

Pioneer Agent: Continual Improvement of Small Language Models in Production

Dhruv Atreja Julia White Nikhil Nayak Kelton Zhang Henrijs Princis
George Hurn-Maloney Ash Lewis Urchade Zaratiana
Fastino Labs

Abstract

Small language models are attractive in production because they are inexpensive, fast, and easy to specialize, but specialization remains a difficult loop. The hard decisions surround training: acquiring and curating supervision, diagnosing failures, selecting a training strategy, and preventing a targeted fix from breaking behavior that already works. We present Pioneer Agent, a closed-loop system that automates this lifecycle. In cold-start mode, the agent turns a task specification into a trained model by searching jointly over data composition, hyperparameters, and learning strategy. In production mode, it converts judged failures into a failure taxonomy and corrective curriculum, retrains, and retains an update only when it passes regression gates. We introduce AdaptFT-Bench, seven staged scenarios that combine fixable noise with harmful supervision at increasing poison rates. Across eight cold-start benchmarks, Pioneer Agent improves over the base checkpoint on every task. Across all seven AdaptFT-Bench scenarios, it improves or preserves the deployed checkpoint while indiscriminate retraining degrades, with final gaps up to 43 points. In two production-style case studies, the same loop raises intent-classification pass rate from 84.9% to 99.3% and entity F1 from 0.345 to 0.810. These results show that deployment failures can be converted into effective training curricula without manual intervention between iterations.

1 Introduction

Frontier language models are strong general-purpose systems (Brown et al., 2020; DeepSeek-AI et al., 2025; Yang et al., 2025), but many production tasks favor smaller models that are cheaper to serve, faster at inference, and specialized for a concrete job such as classification, extraction, summarization, or structured generation (Dettmers

et al., 2023; Fu et al., 2025). Getting such a model to work well is not a single training call. Teams must decide what data to collect, which failures are worth learning from, how to construct supervision, which training recipe to use, and whether a candidate update silently regresses previously correct behavior.

Existing systems automate pieces of this process. AutoML optimizes configurations over pre-defined spaces (Thornton et al., 2013; Feurer et al., 2015); data-centric methods improve labels and training sets (Northcutt et al., 2021; Wang et al., 2023a); prompt optimizers tune the interface to fixed weights (Khatab et al., 2024; Yuksekgonul et al., 2024); and ML-engineering agents execute substantial experimental workflows (Chan et al., 2025; Jiang et al., 2025; Li et al., 2026; Rank et al., 2026). They do not center the production adaptation problem: start from a model’s own failures, determine which observations contain useful supervision, construct a corrective curriculum, retrain, and verify that the update does not break prior behavior.

We study this complete loop. Pioneer Agent operates in two modes. *Cold-start mode* begins with only a task description and autonomously acquires data, constructs a fixed evaluation split, trains candidates, diagnoses errors, and revises the pipeline. *Production mode* begins with an existing model and judged inference traces. The agent clusters failures, separates fixable errors from corrupted supervision, builds a targeted curriculum with hard negatives and replay data, and accepts a candidate only when it fixes the target slice under explicit regression constraints.

Our contributions are:

- **A complete autonomous adaptation system.** Pioneer Agent searches jointly over dataset composition, hyperparameters, and supervision strategy, then closes the loop with

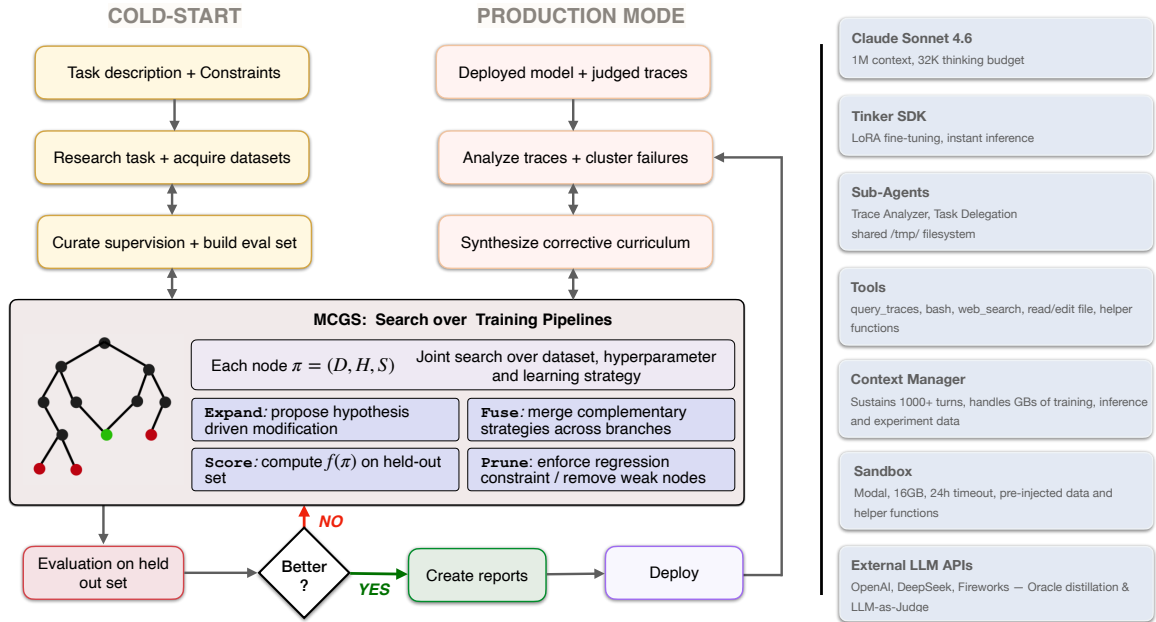


Figure 1: **Pioneer Agent architecture.** An orchestrator LLM drives a LangGraph state machine coordinating task analysis, data curation, training, and evaluation. Cold-start mode builds a model from a task specification; production mode repairs a deployed model from judged traces. Both search over complete training pipelines $\pi = (D, H, S)$ and close the loop through failure diagnosis and regression-gated deployment.

failure diagnosis and regression-gated checkpoint retention.

- **A benchmark for learning from mixed-quality failures.** AdaptFT-Bench evaluates the full failure $\rightarrow$ curriculum $\rightarrow$ retrain $\rightarrow$ verify loop across seven staged scenarios with controlled, increasing corruption.
- **Evidence across cold-start and production settings.** Eight cold-start tasks test autonomous task acquisition; seven staged scenarios test repeated adaptation under poisoned feedback; and two production-style case studies test failure-taxonomy construction, rollback, infrastructure recovery, and threshold selection.

2 Pioneer Agent

2.1 Architecture and state machine

Pioneer Agent is a LangGraph state machine orchestrated by Claude Sonnet 4.6 (Anthropic, 2026) with a 32K-token thinking budget and a 1M-token context window. It executes inside isolated Modal sandboxes with 16 GB host memory and a 24-hour timeout. Decoder models (Qwen3-8B and Llama 3.2-3B (Grattafiori et al., 2024)) are adapted with LoRA (Hu et al., 2022) through the Tinker SDK; GLiNER2 encoder models support LoRA

and full fine-tuning (Zaratiya et al., 2025). The state machine coordinates task analysis, dataset construction, training, inference, evaluation, failure diagnosis, and checkpoint promotion as explicit states rather than a single monolithic prompt.

