What a reply costs in joules, *and where the heat goes*

Published measurements of the energy a phone spends generating text, worked through in joules, and what the phone does when that energy turns to heat.

MITWA LAB, MUMBAI

IN PLAIN WORDS

When a language model writes a reply on the phone, the energy comes out of the battery, and nearly all of it ends up as heat inside the case. This review works through measurements on named phones in joules, the unit of energy, for one read from memory, one token (a word or part of one), one reply, one hour and one full battery. The most careful measurement found puts a token at between about 0.6 and 0.8 joules on an iPhone 14 Pro and a Galaxy S23, so a 135-token reply costs roughly 80 to 100 joules. Most phones are cooled without a fan and slow down as they warm, and one study saw an iPhone 16 Pro's speed fall 41.5% below its peak over 20 back-to-back runs. We found no published energy per token for an iPhone 16.

Memory sets the floor

A language model produces text one token at a time, and for each token the phone reads the model's weights out of main memory (Ehsani et al., 2026). Laskaridis et al. (2024), who measured phones and small computing boards, conclude that running these models on a device is limited mainly by memory, and Huang et al. (2025) find that the decode phase, in which the reply is generated token by token, is bound by memory and takes most of the energy.

Figures that Han et al. (2016) give for a 45 nm chip process show why memory is the expensive part. Reading 32 bits from off-chip DRAM costs 640 pJ (a picojoule is a millionth of a millionth of a joule), against 5 pJ for reading them from on-chip SRAM cache and 0.9 pJ for adding two 32-bit floating-point numbers. Dividing 640 pJ by four bytes gives 160 pJ per byte, so reading a gigabyte once would cost 0.16 J at that rate. That step is our arithmetic and only an illustration, since the 45 nm figures date from 2016 or earlier and no source opened here gives a per-byte cost for the memory in a current phone.

One token, estimated and then measured

Liu et al. (2024) turn figures of this kind into a rule of thumb of 0.1 J per generated token for every billion parameters, citing Han et al. (2016) and an older paper, so that a 7-billion-parameter model spends 0.7 J on each token. The authors did not measure this, and the figure is an estimate built from older hardware numbers.

The most careful measurement we found is by Laskaridis et al. (2024), who opened each phone, bypassed its battery and powered it through a Monsoon power monitor sampling at up to 5 kHz. Running a 3-billion-parameter model stored at 4 bits (Zephyr-3B) through six requests on an iPhone 14 Pro and a Samsung Galaxy S23, they report 0.16, 0.20 and 0.21 mWh per generated token across the phone and app pairs tried. One milliwatt-hour is 3.6 J, so these come to about 0.58 to 0.76 J per token; the conversion is ours and needs no assumption, because both units measure energy. Taking the model as 3 billion parameters, that is roughly 0.19 to 0.25 J per billion, two to two and a half times the rule of thumb. The ratio should be read loosely. Liu et al. illustrate their rule with 8-bit models and count the model alone, whereas Laskaridis et al. measured a 4-bit model on

the whole phone, with its screen on at a fixed 25% brightness.

On a Galaxy S24 Ultra with its screen off, Tummalapalli et al. (2026) report 146 mJ per token for a 4-bit model of 1.5 billion parameters, read from the phone's own battery gauge, whose absolute accuracy they caveat. Guégain and Coignion (2026) placed a current sensor between an iPhone 14 and its battery and converted the discharge into energy using the battery's nominal voltage, an assumption that a monitor reading power directly avoids.

The software can move these numbers a long way. By changing which processor cores did the work, Huang et al. (2025) cut energy by 23% on average against the engine they started from (MNN), across five Android and two iOS devices and four datasets, without slowing generation, and they report savings of 39 to 78% against four other inference engines. Whether storing the weights at lower precision saves energy is less settled. Across 18 configurations, Guégain and Coignion (2026) find that energy does not fall steadily as the bit width falls, and that 4 bits was the most economical setting on both phones they tested. Ehsani et al. (2026), on a single Galaxy S25 Ultra, compared two 4-bit formats and found the choice made almost no difference to energy, while model size and architecture made a large one. Since Ehsani et al. held the bit width fixed, the two findings do not conflict, though each rests on one or two phones.

In our third technical note we argued that a smaller build should spend less energy on each reply and said plainly that we had not measured it on a device. The measurements above show what such a figure looks like, and how much it depends on the phone, the software and the instrument.

One reply, and one hour at full power

The requests in the Laskaridis et al. (2024) power experiment averaged 40 tokens in and 135 tokens out. By our arithmetic, 135 tokens at 0.58 to 0.76 J each come to roughly 78 to 102 J for one reply on those phones. Xiao et al. (2024) report about 20 mAh per round for a 4-bit model of 7 billion parameters running on the processor cores, with 64 tokens in and 128 out. A milliamp-hour measures charge, and turning charge into energy needs a voltage. At 3.87 V, the nominal figure Guégain and Coignion (2026) use for the iPhone 14 battery, 20 mAh comes to about 280 J, though the conversion is only illustrative because the devices in the battery table of Xiao et al. are tablets whose voltages are not given. Ehsani et al. (2026) give whole-task figures for one phone, where a single summarisation run took about 50 J with a model of 0.5 billion parameters and routinely more than 300 J with one of 9 billion.

