ThermalPrint: ML Workload Classification via Thermal Side-Channels

Cenxi Tian
Department of Computing
Imperial College London
London, United Kingdom
c.tian24@imperial.ac.uk

Soteris Demetriou
Department of Computing
Imperial College London
London, United Kingdom
s.demetriou@imperial.ac.uk

Abstract

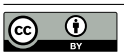
We present ThermalPrint, a thermal side-channel attack that classifies on-device ML inference workloads on the Google Tensor G2 SoC using two previously unstudied Android interfaces: the zero-permission `getThermalHeadroom()` thermal management API, callable without declared permissions since API Level 30 (2020), and the privileged sysfs hardware interfaces of the EdgeTPU and Mali GPU. We collect 60 labelled sessions spanning four on-device ML workload classes on a Pixel 7 and record 227 channels at 1 Hz and 0.5 Hz (226 sysfs at 1 Hz, 1 API at 0.5 Hz). We train InceptionTime under 5-fold stratified cross-validation and show that it achieves 1.0±0.0 across all five folds on all 227 channels. We confirm that Logistic Regression on the identical 34,050-dimensional feature space also achieves 1.0 ± 0.0, matching InceptionTime and ruling out a deep-model-specific artefact rather than overfitting in general. We show in a channel-group ablation that the EdgeTPU sysfs, Mali GPU sysfs, and thermal zone sysfs groups are each independently sufficient for perfect classification; three hardware interfaces accessing separate kernel subsystems each achieve 1.0 ± 0.0. The zero-permission `getThermalHeadroom()` stream alone achieves 0.8 ± 0.16 (3.2× the 25% chance baseline; $p < 0.01$). We obtain these results on a single Pixel 7 in a single indoor laboratory environment with no concurrent foreground workloads. The identified interfaces constitute viable side-channels at two adversary capability levels in the four-class setting: a privileged adversary (Adversary 1) with root sysfs access achieves 1.0 ± 0.0 across all five folds; an unprivileged installed-app adversary (Adversary 2) restricted to `getThermalHeadroom()` alone achieves 0.8 ± 0.16, with per-fold accuracies ranging from 0.5 to 1.0.

CCS Concepts

• **Security and privacy → Side-channel analysis and countermeasures**; *Mobile platform security*.

Keywords

thermal side-channel, on-device machine learning, EdgeTPU, Mali GPU, Android, workload classification, getThermalHeadroom, zero-permission attack, mobile security



MobiSys Workshop '26, Cambridge, United Kingdom

ACM ISBN 979-8-4007-2712-2/26/06
https://doi.org/10.1145/3812836.3814784

ACM Reference Format:
Cenxi Tian and Soteris Demetriou. 2026. ThermalPrint: ML Workload Classification via Thermal Side-Channels. In *The 24th Annual International Conference on Mobile Systems, Applications and Services (MobiSys Workshop '26), June 21–25, 2026, Cambridge, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3812836.3814784

1 Introduction

On-device machine learning inference now runs continuously on mobile SoCs equipped with dedicated accelerators: the Google Tensor G2 EdgeTPU on Pixel 7 executes Photo Unblur [4] and on-device speech transcription [5] without leaving the device.

Recent ML side-channel work targets *model IP*: DNN layer structure from server power interfaces [3], DNN architectures from x86 thermal event interrupts [18], EdgeTPU hyperparameters from EM probes [9], and DNN architecture variants from a mobile-GPU runtime profiler [8]. None of these address *which model is running on a commodity mobile device right now* (§5 compares interfaces and threat assumptions in detail), even though workload identity itself reveals privacy-sensitive activity: a sustained transcription workload signals local audio processing; a vision-search workload signals visual-content lookup.

ML inference exercises the accelerator and GPU under sustained load, and resource-usage patterns have been shown to leak *user-task* identity in adjacent mobile settings. Wang et al. [17] classify iOS app activity at 87.3% AUC from OS-level resource-usage APIs (CPU, memory, network, storage). Oberhuber et al. [12] systematically analyse 9 Android devices and 137 unprivileged sensors in the Android Sensor Framework. Whether on-device ML *workload type* leaks through analogous resource-usage observables on accelerator-equipped Android devices, and through which interfaces, has not been answered.

Heat is one such observable. Hot Pixels [14] exploits DVFS-induced timing on desktop GPUs and ARM SoCs. Miedl et al. [10] read CPU thermal zones on pre-accelerator Android devices (Galaxy S5, Xperia Z5) to identify foreground applications. Neither covers post-accelerator mobile SoCs, where dedicated EdgeTPU and Mali GPU sysfs interfaces become available alongside CPU thermal zones, and where the zero-permission `getThermalHeadroom()` thermal management API [1] has not been studied as a side-channel primitive.

