DOES WIDER SEARCH HELP? AUTONOMOUS RESEARCH FOR IMPROVING VISION-LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based research agents can autonomously propose hypotheses, modify training code, execute experiments, and substantially improve machine-learning models. Because model improvement is iterative, autonomous research can be viewed as search over research states and interventions, which raises the question of how much search control is actually needed. We study this question in autonomous post-training of a vision-language model by comparing a greedy single-trajectory policy ($K{=}1$) with a score-based multi-trajectory policy ($K{=}3$). The two policies use the same research agent, starting model, training setup, evaluation pipeline, and research memory, and differ only in how many research states remain active. Both improve the aggregate score of the starting model from 0.576 to about 0.70, a relative gain of about 22% on average. However, under a round-aligned comparison that favors the wider policy, which trains three times as many candidates per round, $K{=}3$ reaches the plateau in fewer rounds but not a higher one: its best run (0.703) falls below the median greedy run (0.707). Analysis of the research histories shows that the wider policy retains diverse research states and attempts a broader range of intervention families, yet discovers no more productive ones, and that the effect of the same intervention can reverse sign as the surrounding training state changes, by margins well above training-seed noise. Our findings suggest that the large gains of autonomous post-training do not require wider search, and that exploiting additional preserved states, rather than generating them, is the central limitation for wider search in autonomous model research.

1 INTRODUCTION

Improving a machine-learning model is an iterative research process: researchers form hypotheses, modify models or training procedures, execute experiments, interpret the resulting evidence, and decide what to try next. Recent LLM-based agents have begun to automate substantial parts of this loop. Systems such as MLAgentBench (Huang et al., 2024) and MLE-bench (Chan et al., 2025) evaluate agents that inspect machine-learning codebases, modify training pipelines, and iteratively improve solutions, while systems such as the AI Scientist extend this idea toward broader scientific workflows (Lu et al., 2024).

As autonomous experimentation becomes increasingly capable, an important question is not only whether an agent can improve a model, but *how much search complexity is actually required to do so*. Improvements may arise primarily from the quality of the underlying research agent rather than from a sophisticated policy for organizing research states. Distinguishing these effects is important: an autonomous system may substantially improve a model while gaining little from maintaining multiple competing research trajectories.

A central design choice is therefore how many research states to keep active. A single-trajectory policy follows one lineage at a time, whereas a multi-trajectory policy preserves several alternative states that can be revisited in later rounds. Wider search is appealing because intervention value may depend on research state: an intervention that is ineffective in one configuration may become useful after other aspects of the training procedure change. Maintaining alternatives can therefore protect against premature commitment, motivating recent systems based on tree search and parallel

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

trajectories (Jiang et al., 2025a; Toledo et al., 2025). However, preserving more states is useful only if the planner can identify productive follow-up experiments for them.

We study this question in an autonomous post-training framework for improving a vision-language model. An LLM-based research agent reads previous hypotheses and experimental outcomes, proposes code-level interventions, modifies the training procedure, trains candidate models, and evaluates them on a fixed benchmark. We isolate the effect of search control by comparing two policies using the same research agent, starting model, training environment, and evaluation pipeline: a greedy single-trajectory policy ($K = 1$) and a multi-trajectory policy ($K = 3$). We compare them over the same number of *research rounds*, giving both policies the same number of sequential opportunities to incorporate experimental feedback while allowing $K = 3$ to evaluate multiple alternatives within a round.

Our central result is that the large gains from autonomous post-training do not require wider search in the studied setting. Both policies improve the starting model from 0.576 to about 0.70. Under a round-aligned comparison, which favors K=3 because it trains three times as many candidates per round, the wider policy reaches the plateau in fewer rounds but does not reach a higher one: at the common horizon of 22 rounds, its best run (0.703) falls below the median greedy run (0.707). The overall success of autonomous post-training should therefore be distinguished from the marginal value of additional search complexity.

Analysis of the research histories makes this result more informative. The multi-trajectory policy genuinely explores more broadly: it retains diverse research states, covers more intervention families, and distributes experimentation more evenly across them. We also observe strong state dependence: the same intervention can be harmful, beneficial, and later harmful again as the surrounding training configuration changes, by margins well above training-seed noise. These observations confirm the motivation for preserving alternative states, yet those additional states still do not yield a stronger model. Our results therefore suggest that the challenge of wider search is not simply generating alternatives, but making effective *state-conditioned* research decisions about how to exploit them.

Contributions. Our main contributions are:

- We formulate trajectory width as an end-to-end research-policy variable and separate the overall gains of autonomous post-training from the marginal contribution of search complexity.
- We conduct a controlled comparison of single-trajectory ($K = 1$) and multi-trajectory ($K = 3$) autoresearch under a common research-round horizon, holding the underlying research agent and experimental environment fixed.
- We show that both policies improve the starting model by roughly 22%, while wider search reaches its plateau sooner but not higher: under a round-aligned comparison in which K=3 trains three times as many candidates per round, no K=3 run exceeds the median K=1 run.
- Through analysis of the research histories, we show that wider search produces greater exploratory breadth and that intervention effectiveness can depend strongly on research state, highlighting state-conditioned intervention selection as an important challenge for converting search breadth into model improvement.

2 Related Work

LLM Agents for Machine-Learning Research MLAgentBench (Huang et al., 2024) and MLE-bench (Chan et al., 2025) evaluate LLM agents on machine-learning experimentation and engineering, and the AI Scientist (Lu et al., 2024) and Agent Laboratory (Schmidgall et al., 2025) automate broader research workflows. These works establish that LLM agents can perform instruction-based experimentation and we ask how much additional value comes from the policy that organizes autonomous research experiments.

