scAgents: A Multi-Agent Framework for Fully Autonomous End-to-End Single-Cell Perturbation Analysis

Anonymous Authors[1]

Abstract

While AI agents for scientific discovery have proliferated, truly end-to-end autonomous solutions remain scarce, particularly in complex interdisciplinary fields like single-cell genomics. We introduce scAgents, a fully autonomous multi-agent framework that transforms raw single-cell data and task descriptions directly into optimized computational solutions. Given only a dataset and research objective as input, scAgents outputs both a novel model architecture and executable code for training and inference without human intervention. When evaluated on the scPerturb datasets and benchmarks, scAgents consistently outperforms task-specific state-of-the-art methods, achieving up to 49% reduction in prediction error compared to scGPT for gene knockouts and Pearson correlation increases of up to 20% in expression predictions versus ChemCPA for drug perturbations. scAgents' ability to succeed where existing foundation models struggle is particularly significant, adapting effectively to different data types (scRNA-seq, scATAC-seq, CITE-seq) and various perturbation categories with consistent performance across modalities. Our code and some scAgents-designed models are available at https://anonymous.4open.science/r/scAgents-2025-242E/.

1. Introduction

Scientific discovery is undergoing a transformation with the rise of *AI Scientists*autonomous systems designed to conduct research with minimal human intervention. Recent progress in large language models (LLMs) and AI agents has enabled impressive capabilities in literature analysis (Hsu et al., 2024), hypothesis generation (Qi et al., 2023; Yang et al., 2023), and experimental implementation (Tian et al., 2024; Huang et al., 2024b), as demonstrated by emerging benchmarks and agentic toolkits (Skarlinski et al., 2024; Chen et al., 2024; Majumder et al., 2024; Huang et al., 2024a). Yet these systems **remain siloedcompetent at individual tasks, but incapable of orchestrating complete scientific workflows**. Existing AI scientists lack the ability to *propose methods*, refine them through collaborative reasoning, and *carry out empirical validation* in a unified framework (Lu et al., 2024; Li et al., 2024b). This gap becomes especially critical in domains that demand interdisciplinary reasoning, such as computational biology, where effective modeling requires integrating biological priors, statistical rigor, and machine learning design (Boiko et al., 2023; Roohani et al., 2024b).

A particularly compelling instantiation of this challenge is *single-cell perturbation analysis*, a frontier task in computational biology that demands predicting how cells respond to diverse interventionssuch as gene knockouts, drug treatments, or cytokine stimulationby modeling state transitions in high-dimensional expression space (Figure 1). This problem exemplifies the limitations of current AI scientists: Despite progress in foundation models (Cui et al., 2024; Theodoris et al., 2023), existing approaches remain heavily fragmented, relying on human experts to bridge gaps in dataset interpretation, model design, and empirical validation (Lotfollahi et al., 2023; Hao et al., 2024). Moreover, these models often fail to generalize to novel biological scenarios, such as unseen cell types or experimental modalities (Levine et al., 2024; Wenteler et al., 2024). The challenge is further exacerbated by the heterogeneity of data types (e.g., scRNA-seq, scATAC-seq, CITE-seq) and perturbation mechanisms, each requiring domain-specific reasoning to extract meaningful insights (Peidli et al., 2024; Bendidi et al., 2024).

We present scAgents, a fully autonomous multi-agent framework that conducts end-to-end single-cell perturbation analysis by integrating three core modules: (1) **Task Analysis**,

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

which characterizes the dataset and retrieves relevant prior work via a hybrid breadth-first and depth-first agentic retrieval strategy; (2) **Method Design**, where role-specific agents collaboratively analyze baselines, identify design limitations, and iteratively refine novel modeling strategies through graph-based discussions; and (3) **Experiment Execution**, which translates high-level architectural proposals into executable code, complete with training routines and validation protocols. At the heart of scAgents is a persistent knowledge graph that accumulates insights across all phases and ensures a coherent information flow. The agentic retrieval system integrates perturbation-specific biological knowledge, such as regulatory pathway relationships and modality-aware considerations, into model design without requiring human input. The multi-agent discussion module leverages confidence-based scoring and self-critique mechanisms to converge on scientifically grounded and technically feasible solutions. The final system output includes both the model architecture and the complete implementation code for training and inference.

We evaluate scAgents on six diverse perturbation datasets from the scPerturb on single cell perturbation benchmark, encompassing gene knockouts, drug treatments, and cytokine stimulations across modalities such as scRNA-seq, scATAC-seq, and CITE-seq. In all cases, models designed by scAgents significantly outperform prior baselines. For instance, in the drug perturbation task, scAgents improves Pearson correlation by **20%** over the next best method, ChemCPA (Hetzel et al., 2022). On the challenging, sparse, and high-dimensional scATAC-seq dataset, it achieves a **∼16-fold increase** compared to the second-best baseline, a linear regression model. In surface protein prediction (CITE-seq), scAgents boosts Pearson correlation by **∼177%** over the next best baseline method, Random Forest. These results highlight scAgents' strong generalization and substantial performance gains across diverse settings, including the novel application scenarios.

