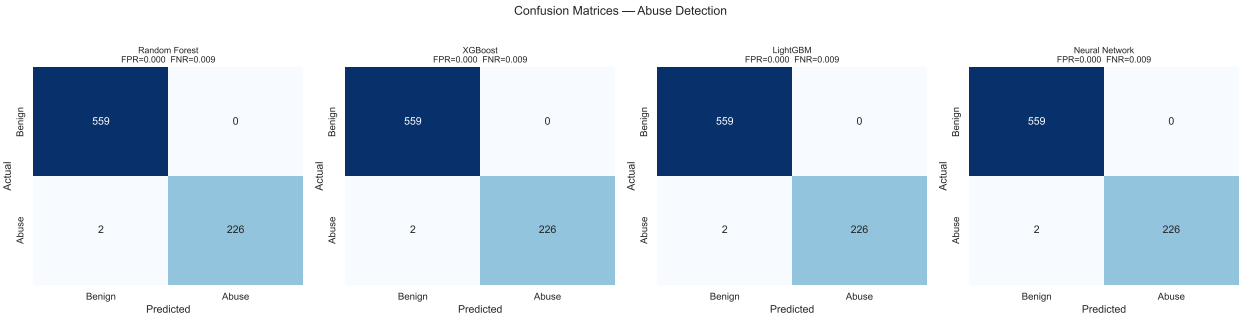

Figure 7: Confusion matrices

4.1.4 Feature Importance

SHAP analysis and tree-model mean decrease in impurity (MDI) converge on a consistent set of top-ranked features across all three tree-based classifiers. The consensus top features, appearing in the top 5 of at least two tree models, are `power_mean_w`, `power_std_w`, `power_min_w`, and `power_range_w`. This consensus confirms that the detection signal is driven by the shape and level

of the power trace rather than any single derived statistic.

