CBM-SCIENTIST: AGENT-DRIVEN AUTOMATED CONCEPT BOTTLENECK MODEL DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Automated research agents hold significant promise for accelerating machine learning research, yet their utility in designing inherently interpretable neural networks remains understudied. We introduce CBM-Scientist, an agent-driven framework for automated concept bottleneck model (CBM) design. Agents iteratively propose, implement, and evaluate model and training changes, using experimental feedback to guide subsequent proposals. We study four proposal strategies across image classification, text generation, question answering, and image generation. Across these settings, agents develop recipes that improve multiple evaluated objectives simultaneously. For example, on AGNews, a loss-constrained recipe improves concept F1 by 15.10 % and steerability by 44.27 % while reducing perplexity from 96.23 to 43.17. Analysis of sprint trajectories shows how agents combine modifications, revisit unsuccessful ideas, and refine training procedures when improvements in one objective do not transfer to others. Together, these findings demonstrate the potential of autoresearch for concept bottleneck model development and highlight the importance of evaluating both task performance and concept use.

1 INTRODUCTION

Deep learning models have achieved strong predictive performance, but the basis of their decisions is often difficult to understand. Concept bottleneck models (CBMs) address this limitation by routing predictions through an explicit intermediate layer of human-interpretable concepts (Koh et al., 2020). For example, in a bird classification task, the concepts in a CBM may describe attributes such as wing color or beak shape. Through this concept interface, users can inspect predicted concepts, correct erroneous concept predictions, and observe how intervention affects the predicted species. CBMs have also been extended beyond image classification to text generation (Sun et al., 2025), and image generation (Ismail et al., 2024; Kulkarni et al., 2025), with efforts in automating concept construction and supervision using language models (Yang et al., 2023), vision-language models (Oikarinen et al., 2023; Rao et al., 2024), and grounding detectors (Srivastava et al., 2024).

A useful CBM should maintain task performance, predict intended concepts accurately, and respond appropriately when those concepts are corrected. Yet the central design decisions for CBMs remain largely manual: researchers must choose how concepts are represented and predicted, how downstream components use them, which losses to optimize, and how to train and intervene on the model. These choices are tightly coupled and must balance task performance, concept prediction, and intervention behavior, but improving one objective does not necessarily improve the others. For instance, soft concept values can carry information beyond their intended meanings and improve task prediction without establishing faithful concept use (Havasi et al., 2022), while accurate concept prediction alone does not ensure the correction of downstream predictions (Espinosa Zarlenga et al., 2023). Designing useful CBMs therefore requires navigating a large, feedback-intensive, multi-objective design space that is difficult to explore through manual iteration alone.

Autoresearch offers a way to explore this design space through executable experiments. Research agents propose changes, implement them in a working codebase, and use measured outcomes to guide subsequent experiments (Huang et al., 2024; Karpathy, 2026; Lu et al., 2026). Existing LLM-assisted concept-model development primarily evolves visual attribute descriptions (Chiquier et al., 2024) or refines concept banks and concept-class constraints (Jiang et al., 2025a). A recent agentic

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.

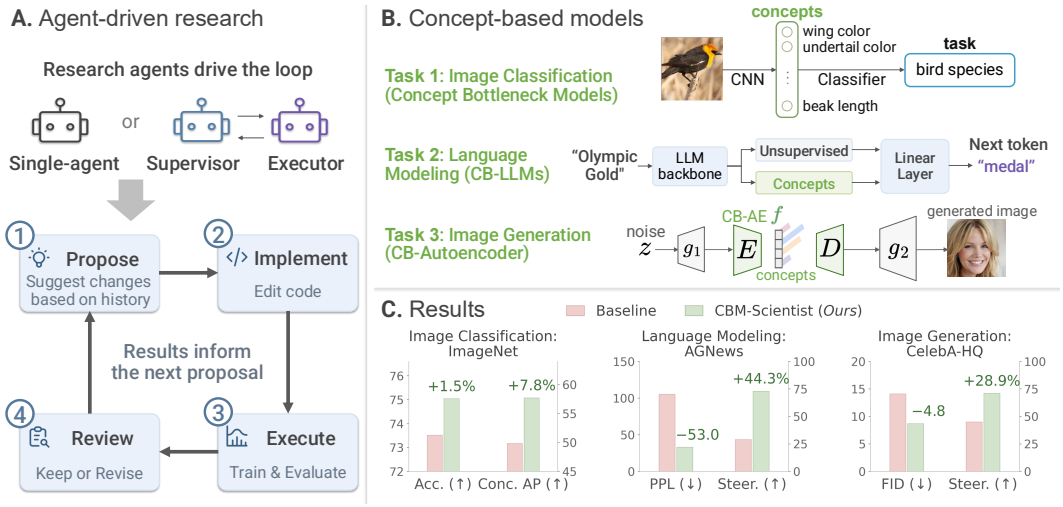


Figure 1: **CBM-Scientist overview.** (A) Agent-driven research loop. (B) CBM (Koh et al., 2020), CB-LLM (Sun et al., 2025), CB-AE (Kulkarni et al., 2025) for image classification, text generation, and image generation, respectively. (C) Results showing human developed baselines and the best agent-developed models on ImageNet, AGNews, and CelebA-HQ. Full results in Tables 1, 2, and 3.

study develops interpretable tabular models under predictive and agent-facing interpretability objectives (Singh et al., 2026), but does not study CBM design. Motivated by this gap, we study whether autoresearch agents can successfully navigate the coupled design choices of CBMs and discover recipes that jointly improve task performance, concept quality, and intervention behavior.

In this work, we introduce CBM-Scientist, an agent-driven research framework for *automated* CBM design. Starting from a baseline model and evaluation protocol, an agent proposes a change, implements it, trains and evaluates the resulting candidate, and reviews the outcome to guide the subsequent experiments. CBM-Scientist supports four proposal strategies: self-directed, loss-constrained, architecture-constrained, and supervisor-directed search. We instantiate the framework across three task groups spanning vision and language: image classification, text generation & question answering, and image generation. In each setting, the agent starts from a strong baseline that it must improve on. We evaluate the resulting models jointly on task performance, concept prediction, and intervention behavior. To our knowledge, this is the first systematic study of agentic autoresearch for CBMs across classification and generative model families.

Our contributions are:

1. **CBM-Scientist for autonomous CBM design.** We introduce an agent-driven research framework in which agents iteratively propose, implement, train, and review changes to neural CBMs. Across vision and language tasks, CBM-Scientist searches architecture, loss, and training choices under self-directed, constrained, and supervisor-guided strategies.
2. **Cross-domain evidence of joint gains.** We identify agent-developed recipes that improve all reported objectives within each task group. Representative joint gains span image classification on ImageNet (+1.52% ANEC-5 accuracy, +7.76% concept AP, +0.53% intervention accuracy), text generation on AGNews (+15.10% concept F1, +44.27% steerability, −53.06 PPL), question answering on FinQA (+3.19% answer accuracy, +38.86% concept F1, +3.31% intervention accuracy), and image generation on CelebA-HQ (+28.92% steerability, −4.76 reconstruction FID, −0.45 intervention FID). % gains denote absolute gains.
3. **Agent-driven search reveals CBM design insights.** We analyze recorded search histories across model families to connect agent proposals to multi-objective outcomes. These analyses reveal how agents can identify staged solutions to CBM tradeoffs, while also exposing cases in which improving one component does not improve downstream concept use. The AGNews case study illustrates this process through separate concept bottleneck and output-layer training followed by intervention-aware refinement.

2 Related Work

Concept Bottleneck Models. Concept bottleneck models embed human-interpretable concepts that users can inspect and edit during inference (Koh et al., 2020). Subsequently, Yuksekgonul et al. (2023) and Oikarinen et al. (2023) introduce post-hoc and label-free concept bottlenecks respectively, to transform pre-trained models into a concept bottleneck model. Subsequent methods ground concepts in spatial evidence (Srivastava et al., 2024; Benou & Riklin Raviv, 2025) or discover them from pretrained representations (Rao et al., 2024; De Santis et al., 2026). Another line of work studies concept interventions including expert-guided debugging (Enouen & Galhotra, 2026a), concept leakage (Havasi et al., 2022), intervention-aware learning (Espinosa Zarlenga et al., 2023), correction policies (Shin et al., 2023), interactive querying (Chauhan et al., 2023), and dependency-aware interventions (Vandenhirtz et al., 2024).

CBMs have also been extended to generative models. CBGM (Ismail et al., 2024) and CB-AE (Kulkarni et al., 2025) introduce editable concept interfaces into image generation, while subsequent methods explore probabilistic hard concepts (Martínez-García et al., 2026), concept-aligned diffusion (Enouen & Galhotra, 2026b), and energy-based steering (Kim et al., 2026). For language tasks, CB-LLM supports concept-based text generation (Sun et al., 2025), while CB-pLM applies concept bottlenecks to protein sequence generation (Ismail et al., 2025). CB-SAE extends concept-based steering to multimodal text and image generation by augmenting sparse autoencoders with user-defined concepts (Kulkarni et al., 2026). In natural-language generation, Bing & Shao (2026) show that the method used to construct inference-time concept codes can substantially affect control.

LLMs and agents for interpretable model design. Recent work has increasingly leveraged LLMs and agents to automate specific aspects of interpretable model design. For instance, LLM-Mutate uses an LLM as a mutation operator to evolve visual attribute descriptions (Chiquier et al., 2024), whereas Concept Agent employs an agent to refine concept banks and concept–class activation constraints within a fixed CoCoBM training procedure (Jiang et al., 2025a). Beyond concept-based models, DiSciPLE evolves interpretable programs over visual primitives using evaluation feedback, criticism, and simplification (Mall et al., 2025), and HACHI alternates agent-driven exploration of clinical concepts with expert feedback to build linear clinical prediction models (Feng et al., 2026). In automated discovery, recent autoresearch frameworks use agents to design interpretable tabular models (Singh et al., 2026). However, existing work either optimizes isolated components of the CBM pipeline or applies agents to fundamentally different model classes. In contrast, CBM-Scientist uses agents to end-to-end rewrite CBM architectures, objectives, and training strategies, guided by holistic task, concept, and intervention feedback.

Autoresearch benchmarks and systems. A growing body of work evaluates agents for ML experimentation and research engineering (Huang et al., 2024; Chan et al., 2025; Wijk et al., 2024) or develops systems that automate research workflows (Lu et al., 2024; Schmidgall et al., 2025). These systems explore different mechanisms for proposing, implementing, and selecting experiments, including iterative code modification, tree search, and evolutionary search (Karpathy, 2026; Jiang et al., 2025b; Yamada et al., 2025; Novikov et al., 2025). However, autoresearch has not yet been studied for end-to-end CBM design, nor is it understood how different forms of search guidance affect the CBM design choices that agents explore. CBM-Scientist addresses these questions by formulating CBM design as a multi-objective autoresearch problem and developing four different search strategies to guide agent exploration across classification and generation tasks.

3 CBM-Scientist: Agent-Driven Automated CBM Design

We present **CBM-Scientist** through three key components: the CBM design space available to the agent in Section 3.1, the shared propose-implement-execute-review autoresearch loop in Section 3.2, and the proposal strategies for guiding agent exploration in Section 3.3.

