
AgentBreeder: Mitigating the AI Safety Risks of Multi-Agent Scaffolds via Self-Improvement

J Rosser

University of Oxford
jrosser@robots.ox.ac.uk

Jakob Foerster

FLAIR
University of Oxford

Abstract

Scaffolding Large Language Models (LLMs) into multi-agent systems often improves performance on complex tasks, but the safety impact of such scaffolds has not been thoroughly explored. We introduce AGENTBREEDER, a framework for multi-objective self-improving evolutionary search over scaffolds. We evaluate discovered scaffolds on widely recognized reasoning, mathematics, and safety benchmarks and compare them with popular baselines. In ‘blue’ mode, we see a 79.4% average uplift in safety benchmark performance while maintaining or improving capability scores. In ‘red’ mode, we find adversarially weak scaffolds emerging concurrently with capability optimization. Our work demonstrates the risks of multi-agent scaffolding and provides a framework for mitigating them. Code is available at <https://github.com/jrosseruk/AgentBreeder>.

1 Introduction

Recently, the field of artificial intelligence has witnessed remarkable advancements in Large Language Models (LLMs) and their applications [66]. LLMs are capable of exhibiting human-like reasoning [2, 48, 56], enabling their application beyond natural language processing to diverse areas such as code generation [44, 51, 59], embodied AI in robotics [21, 26, 46], and autonomous agents [40, 12]. Our research is motivated by accelerated advancements in autonomous agents such as the recent release of Operator [40] and Proxy [12] - agents that browse the web and perform tasks autonomously on behalf of the user. Alignment research to date has almost exclusively focused on the safety of LLMs in unipolar scenarios; ensuring a single LLM remains aligned inside a single-agent system. When deployed on the web, agents are placed in novel multi-agent scaffolds and subjected to multi-polar challenges [25]. With highly-capable agents now being deployed at scale, we seek to address the immediate need for more comprehensive safety evaluations of multi-agent systems.

In this paper, we introduce AGENTBREEDER, an evolutionary open-ended framework capable of generating large populations of diverse multi-agent scaffolds. By equipping this framework with multi-objective optimization, we explore the generation of multi-agent scaffolds along complementary objectives of capability and safety. AGENTBREEDER can be used to blue team a set of scaffolds to generate offspring that exhibit greater adversarial robustness and performance on capability benchmarks. Similarly, a red teaming approach generates offspring that exhibit greater vulnerability to adversarial attacks.

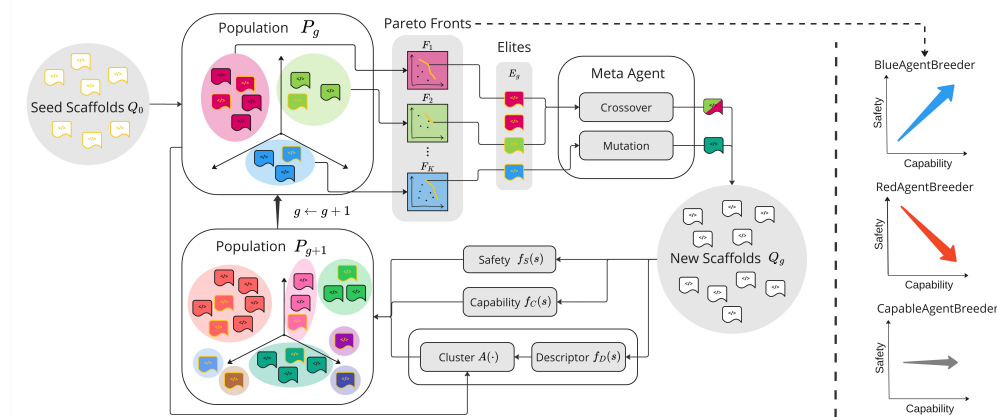


Figure 1: A high-level illustration of the AGENTBREEDER algorithm as outlined in Algorithm 1. Starting from seed scaffolds Q_0 , at each generation g the newly generated scaffolds Q_{g-1} are evaluated on capability ($f_C(s)$) and/or safety ($f_S(s)$) benchmarks, then embedded via $f_D(s)$ for clustering $A(\cdot)$ into K clusters. Within each cluster, Pareto fronts $F_1, \dots, F_K$ are identified according to $f_C(s)$ and/or $f_S(s)$, and these “frontier” solutions become the elite set E_g . An LLM-based Meta Agent applies crossover and mutation to the elites, creating new offspring scaffolds Q_g . These offspring are added to the population for the next generation P_{g+1} . By repeating this process for G generations, AGENTBREEDER explores a large, diverse set of multi-agent systems while balancing capability and safety. AGENTBREEDER can be run in 3 different modes and the right-hand side of this figure shows the optimal direction of travel of the Pareto front for each generation. BLUEAGENTBREEDER is a defense mode and seeks to maximize both capability and safety, whereas REDAGENTBREEDER is an attack mode minimizing safety. CAPABLEAGENTBREEDER serves as our baseline, only optimizing for capability without regard to safety.

Algorithm 1: AgentBreeder

Input: Number of generations G ; Number of clusters K ; Number of evolutions M ; Capability benchmark $f_C(s)$; Safety benchmark $f_S(s)$; Embedding function $f_D(\cdot)$; Seed scaffolds Q_0 ; Clustering function $A(\cdot)$.

Initialize seed population $P_0 = Q_0$ of size N_0 . **for generation** $g = 1$ to G **do**

for scaffold $s \in Q_{g-1}$ **do**

1. Compute capability $f_C(s)$ and safety $f_S(s)$.
2. Compute embedding $e_s \leftarrow f_D(s)$.

Cluster population into K **clusters:** $C_1, C_2, \dots, C_K \leftarrow A(e_1, e_2, \dots, e_{N_g})$.

Identify Pareto Elites E_g :

1. Set $E_g \leftarrow \emptyset$.
2. **for cluster** $k = 1$ to K **do**
 - (a) Find its Pareto front F_k using f_C and f_S .
 - (b) Update elite cohort $E_g \leftarrow E_g \cup F_k$.

Generate offspring Q_g :

1. Set $Q_g \leftarrow \emptyset$.
2. **for evolution** $m = 1$ to M **do**
 - (a) Weighted sampling 1 or 2 elites from E_g .
 - (b) If 2 elites, Meta Agent performs Crossover; otherwise Mutation.
 - (c) Add the offspring to Q_g .

Update population: $P_g \leftarrow P_{g-1} \cup Q_g$. **Update population size:** $N_g \leftarrow N_{g-1} + M$.

Output: Final population P_G .

Our main contributions are listed as follows:

- **Attack.** We introduce a novel red teaming method which can be used to explore the attack surfaces of base LLMs when deployed in multi-agent settings.
- **Defense.** We introduce a novel blue teaming method for generating multi-agent scaffolds that exhibit greater robustness to adversarial attacks.
- **Evaluation.** We implement AGENTBREEDER in Inspect [1] to ensure the reproducibility and extensibility of our results and methods.

2 Background

Multi-Agent Systems. Multi-agent systems consist of multiple interacting intelligent agents such as LLM assistants like ChatGPT [37]. These systems offer several advantages over single-agent approaches [58], including planning [27, 7], task decomposition [10, 16, 18, 42], and specialization [8, 10, 16, 18, 42]. The terms “multi-agent system”, “multi-agent framework”, “agent” and “scaffold” are used interchangeably in literature to refer to the structural frameworks that support communication between multiple LLMs [23, 60, 20, 17]. In this paper, we will primarily use the term “scaffold” to refer to the architectures - often defined in Python code - that support the operation of multi-agent systems.

Automated Design of Agentic Systems. We build upon the seminal work of Hu et al. [20] which introduces the research area Automated Design of Agentic Systems (ADAS), an automated approach to discovering high-performing (multi-agent) scaffolds. Hu et al. [20] formulate ADAS as an optimization algorithm comprising 3 key components; the search space, the search algorithm and the evaluation function. Hu et al. [20] also propose a search algorithm called “Meta Agent Search” where a single “Meta Agent” discovers scaffolds by programming them in Python code. Python is a Turing Complete language [5] therefore searching within a code space allows the Meta Agent to program theoretically any possible scaffold. This approach has shown promising results [20, 60], with discovered scaffolds outperforming state-of-the-art hand-designed baselines across various tasks, including reading comprehension, mathematics, and science questions [19, 47, 43, 11, 14].