Search and Control in Autonomous Research Agents Several systems search over alternative solutions: AIDE (Jiang et al., 2025a) and the AI Scientist-v2 (Yamada et al., 2025) use tree search,

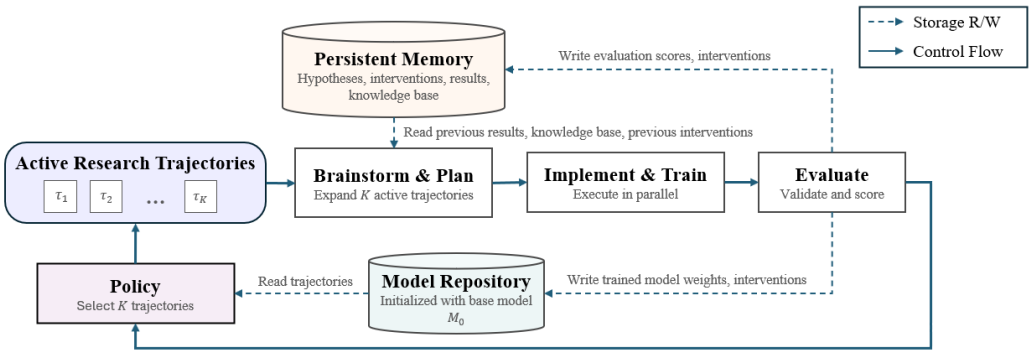

Figure 1: Overview of the autonomous research framework. An LLM-based research agent proposes code-level interventions, launches training experiments, evaluates the resulting models, and records both executable model states and semantic research memory. We hold the planner, training and evaluation infrastructure, starting model, and research-memory mechanism fixed, and vary the number of active research trajectories available for subsequent experimentation.

ML-Master draws on parallel solution trajectories (Liu et al., 2025) and R&D-Agent supports multiple exploration traces (Yang et al., 2025).

Most closely related, Toledo et al. (2025) formulate AI research agents as search policies over candidate machine-learning solutions and compare Greedy, Monte Carlo Tree Search, and Evolutionary strategies, together with different modification operators, on MLE-bench. We ask another question in a different regime: rather than comparing search algorithms on MLE-bench tasks, we vary only trajectory width (K=1 vs. K=3) in autonomous post-training, where each search step is a full model-training run and a research run affords only tens of such expensive experiments. We hold the planner, training setup, evaluation pipeline, and research memory fixed. Because the memory retains every experiment, K changes how many research states can be continued directly, not what history the planner can access.

Automated Model Search Searching over model configurations has a long history in automated machine learning, including random search (Bergstra & Bengio, 2012), Hyperband (Li et al., 2018), Population Based Training (Jaderberg et al., 2017), and neural architecture search (Elsken et al., 2019). These methods generally operate over a researcher-specified space of hyperparameters or architectures, whereas an LLM research agent can formulate and implement code-level interventions during the search, so the intervention space need not be exhaustively parameterized in advance.

3 METHODOLOGY

We study the marginal value of search complexity in autonomous model post-training. An LLM-based research agent proposes hypotheses, modifies training code, executes experiments, interprets outcomes, and selects subsequent experiments. Holding the agent, tools, training environment, and evaluator fixed, we vary only the number of active research trajectories K. This isolates the value of search control from the overall capability of the research agent. We compare a single-trajectory policy ($K = 1$) with a multi-trajectory policy ($K = 3$).

3.1 RESEARCH TRAJECTORIES

Let M_0 denote the starting model and let a denote a code-level intervention proposed by the research agent from a current model state M. Applying intervention a to a parent model M produces a candidate model state M', which is evaluated using the same fixed evaluation pipeline. The evaluator returns task-level metrics together with an aggregate score $S(M')$. The resulting experimental outcome, including the intervention, model state, evaluation results, and interpretation, becomes part of the research history available to subsequent planning decisions.

A research trajectory records the sequence of model states and interventions leading from the starting model to a current state,

$$\tau_i = (M_0, a_1, M_1, \ldots, a_i, M_i)\,.$$

The associated research context also contains the hypotheses, observations, and interpretations accumulated while developing that state. Consequently, two models with similar aggregate scores may still represent different research states if they were reached through different interventions or support different follow-up hypotheses.

At any point, the search policy maintains a set of active research trajectories,

$$\mathcal{B}_t = \left\{\tau_t^{(1)}, \ldots, \tau_t^{(K)}\right\}.$$

All conditions begin with the same initial state, $\mathcal{B}_0 = \{\tau_0\}$. The width K determines how many model states can remain active as candidate parents for future training experiments.

3.2 Search-Control Policies

We compare two search-control policies within the same autonomous post-training system.

Greedy single-trajectory policy ($K = 1$). The greedy policy maintains one active model state. A newly produced candidate replaces the incumbent when it becomes the highest-scoring retained state; otherwise the incumbent remains active. Future rounds therefore continue from a single current parent:

$$\tau_0 \to \tau_1 \to \tau_2 \to \cdots\,.$$

Because only one state receives subsequent training budget, the policy maximizes the number of sequential opportunities to refine a single research lineage.

Multi-trajectory policy ($K = 3$). The wider policy maintains up to three active model states. During research round t, the planner proposes a set of candidate experiments conditioned on the active research contexts. Each proposal specifies an intervention and the active parent state to which it should be applied.

Let $\mathcal{C}_t$ denote the set of children produced and evaluated during that round. After all children in the batch have been evaluated, the policy considers both the previously active states and the new candidates,

$$\mathcal{Q}_t = \mathcal{B}_t \cup \mathcal{C}_t,$$

and retains the highest-scoring trajectories,

$$\mathcal{B}_{t+1} = \mathrm{TopK}(\mathcal{Q}_t; S)\,.$$

where TopK selects K trajectories as scored by the function S. Children proposed within the same round are treated as an *unordered batch*. Their outcomes are not available to the planner when the other children in that batch are proposed, so we do not impose an artificial sequential ordering among them. The next planning decision occurs only after the batch has completed.

Both policies use the same proposer, training setup, evaluation pipeline, score-based retention criterion, and persistent state: a model repository that keeps every evaluated candidates with its code, configuration, scores, and parent, and a research memory of hypotheses, interventions, outcomes, and failures. A state that leaves the active set is therefore not forgotten. Increasing K changes how many states can be continued directly, and hence which experiments the planner proposes, but not what history the planner can access. A detailed description of the persistent states are outlined in Appendix B.

3.3 Research Rounds and Policy Evaluation

Autonomous research proceeds in rounds $t = 1, \ldots, R$. At each round, the planner conditions on the active research states and persistent history, proposes interventions and their parent states, and evaluates the resulting candidates using the fixed benchmark as showed in Figure 1. The resulting models, scores, and experimental interpretations are stored for subsequent planning.

A research round represents one opportunity for *sequential adaptation*: the planner can use outcomes from preceding rounds, but not from other candidates proposed within the same round. We therefore compare policies after the same number of research rounds rather than at matched compute. This deliberately favors $K{=}3$, which has the same number of sequential feedback opportunities as $K{=}1$ but evaluates more candidates per round.