3.1 CBM Design Space

We study three established CBM families spanning image classification, language generation, and image generation. Each exposes a distinct design space of model components, objectives, and training procedures that the agent can modify, while keeping the underlying pretrained models, concept

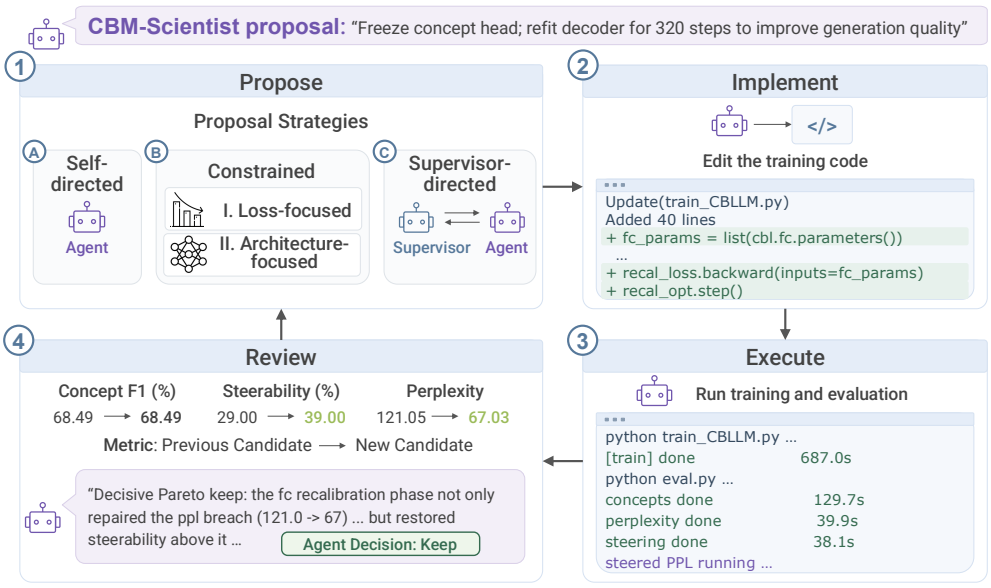


Figure 2: **CBM-Scientist research loop and proposal strategies.** Agents propose, implement, execute, and review candidates.

vocabularies, and evaluation protocols fixed where applicable. Appendix A provides detailed model formulations and baseline training objectives.

Task 1: CBMs for image classification. Our image classification baseline is VLG-CBM (Srivastava et al., 2024), which maps image-encoder features through an explicit concept bottleneck layer followed by a linear predictor. It trains the concept bottleneck layer using GroundingDINO pseudo-labels (Liu et al., 2024), then fits a sparse concept-to-class classifier. CBM-Scientist may modify the concept predictor, downstream training procedure, loss functions, and optimization strategy, while the image encoder, concept vocabulary, and evaluation sparsity budget remain fixed. We further demonstrate the applicability of CBM-Scientist to the original supervised CBM formulation (Koh et al., 2020) using annotated bird attributes (CUB CBM), where agent-driven search also improves upon the baseline. Its setup and results are provided in Appendix D.2.1.

Task 2: Concept bottleneck LLMs. We build our language baselines on the generation variant of CB-LLM (Sun et al., 2025), which predicts the next token from concept activations and a parallel unsupervised representation of language-model features. The parallel branch learns unsupervised concepts beyond the fixed concept set for token prediction; interventions edit concept activations while leaving these unsupervised concepts unchanged. During baseline training, we freeze the pretrained backbone weights and jointly train the backbone's LoRA adapters, concept-bottleneck layer, and output layer. Training combines concept supervision and token prediction with regularization that discourages the unsupervised branch from encoding known concepts. CBM-Scientist can modify the concept predictor, unsupervised branch, output mapping, loss functions, and training schedules.

Task 3: Concept bottleneck autoencoders for image generation. Our image generation baseline is CB-AE (Kulkarni et al., 2025), which inserts an autoencoder into a frozen image generator. The encoder separates intermediate generator features into concept activations and an unsupervised representation; the decoder reconstructs features from both. Editing the concept activations changes the generated image while retaining the unsupervised representation. Training combines reconstruction, concept-supervision, and intervention loss objectives. CBM-Scientist can modify the encoder, decoder, and the loss objectives while keeping the pretrained generator fixed.

3.2 Agent-Driven Research Loop

CBM-Scientist conducts automated CBM design through a sequential research loop in which an agent proposes model or training changes, implements and evaluates them, and uses the result-

ing feedback to guide subsequent experiments. We refer to one complete research run as a sprint. Following the iterative code-editing paradigm of autoresearch (Karpathy, 2026), each sprint is initialized with a `program.md` instruction file tailored to its task, dataset, proposal strategy, and agent organization. These instructions specify the experimental setup, permitted changes, evaluation procedure, and decision metrics (Appendix D.5). Evaluator modifications are prohibited by instruction rather than programmatically blocked. Within a sprint, CBM-Scientist agent develops and evaluates candidates sequentially, using recorded results to guide subsequent proposals.

At the start of a sprint, the CBM-Scientist agent evaluates the baseline implementation to establish reference scores. It then repeats the four steps in Fig. 2:

- **Step 1: Propose.** Review previous results and formulate a testable hypothesis. *E.g.*, for CB-LLM, the agent proposes freezing the concept bottleneck layer and retraining only the decoder to improve generation quality.
- **Step 2: Implement.** Modify the permitted code and configuration, verify the implementation, and commit the changes. *E.g.*, for CB-LLM, `train_CBLLM.py` is updated.
- **Step 3: Execute.** Train and evaluate the candidate using the prescribed procedure. *E.g.*, in CB-LLM, we measure concept F1, perplexity, steerability, and steered perplexity.
- **Step 4: Review.** Evaluate the candidate against the sprint objectives and validity requirements, then record a keep/reject decision with its rationale. Accepted changes are incorporated into the current implementation and carried forward to subsequent experiments, while rejected changes are discarded. *E.g.*, in Fig. 2, Step 4, the agent retains the change because perplexity decreases and steerability improves while concept F1 remains unchanged.

The CBM-Scientist agent repeats this loop, using experimental findings to refine subsequent proposals. Git commits track implementation changes, while experiment records retain candidate scores, descriptions, and keep/reject rationales, providing a complete trace of the agent's search trajectory.

3.3 Proposal Strategies

A central question in automated CBM design is how to guide agent exploration of the design space. We study four proposal strategies that vary the type and degree of guidance provided to the agent: self-directed, loss-constrained, architecture-constrained, and supervisor-directed search as shown in Fig. 2. All strategies use previous experimental results to inform subsequent proposals. The first three use a single-agent organization, whereas supervisor-directed search separates research guidance from execution through a supervisor–executor agent pair (Fig. 2, Panel 1.C).

Self-directed proposals (Fig. 2, Panel 1.A). A single agent chooses the research direction and carries out the research loop, following Karpathy (2026). We impose no additional restriction on the type of model or training change the agent may propose, beyond the sprint's data, evaluation, and implementation constraints. The program may suggest starting ideas, but the agent chooses which to pursue and may explore other architectures, objectives, and training strategies. *E.g.*, it may revise the concept predictor, then adjust downstream training if better concept predictions do not improve task performance.

Constrained proposals (Fig. 2, Panel 1.B). To encourage targeted exploration of CBM design choices, we constrain the class of modifications available to the agent while allowing autonomous experiment selection and refinement. These constraints are specified through agent instructions rather than enforced via code. We consider two forms of constrained search:

- **Loss-constrained (Fig. 2, Panel 1.B.I).** The agent is instructed to modify training objectives, their weights and schedules, and specified loss-related optimization settings, while keeping the architecture fixed. *E.g.*, reweighting rare concepts or replacing cross-entropy with focal loss.
- **Architecture-constrained (Fig. 2, Panel 1.B.II).** The agent is instructed to modify model components while preserving the prescribed objectives and concept interface. *E.g.*, adding a residual branch to the concept bottleneck, introducing gates, or changing normalization.

Supervisor-directed proposals (Fig. 2, Panel 1.C). Rather than fixing the scope of exploration in advance, supervisor-directed search provides adaptive guidance as experimental evidence accumulates. A supervisor agent reviews experimental outcomes and sets research directions for an executor

Table 1: VLG-CBM image classification on CUB and ImageNet. Absolute gains reach 5.90% in CUB intervention accuracy, and 1.52% in ANEC-5 and 7.76% in concept AP on ImageNet. Bold indicates improvement over baseline, blue indicates each agent's best result, and gray-shaded row indicates joint improvement over baseline in all 3 metrics.

	CUB			ImageNet		
Proposal	ANEC-5↑	Concept AP↑	Int. Acc.↑	ANEC-5↑	Concept AP↑	Int. Acc.↑
Baseline (w/o Agent)	75.67 ± 0.06	48.77 ± 0.07	85.20 ± 0.15	73.54 ± 0.03	49.83 ± 0.02	92.21 ± 0.17
GLM 5.3 (Ours)						
Self-directed	75.59 ± 0.06	**54.04** ± 0.03	**86.26** ± 0.35	**74.68** ± 0.01	**53.51** ± 0.04	**92.86** ± 0.15
Loss	**75.99** ± 0.15	**52.62** ± 0.11	**86.61** ± 0.34	**74.77** ± 0.03	**54.53** ± 0.02	**93.14** ± 0.06
Arch.	75.65 ± 0.10	**52.77** ± 0.12	**86.76** ± 0.30	**73.74** ± 0.06	**54.20** ± 0.33	91.97 ± 0.44
Supervisor	**75.85** ± 0.15	**53.47** ± 0.13	**91.10** ± 0.41	**73.89** ± 0.02	**51.03** ± 0.32	**92.64** ± 0.09
Sonnet 5 (Ours)						
Self-directed	**75.80** ± 0.13	**52.84** ± 0.03	**86.34** ± 0.14	**74.54** ± 0.07	**54.16** ± 0.03	**94.31** ± 0.05
Loss	**75.90** ± 0.25	**53.40** ± 0.08	**86.26** ± 0.43	**74.70** ± 0.03	**53.09** ± 0.02	**92.29** ± 0.14
Arch.	**76.00** ± 0.13	**52.48** ± 0.29	**86.28** ± 0.41	**74.22** ± 0.06	**53.80** ± 0.39	**92.41** ± 0.11
Supervisor	**75.79** ± 0.05	**52.79** ± 0.03	**86.76** ± 0.30	**75.06** ± 0.09	**57.59** ± 0.18	**92.74** ± 0.12

agent, which develops concrete proposals, implements them, and runs the experiments. This guidance can evolve during the sprint: *e.g.*, the supervisor may request refinement of a promising change, an ablation to test its contribution, or a shift between architecture and loss changes when progress stalls. This separates high-level research direction from implementation and experimentation: the supervisor's context focuses on experimental summaries and outcomes, while the executor handles implementation and execution details. The resulting guidance can therefore adapt throughout the sprint rather than being fixed a priori. See Appendix D.1 for more details.

As shown in Tables 1, 2, and 3, CBM-Scientist with these proposal strategies improves CBM designs across multiple objectives. We also explored subagent and episodic settings in Appendix D.

4 Experiments and Findings

We conduct a systematic evaluation across three CBM families and six datasets, spanning image classification with VLG-CBM, language generation with CB-LLM, and image generation with CB-AE. For CB-LLM, we consider both text generation and question answering. Our experiments address three questions: (1) Can agent-driven autoresearch discover CBM designs that jointly improve task performance, concept prediction, and intervention outcomes? (2) How do different proposal strategies affect the resulting CBM designs and performance? (3) What design ideas do agents explore, and how do these ideas evolve over the course of the search? Appendix B describes the task-specific baselines and evaluation protocols.

4.1 Setup

Each sprint starts from a baseline established before agent experimentation. We run the main experiments on a single NVIDIA A100 GPU per sprint. Our CBM-Scientist uses Claude Sonnet 5 (Anthropic, 2026c) and GLM 5.3 (Z.ai, 2026b), except on SST2, where the GLM sprints use GLM 5.2 (Z.ai, 2026a). Tables abbreviate Claude Sonnet 5 as Sonnet 5. Appendix B details baseline construction, training, candidate selection, and evaluation protocols.