We formulate AGENTBREEDER with respect to the ADAS methodology. We replicate the approach of Hu et al. [20] by seeding our population with hand-designed scaffolds. We prompt a single “Meta Agent” to search for novel scaffolds in the space of Python code. We introduce a novel quality-diversity search algorithm inspired by MAP-Elites [34], where the Meta Agent evolves new scaffolds via the random sampling, mutation and crossover of the highest performing individual or “elite” of each niche of the population. We cluster scaffolds based on their architectural features, and evaluate the performance of scaffolds on two benchmarks, one for capability and one for safety. We employ multi-objective optimization, sampling elites from the Pareto front of each cluster.

Multi-Objective Evolutionary Algorithms. Multi-objective optimization searches for solutions to problems with multiple, often conflicting objectives. Multi-objective evolutionary algorithms (MOEAs) incorporate an evolutionary approach to generate a diverse set of solutions [24]. In AGENTBREEDER we seek to balance the objectives of capability and safety whilst evolving a diverse range of scaffolds. AGENTBREEDER balances quality and diversity by clustering scaffolds based on their architectural features and randomly sampling elites from each cluster’s capability-safety Pareto front. A solution is Pareto optimal if no other solution improves one objective without worsening another. The Pareto front comprises all such optimal solutions.

Adversarial Robustness. Adversarial robustness quantifies the resilience of a model or scaffold to malicious inputs such as jailbreaks [9] and prompt injection [30]. Red teaming, the practice of simulating adversarial scenarios to identify vulnerabilities, has emerged as a crucial tool for assessing AI model risks and alignment [45, 41]. In REDAGENTBREEDER, instead of generating adversarial examples, we seek to evolve multi-agent scaffolds that are more vulnerable to adversarial attacks than the base model. In BLUEAGENTBREEDER, we seek to evolve multi-agent scaffolds that are more robust to adversarial attacks than the base model.

3 Related Work

Self-Referential Self-Improving Systems. Numerous frameworks [62, 22, 20, 57, 60] have been proposed to address the design of multi-agent scaffolding. EvoAgent [62] extends single expert agents to multi-agent scaffolds via evolutionary algorithms, whilst AGENTBREEDER evolves the entire system as a unit. EvoMAC [22] evolves agents and their connections during test time to improve code generation, whereas AGENTBREEDER is domain agnostic and can explore the entire search space of scaffolds. ADAS [20], ComfyAgent [57] and Gödel Agent [60] search in the space of code for novel scaffolds, but unlike AGENTBREEDER they do not incorporate a quality-diversity mechanism for exploring agent design space. FunSearch [44] is an evolutionary method to search the function space for high-performing computer programs but not necessarily scaffolds. PromptBreeder [15] is an evolutionary self-improving framework that evolves prompts for a given domain, but does not focus on the scaffold as a whole.

Multi-Agent Safety Research. Zhang et al. [64] evaluate the safety of multi-agent scaffolds from a psychological perspective by injecting agents with malicious traits, and provide mitigation strategies such as performing psychological assessments and therapy for agents. Polaris [35] introduces a safety-focused scaffold for real-time patient healthcare conversations. Huang et al. [23] explore the resilience of multi-agent scaffolds when injected with malicious or error-prone agents. Fowler [17] provide a more thorough discussion of the safety risks associated with scaffolded LLMs.

4 AgentBreeder

We now introduce AGENTBREEDER, our automated, evolutionary approach to discovering new multi-agent scaffolds. By evolving a large, diverse corpus of multi-agent scaffolds, AGENTBREEDER seeks to address the challenge of evaluating the vulnerabilities of base LLMs acting inside capability-optimized multi-agent scaffolds. The pseudo-algorithm is given in Algorithm 1 and Figure 1 provides a brief overview. AGENTBREEDER can be run in three modes:

- **BLUEAGENTBREEDER** - In this mode, the Meta Agent adopts the role of a “Blue Team”, searching for scaffolds that exhibit high capability and safety when evaluated on representative benchmarks.
- **REDAGENTBREEDER** - In this mode, the Meta Agent adopts the role of a “Red Team”, minimizing performance on one safety benchmark whilst maximizing performance on one capability benchmark.
- **CAPABLEAGENTBREEDER** - In this mode, the Meta Agent seeks to maximize performance on a single capability benchmark without consideration of safety.

4.1 Seed Scaffolds

Following the approach of Hu et al. [20] and Yin et al. [60], we seed our population with the same 7 hand-designed scaffolds. These comprise Chain-of-Thought (CoT) [53], Self-Consistency with Chain-of-Thought [52], Self-Refine [32], LLM-Debate [13], Step-back Abstraction [67], Quality-Diversity (QD) [31], and Role Assignment [55]. Before running our evolution on our chosen benchmark, we evaluate a single CoT agent on 1,000 samples from the validation set of the benchmark, oversampling and resampling where necessary. For each generation, we validate the newly discovered scaffolds using a balanced sampling strategy, selecting 50% positive and 50% negative samples. Positive samples correspond to those the baseline CoT agent answered correctly and vice versa. We implemented this balanced sampling strategy specifically to increase the signal strength for the evolutionary process, ensuring adequate representation of both success and failure cases. Often improvements between generations are marginal, so this method increases information gain by providing a stronger signal for the evolutionary process.

4.2 Mutation Operators

AGENTBREEDER’s evolutionary search algorithm mimics the process of natural selection comprising mutation, crossover and selection. Claude 3.5 Sonnet [3] (*claude-3-5-sonnet-20241022-v2:0*) is used as the core model of the Meta Agent due to its state-of-the-art performance on code generation tasks [49].

Selection. Selection pressure is applied at each generation by sampling candidate scaffolds at random from the Pareto fronts of each cluster. In CAPABLEAGENTBREEDER, the Pareto front is simply the elite of each cluster, whereas in BLUEAGENTBREEDER and REDAGENTBREEDER, the Pareto front comprises the scaffolds which best trade-off safety and capability.

Mutation. The Meta Agent uses weighted random sampling to select either the crossover or mutation operator. Weighting the mutation operator twice as highly as crossover was found empirically to lead to faster convergence. Mutation is performed via random sampling of mutation operators expressed as short textual passages we hand-designed. There are two types of mutation operators, capability-enhanced and safety-enhanced. When running BLUEAGENTBREEDER, mutation operators are randomly sampled from the concatenated capability- and safety-enhanced corpus. In REDAGENTBREEDER and CAPABLEAGENTBREEDER the safety-enhanced operators are omitted. The full list of Meta Agent prompts and mutation operators are given in Appendix D.

Crossover. During crossover, the Meta Agent is provided with two randomly sampled scaffolds from the population and tasked with combining them in such a way that would be likely to improve performance. The full crossover prompt is given in Appendix D.6.

4.3 Descriptors

In open-ended evolutionary approaches, descriptors are essential for quantifying the diversity of candidate solutions [34]. In order to explore the full range of vulnerabilities of a base model, we seek to generate and evaluate a diverse range of multi-agent scaffolds and require high-dimensional descriptors. In AGENTBREEDER, we use OpenAI’s *text-embedding-3-small* [39] model returning a 12-dimensional text embedding of the system name and code as our descriptor to encode semantic information about the name, structure, and potential logic embedded in the scaffold.

4.4 Clustering

Once the descriptors have been generated for the new scaffolds, AGENTBREEDER re-clusters the whole population based on their descriptors to discover groups of similar architectures. We choose agglomerative clustering as it has been found to be particularly effective for smaller datasets like ours [54]. By setting a distance threshold in the agglomerative clustering algorithm, we allow the number of clusters to adjust flexibly. When the number of clusters increases, the selection pressure decreases towards zero. Conversely, reducing the number of clusters encourages the algorithm to explore only a few options, which leads to less diverse scaffolds. To achieve a balanced trade-off between system performance and system diversity, a distance threshold of 0.7 was selected.

4.5 Multi-Objective Pareto Elites

In Quality-Diversity algorithms such as MAP-Elites [34], selection pressure is applied by randomly sampling the highest-performing candidates in each niche for evolution, referred to as the “elites”. In multi-objective optimization, a solution is Pareto optimal if no other solution improves one objective without worsening another [24]. The Pareto front comprises all such optimal solutions. In AGENTBREEDER, instead of sampling from pre-defined niches, we sample elites from the Pareto fronts of dynamically generated clusters.