`getThermalHeadroom()` has been callable without declared permissions since API Level 30 (Android 11, 2020): any installed app may poll it silently via `JobScheduler`. On a Pixel 7 (Tensor G2), values drop from 0.8–0.9 at idle to 0.4–0.6 after 30 s of sustained EdgeTPU activity. To our knowledge, no prior security work has exploited this sensitivity as an attack primitive, and no prior work

targets the EdgeTPU or Mali GPU sysfs interfaces for workload inference.

We build a dual monitoring pipeline that records 226 sysfs channels via `thermal_collector.c` (1 Hz, root-only) and one `getThermalHeadroom()` stream via `ThermalTestService` (0.5 Hz, zero-permission); UIAutomator2 records workload window timestamps. We collect 60 labelled sessions spanning four on-device ML workload classes and train InceptionTime [2] under 5-fold stratified cross-validation. We show that InceptionTime achieves 1.0 ± 0.0 across all five folds on all 227 channels. We confirm that Logistic Regression on the same 34,050-dimensional feature space also achieves 1.0 ± 0.0: a strictly linear workload classifier on the raw data confirms linear separability, ruling out deep-model overfitting. The zero-permission `getThermalHeadroom()` stream alone achieves 0.8 ± 0.16, 3.2× the 25% chance baseline ($p < 0.01$). We show in an ablation study that the EdgeTPU sysfs, Mali GPU sysfs, and thermal zone sensor groups are each *independently sufficient* for perfect classification: three hardware interfaces accessing separate kernel subsystems each expose the same multi-subsystem thermal signature.

Our results provide the first empirical characterisation of these attack surfaces on the Google Tensor G2 platform, across two adversary capability levels defined in §2.3.

We make the following contributions:

- **Attack surface identification.**
  We demonstrate that two previously unstudied interface classes on the Google Tensor G2 SoC constitute viable side-channels: the zero-permission `getThermalHeadroom()` Android API and the privileged sysfs interfaces of the EdgeTPU (`/sys/devices/platform/1ce00000.janeiro/`) and Mali GPU (`/sys/devices/platform/28000000.mali/`). To the best of our knowledge, no prior security work targets the EdgeTPU sysfs interface, and `getThermalHeadroom()` has never been studied as a security primitive despite five years of availability on Android 11+.
- **Feasibility validation.** We frame the task as four-class workload classification — not binary ML/non-ML detection, model extraction, or generic user activity inference. On 60 labelled sessions spanning four ML workload classes on a single Pixel 7, five workload classifier families — Logistic Regression, Linear SVM, k-NN, Random Forest, and InceptionTime — each achieve ≥ 0.983 accuracy on all 227 channels; the convergence of five classifier families places the burden of explanation on a workload-level signal (§4.3). A permutation test ($p < 0.001$, 1,000 permutations) and a 5×5 nested cross-validation confirm robustness. The `getThermalHeadroom()` stream alone achieves 0.8 ± 0.16; this validates the Adversary 2 deployment path independently of sysfs access.

2 Background

2.1 Google Tensor G2 and the EdgeTPU

The Google Tensor G2 SoC integrates a dedicated ML accelerator — the EdgeTPU (code-name "Janeiro") [6] — alongside an octa-core ARM CPU cluster and a Mali-G710 MP7 GPU [7]. The EdgeTPU exports 22 sysfs channels at `/sys/devices/platform/1ce00000.janeiro/`, including performance counters (`inference_count`, `preempt_count`, `active_cycle_count`), power-management metrics, and utilisation channels. The Mali GPU exports 171 sysfs channels at `/sys/devices/platform/28000000.mali/`, including utilisation metrics, frequency-state counters, and an 11×11 DVFS transition matrix (`trans_stat`). Both interfaces require root or a privileged system process to read.

2.2 Android Thermal Management API

`PowerManager.getThermalHeadroom()` returns a float in $[0, 1]$ that estimates remaining thermal headroom over a 10 s forecast window. It has been accessible without declared permissions since API Level 30 (2020) [1], callable by any installed app. The API enforces a rate limit of approximately one call per second; `ThermalTestService` polls at 0.5 Hz (every 2 000 ms), leaving a two-fold margin below the rate limit.