Task 1: Image classification. We evaluate VLG-CBM (Srivastava et al., 2024) on CUB (Wah et al., 2011) and ImageNet (Deng et al., 2009). We report classification accuracy with a sparse readout averaging five nonzero weights per class (ANEC-5), concept macro AP, and accuracy after up to three positive-concept interventions. Appendix B.1 details training and evaluation; Appendix D.2.1 reports the additional CUB results for original CBM (Koh et al., 2020) (Table 7).

Task 2.1: Text generation. We extend CB-LLM (Sun et al., 2025) with Llama-3-8B (Grattafiori et al., 2024) to AGNews (Zhang et al., 2015) and SST2 (Socher et al., 2013), using 80 and 56 concepts, respectively. We report concept macro F1, generated-text perplexity (PPL) under a fixed

Table 2: Text generation and question answering on AGNews, SST2, and FinQA. Absolute concept-F1 gains reach 15.10%, 6.54%, and 38.86% respectively. The FinQA black-box reference is Qwen2.5-7B-Instruct from Zhu et al. (2025). Bold indicates improvement over baseline, blue indicates each agent's best result, gray-shaded row indicates improvement over baseline in all 3 metrics.

	AGNews			**SST2**			**FinQA**		
Black-box reference	Llama-3-8B: PPL 12.5			Llama-3-8B: PPL 84.7			Qwen2.5-7B-I: Acc. 66.7%		
Proposal	F1↑	Steer↑	PPL↓	F1↑	Steer↑	PPL↓	Ans.↑	F1↑	Int. Acc.↑
Baseline (w/o Agent)	58.7±0.4	26.7±1.5	96.2±19.1	58.6±1.3	24.8±5.2	47.9±9.6	65.2±1.1	39.8±1.7	65.4±1.1
GLM 5.3 (Ours)									
Self-directed	56.7±1.8	**89.9**±0.6	123.5±31.6	**60.6**±0.5	**27.0**±2.1	52.9±10.1	**68.4**±0.8	**78.6**±0.7	**68.7**±0.7
Loss	**73.8**±0.1	**71.0**±0.5	**43.2**±3.5	57.9±1.0	24.3±2.4	52.7±3.6	**65.8**±0.8	**63.9**±1.7	**66.4**±0.7
Arch.	**70.7**±0.4	**74.8**±6.4	**22.2**±1.1	**60.6**±0.8	**36.5**±1.9	63.8±19.6	65.1±3.9	**54.6**±1.8	**65.7**±3.9
Supervisor	**64.7**±0.3	**30.7**±3.3	104.9±18.4	**62.2**±0.3	**33.1**±1.4	59.5±14.9	64.9±1.2	**54.5**±1.0	**66.1**±1.2
Sonnet 5 (Ours)									
Self-directed	58.2±1.4	**31.4**±1.2	96.2±4.4	**61.7**±0.4	**25.7**±1.5	**42.3**±6.8	65.0±1.0	**70.4**±1.5	**65.7**±0.8
Loss	**58.9**±1.2	**32.9**±2.7	**62.1**±8.2	**65.2**±0.2	23.5±1.0	53.3±5.2	65.1±1.2	**57.2**±1.6	**65.5**±1.1
Arch.	57.6±1.0	**38.3**±2.6	106.3±4.8	**59.9**±0.4	**26.1**±1.7	53.9±9.1	**65.6**±1.3	**56.0**±2.8	**66.4**±1.3
Supervisor	**66.0**±0.5	**59.7**±3.8	**40.9**±4.4	**62.8**±0.3	**27.9**±5.0	50.7±9.3	**65.9**±3.9	**43.8**±0.4	**66.3**±3.6

external language model, and steerability: the fraction of steered texts expressing the target concept according to a fixed classifier. Appendix B.2 details training and evaluation.

Task 2.2: Question answering. We adapt Qwen2.5-7B-Instruct (Qwen et al., 2024) with a 45-concept bottleneck for FinQA (Chen et al., 2021), reporting concept macro F1 and answer accuracy from generated-program execution. Intervention accuracy is final answer accuracy after oracle corrections to up to three concepts in initially incorrect examples, retaining initially correct answers. Appendix B.3 details training, candidate selection, and full-test evaluation.

Task 3: Image generation. We use CB-AE (Kulkarni et al., 2025) with a frozen StyleGAN2 generator (Karras et al., 2020) on CelebA-HQ (Karras et al., 2018) and CUB (Wah et al., 2011). Reconstruction and intervention FID (Heusel et al., 2017) compare reconstructed and concept-edited images to real images. Steerability measures how often an edit introduces a target concept into images initially lacking it, as detected by a concept classifier and averaged across targets.

Note that for all metrics: Higher scores are better except for the metrics PPL and FID, where lower scores are better. Tables report mean ± sample standard deviation.

4.2 Quantitative Results

Tables 1–3 show joint improvements in task, concept, and intervention metrics across the task groups, alongside tradeoffs between recipes. To choose one candidate per sprint for reporting, we apply a task-specific weighted improvement score after the sprint. These reporting weights are not provided to the agents and are separate from their keep/reject decisions (Appendix B).

- **Image classification.** On ImageNet, Sonnet 5's supervisor-directed recipe improves ANEC-5 by 1.52% and concept AP by 7.76%, with higher intervention accuracy. Its self-directed recipe achieves the best intervention accuracy instead. On CUB, GLM 5.3's supervisor-directed VLG-CBM recipe improves all three metrics, including a 5.90% intervention gain.
- **Text generation.** On AGNews, GLM 5.3's loss-constrained recipe improves concept F1 by 15.10% and steerability by 44.27% while reducing PPL from 96.23 to 43.17. SST2 gains are smaller: Claude Sonnet 5's self-directed recipe improves all three means, while its loss-constrained recipe improves F1 at the expense of steering and PPL.
- **Question answering.** On FinQA, GLM 5.3's self-directed recipe raises answer accuracy from 65.2% to 68.4% and concept F1 by 38.9%. Its supervisor-directed recipe has lower F1 but better intervention accuracy than the baseline, showing that concept prediction and intervention gains need not be correlated.

Table 3: Image generation (CB-AE) on CelebA-HQ and CUB. Absolute steerability gains reach 28.92% and 6.71% respectively. Bold indicates improvement over the baseline, and blue indicates each agent's best result. Gray-shaded row indicates improvement over baseline in all 3 metrics.

	CelebA-HQ			CUB		
Proposal	Steer. (%, ↑)	Recon. FID (↓)	Interv. FID (↓)	Steer. (%, ↑)	Recon. FID (↓)	Interv. FID (↓)
Baseline (w/o Agent)	44.55 ±0.6	13.76 ±0.3	12.83 ±0.2	10.84 ±1.6	75.97 ±2.5	76.73 ±2.8
GLM 5.3 (Ours)						
Self-directed	**47.66** ±1.6	47.25 ±1.1	46.57 ±1.8	**16.91** ±3.5	78.66 ±1.7	79.51 ±1.8
Loss	**50.68** ±2.7	37.78 ±10.8	40.37 ±9.2	**15.90** ±2.4	**75.68** ±3.9	**76.60** ±3.7
Arch.	**49.45** ±1.2	36.85 ±12.9	37.87 ±9.3	**16.04** ±1.8	76.41 ±7.9	77.29 ±8.0
Supervisor	**55.70** ±1.5	19.11 ±4.4	19.76 ±4.1	**14.99** ±3.7	77.29 ±7.3	78.00 ±7.3
Sonnet 5 (Ours)						
Self-directed	**69.88** ±1.1	**8.60** ±0.2	13.67 ±0.2	**15.92** ±2.5	**64.46** ±3.3	**65.07** ±3.1
Loss	**58.06** ±2.7	16.35 ±1.2	18.23 ±1.0	**17.55** ±4.0	**72.60** ±3.7	**73.62** ±3.5
Arch.	**50.07** ±3.2	45.63 ±9.0	45.57 ±5.5	**16.79** ±5.6	**75.30** ±5.5	**76.20** ±5.6
Supervisor	**73.47** ±3.0	**9.00** ±0.1	**12.38** ±0.6	**14.47** ±2.6	81.94 ±1.2	82.60 ±1.0

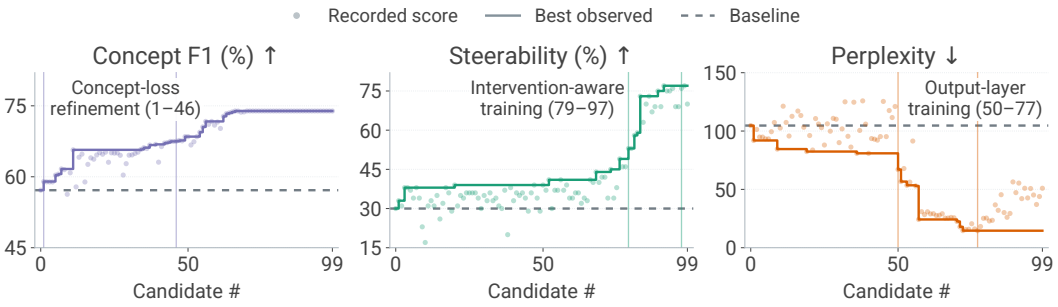


Figure 3: Progress of all 3 metrics during GLM 5.3's loss-constrained CB-LLM sprint on AGNews.

- **Image generation.** On CelebA-HQ, Claude Sonnet 5's supervisor-directed recipe improves steerability by 28.92% while reducing both FIDs. On CUB, its self-directed recipe improves all three metrics, whereas supervisor direction improves steerability but worsens both FIDs.

Appendix D.3 reports estimated agent cost per candidate across model families and datasets.

4.3 Recurring Ideas and Their Outcomes

Across model families, recurring ideas address concept prediction, downstream concept use, and training stability. Appendix D.4 reports how often they occur including unsuccessful trials; below we connect these directions to measured outcomes.

Image classification. On ImageNet, nonlinear concept-prediction branches, focal loss (Lin et al., 2017), positive/negative reweighting, and concept-score scaling raise concept AP from 49.83% to 57.59%, with higher classification and intervention accuracy. On CUB, we observe similar ideas being used in addition to intervention-aware training, leading to an improvement in all metrics.

CB-LLMs. On AGNews, balanced concept sampling, independent bottleneck and decoder training, and supervision under concept edits raise F1 from 58.69% to 73.79% and steerability from 26.73% to 71.00%, while reducing PPL from 96.23 to 43.17. FinQA's concept bottleneck refitting and decoder adaptation raise F1 from 39.76% to 78.62%, with higher answer and intervention accuracy.

CB-AE. For CelebA-HQ, a discriminator-based feature matching loss between original and intervened samples along with related hyperparameter tuning proved useful (steerability 45.9%→59.5%, recon. FID 34.7→15.3, interv. FID 33.5→17.2). For CUB, using a concept-balanced cross-entropy loss for both original and intervened samples along with a cyclic consistency loss improved steerability (13.4%→17.5%), recon. FID (74.74→72.60), and interv. FID (76.04→73.62). These are sprint-internal numbers, not baseline to final candidate results.

4.4 Case Study

To illustrate how an agent progresses through a sprint, we use GLM 5.3's loss-constrained AGNews sprint as an example (Figure 3). We observe that the agent's proposed changes to concept-frequency

penalties, ranking losses, and supervision at the final input position raised concept F1 from 57.16% to 67.51% and lowered PPL from 104.70 to 95.22 by candidate 46. The agent then trains only the concept bottleneck layer with balanced sampling (Kang et al., 2020). F1 improves to 68.49%, but PPL rises to 121.05 (candidate 49). Revisiting this rejected candidate, the agent proposes that the output layer also needs to adapt. Training that layer separately preserves F1 and lowers PPL to 67.03 (candidate 50). The agent then experiments with the duration and learning-rate schedule of each training stage. Longer training initially reduces F1, but shorter concept training followed by gradual learning-rate decay raises it from 71.70% to 72.77% by candidate 62. The agent retries 2,560 steps for the concept bottleneck with this schedule, keeping output training unchanged, and reaches 73.34% (candidate 63). An earlier failure thus guides a later proposal.

