Query-Efficient DAgger for Distilling Expensive Teacher Policies: Motivation, Survey, and a Confidence-Margin Filter

Pannous
Independent Research

September 8, 2026

Abstract

We study the idea of distilling a slow, expensive but capable "teacher" policy — in the motivating case, a Python policy written or driven by a large language model — into a fast neural-network student via DAgger-style on-policy imitation [Ross et al., 2011], only when the network is actually needed for speed or generality, gated by a held-out promotion test. The core distillation mechanism is not new: standard DAgger already solves the compounding-error problem that plain behavior cloning suffers from, and LLM4Teach [Zhou et al., 2024] already demonstrates almost exactly this teacher-to-student shape empirically, using an LLM as the teacher for a lightweight RL student. What has not, to our knowledge, been directly addressed is the practical cost problem this motivates: full DAgger requires querying the expert on every state the learner visits, every round, which is cheap against a scripted heuristic but potentially prohibitive against a real LLM call. We build and test a query-efficient variant of DAgger that queries the expert only on the fraction of visited states where the current student is least confident, and evaluate it on an existing, real DAgger pipeline (LunarLander, using a hand-written heuristic as a stand-in "expensive" expert whose calls we count). Across three seeds, confidence- margin filtering cuts expert queries by 55–58% while matching or slightly exceeding vanilla DAgger's final reward, within the noise band DAgger itself shows round to round. We report this honestly as a real, reproducible query-efficiency result that has not yet been validated against an actual LLM teacher, which we identify as the necessary next step.

1 Introduction

A recurring pattern in agent design is to use a slow but capable "teacher" — a hand-written heuristic, or increasingly, a large language model reasoning step by step about a task — to control an agent well enough to be useful, and then to compile that behavior into a fast neural network for deployment, invoking the network instead of the expensive teacher wherever it is confident enough to reproduce the teacher's decisions. Plain behavior cloning is well known to fail at this because small errors compound: the student drifts into states the teacher never demonstrated, and has no data for how to recover [Ross et al., 2011]. DAgger fixes this by having the student roll itself out and querying the teacher for the correct action at every state it actually visits, then retraining on the growing aggregated dataset.

This mechanism, applied to a hand-written or LLM-authored Python policy as the DAgger expert, is the "language-model-driven policy with teacher extraction to network if necessary" idea we set out to study. Section 2 surveys the literature and shows the core mechanism and even the specific teacher-to-student shape are already established. Section 4 identifies and tests the concrete, still-open problem this motivates: full per-round expert queries are cheap against a scripted heuristic

but likely expensive against a real LLM, and no existing system we found addresses this cost directly for a DAgger-style loop.

2 Related Work

Imitation learning. DAgger [Ross et al., 2011] is the foundational result here: querying the expert on the learner's own visited states, round after round, provably corrects the covariate-shift problem that plain behavior cloning suffers from. DART [Laskey et al., 2017] achieves similar robustness more cheaply by injecting noise into the expert's own demonstrations rather than iteratively re-querying and retraining; HG-DAgger [Kelly et al., 2018] adapts the idea to a human-in-the-loop, gated setting. Neither directly addresses expert-query cost when the expert is an expensive model rather than a human or a cheap function.

Classical policy distillation. Policy Distillation [Rusu et al., 2015] and Actor-Mimic [Parisotto et al., 2015] compress one or several expensive RL policies into a smaller network via supervised distillation on the teacher's action distribution, in the general tradition of knowledge distillation [Hinton et al., 2015]; these typically distill from a policy that is expensive at *inference* time, not one that is expensive to *query* for a training label, which is the specific cost DAgger against an LLM introduces.

LLMs as controllers. A substantial recent literature uses large language models directly as, or to construct, controllers for sequential decision tasks: LLMs as zero-shot planners [Huang et al., 2022], SayCan's grounding of LLM proposals in affordance models [Ahn et al., 2022], Voyager's LLM-driven, code-based skill library in Minecraft [Wang et al., 2023], Eureka's LLM-authored and evolved reward functions [Ma et al., 2023], and Code as Policies' direct LLM emission of control-policy code [Liang et al., 2023] — the last of these is the clearest precedent for treating an LLM-authored Python program as a first-class policy object, exactly the role of the "expert" we consider.