2.3 Threat Model

Attacker goal. We consider an attacker who aims to identify which EdgeTPU-accelerated ML workload runs on the victim device, distinguishing among workload classes such as on-device OCR, speech-to-text, and local image enhancement.

Profiling and deployment asymmetry. We assume the attacker performs profiling on a same-model device, where privileged sysfs channels are readable (e.g., a developer with their own same-model rooted device); no victim access is required. At deployment time on the victim, only the zero-permission `getThermalHeadroom()` stream is available to an installed-app adversary. This training/deployment asymmetry follows the LUPI paradigm [16], in which signals available during training are absent at test time. We evaluate this asymmetry through the channel-group ablation in §4.4 rather than by deploying an end-to-end attack app (§6).

Capability levels. Adversary 1 has root or privileged system access and reads all 226 sysfs channels across the EdgeTPU (`/sys/devices/platform/1ce00000.janeiro/`), Mali GPU (`/sys/devices/platform/28000000.mali/`), and thermal zones during profiling. **Adversary 2** has only installed-app privileges and polls `getThermalHeadroom()` at 0.5 Hz from a foreground service; no permissions beyond those of any Play Store application are required. §4.4 evaluates each capability level independently on the same 60-session dataset and 5-fold splits.

3 System Design

The pipeline runs two collectors per session, one per adversary capability level (§2.3). Each level answers a question that prior work has not addressed on a post-accelerator mobile SoC.

Under Adversary 1, we test whether the EdgeTPU sysfs at `/sys/devices/platform/1ce00000.janeiro/`and the Mali GPU sysfs at `/sys/devices/platform/28000000.mali/`carry ML-workload-discriminative signal. Oberhuber et al. enumerate 137 unprivileged sensors across 9 Android devices in the Android Sensor Framework [12]; neither sysfs path appears among the 137 channels. Miedl et al. use only the 8 CPU thermal zone readings on Galaxy S5 and the single CPU cluster reading on Xperia Z5 as input for application detection, excluding the GPU zone available on S5 [10]. On a Tensor G2 device, neither interface has been measured.

At the unprivileged level (Adversary 2), we test whether the same signal reaches an installed app through `getThermalHeadroom()` alone. The API has been callable since Android 11 (2020) without declared permissions [1], and any Play Store application can poll it. No prior work measures it on post-accelerator devices.

3.1 Monitoring Pipeline

We implement a dual monitoring architecture (Figure 1) that records privileged sysfs channels and the unprivileged `getThermalHeadroom()` API simultaneously, with workload automation and ground-truth timestamps provided by UIAutomator2.

Privileged sysfs collector. We implement `thermal_collector.c` as a root C daemon that polls three sysfs interface groups at 1 Hz. The three groups are: **(i)** 22 EdgeTPU performance and power channels at `/sys/devices/platform/1ce00000.janeiro/`, including `inference_count`, `preempt_count`, and `active_cycle_count`; **(ii)** 171 Mali GPU channels at `/sys/devices/platform/28000000.mali/`, including per-frequency utilisation counters and the 11×11 DVFS frequency-state transition matrix (`trans_stat`, 11 operating points from 202 MHz to 848 MHz); and **(iii)** 33 thermal zone temperatures at `/sys/class/thermal/thermal_zone{0..32}/temp`. `thermal_collector.c` writes one row per second to separate CSV files per channel group; these are concatenated into a single session record during preprocessing.

Unprivileged API collector. We implement `ThermalTestService` as an Android foreground service that calls `PowerManager.getThermalHeadroom()` at 0.5 Hz (every 2 000 ms). Both `thermal_collector.c` and `ThermalTestService` use Unix epoch milliseconds as their time base; the two streams are therefore aligned by timestamp during preprocessing with no clock synchronisation required. Power-budget profiling of the polling service itself is deferred to the system study (§6).

Session coordination. `DualCollector.start(session_id)` starts `thermal_collector.c` and `ThermalTestService` simultaneously at the beginning of each session. UIAutomator2 then triggers the target workload and records workload-window boundaries as Unix epoch millisecond timestamps.

For Google Photos Unblur, Google Lens Camera, and Google Lens Search, the window spans the complete inference episode from trigger to result display (`workload_start` / `workload_end` keys in the results JSON). For Google Recorder Realtime, the window spans a fixed transcription segment (`collectors_started` / `collectors_stopped`), with `inference_count` confirming ≈12 EdgeTPU inference events per second throughout.

