# Duty-Cycled LoRA Training on One Consumer Phone: Step-Count Feasibility and Thermal Stops during Reload

**Emil Shirokikh**
Independent researcher

## Abstract

A few optimizer steps before a phone reaches its thermal limit do not establish local adapter training. We report pre-specified thermal stop-rule studies of CPU inference and LoRA training on one USB-connected Galaxy S25, together with the design assumptions that failed on the way. Ten of 16 short-response inference schedules satisfy a stability criterion, while continuous generation stops. Continuous training stops after 8 updates, and a fixed schedule of 20-second bursts and 90-second pauses after 84. A gate that reused the series' cool-start condition released bursts sooner than the fixed schedule did and stops after 23. A fixed schedule of 20-second bursts and 200-second pauses reached 232, 236 and 225 updates of 400 planned in three repeats with one seed, all above a pre-set 160-step threshold without status 3 during training. The reload check outside that rule reached status 3 in one of three runs, under a gate that differed from the first run's. Across all five reload checks in the series, two stopped. Training reached the step count at one schedule; useful personalization, energy and a sustained deployment loop were not measured.

## 1 Question and context

Low-rank adaptation reduces trainable parameter count (Hu et al., 2022), but a small update does not remove a phone's thermal and lifecycle constraints. Prior work measures language models on phones, including speed, memory, energy and thermal effects during inference (Laskaridis et al., 2024). MobileFineTuner runs fine-tuning natively on phones and schedules around battery state, without reporting temperature (Geng et al., 2025). The QVAC fork used here reports LoRA fine-tuning times on an Adreno 830 GPU without thermal data (Subash SN et al., 2025). Thermo-FL uses temperature as a control signal for federated LoRA, with Jetson boards as its physical clients (Shrestha et al., 2026), production federated learning trains only when phones are idle and charging (Bonawitz et al., 2019), and on-device training is studied under severe memory limits (Lin et al., 2022). The closest work postdates our runs. A recent preprint trains LoRA adapters with a phone NPU and CPU and paces them thermally, pausing at Android status 2 or a 40 °C battery and resuming at status 1 or below with the battery at or below 38 °C (Zheng et al., 2026). On a OnePlus 15 its unpaced fixed load reached status 3 after about 18 minutes, and its paced run trained 313,592 tokens in 3.22 hours, with status 2 starting all 81 pauses.

Our narrower contribution is a thermal stop-rule characterization of LoRA training on one phone's CPU, with an account of the designs that failed before one schedule passed. The fork's announcement reports only GPU timings. The protocol specified a CPU build with the same target and compile settings as the CPU-only llama.cpp build already in use. No further reason is recorded, and GPU or NPU training was not attempted in this series. A limited search found no other phone measurement of thermal stops during CPU-only LoRA training, which does not prove absence.

We use "negative result" for the outcome of a study that ran correctly (Peltonen et al., 2023). Three assumptions failed in that sense: that a fixed 18.2% duty would cool enough to reach 160 updates, that the series' cool-start condition could serve as a per-burst admission rule, and that a reload after training was a mechanism check rather than a thermal load. In each case the runner did what the protocol registered: the trainer's stopped state was read after all 144 pause signals, and every gated burst started under the registered condition. Implementation faults also occurred. Section 5 lists them and counts none as a result.

| | |
|---|---|
| Device | Galaxy S25 (SM-S931U, SM8750, Android 15), CPU only, six threads, Power saving off; a host computer drives pauses (SIGSTOP/SIGCONT) and sampling over adb |
| Training | QVAC fork of llama.cpp (commit 7bb2161, base b10549), Qwen2.5-0.5B-Instruct (Qwen Team, 2024) F32 (self-converted, SHA-256 e63f6b64), rank 8, alpha 16, attention q/k/v/o, AdamW $10^{-4}$, context 256, seed 20260925, 20 epochs, 400 planned steps; fixed synthetic text of 3,264 tokens, 23 datapoints; 13.9 MB checkpoint at every step; screen off |
| Inference | llama.cpp b10991 llama-server, Qwen2.5-1.5B-Instruct Q4_K_M (public GGUF, revision 91cad511), context 4096, temperature 0, 8 round-robin prompts; screen on |
| Continuous | llama-bench, 128 tokens per iteration; screen off |
| Reload | llama-completion b10991, 8 fixed prompts with and without the adapter, greedy, 64 tokens, consecutive runs with no duty cycle |
| Stop | status $\geq 3$ or battery $\geq 42\,^{\circ}$C (continuous cell: status $\geq 4$ or battery $\geq 42\,^{\circ}$C) |
| Unrecorded | ambient temperature, phone surface, USB source |

Table 1: Experimental setup.