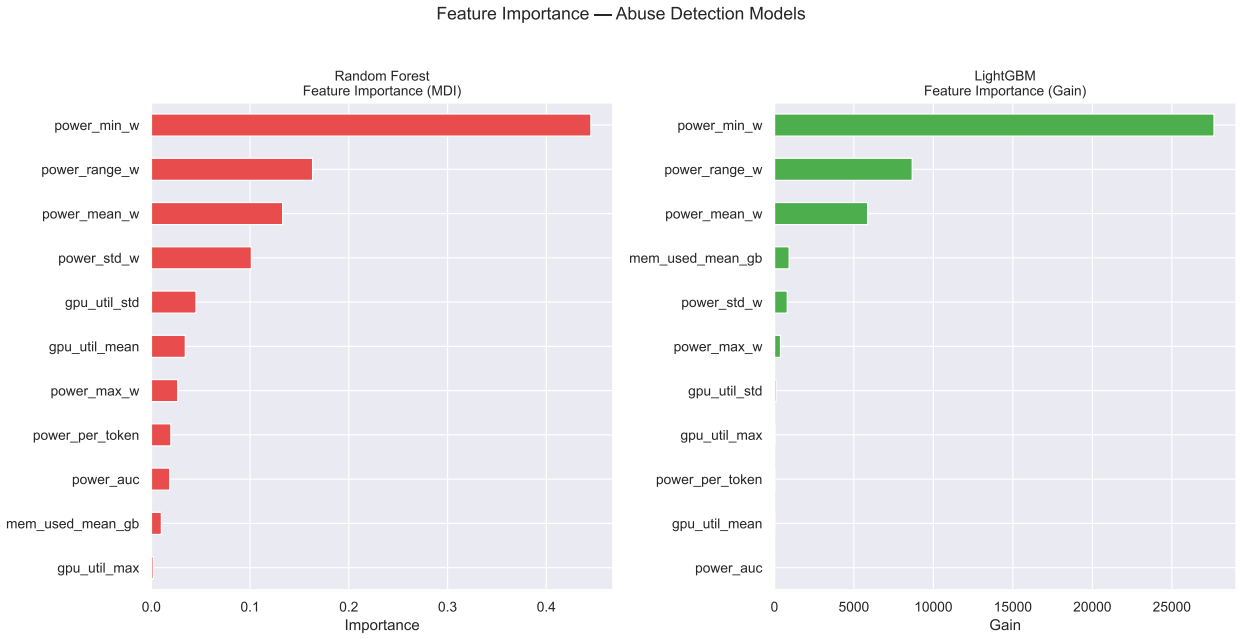

Figure 8: Feature importance across Random Forest (MDI) and LightGBM (Gain)

4.1.5 Neural Network Training Dynamics

The MLP achieved validation AUC of 0.9975 by epoch 10 and converged at 0.9996 by epoch 50 with no evidence of overfitting, confirming that the telemetry signal is learnable within a few thousand gradient steps even under a small training set.

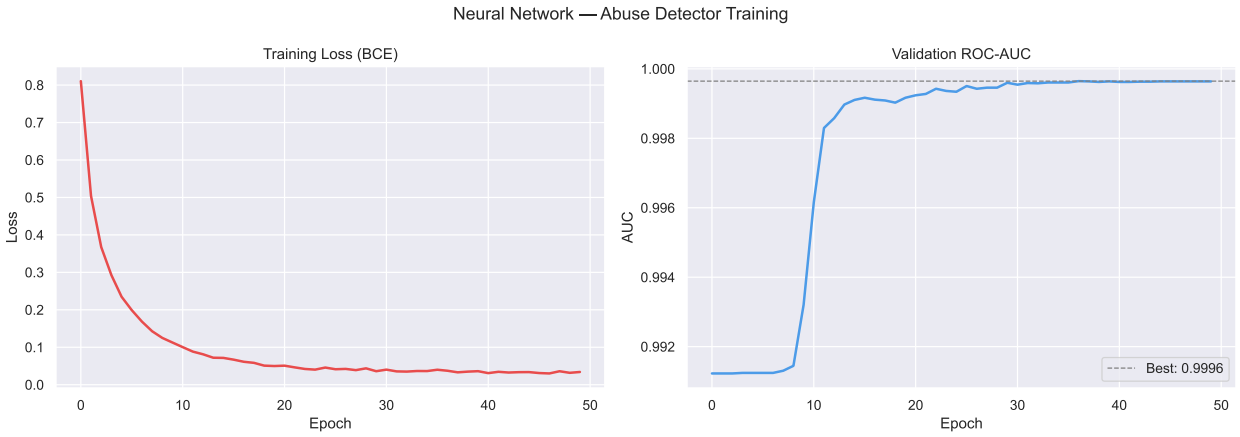

Figure 9: MLP training dynamics. Rapid convergence to AUC > 0.997 within 10 epochs indicates the power telemetry signal is learnable with minimal network capacity.

4.2 Out-of-Distribution Performance

Two out-of-distribution evaluations were conducted to probe the limits of the trained models: a cross-architecture holdout and an independent large-scale validation run (`qwen_wild`).

4.2.1 Cross-Architecture Holdout

Table 10 shows the leave-one-model-out evaluation; classifiers trained on two model configurations generalize well to the excluded third. AUC remains above 0.997 and F1 above 0.990 in all three holdout conditions, demonstrating that the telemetry signature is not specific to a single model architecture but reflects a property of the inference workload class.

