Discovering Structurally Plausible and Interpretable Cognitive Models with Large Language Models

Boyoung Kwon[1], **Jiyoon Seo**[2], **Sang Ho Lee**[1,2,3 †]
[1]Graduate School of Data Science, KAIST
[2]School of Digital Humanities and Computational Social Sciences, KAIST
[3]Department of Brain & Cognitive Sciences, KAIST
{bykwon, cristina0106, sangholee}@kaist.ac.kr

Abstract

Computational cognitive modeling has traditionally relied on manual model construction, which is a time-intensive process that requires substantial domain expertise. Although recent LLM-based approaches have begun to automate model generation, they often prioritize predictive accuracy at the expense of structural plausibility and parameter identifiability. We introduce DisCo, an LLM-guided framework that discovers interpretable symbolic cognitive models through evolutionary search with two structural components: a Validator that filters structurally invalid models, and a Cleaner that promotes parameter identifiability by removing redundant parameters. Across three behavioral tasks, DisCo consistently discovers models that achieve performance competitive with task-specific baselines. Ablation analyses show that the Validator is important for recovering psychologically meaningful structure. Without this component, models tend to violate behavioral constraints and show less consistent associations with external cognitive measures. These results suggest that LLM-driven model discovery, when coupled with appropriate structural constraints, can yield cognitively plausible models that go beyond predictive fit.

1 Introduction

Computational cognitive models have long served as a central tool for explaining human behavior in terms of latent cognitive mechanisms. By formalizing psychological processes as mathematical equations, these models allow researchers to quantify individual differences in cognition and predict behavior under novel conditions. However, developing such models requires substantial domain expertise, coding proficiency, and extensive trial-and-error refinement (Agrawal et al., 2020; Musslick et al., 2025). This manual process is not only time-consuming but also vulnerable to researchers' theoretical preconceptions, which may constrain exploration to a narrow subset of possible model structures.

These limitations have motivated a growing interest in automated approaches to cognitive model discovery. In particular, large language models (LLMs) have emerged as powerful tools for automating scientific discovery, with demonstrated applications ranging from equation discovery in physics to symbolic regression in biology (Romera-Paredes et al., 2024; Shojaee et al., 2025). Building on these advances, researchers have begun applying LLMs specifically to cognitive model generation. Recent frameworks such as GeCCo (Rmus et al., 2025) and CogFunSearch (Castro et al., 2025) have demonstrated that LLMs can generate cognitive models with strong predictive performance, primarily evaluated by their fit to behavioral data.

†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

However, predictive accuracy alone is insufficient for the scientific goals of cognitive modeling. A model may fit behavioral data well while violating the causal structure of the task, for example by using future outcomes that would not be available at the time of decision. Such a model may achieve strong quantitative performance while misrepresenting the underlying cognitive processes it is intended to capture. Moreover, if model parameters are to be interpreted as markers of cognitive traits, such as learning rate or response sensitivity, they should be identifiable from behavioral data (Wilson and Collins, 2019). Goodness of fit is an unreliable indicator of parameter identifiability, as multiple parameter configurations can produce indistinguishable outputs.

We introduce **DisCo** (Discovery of Interpretable Symbolic Cognitive Models), an LLM-guided framework that incorporates two structural components into the discovery loop. First, a Validator filters models that violate causal assumptions of the task, such as referencing future outcomes unavailable at decision time. Second, a Cleaner promotes parameter identifiability by detecting and removing redundant parameters based on posterior covariance structure. Together, these components promote models that are not only predictive but also structurally plausible and interpretable.

We evaluate DisCo across three well-established behavioral tasks representing distinct cognitive functions spanning prospective, adaptive, and retrospective processes: planning (two-step task; Daw et al. 2011), learning (probabilistic reversal learning task; Zorowitz et al. 2023), and memory (recognition memory task; Cataldo et al. 2023). Our results show that explicit validation is critical for recovering psychologically meaningful model structure.

2 Related Work

2.1 Automated Scientific Discovery

Automated scientific discovery has traditionally relied on symbolic regression methods, particularly genetic programming (GP) (Koza, 1992; Schmidt and Lipson, 2009). GP evolves candidate mathematical expressions through evolutionary operations, but faces challenges in scalability and incorporating prior knowledge (Petersen et al., 2019; Makke and Chawla, 2024).

More recently, the emergence of large language models (LLMs) has opened new avenues for automated scientific discovery. FunSearch (Romera-Paredes et al., 2024) combines LLMs with evolutionary search to support mathematical discoveries, producing interpretable programs rather than black-box solutions. LLM-SR (Shojaee et al., 2025) demonstrates that LLMs' scientific priors can guide equation discovery with superior generalization, while EUREKA (Ma et al., 2023) achieves human-level performance in reward function design through LLM-based evolutionary optimization. These developments position LLMs as powerful tools for automated, interpretable scientific discovery.

2.2 Deep Learning in Cognitive Modeling

In cognitive modeling, deep learning approaches have achieved strong predictive performance in complex behavioral tasks (Dezfouli et al., 2019). Deep neural networks can serve as a flexible framework for developing cognitive models, but their black-box nature limits their utility for theory-driven cognitive modeling. Several methods address this limitation, including disentangled representations via information bottlenecks (Miller et al., 2023), simplified architectures with minimal units (Ji-An et al., 2025), and hybrid neural–cognitive models (Eckstein et al., 2023). However, such approaches often rely on predefined architectural constraints or model structures, limiting their ability to automatically discover novel cognitive mechanisms.

2.3 Automated Discovery of Cognitive Models

Recent frameworks leverage LLMs to generate cognitive models that explain behavioral data. CogFunSearch (Castro et al., 2025) adapts the FunSearch (Romera-Paredes et al., 2024) framework by utilizing an LLM as a mutation operator to evolve symbolic programs that maximize the likelihood of observed behavior. Similarly, GeCCo (Rmus et al., 2025) employs in-context learning to translate task descriptions directly into computational models. These approaches primarily evaluate discovered models via goodness-of-fit and qualitative theoretical alignment, with less attention paid to the plausibility of the model structure or to quantitative indices of theoretical alignment. We advance this paradigm by incorporating additional evaluation steps typically performed by human cognitive

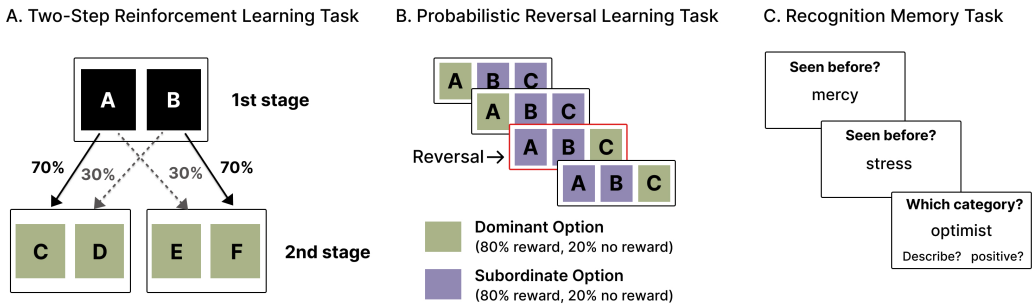

Figure 1: Overview of the behavioral tasks. We utilize large-scale data from (A) the two-step reinforcement learning task, (B) the probabilistic reversal learning task, and (C) the recognition memory task.

modelers to improve the interpretability of the model: assessing model structure for parameter identifiability and causal consistency, and correlating parameter estimates with external indices of cognitive traits.

3 Method

3.1 Dataset

To evaluate our framework, we employed three behavioral datasets that meet the following criteria: (i) sufficiently large sample sizes, and (ii) availability of well-established baseline computational models. Each dataset primarily engages a distinct cognitive domain, namely planning, learning, and memory, allowing us to assess whether the discovered model generalizes across qualitatively different cognitive functions. Although no single task isolates one process entirely, each paradigm places relatively greater demand on one domain. An overview of the task paradigms is presented in Figure 1. Additional details on each dataset, including the baseline models and external measures, are provided in Appendix A.

3.1.1 Planning

Task. The two-step reinforcement learning task is a widely used paradigm for studying planning. We used the dataset from Gillan et al. (2016), which is based on the paradigm originally introduced by Daw et al. (2011). The task was designed to distinguish between habitual (model-free) and goal-directed (model-based) decision-making processes. On each trial, participants chose between two abstract stimuli at the first stage, which led probabilistically to one of two second-stage states. At the second stage, participants selected between two additional stimuli, after which a monetary outcome was delivered probabilistically. We used the trial-by-trial data from Experiment 1 ($N = 548$ participants, 200 trials each).

Baseline Model. For the two-step reinforcement learning task, we used simplified versions of two models from Daw et al. (2011) as baselines: a model-free temporal-difference (TD) learning model that updates action values solely from experienced rewards, and a hybrid model that additionally incorporates model-based planning via the task's transition structure.

3.1.2 Learning

Task. We used the dataset from Zorowitz et al. (2023), which is based on a probabilistic reversal learning task designed to probe reinforcement-learning processes. On each trial, participants chose among three options and received probabilistic feedback. The dominant option yielded a reward with 80% probability, while the subordinate options each yielded 20%. The dominant option changed every 15 trials, requiring flexible adaptation. We used the trial-by-trial data from $N = 386$ participants (90 trials each).

Baseline Model. A temporal-difference (TD) learning model (Mihatsch and Neuneier, 2002) served as the baseline. The model updates expected values for each option using asymmetric learning rates for positive and negative prediction errors, with action selection governed by a softmax rule.

3.1.3 Memory

Task. The dataset from Cataldo et al. (2023) is based on an emotional memory task designed to investigate valence-dependent memory biases. During the encoding phase, participants judged normatively positive and negative words in either a self-reference ("Does this word describe you?") or valence ("Is this word positive?") task. During the test phase, participants viewed an equal mix of studied (old) and unstudied (new) words and judged whether each had been seen before. We used trial-by-trial data from $N = 1{,}358$ participants (approximately 100 recognition trials each).

Baseline Model. We used a trial-level equal-variance signal detection theory (SDT) model (Macmillan and Creelman, 2005) as the baseline. The model treats each trial's binary recognition response as determined by the observer's ability to discriminate old from new items and their response criterion.

3.2 Problem Formulation

We frame cognitive model discovery as the joint optimization of model structure and parameters.

Behavioral Data. Consider a dataset $\mathcal{D} = \{d_1, \ldots, d_N\}$ consisting of behavioral trajectories from N subjects. For each subject i, the data $d_i = \{(s_{i,t}, c_{i,t}, r_{i,t})\}_{t=1}^{T_i}$ consists of a sequence of (possibly implicit) states $s_{i,t} \in \mathcal{S}$, choices $c_{i,t} \in \mathcal{A}$, and rewards $r_{i,t} \in \mathbb{R}$ over T_i trials.