2. Related Work

Agent Systems for Scientific Discovery Researchers have developed specialized AI systems spanning the entire research workflow: from literature analysis tools like PaperQA2 (Skarlinski et al., 2024) and CHIME (Hsu et al., 2024), to hypothesis generation frameworks that range from domain-specific idea creation (Baek et al., 2024; Qi et al., 2023; Yang et al., 2023) to comparative evaluations with expert proposals (Si et al., 2024). These systems increasingly leverage multi-agent architectures (Ghafarollahi & Buehler, 2025; Schmidgall & Moor, 2025; Guo et al., 2024) to facilitate collaborative scientific reasoning. Implementation capabilities have advanced through scientific coding frameworks like SciCode (Tian et al., 2024) and MLAgentBench

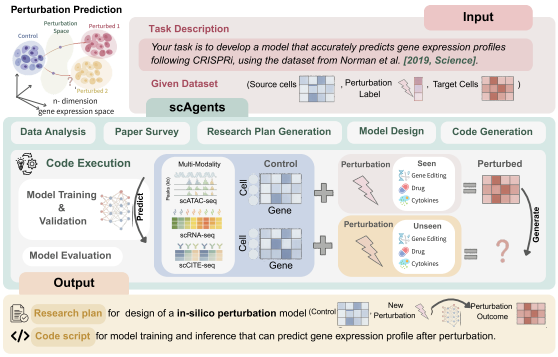

Figure 1: **Our framework predicts cellular responses to unseen interventions by modeling state transitions in high-dimensional expression space.** Given multi-modal single-cell data (scRNA-seq, scATAC-seq, CITE-seq) and a training set of known perturbations (gene knockouts, drugs, cytokines), scAgents learns to map control states to perturbed states, enabling accurate prediction of gene expression profiles under novel experimental conditions without additional wet-lab experimentation.

(Huang et al., 2024a), while benchmarks evaluate these capabilities across diverse domains (Qiu et al., 2025; Ruan et al., 2024; Chen et al., 2025; Jing et al., 2024). The integration of literature analysis with data-driven approaches has proven particularly effective for hypothesis generation (Liu et al., 2024b; Zhong et al., 2023; Majumder et al., 2024), with several frameworks enhancing research ideation through structured feedback mechanisms (Pu et al., 2024; Garikaparthi et al., 2025) and approaches to improve novelty and diversity (Hu et al., 2024; Radensky et al., 2024; Gao et al., 2025). End-to-end systems now attempt to unify these capabilities, including domain-general approaches like AI Scientist (Lu et al., 2024) and MLR-Copilot (Li et al., 2024b), alongside domain-specific implementations for chemistry (Boiko et al., 2023), genomics (Roohani et al., 2024b), materials science (Ghafarollahi & Buehler, 2024), and medicine (Tang et al., 2024; Naumov et al., 2025). Despite these advances, significant challenges remain in developing truly autonomous scientific systems, particularly regarding experimental rigor (Kon et al., 2025), falsification mechanisms (Liu et al., 2024c), and comprehensive evaluation metrics (Beel et al., 2025; Friel et al., 2025), as highlighted in recent surveys (Reddy & Shojaee, 2025; Eger et al., 2025; Kulkarni et al., 2025; Ren et al., 2025).

Single-Cell Perturbation Analysis Single-cell perturbation studies measure how cells respond to genetic or chemical interventions. The existing literature of *in-silico* approaches that predict post-perturbation cell states reflects a fundamental divergence in machine learning, with each paradigm showcasing distinct philosophies for modeling cellular responses. Earlier efforts, such as linear regression (Dixit et al., 2016) or random forest feature selection (Skinnider et al., 2021), treated each gene or cell type in isolation. Deep generative models (Lotfollahi et al., 2019; Hetzel

et al., 2022; Lotfollahi et al., 2023), conceptualize perturbations as latent space transformations through linear shifts or decompositions that separate biological covariates. In contrast, network-based methods (Qiu et al., 2022; Roohani et al., 2024a; Bai et al., 2024; Kamimoto et al., 2023) explicitly incorporate biological knowledge via gene regulatory networks or cellular relationships. To further address the issue of cell heterogeneity, distribution alignment approaches such as optimal transport (Bunne et al., 2023; Dong et al., 2023) have been applied to machine learning models (Jiang et al., 2024), matching the distribution of control cells with perturbed cells. The emergence of transformer architectures represents the latest paradigm shift. These architectures (Hao et al., 2024; Theodoris et al., 2023; Cui et al., 2024; Levine et al., 2024) leverage pre-training at scale and self-attention mechanisms to model complex gene dependencies without explicit biological structure. This theoretical diversity creates a vast design space where selecting optimal architectures, representation strategies, and biological constraints remains highly context-dependent.

Notations. Let $X \in \mathbb{R}^{n \times d}$ denote the matrix of single-cell profiles, where n represents the number of cells and d represents the dimensionality of the measured features (including gene expression counts, chromatin accessibility or surface protein markers, depending on the modality). A dataset $D = \{(x_i, p_i, y_i)\}_{i=1}^{N}$ and a task description S are given, where $x_i \in \mathbb{R}^d$ represents the pre-perturbation profile, $p_i \in \mathcal{P}$ denotes the applied perturbation, and $y_i \in \mathbb{R}^{d'}$ corresponds to the observed post-perturbation profile. To verify the generalizability of our scAgents, we divide D into $D_{\text{train}} = \{(x_i, p_i, y_i)\}_{i=1}^{M}$ and $D_{\text{test}} = \{(x_i, p_i, y_i)\}_{i=1}^{K}$, where $p_i \in \mathcal{P}_{\text{test}}$ constitutes held-out perturbations and $x_i \in \mathcal{X}_{\text{test}}$ represents held-out cell profiles.

Problem Formulation. We formalize the challenge of predicting cellular responses to perturbations as learning a mapping function between pre-perturbation states and their corresponding post-perturbation outcomes. Specifically, for each perturbation $p \in \mathcal{P}$ (e.g., gene knockout, drug treatment, cytokine stimulus) applied to a subset of cells, we model the induced change in the cellular profile as a function $f_p : \mathbb{R}^d \to \mathbb{R}^{d'}$, where $f_p(x)$ predicts the post-perturbation profile of a cell x, potentially in a different modality.