The architecture is hierarchical (Appendix A). The main agent retains the experimental state for up to 500 production turns or 1,500 cold-start turns. A Trace Analyzer sub-agent performs data-heavy SQL analysis with a much larger output allowance, while independent sub-agents can build datasets, run analyses, and prepare evaluation artifacts in parallel. A Context Manager compacts long trajectories while preserving decisions, evaluation results, and dataset lineage. A persistent data-curation.md ledger records dataset versions, composition ratios, quality-control decisions, and per-iteration results so the state survives context compaction and can be audited later. Figure 1 shows the shared control loop.

2.2 Tool interface and operational modes

Cold-start mode exposes web research, shell execution, file editing, training, inference, and evaluation helpers. Production mode adds query_traces, which accepts SQL plus an optional shell pipeline, and a dedicated trace-analysis tool. Large query results can therefore be filtered,

aggregated, sampled, and written to disk without entering the main agent’s context; even when the inference database contains tens of thousands of rows, the orchestrator can reason over compact summaries and representative cases while the full dataset remains available to the training pipeline.

In **cold-start mode**, the input is a task specification and constraints. The agent classifies the task, investigates available datasets, fixes train/evaluation splits before training, selects a model family and metric, constructs supervision, launches candidate jobs, and uses the fixed evaluation split for iterative model selection. In **production mode**, the input is a deployed model plus judged inference traces. The agent queries the trace store, builds a failure taxonomy, confirms suspected weaknesses with live probes, recovers the parent model’s training lineage, constructs corrective supervision, and evaluates each candidate against both the failure slice and previously correct behavior.

2.3 Joint search over data, optimization, and supervision

A training pipeline is $\pi = (D, H, S)$, where D specifies the mixture of gold examples, hard negatives, contrastive pairs, and replay data; H specifies the model, LoRA rank, learning rate, batch size, epochs, and system prompt; and S specifies the supervision format, teacher, and evaluation method. Executing a pipeline produces a score $f(\pi)$ on the fixed evaluation split. Production mode additionally tracks $r(\pi; \mathcal{R})$, the number of correct examples that regress on a fixed set $\mathcal{R}$, and imposes a regression budget ϵ :

$$\pi^* = \arg \max_{\pi} f(\pi), \quad \text{subject to } r(\pi; \mathcal{R}) \leq \epsilon. \quad (1)$$

We use a regression budget of $\epsilon = 2$ in all production-mode experiments. The search space is heterogeneous and non-differentiable: data composition, optimization, and supervision interact, and every score requires an actual train-and-evaluate execution. Pioneer Agent therefore records a directed graph of completed pipelines where each node stores $(\pi_i, f(\pi_i))$ and each edge records the causal intervention that produced a child, such as adding hard negatives, changing the teacher, lowering the learning rate, or switching supervision format. A UCT-like selector (Kocsis and Szepesvári, 2006) (Equation 4) balances exploration and exploitation. Stagnating branches

evolve through trajectory-aware mutation or fuse complementary discoveries from the top-scoring branches. This procedure is an instance of Monte Carlo Graph Search (MCGS) (Leurent and Miallard, 2020; Czech et al., 2021); Appendix B gives the full algorithm.

2.4 Data curation, sizing, and iteration policy

Every curriculum is assembled from three slices: gold examples, hard negatives, and, for an already fine-tuned parent, a 10–20% replay buffer. The agent applies five data-quality controls (Appendix A.3): paired gold and boundary-case negatives, label balancing, context-length matching, entity diversification for named entity recognition (NER), and chain-of-thought annotation for generation tasks. It also requires multiple surface forms per label and prefers compact datasets: typically 100–200 examples for classification or NER and 500–3,000 for generation. Dataset growth is treated as a hypothesis, not an automatic improvement; if validation performance falls, the agent rolls back the dataset and configuration.

The shared iteration policy changes intervention scale with model maturity (Appendix A.4). Below 0.80, the agent treats the failure as a dataset problem and rebuilds coverage. Between 0.80 and 0.95, it holds data fixed and searches optimization choices. Above 0.95, it adds only two or three examples per remaining failure pattern. Any candidate that exceeds the regression budget is rolled back immediately. This produces a broad-to-surgical trajectory: early parallel exploration establishes a working pipeline, later iterations target specific error modes, and a candidate is promoted only after evaluation and regression checks.

2.5 Failure diagnosis and structural safeguards

Production mode partitions judged traces into passing and failing sets, then clusters failures by observable pathology, input characteristics, and likely cause. Each cluster is labeled as fixable by training or external to the model, and targeted probes verify that the weakness is systematic before data is synthesized. Corrected failures populate the gold slice; confusion pairs become hard negatives; prior examples supply replay. Precision failures increase the hard-negative share, while recall failures increase gold coverage. The agent can also change the training method, model scale, sys-

tem prompt, or inference threshold when diagnosis shows that data alone is not the bottleneck.

Four safeguards constrain autonomous updates: balanced curricula, parallel evaluation of competing configurations, rollback-first iteration, and a cross-checkpoint regression gate. The latter evaluates a candidate on both the current failure set and the previous checkpoint’s evaluation set, preventing a new fix from erasing earlier gains. Confidence calibration uses historical per-label accuracy to surface overconfident errors for review, while model-lineage tracking makes every accepted checkpoint reproducible from its dataset and configuration. Together these mechanisms turn adaptation into a ratchet: each promoted model must preserve validated behavior while incorporating the newly repaired slice.

3 AdaptFT-Bench

AdaptFT-Bench asks one operational question: **given a deployed model and a stream of observed failures, can an agent fix them without breaking what already works?** The primary benchmark contains seven synthetic-log scenarios derived from GSM8K (two model families), ARC-Challenge, TriviaQA, HumanEval, XSum, and SAMSum (Cobbe et al., 2021; Clark et al., 2018; Joshi et al., 2017; Chen et al., 2021; Narayan et al., 2018; Gliwa et al., 2019). Two additional case studies use public CLINC150 and CoNLL-2003 data served through the production pipeline (Larson et al., 2019; Tjong Kim Sang and De Meulder, 2003).

Each synthetic scenario contains three sequential stages with poison rates of 15%, 25%, and 40%. A stage contributes approximately 500 inferences, split 70/30 into training and fixed evaluation rows. Every checkpoint is scored on the union of stage test splits. Perturbations include fixable surface noise and harmful semantic corruption; the agent receives traces, not the latent perturbation labels. The realistic naive baseline trains once per stage on every incoming example, performs no poison filtering, and never rolls back. Approximately 30% of noisy failures remain unreviewed; among poisoned examples, roughly half receive a correction that follows the corrupted input. A controlled GSM8K experiment separately provides oracle-corrected labels to every strategy, isolating data selection from annotation quality.

4 Experiments

Protocols. Cold-start runs receive only a task name and must acquire data, construct a split, train, and iterate. Production runs receive a deployed model and staged judged traces, then diagnose failures, construct corrective curricula, train, and verify candidate updates.