During the same experiment the iPhone 14 Pro's averaged draw reached as much as 13.8 W, with instantaneous peaks above 18 W, while the Galaxy S23's sustained draw stayed below 8.5 W (Laskaridis et al., 2024). A draw of 13.8 W held for an hour would be about 50 kJ, close to the whole battery on the estimates in the next section. That is an upper case, and the heating described below makes an hour at that draw unlikely, but the arithmetic shows how fast continuous generation at that rate would empty a battery if the phone did not slow down first.

One full battery

No Apple page opened here gives battery capacity in watt-hours, so it has to be estimated. Liu et al. (2024) take a fully charged iPhone to hold about 50 kJ, and Guégain and Coignion (2026) model the iPhone 14 battery as 3.279 Ah at 3.87 V, which multiplies out to about 12.7 Wh, or 45.7 kJ.

Laskaridis et al. (2024) estimate that at their measured rates each phone could serve about 490 to 591 requests of their size on one charge, without allowing for anything else the phone was doing. On an iPhone 16 Pro, Tummalapalli et al. (2026) saw the battery level fall by 5 percentage points over 20 runs of 819 generated tokens each and project about 400 such runs per charge, a coarse projection when the level is read in whole points. Huang et al. (2025) report that 20 sampled conversations used as much as 6 to 25% of the battery, depending on the phone, in less than a quarter of an hour.

The same token on a server

The most direct comparison with a data centre cuts against the phone. Guégain and Coignion (2026) ran 18 model configurations on a Pixel 8, an iPhone 14 and an Nvidia A100 server. On average the phones used 3 times the energy per token of the server when it batched many requests together, though they did better than the same server answering one user at a time, which used 5.4 times the batched figure. The iPhone measurements were noisy, and for some configurations the 95% confidence interval was wider than the value itself. Their life-cycle model spreads the carbon cost of making the phone over a battery assumed to last 1,000 full charge cycles, and under every assumption they varied, a token generated on the phone carried more environmental impact than one from the batched server. Most of the phone's share came from manufacturing, 88 to 90% by the paper's summary and 84 to 87% by its results section.

The result is modelled rather than measured and turns most on the assumed battery life; the paper is a preprint, and its first author's affiliation is the firm Greenspector.

Google reports per request rather than per token. Elsworth et al. (2025) find that the median text request to the Gemini app used 0.24 Wh, about 860 J, counting the active accelerators, the host machines, idle machines held in reserve and data-centre overhead. A narrower approach that counts only the accelerators, and samples more highly utilised machines, gives 0.10 Wh. The paper gives neither the model's size nor the number of tokens in the median request, so it cannot be set beside the phone figures without assumptions no source here supplies.

Every joule ends as heat

Apart from the small share that leaves as light, sound or radio signal, the electrical energy a phone draws ends up as heat inside it, so the budget above is also a heating budget. Phones are mostly cooled passively, with heat managed by the operating system (Laskaridis et al., 2024), and when a chip passes its temperature thresholds its clock frequency is lowered, which lowers throughput (Tummalapalli et al., 2026). Laskaridis et al. (2024) filmed the iPhone 14 Pro with a thermal camera and recorded 47.9 °C after a full conversation, a level they call uncomfortable. On a Xiaomi 14, Xiao et al. (2024) saw the temperature climb from about 34 °C to above 40 °C and settle near 42 °C on longer inputs, and found that a few runs in a row were enough to make the processor throttle.

Laboratories design around this heat. Xiao et al. (2024) rebooted their devices and rested them for 10 minutes between rounds to prevent overheating, Ehsani et al. (2026) imposed 200 seconds of cooling between runs, and Tummalapalli et al. (2026) let each device settle for 10 minutes at 22 °C, give or take 2 °C, before starting. Ordinary use matches neither these rested starts nor the back-to-back runs described below. Xiao et al. (2024) describe the common case on phones as single requests spaced more than 30 seconds apart, but a phone in use may already be warm from other work, and no study here measured that case.

What Apple says the phone does when it is warm

Apple (2026a) designs iPhones for ambient temperatures from 0 to 35 °C. When the inside of the phone goes beyond its normal operating range, the phone tries to bring its temperature down, and Apple lists what a person may notice: charging slows or stops, the screen dims or turns off, and some apps run at lower frame rates or take longer to process. Apps can read the phone's condition through a thermal state with four levels, which Apple (2026b) defines only in words. Nominal is within normal limits, fair is slightly elevated, serious is high, and at critical the system's performance is significantly affected and the phone needs to cool. Neither page says at what temperature, power or duration the phone moves from one level to the next.