3.2 Dataset Preparation and Workload classifier

Channel alignment and filtering. We process each session in `prepare_phase1_data.py` in two stages. First, we align the 0.5 Hz `getThermalHeadroom()` stream to the 1 Hz sysfs reference frame by resampling both to a common grid equal to the number of thermal rows in the workload window. We then concatenate all channel groups horizontally into a single (raw_rows × 229) array. Second, we remove 2 raw columns (indices 6 and 31) that read zero across all 60 sessions; this leaves 227 channels per timestep.

Temporal normalisation. On-device ML workload durations vary across classes: Lens Search sessions produce ≈6–10 raw 1 Hz rows; Recorder Realtime sessions produce up to ≈40. We normalise each session to 150 timesteps via channel-wise linear interpolation: for each of the 227 channels we apply `scipy.interpolate.interp1d` on a `np.linspace(0, 1)` grid of length 150. We choose 150 because it exceeds the longest observed raw session (≈40 rows), so every session is upsampled and no session is truncated. InceptionTime's three inception modules with parallel convolutions at kernel sizes {10, 20, 40} produce a maximum receptive field of ≈120 timesteps at depth 3; 150 timesteps covers this field without padding waste. We then apply per-channel min-max normalisation to $[0, 1]$ across all 60 sessions. The final dataset satisfies `assert X.shape == (60, 150, 227)`.

Workload classifier. We use InceptionTime [2] as the workload classifier: a residual time-series convolutional network with three stacked inception modules and a global average pooling head. We train with the Adam optimiser (lr = 0.001, batch size 16, 100 epochs, `ReduceLROnPlateau` scheduler) under 5-fold stratified cross-validation (`StratifiedKFold`, `random_state=42`, 48/12 train/val split). §4.3 evaluates Logistic Regression, Linear SVM, k-NN, and Random Forest on the identical folds to validate the signal independently of deep-learning capacity.

4 Evaluation

4.1 Experimental Setup

We conduct all experiments on a single Google Pixel 7 (Tensor G2, Android 16, SELinux enforcing, rooted) in a single indoor laboratory environment without active ambient thermal control or external temperature logging. We select four on-device ML inference workloads on the basis of verified EdgeTPU or multi-subsystem thermal activity: **Google Photos Unblur** (Photos v6.x; `inference_count` increment ≥ 2 per session, EdgeTPU-documented [4]); **Google Lens Camera** (activated from within Google Photos; `inference_count` increment ≈ 0, yet thermally distinct from Lens Search); **Google Lens Search** (activated via standalone search bar; `inference_count` increment 236–276, mean 254); and **Google Recorder Realtime** (on-device speech-to-text [5]; ≈ 12 inferences per second, sustained audio transcription). The four workloads span more than two orders of magnitude in EdgeTPU invocation, yet the discriminative signal does not reduce to `inference_count` alone: Photos Unblur and Lens Camera are nearly indistinguishable on this counter (Δ=2 vs $\Delta\approx$0) and classify with zero error in §4.2. A simple-feature baseline isolating `inference_count` rate or session duration is reserved for the scaled evaluation (§6).

We collect 15 sessions per class (60 total) under UIAutomator2 automation. All sessions satisfy the shape assertion `X.shape == (60, 150, 227)`. We enforce a 60 s thermal equilibration period between sessions and verify that all 60 sessions recover to within 1 °C of their pre-session baseline across all monitored zones.

4.2 Phase 1 Results: Five-Fold Cross-Validation

We train and evaluate InceptionTime [2] under 5-fold stratified cross-validation (`StratifiedKFold`, `random_state=42`, 48/12 train/val split) on the full $X \in \mathbb{R}^{60\times150\times227}$ dataset. All five folds