Symbolic Cognitive Model. We seek to discover a symbolic cognitive model ϕ from a space of candidate models Φ. Crucially, each model ϕ defines not only the executable function f_ϕ but also the valid parameter space $\Theta_\phi \subset \mathbb{R}^k$ (e.g., bound constraints generated by the LLM). Given subject-specific parameters $\theta_i \in \Theta_\phi$, the model maps the interaction history $h_{i,t}$ to a probability distribution over the next action:

$$P(c_{i,t} \mid h_{i,t}; \phi, \theta_i) = f_\phi(h_{i,t}; \theta_i) \tag{1}$$

Optimization Objective. Since the datasets considered in this work involve sequential learning tasks, we employ a chronological split validation strategy to preserve temporal order and to capture trial-by-trial learning dynamics. For each participant i, we partition the data d_i into the first half d_i^{train} for parameter estimation and the subsequent half d_i^{val} for model evaluation. We use the Bayesian Information Criterion (BIC) as the model-selection objective. BIC balances goodness of fit against model complexity by imposing a penalty for the number of free parameters; lower BIC scores indicate a better trade-off between predictive fit and parsimony (Schwarz, 1978). We seek the program ϕ^* that minimizes the average BIC evaluated on the validation segment:

$$\phi^* = \arg\min_{\phi \in \Phi} \frac{1}{N} \sum_{i=1}^{N} \text{BIC}(\phi, \hat{\theta}_i; d_i^{\text{val}}) \tag{2}$$

where $\hat{\theta}_i$ are the parameters estimated from the training segment d_i^{train} via constrained maximum likelihood estimation (MLE) subject to $\theta_i \in \Theta_\phi$. Note that while parameters are fixed after the training phase, the model simulation proceeds continuously to preserve latent state dynamics (e.g., Q-values) accumulated during training when evaluating on d_i^{val}. This within-participant chronological split was used during model discovery on the training participants. For final evaluation, the selected program ϕ^* was applied to held-out test participants using the same procedure (see Appendix B for details of the data-splitting procedure and alternative strategies supported by DisCo).

Parameter Identifiability and Significance. For a discovered model to be interpretable, its parameters must be identifiable and influential. We define two criteria for valid parameters: (i) Multicollinearity: Parameters must not exhibit excessive multicollinearity (e.g., $|\rho| < \tau_{\text{corr}}$), which ensures that each parameter represents a distinct cognitive trait. (ii) Individual Variability: Parameters must demonstrate meaningful variance across subjects ($\sigma^2 > \tau_{\text{var}}$), preventing the inclusion of constants that fail to capture behavioral heterogeneity. Accordingly, we adopt these criteria for evaluating discovered models, such that only scientifically plausible models are considered valid candidates.

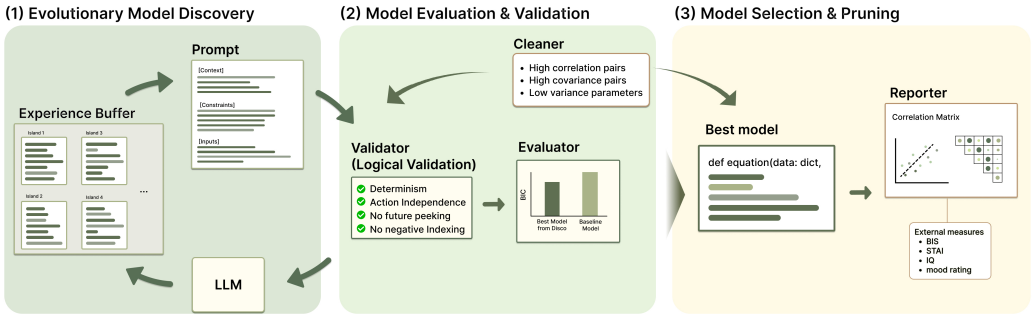

Figure 2: Overview of the DisCo pipeline.

Association with Cognitive Traits. In addition to predictive accuracy and parameter validity, we consider whether fitted parameters exhibit external cognitive relevance. When independent psychological measures $\mathcal{M}$ are available, we compute correlations between fitted parameter estimates and each measure in $\mathcal{M}$. To account for multiple comparisons across parameter–measure pairs, we apply the Benjamini–Hochberg false discovery rate (FDR) correction. We regard an association as externally meaningful when it survives Benjamini–Hochberg FDR correction at $p_{\mathrm{FDR}} < 0.05$. We report the corresponding correlation coefficients as effect-size estimates.

3.3 The DisCo Framework

Building upon the framework of LLM-based evolutionary algorithms (Shojaee et al., 2025; Romera-Paredes et al., 2024), we propose DisCo (Discovery of Interpretable Symbolic Cognitive Models). As illustrated in Figure 2, this approach leverages the code-generation capabilities of LLMs within an evolutionary strategy to explore the space of possible cognitive equations. DisCo specifically tailors this methodology to automate the scientific discovery process, seeking a model ϕ^* that optimizes predictive accuracy and scientific interpretability.

The optimization process is formulated as an island-based evolutionary algorithm where the LLM functions as a mutation and crossover operator. The pipeline operates through iterative evolutionary cycles and a final refinement stage: (1) *Evolutionary Hypothesis Generation* generates candidate programs; (2) *Model Validation* filters invalid models using logical integrity checks and chronological validation; and (3) *Model Selection & Pruning* periodically removes redundant parameters to encourage parsimony (during island updates) and, where applicable, to assess the resulting model based on external cognitive measures. The full discovery procedure is summarized in Appendix C.

3.3.1 Evolutionary Model Discovery

Sampler (LLM-based Mutation). To bootstrap the discovery process, we initialize all island buffers with an initial exemplar provided in the template. Starting from these seeds, we leverage the code-generation capabilities of LLMs to evolve model structures. The Sampler constructs a prompt (see Appendix D) containing the task specification (function signature and state variables) and a set of high-scoring exemplars sampled from the current island's experience buffer. The LLM outputs executable Python code defining the forward pass and parameter bounds.

Experience Buffer. We maintain diversity using an island model approach (Romera-Paredes et al., 2024). Each island buffer B_k stores a population of programs sorted by their BIC scores. Periodically, the pipeline performs a migration step where top-performing models (i.e., those with the lowest BIC) are exchanged between islands to prevent premature convergence to local optima.

3.3.2 Model Evaluation and Validation

Validator (Logical Validation). Before fitting, generated programs undergo strict causal validation to filter out scientifically invalid models. We encourage four integrity constraints to help ensure that generated models respect the causal structure of sequential decision-making:

(i) Determinism: Given identical inputs, the model must produce identical outputs. Formally, for any history $h_{i,t}$ and parameters θ_i, repeated evaluations must satisfy $P(c_{i,t} \mid h_{i,t}; \phi, \theta_i) = P'(c_{i,t} \mid h_{i,t}; \phi, \theta_i)$. This prevents the use of stochastic operations (e.g., `np.random`) within the model.

(ii) Independence of Action: The probability $P(c_{i,t} \mid h_{i,t}; \phi, \theta_i)$ must be calculated without accessing the actual choice $c_{i,t}$. This helps ensure the model predicts choice probabilities before observing the action, preventing trivial solutions that simply copy the observed choice.

(iii) No Future Peeking: Predictions at time t must not change when future data ($t + k$, where $k > 0$) is perturbed. We verify this through randomized causal fuzzing: for randomly selected timesteps t, we corrupt future observations and confirm that $P(c_{i,t} \mid h_{i,t})$ remains invariant.

(iv) No Negative Indexing: Array access with negative indices (e.g., `data[-1]`) is disallowed to prevent cyclic information leakage, where the first prediction could depend on the final observation.

While these constraints specifically address causal validity in sequential tasks, the proposed validation mechanism is customizable, allowing users to specify alternative integrity checks appropriate for their specific domain directly within the code template.

Evaluator (Optimization & Scoring). The Evaluator module quantifies the quality of candidate models that pass the logical validation. It solves the optimization problem defined in Equation 2. Consistent with the chronological split strategy, we estimate the parameters θ on d_i^{train} using the L-BFGS-B algorithm, with parameter estimates being bounded by LLM-generated constraints. The goodness-of-fit score is then computed as the average of BIC scores on the validation set d_i^{val}.

3.3.3 Model Selection and Pruning

Cleaner (Parameter Pruning). To promote parsimony, the Cleaner module periodically inspects the population for over-parameterization. We identify problematic parameters based on two criteria: (i) Multicollinearity ($|\rho| < \tau_{\text{corr}}$) and (ii) Individual variability ($\sigma^2 < \tau_{\text{var}}$). When detected, the LLM is prompted (Appendix D.4) to simplify the equation by removing or consolidating the redundant parameters. If the simplified model ϕ' maintains a comparable BIC score to the original ϕ, it replaces the complex version in the buffer.

Reporter (External Validation). The Reporter module applies the external association criterion defined above to the top candidate models from the evolutionary search. Specifically, it correlates the fitted parameter estimates $\hat{\theta}$ of each candidate model with independent external measures $\mathcal{M}$ and summarizes which parameter–measure associations satisfy the FDR-corrected significance criterion. These results are used to characterize and compare candidate models in terms of external cognitive relevance, complementing the primary selection based on validation BIC.

4 Experiments

4.1 Experimental Setup

We evaluated the performance of DisCo-generated models against established baseline computational models across three tasks: planning, learning, and memory. For the model discovery process, we used Qwen2.5-32B (Yang et al., 2024). All LLM inference was performed using a vLLM server (Kwon et al., 2023) running on two NVIDIA RTX A6000 GPUs (48GB VRAM each).

We used $T = 0.6$ during evolutionary generation to encourage candidate diversity and $T = 0.0$ during the Cleaner phase for deterministic pruning. Participants in each task were split into training and held-out test sets at an 8:2 ratio. For each run, DisCo generated and evaluated 200 candidate models, selected the final model on the training set, and computed BIC on the held-out test set. We repeated all experiments with five random seeds to assess the stability of the discovery process. Additional configuration information is provided in Appendix E.

4.2 Performance Across Cognitive Domains

To evaluate DisCo, we applied the full pipeline to the three behavioral datasets. For each dataset, we compared BIC scores between DisCo-generated models and task-specific baselines: a standard

Table 1: Comparison of BIC scores across tasks, evaluated on held-out test participants. For DisCo, results are reported across five independent runs (mean $\pm$ std and range). For the baseline models, BIC was computed once and is reported as a single score.

Task	Model	BIC Mean $\pm$ Std	BIC range
Planning	Model-Free	257.89	–
	Hybrid	255.22	–
	DisCo (Null)	240.54 ± 8.07	[229.38, 249.36]
	DisCo (MF)	221.16 ± 1.75	[218.27, 223.57]
	DisCo (Hybrid)	221.32 ± 2.83	[216.20, 224.94]
Learning	TD Learning	76.06	–
	DisCo (Null)	70.64 ± 0.15	[70.57, 70.94]
	DisCo (TD Learning)	74.49 ± 2.20	[70.27, 76.71]
Memory	SDT	61.81	–
	DisCo (Null)	51.95 ± 0.04	[51.93, 52.02]
	DisCo (SDT)	56.03 ± 1.93	[53.46, 59.28]

model-free/hybrid RL model for the planning task, TD learning for the learning task, and SDT for the memory task. In addition to seeding the search with these task-specific baseline models, we also included a null-seeded condition, where DisCo was initialized from a minimal model with no task-specific computational assumptions. All DisCo runs used identical hyperparameters across tasks, with prompts reported in Appendix D.