4.1 Cold-start task acquisition

Pioneer Agent improves over the base checkpoint on all eight tasks (Table 1). On ARC-Challenge with Llama 3.2-3B, the agent discovers chain-of-thought supervision, validation-data expansion, and cross-branch fusion, raising accuracy from 5.3% to 72.6%. On GSM8K it reaches 43.7% from an approximately 8% base, while on TriviaQA it reaches 48.6% after diagnosing the remaining errors as knowledge limits of a 3B model.

The same controller succeeds in high-baseline and non-reasoning settings. Qwen3-8B rises from 91.7% to 93.3% on ARC-Challenge; HumanEval reaches 92.7% pass@1 in four iterations using curated MBPP transfer examples; and SMS Spam reaches F1 0.997 through three surgical rounds adding only 55 targeted examples. Across tasks, the first iteration captures only 40–70% of the final improvement; later gains come from diagnosing why the initial pipeline failed and changing data, optimization, or supervision accordingly.

4.2 Production adaptation under corrupted feedback

Across all seven scenarios, Pioneer Agent improves or preserves the deployed checkpoint while naive retraining degrades (Table 2, Figure 2). The largest final-stage gaps are TriviaQA (+43.0 points), XSum (+36.6 ROUGE-2), and ARC-Challenge (+22.6 points). The deployed adaptive curve remains monotone or flat because the system evaluates candidate updates and rolls back regressions instead of allowing them to become production checkpoints.

The mechanism is semantic triage. The naive baseline absorbs false premises, label swaps, and prompt injections as supervision. Pioneer Agent excludes those examples while converting surface corruption into contrastive training pairs. In the matched-stage GSM8K/Qwen3-8B trajectory, the policies start at 75.9% and 71.6%; after 15%, 25%, and 40% poison stages, Pioneer Agent reaches 81.2% while naive retraining falls to 64.7%. The

Benchmark	Model	Base	Instruct	Pioneer	Δ	Iters
ARC-Challenge	Llama 3.2-3B	5.3	78.6	72.6$\pm$2.5	+67.3	11
GSM8K	Llama 3.2-3B	$\sim$ 8	77.7	43.7$\pm$2.7	+35.7	10
TriviaQA	Llama 3.2-3B	$\sim$ 0	–	48.6$\pm$6.2	+48.6	9
ARC-Challenge	Qwen3-8B	91.7	–	93.3$\pm$1.4	+1.6	13
HumanEval	Qwen3-8B	71.3	84.8	92.7$\pm$3.9	+21.4	4
XSum	Qwen3-8B	R-2 6.9 $\pm$ 0.9	–	17.9$\pm$1.4	+11.0	13
SAMSum	Qwen3-8B	R-2 13.4 $\pm$ 1.4	–	25.4$\pm$1.7	+12.0	11
SMS Spam (F1 $\times$ 100)	GLiNER2-base	15.9 $\pm$ 12.3	–	99.7$\pm$0.6	+83.8	10

Table 1: **Cold-start results.** Pioneer Agent receives only a task name. Base and Pioneer are evaluated on the same fixed evaluation split; Instruct is the published instruction-tuned checkpoint in the same model family where available. R-2 denotes ROUGE-2.

Benchmark	Model	Metric	Pioneer Deployed	Naive Final	Final Gap
GSM8K	Qwen3-8B	Acc.	81.2$\pm$2.1	64.7 $\pm$ 2.6	+16.5 $\pm$ 3.3
GSM8K	Llama 3.2-3B	Acc.	34.8$\pm$2.6	14.5 $\pm$ 1.9	+20.3 $\pm$ 3.2
ARC-Challenge	Llama 3.2-3B	Acc.	78.4$\pm$2.1	55.8 $\pm$ 2.5	+22.6 $\pm$ 3.3
TriviaQA	Llama 3.2-3B	Acc.	76.1$\pm$2.2	33.1 $\pm$ 2.4	+43.0 $\pm$ 3.2
HumanEval	Qwen3-8B	pass@1	90.3$\pm$2.3	72.5 $\pm$ 3.5	+17.8 $\pm$ 4.1
XSum	Qwen3-8B	R-2	43.1$\pm$1.5	6.5 $\pm$ 0.8	+36.6 $\pm$ 1.7
SAMSum	Qwen3-8B	R-2	41.4$\pm$1.6	16.8 $\pm$ 1.4	+24.6 $\pm$ 2.2

Table 2: **Production results across AdaptFT-Bench.** Pioneer reports the checkpoint deployed after the final stage, including retention of an earlier checkpoint when a candidate is rolled back; Naive reports its stage-3 retrained checkpoint.

Training strategy	Overall Acc.	Clean-split Acc.
Pioneer (filtered)	72.2	86.2
Clean-only oracle	71.2	82.5
Random augmentation	68.5	81.7
Naive retraining	64.3	75.2

Table 3: **Controlled GSM8K/Qwen3-8B comparison** with oracle-corrected labels on 572 held-out examples. Pioneer Agent outperforms clean-only training, random augmentation, and naive retraining.

16.5-point gap is the accumulated effect of filtering, targeted augmentation, and rollback rather than a single favorable snapshot.

Controlled attribution. Table 3 gives every strategy oracle-corrected labels, holding model family and evaluation data fixed. Pioneer Agent reaches 72.2% overall accuracy, outperforming clean-only training (71.2%), random augmentation (68.5%), and naive retraining (64.3%). On the clean regression split it reaches 86.2%, compared with 82.5% for clean-only and 75.2% for naive retraining. Selective exposure to fixable noise therefore improves robustness while preserving previously correct behavior.

4.3 Production-style case studies

Intent classification. A GLiNER2-base model serving a 30-class CLINC150 subset passes 84.9%

of 2,999 judged inferences, leaving 453 failures. Pioneer Agent identifies confusion pairs, counter-intuitive label semantics, and surface-form overlap, then synthesizes 1,254 targeted examples with hard negatives. Full fine-tuning fixes 450 of 453 failures (99.3 $\pm$ 0.8%) with one regression on the passing set. A second attempt falls to 98.5%, and the gate correctly retains the first checkpoint. Figure 4 shows the resulting taxonomy.

Named entity recognition. On 3,020 CoNLL-2003 inferences, GLiNER2-base begins at entity F1 0.345 with a 6:1 false-positive-to-false-negative ratio. The agent diagnoses precision collapse, constructs a hard-negative curriculum, recovers from failed LoRA inference by switching to full fine-tuning, scales from GLiNER2-base to GLiNER2-large, and tunes the inference threshold to $t = 0.99$. The resulting model reaches entity F1 0.810 ± 0.012 and a $70.0 \pm 1.8\%$ sample pass rate, up from 9.2%. Figure 5 traces the full sequence of interventions.