For policy p, independent run r, and round t, let $\mathcal{C}_{p,r,t}$ denote the candidates evaluated in that round. The best-so-far score through round R is

$$S^*_{p,r}(R) = \max\left\{S(M_0), \max_{1\le t\le R}\max_{M\in\mathcal{C}_{p,r,t}} S(M)\right\}.$$

Runs of different lengths are compared over the common horizon

$$R_{\text{common}} = \min_r R_r.$$

Because research decisions within a run adapt to previous outcomes, the complete autoresearch run is the unit of replication, and policy statistics are computed across independent runs.

We additionally analyze the research histories using the process metrics defined in Section 4.3.

4 Experimental Setup

4.1 Model and Research Environment

We study autonomous post-training of a VLM2Vec-Qwen2-VL-7B (Jiang et al., 2025b) vision-language model initialized from a checkpoint already adapted from Qwen2-VL-7B (Wang et al., 2024) for multimodal contrastive representation learning. All candidate models are required to produce a single ℓ_2-normalized embedding vector per input, and retrieval is performed using cosine similarity. Detailed model configuration, training hyperparameters, and compute settings are provided in the appendix.

During autoresearch, the agent may modify the model and training pipeline, including the training objective, negative construction, batch and data composition, optimization strategy, parameter-efficient adaptation, input processing, and model components. The evaluation benchmark, scoring procedure, and model output interface remain fixed throughout the study.

The research planner is Claude Opus 4.6 and is identical across conditions. Each policy is evaluated over three independent autoresearch runs, each beginning from the same starting model and experimental environment with fresh research memory.

4.2 Evaluation Benchmark and Metric

We evaluate each candidate model on a fixed subset of 12 MMEB-V2 (Meng et al., 2025) tasks covering image classification, visual question answering, image retrieval, visual grounding, video classification, video question answering, video retrieval, moment retrieval, and visual-document retrieval.

For the eight image and video tasks, performance is measured using `hit@1`; for the four visual-document tasks, we use `ndcg_linear@5`. The aggregate score is

$$S(M) = \frac{1}{78}\sum_{c=1}^{12} w_c s_c(M),$$

where $s_c(M)$ denotes the score of model M on task c and the task weights w_c sum to 78.

The starting model is evaluated using the same pipeline before autoresearch begins. Every subsequently produced candidate is evaluated using the same tasks, metrics, and aggregation procedure.

4.3 RESEARCH-PROCESS METRICS

Performance alone does not reveal how trajectory width changes the research process. We therefore analyze the complete research histories to characterize various properties.

Exploratory entropy. Each experiment is assigned to a primary intervention family using a common taxonomy for both policies. The taxonomy includes objective and loss design, negative construction, batch composition, data mixture, optimization and scheduling, parameter-efficient adaptation, prompt or input processing, and pooling or architectural changes.

Let $q_r(f; R)$ denote the fraction of training interventions assigned to intervention family f in run r through research round R. We measure how broadly experimental effort is distributed using normalized entropy,

$$\bar{H}_r(R) = \frac{-\sum_{f \in \mathcal{F}} q_r(f; R) \log q_r(f; R)}{\log |\mathcal{F}|}.$$

These measurements determine whether multiple trajectories produce genuinely broader exploration.

Lineage and frontier advancement. We measure the maximum parent–child lineage depth reached through round R. This statistic characterizes whether the policy sustains continued development along a single sequence of model states or repeatedly shifts development among different branches.

We also count research rounds that advance the best-so-far performance frontier. For run r,

$$A_r(R) = \sum_{t=1}^{R} \mathbf{1}\left[\max_{M \in \mathcal{C}_{r,t}} S(M) > S_r^*(t-1)\right].$$

For the multi-trajectory policy, a round contributes at most one frontier advance even when multiple candidates are evaluated within that round. This avoids assigning a sequential ordering to candidates that were proposed as an unordered batch.

State-dependent intervention effects. We inspect repeated or closely related interventions applied from different parent states. These comparisons test whether the effect of an intervention transfers across the research history or depends on the model state from which it is applied. Because these states arise adaptively during autoresearch, we interpret such comparisons as descriptive evidence of state-dependent behavior rather than as controlled causal estimates.

Intervention-family coverage. Each valid training intervention is assigned to one primary intervention family using the same taxonomy for both policies. The eight families are detailed in the Appendix C.1.

Let $\mathcal{F}$ denote the eight-family taxonomy, and let $N_{r,f}(R)$ denote the number of valid interventions assigned to family $f \in \mathcal{F}$ in run r through research round R. Then, intervention-family coverage is the number of distinct intervention families for which the agent evaluates at least one valid non-control training intervention through research round R:

$$C_r(R) = |\{f \in \mathcal{F} : N_{r,f}(R) > 0\}|.$$

Productive intervention-family coverage. Exploratory entropy alone does not indicate whether the additional experiments discover new useful research directions. We therefore record which intervention families contain at least one valid intervention that improves its attributable parent state or matched control. Let $\mathcal{I}_r(R)$ denote the set of valid non-control training interventions evaluated in run r through round R. For intervention i, let M_i denote the resulting candidate, Q_i its attributable parent state or matched control, and f_i its primary intervention family. Then, productive-family coverage is

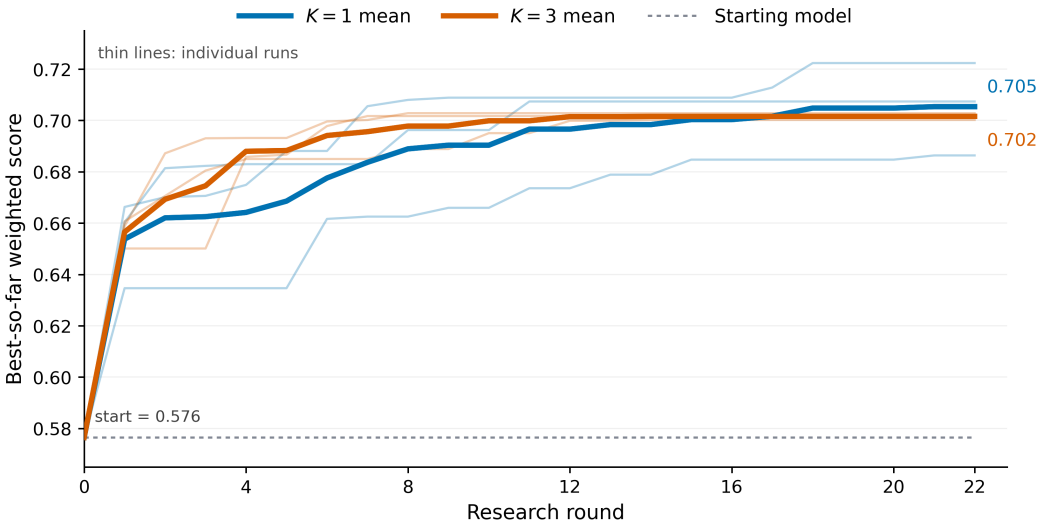

Figure 2: Best-so-far model performance versus research round. Thin lines show independent autoresearch runs and thick lines show policy means (K=1 in blue, K=3 in orange). Candidates proposed within the same K=3 round form an unordered batch, so the frontier is updated only after the round is complete.