Table 10: Cross-architecture holdout results

Holdout Model	Train Models	Train N	Test N	AUC	F1
LLAMA	PHI + QWEN	2,642	1,289	0.9988	0.9925
PHI	LLAMA + QWEN	2,687	1,244	> 0.997	> 0.990
QWEN	LLAMA + PHI	2,533	1,398	> 0.997	> 0.990

4.2.2 `qwen_wild`: Distribution Shift and Primary Limitation

To test generalization to a structurally distinct, higher-volume dataset, all four classifiers were applied to a separate out-of-distribution corpus of 16,000 sessions generated by submitting prompts from the WildJailbreak dataset [21] to Qwen2.5-7B-Instruct under the same telemetry infrastructure but in an independent collection run (8,000 abuse, 8,000 benign). WildJailbreak contains both vanilla harmful prompts and adversarially-crafted variants; the adversarial prompt was used where available, falling back to the vanilla prompt otherwise. This corpus is meaningfully distinct from the HarmBench DirectRequest training data in prompt construction, adversarial diversity, and collection timing, making it a strong test of out-of-distribution generalization.

This evaluation represents near-complete failure on the benign class and is the primary open problem identified by this research. An FPR of 0.999 means that in the qwen_wild results, essentially every benign session is classified as abuse, which would make the system unusable in production.

Performance on this corpus degrades substantially relative to the in-distribution hold-out observed in Table 11. Random Forest AUC drops to 0.45 with an FPR of 0.999, reflecting near-complete failure on the benign class. XGBoost achieves the highest AUC of the four models but still shows near-complete failure on benign classification.

Table 11: qwen_wild out-of-distribution evaluation results.

Model	AUC	F1	FPR	FNR
Random Forest	0.4458	0.334	0.999	0.0001
XGBoost	0.4520	0.334	0.9989	0.0004
LightGBM	0.5391	0.333	0.9991	0.0006
Neural Network	0.8617	0.334	0.9986	0.001

Figure 10: Power distribution comparison between training Qwen sessions and qwen_wild dataset

Distribution analysis in Figure 10 reveals the source of degradation. In the training distribution, benign Qwen sessions exhibit a mean `power_mean_w` of 243.4 W; in the qwen_wild benign corpus, this shifts to 297.6 W — nearly identical to the training abuse mean of 296.7 W. Simultaneously, qwen_wild abuse sessions show a mean of 297.3 W, producing a decision boundary compression from approximately 53 W in training to approximately 0.3 W in the wild corpus. At the 50th percentile, benign and abuse `power_mean_w` values are within 0.8 W of each other; at the 95th percentile, within 0.3 W. The distributions are effectively indistinguishable at any threshold, explaining the FPR of 0.999.

Two plausible causes contributed to this failure:

1. Ambient GPU thermal state drift between collection runs. The training data was collected earlier under cooler conditions than the qwen_wild run, raising the GPU's idle power baseline and shifting all subsequent readings upward.
2. Broader prompt diversity in WildJailbreak, which includes both vanilla and adversarially-crafted variants spanning a wider length and complexity range than the HarmBench DirectRequest training prompts. The adversarial variants in particular may drive benign baseline power upward when the model processes longer, more structurally complex inputs.

Both factors likely contribute and their relative impact is not isolated in this study. This is a critical limitation.

4.2.3 Power Cost Differential

Independent of classification performance, the telemetry data quantifies the compute cost differential between abuse and benign inference. Across models, abuse sessions draw a mean of 65 W more than benign sessions on a per-request basis at the GPU level. Table 12 summarizes this differential per model.

Table 12: Power consumption differential

Model	Benign μ (W)	Abuse μ (W)	Δ (W)	% Increase
LLAMA	156.6	245.5	+88.9	+56.8%
PHI	232.8	279.7	+46.9	+20.1%
QWEN	243.4	296.7	+53.3	+21.9%
Combined	210.6	275.6	+65.0	+30.9%

At cloud GPU spot pricing (approximately \$0.50–\$2.00 per GPU-hour, depending on hardware class), a 30.9% per-request power elevation translates directly to a proportional inference cost increase. Power variance and inference duration were consistently among the strongest predictors. This suggests that sustained compute intensity is a key distinguishing signal. At scale, this differential provides an operational economics rationale for abuse detection that is independent of harm classification: abuse sessions consume disproportionate compute relative to their user-value.