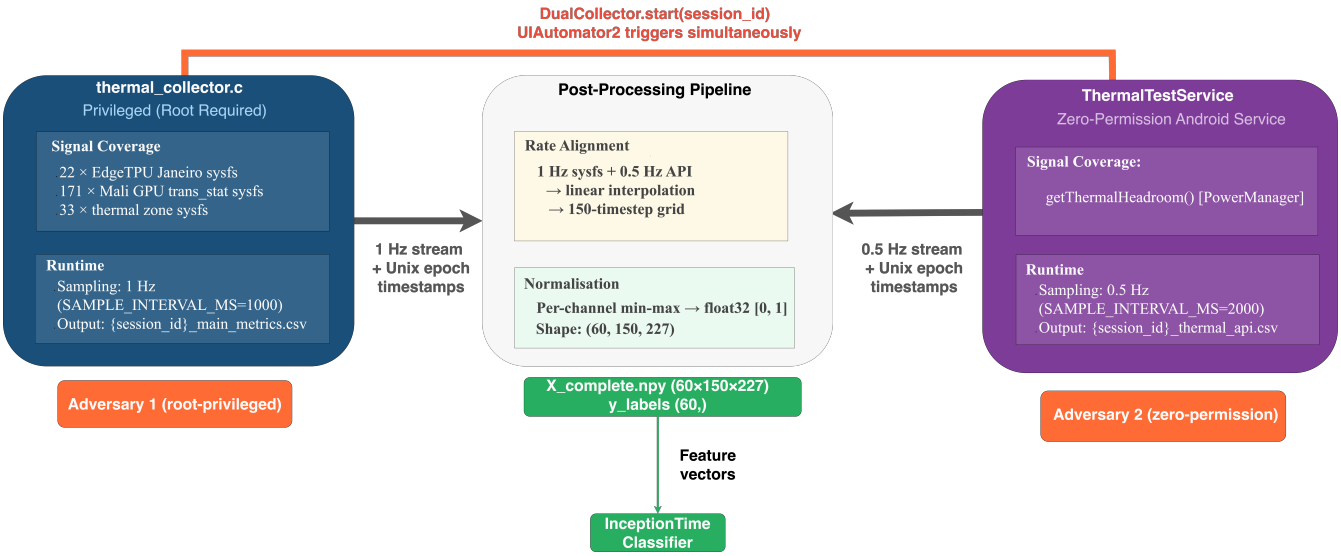


Figure 1: Dual monitoring architecture. `thermal_collector.c` (left) and `ThermalTestService` (right) are launched simultaneously per session via `DualCollector.start(session_id)`. Workload boundaries are defined by UIAutomator2 event timestamps. Sysfs (1 Hz) and API (0.5 Hz) streams are jointly resampled to a common 150-timestep grid. Output: `X_complete.npy` (60×150×227), `y_labels.npy` (60,). Both streams are recorded during profiling under root-privileged access (Adversary 1, §2.3); only the zero-permission `getThermalHeadroom()` stream (right) would be accessible to an installed-app adversary at deployment (Adversary 2; deployment app not implemented, §6).

achieve 1.0 (aggregate: 1.0 ± 0.0, Cohen's $\kappa = 1.0$); the confusion matrix is a perfect 4×4 diagonal: all 15 sessions per class are correctly classified across the full 60-sample dataset, with zero off-diagonal entries across all five partitions. For a balanced four-class dataset with perfect accuracy, macro-precision = macro-recall = macro-F1 = 1.0 ± 0.0 by definition. A retrain-based permutation test (1,000 label-shuffled iterations) [13] yields $p < 0.001$. A 5×5 nested cross-validation outer-loop accuracy matches standard CV [15]; this confirms negligible optimistic bias.

4.3 Overfitting Validation

We validate that perfect accuracy on n=60 reflects genuine signal, not overfitting. We evaluate four constrained workload classifier families on the *identical* five folds, with the same `random_state=42` and the full 34,050-dimensional flattened feature vector (227 × 150 timesteps, all four channel groups, no feature selection). We predefine an operational decision criterion: if the best simple workload classifier achieves ≥ 95% accuracy on identical splits, the signal is not InceptionTime-specific. We choose 95% as a high pass-bar relative to the 25% chance baseline rather than as a statistical significance threshold; the permutation test in §4.2 provides the latter.

Table 1 shows the results. Logistic Regression matches InceptionTime at 1.0 ± 0.0 on the identical splits. We treat this as a necessary consistency check, not a sufficient overfitting bound: with 60 samples in a 34,050-dimensional feature space, linear separability follows from dimensionality alone and does not by itself certify signal quality. The principal evidence for genuine signal is the permutation test and the 5×5 nested cross-validation reported in §4.2. The convergence of five classifier families — linear (Logistic Regression, Linear SVM), instance-based (k-NN), tree-based (Random Forest), and convolutional (InceptionTime) — at ≥ 0.983 accuracy on identical splits places the burden of explanation on a workload-level signal common to all five rather than on classifier-specific memorisation.

Table 1: Simple-baseline workload classifier accuracy on the same five-fold splits as InceptionTime. All classifiers use the full 34,050-dimensional flattened feature space (227 channels × 150 timesteps).