Table 1: Best-so-far performance of the two autonomous research policies at the common research-round horizon $R_{\text{common}} = 22$. Each autoresearch run is treated as one unit of replication. Both policies substantially improve over the common starting model. No $K = 3$ run exceeds the median $K = 1$ run.

Policy	Run 1	Run 2	Run 3	Avg. $\pm$ SD	Avg. gain
Greedy ($K = 1$)	0.7223	0.7074	0.6864	0.7054 ± 0.0181	+0.1289
Multi-trajectory ($K = 3$)	0.7017	0.7002	0.7028	0.7016 ± 0.0013	+0.1251

$$P_r(R) = \left|\{f \in \mathcal{F} : \exists i \in \mathcal{I}_r(R),\ f_i = f \wedge S(M_i) > S(Q_i)\}\right|,$$

where Q_i is the attributable parent or matched control for intervention i.

5 Results

We first evaluate whether autonomous post-training improves the starting model and then ask whether increasing the number of active research trajectories provides additional value, using the round-aligned comparison following Section 3.3.

5.1 Autonomous Post-Training Produces Large Gains Without Requiring Wider Search

All runs begin from the same starting model, with aggregate score $S(M_0) = 0.576474$. Both search policies substantially improve the starting model, reaching about 0.70 at the common horizon $R_{\text{common}} = 22$ (Table 1). The research agent can therefore discover effective changes to the model and training procedure through iterative experimentation, even under the simplest greedy policy.

Maintaining several active trajectories does not raise this level. Figure 2 shows that K=3 advances faster in early rounds and leads the mean frontier until roughly round 12. This early lead is obtained with three times as many training runs per round, and it does not translate into a higher plateau:

Table 2: Research-process statistics over the common research-round horizon $R_{\text{common}} = 22$. Coverage reports the number of intervention families attempted, and exploratory entropy measures how broadly experiments are distributed across those families. Maximum lineage depth measures the longest sequence of parent–child states, while frontier-advancing rounds count research rounds that improve the best-so-far research artifact.

	$K = 1$				$K = 3$			
Metric	Run 1	Run 2	Run 3	Mean ± SD	Run 1	Run 2	Run 3	Mean ± SD
Intervention-family coverage	7	6	7	6.67 ± 0.58	8	8	8	8.00 ± 0.00
Exploratory entropy	0.879	0.835	0.833	0.849 ± 0.026	0.906	0.954	0.884	0.915 ± 0.036
Maximum lineage depth	9	7	7	7.67 ± 1.15	6	6	6	6.00 ± 0.00
Frontier-advancing rounds	10	6	8	8.00 ± 2.00	5	8	7	6.67 ± 1.53

Table 3: State-dependent effect of false-negative filtering. The same intervention is applied from different parent states and changes from harmful to beneficial and back to harmful as the surrounding training configuration changes. These comparisons provide descriptive evidence of state-dependent intervention effects rather than controlled causal estimates.

Parent state	Parent score	Child score	Δ	Outcome
Symmetric InfoNCE, original batch sizes	0.650105	0.647728	-0.002377	Harmful
After image batch increase	0.685797	0.691399	$+0.005602$	Beneficial
After larger video batch	0.697803	0.694662	-0.003141	Harmful

every K=3 run stops improving by round 12, and at R_{common} the best K=3 run (0.703) falls below the median K=1 run (0.707). The plateau is not a hard ceiling of the setting, since one greedy run reaches 0.722.

The gains of about 0.125 over the starting model are more than two orders of magnitude larger than training-seed noise, which is below 0.001 for a fixed configuration. Wider search does make outcomes more consistent across runs (standard deviation 0.001 vs. 0.018): the spread of K=1 outcomes, more than 18 times the seed noise, reflects divergent research paths, whereas the K=3 runs converge to nearly the same score. It does not, however, raise the final score. With three runs per policy we do not claim that the two policies are equivalent, rather, under a comparison designed to favor the wider policy, we observe no evidence that it produces a stronger final frontier.

5.2 Wider Search Produces Greater Exploratory Breadth

The absence of a performance advantage from $K = 3$ is not due to the two policies conducting the same research. Their research histories exhibit meaningfully different exploration patterns.

Table 2 shows that increasing trajectory width changes the research process itself. The experiments attempted by the K=3 policy cover more intervention families and are distributed more evenly across them, so wider search produces genuinely broader exploration rather than reproducing the behavior of the single-trajectory policy. The retained set is also diverse rather than collapsing to near-duplicates of a single state: in every round, the three active states come from at least two distinct lineages. Wider search therefore preserves genuinely different research states; as we show below, the limitation lies in exploiting them.

The lineage statistics show where the additional experiments go. Despite evaluating more candidates, the K=3 policy develops shallower lineages (maximum depth 6.00 vs. 7.67) and advances the frontier in fewer rounds (6.67 vs. 8.00): its additional experiments are spread across branches rather than deepening a single line of development.

5.3 Intervention Effectiveness Depends on Research State

The research histories provide one indication of why additional states are difficult to exploit effectively: the value of an intervention depends on the state from which it is applied.

Table 4: Attempted and productive intervention-family coverage at the common research-round horizon $R_{\text{common}} = 22$. Entries report C_r/P_r: the number of intervention families attempted and the number containing at least one intervention that improves its attributable parent or matched control.