By candidate 76, further output training reduces PPL to 16.10, but steerability remains 49%. The agent returns to intervention-aware training, which had performed poorly when all components were trained jointly. Now applied during the separate output stage, extending supervision from one position to 16 raises steerability from 58% to 73%, while PPL increases from 22.93 to 36.74 (candidates 81 and 83). Later experiments use the model's own generated text as context, apply this loss where its concept predictions drift from the target, and regenerate these contexts during training. Adjusting the loss weights then reaches 73.90% F1, 76% steerability, and PPL 37.07 (candidate 97).

4.5 Reliability Across Repeated Searches

Independent repetitions. We compare three independent searches with Claude Sonnet 5 for each setting, keeping model-training seeds fixed. Table 4 reports the mean and standard deviation of their scores, measuring how results vary when the search is repeated.

Table 4: Evaluation metrics across 3 repeated CBM-Scientist sprints using the same training seed.

CUB: VLG-CBM			**AGNews: CB-LLM**		**CelebA-HQ: CB-AE**	
Metric	Loss	Supervisor-directed	Metric	Self-directed	Metric	Loss
ANEC-5 ↑	76.01 ± 0.18	75.72 ± 0.09	Concept F1 ↑	58.72 ± 0.48	Steerability ↑	56.38 ± 5.38
Concept AP ↑	52.98 ± 0.46	54.39 ± 1.39	Steerability ↑	37.00 ± 5.57	Recon. FID ↓	19.49 ± 8.37
Int. Acc. ↑	86.75 ± 0.25	86.30 ± 0.27	PPL ↓	86.70 ± 4.12	Interv. FID ↓	20.99 ± 7.79

Consistent improvements on VLG-CBM (CUB). Every VLG-CBM CUB sprint finds a candidate that improves classification accuracy, concept prediction, and intervention accuracy over its baseline. Loss-constrained sprints reach similar scores across all three metrics. Supervisor-directed sprints also produce similar classification and intervention accuracy, but differ more in concept AP: the two repeats reach 55.05% and 55.32%, compared with 52.79% in the original sprint.

Modest improvements on CB-LLM (AGNews). All three sprints improve concept F1, steerability, and PPL over the baseline. Their approaches differ: one changes how concept-predictor gradients are computed, another adds a nonlinear concept-prediction branch, and the third aligns concept and intervention-aware supervision across token positions. Concept F1 and PPL are comparatively stable across these recipes, while steerability ranges from 31% to 42%.

Variations in CB-AE (CelebA-HQ). All three sprints improved steerability significantly while FID was worse. In two of these, CBM-Scientist trained a discriminator model and applied consistency losses between original and intervened samples, with one run operating in image space while the other was in the latent space. These two sprints achieved similar steerability (59.69%, 59.28%) and similar FID (14.78, 14.54). However, the third sprint explored vastly different ideas like a VGG16-based perceptual loss, changing ℓ_2 reconstruction to ℓ_1, *etc*. (steerability 50.18%, FID 29.15) yielding lower improvement than the other two sprints. Overall, while FID scores vary moderately, all three sprints significantly improved steerability over the baseline (44.55%, Table 3).

5 Conclusion

We introduced CBM-Scientist, an autoresearch framework for improving CBM architectures and training across vision and language. Agents develop recipes that jointly improve task performance, concept prediction, and intervention outcomes. On AGNews, one recipe increases concept F1 by 15.10 percentage points and steerability by 44.27 points while reducing perplexity from 96.23 to 43.17. Search trajectories show how agents combine ideas and develop staged training procedures, highlighting the importance of evaluating both concept prediction and downstream use.

AI USE STATEMENT

LLM agents are central to the method studied in this paper: they propose changes to CBM architectures, objectives, and training procedures, implement these changes, run experiments, and interpret feedback, as described in Section 3. LLMs also assisted concept vocabulary construction and labeling, as detailed in Appendix B. Beyond these experiments, we used AI tools to assist with coding, analyzing and interpreting sprint records and parts of manuscript drafting and revision. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 describes the agent workflow, editable model components, and proposal strategies. Appendix A specifies the baseline architectures and objectives, while Appendix B documents datasets, concept set construction and labelling, training settings, candidate selection, and evaluation protocols in detail. Appendix C summarizes the best-performing model recipes for each dataset. Appendix D.5 provides a representative ImageNet `program.md`, including task constraints, suggested research directions, and the experiment loop. We report results across training seeds and assess variation across repeated searches in Section 4.5. We will release the codebase and trained model checkpoints at a later point to support reproduction and further analysis.

REFERENCES

Anthropic. Claude API pricing, 2026a. URL https://platform.claude.com/docs/en/about-claude/pricing. Accessed September 25, 2026.

Anthropic. Introducing Claude Opus 5, 2026b. URL https://www.anthropic.com/news/claude-opus-5. Accessed September 23, 2026.

Anthropic. Introducing Claude Sonnet 5, 2026c. URL https://www.anthropic.com/news/claude-sonnet-5. Accessed September 23, 2026.

Itay Benou and Tammy Riklin Raviv. Show and tell: Visually explainable deep neural nets via spatially-aware concept bottleneck models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30063–30072, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Benou_Show_and_Tell_Visually_Explainable_Deep_Neural_Nets_via_Spatially-Aware_CVPR_2025_paper.html.

Qi Bing and Xiaowei Shao. The illusion of control: Why bare classifier inversion silently fails in concept-bottleneck text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2026. URL https://arxiv.org/abs/2608.22956. Accepted, main conference.

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 *International Conference on Learning Representations*, volume 2025, pp. 50466–50494, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/7e3767db483c942b883eb4f8cfb74e31-Paper-Conference.pdf.

Kushal Chauhan, Rishabh Tiwari, Jan Freyberg, Pradeep Shenoy, and Krishnamurthy Dvijotham. Interactive concept bottleneck models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5948–5955, 2023. doi: 10.1609/aaai.v37i5.25736. URL https://ojs.aaai.org/index.php/AAAI/article/view/25736.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.300. URL https://aclanthology.org/2021.emnlp-main.300/.

Mia Chiquier, Utkarsh Mall, and Carl Vondrick. Evolving interpretable visual classifiers with large language models. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 183–201, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-73039-9.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Antonio De Santis, Schrasing Tong, Marco Brambilla, and Lalana Kagal. Learning concept bottleneck models from mechanistic explanations. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/06e823ecae575f716620a75835635cce-Abstract-Conference.html.

DeepSeek-AI. Deepseek-v4-flash-vision-exp. https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash-Vision-Exp, 2026. Hugging Face model card, accessed 2026-09-24.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848. URL https://www.image-net.org/static_files/papers/imagenet_cvpr09.pdf.

Eric Enouen and Sainyam Galhotra. Debugging concept bottleneck models through removal and retraining. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/03d113a060c0ac93a5859517a0f07271-Abstract-Conference.html.

Eric Enouen and Sainyam Galhotra. Concept bottleneck diffusion for steerable generation. In *ICLR Workshop on Unifying Concept Representation Learning*, 2026b. URL https://openreview.net/forum?id=yP7ReFvRGN.

Mateo Espinosa Zarlenga, Katie Collins, Krishnamurthy Dvijotham, Adrian Weller, Zohreh Shams, and Mateja Jamnik. Learning to receive help: Intervention-aware concept embedding models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 37849–37875. Curran Associates, Inc., 2023. doi: 10.52202/075280-1648. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/770cabd044c4eacb6dc5924d9a686dce-Abstract-Conference.html.

Jean Feng, Avni Kothari, Patrick Vossler, Andrew Bishara, Lucas Zier, Newton Addo, Aaron Kornblith, Yan Shuo Tan, and Chandan Singh. Human-AI co-design for clinical prediction models. *npj Digital Medicine*, 2026. doi: 10.1038/s41746-026-02838-5. URL https://www.nature.com/articles/s41746-026-02838-5. Advance online publication.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23386–23397. Curran Associates, Inc., 2022. doi: 10.52202/068431-1699. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/944ecf65a46feb578a43abfd5cddd960-Abstract-Conference.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309. PMLR, 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Aya Abdelsalam Ismail, Julius Adebayo, Héctor Corrada Bravo, Stephen Ra, and Kyunghyun Cho. Concept bottleneck generative models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9149fc44c95ce58e3ca529a1e34c2691-Abstract-Conference.html.

Aya Abdelsalam Ismail, Tuomas Oikarinen, Amy Wang, Julius Adebayo, Samuel Stanton, Héctor Corrada Bravo, Kyunghyun Cho, and Nathan C. Frey. Concept bottleneck language models for protein design. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/8cf2d1a2fef23281af0f50d1a04dfd7c-Abstract-Conference.html.

Yiwen Jiang, Deval Mehta, Wei Feng, and Zongyuan Ge. Enhancing interpretable image classification through LLM agents and conditional concept bottleneck models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12285–12297. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.acl-long.600. URL https://aclanthology.org/2025.acl-long.600/.

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

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1gRTCVFvB.

Andrej Karpathy. autoresearch. GitHub repository, 2026. URL https://github.com/karpathy/autoresearch. Accessed September 23, 2026.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk99zCeAb.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Karras_Analyzing_and_Improving_the_Image_Quality_of_StyleGAN_CVPR_2020_paper.html.

Sangwon Kim, Kyoungoh Lee, Jeyoun Dong, and Kwang-Ju Kim. CoBELa: Steering transparent generation via concept bottlenecks on energy landscapes. *arXiv preprint arXiv:2507.08334*, 2026. URL https://arxiv.org/abs/2507.08334v3. Revised version; original version accepted at ICCV 2025 Workshops.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5338–5348. PMLR, 2020. URL https://proceedings.mlr.press/v119/koh20a.html.

Akshay Kulkarni, Ge Yan, Chung-En Sun, Tuomas Oikarinen, and Tsui-Wei Weng. Interpretable generative models through post-hoc concept bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8162–8171, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Kulkarni_Interpretable_Generative_Models_through_Post-hoc_Concept_Bottlenecks_CVPR_2025_paper.html.

Akshay Kulkarni, Tsui-Wei Weng, Vivek Narayanaswamy, Shusen Liu, Wesam A. Sakla, and Kowshik Thopalli. Interpretable and steerable concept bottleneck sparse autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/papers/Kulkarni_Interpretable_and_Steerable_Concept_Bottleneck_Sparse_Autoencoders_CVPR_2026_paper.pdf.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017. URL https://openaccess.thecvf.com/content_iccv_2017/html/Lin_Focal_Loss_for_ICCV_2017_paper.html.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *Computer Vision – ECCV 2024*, volume 15105 of *Lecture Notes in Computer Science*, pp. 38–55. Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-72970-6_3. URL https://link.springer.com/chapter/10.1007/978-3-031-72970-6_3.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019. URL https://arxiv.org/abs/1907.11692.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 651(8107):914–919, 2026. doi: 10.1038/s41586-026-10265-5. URL https://www.nature.com/articles/s41586-026-10265-5.

Utkarsh Mall, Cheng Perng Phoo, Mia Chiquier, Bharath Hariharan, Kavita Bala, and Carl Vondrick. DiSciPLE: Learning interpretable programs for scientific visual discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 29258–29267, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Mall_DiSciPLE_Learning_Interpretable_Programs_for_Scientific_Visual_Discovery_CVPR_2025_paper.html.

María Martínez-García, Ricardo Vazquez Alvarez, Alejandro Lancho, Pablo M. Olmos, and Isabel Valera. A probabilistic hard concept bottleneck for steerable generative models. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Kcb6WufAco.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131.