Table 1 summarizes BIC scores across five random seeds. Lower scores indicate better predictive fit. For DisCo variants, the labels in parentheses indicate the baseline model used to initialize the search; for example, DisCo (MF) denotes initialization from the model-free baseline, whereas DisCo (Null) denotes initialization from a minimal null model without task-specific assumptions. DisCo consistently discovers models that match or outperform task-specific baselines across all three tasks, with low variance across runs indicating stable convergence. DisCo (Null) also achieves competitive performance on the memory and learning tasks, demonstrating that the LLM-guided search can identify compact model structures without domain-specific initialization. However, baseline seeding provided a clear advantage on the planning task, where DisCo (MF) and DisCo (Hybrid) substantially outperformed DisCo (Null), suggesting that established cognitive mechanisms can serve as useful structural priors when available.

Beyond predictive performance, the discovered models may provide alternative accounts of the cognitive processes represented by the baseline models or extend those accounts with additional mechanisms. The learning task provides a clear example. The TD baseline model explains individual differences in reversal learning through separate learning rates for positive and negative prediction errors. In contrast, the discovered model uses a single learning rate and decay of unchosen-option values. It attributes individual differences to forgetting of unchosen options rather than differential sensitivity to positive and negative outcomes. This account predicts that, after a reversal, participants should be slower to choose the newly advantageous option the longer it has remained unchosen. The best discovered models for all three tasks and their differences from the corresponding baselines are described in detail in Appendix F.

4.3 Comparison with Existing LLM-based Model Discovery

To compare DisCo with existing LLM-based approaches to model discovery, we conducted a direct comparison with GeCCo (Rmus et al., 2025) across all three tasks. For each task, both methods were initialized with the same task-specific baseline model and evaluated under the same computational budget. Five independent runs were performed for each method, with 200 candidate models generated per run. The final model selected from each run was evaluated on held-out data. For this comparison, we define a final model as valid if it satisfies our post-hoc validity checks, which include checks of the structural constraints performed by the Validator as well as basic mathematical and code-level checks required for valid likelihood computation. Table 2 summarizes the held-out BIC scores across valid runs.

Table 2: Held-out BIC comparison between DisCo and GeCCo across tasks. Statistics are computed over runs in which the final selected model satisfied the validation criteria.

Task	Method	Valid runs	BIC Mean $\pm$ Std	BIC Range
Planning	DisCo	5/5	221.32 ± 2.83	[216.20, 224.94]
	GeCCo	3/5	231.65 ± 16.58	[218.95, 255.06]
Learning	DisCo	5/5	74.49 ± 2.20	[70.27, 76.71]
	GeCCo	3/5	80.57 ± 7.18	[73.96, 90.55]
Memory	DisCo	5/5	56.03 ± 1.93	[53.46, 59.28]
	GeCCo	3/5	837.91 ± 25.77	[802.73, 863.73]

For all three tasks, DisCo produced a valid final model in each of the five runs, whereas GeCCo did so in only three of the five runs. Among the final models selected by GeCCo, some violated the Validator's constraints by making their predictions depend on observed behavioral responses, whereas others failed additional post-hoc validity checks because of incorrect likelihood computation. In GeCCo, the LLM can modify the code used to compute the likelihood, allowing errors in the likelihood computation (e.g., choice probabilities that do not sum to one) to persist in the final selected model. In DisCo, the LLM only generates choice probabilities, while the likelihood is computed separately by a fixed evaluation function. After excluding invalid runs, DisCo showed a lower mean BIC score than GeCCo for each task, with substantially lower variability across runs.

The difference between the two methods was particularly pronounced in the memory task. The final DisCo models achieved held-out BIC scores of 53.46–59.28, whereas the valid final GeCCo models yielded scores of 802.73–863.73. This difference may be partly explained by how candidate models are evaluated during the search. DisCo splits each participant's data into training and validation sets for model evaluation during the search (see Appendix B), whereas GeCCo uses the full data without such a split. The final comparison applied the same splitting procedure used during DisCo's search to a separate held-out dataset. DisCo's training–validation split may help reduce overfitting, while also aligning its model-evaluation procedure more closely with the final held-out evaluation. Both factors may contribute to its lower BIC scores. However, the use of different data-splitting strategies does not affect the results of the validity checks, which show a key advantage of DisCo. Additional details, including BIC scores under an alternative evaluation protocol, validation-rule violations in GeCCo-generated models, and qualitative comparisons between the models discovered by DisCo and GeCCo, are provided in Appendix G.

4.4 Validator and Cleaner Ablation Study

To understand the role of each structural component in the DisCo pipeline, we conducted a 2×2 ablation study varying the presence of the Validator and Cleaner modules. Experiments were run on the planning task dataset, with 200 iterations per seed. To assess the utility of the modules, we quantified (i) the proportion of models violating the Validator's constraints across conditions and (ii) the BIC scores of the discovered models.

The results in Table 3 show that the Validator and Cleaner modules play complementary but distinct roles in the DisCo pipeline. The Cleaner primarily contributes to search stability: when present, the variance of BIC remains consistently low across seeds (MF: 1.75, Hybrid: 2.83), whereas removing it leads to higher variance and occasional pathological collapse (MF without Cleaner or Validator: 158.35 ± 65.54). This suggests that the Cleaner prevents the search from exploiting degenerate solutions that achieve low BIC through overfitting rather than genuine model improvement. The Hybrid condition, being more structurally constrained, shows lower violation rates than the MF condition and yields relatively stable results even without the Validator. However, the elevated variance in the Cleaner-only condition (195.09 ± 33.26) indicates that such stability is not guaranteed, supporting the role of the Validator as an additional safeguard.

The Validator module, by contrast, enforces behavioral constraints that BIC alone cannot capture. Removing the Validator results in non-negligible rates of constraint violations (non-deterministic violations: up to 17.8%, independence violations: up to 2.5%). In most conditions without the Validator, BIC scores were relatively low but highly variable, suggesting that constraint violations

Table 3: Ablation study on the planning task across Validator and Cleaner module combinations. Results are reported across five independent runs, each with 200 iterations. Violation rates are computed only among accepted candidate models.

Baseline	Cleaner	Validator	Non-det.	Indep.	Future	Neg-idx	BIC Mean $\pm$ Std
MF	✓	✓	0%	0%	0%	0%	221.16 ± 1.75
	✓		14.3%	2.3%	0%	0.1%	199.41 ± 38.05
		✓	0%	0%	0%	0%	218.07 ± 3.22
			17.8%	2.5%	0%	0%	158.35 ± 65.54
Hybrid	✓	✓	0%	0%	0%	0%	221.32 ± 2.83
	✓		6.8%	0.9%	0%	0.1%	195.09 ± 33.26
		✓	0%	0%	0%	0%	222.34 ± 4.13
			6.7%	2.1%	0%	0%	224.52 ± 3.51

Non-det.: non-deterministic violation; Indep.: independence violation; Future: future peeking; Neg-idx: negative indexing.

may produce artificially favorable BIC scores. Representative examples of these violations at the code level are provided in Appendix H.

4.5 External Validation with Cognitive Measures

Cognitive models often include parameters that relate to underlying psychological or cognitive processes. We therefore examined whether model-derived parameters showed consistent associations with external measures of psychological traits. For the planning and learning datasets, we correlated fitted model parameters with available participant-level cognitive or behavioral measures and applied FDR correction for multiple comparisons. Because the two modules affected external associations differently across tasks, we report in the main text the task-specific contrasts in which each module showed the clearest effect, while providing the full results of FDR-significant associations in Appendix I.

Table 4 shows representative runs that illustrate how the Validator and Cleaner affect parameter interpretability. In the planning task (A), models discovered with the Validator consistently identify a model-based weight (w) that negatively correlates with age and positively correlates with IQ, consistent with prior findings (Eppinger et al., 2013; Otto et al., 2015). They also show a stable correlation between inverse temperature (β) and IQ, consistent with prior evidence linking IQ to decision noise (Ger et al., 2024). Although some associations vary across seeds, the central associations (w with age and IQ, and β with IQ) remain stable across runs (see Appendix I). Without the Validator, some runs still show similar associations, while others show lower BIC yet exhibit additional or shifted associations involving parameters such as γ and α. In the learning task (B), retaining the Cleaner yields some models with significant external associations involving β, whereas removing it produces substantially lower BIC but mostly non-significant or less clearly interpretable associations. These examples suggest that the Validator and Cleaner contribute to parameter interpretability in task-dependent ways.

4.6 Prompt Ablation Study

Motivated by prior LLM-based cognitive model discovery work that analyzed the contribution of prompt components through ablation (Rmus et al., 2025), we examined whether DisCo's prompt-level instructions help guide the search toward high-performing models. Specifically, we systematically removed individual elements from the full DisCo prompt and measured their effects on the BIC scores of the discovered models. The Validator and Cleaner modules were retained across all conditions to isolate the effect of prompt-level guidance while preserving the core structural constraints. For this ablation study, we used the model-free (MF) formulation as the baseline model class, as its simpler parameterization provides a cleaner signal for comparing the effect of individual prompt components.

Results shown in Table 5 suggest that prompt-level optimization has limited effects on model quality, with BIC differences falling within run-to-run variability in most configurations. These results indicate that, in this setting, search outcomes are not strongly determined by any single prompt-level instruction, suggesting that prompt refinement alone may be insufficient for substantially improving the efficiency of model discovery.

Table 4: External validity of model parameters under ablated conditions, shown for representative runs and associations that illustrate key contrasts.

(A) Validator effect
Planning task (Hybrid baseline), Cleaner ON

Condition	BIC	Significant correlations
w/ Validator	224.94	$w \leftrightarrow$ age $-.19$, IQ $+.19$
		$\beta \leftrightarrow$ IQ $+.17$
	221.83	$w \leftrightarrow$ age $-.22$, IQ $+.12$
		$\beta \leftrightarrow$ IQ $+.15$
	216.20	$w \leftrightarrow$ age $-.23$, IQ $+.16$
		$\beta \leftrightarrow$ IQ $+.14$
w/o Validator	221.83	$w \leftrightarrow$ age $-.22$, IQ $+.12$
		$\beta \leftrightarrow$ IQ $+.15$
	139.72	$w \leftrightarrow$ age $-.15$, IQ $+.12$
		$\gamma \leftrightarrow$ age $+.14$
	173.32	$\alpha \leftrightarrow$ IQ $+.20$, $\beta \leftrightarrow$ IQ $-.13$
		$\gamma \leftrightarrow$ age $-.20$

(B) Cleaner effect
Learning task, Validator OFF

Condition	BIC	Significant correlations
w/ Cleaner	76.06	$\beta \leftrightarrow$ BIS $-.16$
	76.06	$\beta \leftrightarrow$ BIS $-.16$
	75.37	n.s.
w/o Cleaner	23.21	n.s.
	29.79	n.s.
	56.48	$\gamma \leftrightarrow$ BIS $-.20$

All reported correlations are FDR-corrected $p < 0.05$.

Table 5: Ablation study on prompt components. Each row removes one component from the full prompt. Lower BIC is better. Results are reported across five independent runs.