5 Conclusion

5.1 Summary of Findings

This research investigated whether GPU hardware telemetry collected during inference carries sufficient signal to classify LLM sessions as benign or abusive without accessing prompt or response content. The results strongly support the hypothesis within the experimental distribution, subject to dataset bias. The power signal may partially reflect distributional differences between the LMSYS and HarmBench corpora rather than abuse-specific GPU behavior exclusively.

Abuse inference sessions produce a statistically significant and directionally consistent elevation in GPU power draw relative to benign sessions across all three model architectures tested. Mann-Whitney U tests on the combined dataset ($n = 3{,}931$) rejected the null hypothesis of equal power distributions at $p < 0.001$ for all five primary power features. The mean power differential is +65.0 W on a per-request basis, corresponding to a 30.9% increase over the benign baseline, with the strongest per-architecture differential observed on Llama-3.2-1B (+88.9 W, +56.8%). Critically, this directional signal is consistent across architecturally distinct models despite substantial differences in absolute power magnitude, confirming that the signal reflects a property of the inference workload class rather than a model-specific artifact.

The most discriminative individual features are `power_min_w` ($\rho = 0.913$), `power_std_w` ($\rho = 0.644$), and `gpu_util_std` ($\rho = 0.614$). The direction of the standard deviation signal is noteworthy: abuse sessions exhibit *lower* power variance than benign sessions, consistent with the sustained plateau-like compute load of generating detailed harmful content versus the burst-and-idle patterns of natural conversational responses. This finding has practical implications for detection design: a classifier that looks only at peak power will underperform one that also captures the *shape* of the power trace.

All four classifiers (Random Forest, XGBoost, LightGBM, and a three-layer MLP) trained exclusively on the 11-feature telemetry-only set achieve near-identical in-distribution performance: AUC ≥ 0.999, F1 = 0.9956, precision = 1.000, FPR = 0.000, and FNR = 0.0088 on the stratified

80/20 hold-out. The MLP converged to AUC > 0.997 within ten training epochs, confirming that the telemetry signal is learnable with minimal model capacity and that deep architectures confer no advantage in this setting. The cross-architecture leave-one-model-out holdout further demonstrates generalization: classifiers trained on two model configurations maintain AUC > 0.997 and F1 > 0.990 when evaluated on the excluded third model.

Taken together, these results establish a proof-of-concept for content-free abuse detection grounded in physical hardware signals. Removing content inspection eliminates the sensitive data store that content-based systems require, which reduces regulatory audit scope under the EU AI Act and CCPA. The power signal of an abusive session is detectable within the first inference window, creating a pathway for early termination in a streaming-capable deployment. At cloud GPU spot pricing of \$0.50–\$2.00 per GPU-hour, a sustained 30.9% per-request power increase directly raises the cost attributable to abuse, giving operators an economics-based rationale for early detection that holds regardless of harm classification.

5.2 Limitations and Threats to Validity

Four threats to validity qualify the results presented in this paper. They are stated here where the reader has full context from the results.

Dataset bias. Benign sessions (LMSYS-Chat-1M) and abuse sessions (HarmBench DirectRequest) come from structurally different corpora with differing prompt and response lengths, and domain distributions. The power signal may partially reflect these differences rather than abuse-specific compute behavior exclusively.

Non-adaptive attack lower bound. The training corpus uses direct prompts with no iterative refinement, gradient optimization, or multi-turn escalation. All reported performance figures are lower bounds on worst-case adversarial attack success rate. An adversary aware of the detector could also craft inputs to minimize power draw.

Single-hardware scope. All experiments ran on one NVIDIA RTX 5080 under WSL 2. The power thresholds defining the decision boundary are hardware-specific and will not transfer directly to A100/H100 deployments without recalibration.