The training objective involves learning a function $f_\theta : \mathbb{R}^d \times \mathcal{P} \to \mathbb{R}^{d'}$ that generalizes effectively to unseen perturbations and cell states, where ϕ is a trainable parameter(i.e., weights and biases). To capture intrinsic cell-state structure and enable efficient modeling, we incorporate learnable encoders $g_\phi : \mathbb{R}^d \to \mathbb{R}^h$, where ϕ is a trainable parameter and $z_i = g_\phi(x_i) \in \mathbb{R}^{n \times h}$ represents the latent embedding that preserves geometric relationships between control and perturbed states, thus facilitating accurate prediction of post-perturbation profiles.

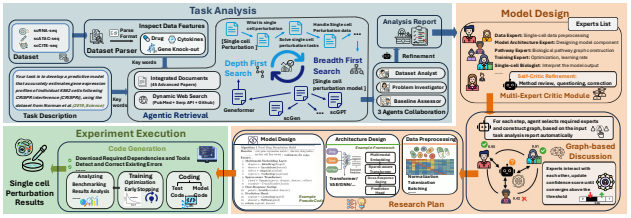

Figure 2: **The scAgents architecture and workflow.** The framework operates through three sequential phases: Task Analysis (dataset characterization and literature retrieval), Hypothesis Generation (collaborative development of novel approaches by role-based, self-refinement agent systems), and Experiment Execution (code generation, training implementation, and results analysis). All phases communicate through a shared knowledge graph that evolves throughout the workflow.

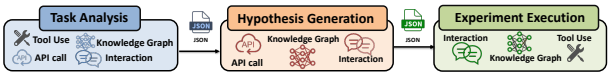

Figure 3: **The scAgents protocol overview.** The protocol framework integrates JSON-RPC with a persistent knowledge graph, combining the strengths of A2A and MCP protocols while adding scientific domain knowledge representation.

perturbation profiles.

Evaluation. For evaluation, we assess $f_\theta(x_i, p_i)$ for all $(x_i, p_i) \in \mathcal{X}_{\text{test}} \times \mathcal{P}_{\text{test}}$ and evaluate the quality of the learned representation $g_\phi(x)$ in terms of its ability to reconstruct gene expression profiles. Our evaluation protocols incorporate metrics such as mean squared error, Pearson's correlation coefficient, and perturbation consistency adapted from (Roohani et al., 2024a; Bendidi et al., 2024) to ensure biological significance. Appendix B provides a detailed explanation of these metrics.

Evaluation. For evaluation, we assess $f_\theta(x_i, p_i)$ for all $(x_i, p_i) \in \mathcal{X}_{\text{test}} \times \mathcal{P}_{\text{test}}$ and evaluate the quality of the learned representation $g_\phi(x)$ in terms of its ability to reconstruct gene expression profiles. Our evaluation protocols incorporate metrics such as mean squared error, Pearson's correlation coefficient, and perturbation consistency adapted from (Roohani et al., 2024a; Bendidi et al., 2024) to ensure biological significance. Appendix B provides a detailed explanation of these metrics.

scAgents orchestrates an end-to-end scientific workflow into a coherent framework (Figure 2): it autonomously (1) analyzes dataset characteristics and identifies optimal modeling strategies, (2) designs neural architectures incorporating bi-

ological knowledge about gene regulatory networks, and (3) generates executable code implementing the complete prediction pipeline, and evaluates prediction accuracy using various metrics. Central to scAgents is a hybrid communication protocol combining JSON-RPC data exchange with a persistent knowledge graph,which is further described in Appendix A.3. This shared representation continuously integrates outputs from individual agents, enabling complex reasoning chains while maintaining traceability throughout the scientific process. Detailed information of agent communication protocol are provided in Appendix E. The exact configurations, prompts, and outputs of each agent are listed in Appendix A.2 and F.

2.1. Task Analysis Module

The Task Analysis phase begins with comprehensive characterization of the scientific problem through three components:

Data Parser. This component extracts key metadata from single-cell datasets, including perturbation types, gene features, and cell populations. It standardizes information across diverse modalities (RNA-seq, ATAC-seq, CITE-seq) and generates summary statistics to establish the data foundation. The parser identifies critical experimental parameters such as perturbation methods, organism details, and cellular contexts without human intervention (detailed examples in Appendix G.1).

Agentic Retrieval. Our retrieval system combines a static corpus of 45 specialized articles with dynamic search capabilities through PubMed and GitHub APIs. Starting with an initial query $Q^{(0)}$ derived from the task description, the system employs Sentence-BERT (Reimers & Gurevych, 2019) embeddings and performs multi-layer retrieval alternating between breadth-first and depth-first search strategies. The detailed algorithm and mathematical formulation are presented in the Appendix D.1.

- *BFS layer* (t odd): Retrieves top-K documents $\mathcal{N}_t = \mathrm{TopK}(Q^{(t)}, \mathrm{mode} = \mathrm{BFS})$.
- *DFS layer* (t even): Follows highest-scoring paths from $\mathcal{N}_{t-1}$ in depth.

Document relevance is computed via cosine similarity: $\mathrm{Score}(Q, d) = \frac{e(Q)\cdot e(d)}{\|e(Q)\|\|e(d)\|}$. The retrieval terminates upon reaching any of three conditions: (1) maximum layers $L_{\max} = 10$, (2) query overlap exceeding threshold $\tau = 0.8$, or (3) document scores below $\epsilon = 0.5$. Results are stored in a vector database for subsequent access.