Workload classifier	Mean acc.	Std
InceptionTime	100.0%	±0.0%
Logistic Regression (L2)	100.0%	±0.0%
Linear SVM	100.0%	±0.0%
k-NN (k=3)	100.0%	±0.0%
Random Forest (10 trees)	98.3%	±3.3%

4.4 Channel Group Ablation

We evaluate 12 channel-group configurations: one full-channel baseline, seven removal tests (NO_X) and four sufficiency tests (ONLY_X). All removal tests maintain 1.0 ± 0.0: no single group is necessary. Three groups are each *independently sufficient*: ONLY_EDGETPU (22 channels at `/sys/devices/platform/1ce00000.janeiro/`), ONLY_GPU (171 channels at `/sys/devices/platform/28000000.mali/`), and ONLY_THERMAL (33 channels, `/sys/class/thermal/`

`thermal_zone{0..32}/temp`) each achieve 1.0 ± 0.0. The multi-subsystem thermal signature of each workload propagates independently through the EdgeTPU sysfs, the Mali GPU frequency fabric, and the thermal zone sensor network — three physically separate kernel subsystems that each constitute a sufficient standalone side-channel.

For Adversary 2, ONLY_THERMAL_API — a single float, 0.5 Hz, no declared permissions — achieves 0.8 ± 0.16 across the five folds; per-fold accuracies are $\{0.83, 0.83, 0.83, 1.0, 0.5\}$, with the worst fold at 2.0× the 25% chance baseline ($p < 0.01$). With each fold containing only 12 test samples, the 16-percentage-point standard deviation reflects the n=60 budget; four of five folds reach ≥ 0.83.

Pairwise sufficiency tests (e.g., EdgeTPU+thermal vs each alone) are not informative when each group already saturates at 1.0 ± 0.0 on the privileged side; redundancy and complementarity will be quantified once within-class variance is restored in the scaled study (§6).

Summary. We show that the EdgeTPU sysfs, Mali GPU sysfs, and `getThermalHeadroom()` API each independently expose on-device ML workload activity on the Tensor G2 platform. Five workload classifier families including strictly linear models each achieve ≥ 0.983; the zero-permission API alone achieves 0.8 ± 0.16, 3.2× the 25% chance baseline, confirming that Adversary 2 achieves viable classification using only installed-app privileges in the four-class setting.

5 Related Work

ML model side-channels. DeepTheft [3] recovers DNN layer structure from RAPL energy readings on x86 MLaaS platforms. Kim et al. [8] extract DNN architecture variants on ARM Mali GPUs via the TVM runtime profiler, an application-layer API that does not access kernel sysfs hardware counters. TPUXtract [9] targets a Coral Dev Board EdgeTPU using physical EM probes and requires close physical proximity. Zhang et al. [18] fingerprint DNN architectures via thermal event interrupts on x86, targeting model IP rather than on-device ML workload identity. Our work targets a different question (ML workload-type classification, not model reconstruction) and different interfaces (sysfs hardware counters and a public API, not physical probes or software profilers).

Mobile thermal and power side-channels. Miedl et al. [10] read CPU thermal zones via `/sys/class/thermal/` on a Samsung Galaxy S5 (Exynos 5422, Android 5.0) and a Sony Xperia Z5 (Snapdragon 810, Android 7.0) to label which of 11 foreground applications is running, reporting up to 90% per-time-step labelling accuracy under laboratory conditions that drops below 20% on real-world traces. Both devices predate dedicated on-device ML accelerators; their work covers CPU zones only and explicitly excludes the GPU. We report 0.8 ± 0.16 on a Tensor G2 EdgeTPU device using a single 0.5 Hz public API channel rather than multi-zone privileged sysfs, and 1.0 ± 0.0 across three independently sufficient hardware interfaces (§4.4); our threat model targets the EdgeTPU, the Mali GPU sysfs, and the zero-permission `PowerManager` thermal API absent from theirs. Hot Pixels [14] demonstrate DVFS timing side-channels on ARM SoC and desktop GPU platforms via JavaScript; they observe frequency transitions indirectly through timing, not by reading sysfs registers. Oberhuber et al. [12] systematically analyse 137 Android sensors across nine devices — including the Pixel 6a, 7a, and 9 — for power-related physical signals via the Android Sensor Framework.

Their study is explicitly scoped to Android Sensor Framework: the EdgeTPU sysfs at `/sys/devices/platform/1ce00000.janeiro/`, the Mali GPU sysfs at `/sys/devices/platform/28000000.mali/`, and the `getThermalHeadroom()` thermal management API are all outside the Android Sensor Framework and appear in none of the 137 enumerated channels.