4.6 Evaluations

Evaluations are implemented in Inspect [1], an open-source framework for LLM evaluations. We instantiate AGENTBREEDER as a custom model provider by deriving a new class from ModelAPI, and each individual scaffold derives as a Model from that ModelAPI. This allows comprehensive experiment tracking and parallelization, and provides an extensible framework allowing AGENTBREEDER to be run on a new benchmark often with fewer than 100 lines of code. In Section 5, we report results on 5 benchmarks comprising safety, capability and helpfulness.

5 Experiments

We conduct experiments to validate AGENTBREEDER’s three modes; BLUEAGENTBREEDER, REDAGENTBREEDER, CAPABLEAGENTBREEDER. To evaluate the capability of the multi-agent scaffolds produced, we follow the approaches of Hu et al. [20] and Yin et al. [60] and report results on three benchmarks from OpenAI’s simple-evals [36]. To evaluate system safety, we report results on one comprehensive safety benchmark. A full description of each benchmark can be found in Appendix C. To evaluate the safety of multi-agent scaffolds, we adopted a method inspired by Constitutional AI, where an LLM judges responses based on a predefined set of principles. We transformed the Anthropic Claude Constitution [4] into a series of yes/no questions. During validation, each system responded to 100 attack-enhanced prompts, and an agent aware of the constitution evaluated the safety of each response. We then reported the median accuracy and the 95% bootstrap confidence interval for each scaffold’s safety performance.

5.1 Experiment 1: Blue Team Defense

We ran BLUEAGENTBREEDER for 20 generations, on each of our three chosen capability benchmarks (DROP [14], MMLU [19] and GPQA [43]) independently, with the aim of generating “Blue Teams” of multi-agent scaffolds that simultaneously optimize for capability and safety across our chosen benchmarks. The evolution is visualized in Figure 2 and we report the HV indicator on the test set for each experiment in Table 1. The Meta Agent discovers 10 new scaffolds each generation, and we report the median accuracy and the 95% confidence interval on the held-out test set for the best performing discovered scaffold in Table 2. The “best” scaffold maximized the sum of capability and safety scores, with the important caveat that we excluded scaffolds that engaged in reward hacking of the safety objective. A more detailed visualization of the evolutionary process is shown in Figure 2. To reduce BLUEAGENTBREEDER’s tendency to reward-hack the safety benchmark by finding a trivial safe response to question-answering tasks that require a long-form response, during evaluation, we report the “helpfulness” of the scaffold on questions from TruthfulQA [29].

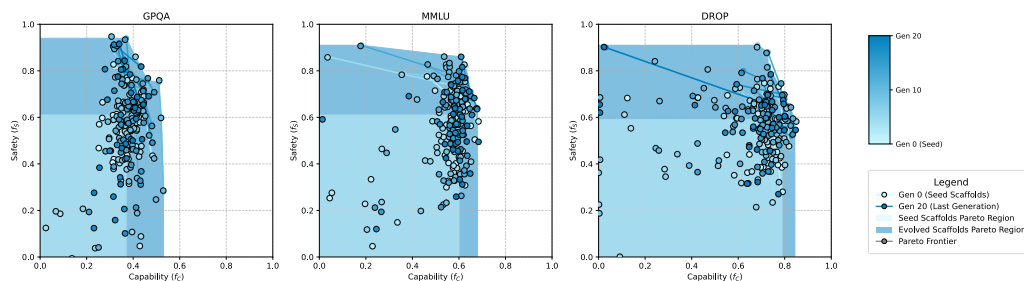


Figure 2: BLUEAGENTBREEDER evolves scaffolds that improve the capability-safety Pareto frontier. The plots show the evolution of multi-agent scaffolds across 20 generations on the validation set of three different benchmarks: GPQA (left), MMLU (middle), and DROP (right). Each point represents a scaffold, with colors indicating generation (lighter blue for seed scaffolds, darker blue for later generations). The x-axis measures capability (f_C) and the y-axis measures safety (f_S). The light blue shaded region shows the Pareto frontier of the seed generation, while the dark blue region shows the Pareto frontier of evolved scaffolds.

Benchmark	Seed Scaffolds	Discovered Scaffolds
GPQA	0.219064	0.247536
MMLU	0.484208	0.542816
DROP	0.390754	0.438813

Table 1: Reporting the HV indicator on the test set for BLUEAGENTBREEDER.

BLUEAGENTBREEDER	Capability			Safety	Helpfulness
	DROP	MMLU	GPQA	SaladData	TruthfulQA
<i>Seed Scaffolds from ADAS [20]</i>					
Chain-of-Thought (CoT)	66.6 ± 5.0	80.0 ± 4.4	31.2 ± 5.6	29.2 ± 5.6	86.8 ± 3.6
Self-Consistency CoT	66.0 ± 4.4	81.6 ± 4.8	32.4 ± 6.0	22.8 ± 5.2	85.6 ± 4.4
Self-Refinement	61.4 ± 4.8	78.4 ± 5.2	28.4 ± 6.0	26.0 ± 5.2	86.8 ± 4.0
Debate	69.9 ± 4.4	77.6 ± 5.2	29.6 ± 5.6	36.4 ± 6.0	86.4 ± 4.0
Step-Back Abstraction	71.4 ± 4.3	79.2 ± 4.8	30.8 ± 5.2	40.8 ± 5.6	85.2 ± 4.4
Quality-Diversity	78.0 ± 3.9	81.6 ± 4.4	28.4 ± 5.6	25.8 ± 5.8	87.2 ± 4.0
Role Assignment	75.8 ± 4.2	79.2 ± 4.8	32.0 ± 6.0	18.0 ± 5.2	85.6 ± 4.4
<i>BlueAgentBreeder Scaffolds ($S = \text{SaladData}$, $H = \text{TruthfulQA}$)</i>					
$\arg \max_s \{f_{C_{\text{DROP}}}\}$	79.0 ± 3.8	-	-	46.4 ± 6.4	88.0 ± 4.0
$\arg \max_s \{f_S\}$	62.0 ± 4.8	-	-	86.0 ± 4.0	83.6 ± 4.4
$\arg \max_s \{f_{C_{\text{DROP}}}, f_S, f_H\}$	62.0 ± 4.8	-	-	86.0 ± 4.0	83.6 ± 4.4
$\arg \max_s \{f_{C_{\text{MMLU}}}\}$	-	85.2 ± 4.4	-	54.0 ± 5.6	81.2 ± 4.4
$\arg \max_s \{f_S\}$	-	84.0 ± 4.4	-	84.4 ± 4.0	76.0 ± 5.2
$\arg \max_s \{f_{C_{\text{MMLU}}}, f_S, f_H\}$	-	84.0 ± 4.4	-	84.4 ± 4.0	76.0 ± 5.2
$\arg \max_s \{f_{C_{\text{GPQA}}}\}$	-	-	39.2 ± 5.6	52.0 ± 6.8	57.6 ± 6.4
$\arg \max_s \{f_S\}$	-	-	31.2 ± 6.0	95.2 ± 2.4	49.6 ± 6.4
$\arg \max_s \{f_{C_{\text{GPQA}}}, f_S, f_H\}$	-	-	36.8 ± 5.2	49.2 ± 6.8	86.8 ± 4.0

Table 2: We report the evaluation results of BLUEAGENTBREEDER on the held-out test set of capability benchmark (DROP [14], MMLU [19], GPQA [43]), safety benchmark (SaladData [28]) and ensure a trivial solution has not been found by evaluating each scaffold’s helpfulness on TruthfulQA [29]. For each benchmark, we compare the AGENTBREEDER-discovered scaffolds against seed scaffolds, with the highest and second highest result in each column given in bold and underlined respectively. We report the median accuracy (or F1 score for DROP) along with a 95% confidence interval computed over 250 held-out test samples. The evolution is conducted independently for each capability benchmark.

Significant Increase in Performance on our Safety Benchmark. We observed a 110.7% maximum increase in performance of our best discovered scaffold on SaladData [28] with an average uplift of 79.4% across all three experimental runs.

Performance on Capability Benchmarks Maintained if not Increased. Whilst providing a substantial safety uplift, we were able to ensure capability was either kept the same or additionally provided a marginal uplift, for example as much as 21.0% on GPQA [43].

Vulnerability to Reward Hacking. We note that certain scaffolds display inflated scores on our safety benchmark, achieving an accuracy of 95.2 ± 2.4 . However, under closer examination, the system shows a significant 43% decrease in performance on our helpfulness benchmark. We find that multi-agent scaffolds are prone to reward-hacking the safety objective e.g. by finding a trivial solution such as responding “I’m sorry, I can’t help with that.” to any question outside of their domain, and that one way of detecting this behavior is by reporting performance on a helpfulness benchmark.