Agent Collaboration. Three specialized agents *Dataset Analyst*, *Problem Investigator*, *Baseline Assessor*process the retrieved information. The Dataset Analyst examines data integrity and characteristics, the Problem Investigator defines research questions and analytical approaches, and the Baseline Assessor establishes reference models and benchmarks. Their outputs are synthesized by a *Refinement Agent* that structures the analysis into a JSON file.

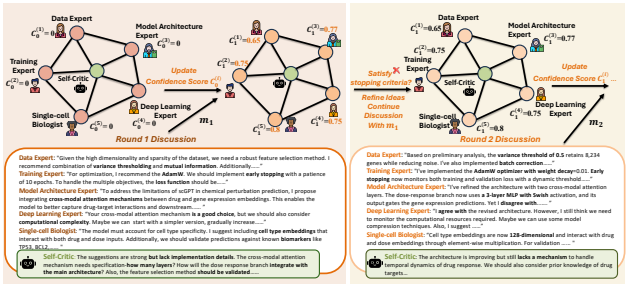

Figure 4: **The Graph-based discussion architecture and workflow.** This is an example of two rounds of discussion from the beginning. After each round, confidence scores are updated, the agentic system will judge if current state satisfy the stopping criteria. If not, each expert will refine their ideas based on the self-critic's suggestions and other experts' viewpoint. This graph-based self-critic refinement continues until reaching the terminate state. Complete multi-rounds of discussions are in Appendix D.2.

Analysis Report At the core of our framework's performance gains is its ability to autonomously discover optimal model architectures for each specific perturbation task. Similar to how human experts approach new domains, the agents systematically analyze scientific literature to identify candidate architectures appropriate for the given data characteristics. The Baseline Assessor specifically evaluates different model architectures (e.g. Transformers, GANs, VAEs, MLPs, etc.) based on their strengths and limitations for the particular perturbation type and data modality. For each architecture component, the agent generates a comprehensive analysis report detailing advantages, disadvantages, and suitability for specific aspects of the task (e.g., handling sparsity in scATAC-seq data). This literature-grounded comparative analysis forms the foundation for the Method Design module, where expert agents collaboratively refine these insights into optimized custom architectures.

2.2. Method Design Module

Overview. This module produces a research plan through collaborative agent discussion, concurrently producing three integrated components: (1) data preprocessing strategies, (2) model architecture design, and (3) concrete model implementation details. Architecture design involves textual descriptions of the neural network components selected for the specific perturbation task and their theoretical justifications, while model design translates these concepts into executable pseudocode. Unlike approaches that merely tune hyperparameters of fixed architectures, scAgents fun-

damentally focuses on discovering optimal architectural combinations tailored to each dataset's unique biological characteristics. This architectural discovery process, rather than hyperparameter optimization, is the primary source of our framework's performance advantages in perturbation prediction tasks.

Multi-Expert Critic System. For each task, the system dynamically selects a subset of domain experts $E^{(k)}$ (e.g., Data Engineer, Single-Cell Biologist, Deep Learning Expert) based on task requirements, along with a permanent Self-Critic. These agents form an undirected collaboration graph $G^{(k)} = (S, E^{(k)})$, where each expert node maintains a confidence score $c_t^{(i)}$ that evolves through discussion rounds.

Graph-based Discussion. The system performs T rounds of graph neural message passing, where experts propose architectural solutions, the Self-Critic identifies weaknesses, and all nodes update their perspectives based on collective feedback. After each round, confidence scores are updated according to the evaluation quality by both the Self-Critic and peer experts. The discussion ends when all experts' confidence scores exceed the threshold $\tau = 0.8$ with minimal variance ($\|c_{t^*}^{(i)} - c_{t^*}^{(i)}\| < \epsilon$). If not, experts refine their proposals using historical context and proceed to the next round. This process ensures convergence toward scientifically valid and technically feasible model designs with explicit reasoning chains. Further information of Experts selection and discussion construction is in Appendix A.4, detailed algorithm and mathematical formulation are presented in the Appendix D.2.

2.3. Experiment Execution Module

The Experiment Execution module transforms conceptual designs into operational implementations:
(1) Code Generation. The Code Generator translates method designs into production-ready implementations with complete dependency management. It produces modular code with built-in error detection mechanisms that automate the correction of common implementation issues. *(2) Training Process.* Model training proceeds through an automated scheduler implementing best-practice optimization routines and early stopping mechanisms. Hyperparameter optimization runs in parallel with model checkpointing to ensure reproducibility and enable efficient experimentation. *(3) Validation & Refinement.* Rather than one-shot generation, the Validation Agent implements an iterative refinement cycle. It evaluates performance against established metrics, identifies failure modes, and progressively enhances implementation quality. This evolutionary approach allows the system to address complex edge cases and optimize implementation based on empirical feedback.

3. Experiments

3.1. Evaluation setup

We evaluate the models designed and implemented by scAgents in various types of perturbation from scPerturb (Peidli et al., 2024), including gene knockouts, drug treatments, and cytokine stimulation across multiple modalities (scRNA-seq, scATAC-seq, CITE-seq).

Each dataset represents distinct biological challenges: The Adamson (Adamson et al., 2016) and Norman (Norman et al., 2019) datasets capture CRISPR gene knockouts in different cell lines, providing fundamental test cases for genetic perturbation. The Papalexi (Papalexi et al., 2021) dataset offers both RNA and protein measurements (CITE-seq), enabling assessment of cross-modality prediction. The Liscovitch (Liscovitch-Brauer et al., 2021) dataset presents the distinct challenge of predicting chromatin accessibility changes (scATAC-seq) rather than gene expression. The Srivatsan (Srivatsan et al., 2020) dataset assesses the prediction of cellular responses to chemical compounds, while the Schiebinger (Schiebinger et al., 2019) dataset examines responses to immune signaling molecules (cytokines).