Policy	Run 1	Run 2	Run 3	Mean C_r/P_r
Single trajectory ($K = 1$)	7/5	6/4	7/3	6.67/4.00
Multi-trajectory ($K = 3$)	8/3	8/4	8/5	8.00/4.00

As shown in Table 3, the effect of false-negative filtering changes from harmful to beneficial and then back to harmful as the surrounding training state changes.

Each entry is a single training run, but training-seed noise is below 0.001, so the change in effect across parent states, from -0.002 to $+0.006$, is several times larger than training noise. The value of an intervention is therefore not a fixed property of the intervention alone; it depends on the surrounding model and training configuration. This matters for wider search: preserving additional states creates more research opportunities, but those states are useful only if the planner can match appropriate follow-up interventions to each of them.

5.4 Wider Search Is Not More Productive

To distinguish exploratory breadth from useful exploration, Table 4 compares attempted and productive family coverage, C_r and P_r (Section 4.3).

Table 4 shows that increasing trajectory width clearly increases exploratory breadth. The $K = 3$ policy attempts all eight intervention families in every run, compared with 7, 6, and 7 for $K = 1$. However, this additional coverage does not produce additional productive families: both policies discover an average of exactly 4.00 productive families per run. Across all three runs, $K = 3$ therefore explores 24 family–run combinations compared with 20 for $K = 1$, while both policies produce 12 productive family–run combinations.

This result distinguishes *search breadth* from *productive search breadth*. Wider search generates more research directions, but the additional directions are not consistently converted into families that improve their local parent state. The state-dependent results in Table 3 help explain why this conversion is difficult: the same false-negative-filtering intervention changes from unfavorable to favorable and back to unfavorable when applied from different parent states. Thus, retaining more states increases the number of research opportunities, but exploiting them requires the planner to identify appropriate state–intervention combinations.

We do not interpret this analysis as establishing a unique causal mechanism. Rather, it shows that the lack of a consistent performance advantage from wider search is not caused by a lack of exploratory breadth: the wider search does explore more broadly, but that additional breadth does not increase productive-family coverage in the evaluated runs.

6 Conclusion

We study how much search control autonomous post-training needs by comparing a greedy single-trajectory policy (K=1) with a score-based multi-trajectory policy (K=3) under a fixed research agent and experimental environment. Autonomous post-training substantially improves the starting model under both policies, but wider search does not produce a higher final frontier, even under a round-aligned comparison in which it trains three times as many candidates. The wider policy retains diverse states and explores more broadly, yet discovers no more productive intervention families, and intervention effects depend on the state from which they are applied. These results suggest that the bottleneck for wider search in autonomous model research is not generating alternative research states but exploiting them, motivating state-conditioned proposal and selection mechanisms for autonomous research agents.

AI USE STATEMENT

Generative AI tools were used both as part of the research methodology and to assist the authors during preparation of this work. The autonomous research agent evaluated in this study uses Claude Opus 4.6 as its research planner. As part of the experimental system, the model proposes and refines research hypotheses, designs experiments, suggests and implements code-level interventions, interprets experimental results, and proposes subsequent research actions. These activities constitute the object of study described in the paper.

The authors additionally used ChatGPT to assist with research discussion, organization and interpretation of experimental results, generation and revision of analysis scripts and figures, and drafting and editing portions of the manuscript. All reported experimental results and numerical values were checked against the underlying experiment logs and evaluation outputs, and the authors reviewed and revised AI-assisted text and analyses. The authors take responsibility for the final content of the paper, including all claims, analyses, and artifacts produced with the assistance of generative AI.

REFERENCES

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. URL http://jmlr.org/papers/v13/bergstra12a.html.

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. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. URL http://jmlr.org/papers/v20/18-598.html.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks, 2017. URL https://arxiv.org/abs/1711.09846.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code, 2025a. URL https://arxiv.org/abs/2502.13138.

Ziyan Jiang, Rui Meng, Xinyi Yang, Semih Yavuz, Yingbo Zhou, and Wenhu Chen. Vlm2vec: Training vision-language models for massive multimodal embedding tasks. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 1255–1279, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/04261fce1705c4f02f062866717d592a-Paper-Conference.pdf.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. URL http://jmlr.org/papers/v18/16-558.html.

Zexi Liu, Yuzhu Cai, Xinyu Zhu, Yujie Zheng, Runkun Chen, Ying Wen, Yanfeng Wang, Weinan E, and Siheng Chen. ML-Master: Towards AI-for-AI via integration of exploration and reasoning, 2025. URL https://arxiv.org/abs/2506.16499.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

Rui Meng, Ziyan Jiang, Ye Liu, Mingyi Su, Xinyi Yang, Yuepeng Fu, Can Qin, Zeyuan Chen, Ran Xu, Caiming Xiong, Yingbo Zhou, Wenhu Chen, and Semih Yavuz. VLM2Vec-V2: Advancing multimodal embedding for videos, images, and visual documents, 2025. URL https://arxiv.org/abs/2507.04590.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.320. URL https://aclanthology.org/2025.findings-emnlp.320/.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Lupidi, Andrei Lupu, Roberta Raileanu, Tatiana Shavrina, Kelvin Niu, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Lars Erik Saito Stenetorp, Nicola Cancedda, Jakob Foerster, and Yoram Bachrach. Ai research agents for machine learning: Search, exploration, and generalization in mle-bench. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 35309–35348. Curran Associates, Inc., 2025. doi: 10.52202/085713-1186. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/328b81881da145412f2bc56c998dfb6a-Paper-Conference.pdf.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution, 2024. URL https://arxiv.org/abs/2409.12191.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-level automated scientific discovery via agentic tree search, 2025. URL https://arxiv.org/abs/2504.08066.

Xu Yang, Xiao Yang, Shikai Fang, Yifei Zhang, Jian Wang, Bowen Xian, Qizheng Li, Jingyuan Li, Minrui Xu, Yuante Li, Haoran Pan, Yuge Zhang, Weiqing Liu, Yelong Shen, Weizhu Chen, and Jiang Bian. R&D-Agent: An LLM-agent framework towards autonomous data science, 2025. URL https://arxiv.org/abs/2505.14738.

A Experimental Details

A.1 Model and Training Configuration

We study autonomous post-training of a VLM2Vec-Qwen2-VL-7B multimodal embedding model initialized from a checkpoint already adapted from Qwen2-VL-7B for contrastive representation learning. The model produces a single embedding vector per input using last-token pooling followed by ℓ_2 normalization, with cosine similarity used for retrieval.

