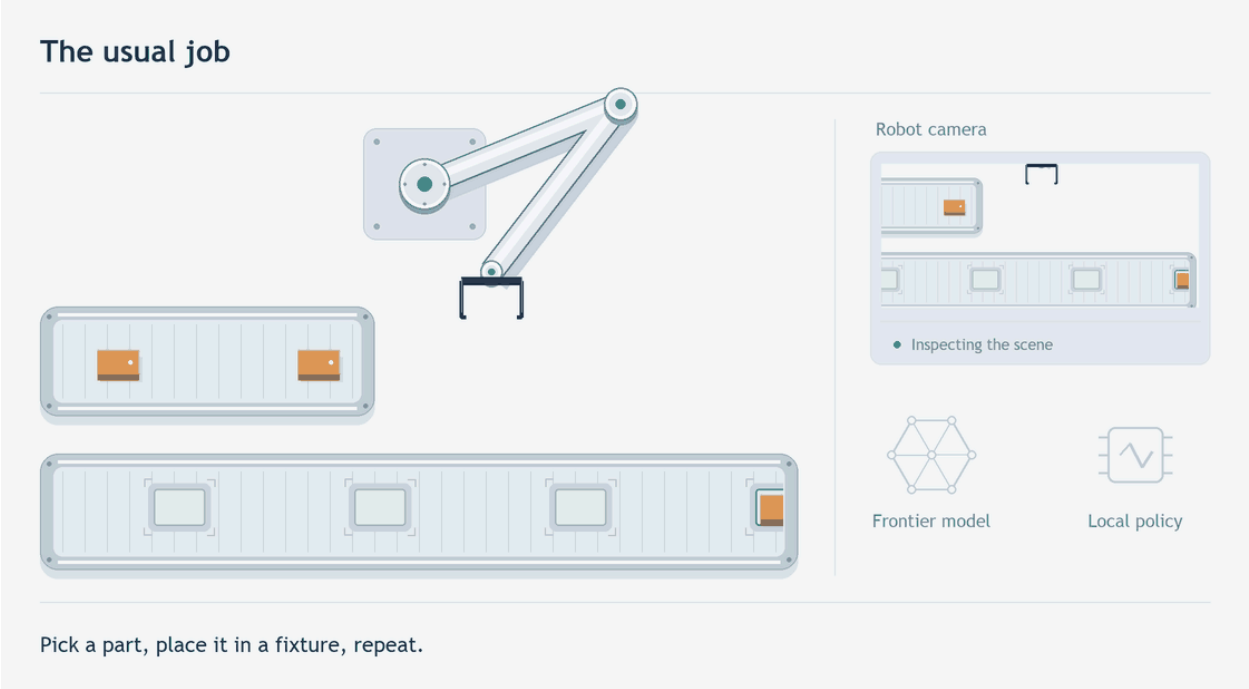
Teach Once at a Premium, Repeat for Pennies: Turning Frontier Intelligence into Fast Robot Skills

Imagine a robot working on an assembly line, placing parts into a fixture. Everything goes smoothly until a part lands on its side and the usual grasp fails. The robot calls on a frontier model for help. The model examines the camera image, tries a different grasp, and adjusts its approach using feedback from the robot. After a few attempts, the part is in place and work resumes.



This is the appeal of bringing frontier models into robotics: they could help a robot work through situations its existing policy cannot handle. Recent demonstrations offer a glimpse of that possibility. Google has shown Gemini 4 Argon using a robotic arm to turn a fallen mug upright. NVIDIA's study of GPT-6 Astra evaluates manipulation across a range of simulated pick-and-place tasks, without additional robot-policy fine-tuning. [1, 2]