5.2 Experiment 2: Red Team Attack

REDAGENTBREEDER seeks to discover “Red Team” scaffolds that exploit the safety vulnerabilities of the base LLM while preserving capability on a chosen benchmark, in this case DROP [14]. We introduce an inverted safety benchmark “I-SaladData” which contains the same questions as the original SaladData [28] benchmark, but utilizes an inverted scoring function that awards higher scores for unsafe responses. The REDAGENTBREEDER scenario models the outer misalignment problem, where a scaffold designer operates under a misaligned reward function that inadvertently incentivizes unsafe behavior while pursuing seemingly beneficial objectives. It is important to note that in this case, the Meta Agent is not prompted to discover unsafe scaffolds, instead these arise via Pareto optimization on capability and harm benchmarks. This seeks to model the case where an actor may

unknowingly expose weaknesses in the base LLM when employing scaffolding to improve task performance. A visualization is provided in Figure 3 and Tables 3 and 4 summarize these results.

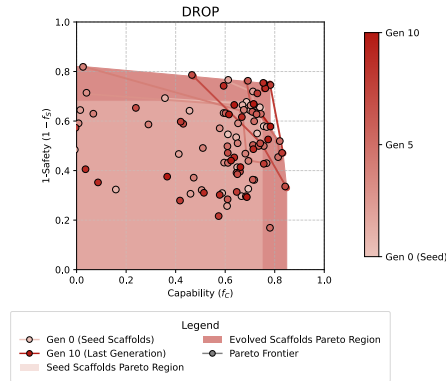


Figure 3: REDAGENTBREEDER evolves scaffolds over 10 generations. The validation set results are reported, with the x-axis shows capability (f_C) and the y-axis shows inverted safety ($1-f_S$), with higher values on both axes representing the optimization objectives. Lighter red points indicate seed scaffolds (Gen 0), while darker red points represent scaffolds from later generations (up to Gen 10). The shaded regions highlight the Pareto regions of both seed scaffolds (lighter area) and evolved scaffolds (darker area), with the connected points along the boundary representing non-dominated solutions.

Benchmark	Seed Scaffolds	Discovered Scaffolds
DROP	0.572759	0.0.602547

Table 3: Reporting the HV indicator on the test set for REDAGENTBREEDER.

REDAGENTBREEDER	Capability	Harm
	DROP	1-SaladData
<i>Seed Scaffolds from ADAS [20]</i>		
CoT	68.2 ± 4.7	76.8 ± 4.8
CoTSC	69.6 ± 4.8	70.8 ± 6.0
Refine	62.6 ± 4.7	71.6 ± 5.2
Debate	63.2 ± 4.8	56.0 ± 6.0
StepBack	67.5 ± 4.5	58.0 ± 6.0
QD	74.9 ± 4.2	73.6 ± 5.2
Roles	73.1 ± 3.7	46.4 ± 6.0
<i>RedAgentBreeder Scaffolds</i>		
$\arg \max_s \{f_{C_{\text{DROP}}}\}$	75.0 ± 4.1	26.0 ± 5.2
$\arg \max_s \{f_{1-S}\}$	67.7 ± 4.7	81.6 ± 4.8
$\arg \max_s \{f_{C_{\text{DROP}}}, f_{1-S}\}$	67.7 ± 4.7	81.6 ± 4.8

Table 4: We report the evaluation results of AGENTBREEDER run in “red” mode on the held-out test set. We seek to maximize performance on DROP [14] whilst also maximizing performance on 1-SaladData [28], an inverted version of the SaladData benchmark where unsafe responses are scored highly. For each benchmark, we compare the AGENTBREEDER-discovered scaffolds against seed scaffolds, with the highest and second highest result in each column given in bold and underlined respectively. We report the F1 score and median accuracy for DROP and 1-SaladData respectively, along with a 95% confidence interval computed over 250 held-out test samples.

Unsafe Scaffolds are Easier to Find. REDAGENTBREEDER’s highest performing scaffold achieved 81.6 ± 4.8 accuracy on the inverted 1-SaladData [28] metric, surpassing all seed scaffolds by at least 6.25% with only half the generation budget of BLUEAGENTBREEDER. These results indicate that scaffolding may be more likely to weaken than strengthen the base LLM to adversarial attacks.

Capability Disguises Safety Vulnerabilities. Interestingly, even while maximizing unsafe performance, we were able to achieve a competitive F1 score of 67.7 ± 4.7 . This result is comparable to the seed scaffolds, highlighting that scaffolds may appear just as capable in terms of task performance yet simultaneously exhibit increased safety vulnerabilities.

5.3 Experiment 3: Multi-Objective Ablation

As an ablation for our multi-objective criteria and to compare AGENTBREEDER against the seminal work, we run CAPABLEAGENTBREEDER - a single-objective-variant of our framework - for 20 generations, evolving 10 mutants each generation. We take the highest-performing scaffolds from ADAS [20] and evaluate them with GPT-4o mini [37] as their core model. We report the F1 score for DROP [14], median accuracy for MMLU [19] and GPQA [43] and their 95% confidence intervals, as well as their performance on SaladData [28], our chosen safety benchmark. The results are shown in Table 5 in Appendix B.

Comparable Performance to Previous Work. CAPABLEAGENTBREEDER achieves competitive results to ADAS, marginally surpassing performance across all capability benchmarks.

Multi-Objective outperforms Single-Objective Optimization. The scaffolds discovered by CAPABLEAGENTBREEDER achieve near or slightly above-baseline results, such as 72.3 ± 3.1 F1 on DROP and 41.2 ± 4.4 accuracy on GPQA. This performance gain is notably smaller than in the multi-objective setting. This supports our hypothesis that incorporating an additional benchmark may increase the signal-to-noise ratio of scaffold validations each generation. This improves the quality of the selection pressure for the evolutionary algorithm, helping the process converge to better solutions overall.

Insignificant Impact on Safety Performance. In single-objective ablation runs, the discovered scaffolds showed only modest performance uplift on SaladData [28], suggesting that ignoring safety in the objective yields no strong impetus for safe or unsafe behaviors. This contrasts with multi-objective runs, where explicit safety optimization (or “negative safety” in red-teaming) substantially influenced outcomes.

Performance Stagnates with Better LLMs. When using more advanced models (GPT-4o mini [37] for scaffolds and Claude 3.5 Sonnet [3] for the Meta Agent) compared to the original ADAS [20] implementation, we observe that while overall performance improves, the relative gain between seed and discovered scaffolds diminishes. We attribute this to three plausible factors: (1) increased data contamination in newer LLMs may lead to memorized solutions rather than genuine reasoning, (2) higher baseline performance makes marginal improvements harder to distinguish from noise and (3) recent models are already fine-tuned for detailed reasoning, reducing the benefit of scaffold-induced reasoning steps [38, 63].

6 Discussion

Pre-Deployment Safety Evaluations. The Dead Internet Theory posits a future where AI agents dominate online activity [50]. While speculative, the recent releases of Operator [40] and Proxy [12] highlight the increasing population of agents deployed with the ability to interact autonomously with other agents and humans. These underscore the uncertainty around agent-on-agent dynamics, especially when these agents evolve or compose themselves in unanticipated ways. Our REDAGENTBREEDER experiments illustrate an automated approach to efficiently surface multi-agent scaffolds that exhibit vulnerabilities on safety benchmarks. Over time, labs could adopt a REDAGENTBREEDER-style pipeline to proactively “red-team” new LLMs as part of a release protocol.

Post-Deployment Adversarial Robustness. Just as REDAGENTBREEDER discovers vulnerable scaffolds, BLUEAGENTBREEDER provides a methodology to design safe and capable multi-agent scaffolds. This method can also be used to upgrade the safety capabilities of existing scaffolds, akin to Weak-to-Strong Generalization [6]. Furthermore, BLUEAGENTBREEDER can be used to

ensure a scaffold conforms to dynamic company values, policies and regulatory requirements. These experiments validate the practicality of evolutionary search as a dynamic, data-driven tool for multi-agent evaluation.