Prompt Configuration	BIC Mean $\pm$ Std	BIC range
Full prompt (all components)	221.16 ± 1.75	[218.27, 223.57]
w/o Context	220.29 ± 3.93	[216.24, 227.46]
w/o Constraints	220.23 ± 1.90	[218.25, 223.36]
w/o Parsimony	224.61 ± 3.50	[218.25, 228.17]
w/o Validation Method	219.77 ± 2.03	[218.25, 223.36]

5 Conclusion

We introduced DisCo, an LLM-guided framework for automatically discovering interpretable cognitive models from behavioral data. By combining LLM-based evolutionary search with structural constraints, DisCo aims to reduce the manual effort and theoretical bias inherent in traditional model construction. Across three cognitive tasks, DisCo discovered models with competitive or superior BIC scores compared to task-specific baselines, suggesting that LLM-guided search can offer a viable alternative to manual model construction.

However, the ablation study suggests that predictive performance alone is inadequate for evaluating model quality. Without the Validator, discovered models tend to violate constraints and show inconsistent associations with external cognitive measures, while occasionally achieving lower BIC scores. This dissociation between predictive fit and psychological validity suggests that effective LLM-guided model discovery requires integrating LLM-based search with domain-grounded validation.

Despite DisCo's reliable performance, several limitations should be considered. First, its performance is sensitive to the provided baseline model, suggesting that prior assumptions derived from domain knowledge can shape the search process in both beneficial and detrimental ways. Second, although DisCo automates model discovery, the Validator relies in part on task-specific constraints. Establishing standardized evaluation protocols that extend beyond the validation criteria considered here, therefore remains an important open problem for cognitive model discovery. Looking forward, we hope DisCo contributes to a broader conversation about how cognitive modeling can be automated without compromising psychological validity and interpretability.

Acknowledgments and Disclosure of Funding

This work was supported by the New Faculty Settlement Research Program through Korea Advanced Institute of Science & Technology (Grant No. G04240059 [to SHL]); the Data Science Convergence

Human Resource Development Program through the National Research Foundation (NRF) of Korea funded by the Ministry of Science and ICT (Grant No. RS-2022-NR068758 [to SHL]); the AI Science Hub Program through the Ministry of Science and ICT (Grant No. N10260136 [to SHL]); and the Industry Research and Development Program funded by FMMENTALCARE (Grant No. G01250572 [to SHL]).

References

Agrawal, M., Peterson, J. C., and Griffiths, T. L. (2020). Scaling up psychology via scientific regret minimization. *Proceedings of the National Academy of Sciences*, 117(16):8825–8835.

Castro, P. S., Tomasev, N., Anand, A., Sharma, N., Mohanta, R., Dev, A., Perlin, K., Jain, S., Levin, K., Elteto, N., et al. (2025). Discovering symbolic cognitive models from human and animal behavior. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 6849–6890.

Cataldo, A. M., Scheuer, L., Maksimovskiy, A. L., Germine, L. T., and Dillon, D. G. (2023). Abnormal evidence accumulation underlies the positive memory deficit in depression. *Journal of Experimental Psychology: General*, 152(1):139.

Daw, N. D., Gershman, S. J., Seymour, B., Dayan, P., and Dolan, R. J. (2011). Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6):1204–1215.

Dezfouli, A., Griffiths, K., Ramos, F., Dayan, P., and Balleine, B. W. (2019). Models that learn how humans learn: The case of decision-making and its disorders. *PLOS Computational Biology*, 15(6):1–33.

Eckstein, M. K., Summerfield, C., Daw, N. D., and Miller, K. J. (2023). Predictive and interpretable: Combining artificial neural networks and classic cognitive models to understand human learning and decision making. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Eppinger, B., Walter, M., Heekeren, H. R., and Li, S.-C. (2013). Of goals and habits: age-related and individual differences in goal-directed decision-making. *Frontiers in neuroscience*, 7:253.

Ger, Y., Shahar, M., and Shahar, N. (2024). Using recurrent neural network to estimate irreducible stochasticity in human choice behavior. *Elife*, 13:RP90082.

Gillan, C. M., Kosinski, M., Whelan, R., Phelps, E. A., and Daw, N. D. (2016). Characterizing a psychiatric symptom dimension related to deficits in goal-directed control. *elife*, 5:e11305.

Ji-An, L., Benna, M. K., and Mattar, M. G. (2025). Discovering cognitive strategies with tiny recurrent neural networks. *Nature*, pages 1–9.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. (2023). Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

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

Macmillan, N. A. and Creelman, C. D. (2005). *Detection Theory: A User's Guide*. Lawrence Erlbaum Associates Publishers, Mahwah, NJ, 2 edition.

Makke, N. and Chawla, S. (2024). Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review*, 57(1):2.

Mihatsch, O. and Neuneier, R. (2002). Risk-sensitive reinforcement learning. *Machine learning*, 49(2):267–290.

Miller, K. J., Eckstein, M., Botvinick, M. M., and Kurth-Nelson, Z. (2023). Cognitive model discovery via disentangled rnns. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.

Musslick, S., Bartlett, L. K., Chandramouli, S. H., Dubova, M., Gobet, F., Griffiths, T. L., Hullman, J., King, R. D., Kutz, J. N., Lucas, C. G., et al. (2025). Automating the practice of science: Opportunities, challenges, and implications. *Proceedings of the National Academy of Sciences*, 122(5):e2401238121.

Otto, A. R., Skatova, A., Madlon-Kay, S., and Daw, N. D. (2015). Cognitive control predicts use of model-based reinforcement learning. *Journal of cognitive neuroscience*, 27(2):319–333.

Petersen, B. K., Landajuela, M., Mundhenk, T. N., Santiago, C. P., Kim, S. K., and Kim, J. T. (2019). Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*.

Rmus, M., Jagadish, A. K., Mathony, M., Ludwig, T., and Schulz, E. (2025). Generating computational cognitive models using large language models. *Advances in Neural Information Processing Systems*.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., et al. (2024). Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475.

Schmidt, M. and Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, pages 461–464.

Shojaee, P., Meidani, K., Gupta, S., Barati Farimani, A., and Reddy, C. (2025). LLM-SR: Scientific equation discovery via programming with large language models. In *International Conference on Learning Representations*.

Stanislaw, H. and Todorov, N. (1999). Calculation of signal detection theory measures. *Behavior research methods, instruments, & computers*, 31(1):137–149.

Wilson, R. C. and Collins, A. G. (2019). Ten simple rules for the computational modeling of behavioral data. *elife*, 8:e49547.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. (2024). Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Zorowitz, S., Solis, J., Niv, Y., and Bennett, D. (2023). Inattentive responding can induce spurious associations between task behaviour and symptom measures. *Nature human behaviour*, 7(10):1667–1681.

A Additional Dataset and Baseline Model Details

A.1 Baseline Model Specifications

A.1.1 Two-step Reinforcement Learning Task

Model-Free Temporal-Difference Learning Model. The first baseline model is a simplified variant of the model-free temporal-difference (TD) learning model of Daw et al. (2011). The model maintains separate action values for the first and second stages, updating first-stage values via credit assignment from second-stage outcomes without representing the task's transition structure.

The model maintains action values for the first stage, $Q_1(a)$, and state–action values for the second stage, $Q_2(s, a)$. At each stage $i \in \{1, 2\}$, choices are generated according to a softmax decision rule with inverse temperature parameter $\beta > 0$:

$$P(a_{i,t} = a \mid s_{i,t}) \propto \exp\big\{\beta\, Q_i(s_{i,t}, a)\big\}.$$

Learning proceeds sequentially within each trial. Following the second-stage choice, the second-stage value is first updated using the obtained terminal reward r_t:

$$Q_2^{t+1}(s_{2,t}, a_{2,t}) = Q_2^t(s_{2,t}, a_{2,t}) + \alpha\big[r_t - Q_2^t(s_{2,t}, a_{2,t})\big],$$

where $\alpha \in [0, 1]$ denotes the learning rate. Subsequently, the first-stage action value is updated via credit assignment from the updated second-stage value estimate:

$$Q_1^{t+1}(a_{1,t}) = Q_1^t(a_{1,t}) + \alpha\big[Q_2^{t+1}(s_{2,t}, a_{2,t}) - Q_1^t(a_{1,t})\big].$$

Importantly, the first-stage update explicitly uses the post-update second-stage value, thereby implementing a fully model-free credit assignment mechanism without representing the task's transition structure. Unlike the original formulation of Daw et al. (2011), which includes separate learning rates and inverse temperatures for each stage, an eligibility trace parameter λ, and a perseveration parameter, the present implementation shares a single learning rate and inverse temperature across stages and omits the eligibility trace and perseveration terms, yielding a two-parameter model (α, β) that prioritizes parsimony.

Hybrid Model-Based and Model-Free Learning Model. The second baseline integrates model-free (MF) temporal-difference learning with model-based (MB) planning using a mixing weight $w \in [0, 1]$, following the hybrid architecture of Daw et al. (2011). First-stage values are computed as a weighted combination of cached model-free values and model-based values obtained via one-step look-ahead using the known transition probabilities. Both models use softmax action selection and TD learning for value updates. This model maintains a model-free first-stage value $Q_{\text{MF}}(a_1)$, which is updated identically to Q_1 in the model-free baseline above, alongside second-stage values $Q_2(s, a)$. At the first stage, the net action value is computed as a convex combination of MF and MB values:

$$Q_{\text{net}}(a_{1,t}) = w \cdot Q_{\text{MB},t}(a_{1,t}) + (1 - w) \cdot Q_{\text{MF},t}(a_{1,t}),$$

where $w = 0$ corresponds to purely model-free control and $w = 1$ to purely model-based control. The model-based value is obtained via a one-step look-ahead using the known transition structure of the task:

$$Q_{\text{MB},t}(a_{1,t}) = \sum_{s_2} P(s_2 \mid a_{1,t}) \max_{a_2} Q_{2,t}(s_2, a_2).$$

First-stage choices are generated according to $P(a_{1,t} = a) \propto \exp\big\{\beta\, Q_{\text{net},t}(a)\big\}$, while second-stage choices follow $P(a_{2,t} = a \mid s_{2,t}) \propto \exp\big\{\beta\, Q_{2,t}(s_{2,t}, a)\big\}$. As with the model-free baseline, learning rates and inverse temperatures are shared across stages and the eligibility trace and perseveration parameters are omitted, resulting in a three-parameter model (α, β, w).

A.1.2 Probabilistic Reversal Learning Task: Temporal-Difference Learning Model

The temporal-difference learning model (Mihatsch and Neuneier, 2002) maintains stimulus values $V(j)$ for each of the three options $j \in \{0, 1, 2\}$, initialized at zero. On each trial t, choices are generated according to a softmax decision rule with inverse temperature $\beta > 0$:

$$P(c_t = j) \propto \exp\big\{\beta\, V_t(j)\big\}.$$

Following each choice, a prediction error is computed as $\delta_t = o_t - V_t(c_t)$, where $o_t \in \{0, 1\}$ denotes the observed outcome. The chosen stimulus value is then updated asymmetrically depending on the sign of the prediction error:

$$V_{t+1}(c_t) = V_t(c_t) + \eta(\delta_t) \cdot \delta_t, \quad \text{where} \quad \eta(\delta_t) = \begin{cases} \eta_+ & \text{if } \delta_t \geq 0, \\ \eta_- & \text{if } \delta_t < 0, \end{cases}$$

with $\eta_+, \eta_- \in [0, 1]$ denoting separate learning rates for positive and negative prediction errors, respectively. Values for unchosen stimuli remain unchanged. This asymmetry allows the model to capture individual differences in the relative weighting of better- and worse-than-expected outcomes. When $\eta_- > \eta_+$, negative prediction errors have a larger impact on subsequent values than positive prediction errors, which can be interpreted as relatively greater sensitivity to losses or non-rewards; the reverse holds when $\eta_+ > \eta_-$.