5 Analysis and Discussion

The system composes strategies rather than replaying a fixed recipe. No run was given a benchmark-specific recipe specifying which training techniques to combine. From outcome

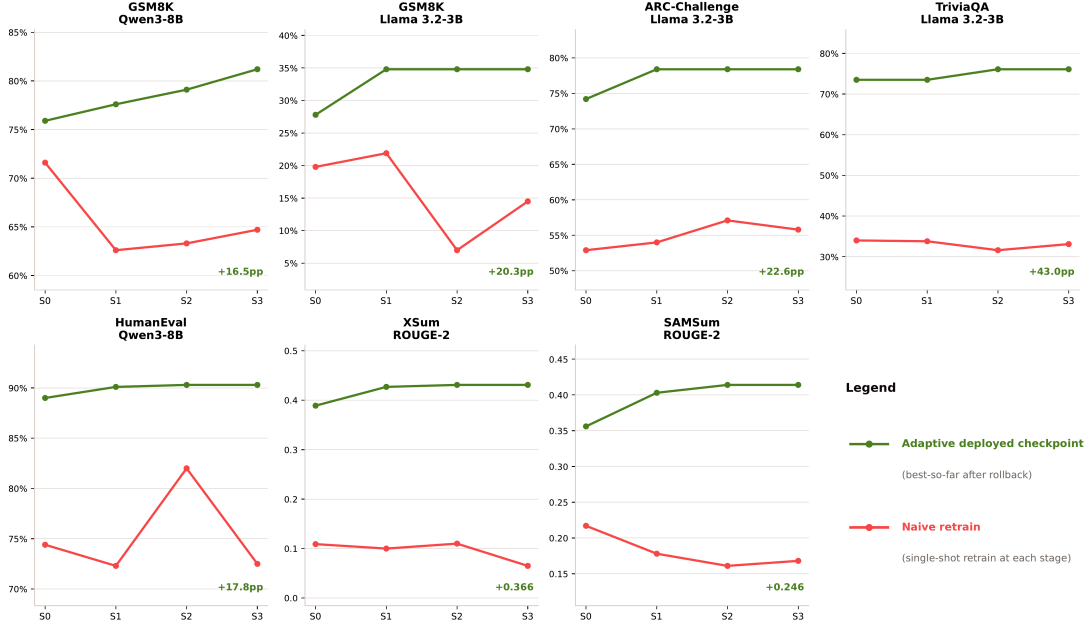


Figure 2: **Final-stage policy behavior across AdaptFT-Bench.** Green denotes Pioneer Agent’s deployed checkpoint; red denotes the naive checkpoint. The same pattern holds across tasks.

feedback, the agent selects chain-of-thought supervision for reasoning, low learning rates for knowledge-sensitive tasks, prompt constraints for summarization, hard negatives for precision failures, and small high-quality datasets when larger mixtures degrade performance. On HumanEval, 173 curated MBPP examples outperform a 348-example mixture; on ARC-Challenge, removing a noisy external mixture raises accuracy from 69.6% to 72.6%.

Search narrows from broad exploration to surgical repair. On SMS Spam, four initial jobs establish F1 0.983, after which three targeted rounds add only 55 examples and reach 0.997. On CLINC150, the agent instead spends its early effort constructing a five-group failure taxonomy, then builds one 1,254-example curriculum that fixes 450 of 453 failures. In both trajectories, diagnosis determines whether the next intervention changes data composition, optimization, supervision format, or the deployment threshold; training is not treated as the default response to every error. Figure 3 contrasts these two trajectories.

Optimization choices are task dependent. The selected training duration ranges from one epoch on XSum to two on GSM8K, five on ARC-Challenge, and eight on SAMSUm. Knowledge-sensitive tasks converge on a lower learning rate of 10^{-4} , while format-learning tasks tolerate $2 \times$

10^{-4} . On SAMSUm, the decisive intervention is not more data but replacing an underspecified summarization prompt with explicit length, outcome, perspective, and brevity constraints. These choices arise through evaluation feedback rather than a fixed per-benchmark recipe.

Graph search recombines independently useful branches. On ARC-Challenge, the Llama model moves from 5.3% to 61.3% after learning the output format, then to 68.9% with DeepSeek-R1 reasoning traces. Fusion combines that reasoning branch with validation-data expansion from another branch and reaches the retained 72.6% checkpoint. Five subsequent expansions with altered data and hyperparameters regress and are pruned, demonstrating how graph-structured lineage supports both recombination and convergence detection.

Rollback is a first-class intervention. Later fine-tuning is frequently harmful near saturation. Retaining the best regression-safe checkpoint prevents those failed experiments from becoming production regressions. This is visible in the rejected second CLINC150 candidate and in every stage-based trajectory. The monotone deployment curve is therefore a systems result: evaluation and checkpoint retention are part of adaptation, not book-keeping after training.



Figure 3: **Representative trajectories.** The controller adapts its intervention to the diagnosed failure rather than following a fixed fine-tuning recipe.

Failure type determines the intervention axis.

The nine production scenarios expose distinct failure geometries. Semantic poison calls for exclusion; recoverable surface corruption calls for contrastive pairs; confusion between neighboring labels calls for hard negatives; precision collapse calls for entity-free examples and threshold calibration; and infrastructure failure calls for changing the training method rather than changing the dataset. This is the value of searching over (D, H, S) jointly: the system can act on the causal diagnosis instead of forcing every failure through a larger dataset or longer training run.

The agent can decide that further training is the wrong action. On TriviaQA, the remaining failures are predominantly missing parametric knowledge at 3B scale. On adversarial or contradictory inputs, the target itself is not recoverable from the corrupted example. In both cases, the productive decision is to stop or exclude the row. Autonomous adaptation therefore requires selective restraint as well as the ability to train.

6 Related Work

Classical AutoML searches hyperparameters over predefined spaces (Thornton et al., 2013; Feurer et al., 2015; Li et al., 2018); data-centric methods identify label errors or value training examples (Northcutt et al., 2021; Ghorbani and Zou, 2019); and prompt or LM-program optimizers tune a fixed model’s interface (Khatab et al., 2024; Yang et al., 2024; Yuksekgonul et al., 2024). Pioneer Agent searches across all three layers because production failures can originate from data, optimization, or supervision format. Execution-

grounded agents including AIDE, MLE-bench, MLEvolve, AutoMLGen, InternAgent-1.5, FT-Dojo, and PostTrainBench automate ML experimentation or fine-tuning against static objectives (Chan et al., 2025; Jiang et al., 2025; InternScience, 2026; Du et al., 2025; Feng et al., 2026; Li et al., 2026; Rank et al., 2026). Pioneer Agent targets a different unit of evaluation: a deployed model must turn its own observed failures into supervision, reject harmful feedback, and preserve prior behavior across successive updates. Continual-learning methods address forgetting through replay or constrained updates (Kirkpatrick et al., 2017; Lopez-Paz and Ranzato, 2017; Wang et al., 2023b; Nayak et al., 2025); Pioneer Agent incorporates replay but adds failure taxonomy construction, curriculum synthesis, executable training search, and an operational acceptance gate. LLM-as-judge methods supply scalable quality signals (Zheng et al., 2023; Kim et al., 2024); our setting tests what an adaptation system can do when those signals contain both useful and harmful supervision.

7 Conclusion

Pioneer Agent automates the adaptation lifecycle: it converts task specifications or deployment failures into training curricula, executes candidate pipelines, and retains only regression-safe updates. Across eight cold-start tasks, seven corrupted AdaptFT-Bench scenarios, and two production-style case studies, it closes the loop from failure to verified model update.