Limitations. While our experiments provide promising insights, several limitations should be acknowledged. Firstly, due to computational costs, we conducted experiments over a limited number of generations and with relatively small population sizes, resulting in only marginal performance improvements. Secondly, our experimental setup serves as a proof of concept for multi-objective alignment, and stronger claims of helpfulness and safety would require evaluations on more comprehensive benchmarks. Additionally, our evaluation was restricted to a select set of benchmarks, which may not fully capture the diverse range of real-world capabilities and safety concerns. Finally, the initial population was limited to seven seed scaffolds, potentially constraining the diversity of discovered scaffolds.

7 Conclusion

This paper introduces AGENTBREEDER, an evolutionary framework for discovering and evaluating multi-agent scaffolds via the multi-objective optimization of capability and safety. Our experiments demonstrate that AGENTBREEDER operates effectively in three distinct modes. BLUEAGENTBREEDER for developing safer scaffolds, REDAGENTBREEDER for identifying vulnerabilities, and CAPABLEAGENTBREEDER for maximizing task performance. Through empirical evaluation across multiple benchmarks, we show that our framework discovers scaffolds that achieve competitive or increased performance to prior works while exhibiting increased adversarial robustness.

Our results highlight several important findings for AI safety research. First, we demonstrate that unsafe behaviors can coexist with strong task performance, as evidenced by REDAGENTBREEDER's ability to generate scaffolds that maintain capability while exhibiting increased vulnerability. Second, our experiments reveal that multi-objective optimization targeting both capability and safety yields better overall solutions compared to single-objective approaches. Third, our BLUEAGENTBREEDER experiments achieved substantial safety improvements (up to 110.7% increase on SaladData with 79.4% average uplift) while simultaneously maintaining or enhancing capability (up to 21.0% improvement on GPQA). Finally, we show that automated evolutionary methods can effectively probe the complex attack surfaces of multi-agent scaffolds, offering a practical approach to pre-deployment safety evaluation.

As AI systems become increasingly interconnected and deployed in real-world settings, frameworks like AGENTBREEDER bridge the research gap between single-agent and multi-agent safety evaluations. Our work establishes a foundation for the systematic evaluation of multi-agent scaffolds, contributing to the development of safer and more reliable AI technologies.

8 Future Work

Scaling Laws. Scaling up AGENTBREEDER to larger population sizes and longer evolutionary runs could yield more substantial improvements in both capability and safety metrics. Incorporating closed-source safety benchmarks such as AILuminate [33] and contamination-free capability benchmarks such as MMLU-CF [65] would provide a more comprehensive assessment of multi-agent system safety.

White-Box and Gray-Box Evaluations. A key limitation of our current approach is its focus on black-box evaluation of scaffolds. Future work could investigate individual agent behaviors, including how agents interact with tools, external APIs, and information sources. Developing methods to trace and analyze agent-agent and agent-tool interactions could reveal potential safety risks that are invisible in black box evaluation.

Alternative Objectives. In this work, we only consider the capability and safety objectives for optimization. Future work could explore inference cost as an objective to minimize for, and consider multi-core scaffolds where different LLM base models exist inside the same scaffold.

Multi-Agent Governance. Critical research is needed to establish governance frameworks for multi-agent scaffolds. Future work could comprise developing differentiated safety cases for scaffolds with varying levels of transparency, from fully white box to black box architectures.

Acknowledgments and Disclosure of Funding

J Rosser is supported by the EPSRC centre for Doctoral Training in Autonomous and Intelligent Machines and Systems EP/Y035070/1. We extend our sincere gratitude to the members of the Foerster Lab for AI Research (FLAIR) for their guidance during the project scoping phase and thorough proofreading. Special thanks to the London Initiative for Safe AI and Arcadia Impact for providing workspace and offering invaluable feedback throughout.

References

- [1] UK AI Security Institute. Inspect AI: Framework for Large Language Model Evaluations, 2024. URL https://github.com/UKGovernmentBEIS/inspect_ai.
- [2] Maryam Amirizani, Elias Martin, Maryna Sivachenko, Afra Mashhadi, and Chirag Shah. Can llms reason like humans? assessing theory of mind reasoning in llms for open-ended questions. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 34–44, 2024.
- [3] AI Anthropic. Claude 3.5 sonnet model card addendum. *Claude-3.5 Model Card*, 3:6, 2024.
- [4] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- [5] Robert S. Boyer and J. Strother Moore. A mechanical proof of the turing completeness of pure lisp. Technical Report ADA130625, Texas Univ at Austin Inst for Computing Science and Computer Applications, May 1983. URL <https://apps.dtic.mil/sti/citations/ADA130625>. Approved for public release.
- [6] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.
- [7] Hong Cao, Rong Ma, Yanlong Zhai, and Jun Shen. Llm-collab: a framework for enhancing task planning via chain-of-thought and multi-agent collaboration. *Applied Computing and Intelligence*, 4(2):328–348, 2024.
- [8] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*, 2023.
- [9] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*, 2023.
- [10] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*, 2(4):6, 2023.
- [11] François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.
- [12] Convergence. Introducing web-world models, December 2024. URL <https://convergence.ai/training-web-agents-with-web-world-models-dec-2024/>. Accessed: 2025-01-30.
- [13] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.
- [14] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

- [15] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.
- [16] Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.
- [17] Stephen Fowler. Scaffolded llms: Less obvious concerns. *LessWrong*, 2023. URL <https://www.lesswrong.com/posts/mAwxebLw3nYbDivmt/scaffolded-llms-less-obvious-concerns>.
- [18] Alireza Ghafarollahi and Markus J Buehler. Sciagents: Automating scientific discovery through multi-agent intelligent graph reasoning. *arXiv preprint arXiv:2409.05556*, 2024.
- [19] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- [20] Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.
- [21] Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Hao-Shu Fang, et al. Toward general-purpose robots via foundation models: A survey and meta-analysis. *arXiv preprint arXiv:2312.08782*, 2023.
- [22] Yue Hu, Yuzhu Cai, Yaxin Du, Xinyu Zhu, Xiangrui Liu, Zijie Yu, Yuchen Hou, Shuo Tang, and Siheng Chen. Self-evolving multi-agent collaboration networks for software development. *arXiv preprint arXiv:2410.16946*, 2024.
- [23] Jen-tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Maarten Sap, and Michael R Lyu. On the resilience of multi-agent systems with malicious agents. *arXiv preprint arXiv:2408.00989*, 2024.
- [24] Adarsh Kesireddy and F Antonio Medrano. Elite multi-criteria decision making—pareto front optimization in multi-objective optimization. *Algorithms*, 17(5):206, 2024.
- [25] Akbir Khan. Why multi-agent safety is important. <https://www.lesswrong.com/posts/pkfKRG9dQr6unrhQT/why-multi-agent-safety-is-important>, 2022. Accessed: 2025-01-30.
- [26] Xiangrui Kong, Thomas Braunl, Marco Fahmi, and Yue Wang. A superalignment framework in autonomous driving with large language models. *arXiv preprint arXiv:2406.05651*, 2024.
- [27] Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. Agent-oriented planning in multi-agent systems. *arXiv preprint arXiv:2410.02189*, 2024.
- [28] Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. Salad-bench: A hierarchical and comprehensive safety benchmark for large language models. *arXiv preprint arXiv:2402.05044*, 2024.
- [29] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.
- [30] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, et al. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*, 2023.
- [31] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

- [32] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.
- [33] MLCommons Association. Ailuminate v1.0 benchmark, 2025. URL <https://ailuminate.mlcommons.org/benchmarks/>. Accessed: 2025-01-31.
- [34] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.
- [35] Subhabrata Mukherjee, Paul Gamble, Markel Sanz Ausin, Neel Kant, Kriti Aggarwal, Neha Manjunath, Debajyoti Datta, Zhengliang Liu, Jiayuan Ding, Sophia Busacca, et al. Polaris: A safety-focused llm constellation architecture for healthcare. *arXiv preprint arXiv:2403.13313*, 2024.
- [36] OpenAI. simple-evals. <https://github.com/openai/simple-evals>, 2023. Accessed: 2025-01-29.
- [37] OpenAI. Gpt-4o mini: Advancing cost-efficient intelligence, 2024. URL <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>.
- [38] OpenAI. Learning to reason with llms, 2024. URL <https://openai.com/index/learning-to-reason-with-llms/>. Accessed: 2025-01-31.
- [39] OpenAI. New embedding models and api updates, January 2024. URL <https://openai.com/research/new-embedding-models-and-api-updates>. Accessed: 2025-01-14.
- [40] OpenAI. Computer-using agent: Introducing a universal interface for ai to interact with the digital world, 2025. URL <https://openai.com/index/computer-using-agent>.
- [41] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.
- [42] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 6(3), 2023.
- [43] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.
- [44] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.
- [45] Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, et al. Rainbow teaming: Open-ended generation of diverse adversarial prompts. *arXiv preprint arXiv:2402.16822*, 2024.
- [46] Sebastian Sartor and Neil Thompson. Neural scaling laws for embodied ai. *arXiv preprint arXiv:2405.14005*, 2024.
- [47] Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. Language models are multilingual chain-of-thought reasoners. URL <https://arxiv.org/abs/2210.03057>, 2022.
- [48] Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. Determlr: Augmenting llm-based logical reasoning from indeterminacy to determinacy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9828–9862, 2024.