To assess generalization to **unseen perturbations**, we select baselines accordingly: scGPT (Cui et al., 2024) for gene knockouts, ChemCPA (Hetzel et al., 2022) for drug treatments. For modalities lacking established models (scCITE-seq, scATAC-seq, cytokine), we employ Random Forest and Linear Regression using one-hot encoded perturbations concatenated with expression profiles as inputs.

3.2. Predictive Performance

Table 1 evaluates the prediction accuracy of models designed by scAgents across diverse perturbation datasets. We employ multiple complementary metrics: **Basic prediction accuracy** is measured via mean squared error (MSE↓), where lower values indicate predictions closer to actual gene expression; Pearson correlation coefficient (PCC↑), which quantifies how well predicted expression patterns correlate with actual patterns; and coefficient of determination (R^2 ↑), measuring the proportion of expression variance explained by the model. **Biological relevance** is assessed through metrics on differentially expressed genes (DE) –those exhibiting significant expression changes after perturbation and thus most biologically meaningful. For each dataset, top 20 DE genes are selected based on ground truth expression changes under perturbation. We then compute the same metrics restricted to these DE gene subsets, denoted as MSE_{DE}, PCC_{DE}, and R^2_{DE}. Across these settings, models designed by scAgents consistently outperform baselines. For gene knockouts (Adamson dataset), our models achieve 49% reduction in prediction error (MSE = 0.0051 vs. 0.0100 for scGPT) with improved variance explained ($R^2 = 0.9761$ vs. 0.9649). This improvement extends to differentially

Table 1: Post-perturbation gene expression prediction results, where scAgents-Model is the prediction model automatically designed and implemented by scAgents. The reported metrics for *scAgents-Models* represent the best performance from three separate models automatically designed and trained by the framework, with standard deviations reflecting the variability across these runs.

MODEL	MSE ↓	PCC ↑	R^2 ↑	MSE (DE) ↓	PCC (DE) ↑	R^2 (DE) ↑
Gene Knock Out Perturbation – scRNAseq Dataset (Adamson et al. (Adamson et al., 2016))						
Random Forest	0.3053	0.2063	0.0504	0.5923	0.2632	0.1653
Linear Regression	0.5803	0.0026	0.0435	0.6995	0.0257	0.1074
scGPT (Cui et al., 2024)	0.0100	0.9861	0.9649	0.2562	0.9088	0.7911
scAgents-Models	**0.0051** ± 0.0063	**0.9883** ± 0.0459	**0.9761** ± 0.0803	**0.2013** ± 0.0444	0.9474 ± 0.0601	**0.8912** ± 0.0518
Gene Knock Out Perturbation – scRNAseq Dataset (Norman et al. (Norman et al., 2019))						
Random Forest	0.4059	0.1625	0.0623	0.6817	0.1428	0.0498
Linear Regression	0.4989	0.0244	0.0314	0.7331	0.0265	0.0238
scGPT (Cui et al., 2024)	0.0076	0.9823	0.9536	0.5318	0.8630	0.5652
scAgents-Models	**0.0034** ± 0.0023	**0.9846** ± 0.0418	**0.9609** ± 0.0081	**0.1736** ± 0.0677	0.8109 ± 0.0133	**0.5975** ± 0.0539
Gene Knock Out Perturbation – scCITEseq (RNA) Dataset (Papalexi et al. (Papalexi et al., 2021))						
Random Forest	0.0763	0.2124	0.4186	0.0911	0.2455	0.2185
Linear Regression	0.0764	0.0170	0.0254	0.0909	0.0218	0.0163
scAgents-Models	**0.0417** ± 0.0051	**0.6935** ± 0.1995	**0.3687** ± 0.0651	**0.0535** ± 0.1566	**0.6406** ± 0.1940	**0.2354** ± 0.0224
Gene Knock Out Perturbation – scCITEseq (Protein) Dataset (Papalexi et al. (Papalexi et al., 2021))						
Random Forest	0.0982	0.2704	0.0829	0.3071	0.4024	0.0466
Linear Regression	0.4901	0.3396	0.1241	0.4551	0.3087	0.3523
scAgents-Models	**0.0070** ± 0.0387	**0.7495** ± 0.0653	**0.6872** ± 0.0956	**0.2921** ± 0.0045	**0.7409** ± 0.0970	**0.5489** ± 0.0749
Gene Knock Out Perturbation – scATACseq Dataset (Liscovitch et al. (Liscovitch-Brauer et al., 2021))						
Random Forest	0.0432	0.0638	0.0040	0.0510	0.0509	0.0035
Linear Regression	0.5767	0.0486	0.0229	0.7750	0.0457	0.0021
scAgents-Models	**0.0327** ± 0.0430	**0.0855** ± 0.0357	**0.0678** ± 0.0120	**0.0406** ± 0.0268	**0.6991** ± 0.3173	**0.6400** ± 0.2779
Drug Perturbation – scRNA-seq Dataset (Srivatsan et al. (Srivatsan et al., 2020))						
Random Forest	0.5289	0.0527	0.0986	0.6138	0.0945	0.0817
Linear Regression	0.6703	0.0711	0.2826	0.5625	0.0763	0.0421
ChemCPA (Hetzel et al., 2022)	0.0847	0.7221	0.6930	0.1035	0.8053	0.7412
scAgents-Models	**0.0053** ± 0.0290	**0.8664** ± 0.1332	**0.7137** ± 0.0740	**0.0080** ± 0.0835	**0.9278** ± 0.1001	**0.7887** ± 0.0548
Cytokine Perturbation – scRNA-seq Dataset (Schiebinger et al. (Schiebinger et al., 2019))						
Random Forest	0.0762	0.2704	0.4186	0.0910	0.2124	0.2185
Linear Regression	0.4855	0.0785	0.0034	0.4359	0.0847	0.0013
scAgents-Models	**0.0428** ± 0.0205	**0.5697** ± 0.0943	**0.5043** ± 0.0541	**0.0144** ± 0.0349	**0.3396** ± 0.0403	**0.2832** ± 0.1154

expressed genes ($\text{MSE}_{\text{DE}} = 0.2013$ vs. `0.2562`), demonstrating enhanced biological fidelity.