8 Limitations

While the results demonstrate strong performance across a range of settings, several limitations remain.¹

Model families. The current system supports a limited set of model architectures: GLiNER2 encoders (Zaratiana et al., 2025) and Qwen/Llama decoders. Extending to other modalities, such as vision models or embedding models, as well as additional architectures, would require new tooling and integration work.

Training methods. Support for training paradigms is similarly constrained. Decoder models rely exclusively on LoRA fine-tuning (Hu et al., 2022), while encoder models support both full fine-tuning and LoRA. More advanced alignment methods, such as RLHF (Ouyang et al., 2022) or DPO (Rafailov et al., 2023), are not yet incorporated, limiting applicability in settings that require preference optimization.

Evaluation dependency. The production improvement setting depends on access to reliable evaluation signals, either through LLM-as-judge (Zheng et al., 2023; Gu et al., 2024; Li et al., 2024; Chen et al., 2024; Ho et al., 2025) or human annotations on inference logs. As a result, deployments without systematic quality monitoring cannot directly benefit from the agent.

Single-platform evaluation recipe. Empirical validation is tied to a single evaluation stack: the production-style case studies are run through the Pioneer Agent stack, and the synthetic scenarios are generated using our perturbation pipeline. The extent to which these results generalize across different platforms, logging infrastructures, or data distributions remains an open question.

Agent cost. The orchestrator LLM adds fixed overhead in token usage and latency, which can dominate in lightweight settings; Appendix C reports representative end-to-end costs. For simple fine-tuning tasks such as small hyperparameter sweeps or clean datasets, this overhead may outweigh the benefits of autonomous reasoning. The iterative loop further compounds cost through repeated diagnosis, data construction, training, and evaluation. This trade-off shifts with task complexity: in settings with noisy data or interacting

failure modes, the agent’s ability to diagnose errors and build targeted curricula can replace multiple rounds of manual debugging and make the system more cost-effective than naive iteration.

9 Ethical Considerations

Autonomous fine-tuning lowers the expertise required to specialize language models, but the same machinery can optimize harmful objectives if its evaluation signal rewards them. Regression gates reduce accidental drift but are not safety guarantees. Production use should preserve human approval before deployment, audit the provenance of synthesized supervision, restrict accessible models and data, and monitor for judge bias. Dataset-specific licenses also constrain use: in particular, SAMSum is non-commercial and CoNLL-2003 depends on Reuters data distributed for research use.

References

- Anthropic. 2026. Claude sonnet 4.6 system card. <https://www.anthropic.com/claude-sonnet-4-6-system-card>.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*.
- Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. 2025. [MLE-Bench: Evaluating machine learning agents on machine learning engineering](#). In *The Thirteenth International Conference on Learning Representations*.
- Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. Humans or llms as the judge? a study on judgement biases. *arXiv preprint arXiv:2402.10669*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

¹A previous version of this work appeared as an arXiv preprint.

- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Johannes Czech, Patrick Korus, and Kristian Kersting. 2021. Improving alphazero using monte-carlo graph search. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 31.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiaoling Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 179 others. 2025. *Deepseek-r1 incentivizes reasoning in llms through reinforcement learning*. *Nature*, 645:633 – 638.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. *Qlora: Efficient finetuning of quantized llms*. In *Advances in Neural Information Processing Systems*, volume 36.
- Shangheng Du, Xiangchao Yan, Dengyang Jiang, Jiakang Yuan, Yusong Hu, Xin Li, Liang He, Bo Zhang, and Lei Bai. 2025. Automlgen: Navigating fine-grained optimization for coding agents. *arXiv preprint arXiv:2510.08511*.
- Shiyang Feng, Runmin Ma, Xiangchao Yan, Yue Fan, Yusong Hu, Songtao Huang, Shuaiyu Zhang, Zongsheng Cao, Tianshuo Peng, Jiakang Yuan, Zijie Guo, Zhijie Zhong, Shangheng Du, Weida Wang, Jinxin Shi, Yuhao Zhou, Xiaohan He, Zhiyin Yu, Fangchen Yu, and 38 others. 2026. Internagent-1.5: A unified agentic framework for long-horizon autonomous scientific discovery. *arXiv preprint arXiv:2602.08990*.
- Matthias Feurer, Aaron Klein, Katharina Eggenberger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Yonggan Fu, Xin Dong, Shizhe Diao, Matthijs Van keirsbilck, Hanrong Ye, Wonmin Byeon, Yashaswi Karnati, Lucas Liebenwein, Maksim Khadkevich, Alexander Keller, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. 2025. *Nemotron-flash: Towards latency-optimal hybrid small language models*. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*.
- Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79.
- Aaron Grattafiori et al. 2024. *The llama 3 herd of models*. *Preprint*, arXiv:2407.21783.
- Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2024. A survey on LLM-as-a-judge. *arXiv preprint arXiv:2411.15594*.
- Xanh Ho, Jiahao Huang, Florian Boudin, and Akiko Aizawa. 2025. Llm-as-a-judge: Reassessing the performance of llms in extractive qa. *arXiv preprint arXiv:2504.11972*.
- 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*.
- InternScience. 2026. MLEvolve. <https://github.com/InternScience/MLEvolve>.
- Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. 2025. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1601–1611.
- Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Matei Zaharia, et al. 2024. *Dspy: Compiling declarative language model calls into state-of-the-art pipelines*. In *The Twelfth International Conference on Learning Representations*.
- Seungone Kim, Jamin Shin, Yejin Choi, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwińska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.
- Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Proceedings*, pages 282–293. Springer.

- Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of EMNLP-IJCNLP*, pages 1311–1316.
- Edouard Leurent and Odalric-Ambrym Maillard. 2020. Monte-carlo graph search: the value of merging similar states. In *Proceedings of the Asian Conference on Machine Learning (ACML)*, pages 577–592.
- Haitao Li et al. 2024. Llm-as-judges: A comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*.
- Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Roshtamizadeh, and Ameet Talwalkar. 2018. [Hyperband: A novel bandit-based approach to hyperparameter optimization](#). *Journal of Machine Learning Research*, 18(185):1–52.
- Qizheng Li, Yifei Zhang, Xiao Yang, Xu Yang, Zhuo Wang, Weiqing Liu, and Jiang Bian. 2026. Ft-dojos: Towards autonomous llm fine-tuning with language agents. *arXiv preprint arXiv:2603.01712*.
- David Lopez-Paz and Marc’Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*.
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don’t give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.
- Nikhil Shivakumar Nayak, Krishnateja Killamsetty, Ligong Han, Abhishek Bhandwaldar, Prateek Chanda, Kai Xu, Hao Wang, Aldo Pareja, Oleg Silkin, Mustafa Eyceoz, et al. 2025. Sculpting subspaces: Constrained full fine-tuning in llms for continual learning. *arXiv preprint arXiv:2504.07097*.
- Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research (JAIR)*.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. [Training language models to follow instructions with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 35.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*.
- Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. 2026. Posttrainbench: Can llm agents automate llm post-training? *arXiv preprint arXiv:2603.08640*.
- Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.
- Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning (CoNLL-2003)*, pages 142–147.
- Ruida Wang, Wangchunshu Zhou, and Mrinmaya Sachan. 2023a. Let’s synthesize step by step: Iterative dataset synthesis with large language models by extrapolating errors from small models. In *Findings of the Association for Computational Linguistics (EMNLP Findings)*.
- Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. 2023b. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10658–10671.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. [Large language models as optimizers](#). In *The Twelfth International Conference on Learning Representations*.
- Mert Yuksekogunul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic “differentiation” via text. *arXiv preprint arXiv:2406.07496*.
- Urchade Zaratiana, Gil Pasternak, Oliver Boyd, George Hurn-Maloney, and Ash Lewis. 2025. [GLiNER2: Schema-driven multi-task learning for structured information extraction](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 130–140, Suzhou, China. Association for Computational Linguistics.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena.