Prediction target scope. This study detects harmful content generation events: sessions where the model produces a non-refusing response to a harmful prompt. The classifier identifies that the GPU did abuse-level work, not the specific attack method used.

Distribution sensitivity. The qwen_wild evaluation reveals a fifth and most practically urgent limitation. The benign power baseline shifted from 243.4 W in training to 297.6 W in the wild corpus, compressing the decision boundary from 53 W to 0.3 W and collapsing Random Forest AUC to 0.45 with FPR = 0.999. Until per-session baseline calibration is implemented, in-distribution results should not be extrapolated to environments with different thermal profiles or benign query populations.

Taken together, these limitations define the boundary conditions for interpreting all results in this paper: strong in-distribution performance on a single GPU under controlled conditions, with an unresolved distribution shift problem that currently prevents deployment outside the training environment.

5.3 Future Work

Per-session baseline normalization. The `qwen_wild` distribution shift is attributable in part to GPU thermal state variation between collection runs. A dynamic calibration procedure—measuring idle power at session start—is expected to significantly narrow the in-distribution/out-of-distribution performance gap. A rolling window approach, updating the baseline continuously from the most recent k benign-labeled sessions, warrants evaluation.

Multi-GPU at scale. Re-running the telemetry collection pipeline on NVIDIA A100 and H100 hardware under production vLLM configurations will determine whether the power signature transfers to the infrastructure class where commercial LLM abuse is most consequential.

Adaptive attack evaluation. A rigorous adversarial evaluation should construct prompts explicitly designed to minimize power draw. Measuring the resulting degradation in detection accuracy will establish the true operational floor of hardware-signature-based detection.

Multi-turn and streaming detection. The current design aggregates telemetry over a completed request window. Multi-turn attack methods such as Crescendo and AutoAdv distribute the attack signal across multiple inference calls, each of which may individually resemble a benign session.

Formal baseline classifier comparison. A timing-only baseline (classifiers trained on `latency_s`, `tokens_out`, and `tokens_in`) was evaluated and reported in Section 4.1.3 alongside the telemetry classifiers. This result supports the primary contribution claim: GPU power features generalize across model architectures in a way that timing and token-count features do not. The remaining open question is whether the in-distribution AUC gap between the baseline (0.96) and the telemetry classifier (0.999) is fully explained by the hardware signal or partially by `tokens_in`'s corpus-level verbosity or distribution difference.

Defense ensemble integration. This detector is most appropriately deployed as one layer in a heterogeneous defense stack. An ensemble combining the power telemetry classifier with a lightweight semantic perplexity filter and a session-level KV-cache anomaly detector would address both the adaptive attack limitation and the prompt-level coverage gaps inherent to any single-signal approach. The power signal maps directly to a physical quantity with no ambiguity about what it measures, making it a natural anchor for such an ensemble. It provides a backstop that is immune to semantic obfuscation while semantic filters cover low-cost, low-power attack classes.

5.4 Closing Remarks

The central contribution of this work is that abusive LLM inference sessions leave a measurable and classifiable footprint in GPU hardware telemetry that is detectable without reading session prompts or responses. Within the experimental distribution, that signal supports near-perfect discrimination with off-the-shelf classifiers trained on eleven features. The out-of-distribution results are an honest acknowledgment that this signature is sensitive to environmental variation and not yet sufficient for production deployment without per-session calibration. Harmful content generation imposes sustained, intensive, low-variance compute load. That is a consequence of what the model is being asked to do. This work demonstrates that hardware-level telemetry contains a measurable signal, but that signal is fragile and highly context-dependent.