## 2 Platform and measurement

Table 1 gives the setup. Thermal status is Android's reported severity (Android Developers, 2026): 0 none, 1 light throttling, 2 moderate, 3 severe. On this device it equals the skin-sensor status in all 3,952 samples of the series' 28 monitor files, so the criteria measure skin-sensor bins more than compute throttling; the application-processor (AP) sensor's own status stays 0 near 68 °C. Read post hoc, the status behaves as a quantized skin reading with hysteresis. The first sample after a rise reads 38.0–39.9 °C for status 1, 39.8–41.8 for status 2 and 42.0–43.0 for status 3, and the first sample after a fall to status 0 reads 34.3–35.9. Status 0 therefore means below 36 °C on cooling but as high as 37.9 on heating. The hardware abstraction layer (HAL) names this sensor SKIN. We did not measure the phone's surface, and the reading moves 5.0–8.9 °C within one burst cycle, so we report it as the platform's reading and not as a surface measurement. Status is sampled every ten seconds and is not a calibrated temperature or an energy meter. RSS is process memory. The phone was USB-connected to a host computer, and the supply did not cover peak draw: battery current reached $-3.7$ to $-4.4$ A (assuming negative means discharge), and battery level fell by up to 5 points during runs, so we make no energy estimate. Signal-driven pauses mean these experiments do not validate an ordinary background app scheduler.

The fork was chosen after the stock route closed. By source inspection in September 2026, llama.cpp b10991, release b11179 and upstream master had no LoRA training path, and the full-parameter path in b10991 aborted during backward-graph construction after 2.0 seconds with no optimizer step. ExecuTorch v1.5.1 had a Qwen2 LoRA configuration, but its training was driven from host Python, so it was not pursued.

"Pre-specified" means written in dated protocol files committed to a private repository before each run; no external registry was used. The training rule is at least 160 optimizer steps in the window with no status 3 at any training-monitor sample. The value 160 is an operational threshold without a quality basis. The second and third runs were pre-specified with this rule: "replicated" applies only if all three runs reach 160 steps without status 3, and any run reaching status 3 is recorded and the schedule is not repeated hotter. The reload check is a mechanism check reported beside the verdict and outside the rule. The amendment did not settle whether a reload status 3 voids "replicated". A ruling recorded after all three runs counted the second run's reload stop under the no-hotter rule, so a planned 20/150 schedule was not run. "Full-cycle" is a reading applied after these runs and appears nowhere in the protocols.

## 3 Inference envelope

One pass in fixed order crosses reply lengths of 16, 32, 64 and 128 tokens with gaps of 30, 15, 5 and 0 seconds, each cell requesting ten minutes. The criterion, labelled "never throttles" in the protocol, requires status at most 1 in every sample and last-minute decode rate at least 90% of first-minute rate. We call it a stability criterion, since Android defines status 1 as light throttling. Gap is confounded with start temperature and time of day (start skin 30.5–32.0 °C at 30 seconds, 33.8–34.0 °C at 5 and 0 seconds), and ratios rest on few replies (1–2 per measured minute at 30 seconds, 3–4 at 15, 7–11 at 5).