Tuomas Oikarinen, Subhro Das, Lam M. Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FlCg47MNvBA.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Qwen Team. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Sukrut Rao, Sweta Mahajan, Moritz Böhle, and Bernt Schiele. Discover-then-name: Task-agnostic concept bottlenecks via automated concept discovery. In *Computer Vision – ECCV 2024*, volume 15135 of *Lecture Notes in Computer Science*, pp. 444–461. Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-72980-5_26. URL https://link.springer.com/chapter/10.1007/978-3-031-72980-5_26.

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 *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/.

Sungbin Shin, Yohan Jo, Sungsoo Ahn, and Namhoon Lee. A closer look at the intervention procedure of concept bottleneck models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31504–31520. PMLR, 2023. URL https://proceedings.mlr.press/v202/shin23a.html.

Chandan Singh, Yan Shuo Tan, Weijia Xu, Zelalem Gero, Weiwei Yang, Michel Galley, and Jianfeng Gao. Agentic-imodels: Evolving agentic interpretability tools via autoresearch. *arXiv preprint arXiv:2605.03808*, 2026. URL https://arxiv.org/abs/2605.03808.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. Association for Computational Linguistics, 2013. URL https://aclanthology.org/D13-1170/.

Divyansh Srivastava, Ge Yan, and Tsui-Wei Weng. VLG-CBM: Training concept bottleneck models with vision-language guidance. In *Advances in Neural Information Processing Systems*, volume 37, pp. 79057–79094. Curran Associates, Inc., 2024. doi: 10.52202/079017-2510. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/90043ebd68500f9efe84fedf860a64f3-Abstract-Conference.html.

Chung-En Sun, Tuomas Oikarinen, Berk Ustun, and Tsui-Wei Weng. Concept bottleneck large language models. In *International Conference on Learning Representations*, pp. 89371–89411, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/de4ce91dfe56b919ee1c228d6a78f866-Abstract-Conference.html.

Moritz Vandenhirtz, Sonia Laguna, Ričards Marcinkevičs, and Julia E. Vogt. Stochastic concept bottleneck models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 51787–51810. Curran Associates, Inc., 2024. doi: 10.52202/079017-1641. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5c7894ac8788555f1cecf536f1e0fd35-Paper-Conference.pdf.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 dataset. Technical report, California Institute of Technology, 2011. URL https://authors.library.caltech.edu/records/cvm3y-5hh21.

Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Holden Karnofsky, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. *arXiv preprint arXiv:2411.15114*, 2024. URL https://arxiv.org/abs/2411.15114.

Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11205–11216. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/wong21b.html.

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. *arXiv preprint arXiv:2504.08066*, 2025. URL https://arxiv.org/abs/2504.08066.

Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19187–19197, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Yang_Language_in_a_Bottle_Language_Model_Guided_Concept_Bottlenecks_for_CVPR_2023_paper.html.

Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=nA5AZ8CEyow.

Z.ai. GLM-5.2 model card, 2026a. URL https://huggingface.co/zai-org/GLM-5.2. Accessed September 23, 2026.

Z.ai. GLM-5.3 model card, 2026b. URL https://huggingface.co/zai-org/GLM-5.3. Accessed September 23, 2026.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html.

Jie Zhu, Qian Chen, Huaixia Dou, Junhui Li, Lifan Guo, Feng Chen, and Chi Zhang. DianJin-R1: Evaluating and enhancing financial reasoning in large language models. *arXiv preprint arXiv:2504.15716*, 2025. doi: 10.48550/arXiv.2504.15716. URL https://arxiv.org/abs/2504.15716.

APPENDIX

The appendix provides the model details, experimental protocols, and additional results supporting the main paper. Appendix A describes the baseline architectures and training objectives; Appendix B explains baseline construction, candidate selection, and evaluation. Appendix C summarizes the agent-developed model recipes, while Appendix D presents additional sprint settings, benchmarks, recorded research costs, and representative sprint instructions. For reproducibility, we will also release our codebase and pretrained models from different sprints.

Appendix contents

A Model Formulations and Training Objectives . . . 16
B Baseline Setup and Evaluation . . . 17
B.1 Image Classification: CUB and ImageNet . . . 17
B.2 Text Generation: AGNews and SST2 . . . 18
B.3 Question Answering: FinQA . . . 18
B.4 Image Generation: CelebA-HQ and CUB . . . 19
C Agent-Developed Model Recipes . . . 20
D Additional Experiments . . . 22
D.1 Sprint Organization . . . 22
D.2 Additional Model and Dataset Comparisons . . . 23
D.2.1 Additional Classification Experiment: CUB CBM . . . 23
D.2.2 Additional CB-LLM Results: DBPedia and GSM8K . . . 23
D.2.3 Additional CB-AE Comparisons . . . 24
D.3 Agent Access and API Cost Estimates . . . 25
D.4 Recurring Ideas Across Sprints . . . 26
D.5 Representative Sprint Instructions . . . 27

A MODEL FORMULATIONS AND TRAINING OBJECTIVES

We expand the baseline descriptions in Section 3.1 with the mathematical formulations for VLG-CBM, CB-LLM, and CB-AE. For each model, we explain how concepts enter the prediction or generation process and how the training objectives supervise their learning and use. Appendix B gives the dataset-specific setup and evaluation procedures.

VLG-CBM baseline. Let f be the frozen image encoder, g the concept predictor, and h the sparse linear classifier. For an image x, the concept scores c and class prediction $\hat{y}$ are

$$c = g(f(x)), \qquad \hat{y} = h(c). \tag{1}$$

We first train g using binary cross-entropy (BCE) against concept pseudo-labels produced by GroundingDINO (Liu et al., 2024). We then freeze g and train h using class cross-entropy with elastic-net regularization to obtain sparse concept-to-class weights. ANEC-5 reports accuracy under a global sparsity budget averaging five nonzero concept weights per class.

Additional experiment: CUB CBM. This baseline uses 112 human-annotated attributes and jointly optimizes concept and class prediction during training. A separate sparse readout is fitted afterward for ANEC-5 and intervention evaluation. Its setup and results are given in Appendix D.2.1.

CB-LLM baselines. For a token prefix $x_{\leq t}$, let $z_t = f(x_{\leq t})$ denote the backbone features. The concept predictor g and unsupervised projection q feed a linear next-token output layer:

$$c_t = \mathrm{ReLU}(g(z_t)), \quad u_t = q(z_t), \quad p(x_{t+1} \mid x_{\leq t}) = \mathrm{softmax}(W[c_t; u_t] + b), \tag{2}$$

Here, c_t contains concept activations, u_t contains unsupervised features, and $[c_t; u_t]$ denotes concatenation. Concept supervision uses the pre-ReLU activations $g(z_t)$; token prediction uses c_t.

Across the AGNews, SST2, and FinQA baselines, we adapt the backbone through trainable LoRA adapters (Hu et al., 2022), keeping its original pretrained base weights frozen. The adapters, concept

bottleneck layer, unsupervised projection, and output layer are trained jointly using concept BCE and next-token cross-entropy, with elastic-net regularization on the final layer. An auxiliary classifier is trained to recover concept labels from the unsupervised features. With this classifier fixed, q is then optimized to make those predictions uninformative, discouraging u_t from encoding named-concept information. Additional training details are provided in Appendices B.2 and B.3.

CB-AE baseline. Write the frozen generator as $g_2 \circ g_1$, with intermediate features h. The encoder E produces concept activations c and unsupervised features u; the decoder D reconstructs h:

$$h = g_1(z), \quad (c, u) = E(h), \quad \hat{h} = D(c, u), \quad x = g_2(h), \quad \hat{x} = g_2(\hat{h}). \tag{3}$$

Here, z is sampled noise, x is the original generated image, and $\hat{x}$ is its reconstruction through CB-AE. Each concept has an activation for each possible value, while u supplies additional features for reconstruction.

Only E and D are trained, using generated images and concept pseudo-labels from a fixed predictor M. The baseline objective is

$$\mathcal{L}_{\text{AE}} = \lambda_h \mathcal{L}_h + \lambda_x \mathcal{L}_x + \lambda_c \mathcal{L}_c + \lambda_i \mathcal{L}_{\text{edit}}, \tag{4}$$

where $\mathcal{L}_h$ and $\mathcal{L}_x$ are mean-squared reconstruction errors for features and images, $\mathcal{L}_c$ aligns concept activations with pseudo-labels $M(x)$ using cross-entropy, and the λ terms are loss weights. For direct interventions, the activations of the current and desired concept values are swapped while u is unchanged. The edited representation is decoded into generator features and then an image. $\mathcal{L}_{\text{edit}}$ combines two cross-entropy losses that match the target concept labels: one uses M's predictions on the edited image; the other uses E's predictions after re-encoding the edited generator features.

B Baseline Setup and Evaluation

We describe the baselines from which the sprints begin, covering dataset splits, concept construction and labeling, training settings, and evaluation procedures. Each sprint uses one training seed; the candidate chosen for reporting is then retrained and evaluated with three seeds. We first explain this post hoc selection, followed by the setup for each task.

Choosing the best candidate within a sprint. A sprint can produce several promising candidates, with different Pareto candidates improving different metrics. To choose the best one for reporting, we apply task-specific weights to their improvements over the corresponding search baseline and select the highest-scoring eligible candidate. This weighting is applied after the sprint and is not provided to the agent; it is separate from the agent's keep/reject decisions.

For image classification, the weights on classification accuracy, concept AP, and intervention accuracy are 0.5, 0.25, and 0.25, respectively. For CB-LLM text generation, the weights on improvements in concept F1, steerability, and perplexity are 1, 1, and 0.005, respectively, with lower perplexity counted as an improvement. F1 and steerability use a 0–1 scale, while perplexity remains in its original units. FinQA instead weights improvements in answer accuracy, concept F1, and final intervention accuracy by 0.5, 0.25, and 0.25. FinQA selection includes rejected candidates; the updated AGNews GLM loss-constrained recipe instead uses equally weighted relative improvements and requires all three metrics to improve.

B.1 Image Classification: CUB and ImageNet

Data and concepts. Following VLG-CBM (Srivastava et al., 2024), we use 5,990 training and 5,790 test images for CUB. On ImageNet, we use a subset of 400,000 images for training (400 per class) and 50,000 validation images for evaluations. The concept sets contain 671 and 4,729 concepts for CUB and ImageNet, respectively, following VLG-CBM (Srivastava et al., 2024).

Baseline training. We freeze the CUB-pretrained ResNet-18 and ImageNet-pretrained ResNet-50 v1 backbones (He et al., 2016). On CUB, we train with binary cross-entropy, Adam (Kingma & Ba, 2015), learning rate 5×10^{-4}, batch size 32, and up to 75 epochs with validation-based early stopping, retaining the best checkpoint. Concept-region cropping is applied with probability 0.5. ImageNet uses weighted binary cross-entropy (positive weight 100), AdamW (Loshchilov & Hutter, 2019), learning rate 10^{-3} with cosine decay, batch size 256, and five epochs.

Evaluation. We fit a sparse linear head on the concept bottleneck using GLM-SAGA (Wong et al., 2021), targeting an average of five nonzero weights per class, rather than exactly five for every class. This corresponds to approximately 99.25% zero weights on CUB and 99.89% on ImageNet. On CUB, we retain the 1,000 largest-magnitude weights and use the resulting head for both ANEC-5 and intervention accuracy. On ImageNet, ANEC-5 uses the fitted regularization-path head; intervention evaluation retains its 5,000 largest-magnitude weights without refitting or changing the bias.

For each evaluation image, we uniformly sample up to three concepts without replacement from those labelled present by GroundingDINO and used by the sparse classifier. We replace their activations with their respective training-set maxima, leaving other activations and the class label unchanged. We apply this procedure to all images, including initially correct predictions, and average accuracy over five random selections per image. These selections are separate from the three training-seed evaluations. Concept AP is computed from continuous concept activations against GroundingDINO labels and averaged across concepts; it does not require an activation threshold.