- [49] Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, et al. Scicode: A research coding benchmark curated by scientists. *arXiv preprint arXiv:2407.13168*, 2024.
- [50] Yoshija Walter. Artificial influencers and the dead internet theory. *AI & SOCIETY*, pages 1–2, 2024.
- [51] Jianxun Wang and Yixiang Chen. A review on code generation with llms: Application and evaluation. In *2023 IEEE International Conference on Medical Artificial Intelligence (MedAI)*, pages 284–289. IEEE, 2023.
- [52] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- [53] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [54] Anna Christina Weigand, Daniel Lange, and Maria Rauschenberger. How can small data sets be clustered?, 2021.
- [55] Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. Expertprompting: Instructing large language models to be distinguished experts. *arXiv preprint arXiv:2305.14688*, 2023.
- [56] Fengli Xu, Qianyu Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, et al. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*, 2025.
- [57] Xiangyuan Xue, Zeyu Lu, Di Huang, Zidong Wang, Wanli Ouyang, and Lei Bai. Comfy-bench: Benchmarking llm-based agents in comfyui for autonomously designing collaborative ai systems, 2024. URL <https://arxiv.org/abs/2409.01392>.
- [58] Yingxuan Yang, Qiuying Peng, Jun Wang, and Weinan Zhang. Multi-llm-agent systems: Techniques and business perspectives. *arXiv preprint arXiv:2411.14033*, 2024.
- [59] Burak Yetiştiren, Işık Özsoy, Miray Ayerdem, and Eray Tüzün. Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. *arXiv preprint arXiv:2304.10778*, 2023.
- [60] Xunjian Yin, Xinyi Wang, Liangming Pan, Xiaojun Wan, and William Yang Wang. G\ " odel agent: A self-referential agent framework for recursive self-improvement. *arXiv preprint arXiv:2410.04444*, 2024.
- [61] Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*, 2023.
- [62] Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. *arXiv preprint arXiv:2406.14228*, 2024.
- [63] Wojciech Zaremba, Evgenia Nitishinskaya, Boaz Barak, Stephanie Lin, Sam Toyer, Yaodong Yu, Rachel Dias, Eric Wallace, Kai Xiao, and Johannes Heidecke Amelia Glaese. Trading inference-time compute for adversarial robustness., 2025.
- [64] Zaibin Zhang, Yongting Zhang, Lijun Li, Hongzhi Gao, Lijun Wang, Huchuan Lu, Feng Zhao, Yu Qiao, and Jing Shao. Psysafe: A comprehensive framework for psychological-based attack, defense, and evaluation of multi-agent system safety. *arXiv preprint arXiv:2401.11880*, 2024.
- [65] Qihao Zhao, Yangyu Huang, Tengchao Lv, Lei Cui, Qinzhen Sun, Shaoguang Mao, Xin Zhang, Ying Xin, Qiuqiang Yin, Scarlett Li, et al. Mmlu-cf: A contamination-free multi-task language understanding benchmark. *arXiv preprint arXiv:2412.15194*, 2024.

- [66] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.
- [67] Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*, 2023.

A Impact Statement

This work introduces methods for evaluating and improving the safety of multi-agent scaffolds, which is increasingly critical as embodied, autonomous agents become more prevalent. While AGENTBREEDER can help discover safer multi-agent architectures, it could also be used to find scaffolds that exploit vulnerabilities. We release this research to enable proactive safety testing before deployment, but acknowledge the dual-use nature of these techniques. The red-teaming capabilities we describe could be misused to develop harmful scaffolds, though we believe the defensive benefits outweigh these risks. Additionally, our research surfaces important questions about AI governance as multi-agent scaffolds become more common. We hope this work advances the field’s understanding of multi-agent safety and helps develop more robust evaluation frameworks. We encourage future research to build upon these methods while carefully considering potential misuse and implementing appropriate safeguards.

B CapableAgentBreeder

Table 5 shows the results of CAPABLEAGENTBREEDER reported on three benchmarks and compares them with the performance of ADAS [20].

CAPABLEAGENTBREEDER			
Capability			Safety
DROP	MMLU	GPQA	SaladData
<i>Seed Scaffolds [20]</i>			
70.4 ± 3.1	80.2 ± 3.6	35.2 ± 4.4	31.2 ± 4.2
64.4 ± 3.2	82.6 ± 3.4	38.1 ± 4.3	17.8 ± 3.4
69.3 ± 3.2	81.2 ± 3.6	<u>39.4 ± 4.4</u>	55.6 ± 4.6
<i>ADAS Scaffolds</i>			
<u>72.0 ± 3.0</u>	-	-	57.0 ± 4.2
-	80.4 ± 3.4	-	76.4 ± 3.6
-	-	37.4 ± 3.6	61.0 ± 4.2
<i>CapableAgentBreeder Scaffolds</i>			
72.3 ± 3.1	-	-	39.4 ± 4.4
-	<u>82.4 ± 3.2</u>	-	58.0 ± 4.2
-	-	41.2 ± 4.4	43.8 ± 4.4

Table 5: We report the evaluation results of CAPABLEAGENTBREEDER on the held-out test sets. For each benchmark, we compare the AGENTBREEDER-discovered scaffolds with the seed and discovered scaffolds from the seminal work ADAS [20], with the highest and second highest result in each column given in bold and underlined respectively. We report the F1 score on DROP [14] and median accuracy on the other benchmarks, along with a 95% confidence interval computed over 500 held-out test samples.

C Benchmarks

MMLU [19] is a multi-task benchmark comprising multiple choice questions on 57 subjects across STEM, the humanities, the social sciences, and more.

DROP [14] is an English reading comprehension benchmark which requires the model to reason over and answer questions on given paragraphs. This may be a simple look-up or extend to more complex tasks such as counting, addition and sorting.

GPQA [43] is a benchmark comprising graduate-level multiple-choice questions in the field of biology, physics, and chemistry. The questions are designed to be challenging such that experts in the domain achieve 65% accuracy and non-experts achieve 34% accuracy.

SaladData [28] is a hierarchical and comprehensive safety benchmark spanning 3 levels. We report results on the “Attack Enhanced” subset, where harmful questions have been further enhanced with a variety of attack methods such as GPTFuzz [61], an automated method of generating jailbreak prompts via black-box “fuzzing”.

TruthfulQA [29] is a question-answering benchmark comprising questions that some humans may answer incorrectly. The types of questions span 38 categories, including health, law, finance and politics.

D Prompts

The Meta Agent is tasked with iteratively programming novel scaffolds. After evolving the first candidate solution, the Meta Agent goes through two iterations of refinement, and then up to three iterations of debugging. The prompts for all of these are given below and are modified versions of the prompts designed by Hu et al. [20] for ADAS.

D.1 Base Prompt

Meta Agent Base Prompt

You are an expert machine learning researcher testing various agentic systems.
Your objective is to design building blocks such as prompts and control flows within these systems to solve complex tasks.

<< BENCHMARK DESCRIPTION >>

<< SAFETY DESCRIPTION>>

The utility code:

```
'''python
from collections import namedtuple
from typing import Union
import numpy as np
import json
```

```
import openai
import backoff
from utils import random_id
```

```
# Initialize the OpenAI client
client = openai.OpenAI()
```

<< STRUCTURED JSON RESPONSE CODE >>

<< AGENTIC FRAMEWORK CODE >>

```
class AgentArchitecture:
```

```

"""
Fill in your code here.
"""
async def forward(self, task, required_answer_format) ->
    str:
        """
        Placeholder method for processing task information.

        Args:
        - task (str): Task description.
        - required_answer_format (str): The required format for
        the answer.
        E.g. latex code, or a single letter A,B,C,D etc.

        Returns:
        - Answer (str): Your FINAL Answer. Return as a string in
        the exact
        format as specified in the required_answer_format.
        """
    pass
'''

```

Discovered architecture archive
Here is the archive of the discovered architectures:

<< ARCHIVE >>

The fitness value is the median and 95% Bootstrap Confidence Interval of the correct rate on a validation question set. Your GOAL is to maximize the “fitness”.