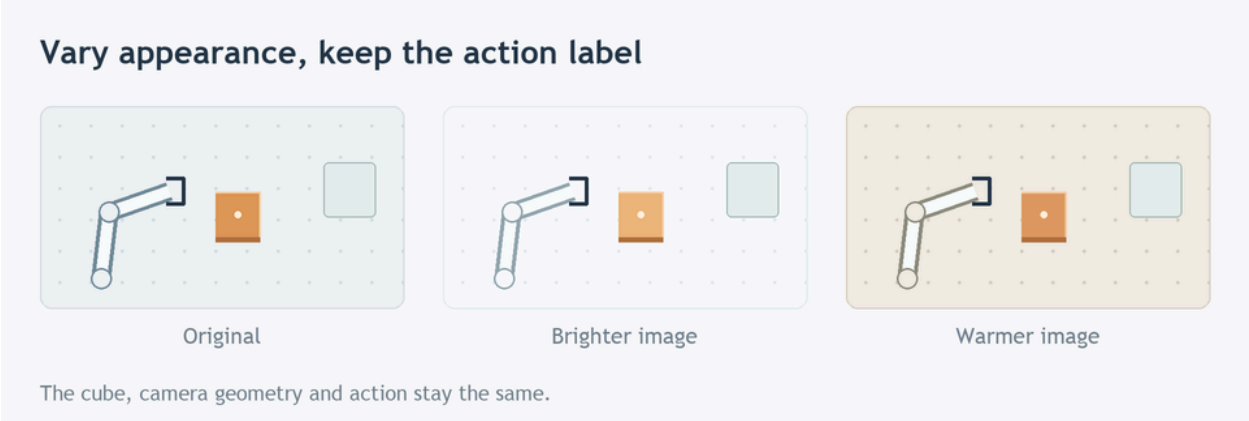
But this help costs both money and time. In NVIDIA's evaluation, direct-policy inference averaged about 19 seconds per call, where the more successful rollouts had an estimated token cost of $36.40 per successful episode.[2] In a step-by-step control setup, the robot may pause between movements while the model decides what to do next. That may be acceptable for an occasional difficult recovery. It becomes harder to justify when the same problem keeps coming back.

Going back to our example, suppose another part lands on its side a few minutes later. If the robot has learned nothing from the first recovery, it needs help again. We pay for another round of reasoning and wait while the model works through a familiar problem. To this end when the frontier model helps the robot recover, we could save a record of what worked. As these successful examples accumulate, we could use them to train a small policy that runs locally. The next time a part lands on its side, the robot could use that skill without waiting for another round of model reasoning. The effort spent solving the problem would then help with many future recoveries.

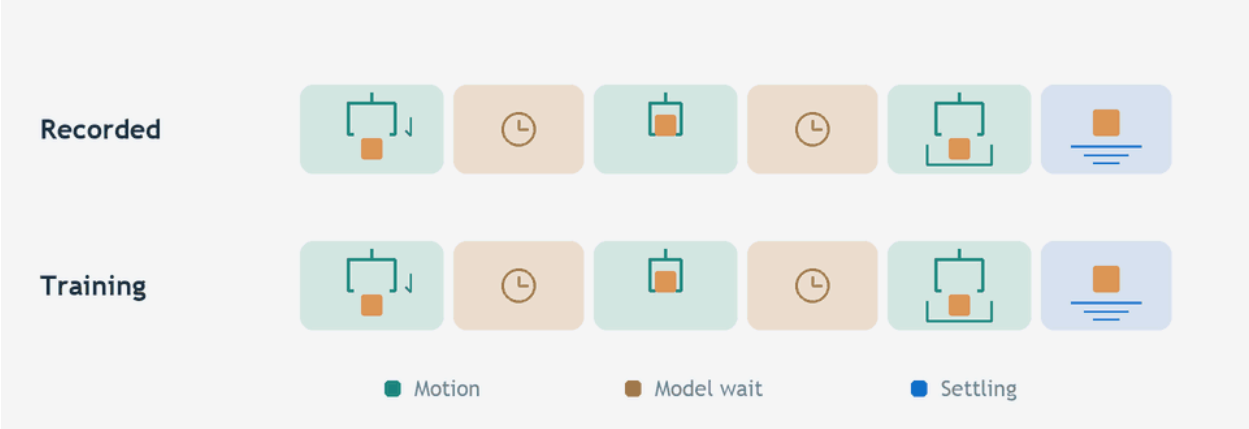
Turning a recovery into a skill

The assembly-line example gives us a useful place to start. The robot already knows how to place a correctly oriented part into the fixture. What it lacks is a reliable way to get a sideways part into that position. We can focus the teaching effort on that specific behavior: reposition the part so the existing placement policy can continue.

To teach it, we would record the camera observations, robot state, and actions executed during model-assisted attempts. Verified recovery sequences would become demonstrations for a compact policy. We can expand this dataset with a few practical engineering techniques: augmenting the camera images to vary their appearance, resetting the simulator to slightly different object poses, and collecting additional recovery attempts. Image augmentation should preserve the corresponding action labels, while new rollouts would need their own success checks.