Bibliography

[1] Nazi, Z. A. and Peng, W. (2024). *Large Language Models in Healthcare and Medical Domain: A Review. Informatics*, 11(3), 57. https://doi.org/10.3390/informatics11030057

[2] Jin, W., Wang, N., Zhang, L., Tian, X., Shi, B., and Zhao, B. (2025). *A Review of AI-Driven Automation Technologies: Latest Taxonomies, Existing Challenges, and Future Prospects. Computers, Materials & Continua*, 84(3), 3961–4018. https://doi.org/10.32604/cmc.2025.067857

[3] Roumeliotis, K. I., Tselikas, N. D., and Nasiopoulos, D. K. (2024). *LLMs in E-Commerce: A Comparative Analysis of GPT and LLaMA Models in Product Review Evaluation. Natural Language Processing Journal*, 6, 100056. https://doi.org/10.1016/j.nlp.2024.100056

[4] Malik, S., Muhammad, K., and Waheed, Y. (2024). *Artificial Intelligence and Industrial Applications — A Revolution in Modern Industries. Ain Shams Engineering Journal*, 15(9), 102886. https://doi.org/10.1016/j.asej.2024.102886

[5] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. *Jailbreaking Black Box Large Language Models in Twenty Queries.* arXiv:2310.08419, 2023.

[6] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. *Tree of Attacks with Pruning: Automatic Jailbreaking of Large Language Models.* NeurIPS 2024.

[7] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. *Universal and Transferable Adversarial Attacks on Aligned Language Models.* arXiv:2307.15043, 2023.

[8] Yige Li, Zhe Li, Wei Zhao, Nay Myat Min, Hanxun Huang, Xingjun Ma, and Jun Sun. *AutoBackdoor: Automating Backdoor Attacks via LLM Agents.* arXiv:2511.16709, 2025.

[9] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. *"Do Anything Now": Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models.* arXiv:2308.03825, 2024.

[10] Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. *A Comprehensive Study of Jailbreak Attack versus Defense for Large Language Models.* arXiv:2402.13457, 2024.

[11] OWASP Foundation. *OWASP Top 10 for Large Language Model Applications.* OWASP Foundation, 2025.

[12] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. *AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models.* arXiv:2310.04451, 2024.

[13] Matthew Pisano, Peter Ly, Abraham Sanders, Bingsheng Yao, Dakuo Wang, Tomek Strzalkowski, and Mei Si. *Bergeron: Combating Adversarial Attacks through a Conscience-Based Alignment Framework.* arXiv:2312.00029, 2024.

[14] Shenyi Zhang, Yuchen Zhai, Keyan Guo, Hongxin Hu, Shengnan Guo, Zheng Fang, Lingchen Zhao, Chao Shen, Cong Wang, and Qian Wang. *JBShield: Defending Large Language Models from Jailbreak Attacks through Activated Concept Analysis and Manipulation.* arXiv:2502.07557, 2025.

[15] Najmeh Nazari Bavarsad, Furi Xiang, Chongzhou Fang, Hosein Mohammadi Makrani, Aditya Puri, Kartik Patwari, Hossein Sayadi, Setareh Rafatirad, Chen-Nee Chuah, and Houman Homayoun. *LLM-FIN: Large Language Models Fingerprinting Attack on Edge Devices*. In Proceedings of the IEEE International Symposium on Quality Electronic Design (ISQED), 2024.

[16] Daniele Esposito, Francesco Lomio, Heikki Huttunen, and Pekka Jääskeläinen. *GPU Under Pressure: Estimating Application's Stress via Telemetry and Performance Counters*. arXiv:2511.05067, 2025.

[17] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. *Characterizing Power Management Opportunities for LLMs in the Cloud*. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2024.

[18] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, and George J. Pappas. *JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models*. NeurIPS 2024 Datasets and Benchmarks Track.

[19] Mazeika, M., Phan, L., Yin, X., Zou, A., Wang, Z., Mu, N., Sakhaee, E., Li, N., Basart, S., Li, B., Forsyth, D., & Hendrycks, D. (2024). HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. *arXiv:2402.04249*.

[20] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *arXiv:2306.05685*.