Ten of 16 cells satisfy the criterion in this pass (Table 2), and the highest rate among them is 446 tokens per minute. Eight of 16 cells stay at status 0. Two passing cells reach status 1, including the 446 cell. Frequency caps below hardware maxima appear in 53 of 58 samples of the 446 cell and in 19 and 35 of 59 samples in two status-0 cells, and only six cells have no capped sample. Samples are

| Gap | 16 tok | 32 tok | 64 tok | 128 tok |
|---|---|---|---|---|
| 30 | 33 (0, 0.99) | 66 (0, 1.01) | 129 (0, 0.98) | 248 (0, 1.01) |
| 15 | 64 (0, 1.00) | 125 (0, 1.00) | 241 (0, 1.00) | 446 (1, 0.97) |
| 5 | 177 (0, 1.01) | 334 (1, 0.97) | *591 (1, 0.86)* | *961 (2, 0.76)* |

Table 2: Tokens per minute (maximum status, last/first-minute decode ratio). Italic cells fail the criterion. All four cells at a 0-second gap reach status 3 and stop within 30–61 seconds.

not aligned to decoding and caps may have other sources, while decode retention stays at 0.97 or above in those cells, so this shows that the protocol's label is unsafe more than it shows lost speed. A separate 30-minute continuous cell stops at 355 seconds when battery temperature reaches 42.3 °C. It completes 21 of 23 iterations (one timed out and one was terminated), first reaches status 3 at 243 seconds, and falls from 58.3 to about 37.6 tokens per second.

## 4 Training schedules

Continuous training completes eight steps (3.56 seconds per step, peak RSS 2.93 GiB) before status 3 stops it about 30 seconds in. A fixed schedule of 20 seconds on and 90 off (duty 18.2%) reaches 84 updates before stopping at 1,674 seconds, below 160. A later gated schedule resumes when status and battery appear cool, with effective duty of 66.8, 39.5, 32.7 and 24.4% over its first four cycles, and it stops at 255.9 seconds with 23 updates when the skin sensor reaches status 3. The recorded reading is a gate design error: the skin sensor that fired the stop was not in the gate, so the gate allowed a hotter duty than the fixed schedule. Skin-aware gating is untested.

The next schedule keeps 20-second bursts and fixes pauses at 200 seconds (duty 9.1%) for a 2.5-hour window, extended from the earlier two hours. All three runs reach 160 steps inside two hours, so the extension does not decide the verdict. They reach 232, 236 and 225 updates, 56–59% of the 400 planned, and stop when the window ends (Table 3, Figure 1). The repeats use one seed and produce identical running loss at all 225 shared steps, so they repeat timing and thermal behaviour only. Three of three gives a one-sided 95% lower bound of 37% on this device's pass rate under these conditions. Validation loss on two chunks of the same text falls from 1.43 to 0.19 by epoch 10, which indicates optimizer progress and says nothing of quality. Median step time is 3.40–3.43 seconds and peak RSS 2.92–2.93 GiB.

| | Run 1 | Run 2 | Run 3 |
|---|---|---|---|
| Updates (41 bursts) | 232 | 236 | 225 |
| Minutes to 160 updates | 102.9 | 99.3 | 106.5 |
| Maximum training status | 2 | 2 | 2 |
| Peak skin, AP (°C) | 41.6, 67.9 | 41.0, 67.6 | 41.6, 68.0 |
| Cycles with status $\geq 2$ | 23 | 18 | 28 |
| Start battery, skin (°C) | 33.5, 34.0 | 25.7, 27.8 | 27.6, 29.4 |
| Battery level first/min/last | 98/95/100 | 100/95/97 | 100/95/99 |

Table 3: Repeats of the 20/200 schedule. Run 1 began in the evening after another phone job, and Runs 2 and 3 overnight.

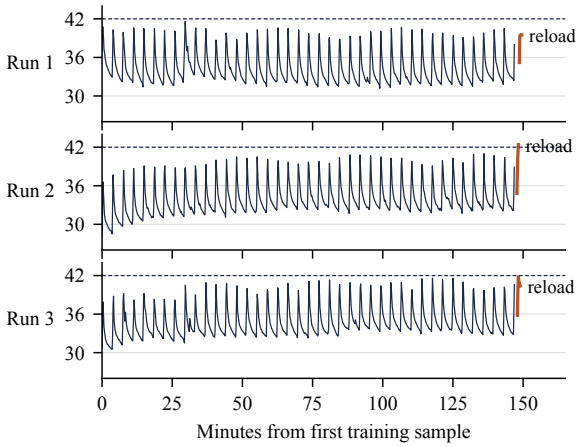



Figure 1: Skin temperature during the three 20/200 runs, with the reload check in the darker segment. The dashed line marks 42 °C. Only Run 2's reload reached status 3.