The initial post-training configuration uses one-directional in-batch InfoNCE from query to target with temperature $\tau = 0.02$ and cross-device gathering of in-batch negatives. The language-model projections are adapted using LoRA with rank $r = 32$ and scaling parameter $\alpha = 32$, while the vision tower is initially frozen.

Training draws from image, visual-document, and video streams with sampling weights 1.0, 0.5, and 0.5, respectively. The initial per-device batch size is 4 for each stream. The default configuration

Table A.1: Initial model and training configuration shared by all autoresearch runs. These values define the starting research state and may subsequently be modified by the research agent.

Component	Initial configuration
Model	VLM2VEC-Qwen2-VL-7B
Embedding readout	Last-token pooling
Embedding normalization	ℓ_2 normalization
Similarity	Cosine similarity
Training objective	One-directional in-batch InfoNCE
Temperature	$\tau = 0.02$
Negatives	Cross-device in-batch negatives
Adaptation	LoRA on language-model projections
LoRA rank	$r = 32$
LoRA scaling	$\alpha = 32$
Vision tower	Frozen
Image sampling weight	1.0
Visual-document sampling weight	0.5
Video sampling weight	0.5
Per-device batch size	4 per stream
Optimizer	AdamW
Learning rate	2×10^{-4}
Learning-rate schedule	Constant
Training length	1000 steps
Compute	Four compute nodes

trains for 1000 steps using AdamW with a constant learning rate of 2×10^{-4} on four compute nodes. Table A.1 summarizes the initial configuration.

During autoresearch, the agent may modify the editable model and training pipeline, including the training objective, negative construction, batch and data composition, optimization strategy, parameter-efficient adaptation, prompt or input processing, and model components. The evaluation benchmark, scoring procedure, and required output interface remain fixed. All candidate models must produce a single ℓ_2-normalized embedding vector per input.

Across trajectory-width conditions, we use the same starting checkpoint, planner configuration, training environment, available tools, evaluation pipeline, and persistent research-memory mechanism. Each independent autoresearch run begins from the same starting checkpoint with fresh research memory.

A.2 Evaluation Tasks and Metrics

We evaluate every candidate model on a fixed 12-task subset of MMEB-V2, containing one representative task from each benchmark subcategory. The subset covers image classification, visual question answering, image retrieval, visual grounding, video classification, video question answering, video retrieval, moment retrieval, and visual-document retrieval.

For the eight image and video tasks, performance is measured using `hit@1`; for the four visual-document tasks, we use `ndcg_linear@5`. Table A.2 lists the complete evaluation suite and aggregation weights.

The aggregate score for model M is

$$S(M) = \frac{1}{78} \sum_{c=1}^{12} w_c s_c(M),$$

where $s_c(M)$ is the score of model M on task c and the weights w_c sum to 78. The common starting checkpoint achieves

$$S(M_0) = 0.576474.$$

Table A.2: Fixed 12-task MMEB-V2 evaluation subset used throughout the study. Tasks, metrics, and aggregation weights are identical across all policies and runs.

Category	Task	Metric	Weight
I-CLS	ImageNet-1k	hit@1	10
I-QA	OK-VQA	hit@1	10
I-RET	MSCOCO i2t	hit@1	12
I-VG	RefCOCO	hit@1	4
V-CLS	Kinetics-700	hit@1	5
V-QA	ActivityNet-QA	hit@1	5
V-RET	MSR-VTT	hit@1	5
V-MRET	QVHighlight	hit@1	3
ViDoRe-V1	arXivQA	ndcg_linear@5	10
ViDoRe-V2	ESG Multilingual	ndcg_linear@5	4
VisRAG	arXivQA	ndcg_linear@5	6
VisDoc-OOD	VidoSeek Page	ndcg_linear@5	4
		Total	78

Every subsequently produced candidate is evaluated using the identical task set, metrics, weights, and aggregation procedure.

B PERSISTENT MODEL REPOSITORY AND RESEARCH MEMORY

The autoresearch system maintains two complementary forms of persistent state: an executable *model repository* and a semantic *research memory*.

Model repository. The model repository stores executable research artifacts produced during the search. For each evaluated model state, it records the trained checkpoint, associated code and configuration, evaluation results, and parent relationship. The repository is initialized with the common starting model M_0 and retains evaluated artifacts even after they leave the active trajectory set.

The active set $\mathcal{B}_t$ therefore does not represent the complete set of models discovered during the run. Rather, it identifies the subset of model states that remain eligible for direct continuation in the next research round.

Research memory. Research memory stores the semantic evidence accumulated during the run, including hypotheses, attempted interventions, observed outcomes, interpretations, and previous failures. This information remains available to the planner even when the corresponding model is no longer active.

This separation is important for the trajectory-width comparison. Increasing K does not simply give the wider policy access to more historical information: both policies retain access to their accumulated experimental evidence. Instead, K controls how many executable model states remain available as direct parents for future interventions.

For example, under $K = 1$, an unsuccessful alternative can still inform later hypothesis generation even though subsequent training cannot continue directly from that checkpoint. Under $K = 3$, several such states may remain simultaneously available for direct continuation. The wider policy therefore changes the set of actionable research states while preserving the same general mechanism for remembering previously observed evidence.

C ADDITIONAL RESEARCH-PROCESS ANALYSIS

C.1 INTERVENTION-FAMILY TAXONOMY

Each valid training intervention is assigned to one primary intervention family using the same taxonomy for both search policies. The taxonomy characterizes the primary research mechanism tested by an experiment; it does not define a closed search space. The research agent remains free to formulate and implement new code-level interventions during autoresearch.

Table C.1: Intervention-family taxonomy used for the research-process analysis. Examples illustrate interventions observed in the autoresearch histories and are not an exhaustive specification of the search space.

Intervention family	Representative interventions
Objective / loss design	Contrastive-loss direction, temperature, margin-based objectives, hardness weighting, distillation objectives, and other changes to the training loss.
Negative construction	False-negative filtering, duplicate-target masking, mined hard negatives, and changes to the construction or weighting of contrastive negatives.
Batch composition	Changes to image, video, or visual-document batch sizes and the resulting in-batch negative pool.
Data mixture	Changes to stream sampling weights, data balance, curriculum, or inclusion of training-data sources.
Optimization / scheduling	Learning rate, training horizon, checkpoint selection, warmup or decay schedule, gradient clipping, and related optimization changes.
Parameter-efficient adaptation	LoRA rank or scaling, adapted-module selection, vision-tower adaptation, and alternative parameter-efficient adaptation mechanisms.
Prompt / input processing	Prompt or instruction changes, input formatting, and other modifications to the representation presented to the encoder.
Pooling / architecture	Pooling and readout changes, learned heads, intermediate-layer readout, and other architectural modifications to the embedding model.