[21] Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. *WildTeaming at Scale: From In-the-Wild Jailbreaks to (Adversarially) Safer Language Models*. arXiv:2406.18510, 2024.

[22] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. *SOSP 2023*.

[23] Mark Russinovich, Ahmed Salem, and Ronen Eldan. *Great, Now Write an Article About That: The Crescendo Multi-Turn LLM Jailbreak Attack*. arXiv:2404.01833, 2024.

[24] Chenxu Niu, Wei Zhang, Jie Li, Yongjian Zhao, Tongyang Wang, Xi Wang, and Yong Chen. *TokenPowerBench: Benchmarking the Power Consumption of LLM Inference*. arXiv:2512.03024, 2025.

[25] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. *On Evaluating Adversarial Robustness*. arXiv:1902.06705, 2019.

[26] Chen, T. & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *KDD 2016*, pp. 785–794.

[27] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *NeurIPS 2017*.

6 Appendix

A. Code and Notebooks

All source code, data collection notebooks, and analysis notebooks for this project are publicly available at:

`https://github.com/dataglancer/llmabusedetection`

The repository includes:

- model_telemetry_collector_public.py - the Flask proxy and pynvml GPU sampler used for all data collection
- abuse_detection_eda_ml_public.ipynb - notebook used for all of the analysis and ML generation for the report.
- time_branch_abuse_detection_eda_ml_public.ipynb - identical notebook except branches using only time-based features.

Note, these files do not include prompt generation code for safety reasons. Those libraries can be obtained outside of this project.

B. Classifier Hyperparameters

Table 13 summarizes the configuration used for each classifier. Full descriptions are in Section 3.3.

Table 13: Classifier hyperparameter configurations.

Classifier	Trees / Est.	Max Depth	LR	Subsample	Class Balance
Random Forest	300	—	—	—	`class_weight=balanced`, `min_samples_leaf=5`
XGBoost	400	6	0.05	80%	`scale_pos_weight` (neg/pos ratio)
LightGBM	400	7 (63 leaves)	0.05	80%	`class_weight=balanced`
MLP (PyTorch)	—	3 layers [128, 64, 32]	1×10^{-3}	—	pos_weight = imbalance ratio

MLP trained for 50 epochs; dropout $p = 0.3$; Adam optimizer; weight decay 1×10^{-4}.

C. LLM Attack Tools

Technique	Description
PAIR (Prompt Automatic Iterative Refinement)	Uses an attacker LLM to iteratively refine prompts via conversation history and chain-of-thought reasoning until the target model complies.
TAP (Tree of Attacks with Pruning)	Employs tree-of-thoughts reasoning; a separate evaluator LLM prunes low-probability branches before querying the target, improving query efficiency over PAIR.
GAP (Graph of Attacks with Pruning)	Extends TAP with a graph structure that shares successful vulnerability patterns across iterations, enabling knowledge reuse and more efficient exploration.
AutoDAN	Applies a hierarchical genetic algorithm at both sentence and word level to evolve semantically coherent jailbreak prompts, seeded from effective handcrafted examples.
GCG (Greedy Coordinate Gradient)	Gradient-based white-box optimization that greedily updates an adversarial token suffix appended to the prompt to maximize the likelihood of harmful output.
Crescendo	Gradually escalates a multi-turn dialogue from benign topics toward harmful content, exploiting the model's own prior outputs to steer the conversation via foot-in-the-door psychology.
AutoAdv	A two-phase multi-turn framework: a pattern manager learns successful strategies across turns while a temperature manager adjusts sampling; outputs are rewritten to disguise intent.
Tempest	Performs tree-search over conversational branches, exploiting partial compliance at each node to explore alternative dialogue paths toward full harmful disclosure.
AutoDAN-Turbo	A lifelong black-box agent that autonomously discovers and accumulates novel jailbreak strategies from scratch without human-crafted seeds.
IRIS (Iterative Refinement Induced Self-Jailbreak)	Prompts the target model to self-explain each step of a harmful task, then iteratively feeds those self-generated explanations back as refinement context, exploiting its own reasoning.
AdvPrompter	Trains a separate LLM via alternating optimization to generate human-readable adversarial prompts that conceal harmful intent from both the target model and safety classifiers.
ArrAttack	Trains a universal robustness-judgment model to generate jailbreak prompts that remain effective across multiple defenses, explicitly bridging the gap between attacks and mitigations.