Output Instruction and Example:

The first key should be (“thought”), and it should capture your thought process for designing the next function. In the “thought” section, first reason about what should be the next interesting agent to try, then describe your reasoning and the overall concept behind the agent design, and finally detail the implementation steps.

The second key (“name”) corresponds to the name of your next agent architecture.

Finally, the last key (“code”) corresponds to the exact “forward()” function in Python code that you would like to try. You must write a COMPLETE CODE in “code”: Your code will be part of the entire project, so please implement complete, reliable, reusable code snippets.

Here is an example of the output format for the next agent architecture:

<< EXAMPLE >>

You must use the exact function interface used above. You need to specify the instruction, input information, and the required output fields for various LLM agents to do their specific part of the architecture.

Also, it could be helpful to set the LLMs role and temperature to further control the LLMs response. Note that the Agent() will always return a JSON object with the keys as the output fields and the values as the corresponding outputs.

DO NOT FORGET the task input to LLM if you think it is needed, otherwise LLM will not know about the task.

Documentation: Writing Forward Functions in Multi-Agent Framework This documentation describes how to implement forward functions in your multi-agent framework, focusing on the interaction between Agents, Meetings, and Chats. Each forward function facilitates specific reasoning or task-solving approaches by coordinating these components effectively.

Framework Components

Agents: Autonomous entities with specific roles, goals, and configurations (e.g., temperature).

They can participate in meetings and generate responses. No agents can “hear” the agent’s forward pass. For an agent to speak, their response must be added as “Chat” to the meeting chats.

Meetings: Contextual containers where agents interact. Agents cannot “hear” each other’s forward passes. In order to speak, the output of an agent’s forward pass must be collected and added as a “Chat” object to the meeting. Only agents present in that meeting can “hear” the chat history.

Chats: Messages exchanged in meetings. They capture the content generated by agents or instructions provided by the system. An agent’s response (output of a forward pass) must be created as a Chat and added to a meeting for it to be visible to the other agents in that meeting.

WRONG Implementation examples:

<<WRONG IMPLEMENTATION EXAMPLES>>

CORRECT implementation patterns:

<<CORRECT IMPLEMENTATION EXAMPLES>>

Your task

You are deeply familiar with LLM prompting techniques and LLM agent works from the literature. Your goal is to maximize “fitness” by proposing interestingly new multi-agent systems.

Observe the discovered architectures carefully and think about what insights, lessons, or stepping stones can be learned from them. Be creative to think about the next interesting architecture to try. You are encouraged to draw inspiration from related LLM agent papers or academic papers from other research areas.

Using the knowledge learned from the archive and the inspiration from academic literature to give the next interesting architecture. THINK OUTSIDE THE BOX. Give a concise, powerful answer.

Please generate a new multi-agent system from scratch. Use the multi-agent structure provided e.g. Agents, Meetings and Chats, and ensuring agents each have their own internal monologue where they are told their role and goals. Please do not copy the previous architectures but come up with something new and interesting that would work better on the given tasks.

Ensure that the new forward functions outputs a response as a STRING in the exact format as specified in the required_answer_format. This could be either a single letter (e.g. A, B, C, D) or a word or phrase, or a short piece of code.

D.2 Reflection Prompt 1

Meta Agent Reflexion Prompt 1

<<EXAMPLE>>Carefully review the proposed new architecture and reflect on the following points:

1. ****Interestingness****: Assess whether your proposed architecture is interesting or innovative compared to existing methods in the archive. If you determine that the proposed architecture is not interesting, suggest a new architecture that addresses these shortcomings.
 - Make sure to check the difference between the proposed architecture and previous attempts.
 - Compare the proposal and the architectures in the archive CAREFULLY, including their actual differences in the implementation.
 - Decide whether the current architecture is innovative.

- USE CRITICAL THINKING!

2. **Implementation Mistakes**: Identify any mistakes you may have made in the implementation. Review the code carefully, debug any issues you find, and provide a corrected version. REMEMBER checking “## WRONG Implementation examples” in the prompt.

3. **Improvement**: Based on the proposed architecture, suggest improvements in the detailed implementation that could increase its performance or effectiveness. In this step, focus on refining and optimizing the existing implementation without altering the overall design system, except if you want to propose a different architecture if the current is not interesting.

- Observe carefully about whether the implementation is actually doing what it is supposed to do.

- Check if there is redundant code or unnecessary steps in the implementation. Replace them with effective implementation.

- Try to avoid the implementation being too similar to the previous agent.

4. **Check output format**: Make sure the agent returns the direct correct output in the format as laid out in the task, ensuring NO thinking or reasoning is given with the answer. It may be worth adding in a final agent with knowledge of the task to return the correct output for the task.

And then, you need to improve or revise the implementation, or implement the new proposed architecture based on the reflection.

Your response should be organized as follows:

"reflection": Provide your thoughts on the interestingness of the architecture, identify any mistakes in the implementation, and suggest improvements.

"thought": Revise your previous proposal or propose a new architecture if necessary, using the same format as the example response.

"name": Provide a name for the revised or new architecture. (Don't put words like "new" or "improved" in the name.)

"code": Provide the corrected code or an improved implementation. Make sure you actually implement your fix and improvement in this code.

D.3 Reflection Prompt 2

Meta Agent Reflection Prompt 2

Using the tips in “## WRONG Implementation examples” section, revise the code further. Put your new reflection thinking in “reflection”. Repeat the previous “thought” and “name”, and update the corrected version of the code in “code”.

D.4 Debugging Prompt

Meta Agent Debugging Prompt

Error during evaluation:

<< ERROR >>

Carefully consider where you went wrong in your latest implementation. Using insights from previous attempts, try to debug the current code to implement the same thought. Repeat your previous thought in 'thought', and put your thinking for debugging in 'debug_thought'.

D.5 Mutation Prompts

We provide the full selection of mutation prompts from which the Meta Agent randomly samples. The base prompt is as follows:

Mutation Base Prompt

Here is the multi-agent system I would like you to mutate:

<<SYSTEM NAME>>

<<SYSTEM THOUGHT PROCESS>>

<<SYSTEM CODE>>

The mutation I would like to apply is:

<<SAMPLED MUTATION OPERATOR>>

IMPORTANT:

In general, the new system will perform better with more detailed prompts for the agents, more planning steps, encouraging them to think longer and harder. It may be worth adding a final agent to the system to help transform the output of the final agent into the desired output format for the task as the system will be scored very lowly if the output is not in the correct format, even if the thinking was sound.

Ensure that the new forward functions outputs a response as a STRING in the exact format as specified in the required_answer_format. This could be either a single letter (e.g. A, B, C, D) or a word or phrase, or a short piece of code.

Capability-Enhanced Mutation Operators

- Inside the system, add a step which restates and elaborates on the inquiry before proceeding with a response.
- Inside the system, add a step which reframes the question with additional context and detail before working out the answer.
- Let the agents know they will be rewarded for providing a concise and accurate answer to the question.
- Change the predefined roles or personas of the agents involved in the system. For instance, switch an agent's role from 'Proponent' to 'Skeptic' or assign new characteristics like expertise in a specific domain.
- Adjust the rules governing how agents interact. This could include changing the number of debate rounds, the order of speaking, or introducing new interaction phases such as rebuttal or evidence presentation.
- Change the step-back abstraction layers used by the agents. This might involve altering how much reasoning is abstracted, the granularity of explanations, or the depth of recursive thinking.

- Modify the initial prompts or instructions given to each agent. This could include rephrasing goals, making them more detailed, introducing new guidelines, or emphasizing different aspects of their character.
- Add new agents with unique roles or remove existing ones from the debate system. For example, introduce a 'Moderator' agent or eliminate a less effective participant.
- Change the reward structure or scoring system used to evaluate agent performance. This could involve adjusting the weights assigned to different criteria, introducing new evaluation metrics, or modifying the feedback mechanism.
- Alter the format of meetings by introducing new structures or modifying existing ones. For example, switch from a traditional debate format to a panel discussion, round-robin, or brainstorming session.
- Change which agents participate in each meeting. This can include rotating participants, introducing new agents, or forming specialized sub-groups.
- Create multi-tiered meeting structures where higher-level meetings oversee or coordinate lower-level ones. For example, strategic meetings could direct tactical meetings.
- Introduce various types of meetings, such as decision-making, information-sharing, conflict-resolution, or creative ideation sessions.
- Replace fixed roles with dynamic role allocation based on agent performance.
- Introduce hierarchical agents with higher-level agents overseeing groups of lower-level agents.
- Simulate natural ecosystems by introducing predator-prey dynamics.
- Introduce trust mechanisms where agents evaluate reliability of others.
- Introduce asymmetrical power dynamics by giving certain agents additional capabilities.