The training margin is small. First samples at status 3 in the series read 42.0–43.0 °C and the highest skin reading at status 2 is 41.9 °C, so peak skin in Runs 1 and 3 sits about 0.4 °C below the lowest stop reading. Status-2 samples fall almost entirely in pauses after bursts (39 of 40, 28 of 28, 59 of 59), which means peaks lag the bursts. Frequency caps cover 3/80, 2/82 and 7/75 burst samples against 68/779, 67/780 and 82/787 pause samples.

Figure 2 shows the median 20/200 cycle, computed post hoc over 117 cycles. The AP reading reaches about 63 °C in the burst and falls to about 45 within 20 seconds of its end, while skin and battery keep rising after compute has stopped. Skin temperature following chip temperature is expected on phones (Bhat et al., 2019); what matters for an admission rule is the delay. At ten-second sampling, the cycle's peak follows the burst's end by a median of 5.6–7.6 seconds for skin and 14.9–15.6 for the HAL battery sensor, against −1.2 to

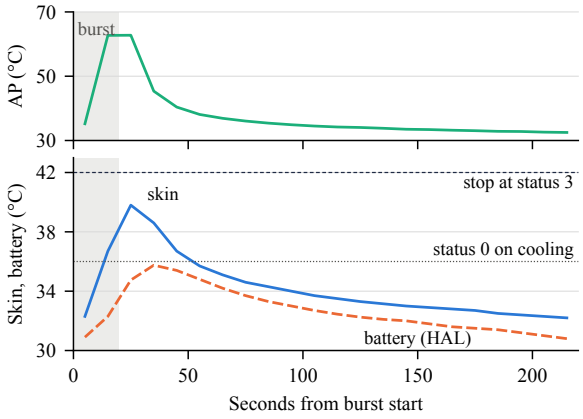



Figure 2: Median 20/200 cycle over bursts 2–40 of Runs 1–3 (117 cycles), in ten-second bins; the shaded span is the burst. The status 0 line is the level at which status returned to 0 on cooling. Post hoc.

| After | Gate | Wait (s) | Skin | Batt. | Peak | Pairs | Max status |
|---|---|---|---|---|---|---|---|
| 20/90 | skin | 213.8 | 34.0 | 33.7 | 43.0 | 2 of 8 | 3 (stop) |
| Gated | strict | 50.2 | 35.3 | 34.6 | 40.0 | 8 of 8 | 2 |
| Run 1 | skin | 91.8 | 33.7 | 32.9 | 39.5 | 8 of 8 | 1 |
| Run 2 | strict | 40.2 | 35.7 | 34.7 | 42.4 | 4 of 8 | 3 (stop) |
| Run 3 | strict | 50.2 | 35.2 | 34.8 | 41.8 | 8 of 8 | 2 |

Table 4: All five reload checks, with gate wait, temperatures (°C) at reload start, and peak skin. The skin gate included a skin target (at most 34 °C); the strict gate had none. The 20/90 runner lacked later reload fixes, is not directly comparable, and had two 300-second timeouts.

0.2 for the AP. Skin falls below 36 °C a median 37.1–46.3 seconds after a burst and below 34 °C after 72.7–97.9 seconds, and it reads 32.0–32.8 at 190 seconds. Burst-start skin approaches a similar level from both sides: the mean of the first and last ten bursts moves from 32.5 to 31.9 in Run 1, from 30.7 to 32.2 in Run 2 and from 32.0 to 33.2 in Run 3.

Two of five reload checks stopped (Table 4). Run 2's reload reached status 3 at 40.4 seconds with skin at 42.4 °C. The gates differed between Run 1 and the later runs, in the direction of the observed order, so with three runs this does not isolate run-to-run variability. A full reload takes only 65–74 seconds, and it runs consecutive completions, the regime that reaches status 3 in 30–61 seconds in the inference sweep.