Continued on next page...

Technique	How It Works
Rainbow Teaming	Frames jailbreak discovery as a quality-diversity search problem, generating a maximally diverse set of effective strategies to map the full vulnerability landscape of a model.
Adaptive Attacks	Performs random search over adversarial suffix space using model-specific adaptive prompt templates, exploiting known structural vulnerabilities with minimal queries.
DAN (Do Anything Now)	Manually crafted role-playing prompts that instruct the model to adopt an alter-ego unconstrained by safety guidelines, using instruction-override framing to suppress refusals.
Multilingual GCG / AutoDAN	Extends gradient-based (GCG) and genetic (AutoDAN) attacks across multiple languages, exploiting cross-lingual safety gaps where non-English inputs bypass alignment training.
MAGIC (Index GCG)	Improves GCG efficiency via gradient-based index selection and adaptive multi-coordinate updates, reducing the number of iterations needed to find an effective adversarial suffix.
Prefilling / In-Context Attacks	Exploits API-level prompt prefilling or in-context learning to manipulate the model's context window, inducing compliance before safety evaluation can occur.
garak	Open-source LLM vulnerability scanner with a modular probe library covering encoding attacks, continuation exploits, DAN variants, and malware generation; integrates with HuggingFace and OpenAI APIs for batch automated testing.
PyRIT	Microsoft's Python orchestration framework for multi-turn red-teaming; provides a memory store, scoring engine, and pluggable target adapters to automate and log attack campaigns at scale.
HarmBench	Unified evaluation framework implementing 18 standardized attack methods (GCG, PAIR, TAP, AutoDAN, PAP, and others) within a single reproducible pipeline for comparative benchmarking.
LLM-Fuzzer	Applies coverage-guided mutation-based fuzzing to jailbreak template discovery; seeds an initial template corpus, applies semantic-preserving mutations, and prioritizes inputs that reach new model behaviors.
EasyJailbreak	Modular Python framework built around a Selector–Mutator–Constraint–Evaluator pipeline that enables rapid composition and prototyping of novel attack strategies from existing components.

Continued on next page...

Technique	How It Works
JailbreakBench	NeurIPS 2024 standardized benchmark providing the 100-behavior JBB-Behaviors dataset, pre-generated attack artifacts (PAIR, TAP, GCG, AutoDAN), a Llama-3-70B judge, and a public leaderboard for reproducible evaluation.
PAP (Persuasion-based Adversarial Prompts)	Applies over 40 social-scientific persuasion techniques—including appeals to authority, reciprocity, and logical framing—to reframe harmful requests as benign, producing fully human-readable outputs.
BEAST (Beam Search Adversarial Attack)	Applies beam search over the token space to craft adversarial suffixes, balancing attack success against linguistic fluency and achieving higher efficiency than standard GCG.
Jailbroken (Formal Analysis)	Formal compositional analysis identifying competing objectives and generalization flaws as root causes of jailbreak success; provides reusable attack primitives such as Base64 encoding, prefix injection, and refusal suppression.
Many-shot Jailbreaking (MSJ)	Embeds hundreds of faux few-shot question–answer demonstrations of harmful compliance into the prompt, exploiting long-context windows so that sheer in-context volume overrides safety alignment.
Cipher / Encoding Attacks	Encodes harmful payloads using Caesar cipher, Base64, Morse code, or model-invented ciphers; the model's code-training causes it to decode and comply without safety filters activating on the obfuscated input.