The performance advantages become more pronounced in challenging cross-modality scenarios. For CITE-seq protein measurements, scAgents achieves 177% improvement in correlation (`PCC = 0.7495` vs. `0.2704` for Random Forest). For drug perturbations, our models deliver near-perfect predictions (`MSE = 0.0053`) and 20% higher correlation (`PCC = 0.8664` vs. `0.7221`) compared to ChemCPA. Perhaps most remarkably, scAgents maintains superior performance even on fundamentally different modalities such as chromatin accessibility (scATAC-seq), achieving tremendous improvement in variance explained ($R^2 = 0.0678$ vs. `0.0040`) and correlation for key regulatory regions ($\text{PCC}_{\text{DE}} = 0.6991$ vs. `0.0509`).

The input datasets, task descriptions, and the LLM interface (Claude 3.7) were held constant for each perturbation type throughout these ablation studies; only the internal modules were varied. Table 2 reveals the critical contributions of our key framework components. The agentic retrieval system improves performance substantially over the baseline (`PCC` from `0.0087` to `0.5643` on the Adamson dataset), demonstrating the importance of domain knowledge integration. Similarly, the graph-based discussion component with self-critic refinement provides complementary benefits (`PCC` from `0.0087` to `0.5310`). The combination of both components yields synergistic effects far exceeding their individual contributions (`PCC` reaching `0.9883`), highlighting how knowledge-guided collaborative reasoning enables effective scientific discovery. This pattern remains consistent across all perturbation types, suggesting scAgents addresses fundamental challenges rather than exploiting dataset-specific characteristics.

Table 2: Ablation study on the impact of key framework components. Performance comparison of different scAgents components across gene knockout, drug, and cytokine perturbation datasets.

MODEL	MSE ↓	PCC ↑	R^2 ↑	MSE (DE) ↓	PCC (DE) ↑	R^2 (DE) ↑
Gene Knock Out Perturbation (Adamson Dataset (Adamson et al., 2016))						
scAgents(baseline)	0.4776	0.0087	0.0410	0.6061	0.0940	0.1280
+ Normal RAG	0.2442	0.1008	0.1119	0.3997	0.3354	0.3667
+ Agentic Retrieval	0.1267	0.5643	0.5431	**0.1152**	0.5922	0.6067
+ Graph-Based Discussion	0.2751	0.5310	0.5874	0.2792	0.6540	0.5311
+ Normal RAG & Graph-Based Discussion	0.0909	0.8951	0.8658	0.3416	0.8547	0.6770
+ Agentic Retrieval & Graph-Based Discussion	**0.0051**	**0.9883**	**0.9761**	0.2013	**0.9474**	**0.8912**
Drug Perturbation (Srivatsan Dataset (Srivatsan et al., 2020))						
scAgents(baseline)	0.5760	0.0298	0.0475	0.6409	0.0992	0.1039
+ Normal RAG	0.2572	0.1584	0.1038	0.3022	0.3472	0.2901
+ Agentic Retrieval	0.1309	0.3437	0.4350	0.1210	0.3836	0.4169
+ Graph-Based Discussion	0.1670	0.4193	0.3764	0.1325	0.4266	0.3865
+ Normal RAG & Graph-Based Discussion	0.0995	0.6512	0.5933	0.985	0.6784	0.7548
+ Agentic Retrieval & Graph-Based Discussion	**0.0053**	**0.9881**	**0.9665**	**0.0080**	**0.9953**	**0.9802**
Cytokine Perturbation (Schiebinger Dataset (Schiebinger et al., 2019))						
scAgents-Model (baseline)	0.5892	0.0065	0.0021	0.5876	0.0797	0.0999
+ Normal RAG	0.4321	0.1765	0.0243	0.4756	0.1987	0.0934
+ Agentic Retrieval	0.3456	0.2034	0.2421	0.3076	0.2068	0.1176
+ Graph-Based Discussion	0.3512	0.2051	0.2765	0.2454	0.2239	0.1123
+ Normal RAG & Graph-Based Discussion	0.0987	0.4875	0.4654	0.1065	0.2534	0.1053
+ Agentic Retrieval & Graph-Based Discussion	**0.0428**	**0.5697**	**0.5042**	**0.0144**	**0.3396**	**0.1240**

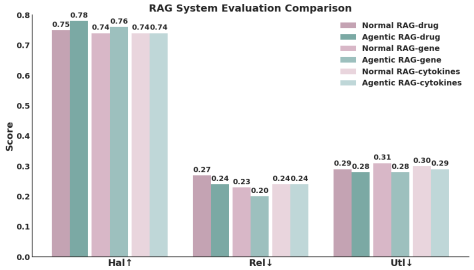

Figure 5: The performance of scAgents' RAG compared to standard RAG methods. (1) hal: hallucination detection, (2) rel: context relevance, (3) utl: context utilization. Results are stratified by perturbation type (Drug, Cytokine, Gene). Detailed evaluation methods are stated in Appendix C.