Each saved adapter holds 336 tensors, of which 192 are active weight factors and 144 are inert bias factors with all-zero second factors. The recorded checker flag is false in all runs because of those bias pairs, although llama.cpp b10991 loaded the tensors. Outputs under greedy decoding differed with the adapter loaded on 8 of 8, 4 of 4 and 8 of 8 completed pairs, and on 4 of 8 in the gated run. No base-versus-base repeat exists, so this shows only that loading the adapter changed output.

## 5 Post-mortem of the designs

Table 5 sets each design beside what was recorded before it ran.

**The 20/90 schedule.** The recorded expectation was about an hour to 160 updates if the cool-down held. Burst-start skin instead rose from 33.7 °C at the second burst to 35.9 at the sixteenth, whereas 90 seconds after a burst the later 20/200 runs read a median of 33.4–34.0, so the shorter cycle held more heat at the same point. Peaks stayed at 40.6–41.7 °C for bursts 2 to 15. One sample then read 43.0, which is 2.7 °C above the sample before it and 1.8 seconds after the sixteenth burst ended. The stop was a single-sample jump after a long run of status 2, and a rule acting on one sample at ten-second spacing cannot separate that from a steady climb.

**The gated schedule.** The gate required status 0 and a battery-service reading below 35 °C before each burst. It reused the condition that started every earlier cell. The rate estimate drafted before the run modelled battery cooling alone and gave pauses of 95–200 seconds, and it noted that the gate conditioned on the battery while the 20/90 stop came from the skin sensor. Measured pauses were 9.9–61.9 seconds. The first pause passed at its first sample, status held the next two, and status and then the battery held the fourth, so the status term decided most of the waiting. Because status returns to 0 once skin falls below 36 °C, that value was the gate's working limit on skin, and gate-pass samples after a burst read 33.5–35.4. The battery field was also coarse: in the gated run it changed in 8 of 24 sampling intervals, against all 24 for the HAL battery sensor, and the two differed by up to 4.3 °C.

The fifth burst shows the consequence. At its end the reading was status 1 and 39.9 °C. Ten seconds later, with the AP down from 66.7 to 55.8 °C and no compute running, it was status 3 and 42.1 °C. Figure 3 sets every cycle's peak against the reading before its burst. None of the 119 cycles that started below 34 °C reached 42, the highest reading being 41.6, and two of the 17 that

| Design | Recorded before the run | Outcome | Reading |
|---|---|---|---|
| Stock runtime, full-parameter | No LoRA path by source inspection; a failure is the result | Aborted in backward-graph construction after 2.0 s; no step | The pinned runtime could not train adapters on the phone |
| Fork, continuous | 20 steps under the stop rule | 8 steps; status 3 about 30 s in; 3.56 s per step | Twenty continuous steps did not fit inside the thermal limit |
| Fixed 20/90 | About an hour to 160 updates "if the cool-down holds" | 84 updates; status 3 at 27.9 min | A 90-second pause did not hold burst-start skin down |
| Gated: status 0, battery below 35 °C | Pauses of 95–200 s; 160 updates in 62–120 min; nothing on status 3 | Pauses of 9.9–61.9 s; duty 24.4–66.8%; 23 updates; status 3 at 255.9 s | The reused start condition shortened the pauses |
| Continuous on efficiency cores | An efficiency cluster exists | Clusters cpu0–5 and cpu6–7, neither for efficiency; dropped before any run | A premise about the chip was false |
| Fixed 20/200 | 205–215 updates; 160 at 110–114 min; nothing on status 3 | 225–236 updates; 160 at 99.3–106.5 min; no status 3 in training | The rate estimate was low; the peak margin is 0.4 °C |
| Reload after training | A mechanism check outside the verdict | Two of five reach status 3 and stop | A reload is a thermal load of its own |
| Fixed 20/150 | Runs only if Runs 2 and 3 pass | Not run, by a ruling on Run 2's reload stop | The rule did not cover reload stops |

Table 5: Each design beside what was recorded before it ran. Expectations come from dated protocol text; the protocols labelled rate figures as estimates and not as predictions of the verdict.