We also need to separate the recovery itself from time spent waiting for the model. The robot may sit idle for several seconds while the frontier model chooses its next action. For training, we would exclude intervals caused solely by that inference delay, provided the robot and scene remain unchanged. Pauses that serve a physical purpose, such as allowing an object to settle, would stay.



A recurrent behavior-cloning model, such as a BC-RNN, is one possible starting point for learning from these demonstrations. It learns to choose actions from the current observation and recent history. Once trained, a compact policy could generate actions locally with much shorter inference delays. By learning the recovery without the unnecessary waiting, it could complete the same behavior faster and at a lower inference cost.

There is already work showing how model-generated experience can support this kind of learning. Scaling Up and Distilling Down uses a language model to guide robot planners, generate manipulation trajectories, and train a visuomotor policy from the resulting data.[3] Our interest is in directing that process toward failures that recur during normal operation. Those failures tell us which skills are missing and where learning a new behavior might save the most effort.

Collecting the right demonstrations

A practical way to build this dataset is to let it grow during normal operation. When a failure occurs, the robot gets help, completes the recovery, and goes back to work. We save the verified recovery. The next time a similar failure occurs, we save that recovery too. Over time, the assistance the robot already needs could provide the examples for training, with fewer extra calls made solely to collect data.

We would group these examples by the behavior they teach. Once a recurring failure has enough successful recoveries across a useful range of starting conditions, we could train a policy between runs and test it on cases it has not seen. The number of demonstrations alone would not tell us whether the skill is ready. The robot would keep using assistance until the learned recovery has passed those checks.

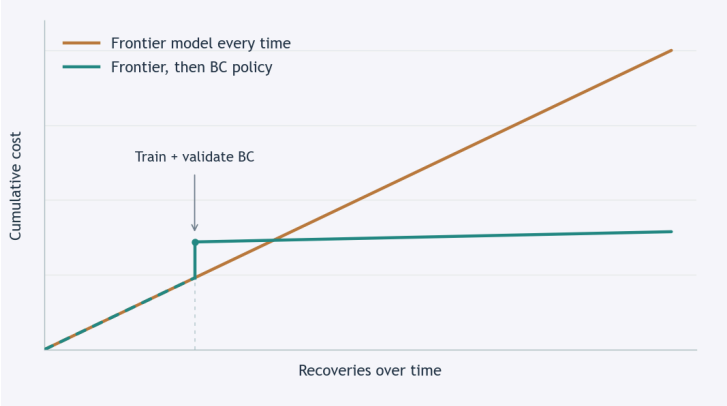


That gives the frontier model a more focused job. It can spend time working through variations the smaller policy has not mastered. A recovery that occurs several times a day would be a natural priority; an unusual failure that never returns may not justify a separate skill.

The payoff from repeated use

Teaching concentrates the expensive work at the beginning. We spend time and computation collecting demonstrations, training the recovery policy, and checking that it works reliably. Once that skill is in place, the robot can reuse it whenever a familiar failure occurs. Each successful local recovery avoids another round of frontier-model reasoning.

The benefit is both financial and practical. A compact policy uses less computation, and the robot no longer has to wait through the teacher's deliberation between movements. By excluding unnecessary inference pauses from the demonstrations, we give the policy a recovery sequence it can execute without reproducing those delays.



For a failure that occurs several times a day, these savings accumulate. The teaching cost is spread across more recoveries, while each reuse saves another model call and another interruption. The investment pays for itself once those accumulated savings exceed the cost of collecting demonstrations, training, and validation. The measure that matters is reliable task completion with less time and money spent per job.

This is the idea behind "teach once at a premium, repeat for pennies." The first recovery takes effort to work through, but that effort leaves behind a reusable skill. When another part lands on its side tomorrow, the robot can use what it has learned and get back to work.

References

[1] Google. Gemini 4 Argon for Robotics.
https://developers.googleblog.com/en/gemini-4-argon-for-robotics/

[2] NVIDIA Seattle Robotics Lab. Astranomy: An Empirical Study of GPT-6 Astra as a Native Robot Policy. https://research.nvidia.com/labs/srl/projects/astranomy/

[3] Huy Ha, Pete Florence, and Shuran Song. Scaling Up and Distilling Down: Language-Guided Robot Skill Acquisition. CoRL 2023.
https://www.cs.columbia.edu/~huy/scalingup/