A.1.3 Recognition Memory Task: Signal Detection Model

We used a two-parameter equal-variance signal detection theory (SDT) model for the recognition memory task (Macmillan and Creelman, 2005; Stanislaw and Todorov, 1999). Whereas the original study reported SDT measures by computing d' and c from hit and false-alarm rates, we implement the same equal-variance SDT assumptions as a two-parameter generative model for trial-level likelihood evaluation. The model assumes that each presented item elicits a continuous memory-strength signal drawn from one of two Gaussian distributions: $\mathcal{N}(d'/2,\, 1)$ for old items and $\mathcal{N}(-d'/2,\, 1)$ for new items, where d' denotes discriminability. The observer responds "Yes" whenever this signal exceeds a criterion c, yielding the following endorsement probabilities:

$$P(\text{Yes} \mid \text{old}) = \Phi\big(d'/2 - c\big), \qquad P(\text{Yes} \mid \text{new}) = \Phi\big(-d'/2 - c\big),$$

where $\Phi(\cdot)$ denotes the standard normal cumulative distribution function. A liberal criterion ($c < 0$) produces elevated endorsement rates for both old and new items, whereas a conservative criterion ($c > 0$) suppresses them. The model has two free parameters: d' and c.

A.2 External Measures

In the main text, we reported external validity results for the planning and learning tasks. Here we describe the external measures available for each dataset.

Two-step reinforcement learning task. The dataset from Gillan et al. (2016) included the following self-report measures collected alongside the behavioral task: the Obsessive-Compulsive Inventory–Revised (OCI-R), the Self-Rating Depression Scale (SDS), the trait portion of the State-Trait Anxiety Inventory (STAI), and an estimate of general cognitive ability (IQ). Demographic variables included participant age and gender.

Probabilistic reversal learning task. The dataset from Zorowitz et al. (2023) included six self-report symptom and personality-trait measures administered before the behavioral task: the 7-Down scale, assessing depressive symptoms; the 7-Up scale, assessing hypomanic symptoms; the Generalized Anxiety Disorder 7-item scale (GAD-7); the Behavioral Inhibition System/Behavioral Activation System scales (BIS/BAS); the Snaith-Hamilton Pleasure Scale (SHAPS), assessing anhedonia; and the Penn State Worry Questionnaire (PSWQ). Demographic variables included participant age and gender.

Recognition memory task. The dataset from Cataldo et al. (2023) included depressive symptom severity measured by the Beck Depression Inventory–II (BDI-II), along with participant age and gender. No additional clinical or personality measures were available in this dataset.

B Dataset Splits and Model Evaluation

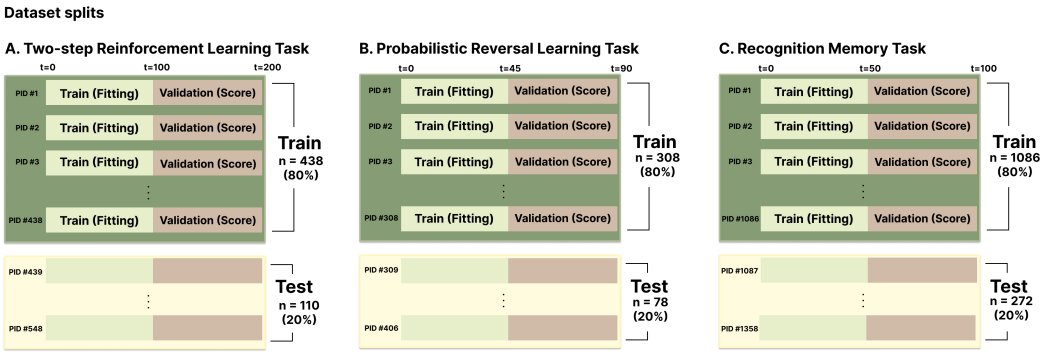

Figure 3: Overview of the behavioral datasets and data-splitting procedure. Participants were divided into train (80%) and test (20%) sets using random sampling.

To reduce overfitting during model selection and evaluate generalization to data from unseen participants, we employed a two-level data splitting strategy (Figure 3). At the participant level, the full sample was divided into train (80%) and test (20%) sets using random sampling. The held-out test set was reserved exclusively for final model evaluation after all model selection procedures were completed using the train set.

At the trial level, each participant's behavioral data were split temporally. For the two-step reinforcement learning task (200 trials), the first 100 trials were used for model fitting and the remaining 100 trials for computing validation scores. For the probabilistic reversal learning task (90 trials), the first 45 trials were used for fitting and the last 45 for validation. Lastly, for the recognition part of the memory task (100 trials), the first 50 trials were used for fitting and the last 50 trials for validation. During validation, model parameters remain fixed at the values estimated from the first half of the trials, while the model's latent states (e.g., Q-values) carry over and continue to update (see Optimization Objective in Section 3.2).

We adopted this data-splitting strategy to assess generalization to unseen trials while preserving the temporal order of learning. This was particularly important for learning tasks, in which behavior in later trials (i.e., second half) may depend on learning experiences in early trials (i.e., first half). Reversing the training and validation halves, as in bidirectional split-half cross-validation, would be unsuitable for evaluating predictions of future behavior.

However, this splitting method may not be appropriate for all cognitive tasks, particularly when the two halves differ qualitatively or when splitting the trial sequence is impractical. Therefore, DisCo allows users to specify alternative evaluation strategies, including using the full trial sequence without splitting, n-fold cross-validation, and random trial-level splitting, depending on the characteristics of the task.

C DisCo Algorithm

Algorithm 1 summarizes the full DisCo procedure, including LLM-based candidate generation, logical validation, validation-set scoring, and periodic pruning and migration across island buffers.

Algorithm 1 DisCo: Discovery of Interpretable Symbolic Cognitive Models

Input: Task $\mathcal{T}$, Template $\mathcal{S}$, behavioral data $\mathcal{D}$, optional external measures $\mathcal{M}$
Output: Optimal cognitive model ϕ^*

$(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}) \leftarrow \text{ChronologicalSplit}(\mathcal{D})$
Initialize island buffers $\mathcal{B} = \{B_1, \ldots, B_K\}$ with seed programs from $\mathcal{S}$
while budget not exhausted **do**
    // Evolutionary Hypothesis Generation
    Select island B_k and sample exemplars $\mathcal{P} \sim B_k$
    $\Phi_{\text{new}} \leftarrow \text{LLM.evolve}(\mathcal{T}, \mathcal{P})$
    // Model Validation
    for all $\phi \in \Phi_{\text{new}}$ **do**
        if $\text{LogicalValidation}(\phi)$ **then**
            $\hat{\theta} \leftarrow \text{MLE}(\phi, \mathcal{D}_{\text{train}})$
            $s \leftarrow \text{BIC}(\phi, \hat{\theta}, \mathcal{D}_{\text{val}} \mid \mathcal{D}_{\text{train}})$ // Compute BIC without resetting latent states
            Add (ϕ, s) to B_k
        end if
    end for
    // Model Selection & Pruning
    if $t \bmod T_{\text{clean}} = 0$ **then**
        $\mathcal{B} \leftarrow \text{Cleaner}(\mathcal{B})$ // Prune redundant params
        Migrate($\mathcal{B}$) // Exchange models between islands
    end if
end while
$\phi^* \leftarrow \text{argmin}_{\phi \in \Phi_{\text{candidates}}} \text{BIC}(\phi)$
if $\mathcal{M}$ is available **then**
    $\mathcal{R}_{\text{ext}} \leftarrow \text{Reporter}(\phi^*, \mathcal{M})$ // Summarize parameter–measure associations
end if
return ϕ^*

D Prompts

This section provides the task-specific candidate-generation prompts for each behavioral task, as well as the Cleaner prompt used for parameter pruning.

D.1 Two-step Reinforcement Learning Task

LLM Prompt for two-step reinforcement learning task

```
"""
Write a Python function for the Two-step Decision-Making Task.

[Context]
- Domain: Cognitive Science
- Goal: Maximize reward by making two consecutive choices.
- Structure:
  Stage 1 action (0/1) determines Stage 2 state (0/1) via a FIXED transition matrix:
  * Action 0 -> State 0 (70%), State 1 (30%)
  * Action 1 -> State 1 (70%), State 0 (30%)

[Constraints]
1. No `np.clip`, `np.random`, or external libraries (except `numpy` as `np`).
2. Minimize parameter count (Parsimony).
3. Do not modify Input/Output signatures.
4. No peeking: probs[t] must not depend on reward[t] (only past rewards).
5. No action cheating: probs1[t] must not use a1[t], probs2[t] must not use a2[t].
6. No negative indexing at t=0: avoid a1[t-1] without checking t > 0.

[Evaluation]
- BIC on train/validation splits.
- Validate no peeking, no action cheating, no negative indexing.
- Validate probability structure.
- Max parameters <= 10. But, less is better (Because BIC penalizes number of params).

[Input]
Dictionary `data` with fields (all shape: (n_trials,)):
- 'stage1_response': int {0,1} (Choice 1)
- 'stage2_response': int {0,1} (Choice 2)
- 'stage2_state'   : int {0,1} (State reached)
- 'reward'         : int {0,1} (Outcome)

[Output]
Return `((probs1, probs2), bounds)`:
- probs1, probs2: (n_trials, 2) arrays, rows sum to 1.0.
- bounds: list of (min, max) for each param.
"""

@evaluator.run
{evaluator_function}

@equation.evolve
{equation_to_evolve}
```

D.2 Probabilistic Reversal Learning Task

LLM Prompt for probabilistic reversal learning task

```
"""
Write a Python function for the Probabilistic Reward Learning (PRL) Task.

[Context]
- Domain: Cognitive Science
- Goal: Model how participants learn to choose the most rewarding option.
- Structure:
  6 blocks × 15 trials = 90 trials per participant.
  Each trial: participant selects one of 3 stimuli → receives binary reward (0 or 1).
  One stimulus is "correct" (higher reward probability) per block; it changes
  every 15 trials (reversal).
  The model must learn option values from experience (choices and outcomes),
  WITHOUT knowing  which stimulus is correct.

[Constraints]
1. No `np.clip`, `np.random`, or external libraries (except `numpy` as `np`).
2. Minimize parameter count (Parsimony).
3. Do not modify Input/Output signatures.
4. No peeking: probs[t] must not depend on outcomes[t] (only past outcomes).
5. No action cheating: probs[t] must not depend on choices[t] (only past choices).
6. No negative indexing at t=0: avoid outcomes[t-1] or choices[t-1] without
checking t > 0.

[Evaluation]
- BIC on train/validation splits (50:50 temporal split within each participant).
- Validate no peeking, no action cheating, no negative indexing.
- Validate probability structure.
- Max parameters <= 10. But, less is better (Because BIC penalizes number of params).

[Input]
Dictionary `data` with fields:
- 'choices'   : (n_trials,) int {0, 1, 2} - chosen stimulus index each trial
- 'outcomes'  : (n_trials,) int {0, 1}    - reward received each trial
- 'n_options' : int (always 3)             - number of available stimuli

[Output]
Return `((probs,), bounds)`:
- probs  : (n_trials, n_options) array - rows must sum to 1.0, values in [0, 1].
- bounds : list of (min, max) for each param.
"""


@evaluate.run
{evaluator_function}

@equation.evolve
{equation_to_evolve}
```