A Detailed Pioneer Agent Architecture and Operation

A.1 Hierarchical State Machine

The agent runs inside an isolated Modal sandbox with access to data manipulation, model training, inference, evaluation, and deployment tools. Decoder training uses the Tinker SDK for LoRA fine-tuning and immediate inference on the resulting checkpoint.

The main agent owns the complete experimental state. It plans work, launches training jobs, evaluates models, diagnoses failures, and decides which branch to expand. A specialized Trace Analyzer handles data-heavy SQL analysis and writes structured artifacts such as taxonomies, confusion matrices, sampled traces, and dataset statistics to the shared filesystem. Independent sub-agents can be delegated parallel work, for example constructing a candidate dataset while another job is training. A Context Manager compacts long trajectories while preserving decisions, evaluation results, dataset lineage, and unresolved hypotheses. The durable `data-curation.md` ledger records dataset versions, composition ratios, quality-control decisions, training configurations, and evaluation outcomes so the agent can reconstruct the entire trajectory after compaction.

The state graph contains explicit task-analysis, research, data-acquisition, evaluation-construction, curriculum-synthesis, training, inference, diagnosis, and checkpoint-promotion states. Transitions are driven by the latest evaluation and the agent’s causal hypothesis about the remaining failures. This makes the experiment history executable: every edge corresponds to a concrete intervention, and every promoted checkpoint has an associated dataset, configuration, supervision strategy, and evaluation artifact.

A.2 Tool Interface

Cold-start mode exposes web research, shell execution, file reading and editing, dataset helpers, training, inference, and evaluation. The agent can research a task, download public data, normalize schemas, generate supervision, launch multiple candidate jobs, and score the resulting models without leaving the sandbox.

Production mode adds five core tools: `query_traces`, `trace_analysis_subagent`,

`bash`, `read_file`, and `edit_file`. The trace-query tool accepts both a SQL query and an optional shell pipeline. Query results can therefore be filtered, aggregated, sampled, or persisted to disk in the same invocation. Only the compact printed summary enters the agent’s context; the complete result remains on disk for curriculum construction and training.

The agent uses the same filesystem as its sub-agents. A delegated analysis can save a confusion matrix, failure taxonomy, or filtered training set, after which the main agent reads only the summary and artifact path. This division of labor prevents raw trace volume from crowding out high-level experimental reasoning.

A.3 Data Curation and Dataset Sizing

Every training dataset is assembled from three slices. **Gold examples** are correct input-output pairs sourced from public benchmarks in cold-start mode or corrected deployment failures in production mode. **Hard negatives** are confusable inputs whose correct output differs from a plausible alternative, teaching the model when not to emit a label, entity, or answer. A **replay buffer** samples 10–20% of the parent dataset when adapting an already fine-tuned model, preserving previously learned behavior while new failures are repaired.

Five constraints govern data quality:

1. **Paired boundary supervision.** Each difficult gold example is accompanied by a nearby counterexample with a different correct output.
2. **Label balancing.** No label may dominate the smallest class by more than a factor of three.
3. **Context-length matching.** Training examples follow the length distribution of realistic inputs or observed failures.
4. **Entity diversification.** NER datasets replace repeated entity values with diverse synthetic equivalents to prevent surface-form memorization.
5. **Reasoning supervision.** Generation tasks can use teacher-produced chains of thought when direct-answer supervision fails to teach the required reasoning procedure.

Each label is also represented through multiple surface patterns. The agent targets compact datasets

rather than indiscriminate scale: approximately 100–200 examples for classification and NER, and 500–3,000 for generation depending on task complexity. It treats every expansion as an experiment. If a larger dataset regresses on the fixed evaluation split, the added data and its associated configuration are rolled back.

A.4 Iteration Policy

The iteration policy changes intervention scale as performance improves:

1. **Score below 0.80: rebuild coverage.** The agent diagnoses missing labels, distribution mismatch, insufficient hard negatives, or incorrect supervision and reconstructs the dataset.
2. **Score from 0.80 to 0.95: isolate optimization.** The dataset remains fixed while the agent searches learning rate, epochs, batch size, LoRA rank, base model, and system prompt.
3. **Score above 0.95: repair surgically.** The agent adds only two or three targeted examples for each remaining failure pattern.
4. **Regression budget exceeded: roll back.** If $r(\pi; \mathcal{R}) > \epsilon$, the previous dataset, configuration, and checkpoint remain the active branch.

This policy produces broad-to-surgical trajectories. Early iterations explore multiple data and optimization hypotheses in parallel; mid-stage iterations isolate the highest-leverage variable; late iterations use targeted augmentation and threshold calibration. The search ends when the required score is reached, the remaining errors are diagnosed as external to training, or additional branches fail to improve the best checkpoint.

A.5 Production Failure Diagnosis

Production mode receives a deployed model M_0 and judged traces

$$\mathcal{T} = \{(x_i, \hat{y}_i, y_i^*, v_i, r_i, m_i)\}_{i=1}^n, \quad (2)$$

where x_i is the input, $\hat{y}_i$ the model output, y_i^* the corrected output, v_i the pass/fail verdict, r_i the judge reasoning, and m_i the evaluation metadata. Human overrides supersede automated verdicts. The agent partitions the traces into passing

and failing sets, then diagnoses the failure set in four stages.

Trace ingestion queries the inference store and materializes filtered artifacts without loading raw rows into the main context. **Taxonomy construction** clusters failures by input characteristics, confusion pattern, and likely root cause, recording cluster size and whether the cause is fixable through training or external to the model. **Live confirmation** synthesizes targeted probes to verify that a suspected weakness is systematic. **Parent-model awareness** recovers the deployed model’s training lineage and decides whether the corrective dataset should complement prior supervision or be constructed from scratch.

For a fine-tuned parent with dataset D_{parent} , the agent constructs the corrective dataset:

$$D_{\text{post}} = D_{\text{gold}} \cup D_{\text{hard}} \cup D_{\text{replay}}, \quad (3)$$

where D_{gold} holds the corrected failures, D_{hard} the confusion pairs they induce, and D_{replay} a 10–20% sample of D_{parent} . Recall failures increase the gold share; precision failures increase hard negatives; model-capacity failures can trigger scale-up; prompt failures trigger a prompt change; and infrastructure failures can trigger a different training method. Candidate models are evaluated on the failure set and a structured regression sample of previously passing traces before promotion.