Table 3: Performance comparison on scPerturb datasets and benchmark tasks (all values are in %). Results show scAgents consistently outperforms both scGPT and Geneformer across multiple metrics and perturbation types. Each score represents the average of five independent runs, with higher values indicating better performance.

MODEL	TOP5 LIN ↑	TOP1 LIN ↑	PERT CONS ↑	TOP5 KNN ↑	TOP1 KNN ↑	SPEAR CORR ↑	STRUCT INT ↑
Drug Perturbation (Srivatsan Dataset (Srivatsan et al., 2020))							
scGPT(Cui et al., 2024)	5.2	**4.4**	**11.4**	5.6	5.1	18.8	54.2
Geneformer(Cui et al., 2023)	4.4	3.1	0.9	5.1	4.8	17.3	54.1
scAgents-Model	**7.0**	4.2	**11.4**	**6.4**	**5.3**	**19.1**	**54.5**
Gene Knock Out Perturbation (Adamson Dataset (Adamson et al., 2016))							
scGPT (Cui et al., 2024)	2.2	0.8	5.6	26.2	25.5	87.3	**96.1**
Geneformer (Cui et al., 2023)	2.1	0.8	4.3	25.9	24.1	86.6	95.9
scAgents-Model	**2.4**	**0.9**	**6.9**	**26.6**	**25.9**	**89.9**	96.0
Cytokine Perturbation (Schiebinger Dataset (Schiebinger et al., 2019))							
scGPT(Cui et al., 2024)	2.1	4.8	4.6	8.2	5.5	66.9	57.1
Geneformer(Cui et al., 2023)	1.4	4.2	4.4	8.3	**9.9**	68.2	57.6
scAgents-Model	**2.5**	**5.3**	**4.9**	**8.6**	8.8	**68.5**	**59.6**

3.3. Component Contributions
3.4. Information Integration

Evaluation on RAGBench (Friel et al., 2025) with the PubMedQA dataset (Jin et al., 2019) demonstrates scAgents' ability to accurately identify and contextualize relevant scientific literature. In this evaluation: hallucination detection (measured by AUROC) evaluates the system's ability to avoid generating false information; context relevance (measured by RMSE) assesses how well the retrieved documents match the query; and context utilization (measured by RMSE) quantifies how effectively the system incorporates retrieved information into responses. In Figure 5, for gene perturbations, context utilization showed the greatest improvement. Performance of scAgents remains consistent across all perturbation types, suggesting scAgents addresses fundamental challenges rather than exploiting dataset-specific characteristics.

3.5. Representation Quality Evaluation on scPerturb Benchmark

While scAgents primarily performs gene expression prediction following perturbations, the quality of learned representations is equally important for biological interpretability. Following evaluation practices established in previous works (Cui et al., 2024; Theodoris et al., 2023), we benchmark scAgents against specialized foundation models (scGPT & Geneformer) on representation quality metrics (Table 3). We assess different aspects of latent space organization: **(1) Linear separability metrics** (TOP5_LIN, TOP1_LIN) measure how distinguishable different perturbation types are in the latent space. The top5_lin score of 0.070 achieved by scAgents for drug perturbations (vs. 0.052 for scGPT) indicates that 7.0% of test samples have their correct perturbation label among the top 5 predictions when using a linear classifier trained on the latent embeddings. This improvement suggests scAgents learns representations where perturbation effects are more linearly separable, facilitating downstream analyses that rely on perturbation classification. **(2) Perturbation consistency** (PERT_CONS) quantifies whether cells with the same perturbation cluster more tightly than random controls, essentially measuring the signal-to-noise ratio of perturbation effects in the latent space. For gene knockouts, scAgents achieves a consistency of 0.069 vs. 0.056 for scGPT, representing a 23.2% improvement. This indicates that scAgents creates a latent space where cells experiencing the same perturbation are more reliably grouped together, reflecting better capture of perturbation-specific biological responses. **(3) Local structure in the latent space** is assessed through nearest-neighbor metrics (TOP5_KNN, TOP1_KNN), which evaluate whether perturbations form locally coherent clusters. For drug perturbations, scAgents achieves a TOP5_KNN score of 0.064 vs. 0.056 for scGPT, indicating that a higher proportion of test samples have correctly labeled neighbors in embedding space. **(4) The Spearman correlation metric** (SPEAR_CORR) evaluates how accurately the latent embeddings can be mapped back to original gene expression space using a linear transformation. The score of 0.191 for drug perturbations (vs. 0.188 for scGPT) represents a higher rank correlation between predicted and actual expression values after linear decoding. **(5) Structural integrity** (STRUCT_INT) measures how well control-perturbation relationships are preserved in the latent space. **0.596** for cytokine perturbations (vs. 0.571 for scGPT) indicates that scAgents better maintains the biological relationship between control and perturbed states for complex signaling cascades. We include additional UMAP visualizations in the Appendix O.

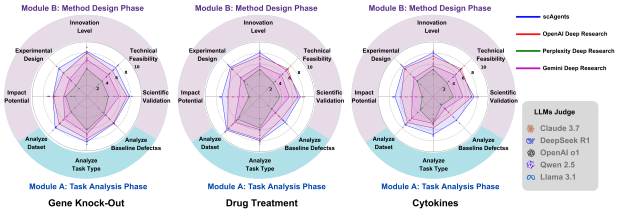

Figure 6: We manually prompt four different DeepResearch variants to generate research plans, which were then evaluated by five independent LLMs across eight dimensions, with scores ranging from 1 to 10. Detailed prompt, outputs and scores are provided in Appendix I.