B.2 Text Generation: AGNews and SST2

Data and concepts. AGNews (Zhang et al., 2015) contains 120,000 training samples; for the sprints, we use a fixed subset of 10,000 samples, with 2,500 from each topic class, and all 7,600 test samples. For SST2, we use 6,920 samples for training, 872 samples for validation, and 1,821 for concept evaluation during the sprints. We reuse the concept banks developed for classification in CB-LLM (Sun et al., 2025): 216 news concepts for AGNews and 208 movie-review concepts for SST2, such as company earnings and emotional depth. We use Qwen3-32B (Qwen Team, 2025) to label the presence or absence of each concept in the training and evaluation samples. We filter out rare concepts and concepts with a low agreement with a second label source (MPNet-derived labels for AGNews and Llama labels for SST2). We remove near-duplicate concept names using cosine similarity between character-level TF-IDF representations (3–5 character sequences), with a threshold of 0.78. We retain 80 AGNews and 56 SST2 concepts. Concept selection and labeling are completed before the sprints, and the concept sets remain fixed throughout.

Baseline training. We initialize Llama-3-8B with the released CB-LLM LoRA adapter (Sun et al., 2025) and fine-tune the adapters together with the concept-bottleneck components and output layer. The bottleneck includes a 768-dimensional unsupervised branch. We use the objectives in Appendix A and add a concept-frequency calibration loss. Adam uses learning rate 5×10^{-5}, increased sixfold for the concept bottleneck layer, and input length 128. AGNews trains for one epoch at batch size 24. SST2 trains for two epochs at batch size 8, retaining the best validation checkpoint.

Evaluation. For concept evaluation, we threshold the pre-ReLU concept activations at zero at the final input position and compute macro F1 against the Qwen-generated labels. For the final results in Table 2, each training seed is evaluated using 500 ordinary and 500 steered texts generated from a beginning-of-sequence token, with at most 100 new tokens. The historical AGNews evaluations used in Sections 4.4 and 4.5 instead use 100 texts per condition. While steering, we set the target concept activation to 80 throughout generation. We separately fine-tune a RoBERTa-base multilabel classifier (Liu et al., 2019) on the Qwen-generated concept labels before the sprints and keep it fixed. The classifier reads each generated text and predicts a probability for each concept. Steerability is the fraction of steered texts for which the target concept's probability is at least 0.40. We use an unadapted Llama-3-8B to score each generated text and report the arithmetic mean of per-text perplexities. Final table evaluations use temperature 0.5, top-k 50, top-p 0.9, repetition penalty 1.3, and normal end-of-sequence stopping. The SST2 black-box PPL of 84.70 in Table 2 is a published reference (Sun et al., 2025), not a rerun under these common settings.

Implementation provenance. For repeated evaluation, the AGNews architecture-constrained implementation was reconstructed from recorded code reads, edits, and surviving source files; it is not a fully recovered original source tree.

B.3 Question Answering: FinQA

Data and concepts. FinQA (Chen et al., 2021) is a numerical-reasoning question-answering benchmark derived from financial reports. Each example contains a natural-language question, report text and tables, annotated supporting evidence, and a reference executable program whose result deter-

mines the answer. The dataset contains 6,251 training, 883 development, and 1,147 test examples. All sprints use fixed subsets of 2,000 training and 300 development examples, while results are reported on the full test set. Following researcher-specified design requirements and candidate categories, a Codex agent constructed a fixed vocabulary of 45 concepts by analyzing FinQA's schema, reference programs, supporting-evidence annotations, and dataset-wide concept prevalence. The concepts represent arithmetic operations, program structure, evidence composition, units and scales, temporal relations, and financial quantities. Deterministic program parsing and rule-based matching over the reference program, question, and annotated evidence assign a 45-dimensional binary concept vector to every example. For instance, division is present when the reference program contains a division operation, while table-and-text evidence is present when the supporting-evidence annotations identify both sources. The vocabulary and labeling rules remain fixed across all sprints.

Baseline training. We adapt Qwen2.5-7B-Instruct with rank-8 LoRA (Hu et al., 2022), a 45-concept bottleneck, and a 768-dimensional unsupervised branch. Inputs contain the question, retrieved text, and table; targets contain a program and answer. The baseline trains for one epoch with batch size 1 and sequence length 512. Adam uses learning rate 5×10^{-5} for the backbone adapters and bottleneck, and 10^{-3} for the adversarial concept probe. The training objective combines target-token loss, concept binary cross-entropy with weight 1.5, and elastic-net regularization. Separate adversarial updates alternate between training the probe to predict concepts and training the unsupervised branch to push the probe's predictions toward 0.5. No additional calibration loss is used.

Evaluation. Greedy decoding stops at end-of-sequence or 80 tokens. Answer accuracy scores the executed program's result. For each checkpoint, a single shared concept-activation threshold is selected from $\{0.05, 0.10, \ldots, 1.00\}$ to maximize macro-F1 on all 883 development samples. This threshold applies across all 45 concepts to the nonnegative activations at the final prompt position. For incorrectly answered questions, we correct up to three concept predictions using reference labels and regenerate the answer. We identify incorrect concept predictions using an activation threshold of 0.05 and prioritize those furthest from their correction values. Present concepts are set to their maximum development-set activations and absent concepts to their minima, with these values fixed throughout generation. Intervention accuracy is the percentage of questions answered correctly after correction, retaining initially correct answers.

B.4 Image Generation: CelebA-HQ and CUB

Data and concepts. Following CB-AE (Kulkarni et al., 2025), we use CelebA-HQ with 8 balanced concepts and CUB dataset with 10 balanced concepts. The unsupervised concept embedding in CB-AE is 40-dimensional as in the original paper. Similar to CB-AE, no real images from the datasets are used for training. Instead, a pretrained StyleGAN2 (Karras et al., 2020) model is used to generate images during training. Concept supervision for training is obtained from supervised ResNet-18 classifiers as the pseudo-label predictor M following CB-AE.

Baseline training. We follow the original paper (Kulkarni et al., 2025) for baseline training. A 4-layer MLP is used as the CB-AE with a frozen pretrained StyleGAN2 as the base generative model. This baseline CB-AE is trained for 50 epochs with the same hyperparameters as in the original paper's released codebase.

Evaluation. Unlike the original paper (Kulkarni et al., 2025), we exclude concept accuracy evaluation since the baseline already achieves quite high accuracy ($\sim$86%) and does not vary much with agentic changes as well. On the other hand, intervention success rate or steerability is relatively lower (44.55%) with more room for improvement. Following Kulkarni et al. (2025), interventions involve swapping the concept activations for a target concept, generating the intervened image, and using the pretrained concept classifier to evaluate if the generated image contains the target concept. We also evaluate the FID (Fréchet Inception Distance) for CB-AE-reconstructed images as well as CB-AE-intervened images to evaluate the perceptual quality of images apart from the steerability.

C AGENT-DEVELOPED MODEL RECIPES

Table 5: Representative agent-developed recipes and their measured results. Appendix B describes candidate selection within each sprint. Base and Ours are the reporting-baseline and agent-recipe means; standard deviations appear in the main results tables. Accuracy, AP, F1, and steerability are percentages; lower PPL and FID are better. AP and F1 denote concept metrics; Int. acc. denotes intervention accuracy. These are complete recipes, not component ablations.

VLG-CBM (CUB)	**Results**		
Architecture: Kept the frozen ResNet-18 backbone and mixed its original and layer-normalized features in a 5:3 ratio. Added a width-32 correction branch with LayerNorm and ReLU, a learned sigmoid gate for each concept, and learned scales and biases for the concept scores.		Base	Ours
	ANEC-5	75.67	75.85
	AP	48.77	53.47
	Int. acc.	85.20	91.10
Loss: Kept concept BCE and added classifier supervision under five concept corrections, with weight 3.0 (0.75 during fine-tuning).			
Other: Trained with Adam at LR 10^{-3} for up to 75 epochs, patience 30; then fine-tuned the classifier for ten epochs.			
VLG-CBM (ImageNet)	**Results**		
Architecture: Kept the frozen ResNet-50 backbone and added two nonlinear branches of widths 2,048 and 512 alongside the linear concept predictor, each with LayerNorm and ReLU. Summed their outputs and scaled the combined pre-sigmoid concept scores by 0.35.		Base	Ours
	ANEC-5	73.54	75.06
	AP	49.83	57.59
	Int. acc.	92.21	92.74
Loss: Used focal concept loss with $\gamma = 1$, $\alpha = 0.03125$, and positive weight 100.			
Other: Trained for five AdamW epochs with cosine decay. LRs: linear head 5×10^{-4}; branches 2.5×10^{-4}/5×10^{-4}. Retained the sparse classifier.			
CB-LLM (AGNews)	**Results**		
Architecture: Retained the CB-LLM architecture. This is candidate 97 from the GLM 5.3 loss-constrained sprint.		Base	Ours
Loss: Refined concept predictions with BCE, calibration, ranking, and sparsity losses. Added output-layer supervision with a target concept activation set to 80, using both training-text prefixes and model-generated contexts.	F1	58.69	73.79
	Steer.	26.73	71.00
	PPL	96.23	43.17
	Steered PPL	258.36	67.36
Other: After initial training, trained only the concept head for 5,120 steps with concept-balanced sampling, then only the output layer for 20,480 steps. Both stages used cosine learning-rate decay. Refreshed generated training contexts during the output stage.			

Table 5 continued: agent-developed model recipes and results.

CB-LLM (SST2)	**Results**		
Architecture: Added a width-56 GELU correction to concept scores, then applied $5\tanh(s/5)$ to the corrected scores s, followed by ReLU. Added a rank-32 path that rescales concept activations and a rank-one path from concepts to token scores (training dropout 0.1). **Loss:** Kept token loss, concept BCE (weight 1.5; log-capped positive weights, cap 5), and rate calibration (weight 10). **Other:** During generation, bounded the rescaling adjustment with $\tanh$ and doubled the added token scores. Common-policy evaluation removes the steering-only delay in end-of-sequence stopping.		Base	Ours
	F1	58.63	60.59
	Steer.	24.80	36.47
	PPL	47.85	63.84

CB-LLM (FinQA)	**Results**		
Architecture: Retained Qwen2.5-7B-Instruct CB-LLM. Refit the linear concept bottleneck layer, rescaled its outputs, and mixed 75% fitted weights and biases with 25% original parameters. **Loss:** Used class-balanced logistic regression ($C = 1$, at most 3,000 iterations) for concept bottleneck refitting and answer-token loss for decoder adaptation. **Other:** Froze features and the concept bottleneck layer; adapted the vocabulary projection and unsupervised branch for 400 Adam steps (LR 10^{-4}). Averaged parameters over the last 150 steps of initial training and adaptation.		Base	Ours
	Ans. acc.	65.16	68.35
	F1	39.76	78.62
	Int. acc.	65.42	68.73

CB-AE (CelebA-HQ)	**Results**		
Architecture: Replaced BatchNorm with LayerNorm, tuned CB-AE weight initialization: lower initial weight for more frequent concepts in encoder final layer, added residual connections from encoder to decoder. **Loss:** Added loss weights for $\mathcal{L}_x$ and $\mathcal{L}_{\text{edit}}$ with linear warmup schedule for the weights. **Other:** Changed optimizers to have 200-step linear warmup from 0 to 2e-4 followed by linear decay to 1e-5.		Base	Ours
	Steer.	44.55	73.47
	Recon. FID	13.76	9.00
	Int. FID	12.83	12.38

CB-AE (CUB)	**Results**		
Architecture: Unchanged CB-AE. **Loss:** Class-balanced concept loss, KL divergence loss for encoder $\rightarrow$ decoder $\rightarrow$ encoder cyclic consistency, cosine similarity loss for consistency in non-intervened concepts. **Other:** Tuned loss weights.		Base	Ours
	Steer.	10.84	17.55
	Recon. FID	75.97	72.60
	Int. FID	76.73	73.62

D Additional Experiments

D.1 Sprint Organization

A sprint requires agents to conduct experiments, review their outcomes, and carry useful findings forward. Figure 4 shows four ways to organize this work. We first describe the continuous-agent and supervisor settings used in the main paper (Section 3.3), then introduce subagent and episodic settings that delegate work or start each candidate with a fresh conversation. Table 6 summarizes the available results and CUB costs; the additional runs provide an initial exploration of these settings.