D.3 Memory Task

LLM Prompt for recognition memory task

```
"""
Write a Python function for Recognition Memory Task.

[Context]
- Domain: Cognitive Science / Memory
- Goal: Model the probability of recognizing a word as "old" or "new".
- Structure:
  Each trial: Presented word (old/new) → Participant responds Yes/No

[Constraints]
1. No `np.clip`, `np.random`, or external libraries (except `numpy` as `np` and
↪  `scipy`).
2. Minimize parameter count (Parsimony).
3. Do not modify Input/Output signatures.
4. The function must NOT use participant responses - only `is_old` is available as
↪  input.

[Evaluation]
- BIC on train/validation splits (50:50 within-subject temporal split).
- Max parameters <= 10. But, less is better (Because BIC penalizes number of params).

[Input]
Dictionary `data` with fields (all shape: (n_trials,)):
- 'is_old' : bool - True if the word was previously studied (old), False if new

[Output]
Return `(p_yes, bounds)`:
- p_yes  : (n_trials,) array - P(response=1 | trial) for each trial
- bounds : list of (min, max) for each param
"""

@evaluate.run
{evaluator_function}

@equation.evolve
{equation_to_evolve}
```

D.4 Cleaner Prompt

This prompt is used when the DisCo pipeline detects significant flaws in the generated model codes.

LLM Prompt for Cleaner

```
You are an expert in cognitive modeling parameter optimization.

**TASK:** Clean the following function by removing or merging redundant parameters.

**ORIGINAL FUNCTION:**
```python
{function_code}
```

REQUIRED CHANGES (based on statistical analysis across subjects):
{chr(10).join(f"{i+1}. {inst}" for i, inst in enumerate(instructions))}

RULES:
1. Remove or merge ONLY the parameters mentioned above
2. Update the bounds list accordingly (remove bounds for removed parameters)
3. Adjust parameter indexing
   (e.g., if you remove params[1], params[2] becomes params[1])
4. Keep the core model logic intact
5. Ensure the function still produces valid probability outputs

OUTPUT:
Return ONLY the cleaned Python function code.
```

E Additional Configuration Details

In addition to the settings reported in the main text, all DisCo experiments used 10 islands, generated four candidate programs per LLM prompt, included two exemplar functions as in-context examples per prompt, and used an island reset period of 3600 seconds. The Cleaner used thresholds of 0.9 for parameter correlation, 0.8 for covariance, and 0.01 for the variance ratio. External associations were assessed using Benjamini–Hochberg FDR correction with $p_{\mathrm{FDR}} < 0.05$, and candidate evaluation used a 60-second timeout per model. Candidate model parameters were estimated using constrained maximum likelihood estimation with L-BFGS-B.

Experiments were run on a workstation with an AMD Ryzen Threadripper PRO 5965WX CPU (24 cores / 48 threads), 512 GB RAM, and two NVIDIA RTX A6000 GPUs with 48 GB VRAM each. Each seed required approximately 1–2 hours of wall-clock time, depending on the task and the generated candidate models. Code and reproduction instructions are included in the supplementary materials.

Table 6: External assets used in this work.

Asset	**Source**	**Use in this work**	**License / Usage information**
Gillan et al. (2016) dataset	Public dataset / paper	Planning task evaluation	No explicit license found; used for academic research.
Zorowitz et al. (2023) dataset	Public dataset / paper	Learning task evaluation	MIT license.
Cataldo et al. (2023) dataset	Public dataset / paper	Memory task evaluation	No explicit license found; used for academic research.
Qwen2.5-32B	Model repository	LLM-based model generation	See original model license / terms.
vLLM	Open-source software	LLM serving	See original software license.

F Best Discovered Model

We present the best discovered models, selected by the lowest validation BIC across five random seeds. DisCo improves upon established models in two ways: by generating interpretable extensions of existing mechanisms and by identifying theoretically distinct candidate accounts that replace central explanatory mechanisms. The form of improvement depends on the task and the seed model.

F.1 Planning

For the planning task, we considered three baseline models. The null baseline predicts choices at chance level and contains no free parameters. The model-free TD learning model updates first-stage values based on second-stage outcomes. The hybrid model extends this model by additionally incorporating model-based planning through a mixing weight (w).

The models discovered from these baselines differed in how they modified the corresponding baseline accounts. The models discovered from the null and model-free baselines (F.1.1 and F.1.2, respectively) largely recovered mechanisms already present in the baseline models. We therefore focus on the model discovered from the hybrid baseline (F.1.3), which extends the baseline account with an additional interpretable mechanism.

This model preserves the mixture of model-based and model-free learning in the hybrid baseline. It adds two components: a global value-decay term and an outcome-dependent bias against repeating the previous first-stage action following an unrewarded trial. Importantly, this bias affects choice separately from the Q-learning value update. The discovered model therefore introduces an explicit outcome-dependent switching mechanism while preserving the core model-based and model-free structure of the hybrid baseline.

F.1.1 Null Baseline

Best discovered model from null baseline

```
def equation(data, params):
    """
    Improved model using Q-learning with decay for past rewards and incorporating
    state transitions explicitly.
    """
    n = len(data["stage1_response"])

    # Initialize Q-values for both stages
    Q1 = np.zeros(2)  # Q-values for stage 1 actions
    Q2 = np.zeros(2)  # Q-values for stage 2 actions

    alpha = params[0]  # Learning rate
    beta = params[1]   # Temperature parameter for softmax
    gamma = params[2]  # Discount factor for future rewards
    delta = params[3]  # Decay factor for past rewards

    # Ensure parameters are within valid ranges
    bounds = [
        (0.0, 1.0),  # Learning rate between 0 and 1
        (0.0, 10.0),  # Temperature parameter between 0 and 10
        (0.0, 1.0),  # Discount factor between 0 and 1
        (0.0, 1.0)   # Decay factor between 0 and 1
    ]

    probs1 = np.zeros((n, 2))
    probs2 = np.zeros((n, 2))

    for t in range(n):
        # Compute probabilities for stage 1 action
        prob_1_0 = np.exp(beta * Q1[0]) / (np.exp(beta * Q1[0]) + np.exp(beta * Q1[1]))
        prob_1_1 = 1 - prob_1_0
        probs1[t] = [prob_1_0, prob_1_1]
```

```
        # Update Q-values for stage 1 based on chosen action and transition probabilities
        if data['stage1_response'][t] == 0:
            next_state = data['stage2_state'][t]
            reward = data['reward'][t]
            Q1[0] = (1 - alpha) * Q1[0] + alpha * (reward + gamma * Q2[next_state])
        else:
            next_state = data['stage2_state'][t]
            reward = data['reward'][t]
            Q1[1] = (1 - alpha) * Q1[1] + alpha * (reward + gamma * Q2[next_state])

        # Compute probabilities for stage 2 action
        prob_2_0 = np.exp(beta * Q2[0]) / (np.exp(beta * Q2[0]) + np.exp(beta * Q2[1]))
        prob_2_1 = 1 - prob_2_0
        probs2[t] = [prob_2_0, prob_2_1]

        # Update Q-values for stage 2 based on chosen action and reward
        if data['stage2_response'][t] == 0:
            reward = data['reward'][t]
            Q2[0] = (1 - alpha) * Q2[0] + alpha * reward
        else:
            reward = data['reward'][t]
            Q2[1] = (1 - alpha) * Q2[1] + alpha * reward

        # Apply decay to Q-values
        Q1 *= (1 - delta)
        Q2 *= (1 - delta)

    return (probs1, probs2), bounds
```

F.1.2 Model-free Baseline

Best discovered model from model-free baseline

```
def equation(data, params):
    """
    Improved Python function for the Two-step Decision-Making Task.
    Stage 1 action (0/1) determines Stage 2 state (0/1) via a FIXED transition matrix:
    - Action 0 -> State 0 (70%), State 1 (30%)
    - Action 1 -> State 1 (70%), State 0 (30%)
    """
    a1 = data["stage1_response"]
    a2 = data["stage2_response"]
    r  = data["reward"]
    s2 = data["stage2_state"]
    n = len(a1)

    # Parameters
    alpha = params[0]      # Learning rate
    beta  = params[1]      # Inverse temperature
    gamma = params[2]      # Decay factor for past rewards

    # Parameter bounds
    bounds = [
        (0.0, 1.0),        # alpha
        (0.0, 20.0),       # beta
        (0.0, 1.0),        # gamma
    ]

    def softmax(beta, q):
        exps = np.exp(beta * (q - np.max(q)))
        return exps / np.sum(exps)

    # Initialize Q-values
    Q1 = np.zeros(2)          # Stage 1 Q-values (Action L, R)
    Q2 = np.zeros((2, 2))     # Stage 2 Q-values (State 0/1, Action L/R)
```

```
    # Initialize outputs
    probs1 = np.zeros((n, 2))
    probs2 = np.zeros((n, 2))

    for t in range(n):
        # Stage 1 (Choice)
        probs1[t] = softmax(beta, Q1)

        # Stage 2 (Choice)
        s = s2[t]
        probs2[t] = softmax(beta, Q2[s])

        # Learning
        a1_t = a1[t]  # Stage 1 action at trial t
        a2_t = a2[t]  # Stage 2 action at trial t

        # Stage 2 Update
        td2 = r[t] - Q2[s, a2_t]
        Q2[s, a2_t] += alpha * td2

        # Stage 1 Update
        td1 = Q2[s, a2_t] - Q1[a1_t]
        Q1[a1_t] += alpha * td1

        # Apply decay factor to past Q-values
        if t > 0:
            Q1 *= gamma
            Q2 *= gamma

    # Must return (probs1, probs2), bounds
    return (probs1, probs2), bounds
```