A.6 Structural Safeguards

Autonomous adaptation is constrained at both the data and checkpoint layers. Balanced curricula and paired hard negatives prevent majority-class shortcuts. Parallel candidate evaluation separates competing hypotheses. The rollback-first policy stops later experiments from overwriting a stronger checkpoint. The cross-checkpoint regression gate evaluates a candidate on both the current failure set and the preceding checkpoint’s evaluation set, preventing temporal overfitting to the newest slice.

The agent also calibrates raw confidence using historical accuracy for the predicted label. High-confidence errors on labels with poor empirical accuracy are pulled toward the observed rate and prioritized for review. When a human corrects an item, similarity search identifies related unlabeled examples so the same pattern can be audited in bulk.

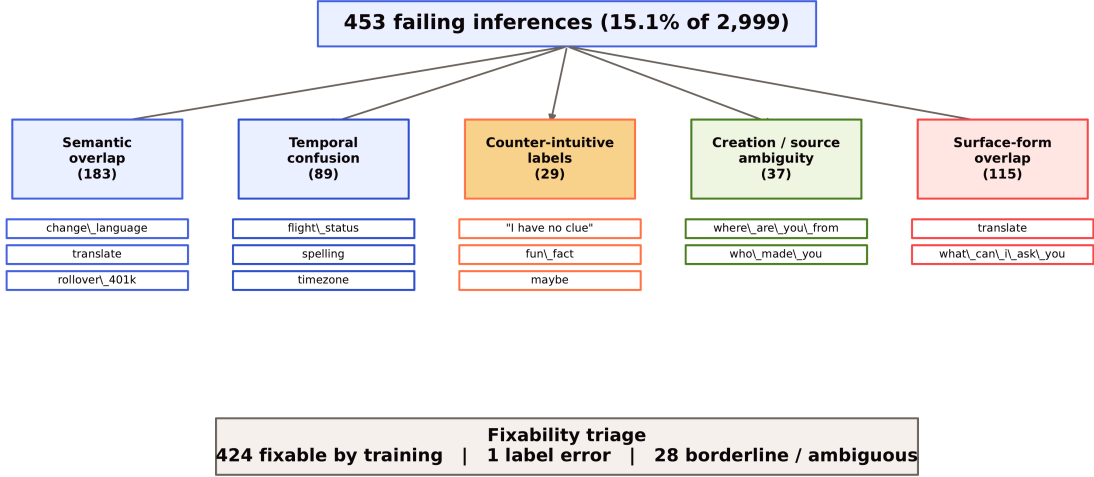


Figure 4: The CLINC150 taxonomy converts 453 failures into actionable confusion categories, then separates fixable cases from label and boundary problems.

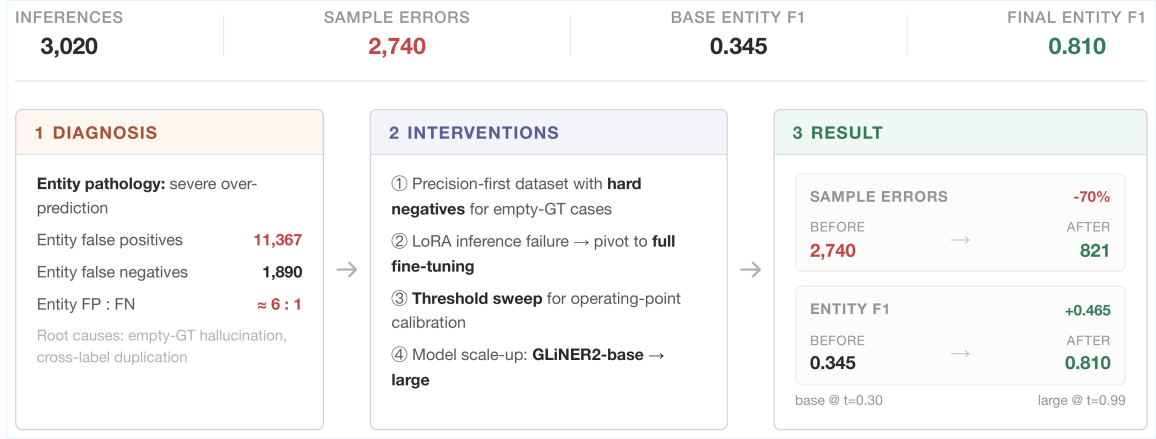


Figure 5: The CoNLL-2003 trajectory: diagnosis, precision-oriented interventions, infrastructure recovery, scale-up, and threshold tuning.

B Monte Carlo Graph Search for Training Pipelines

Pioneer Agent applies Monte Carlo Graph Search (Leurent and Maillard, 2020; Czech et al., 2021) to autonomous fine-tuning, searching over complete pipelines $\pi = (D, H, S)$, where D is the dataset specification, H the optimization configuration, and S the learning strategy. Each evaluated node $v = (\pi, s)$ stores a pipeline and its realized validation score $s = f(\pi)$. Directed edges preserve the parent-child intervention that generated a new pipeline. What is specific to our setting is the node semantics and the reward: a node is an executable training pipeline, and its score is obtained by actually training and

evaluating a model.

Figure 6 states the procedure. Node selection uses a UCT-like score (Kocsis and Szepesvári, 2006)

$$\text{UCT}(v_i) = \bar{f}(v_i) + c(t) \sqrt{\frac{\ln N}{n_i}}, \quad (4)$$

where $\bar{f}(v_i)$ is the mean descendant reward, N is the total visit count, n_i is the visit count for v_i , and $c(t)$ is a decaying exploration coefficient managed by the orchestrator. Expansion is hypothesis-driven: the agent inspects the parent trajectory and proposes a causal modification to data, optimization, or supervision. Every proposal is scored by executing the full training and evaluation pipeline rather than a surrogate. When a branch stagnates,

Input: failure set or task specification $\mathcal{F}$; evaluation function $f(\pi)$ that trains and evaluates pipeline π . **Output:** best pipeline π^* .

1. Initialize $\pi_0 \leftarrow \text{Baseline}(\mathcal{F})$, evaluate $s_0 \leftarrow f(\pi_0)$, create $v_0 = (\pi_0, s_0)$, and set $\mathcal{G} = (\{v_0\}, \emptyset)$.
2. While Continue($\mathcal{G}, \mathcal{F}$):
 - (a) Select $v_{\text{parent}} \leftarrow \arg \max_{v \in \text{Leaf}(\mathcal{G})} \text{UCT}(v)$.
 - (b) Expand $\pi_{\text{new}} \leftarrow \text{Agent}(v_{\text{parent}}, \mathcal{G}, \mathcal{F})$; evaluate $s_{\text{new}} \leftarrow f(\pi_{\text{new}})$; add $v_{\text{new}} = (\pi_{\text{new}}, s_{\text{new}})$ and edge $(v_{\text{parent}}, v_{\text{new}})$.
 - (c) Run Backprop($\mathcal{G}, v_{\text{new}}$) to update ancestor score and visit statistics.
 - (d) If the branch is stagnant, either evolve it with $\pi_{\text{evo}} \leftarrow \text{Evolve}(v_{\text{parent}}, \mathcal{G}, \mathcal{F})$ or obtain $\mathcal{V}_{\text{top}} \leftarrow \text{TopK}(\mathcal{G})$ and fuse $\pi_{\text{fused}} \leftarrow \text{Fuse}(\mathcal{V}_{\text{top}})$. Execute the selected pipeline, add the evaluated node, and update $\mathcal{G}$.
3. Return π^* from $(\pi^*, s^*) = \arg \max_{(\pi, s) \in \mathcal{V}} s$.