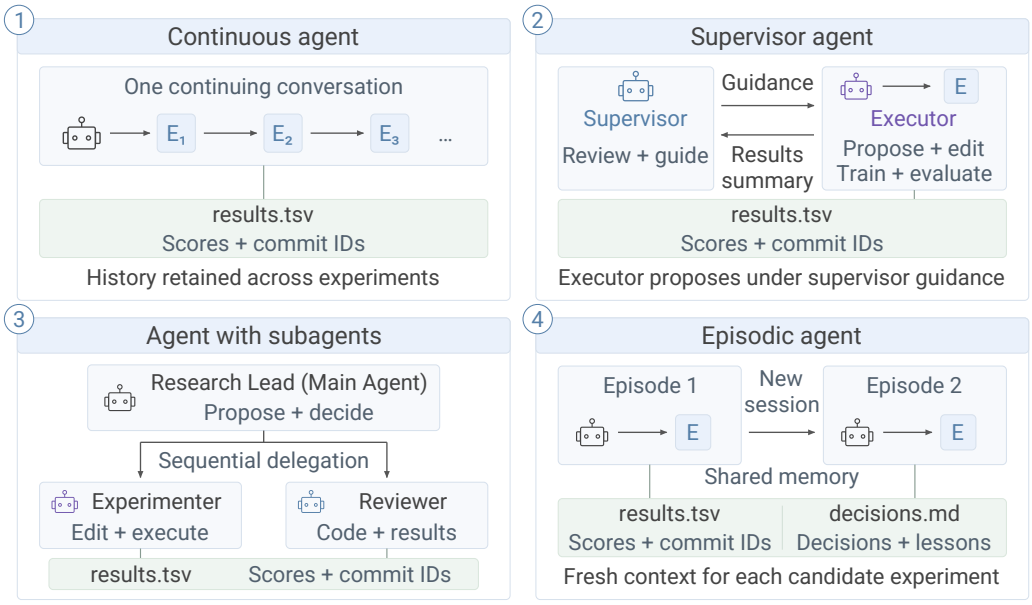


Figure 4: **Sprint organization.** A continuing agent session and a supervisor/executor pair are compared with sequential subagent delegation and fresh sessions for successive candidates. Green boxes show the files that carry results and decisions across experiments; E denotes a candidate experiment.

Continuous agent. A single agent proposes, implements, evaluates, and reviews candidates within a continuing conversation, as shown in Fig. 4 (panel 1). Self-directed, Loss-constrained, and Architecture-constrained all use this setting, differing only in the permitted proposal scope.

Supervisor agent. The executor develops candidate proposals, edits code, and runs experiments, while a supervisor reviews progress and guides the research direction, as shown in Fig. 4 (panel 2). Separate conversation contexts let the supervisor focus on general direction while the executor handles specific proposal and implementation details. A Python controller, *monitor.py*, resumes the executor for one candidate at a time and checks the recorded outcome. At scheduled review points or when an issue is detected, it resumes the supervisor and passes its guidance to the executor's next turn. The executor uses this feedback to develop its next proposal. The controller coordinates these handoffs and keeps candidate experiments sequential.

Subagents. Here, the main agent retains responsibility for proposals and final keep/reject decisions. It delegates implementation and execution to an experimenter agent and code and result inspection to a reviewer agent, each with a separate conversation context, as shown in Fig. 4 (panel 3). This gives review a dedicated role and reduces the main agent's implementation workload. In an AGNews sprint, the Reviewer identified repetitive generations despite the candidate passing the steered-PPL check. This issue was cited in the subsequent rejection decision, revealing a problem that the numerical check alone did not capture.

Episodic agent. A fresh agent session handles each candidate, as shown in Fig. 4 (panel 4). The code persists, `results.tsv` records scores, `decisions.md` accumulates decisions and lessons, and a separate `handoff.md` summarizes the previous episode. The runner includes this handoff

in the next session's context. This makes the transfer of findings explicit and limits the effect of context compaction done by the agent (which can be out of our control).

Results. Table 6 summarizes the single-seed results for subagent and episodic sprints with Claude Sonnet 5.

Table 6: Sprint settings with Claude Sonnet 5. † marks single-seed results. Bold indicates improvements over the baseline, blue indicates the best result for each metric.

	VLG-CBM, CUB				CB-LLM, AGNews			CB-AE, CelebA-HQ		
Setting	ANEC-5 (↑)	Concept AP (↑)	Int. Acc. (↑)	Cost/cand. ($)	Concept F1 (↑)	Steer. (↑)	PPL (↓)	Steer. (↑)	Recon. FID (↓)	Int. FID (↓)
Baseline	75.67	48.77	85.20	—	58.69	26.73	96.23	44.55	13.76	12.83
Self-directed	**75.80**	**52.84**	**86.34**	2.21	58.16	**31.40**	96.22	**69.88**	**8.60**	13.67
Supervisor	**75.79**	**52.79**	**86.76**	0.70	**66.03**	**59.67**	**40.88**	**73.47**	**9.00**	**12.38**
Subagents†	**75.96**	**53.82**	**91.23**	2.88	**65.35**	**36.20**	**26.53**	**53.63**	19.40	19.06
Episodic†	**77.05**	**54.14**	**88.37**	1.13	**59.94**	**35.60**	**21.85**	**54.08**	19.93	22.03

D.2 Additional Model and Dataset Comparisons

D.2.1 Additional classification experiment: CUB CBM

We adapt the original CBM formulation (Koh et al., 2020) using a frozen CUB-pretrained ResNet-18 and 112 human-annotated bird attributes. Each baseline and candidate uses the same training budget on a single NVIDIA A40 GPU. Evaluation follows the VLG-CBM CUB procedure in Appendix B.1, reporting ANEC-5, concept AP, and intervention accuracy over three training seeds on the sprint evaluation set.

Table 7 reports self-directed, loss-constrained, architecture-constrained, and subagent runs with Claude Sonnet 5. The subagent run uses a main agent assisted by an experimenter and a reviewer.

Table 7: CUB results for original CBM (Koh et al., 2020) over three training seeds. Bold indicates improvements over the baseline, blue indicates the best result for each metric, gray-shaded row indicates improvement on all three metrics.

Setting	ANEC-5↑	Concept AP↑	Int. Acc.↑
Baseline (w/o Agent)	67.90 ± 0.01	89.10 ± 0.00	71.04 ± 0.07
Sonnet 5 (Ours)			
Self-directed	**73.01** ± 0.18	**91.07** ± 0.03	**75.43** ± 0.27
Loss	**71.62** ± 0.11	**89.82** ± 0.01	**74.45** ± 0.06
Arch.	**72.75** ± 0.32	**93.12** ± 0.02	**75.03** ± 0.26
Subagents	**71.78** ± 0.02	**91.52** ± 0.01	**74.16** ± 0.04

D.2.2 Additional CB-LLM Results: DBpedia and GSM8K

We extend our CB-LLM experiments to topic-controlled generation on DBpedia and mathematical question answering on GSM8K.

DBpedia. DBpedia (Zhang et al., 2015) uses its 14 topic labels as concepts, following CB-LLM (Sun et al., 2025). We initialize Llama-3-8B with the released DBpedia adapter and concept head, and train on 10,000 samples for one epoch with batch size 4. Each training sample is tokenized and truncated to at most 350 tokens. Concept evaluation uses 20,000 samples. The baseline generates up to 100 new tokens and steers by setting the target concept activation to 150 and the others to zero. We use the DBpedia RoBERTa topic-classifier checkpoint released by the

CB-LLM authors (Sun et al., 2025), without retraining it. Steerability is the percentage of 98 generated texts whose predicted topic matches the requested topic. Int. PPL measures perplexity after intervention. Results retain each candidate's generation and intervention settings. Selection uses the text-generation weights, requiring ordinary PPL no higher than baseline and intervention PPL at most 60 above baseline; scored candidates remain eligible regardless of keep/reject decisions.

GSM8K. GSM8K (Cobbe et al., 2021) tests mathematical question answering. Our CB-LLM baseline uses Llama-3-8B-Instruct, with 5,000 training and 500 development examples. Its 20 arithmetic and reasoning concepts are labelled by a Qwen model from reference solutions; uncertain labels are excluded from concept evaluation.

We report answer exact match, concept macro-F1, generated-response perplexity, and intervention accuracy after correcting up to two concept mismatches using reference labels. Corrections apply during the first 32 generated tokens, and evaluation includes all 500 development examples. Intervention accuracy (Int. Acc.) is the percentage of answers that are correct after these corrections. Post hoc selection retains the existing weighted score based on answer accuracy, concept F1, intervention gain, and PPL, subject to an absolute steered-PPL ceiling of 30. These are development results, not held-out test results.

Table 8 reports single-seed results; dashes denote unreported entries. All scores except PPL are percentages.

Table 8: Additional CB-LLM results on DBPedia and GSM8K (single seed). Bold indicates improvement over baseline; blue marks the best result within each agent and dataset. Gray-shaded row indicates improvement in all metrics.

	DBPedia				GSM8K			
Proposal	Concept F1↑	Steer↑	PPL↓	Int. PPL↓	Acc.↑	Concept F1↑	Int. Acc.↑	PPL↓
Baseline (w/o Agent)	99.15	79.59	29.89	69.33	69.80	68.81	69.60	7.52
GLM 5.3 (Ours)								
Self-directed	99.15	**94.90**	29.89	77.60	—	—	—	—
Supervisor	**99.18**	71.43	**26.96**	87.93	—	—	—	—
Claude Sonnet 5 (Ours)								
Self-directed	99.15	**80.61**	**22.80**	**63.75**	**76.80**	65.05	**77.00**	**5.26**
Loss	99.15	**80.61**	**24.07**	75.17	**74.00**	**72.21**	**74.00**	7.61
Arch.	**99.25**	**86.73**	**25.88**	76.01	**71.00**	**72.48**	**71.20**	7.54
Supervisor	**99.23**	**87.76**	**22.46**	101.68	—	—	—	—

Results. On DBpedia, baseline concept F1 is already 99.15%, so the main gains concern generation and steering. Claude Sonnet 5's supervisor-directed candidate lowers PPL from 29.89 to 22.46 and raises steerability from 79.59% to 87.76%, although intervention PPL increases from 69.33 to 101.68. GLM's self-directed candidate reaches 94.90% steerability by changing which competing concept activations are suppressed, with unchanged ordinary F1 and PPL.

On GSM8K, Claude Sonnet 5's self-directed sprint raises answer accuracy from 69.80% to 76.80% and lowers PPL from 7.52 to 5.26, but reduces concept F1 from 68.81% to 65.05%. The loss- and architecture-constrained sprints improve both answer accuracy and concept F1. Intervention gains remain small: 0.20 percentage points corresponds to one net additional correct answer among 500 examples.

D.2.3 Additional CB-AE comparisons

Comparisons with additional agentic LLMs. Beyond GLM 5.3 (Z.ai, 2026b) and Claude Sonnet 5 (Anthropic, 2026c) experiments in the main paper, we also experiment with DeepSeek-v4 (DeepSeek-AI, 2026) and Claude Opus 5 (Anthropic, 2026b) in Table 9. We find that DeepSeek-v4 can provide improvements over GLM 5.3, despite having fewer total parameters (284B vs 744B) as well as fewer active parameters (13B vs 40B), both being mixture-of-experts models where only a part of the total parameters are active per token inference. Further, Opus 5 outperforms most other

Table 9: Extended evaluations for image generation on CelebA-HQ. We also compare to recent human-designed SOTA method VHCB (Martínez-García et al., 2026). Bold indicates improvement over the baseline, and blue indicates each agent's best result. Gray-shaded row indicates improvement over baseline in all 3 metrics.