F.1.3 Hybrid Baseline

Best discovered model from hybrid baseline

```
def equation(data, params):
    """Improved version of `equation_v1` with decay factor."""
    a1 = data["stage1_response"]
    a2 = data["stage2_response"]
    r  = data["reward"]
    s2 = data["stage2_state"]
    n = len(a1)

    # Parameters
    alpha = params[0] # Learning rate
    beta  = params[1] # Inverse temperature
    w     = params[2] # Mixing weight (0: Pure MF, 1: Pure MB)
    bias  = params[3] # Bias towards avoiding previously punished actions
    decay = params[4] # Decay factor for Q-values

    # Parameter bounds
    bounds = [
        (0.0, 1.0),   # alpha
        (0.0, 20.0),  # beta
        (0.0, 1.0),   # w
        (-1.0, 1.0),  # bias
        (0.0, 1.0)    # decay
    ]

    def softmax(beta, q):
        exps = np.exp(beta * (q - np.max(q)))
        return exps / np.sum(exps)

    # Initialize Q-values
```

```
    Q_mf = np.zeros(2)
    Q2   = np.zeros((2,2))
    prev_a1 = 0
    prev_a2 = 0

    # Initialize outputs
    probs1 = np.zeros((n,2))
    probs2 = np.zeros((n,2))

    # Transition probabilities
    trans_probs = np.array([
        [0.7, 0.3], # Action 0 -> [State 0, State 1]
        [0.3, 0.7]  # Action 1 -> [State 0, State 1]
    ])

    for t in range(n):
        # Stage 1 Decision
        # Calculate Model-Based Values (Q_mb)
        max_Q2 = np.max(Q2, axis=1) # [Max val of State 0, Max val of State 1]
        Q_mb = np.dot(trans_probs, max_Q2)

        # Combine MB and MF (Q_net)
        Q_net = w * Q_mb + (1 - w) * Q_mf

        # Add bias towards avoiding previously punished actions
        if t > 0:
            if r[t-1] == 0:
                Q_net[prev_a1] -= bias

        # Action Selection using Q_net
        probs1[t] = softmax(beta, Q_net)

        # Stage 2 Decision
        s = s2[t]
        probs2[t] = softmax(beta, Q2[s])

        # Learning
        a1_t = a1[t]
        a2_t = a2[t]

        # Stage 2 Update (Standard TD)
        td2 = r[t] - Q2[s, a2_t]
        Q2[s, a2_t] += alpha * td2

        # Stage 1 Update (Model-Free only)
        td1 = Q2[s, a2_t] - Q_mf[a1_t]
        Q_mf[a1_t] += alpha * td1

        # Apply decay to all Q-values
        Q_mf *= (1 - decay)
        Q2 *= (1 - decay)

        # Update previous actions
        prev_a1 = a1_t
        prev_a2 = a2_t

    return (probs1, probs2), bounds
```

F.2 Learning

Two baseline models were tested for the learning task: the null baseline, which assigns equal choice probability to each option, and the TD learning baseline (Appendix A.1.2).

The model discovered from the null baseline (F.2.1) corresponds to a restricted case of the TD learning baseline in which $\eta_+ = \eta_-$, without a distinct learning mechanism. In contrast, the model discovered from the TD learning baseline (F.2.2) provides an interpretable alternative to the baseline. The model replaces the separate learning rates for positive and negative prediction errors with a single learning rate. It also introduces decay of unchosen-option values over time. As a result, individual differences are explained not by differential sensitivity to positive and negative outcomes, but by how quickly information about unchosen options is forgotten. A novel behavioral prediction of this account is that participants should switch more slowly to the newly advantageous option after a reversal when it has remained unchosen for longer.

F.2.1 Null Baseline

Best discovered model from null baseline

```
def equation(data, params):
    """Improved version of `equation_v1` with a temperature parameter."""
    n_trials = len(data['choices'])
    n_options = data['n_options']

    # Initialize value estimates for each option
    values = np.zeros(n_options)

    # Learning rate (free parameter)
    alpha = params[0]

    # Temperature parameter (free parameter) to control exploration
    temperature = params[1]

    # Initialize probability matrix
    probs = np.zeros((n_trials, n_options))

    for t in range(n_trials):
        # Softmax transformation to get probabilities
        exp_values = np.exp(values / temperature)
        total_exp_value = np.sum(exp_values)
        probs[t] = exp_values / total_exp_value

        # Update values based on the outcome
        if t < n_trials - 1:
            chosen_option = data['choices'][t]
            reward = data['outcomes'][t]
            prediction_error = reward - values[chosen_option]
            values[chosen_option] += alpha * prediction_error

    # Bounds for the learning rate (alpha) and temperature
    bounds = [(0, 1), (0.1, 10)]

    return (probs,), bounds
```

F.2.2 TD Learning Baseline

Best discovered model from TD learning baseline

```
def equation(data, params):
    """
    Risk-Sensitive TD Learning with Decay for Probabilistic Reward Learning.

    params: [alpha, beta, gamma]
      alpha : learning rate (single rate for both positive and negative prediction errors)
```

```
      beta  : inverse temperature (softmax)
      gamma : decay factor to reduce the value of unchosen options over time
    """
    alpha, beta, gamma = params[0], params[1], params[2]
    bounds = [
        (0.0,  1.0),   # alpha
        (0.0, 20.0),   # beta
        (0.0,  1.0),   # gamma
    ]

    choices   = data['choices']
    outcomes  = data['outcomes']
    n_options = data['n_options']
    n_trials  = len(choices)

    V     = np.zeros(n_options)
    probs = np.zeros((n_trials, n_options))

    for t in range(n_trials):
        logits   = beta * (V - V.max())
        probs[t] = np.exp(logits) / np.exp(logits).sum()

        delta = outcomes[t] - V[choices[t]]
        V[choices[t]] += alpha * delta

        # Decay the value of unchosen options
        for i in range(n_options):
            if i != choices[t]:
                V[i] *= (1 - gamma)

    return (probs,), bounds
```

F.3 Memory

For the memory task, we considered two baseline models. The null baseline predicts "yes" at a constant rate. The SDT baseline (Appendix A.1.3) predicts the responses using two parameters: the discriminability parameter d' captures the separation between the memory strength distributions for old and new items, while the criterion c sets the response threshold for a "yes" response.

The model discovered from the null baseline recovered separate parameters for the old-new distinction and overall response bias. The structure is analogous to the discriminability and criterion structure of the SDT baseline, although the discovered model uses a logit rather than the probit link used by the baseline.

The model discovered from the SDT baseline preserves the discriminability and criterion structure but applies a bounded sigmoid transformation to the criterion c. In the standard SDT model, changes in c continue to shift the decision threshold without saturation. In the discovered model, by contrast, the sigmoid transformation limits the influence of c on the threshold at extreme values.

F.3.1 Null Baseline

Best discovered model from null baseline

```
def equation(data, params):
    """
    Enhanced model: Introduces two parameters to capture the difference in recognition
    probabilities and overall response bias.
    - params[0]: bias (difference in log odds between old and new words)
    - params[1]: overall response bias (log odds of saying "Yes" regardless of old/new)
    """
    n = len(data['is_old'])
    bias = params[0]
    response_bias = params[1]

    # Compute the log odds for each trial
    log_odds = np.where(data['is_old'], bias + response_bias, response_bias)

    # Convert log odds to probabilities
    p_yes = 1 / (1 + np.exp(-log_odds))

    # Define bounds for the parameters
    bounds = [(-5, 5), (-5, 5)]

    return p_yes, bounds
```

F.3.2 SDT Baseline

Best discovered model from SDT baseline

```
def equation(data, params):
    """
    Improved version of `equation_v1` with a non-linear transformation of the criterion.

    Hit  = P("Yes" | old) = Phi( d_prime / 2 - sigmoid(c))
    FA   = P("Yes" | new) = Phi(-d_prime / 2 - sigmoid(c))

    Parameters:
    - d_prime: Sensitivity to the stimulus
    - c: Criterion (decision threshold)
    """
    d_prime, c = params[0], params[1]
    is_old = data['is_old']

    # Sigmoid function to transform the criterion
    sigmoid_c = 1 / (1 + np.exp(-c))

    p_hit = norm.cdf(d_prime / 2 - sigmoid_c)
```

```
    p_fa = norm.cdf(-d_prime / 2 - sigmoid_c)

    p_yes = np.where(is_old, p_hit, p_fa)

    bounds = [
        (-1.0, 10.0),  # d_prime
        (-3.0, 3.0),   # c
    ]

    return p_yes, bounds
```

G Comparison with GeCCo

We provide additional details on the comparison between DisCo and GeCCo (Rmus et al., 2025) reported in Section 4.3. We first describe the experimental setup, followed by an analysis of validation-rule violations in GeCCo-generated models and a qualitative comparison of the final model structures discovered by the two methods.

G.1 Experimental Setup

We compared DisCo and GeCCo on the planning, learning, and memory tasks. For each task, both methods were initialized from the same task-specific baseline model: the hybrid model for planning, the TD learning model for learning, and the signal detection theory (SDT) model for memory. Each method was run with five independent seeds under the same computational budget and LLM model (Qwen2.5-32B), generating 200 candidate models per seed. The final selected model from each run was evaluated on held-out data using the same evaluation procedure described in the main text.

For the BIC comparison, we report statistics only for runs in which the final selected model satisfied the validation criteria. This prevents mathematically or structurally invalid models from being treated as competitive simply because they achieve favorable numerical scores. In particular, because GeCCo allows LLM-generated models to implement their own likelihood computation, the validation procedure was extended to check consistency between the generated probability outputs and the corresponding likelihood calculation.

G.2 Validation-Rule Violations

We additionally examined the candidate models generated by GeCCo for violations of the validation criteria used in our evaluation. Table 7 summarizes the frequency of each violation type across tasks.

Table 7: Frequency of validation-rule violations among GeCCo-generated candidate models. A single candidate may violate more than one rule.

Task	Future peeking	Independence violation	Negative indexing	Non-determinism	Log-likelihood mismatch
Planning	56	105	0	33	42
Learning	7	149	0	46	11
Memory	0	69	1	83	51

Multiple types of violations occurred across the three tasks. Independence violations were particularly frequent in the planning and learning tasks, whereas uncontrolled non-determinism was common in the memory task. Future peeking and inconsistencies in likelihood computation were also observed, while negative indexing occurred only rarely.

Importantly, some of these violations affected final model selection. Across all three tasks, DisCo selected valid models in all five runs, whereas only three of five GeCCo runs produced valid final models. These results indicate that high model-fit scores alone do not necessarily imply that an LLM-generated model satisfies the structural and mathematical requirements of the task.

G.3 Qualitative Comparison of Discovered Models

Beyond validity and predictive performance, we qualitatively compared the structures of the final models selected by DisCo and GeCCo for each task.

Planning. The model discovered by DisCo retained the hybrid model-based/model-free architecture while introducing an outcome-dependent bias on the previous first-stage action. Specifically, the bias lowered the value of the previous first-stage action after an unrewarded trial. In contrast, the model discovered by GeCCo removed the explicit model-based/model-free mixture weight and did not include a mechanism that directly captured the effect of the previous outcome. Thus, the two methods produced structurally different accounts of behavior in the planning task.

Learning. The models discovered by DisCo and GeCCo for the learning task were somewhat similar to each other. Both replaced the baseline model's separate learning rates for positive and negative prediction errors with a single learning rate and introduced a decay mechanism. However, DisCo applied decay selectively to unchosen-option values, whereas GeCCo applied decay more broadly to all option values. Both suggested that forgetting is a relevant mechanism, despite differences in how it is implemented.