Figure 6: **Monte Carlo Graph Search for training pipelines.** Each node stores a complete pipeline and realized score. The agent expands promising leaves, executes train-and-test evaluation, backpropagates score statistics, and invokes evolution or cross-branch fusion when progress stalls.

the search either evolves it through a trajectory-aware mutation or fuses complementary discoveries from the top-scoring branches.

The ARC-Challenge Llama trajectory in Figure 7 demonstrates the search dynamics, including escalation from selection to fusion. One branch acquires the required output format, another discovers chain-of-thought supervision, and two branches compare DeepSeek-R1 and GPT-4.1 reasoning teachers. Fusion combines DeepSeek-R1 reasoning traces with validation-data expansion and reaches 72.6%. Subsequent expansions alter data, LoRA rank, and learning rate but regress and are pruned, leaving the fused branch as the selected pipeline.

C Reproducibility and Configuration

Orchestrator. All reported runs use Claude Sonnet 4.6 with temperature 1.0, a 32K-token extended-thinking budget, and a 1M-token context window. Cold-start runs permit up to 1,500 LangGraph turns; production runs permit up to 500. Sandboxes provide 16 GB host memory and a 24-hour timeout. Search terminates when the acceptance criterion is met, the remaining failures are diagnosed as external to training, or the sandbox reaches its timeout. The source package includes the April 3, 2026 prompt and plan snapshot at repository commit 2febb0664f05 under supplementary/prompt_snapshot/.

Training. Decoder models use Tinker LoRA. Unless an agent-selected pipeline overrides them, defaults are batch size 8, learning rate 2×10^{-4} , cosine decay with 10% warmup, and LoRA rank 16. GLiNER2 supports full fine-tuning and LoRA.

Run	Hours	LLM	Train	Total
GSM8K production	11.5	\$22.98	\$10.00	\$32.98
CLINC150	1.0	\$1.98	\$1.50	\$3.48
ARC-Challenge	12.0	\$24.00	\$12.00	\$36.00
HumanEval	4.0	\$7.98	\$4.50	\$12.48
XSum	10.0	\$19.98	\$35.00	\$54.98

Table 4: Representative end-to-end costs.

Evaluation. ARC-Challenge uses exact match on the answer letter; GSM8K exact match on the final number; TriviaQA case-insensitive alias matching; HumanEval pass@1 with the official execution harness; XSum and SAMSum ROUGE-2; classification F1 or exact match; and NER entity F1.

Cost. Table 4 reports wall-clock hours, orchestrator token cost, and training cost for five representative end-to-end runs.

D Matched-Stage GSM8K Trajectory

Table 5 reports the GSM8K/Qwen3-8B checkpoints behind the summary in Section 4, and Figure 8 annotates the intervention chosen at each stage. The two policies start within 4.3 points of each other on clean feedback; the gap widens monotonically as the poison rate rises to 40%, because naive retraining absorbs corrupted supervision while Pioneer Agent filters it and rolls back failed candidates.

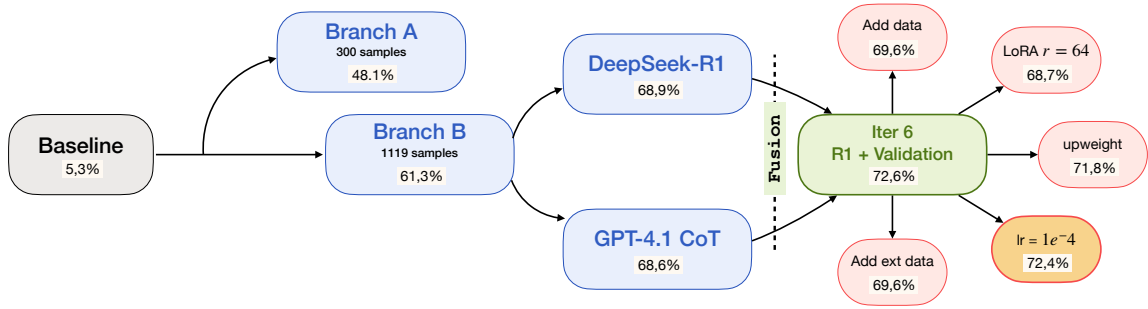


Figure 7: ARC-Challenge MCGS trajectory. Branch fusion reaches the retained 72.6% checkpoint; subsequent regressions are pruned.

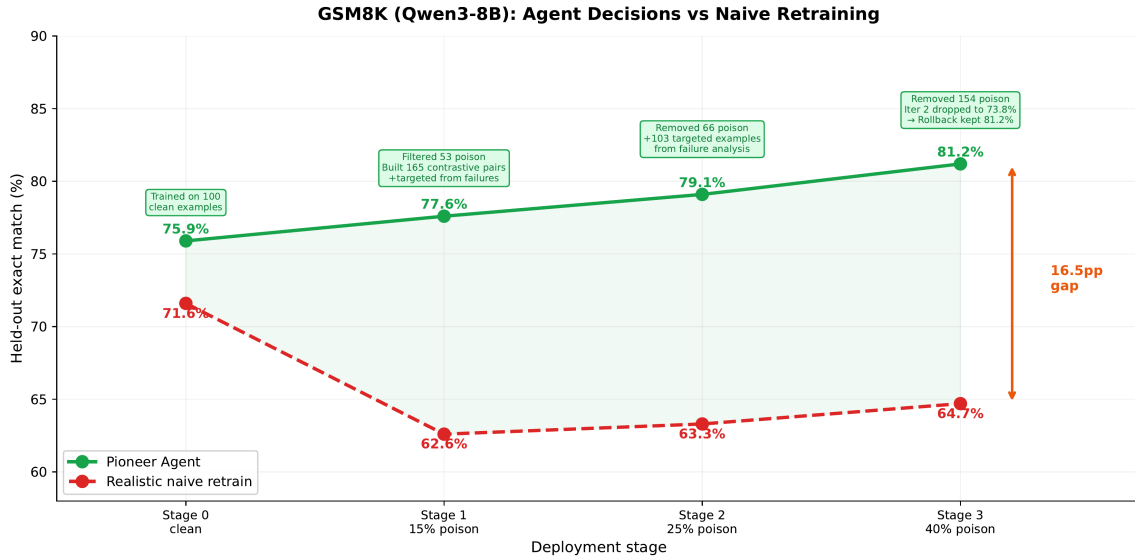


Figure 8: The matched-stage GSM8K trajectory with the intervention selected at each stage.

Checkpoint	Naive	Pioneer	Gap	Poison
C1: clean	71.6	75.9	+4.3	0%
C2: stage 1	62.6	77.6	+15.0	15%
C3: stage 2	63.3	79.1	+15.8	25%
C4: stage 3	64.7	81.2	+16.5	40%

Table 5: GSM8K/Qwen3-8B checkpoints after each matched feedback stage.