

Ethical AI Tools — CS 193T

We understand that some of you may be taking this class because you're not a big fan of how AI is used commercially, and you'd like to learn more about various ways to use it more morally, ethically, environmentally, or with a special focus on privacy.

For that reason, we've compiled this list of suggestions, resources, and models to use when you'd like your AI practices to align more closely with certain values.

What you're actually changing

Almost everything on this page is some combination of three moves. Knowing which move a tool makes tells you what you get out of using it.

1. Run the model locally instead of in the cloud. When a model runs on your own laptop, your prompts and documents never leave it — nothing is sent to a company, stored on their servers, or retained for training. That's the difference between "I trust this vendor's privacy policy" and "there is nothing to trust, because the data never moved." It matters most for work you aren't free to share: unpublished research, interview transcripts, student records, anything under an NDA. You also stop drawing on a datacenter for every query, and you can work offline. The tradeoff is real, though: models small enough to run locally are less capable than frontier models, and your own battery does the work.

2. Use an open model instead of a closed one. "Open" is a spectrum — some projects release only the weights, others release the training data and code too. The further along that spectrum, the more you can actually check: what the model was trained on, whose work it learned from, where its biases might come from, and whether a result you published can be reproduced by someone else. With a closed model, every one of those questions has the same answer: take the vendor's word for it.

3. Use the smallest model that does the job. Bigger models cost dramatically more energy and water, both to train and to answer each query. Most everyday tasks — summarizing, reformatting, drafting, simple Q&A — don't need a frontier model. So: **use the least powerful, big, or energy-intensive system that adequately does the job.** This is the easiest of the three changes to make, and it's why you'll see so many "mini" models below.

Models that require no technical expertise to use

1. **Apertus Mini** — runs entirely inside your browser tab, so nothing you type is sent anywhere. The lowest-effort way to experience private, local inference: no install, no account.

a. <https://huggingface.co/spaces/swiss-ai/apertus-mini-webgpu>

2. **OLMo** — the openness end of the spectrum. AI2 publishes the weights, the training data, and the training code, so you can inspect what went into it. Reach for this when you need to justify, audit, or reproduce a result rather than just get one.
 - a. <https://playground.allenai.org/>
3. **Granite (IBM)** — an openly licensed model from a major vendor. A middle ground if you want the transparency of open weights without giving up a polished, supported product.
 - a. <https://www.ibm.com/granite/playground>

Models/tools that require technical expertise to use

Most of this section is about move #1: these tools run models on hardware you control, so your prompts and files never leave your machine and no datacenter runs on your behalf. They differ mainly in how much setup they demand and how much control they hand you. In order of increasing difficulty:

1. **LM Studio** — the gentlest way to get local inference working. A point-and-click app for downloading and running open models on your laptop.
 - a. <https://lmstudio.ai>
2. **Jan Desktop** — a local ChatGPT replacement: the chat interface you're already used to, with nothing leaving your computer.
 - a. <https://www.jan.ai>
3. **Ollama** — the same idea from the terminal. Easiest option to script against or build your own tools on top of.
 - a. <https://ollama.com>
4. **OpenJarvis** — a Stanford project optimized for efficiency, so you get more useful work out of the same energy.
 - a. <https://open-jarvis.github.io/OpenJarvis>
5. **MLX-LM** — runs and fine-tunes models locally on Apple silicon, with HuggingFace integration. Fine-tuning a small model on your own data is often both cheaper and greener than repeatedly prompting a large one.
 - a. <https://github.com/ml-explore/mlx-lm>
6. **Llama.cpp** — low-bit quantization shrinks a model so it runs on modest hardware. That cuts the energy each query costs and lowers the hardware barrier for people without a powerful machine.
 - a. <https://github.com/ggml-org/llama.cpp>
7. **HuggingFace transformers** — Python toolkit for running and inspecting models directly. Where you go to study how a model behaves, not just to use it.
 - a. <https://huggingface.co/docs/transformers/index>

8. **Intelligence per Watt** — profiles your local inference and records what it actually costs in energy. Makes an otherwise invisible cost visible, so you can compare options with evidence instead of guesswork.

a. <https://hazyresearch.stanford.edu/intelligence-per-watt>

Within the Stanford ecosystem

If you're staying inside Stanford's licensed tools, move #3 is the one available to you: pick the smallest model that works.

AI Playground mini models — start here by default. These handle most drafting, summarizing, reformatting, and everyday Q&A at a small fraction of the energy cost of a frontier model.

- Gemini 2.5 Flash-Lite
- Claude Haiku 4.5
- GPT 5.6 Luna

AI Playground medium models — step up to these when a mini model has actually failed at the task, rather than as a default.

- Gemini 3.5 Flash
- Claude Sonnet 4.6
- GPT 5.6 Terra

Depending on availability, our teaching team might not have time to answer your questions about these models, so feel free to explore on your own! We'll try to help if possible.