		CelebA-HQ		
Agent	Proposal	Steer. (%, ↑)	Recon. FID (↓)	Interv. FID (↓)
VHCB (w/o Agent)		55.59 ± 0.2	23.81 ± 0.2	18.23 ± 0.3
Baseline (w/o Agent)		44.55 ± 0.6	13.76 ± 0.3	12.83 ± 0.2
DeepSeek-v4 (Ours)	Self-directed	**62.10** ± 0.6	16.76 ± 0.8	14.83 ± 0.4
	Arch.	**52.00** ± 2.3	16.24 ± 1.2	16.64 ± 1.2
GLM 5.3 (Ours)	Self-directed	**47.66** ± 1.6	47.25 ± 1.1	46.57 ± 1.8
	Loss	**50.68** ± 2.7	37.78 ± 10.8	40.37 ± 9.2
	Arch.	**49.45** ± 1.2	36.85 ± 12.9	37.87 ± 9.3
	Supervisor	**55.70** ± 1.5	19.11 ± 4.4	19.76 ± 4.1
Sonnet 5 (Ours)	Self-directed	**69.88** ± 1.1	**8.60** ± 0.2	13.67 ± 0.2
	Loss	**58.06** ± 2.7	16.35 ± 1.2	18.23 ± 1.0
	Arch.	**50.07** ± 3.2	45.63 ± 9.0	45.57 ± 5.5
	Supervisor	**73.47** ± 3.0	**9.00** ± 0.1	**12.38** ± 0.6
Opus 5 (Ours)	Self-directed	**81.97** ± 0.8	**9.67** ± 0.4	**10.44** ± 0.6

agents with just the self-directed sprint, being the stronger and more expensive model. Due to usage limit constraints, we could not experiment more with the stronger Opus 5 model.

Comparisons to SOTA methods. In Table 9, we compare our CBM-Scientist approach to a recent human-designed state-of-the-art method VHCB (Martínez-García et al., 2026). We find that GLM 5.3 tends to underperform compared to VHCB, while DeepSeek-v4 and closed-source LLMs can improve steerability up to 26.38% and FID by ∼8-14 points compared to VHCB.

D.3 Agent Access and API Cost Estimates

We ran our Claude experiments through the Claude Max subscription ($200 per month). For GLM 5.3, we used an external provider serving `Inferact/GLM-5.3-NVFP4` on four NVIDIA H200 NVL GPUs.

To estimate costs from usage, we apply Anthropic's published API rates (Anthropic, 2026a) to the recorded input, output, and cache-token counts. Table 10 reports the estimated cost per completed, scored research candidate for the four main Claude Sonnet 5 proposal settings across the model families and datasets shown. These API-equivalent estimates are different from the actual subscription fees paid.

Table 10: Estimated agent cost per candidate (USD) for Claude Sonnet 5. Costs apply API rates to recorded token usage and are averaged over completed, scored candidates.

	VLG-CBM		CB-LLM			CB-AE
Proposal setting	**CUB**	**ImageNet**	**AGNews**	**SST2**	**FinQA**	**CelebA-HQ**
Self-directed	2.21	1.21	2.00	1.67	1.52	1.88
Loss-constrained	1.57	1.18	1.27	1.64	1.82	1.13
Architecture-constrained	1.43	1.87	1.16	1.57	2.88	2.58
Supervisor-directed	0.70	0.57	0.89	1.68	1.86	0.91

D.4 Recurring Ideas Across Sprints

Tables 11–13 give the prevalence counts underlying Section 4.3. Each dataset has eight sprints: two agents and four proposal strategies. Counts describe recorded trials, including unsuccessful experiments and ablations, rather than successful models or independent discoveries. The categories retain their model-specific definitions; recurrence alone does not establish a performance benefit.

Table 11: Recurring ideas in image-classification sprints (VLG-CBM). Entries count sprints containing a recorded trial of the idea, out of eight per dataset.

Idea	CUB	ImageNet
Focal loss	5/8	5/8
Concept reweighting	6/8	6/8
Concept ranking	6/8	5/8
Decorr. / orthogonality	5/8	3/8
Residual predictors	6/8	5/8
Intervention training	6/8	1/8

Table 12: Recurring ideas in language-model sprints (CB-LLM). Entries count sprints containing a recorded trial of the idea, out of eight per dataset.

Idea	AGNews	SST2	FinQA
Loss / gradient balance	8/8	7/8	8/8
Concept reweighting	7/8	7/8	6/8
Concept calibration	8/8	7/8	3/8
Edited-concept training	7/8	8/8	5/8
Bypass control	4/8	8/8	7/8
Concept-to-token path	7/8	6/8	8/8

Table 13: Recurring ideas in image-generation sprints (CB-AE). Entries count sprints containing a recorded trial of the idea, out of eight per dataset.

Idea	CelebA-HQ	CUB
Concept-target calibration	5/8	6/8
Loss rebalancing	5/8	5/8
Intervention loss redesign	5/8	5/8
Reconstr./perceptual/latent losses	5/8	5/8
BatchNorm/LayerNorm freezing	4/8	6/8
EMA of weights	3/8	5/8
Optimizer hyperparam. tuning	5/8	2/8
Residual/gated pathways	3/8	3/8
Width/depth/bottleneck tuning	3/8	3/8

D.5 Representative Sprint Instructions

We show the complete `program.md` for a self-directed ImageNet VLG-CBM sprint, with minor wording edits for clarity; the original file is retained with the experiment records. Some runtime and resource-management instructions are specific to the hardware and computing environment used for this sprint.

Self-directed: ImageNet (complete instruction file).

````
# ImageNet VLG-CBM Autoresearch Program

You are an autonomous research agent working on one ImageNet VLG-CBM
    sprint. Make one focused, falsifiable change at a time, measure it with
    the fixed evaluation protocol, and retain only research changes that
    improve the three-metric frontier.

## Repository Roles

- `CBM/` is the shared CUB/ImageNet research implementation. Candidate
    edits belong only in its model and training surfaces, while preserving
    CUB.
- `prepare.py`, `harness/`, `run_candidate.py`, `supervisor/`, the
    single-agent drivers, and this program are fixed infrastructure. Do not
    edit them during a sprint.
- `experiments/imagenet/program.md` is the canonical ImageNet scientific
    contract. A single-agent ImageNet worktree materializes it as root
    `program.md` before launch.

Run each candidate with arguments, never by editing a wrapper:

```bash
python run_candidate.py \
  --dataset imagenet \
  --name "short_candidate_name" \
  --hypothesis "One falsifiable hypothesis" \
  --dataset-root "$IMAGENET_TRAIN_ROOT" \
  --val-root "$IMAGENET_VAL_ROOT" \
  --annotation-dir "$IMAGENET_ANNOTATION_DIR" \
  --activation-dir "$ACTIVATION_DIR" \
  --precomputed-target-dir "$IMAGENET_PRECOMPUTED_TARGET_DIR" \
  --devkit-dir "$IMAGENET_DEVKIT_DIR" \
  > run.log 2>&1
```

Run `run_candidate.py` synchronously in the foreground. Never use shell
    backgrounding (`&`, `nohup`, or `disown`), an agent background task, a
    watcher, or a scheduled wakeup. Stay with the command until it returns,
    inspect its score, and finalize the candidate before beginning another.

The first candidate in a new ImageNet sprint must be the untouched
    baseline: use `--name baseline` with no `--set` arguments. The harness
    rejects research changes until that baseline has a scored result.

## Fixed Evaluation Protocol

Keep these fixed:

- dataset: ImageNet-1k
- backbone: torchvision ResNet-50 with ImageNet v1 weights
- concept head format: `global_only`, operating on fixed 2048-dimensional
    pooled features
- concept set: `CBM/concept_files/imagenet_filtered.txt` (4,729 concepts)
- training data: the fixed balanced 400k manifest and precomputed global
    GroundingDINO targets
- CBL training: exactly 5 epochs
````

```
- validation data: the complete fixed 50k-image validation split
- NEC level: `5`
- frontier accuracy: `acc_nec_5`
- frontier concept metric: `concept_prediction_macro_ap`
- frontier intervention metric: `tti_acc_at_3`

ImageNet TTI uses `W_g@NEC=5.pt`. For each validation image it randomly
    selects up to three concepts marked present in the reference
    annotations that carry nonzero NEC=5 weight and overwrites them with
    their train-set maximum activation. It only asserts presence. A TTI
    gain must not come with an accuracy or concept-AP regression beyond the
    tolerances below.

The canonical baseline under this protocol is `acc_nec_5=0.7351799607`,
    `concept_prediction_macro_ap=0.4983148292`, and
    `tti_acc_at_3=0.922668`. Use it when the current `results.tsv` has no
    baseline row. Use search tolerances of `0.003` accuracy and `0.005`
    concept AP; these are heuristic thresholds, not estimates of
    statistical uncertainty.

Do not modify the evaluator, dataset membership, target cache, manifest,
    annotation mapping, concept set, backbone, global-only head interface,
    NEC definition, or removal map. In single-agent mode, research changes
    belong only in the model/training implementation under `CBM/`. In
    supervised mode, the monitor's current phase may narrow that surface
    further.

## Fixed Runtime Accelerations

`IMAGENET_BACKBONE_CACHE_DIR` contains immutable pooled ResNet-50 training
    and validation features. The harness injects it automatically and
    maintains a derived, validated global-target matrix beside them. Do not
    edit, relocate, or override these caches. The fixed NEC solver uses
    `lam_max=0.001`, up to 150 GLM path steps, and 500 iterations, stopping
    as soon as NEC=5 is reached; these are evaluation settings, not
    research knobs.

Source the private runtime file before a manual status or candidate
    command:

```bash
source .agent-runtime.env
```

Autonomous Research Direction

In single-agent mode you own the scientific direction. Choose focused,
    falsifiable candidates from evidence in prior scores, primary sources,
    and the implementation.

Research Loop

1. Read recent `runs/*/score.json` files and identify the current frontier
    for `acc_nec_5`, `concept_prediction_macro_ap`, and `tti_acc_at_3` only.
2. State one falsifiable hypothesis grounded in a primary source or local
    empirical result, then make the smallest relevant change.
3. Commit the candidate research change before GPU execution.
4. Run one candidate synchronously through `run_candidate.py`.
5. Inspect bounded output: score lines and a short log tail only.
6. Finalize with `completed_kept` or `completed_rejected` and a factual
    reason.
7. Retain a research branch only when it provides useful evidence; never
    call TTI alone a frontier win when accuracy or concept AP violates its
    guardrail.
```

```
A pre-candidate smoke test for a newly introduced module is limited to
    correctness on synthetic data or at most one minibatch. Before any full
    training or metric evaluation, register the candidate and use
    `run_candidate.py`; never use an untracked multi-arm or performance
    screen to choose a candidate.

If the allocation ends, rerun the exact same candidate command. Durable
    stage state skips completed work. Do not change code or configuration
    around an active candidate. Start every turn with `python
    run_candidate.py status`; when it reports an active candidate, resume
    only that exact candidate. Never launch timing, abbreviated,
    smoke-training, or diagnostic training while it remains active.

Single-agent mode: Continue this loop indefinitely without waiting for
    human input; finalize one candidate and immediately begin the next. If
    a finite client turn exits, return normally; the shell driver resumes
    the same session. Every resumed turn must start or resume a real
    candidate; never reply that no candidate is warranted or that the
    search is exhausted. If a local direction is exhausted, pivot to a
    materially different mechanism. Marginal gains or a local optimum do
    not end the sprint. In supervised mode, follow the monitor's
    one-candidate turn boundary.
```