To the best of our knowledge, neither the EdgeTPU nor Mali GPU sysfs interfaces, nor the `getThermalHeadroom()` *API, have previously been studied as side-channels for on-device ML workload inference.* Wang et al. [17] classify iOS app activity using multi-channel time-series with InceptionTime on an iPhone, achieving 87.3% AUC; their channels are iOS OS-level resource statistics APIs (CPU, memory, network, storage), not thermal APIs or hardware performance counters.

6 Limitations and Future Work

Sample size. n=60 with four classes bounds this to a single-device feasibility study. Logistic Regression confirms linear separability; a permutation test ($p < 0.001$) and nested cross-validation together rule out a small-sample artefact. A scaled evaluation with ≈600 sessions will sample additional representative apps within each of the four workload categories to verify that the four-class signal persists under broader within-class sampling. All cross-validation results above are reported under a single fixed seed (`random_state=42`, §3.2); multi-seed validation is part of the scaled evaluation.

Temporal interpolation. The 150-timestep normalisation in §3.2 applies the same channel-wise linear interpolation to every session regardless of class, operating on each of the 227 channels independently; the interpolation step itself therefore cannot encode the class label. We nonetheless do not repeat the cross-validation on raw, non-interpolated sessions; this evaluation, together with the simple-feature baseline isolating session duration or `inference_count` rate (§4.1), is deferred to the scaled study.

Single device. All results derive from one Pixel 7 (Tensor G2, Android 16); cross-device and cross-SoC-revision generalisation is untested. We will validate across Tensor G3 (Pixel 8) and Tensor G4 (Pixel 9) to assess whether the structure persists across EdgeTPU generations, and across all Android 11+ devices where `getThermalHeadroom()` is available without declared permissions.

Controlled conditions. We collected all sessions in a single indoor environment without concurrent foreground workloads, and without active ambient thermal control or external temperature logging. The 0.8 ± 0.16 API-only result is therefore an upper bound; real-world ambient temperature variation and concurrent app activity will reduce this figure, which we will quantify under adversarial conditions in follow-up evaluation.

Deployment fidelity. We do not implement a complete end-to-end attack app. We reserve cross-adversary training-to-deployment transfer for a follow-up system study.

Countermeasures. Throttling `getThermalHeadroom()` below 0.5 Hz targets the zero-permission path directly; Naghibijouybari et al. [11] report that aggressive rate-limiting on analogous GPU performance APIs degrades attack accuracy gradually rather

than eliminating it, with 40% precision retained under their tested limits. The EdgeTPU, Mali GPU, and thermal-zone sysfs groups must be restricted simultaneously to defend the privileged path, since each independently sustains 1.0 ± 0.0 in §4.4; partial restriction leaves the remaining paths intact. Single-channel mitigations fail when multiple channels independently carry sufficient signal [17]; whether coarsening applied jointly across all three sysfs groups can be reconciled with Android's thermal management policy is open.

7 Conclusion

We demonstrate that `getThermalHeadroom()`, accessible without declared permissions since API Level 30 (2020), and the sysfs hardware interfaces of the Google Tensor G2 EdgeTPU and Mali GPU, constitute previously unstudied side-channels for inferring on-device ML inference workloads.

We show that five workload classifier model families — Logistic Regression, Linear SVM, k-NN, Random Forest, and InceptionTime — each achieve ≥ 0.983 five-fold cross-validation accuracy on 227 channels (Adversary 1: root-privileged sysfs access), and that the zero-permission `getThermalHeadroom()` stream alone achieves 0.8 ± 0.16 (Adversary 2: installed-app privileges only; $3.2\times$ the chance baseline, $p < 0.01$). Three hardware interfaces — EdgeTPU sysfs, Mali GPU sysfs, and thermal zone sysfs — each independently achieve 1.0 ± 0.0; this confirms that the attack surface is both broad and structurally robust across physically separate kernel subsystems. These results establish that `getThermalHeadroom()` and the EdgeTPU and Mali GPU sysfs interfaces require the same security assessment as the motion sensors and power APIs already studied in the literature.

Acknowledgments

We acknowledge use of the CX3 HPC cluster of the Imperial College Research Computing Service (https://doi.org/10.14469/hpc/2232).