Memory. The DisCo model used a static, sigmoid-bounded decision criterion that depended on the stimulus but not on the participant's preceding responses. In contrast, the GeCCo models updated their decision criterion across trials based on response history. This allowed them to closely track the response sequence used for parameter estimation but led to poor generalization to held-out trials, as reported in the main text.

G.4 Held-Out Evaluation with Full Sequence Fitting

For the comparison in Table 2, we computed BIC scores for both methods using the data-splitting procedure that DisCo uses during model search. This alignment may favor DisCo. We therefore reevaluated the same set of valid final models under a protocol that matches how GeCCo fits models during search. Specifically, for each held-out test participant, we used the full trial sequence to estimate model parameters and to compute BIC.

Table 8: Held-out BIC comparison between baseline models, DisCo, and GeCCo after fitting model parameters on the full trial sequence. Statistics are reported as mean $\pm$ standard deviation over valid runs, excluding runs in which the final selected model failed to satisfy the validation criteria.

Task	Method	Valid runs	BIC Mean $\pm$ Std	BIC Range
Planning	Baseline (Hybrid)	–	478.05	–
	DisCo	5/5	384.91 ± 6.71	[372.55, 393.06]
	GeCCo	3/5	384.57 ± 6.00	[377.50, 392.16]
Learning	Baseline (TD Learning)	–	134.08	–
	DisCo	5/5	125.02 ± 2.67	[123.29, 130.31]
	GeCCo	3/5	125.41 ± 1.76	[123.92, 127.88]
Memory	Baseline (SDT)	–	83.51	–
	DisCo	5/5	89.92 ± 4.80	[83.75, 95.85]
	GeCCo	3/5	76.87 ± 0.39	[76.49, 77.41]

Table 8 summarizes the results. Under the full-sequence evaluation protocol, which may favor GeCCo, DisCo and GeCCo achieved comparable BIC scores for held-out participants in the planning and learning tasks. In the planning task, DisCo obtained a mean BIC of 384.91, compared with 384.57 for GeCCo. Similarly, in the learning task, the mean BIC scores were 125.02 and 125.41, respectively. Note that the BIC scores are not directly comparable between Tables 2 and 8 because the log-likelihood term in BIC is summed over different numbers of trials in the two analyses.

In the memory task, GeCCo achieved a lower mean BIC than DisCo (76.87 vs. 89.92). This contrasts with the primary analysis (Table 2), which used DisCo's split-half protocol for BIC calculation. This reversal suggests that the memory task may differ from planning and learning tasks in how consistently response patterns carry over from the first half to the second. Under the full-sequence protocol, shifts in response patterns across halves may be less apparent because BIC is computed on the same trials used to estimate model parameters. This protocol may favor more flexible models that can describe shifts in response patterns across the trial sequence, such as the history-dependent model discovered by GeCCo. Under the split-half protocol, however, this flexibility may have led to overfitting, as suggested by the substantially higher BIC of GeCCo's valid final models (837.91 vs. 56.03 for DisCo). DisCo appeared more robust across protocols: it showed a much smaller disadvantage under full-sequence evaluation than GeCCo did under split-half evaluation.

H Representative Validator Violations

This appendix provides representative examples of validity violations detected by the Validator. While the main results in Section 4.4 summarize violation rates across module ablations, here we illustrate what these violations look like at the program level. The examples are excerpts from generated candidate models, selected to illustrate the core invalid behavior. Future-peeking violations were not observed among the accepted candidate models in our ablation study, so we do not include a separate example for this category.

H.1 Non-determinism

This example shows a model that uses a random draw while computing choice probabilities. Here, `epsilon` denotes an exploration probability, and `probs1[t]` denotes the predicted probability distribution over the two first-stage actions at trial t. Because `np.random.rand()` changes across repeated calls, the same input data and parameters can produce different predicted probabilities.

```
if np.random.rand() < epsilon:
    probs1[t] = np.ones(2) / 2
```

H.2 Independence of Action

This example shows a model that uses the observed current action while computing the probability of that same action. Here, `a1[t]` is the participant's observed first-stage action at trial t, `Q_net` is the action-value vector used for prediction, and `probs1[t]` is the predicted choice probability. Because the model modifies `Q_net` using `a1[t]` before computing `probs1[t]`, the predicted probability depends on the answer it is supposed to predict.

```
# Add optimism bias
Q_net[a1[t]] += optim

# Action Selection using Q_net
probs1[t] = softmax(beta, Q_net)
```

H.3 Negative Indexing

This example shows a model that accesses the previous action using `a1[t-1]`. Here, `a1` is the sequence of observed first-stage actions and t is the trial index. At the first trial ($t = 0$), Python interprets `a1[t-1]` as `a1[-1]`, which refers to the final trial rather than a previous trial. This can inadvertently leak information from the end of the sequence into the first prediction.

```
h1 = H1 + np.array([eta * (a1[t-1] == 0), eta * (a1[t-1] == 1)])
```

I External Validity Study

Table 9: Full external validation results across random seeds. ✓ indicates the module was active; × indicates it was disabled.

Task	Model	Valid.	Clean.	Seed	BIC	FDR-significant correlations
Planning	MF	✓	✓	42	218.27	α: age $+.12$; β: IQ $+.16$
				123	221.24	β: IQ $+.16$
				456	223.57	α: age $+.13$; β: IQ $+.16$; δ: OCI $+.12$
				789	222.09	β: IQ $+.13$; γ: age $+.19$
				1234	220.66	α: age $+.12$; β: IQ $+.17$
		×	✓	42	123.48	n.s.
				123	218.27	α: age $+.12$; β: IQ $+.16$
				456	214.43	α: IQ $+.18$; β: age $+.13$; γ: age $+.17$
				789	222.63	α: age $+.14$; β: IQ $+.15$; δ: OCI $+.14$
				1234	218.25	α: age $+.12$; β: IQ $+.15$; γ: OCI $+.12$
		✓	×	42	212.72	α: age $+.14$; γ: age $+.13$, OCI $-.12$; δ: OCI $+.16$
				123	218.25	α: age $+.12$; β: IQ $+.15$; γ: OCI $+.12$
				456	218.27	α: age $+.12$; β: IQ $+.16$
				789	222.87	α: age $+.12$; β: IQ $+.16$
				1234	218.25	α: age $+.12$; β: IQ $+.15$; γ: OCI $+.12$
		×	×	42	208.13	α: IQ $+.15$; β: IQ $-.16$
				123	36.76	n.s.
				456	210.39	β: IQ $+.15$
				789	193.81	n.s.
				1234	142.65	α_1: age $+.15$, IQ $-.16$; decay: IQ $+.14$
Planning	Hybrid	✓	✓	42	224.94	β: IQ $+.17$; w: age $-.19$, IQ $+.19$
				123	221.83	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; γ: OCI $-.12$
				456	221.83	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; γ: OCI $+.12$
				789	221.83	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; γ: OCI $-.12$
				1234	216.20	β: IQ $+.14$; w: age $-.23$, IQ $+.16$; decay: OCI $+.15$
		×	✓	42	221.83	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; γ: OCI $+.12$
				123	221.83	β: IQ $+.15$; γ: OCI $+.12$; w: age $-.22$, IQ $+.12$
				456	218.75	β: OCI $+.11$; w: age $-.14$, IQ $+.15$; γ: OCI $+.12$
				789	139.72	α: IQ $+.11$; β: OCI $+.13$; w: age $-.15$, IQ $+.12$; γ: age $+.14$, OCI $-.11$; η: OCI $+.13$
				1234	173.32	α: IQ $+.20$; β: IQ $-.13$; γ: age $-.20$
		✓	×	42	226.43	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; δ: OCI $+.12$
				123	221.83	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; γ: OCI $+.12$
				456	215.18	w: age $-.17$, IQ $+.18$; bias: age $+.14$, OCI $-.13$; γ: OCI $+.14$
				789	221.83	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; γ: OCI $+.12$
				1234	226.43	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; γ: OCI $+.12$
		×	×	42	230.48	w: age $-.18$, IQ $+.15$; γ: IQ $-.13$
				123	226.64	β: IQ $+.15$; w: age $-.23$, IQ $+.12$; γ: OCI $+.12$; η: age $+.12$
				456	221.83	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; γ: OCI $-.12$
				789	221.83	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; γ: OCI $-.12$
				1234	221.83	β: IQ $+.15$; w: age $-.22$, IQ $+.12$; γ: OCI $+.12$
Learning	TD Learning	✓	✓	42	76.06	β: BIS $-.16$
				123	70.27	n.s.
				456	75.28	n.s.
				789	74.84	n.s.
				1234	75.37	n.s.
		×	✓	42	75.37	n.s.
				123	76.06	β: BIS $-.16$
				456	76.06	β: BIS $-.16$
				789	75.37	n.s.
				1234	75.46	n.s.
		✓	×	42	75.46	n.s.

(Continued from previous page)

Task	Model	Valid.	Clean.	Seed	BIC	FDR-significant correlations
				123	75.20	n.s.
				456	75.37	n.s.
				789	75.46	n.s.
				1234	70.27	n.s.
				42	70.27	n.s.
				123	23.21	n.s.
		×	×	456	29.79	n.s.
				789	75.46	n.s.
				1234	56.48	γ: BIS $-.20$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately describe DisCo's components and evaluation across three tasks; claims about Validator importance are supported by ablation results in Section 4.4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the Conclusion, including sensitivity to the provided baseline model, reliance on task-specific validation constraints, and the need for standardized evaluation protocols for cognitive model discovery.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper proposes an empirical framework and does not include theoretical proofs or formal theorems.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Experimental details including data splits, optimization procedure, hyperparameters, prompts, discovered model programs, and hardware are described in Section 4.1 and the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All datasets used are publicly available from the original publications. An anonymized supplementary code package is provided for reproduction, and the code will be publicly released on GitHub upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training/validation/test splits, optimizer (L-BFGS-B), temperature settings, number of generated candidates, search configuration, Cleaner thresholds, external validation criteria, and hardware specifications are described in Section 4.1 and the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Results are reported across 5 independent random seeds with mean and standard deviation. BIC ranges are also provided in Table 1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources are specified in Section 4.1 and Appendix E: experiments used Qwen2.5-32B served with vLLM on two NVIDIA RTX A6000 GPUs (48GB VRAM each), with an AMD Ryzen Threadripper PRO 5965WX CPU (24 cores / 48 threads) and 512 GB RAM. Each seed required approximately 1–2 hours of wall-clock time, depending on the task and generated candidate models.

   Guidelines:F

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research analyzes existing publicly available behavioral datasets and does not involve new data collection or harmful applications.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses potential positive impacts, including reduced manual effort and theoretical bias in cognitive model construction, as well as potential negative impacts from invalid or misleading discovered models and the need for domain-grounded validation.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release pretrained models or scraped datasets; all datasets are previously published and publicly available.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We provide source and license or usage information for the external datasets, models, and software used in this work in Appendix E. For assets without an explicit license, we indicate that no explicit license was found and use them only for academic research.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new datasets or pretrained models are released as part of this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper uses existing publicly available datasets; no new human subject data was collected.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No new human subjects research was conducted; all data used are from previously published studies.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are a core methodological component of DisCo, used as mutation operators in the evolutionary model discovery process. Details are provided in Section 3.3.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.