When an experiment changes multiple implementation details, we assign it to the family corresponding to the primary hypothesis being tested. The same assignment procedure is used for both $K = 1$ and $K = 3$.

For the process analysis, we include successfully executed training interventions that produce a valid evaluable candidate. Matched controls, seed-only replications, evaluation-only combinations such as soups or ensembles, experiments that fail before the intended intervention is executed, and experiments whose evaluations are known to be invalid or contaminated are excluded. This prevents infrastructure operations and repeated measurements from being counted as additional research directions.

C.2 Productive-Family Coverage

Greater intervention-family coverage is useful only if the additional research directions produce useful model changes. We therefore distinguish *attempted breadth* from *productive breadth*.

Let $\mathcal{I}_r(R)$ denote the set of valid non-control training interventions evaluated in run r through round R. For intervention i, let M_i denote the resulting candidate, Q_i its attributable parent state or matched control, and f_i its primary intervention family.

We call family f *productive* in run r if at least one intervention from that family produces a positive observed change relative to its attributable reference. Productive-family coverage is therefore

$$P_r(R) = \left|\{f \in \mathcal{F} : \exists i \in \mathcal{I}_r(R),\ f_i = f \wedge S(M_i) > S(Q_i)\}\right|.$$

We use a parent-relative criterion rather than requiring the intervention to advance the global best-so-far frontier. This distinction is important for the multi-trajectory policy: an intervention can improve an alternative retained state without immediately exceeding the strongest model elsewhere in the active set.

Despite the difference in attempted breadth, productive-family coverage is identical on average. For $K = 1$,

$$(C_r, P_r) = (7, 5),\ (6, 4),\ (7, 3),$$

whereas for $K = 3$,

Table C.2: Per-run intervention-family productivity audit through $R_{\text{common}} = 22$. C_r denotes the number of intervention families attempted and P_r the number containing at least one valid training intervention with a positive observed score change relative to its attributable parent or matched control.

Policy	Run	C_r	P_r	Productive intervention families
$K = 1$	1	7	5	Objective/loss; negatives; batch; PEFT; prompt/input
$K = 1$	2	6	4	Objective/loss; negatives; batch; prompt/input
$K = 1$	3	7	3	Objective/loss; optimization/scheduling; PEFT
$K = 3$	1	8	3	Objective/loss; negatives; batch
$K = 3$	2	8	4	Objective/loss; negatives; PEFT; prompt/input
$K = 3$	3	8	5	Objective/loss; data mixture; optimization/scheduling; PEFT; prompt/input

$$(C_r, P_r) = (8, 3),\ (8, 4),\ (8, 5).$$

Hence,

$$\frac{1}{3}\sum_r C_r = 6.67 \quad \text{for } K = 1, \qquad 8.00 \quad \text{for } K = 3,$$

while

$$\frac{1}{3}\sum_r P_r = 4.00$$

for both policies.

Across all three runs, the single-trajectory policy therefore explores

$$7 + 6 + 7 = 20$$

family–run combinations, while the multi-trajectory policy explores

$$8 + 8 + 8 = 24.$$

However, both policies contain the same number of productive family–run combinations,

$$5 + 4 + 3 = 3 + 4 + 5 = 12.$$

The additional family coverage produced by $K = 3$ therefore does not yield additional productive-family coverage in these runs. This is the family-level evidence underlying the distinction in Section 5.4 between *exploration* and *productive exploration*.

The productivity labels above are descriptive rather than statistical significance tests. In particular, a small positive observed difference may fall within experimental variation. The purpose of this analysis is therefore not to claim that every positive intervention is a statistically resolved improvement, but to test whether the greater family coverage produced by $K = 3$ is accompanied by an evident increase in productive research directions. It is not.

Figure C.1 visualizes the per-run statistics.

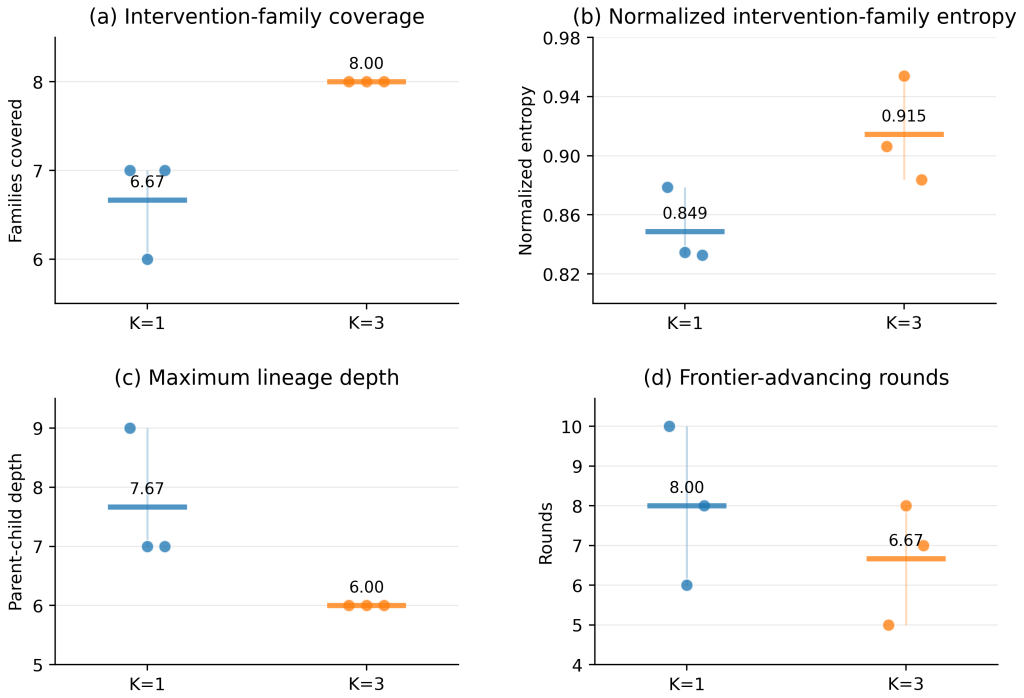

Figure C.1: Research-process characteristics at the common research-round horizon $R_{\text{common}} = 22$. Each point denotes an independent autoresearch run and each horizontal marker denotes the policy mean. The top row characterizes exploratory breadth: (a) intervention-family coverage and (b) normalized intervention-family entropy. The bottom row reports secondary trajectory-structure statistics: (c) maximum parent–child lineage depth and (d) the number of frontier-advancing research rounds. For $K = 3$, an unordered research round contributes at most one frontier advance even when multiple candidates are evaluated within the round. These measurements characterize the realized research behavior and are not interpreted as direct measures of search quality.