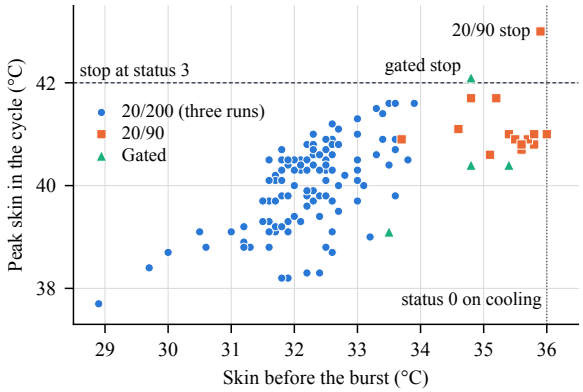



Figure 3: Peak skin in each training cycle against the last reading before its burst (136 cycles). First bursts and the truncated final cycle of Runs 1–3 are excluded. The vertical line is the level below which status returned to 0 on cooling. Post hoc, and confounded with schedule.

started at or above 34 did. We chose that split after seeing the data, and it is confounded with schedule, since all 17 come from the 20/90 and gated runs. It is a description and not a validated limit.

This was an error in the design and not in its implementation: the runner applied the registered condition at every burst, and the condition was the wrong one to reuse. It does not show that gating fails. A gate that reads skin with a margin for the burst's rise has not been run. The controller in the recent preprint (Zheng et al., 2026) also differs, since it interrupts work when status reaches 2. On our phone that rule would not have caught the final rise, because status moved from 1 to 3 across one sample after the burst had ended, but we cannot say whether this holds at other sampling intervals or on other phones.

**Core pinning.** The next proposal was continuous training pinned to efficiency cores. A read-only query of the CPU topology found clusters cpu0–5 (up to 3.53 GHz) and cpu6–7 (up to 4.47 GHz) and no efficiency cluster, so the premise was wrong for this chip and the design was dropped before any run. No result of any kind exists for it.

**The 20/200 schedule.** Its rate estimate, taken from the 20/90 run, was 205–215 updates and 160 updates at 110–114 minutes. The runs reached 225–236 updates and 99.3–106.5 minutes, at 5.49–5.76 updates per burst against 5.25. Nothing was predicted about status 3. Updates per burst dip in isolated bursts (2 in Run 1's ninth; 3, 4 and 1 in Run 3's fourth, eighth and tenth), and those runs total 218,482 and 109,027 major page faults against 112 in Run 2, which has no burst below 5. Run 1's ninth and Run 3's fourth bursts coincide with the largest falls in the trainer's resident memory between pauses, 194 and 178 MiB. With three runs this is an association only, and steps are assigned to bursts by arrival time at the host. It is consistent with the system reclaiming pages of a stopped trainer, but we did not trace the mechanism.

**The reload check.** The protocols treated the reload as a mechanism check. The replicates were registered with identical settings, and the night rules adopted the strict gate because it matched the condition under which Run 1's training actually started. The reload gate changed with it, although Run 1's reload had passed through a skin target that the strict gate lacks. The omission recorded for the gated schedule therefore returned in the reloads of Runs 2 and 3, which started at 35.7 and 35.2 °C and peaked at 42.4 and 41.8, against 33.7 and 39.5 in Run 1. Five checks cannot isolate the gate from other variation, and the gated run's reload started at 35.3 °C and peaked at 40.0.

**The replication rule.** The rule did not say whether a reload stop voids "replicated". The precondition for the hotter 20/150 schedule read only the two training verdicts, so the automated queue would have scheduled that run. The queue was paused, and the ruling then counted Run 2's reload stop.

**Faults not counted as results.** The series also had implementation faults, which we keep apart from the results above. The first attempt on the stock runtime passed an invalid flag and exited in 0.03 seconds. The 20/90 runner's reload printed no tensor count, and two of its completions stalled until a 300-second timeout; the cause is not in the saved data, and later runners added verbose loader output and per-run time limits. One launch of Run 1 was refused before any phone contact, because the launching shell's text matched the runner's busy check. None of these changed a training verdict, although the 20/90 row of Table 4 is not directly comparable.