Directly on point: LLM-teacher-to-network distillation. LLM4Teach [Zhou et al., 2024] is, to our knowledge, the closest existing work to the idea we set out to study: an LLM acts as a teacher providing guidance to a lightweight student RL policy, which is distilled from it and then further fine-tuned with environment reward, eventually surpassing the teacher on MiniGrid and Habitat tasks. It uses a guided policy-gradient style distillation rather than DAgger's specific on-policy relabeling loop — plausibly because querying an LLM at every visited state, every DAgger round, is expensive enough that a lighter guidance signal is preferable. This is exactly the gap we take up: LLM4Teach already shows an LLM-teacher- to-network pipeline works, but does not test, and to our knowledge no other paper tests, whether full DAgger against a real LLM teacher can be made tractable, or by how much a targeted query-reduction strategy would help. More recent agent-distillation work [Reinforced Distillation, 2025] distills full multi-step LLM-agent trajectories into smaller student agents, a related but distinct (multi-step, tool-using) setting from single-step control distillation.

3 The Transplorer Distillation Pipeline

The system we build on (`harness/dagger.py` in the Transplorer project) implements standard DAgger exactly as described above: round zero is plain behavior cloning; each later round rolls the

current network out, relabels every visited state with the expert's action, and retrains on the aggregated dataset. A held-out promotion gate (`harness/policy_runner.evaluate_for_promotion`) requires a distilled policy to clear a reward threshold on both its training-time evaluation batch and a disjoint held-out batch before it is trusted. On LunarLander-v3, using a hand-written heuristic policy as the expert, plain behavior cloning collapses (training reward -150.5, held-out reward -179.8, both below the 200-point promotion threshold, correctly refused), while DAgger repairs this over four rounds (round zero 148.9 $\rightarrow$ round three peak 265.3 $\rightarrow$ round four 251.9, against an expert baseline of 267.0), passing an automated regression test that requires DAgger to beat behavior cloning by more than 50 reward. `harness/coroutine_policy.py` provides the generic plumbing needed to run any blocking controller, including an LLM-driven one, as the DAgger expert, but as of this writing no experiment in the project has actually exercised an LLM as the expert — every recorded result uses the hand-written heuristic. We are explicit that any claim about LLM-authored policies in this paper describes the design's motivating, currently untested half, not a completed result.

4 Our Contribution: Confidence-Margin Query Filtering

4.1 Motivation

Vanilla DAgger queries the expert on *every* state the student visits, every round. Against a hand-written heuristic this is essentially free; the real LunarLander run above used 12,883 expert queries in total. Against a real LLM call, each of those queries could cost real latency and money, and we conjecture this cost — not an algorithmic limitation of DAgger itself — is the practical reason LLM4Teach and similar systems use a lighter guidance signal instead of full per-round DAgger relabeling. If query count can be substantially reduced without sacrificing final policy quality, full DAgger against a real LLM teacher becomes far more plausible.

4.2 Method

We implement `probes/dagger_query_efficient.py`, purely additive and requiring no change to `harness/dagger.py` or any existing test. Round zero (bootstrap behavior cloning) is identical to vanilla DAgger, for a fair comparison. In every subsequent round, we compute the current network's softmax top-1-minus-top-2 confidence margin at each visited state, and send only the least-confident 30% of states to the expert for relabeling; the remaining, confidently-handled states are dropped from that round's labeling entirely rather than kept with the student's own (possibly wrong) label, so the change is purely about reducing expert queries, not about introducing self-training noise.

4.3 Experiment

We use the identical LunarLander-v3 setup, the identical heuristic expert (whose calls we count as the resource being minimized, standing in for an expensive LLM call), and the identical held-out evaluation seed as the existing vanilla-DAgger probe, across three independent seeds.