D Discovered Recipes and Research Artifacts

The autonomous research process produces two related but distinct outputs. First, it discovers *training recipes*: configurations that produce a single trained model. Second, it may construct a *research artifact* from one or more previously trained models, for example through checkpoint averaging, model souping, or embedding-space ensembling.

This distinction matters because the highest-scoring evaluated artifact in a run is not always the highest-scoring single trained model. We therefore report both separately. All results in this appendix are restricted to the common research-round horizon $R_{\text{common}} = 22$, consistent with the primary policy comparison. Unless noted otherwise, configuration values not listed below retain the starting values given in Appendix A.1.

D.1 Retained Single-Model Recipes

Table D.1 summarizes the strongest or retained single-model recipe identified in each run through the common horizon. The configuration column reports changes relative to the common starting configuration rather than reproducing the complete training configuration.

These recipes illustrate that the autonomous agent does not converge to a single common post-training configuration. For example, one run emphasizes stream-specific batch composition, another changes adapter placement and objective weighting, and another primarily changes the optimization trajectory. The resulting recipes should therefore be interpreted as products of the corresponding

Table D.1: Retained single-model recipes discovered through $R_{\text{common}} = 22$. "Score" is the aggregate score of the corresponding single trained model and excludes evaluation-only model combinations. Configuration descriptions list the principal changes from the common starting configuration; unspecified settings retain their initial values.

Policy	Run	Score	Principal discovered configuration
$K = 1$	1	0.710590	Frozen-base representation anchor; train/eval pixel-budget and video-input alignment; mined visual-document hard negatives; representation-prompt suffix; gated false-negative handling; vision LoRA with gradient clipping; image-stream batch size increased to 16.
$K = 1$	2	0.707380	Image-stream batch size 8; train/eval pixel-budget alignment; 8-frame video input; duplicate-target false-negative masking; representation-prompt suffix; symmetric query–target / target–query InfoNCE.
$K = 1$	3	0.686364	Hardness-weighted InfoNCE; attention-only LoRA in both the vision and language towers; LoRA rank/scaling $64/64$; 400-step cosine-annealed optimization with peak learning rate 2×10^{-4}.
$K = 3$	1	0.701703	Symmetric InfoNCE; image-stream batch size 8; video-stream batch size 12; visual-document batch size retained at its smaller value; contrastive temperature retained at $\tau = 0.02$.
$K = 3$	2	0.688857	Base-embedding regularization anchor; contrastive temperature $\tau = 0.05$; vision-tower LoRA; combined pointwise and relational representation anchoring.
$K = 3$	3	0.698782	Improved InfoNCE with false-negative handling; global representation suffix; reduced learning rate; train/eval visual-resolution alignment; warmup followed by cosine learning-rate annealing.

Table D.2: Best evaluated research artifact through $R_{\text{common}} = 22$. Combined artifacts reuse previously trained models; their construction does not require an additional SFT training call.

Policy	Run	Single model	Best artifact	Artifact construction
$K = 1$	1	0.710590	**0.722344**	Four-member delta-space model soup.
$K = 1$	2	**0.707380**	**0.707380**	Single trained model.
$K = 1$	3	**0.686364**	**0.686364**	Single trained model.
$K = 3$	1	**0.701703**	**0.701703**	Single trained model.
$K = 3$	2	0.688857	**0.700184**	Five-member embedding-space ensemble.
$K = 3$	3	0.698782	**0.702827**	Equal-weight pairwise model soup of two compatible training trajectories.

autonomous research histories rather than as independently tuned points from a predefined hyperparameter grid.

Several of the reported score differences are smaller than the measured run-to-run variation. We therefore report the recipes that were retained by the research policy rather than interpreting every numerical promotion as a statistically resolved improvement. In particular, the purpose of Table D.1 is to make the discovered configurations reproducible, not to rank individual recipe components by causal importance.

D.2 Best Evaluated Research Artifacts

The best evaluated output of a research run can differ from its retained single-model recipe because the agent is allowed to operate on previously produced checkpoints after training. Table D.2 reports the artifact used for the primary round-based comparison.

Delta-space model soup. In $K = 1$ Run 1, directly averaging the low-rank LoRA factors is inappropriate because their factorization is not unique. Instead, the agent reconstructs each member's effective parameter update, averages these updates in parameter-delta space, and applies the result to

the common frozen base model. If an adapter site is absent from a particular member, that member contributes no update at that site. This construction raises the Run-1 artifact frontier from the best individual training recipe to 0.722344 by round 18 without another SFT training call.

Embedding-space ensemble. In $K = 3$ Run 2, combining independently trained models in parameter space is less effective than combining their representations. The agent averages the normalized embeddings produced by the ensemble members and renormalizes the resulting vector, preserving the fixed single-vector evaluation interface. Expanding the ensemble to five strong members produces an aggregate score of 0.700184. Unlike a weight-space soup, however, this artifact requires one forward pass per ensemble member at inference time.

Pairwise checkpoint soup. In $K = 3$ Run 3, the best evaluated artifact is obtained by combining two compatible but differently trained models: a train/eval-resolution-aligned model and a deliberately decorrelated partner. The pairwise soup reaches 0.700179 at one checkpoint and is re-evaluated at nearby checkpoint positions; the highest measured artifact at step 440 scores 0.702827.

The neighbouring pairwise soups score similarly, so the particular checkpoint position should not be interpreted as a precisely resolved optimum. The important result is that the pairwise combination consistently exceeds either constituent single model in these evaluations.

These cases motivate reporting the *best evaluated research artifact* rather than silently equating the research frontier with the score of the most recently trained checkpoint. They also motivate separately reporting the single-model recipes in Section D.1, since different artifacts can have different deployment costs even when they satisfy the same evaluation contract.