References

[1] Android Developers. 2020. *PowerManager.getThermalHeadroom(int) — Android Developers.* https://developer.android.com/reference/android/os/PowerManager#getThermalHeadroom(int)

[2] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. InceptionTime: Finding AlexNet for Time Series Classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962. doi:10.1007/s10618-020-00710-y

[3] Yansong Gao, Huming Qiu, Zhi Zhang, Binghui Wang, Hua Ma, Alsharif Abuadbba, Minhui Xue, Anmin Fu, and Surya Nepal. 2024. DeepTheft: Stealing DNN Model Architectures through Power Side Channel. In *Proc. IEEE Symp. Security and Privacy (S&P).* IEEE, San Francisco, CA, USA, 3311–3326. doi:10.1109/SP54263.2024.00250

[4] Google. 2022. *Pixel 7 and Pixel 7 Pro: The Next Generation of Pixel.* https://blog.google/products/pixel/pixel-7-pixel-7-pro/

[5] Google. 2022. *Pixel Feature Drop: December 2022.* https://blog.google/products/pixel/feature-drop-december-2022/

[6] Google AOSP. 2022. *EdgeTPU Janeiro Kernel Module.* https://android.googlesource.com/kernel/google-modules/edgetpu/janeiro/

[7] GSMArena. 2022. *Google Pixel 7 Specifications.* https://www.gsmarena.com/google_pixel_7-11903.php

[8] Dong Hyub Kim, Jonah O'Brien Weiss, and Sandip Kundu. 2024. Extracting DNN Architectures via Runtime Profiling on Mobile GPUs. *IEEE J. Emerg. Sel. Topics Circuits Syst.* 14, 4 (Dec. 2024), 620–633. doi:10.1109/JETCAS.2024.3488597

[9] Ashley Kurian, Anuj Dubey, Ferhat Yaman, and Aydin Aysu. 2024. TPUXtract: An Exhaustive Hyperparameter Extraction Framework. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025, 1 (Dec. 2024), 78–103. doi:10.46586/tches.v2025.i1.78-103

[10] Philipp Miedl, Rehan Ahmed, and Lothar Thiele. 2021. We Know What You're Doing! Application Detection Using Thermal Data. *Leibniz Trans. Embedded Syst. (LITES)* 7, 1 (2021), 02:1–02:28. doi:10.4230/LITES.7.1.2

[11] Hoda Naghibijouybari, Ajaya Neupane, Zhiyun Qian, and Nael Abu-Ghazaleh. 2018. Rendered Insecure: GPU Side Channel Attacks are Practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18).* ACM, Toronto, ON, Canada, 2139–2153. doi:10.1145/3243734.3243831

[12] Mathias Oberhuber, Martin Unterguggenberger, Lukas Maar, Andreas Kogler, and Stefan Mangard. 2025. Power-Related Side-Channel Attacks using the Android Sensor Framework. In *Proc. Network and Distributed System Security Symp. (NDSS).* Internet Society. doi:10.14722/ndss.2025.240092

[13] Markus Ojala and Gemma C. Garriga. 2010. Permutation Tests for Studying Classifier Performance. *Journal of Machine Learning Research* 11 (2010), 1833–1863.

[14] Hritvik Taneja, Jason Kim, Jie Jeff Xu, Stephan van Schaik, Daniel Genkin, and Yuval Yarom. 2023. Hot Pixels: Frequency, Power, and Temperature Attacks on GPUs and Arm SoCs. In *32nd USENIX Security Symposium (USENIX Security 23).* USENIX Association, Anaheim, CA, 6275–6292.

[15] Andrius Vabalas, Emma Gowen, Ellen Poliakoff, and Alexander J. Casson. 2019. Machine Learning Algorithm Validation with a Limited Sample Size. *PLOS ONE* 14, 11 (2019), e0224365. doi:10.1371/journal.pone.0224365

[16] Vladimir Vapnik and Akshay Vashist. 2009. A new learning paradigm: Learning using privileged information. *Neural Networks* 22, 5–6 (2009), 544–557. doi:10.1016/j.neunet.2009.06.042

[17] Zihao Wang, Jiale Guan, XiaoFeng Wang, Wenhao Wang, Luyi Xing, and Fares Alharbi. 2023. The Danger of Minimum Exposures: Understanding Cross-App Information Leaks on iOS through Multi-Side-Channel Learning. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23).* ACM, Copenhagen, Denmark, 281–295. doi:10.1145/3576915.3616655

[18] Xin Zhang, Zhi Zhang, Qingni Shen, Wenhao Wang, Yansong Gao, Zhuoxi Yang, and Zhonghai Wu. 2024. ThermalScope: A Practical Interrupt Side Channel Attack Based on Thermal Event Interrupts. In *Proc. Design Automation Conf. (DAC).* ACM, San Francisco, CA, USA. doi:10.1145/3649329.3656525