Speed once the phone has warmed

Laskaridis et al. (2024) ran 50 requests back to back on the iPhone 14 Pro, three times over, and saw throughput fall from the start, with further drops near the 20th and 32nd requests that they put down, as a hypothesis, to the phone switching power modes. Tummalapalli et al. (2026), the only measurement on an iPhone 16-generation phone that we found, ran a 4-bit model of 1.5 billion parameters on an iPhone 16 Pro 20 times in a row, producing 819 tokens each time with one second between runs, a gap that gave the phone no chance to cool. In the first two runs the phone reached 40.49 tokens a second at best and averaged 36.83. From the third run it reported a warmer state and averaged 24.96, and from the seventeenth it reported its hottest state and held at 23.67 tokens a second, 41.5% below the peak. A Galaxy S24 Ultra in the same study lost about 15% over its 20 runs, with the screen off and settings chosen to keep within the Android power budget.

The study is a preprint from a company, and the iPhone 16 Pro result covers one model, one input and one software stack, as the authors say. Because iOS gives third-party apps no per-component power reading, they could not report energy per token for the iPhone at all. They report three thermal states, named Normal, Warm and Hot, while Apple's documentation lists four under other names, and the paper does not say how one set maps onto the other.

What is not yet known

- Energy per token on any iPhone 16 model. The one study we found that ran a model on an iPhone 16 Pro had no per-component power reading and could not report it.
- The temperatures, power levels or durations at which iOS moves between its four thermal states, and how the three labels in Tummalapalli et al. (2026) correspond to them.

- An Apple-published battery capacity in watt-hours for the iPhone 16 range. The battery figures used here are a round estimate and a modelling value for the iPhone 14.
- Whether the instruments used here (a power monitor with the battery bypassed, a sensor at the battery, the phone's own gauge, logging from inside an app) agree on the same phone and model. No study opened compared them.
- Whether anyone has repeated the 13.8 W and 47.9 °C measurements on an iPhone 14 Pro, or made comparable ones on a newer iPhone. We found no such replication.
- How energy and speed behave in ordinary use, with pauses between replies and a phone that may already be warm. Every study here that states its protocol either ran requests back to back or rested the phone before measuring.
- How much running models wears a battery, measured over months rather than modelled from a rated cycle count, the assumption the life-cycle comparison leans on most.

References

Apple (2026a). If your iPhone or iPad gets too hot or too cold. Apple Support article 118431, published 14 May 2026. https://support.apple.com/en-us/118431

Apple (2026b). ProcessInfo.ThermalState. Apple Developer Documentation, Foundation. Undated page, opened 12 November 2026. https://developer.apple.com/documentation/foundation/processinfo/thermalstate-swift.enum

Ehsani, E., Giamattei, L., Malavolta, I., and Pietrantuono, R. (2026). Sustainability is not linear: Quantifying performance, energy, and privacy trade-offs in on-device intelligence. arXiv:2603.26603. https://arxiv.org/abs/2603.26603

Elsworth, C., Huang, K., Patterson, D., Schneider, I., Sedivy, R., Goodman, S., Townsend, B., Ranganathan, P., Dean, J., Vahdat, A., Gomes, B., and Manyika, J. (2025). Measuring the environmental impact of delivering AI at Google scale. arXiv:2508.15734. https://arxiv.org/abs/2508.15734

Guégain, É., and Coignion, T. (2026). The battery price of edge AI: A study of the environmental impact of LLM inference on mobile devices. arXiv:2609.11940. https://arxiv.org/abs/2609.11940

Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In International Conference on Learning Representations (ICLR 2016). arXiv:1510.00149. https://arxiv.org/abs/1510.00149

Huang, Z., Niu, C., Wang, Z., Xue, J., Zhang, H., Wang, Y., Xin, Z., Jiang, X., Lv, C., Wu, F., and Chen, G. (2025). MNN-AECS: Energy optimization for LLM decoding on mobile devices via adaptive core selection. arXiv:2506.19884. https://arxiv.org/abs/2506.19884

Laskaridis, S., Katevas, K., Minto, L., and Haddadi, H. (2024). MELTing point: Mobile evaluation of language transformers. In Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (MobiCom 2024). arXiv:2403.12844. https://arxiv.org/abs/2403.12844

Liu, Z., Zhao, C., Iandola, F., Lai, C., Tian, Y., Fedorov, I., Xiong, Y., Chang, E., Shi, Y., Krishnamoorthi, R., Lai, L., and Chandra, V. (2024). MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. In Proceedings of the 41st International Conference on Machine Learning (ICML 2024), PMLR 235. arXiv:2402.14905. https://arxiv.org/abs/2402.14905

Tummalapalli, P., Arayakandy, S., Pal, R., and Kundan, K. (2026). LLM inference at the edge: Mobile, NPU, and GPU performance efficiency trade-offs under sustained load. arXiv:2603.23640. https://arxiv.org/abs/2603.23640

Xiao, J., Huang, Q., Chen, X., and Tian, C. (2024). Understanding large language models in your pockets: Performance study on COTS mobile devices. arXiv:2410.03613 (version 5, 9 February 2026). https://arxiv.org/abs/2410.03613