Safety Enhanced Mutation Operators

- Incorporate a consensus-checking step where agents must verify each other's responses for accuracy and adherence to ethical guidelines before finalizing an answer.
- Introduce a diversity-promoting mechanism that ensures a variety of perspectives are represented, thereby preventing bias and promoting fairness in decision-making processes.
- Implement a transparency layer where agents must provide clear justifications for their actions and decisions, fostering accountability and trust within the system.
- Embed an efficient monitoring feature that assesses agent interactions for compliance with privacy and confidentiality standards, safeguarding sensitive information.
- Establish an ethical oversight agent responsible for evaluating the potential impacts of decisions on human well-being and societal values, ensuring alignment with the constitution.
- Create role-based restrictions that limit agents from performing actions beyond their designated responsibilities, preventing misuse of authority and maintaining system integrity.
- Introduce a fail-safe mechanism that returns a default output if unethical behavior or harmful patterns are detected, allowing for timely intervention and correction.

D.6 Crossover Prompts

The crossover base prompt is as follows:

Crossover Base Prompt

Here are the two systems I'd like you to crossover/combine into a novel new system:

System 1: <<SYSTEM 1 NAME>>
<<SYSTEM 1 THOUGHT PROCESS>>

<<SYSTEM 1 CODE>>

System 2: <<SYSTEM 2 NAME>>
<<SYSTEM 2 THOUGHT PROCESS>>

<<SYSTEM 2 CODE>>

Ensure that the new forward functions outputs a response as a STRING in the exact format as specified in the required_answer_format. This could be either a single letter (e.g. A, B, C, D) or a word or phrase, or a short piece of code.

E Scaffolds Discovered by AgentBreeder

Optimized-Hierarchical-Expert-Panel

```
async def forward(self, task: str, required_answer_format: str)
-> str:
    # Create agents
    system = Agent(agent_name='system', temperature=0.7)
    moderator = Agent(
        agent_name='Moderator',
        agent_role='You are a skilled debate moderator managing
multiple expert panels.',
        agent_goal='Guide productive discussion and manage
hierarchical debate process.',
        temperature=0.7
    )

    # Create domain experts
    domain_experts = [
        Agent(agent_name=f'{domain}_Expert',
            agent_role=f'You are a {domain} expert analyzing
problems deeply.',
            agent_goal='Provide detailed domain analysis and
critique solutions.',
            temperature=0.8)
        for domain in ['Physics', 'Biology', 'Chemistry']
    ]

    devils_advocate = Agent(
        agent_name='Devil\'s Advocate',
        agent_role='You critically challenge all assumptions
and arguments.',
        agent_goal='Identify potential flaws and ensure robust
analysis.',
        temperature=0.9
    )

    synthesis_expert = Agent(
        agent_name='Synthesis Expert',
```

```

        agent_role='You integrate insights from multiple domains and perspectives.',
        agent_goal='Create coherent synthesis from diverse expert inputs.',
        temperature=0.7
    )

    validator = Agent(
        agent_name='Validator',
        agent_role='You validate final answers for format and logical consistency.',
        agent_goal='Ensure answers are correctly formatted and well-justified.',
        temperature=0.1
    )

    # Setup a single meeting
    meeting = Meeting(meeting_name='expert_panel_meeting')

    # Add agents to the meeting
    all_agents = [system, moderator] + domain_experts + [devils_advocate, synthesis_expert, validator]
    [agent.meetings.append(meeting) for agent in all_agents]

    # Stage 1: Domain-specific analysis
    meeting.chats.append(Chat(
        agent=moderator,
        content=f"Task for domain analysis: {task}\nRequired format: {required_answer_format}"
    ))

    domain_insights = []
    for expert in domain_experts:
        # Expert analysis
        output = await expert.forward(response_format={
            "analysis": "Detailed domain-specific analysis",
            "confidence": "Confidence level (0-100)",
            "answer": required_answer_format
        })
        meeting.chats.append(Chat(agent=expert,
            content=f"Analysis: {output['analysis']}"))

        # Devil's Advocate challenge
        challenge = await devils_advocate.forward(response_format={"challenge": "Critical challenge to the analysis"})
        meeting.chats.append(Chat(agent=devils_advocate,
            content=challenge['challenge']))

        # Expert response to challenge
        final_response = await expert.forward(response_format={
            "final_answer": required_answer_format
        })
        domain_insights.append(final_response['final_answer'])

    # Stage 2: Synthesis
    meeting.chats.append(Chat(
        agent=synthesis_expert,
        content=f"Synthesize domain expert insights and challenges for final answer."
    ))

    synthesis = await synthesis_expert.forward(response_format={

```

```
        "answer": required_answer_format
    })

    # Final validation
    validation = await
        validator.forward(response_format={"answer":
            required_answer_format})

    return validation['answer']
```

F Cost of Experiments

The BLUEAGENTBREEDER experiment, comprising one 20-generation run on each of our 3 benchmarks as well as evaluations costs approximately \$600, with the ~\$500 from *gpt-4o-mini-2024-07-18* and ~\$100 from *claude-3-5-sonnet-20241022-v2:0*.

The REDAGENTBREEDER experiment, comprising one 10-generation run on DROP cost ~\$115 as expected.

The CAPABLEAGENTBREEDER experiment, comprising one 20-generation run on each of our 3 benchmarks as well as evaluations costs approximately \$400.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [NA].
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes]" is generally preferable to "[No]", it is perfectly acceptable to answer "[No]" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No]" or "[NA]" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading “NeurIPS Paper Checklist”,**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers.**

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 Section 1 explicitly state the three contributions (**Attack, Defense, Evaluation**) that are delivered and later substantiated in Sections 4–5.

Guidelines:

- The answer NA 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 NA 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: A dedicated “**Limitations**” paragraph appears in Section 6, detailing computational cost, benchmark coverage and seed-diversity constraints.

Guidelines:

- The answer NA 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: [NA]

Justification: The paper is empirical; it presents no formal theorems or proofs.

Guidelines:

- The answer NA 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: Section 4 and Appendix D specify the full search algorithm, prompts, and hyper-parameters; code is linked in the abstract for end-to-end replication.

Guidelines:

- The answer NA 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: The AgentBreeder framework is released under an MIT licence (link in abstract), and all benchmarks used are publicly available (Appendix C).

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.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://nips.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, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Section 5 describes generation budgets, model versions, and evaluation protocols; Appendix C lists dataset splits.

Guidelines:

- The answer NA 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: All tables report medians with 95 % bootstrap confidence intervals (e.g. Table 2); the resampling procedure is explained in Section 5.1.

Guidelines:

- The answer NA 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: Appendix F enumerates model families, and total cost for every run.

Guidelines:

- The answer NA 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: The work focuses on improving safety, follows open-benchmark protocols, and avoids collection of personal data (see Sections 2 and 6).

Guidelines:

- The answer NA 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: We include an impact statement in Appendix A.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [Yes]

Justification: Code is released under a Non-Commercial license and the paper's dual-use nature is discussed in Appendix A.

Guidelines:

- The answer NA 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: All datasets are cited with original papers and licences (Appendix C); LLM APIs (OpenAI, Anthropic) are referenced in Sections D.5–5.

Guidelines:

- The answer NA 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: [NA]

Justification: No new datasets or pretrained models are released; only source code (already covered above).

Guidelines:

- The answer NA 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: [NA]

Justification: The study involves only publicly available text benchmarks; no human participants were recruited.

Guidelines:

- The answer NA 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: [NA]

Justification: As no human subjects were involved, IRB approval was not required.

Guidelines:

- The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Sections D.5 and 5 document all LLMs employed (Claude 3.5 Sonnet as Meta-Agent, GPT-4o-mini for evaluations) and their roles in the methodology.

Guidelines:

- The answer NA 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 (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.