## 6 Interpretation

At the tested schedule, small F32 LoRA training reached a pre-set step count on this phone. The word "feasible" carries no more than that: the threshold has no quality basis, no run finished its planned training, and peak temperature sits close to the stop level. A reload check that runs consecutive completions, placed after training, can reach the thermal limit even when training does not, so a job that includes reload and generation needs its own admission rule. One observation under a changed gate is thin evidence, however, and it is not a replication failure in any pre-specified sense. Output changes after reload do not show improved personalization.

## 7 Conclusion

Training reached the pre-set step count at one fixed schedule on one phone, while other schedules and two of five reload checks stopped at the thermal limit. The training schedules that stopped failed on assumptions about cooling and admission, not on implementation faults. Quality and energy remain unmeasured. These results support a limited execution claim and show why a local-learning product cannot be inferred from successful optimizer steps alone.

## Limitations

One phone is studied, with one model size for training and one for inference, and two runtimes. The three repeats share one device and one seed and do not represent a population of phones or training seeds. Ambient temperature, phone surface and USB source were not recorded. Ten-second sampling can miss excursions, and the stop rule acts on one sample. All duty-cycled runs wrote a checkpoint at every step, so the thermal share of those writes is unknown. The status thresholds, cycle responses, the 34 °C split and the memory association are post hoc and descriptive. The inference grid is one fixed-order pass with no repeat. USB connection to a host, screen-off execution and signal-driven pauses differ from battery use and a production mobile lifecycle, and the workload is CPU-only and host-orchestrated. Quantization, training targets and task quality are not varied. Electrical energy and learned personalization are unmeasured in these runs. Other duty schedules, larger models and NPU or GPU training cannot inherit this result. The reload comparison confounds runs with a changed gate.

## Ethical considerations

This study used only synthetic text and involved no human participants or personal data. Inference and training models are released under Apache 2.0, and the training fork's announcement states Apache 2.0 for its code. Protocols, monitor records, synthetic text and adapters are not publicly released at the time of writing.

## Funding and competing interests

No external funding. The author develops SlyOS, a commercial product related to this work.

## References


Android Developers. 2026. PowerManager. Android API reference, thermal status constants. Accessed 4 November 2026.

Ganapati Bhat, Suat Gumussoy, and Umit Y. Ogras. 2019. Power and thermal analysis of commercial mobile platforms: Experiments and case studies. In *Design, Automation and Test in Europe Conference and Exhibition*.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards federated learning at scale: System design. In *Proceedings of Machine Learning and Systems*, volume 1.

Jiaxiang Geng, Lunyu Zhao, Yiyi Lu, and Bing Luo. 2025. MobileFineTuner: A mobile-native framework for on-device LLM fine-tuning in real-world embedded AI applications. *Preprint*, arXiv:2512.08211.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Stefanos Laskaridis, Kleomenis Katevas, Lorenzo Minto, and Hamed Haddadi. 2024. MELTing point: Mobile evaluation of language transformers. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*.

Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. 2022. On-device training under 256KB memory. In *Advances in Neural Information Processing Systems*.

Ella Peltonen, Nitinder Mohan, Peter Zdankin, Tanya Shreedhar, Tri Nguyen, Suzan Bayhan, Jon Crowcroft, Jussi Kangasharju, and Daniela Nicklas. 2023. Perspectives on negative research results in pervasive computing. *IEEE Pervasive Computing*, 22(3):63–72.

Qwen Team. 2024. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Shiva Shrestha, Kazi Shaharair Sharif, Zongxing Xie, Jiajing Huang, Anhao Xiang, and Honghui Xu. 2026. Thermo-FL: Thermal-aware robust federated fine-tuning of large language models for edge AI. *Preprint*, arXiv:2608.21172.

Subash SN, Akshay Nambiar, Patrik Lambert, Milan Gritta, Gianfranco Cordella, and Amril Nurman. 2025. An edge-first generalized LLM LoRA fine-tuning framework for heterogeneous GPUs. Hugging Face blog.

Xu Zheng, Wei Cheng, Zhuomin Chen, Dongkuan Xu, Mo Sha, Jingchao Ni, and Dongsheng Luo. 2026. First-order LoRA training with phone NPUs. *Preprint*, arXiv:2610.24816.