4.4 Honest discussion

Confidence-margin filtering cuts expert queries by a consistent 55–58% across all three seeds, while final reward is a toss-up against vanilla DAgger — better in two of three seeds, slightly worse in one — and in every case within the same noise band that vanilla DAgger itself shows from round to round. We read this as a genuine, reproducible result specifically about query efficiency, not

Seed	Vanilla queries	Vanilla reward	Query-efficient queries	QE reward	Query savings
0	12,883	251.9	5,798	257.2	55.0%
1	13,087	262.5	5,529	268.1	57.8%
2	13,027	247.5	5,656	241.5	56.6%

Table 1: Real, measured results on LunarLander-v3. Expert (held-out) baseline reward is 267.0 across all runs. Both variants clear the held-out promotion gate (200.0 threshold) in every seed.

about reward improvement: roughly half the expert-query cost for comparable final policy quality. This is exactly the trade-off that matters when the expert is a slow or metered call rather than a free local function.

The central limitation is that this has only been tested against a hand-written heuristic expert, not an actual LLM. We have not measured whether a real LLM's per-state action recommendations are stable and consistent enough under this filtering scheme, nor accounted for the additional latency of a live model call within the training loop, nor established whether the specific 30% query fraction generalizes to a setting where individual queries are far more expensive and a much smaller fraction might be preferable. We present the filtering *mechanism* as validated and worth trying against a real LLM teacher next, not as an end-to-end validated LLM-distillation result.

Incidental finding. While reproducing the existing test suite around this pipeline, we found that `probes/test_promotion_gate.py`'s regression test for rejecting a collapsed distilled network (`test_collapsed_distilled_net_is_refused`) currently fails against the checked-in LunarLander network artifact — the gate now reports that artifact as promotable. This is a pre-existing issue unrelated to and untouched by the work in this paper; we flag it here for the record rather than fix it, since it was out of scope for this contribution.

5 Conclusion

Distilling an expensive teacher policy into a fast network via DAgger, and gating promotion with a held-out test, is not a new mechanism, and an almost identical LLM-teacher-to-student shape already exists in the literature (LLM4Teach). The genuinely open, practically important question this motivates — can full per-round DAgger expert queries be made cheap enough to run against a real LLM — has, to our knowledge, not been directly tested elsewhere. Our confidence-margin filtering mechanism cuts expert queries by roughly half with no measurable cost to final policy quality on a real, reproducible LunarLander benchmark, using a heuristic stand-in for the expensive expert. Testing this mechanism against an actual LLM-driven expert, and studying how query fraction should scale with the real per-query cost, is the natural and, we think, tractable next step.

Code Availability

The DAgger pipeline, promotion gate, and the query-efficient probe described in Section 4 are available at `https://github.com/pannous/transplorer`.

References

M. Ahn, A. Brohan, N. Brown, et al. Do as I can, not as I say: Grounding language in robotic affordances. arXiv:2204.01691, 2022.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. arXiv:1503.02531, 2015.

W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *ICML*, 2022.

M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. HG-DAgger: Interactive imitation learning with human experts. arXiv:1810.02890, 2018.

M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. DART: Noise injection for robust imitation learning. In *CoRL*, 2017. arXiv:1703.09327.

J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *ICRA*, 2023. arXiv:2209.07753.

Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. arXiv:2310.12931, 2023.

E. Parisotto, J. L. Ba, and R. Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. arXiv:1511.06342, 2015.

From correction to mastery: Reinforced distillation of large language model agents. arXiv:2509.14257, 2025.

S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, et al. Policy distillation. arXiv:1511.06295, 2015.

G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. Voyager: An open-ended embodied agent with large language models. arXiv:2305.16291, 2023.

Z. Zhou, B. Hu, C. Zhao, P. Zhang, and B. Liu. Large language model as a policy teacher for training reinforcement learning agents. In *IJCAI*, 2024. arXiv:2311.13373.