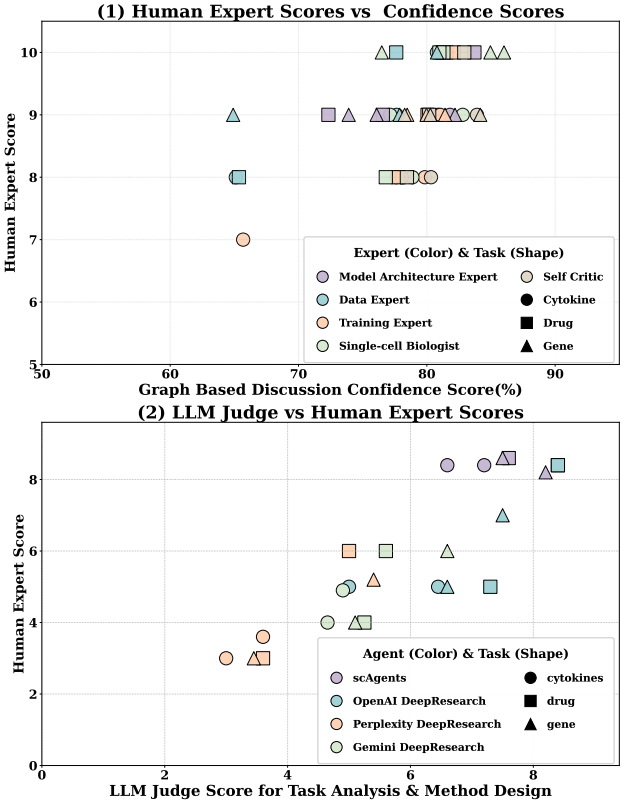

Figure 7: Human expert evaluations show strong correlation with agent-generated confidence and LLM judge scores. Detailed scores are provided in Appendix I and J.

3.6. Evaluation on Task Analysis Module and Method Design Module

Beyond predictive performance, Figure 6 presents expert evaluations comparing scAgents with DeepResearch variants. We employ an evaluation protocol involving five independent LLM judges that evaluated research plans across eight dimensions. Each judge assesses outputs in a randomized, blinded manner without knowledge of which system generated each plan. The evaluation shows that scAgents demonstrates superior performance in dataset analysis and baseline defect identification. In method design, scAgents consistently outperforms alternatives on scientific validity, innovation level, experimental design, and impact potential, while maintaining comparable technical feasibility. These advantages are particularly pronounced for cytokine perturbations, where immune signaling complexity demands sophisticated biological reasoning.

Notably, both the LLM-assigned scores for task analysis and research plans, and the confidence scores generated by our agent system, show strong correlation with evaluations from three human domain experts (Pearson $r = 0.83$, $p < 0.01$ in Figure 7). Each expert independently spent approximately 10 hours scoring the same outputs under identical criteria, blinded to the source system.

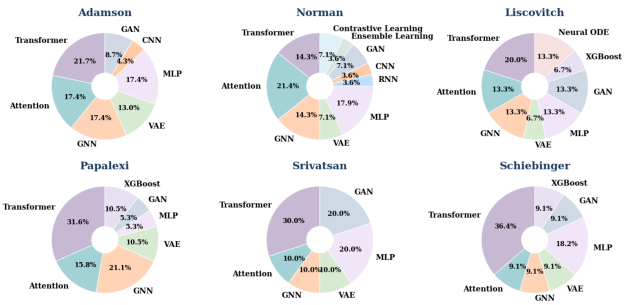

Figure 8: A manual post-hoc analysis where we categorized and quantified the architectures designed by scAgents for six datasets.

To better understand how scAgents adapts to different perturbation scenarios, we conducted a post-hoc analysis of the architectures it generated. For each of the six datasets, we manually categorized and quantified the neural network components present in scAgents designs. Figure 8 shows the distribution of models across datasets. This analysis reveals how the framework naturally tailored its designs to different data types - for example, preferring Transformers (36.4%) for cytokine perturbations to capture complex signaling dependencies. These adaptive design choices emerged organically from the system's literature-based knowledge and expert discussions rather than through predefined rules or templates.

4. Conclusion

scAgents demonstrates how autonomous multi-agent systems can successfully improve single-cell perturbation analysis through integrated expertise across computational, biological, and statistical domains. Our framework achieves consistent performance improvements across diverse datasets, perturbation types, and modalities. The synergistic combination of knowledge integration (via agentic retrieval) and collaborative reasoning (via graph-based discussion) enables discovery of optimal modeling strategies without human intervention, with emergent architecture adaptation to dataset-specific challenges.

References

Adamson, B., Norman, T. M., Jost, M., Cho, M. Y., Nuñez, J. K., Chen, Y., Villalta, J. E., Gilbert, L. A., Horlbeck, M. A., Hein, M. Y., et al. A multiplexed single-cell CRISPR screening platform enables systematic dissection of the unfolded protein response. *Cell*, 167(7):1867–1882, 2016.

Baek, J., Jauhar, S. K., Cucerzan, S., and Hwang, S. J. ResearchAgent: Iterative research idea generation over scientific literature with large language models. *arXiv preprint arXiv:2404.07738*, 2024.

Bai, D., Ellington, C. N., Mo, S., Song, L., and Xing, E. P.

AttentionPert: accurately modeling multiplexed genetic perturbations with multi-scale effects. *Bioinformatics*, 40 (Supplement_1):i453–i461, 2024.

Beel, J., Kan, M.-Y., and Baumgart, M. Evaluating sakana's AI scientist for autonomous research: Wishful thinking or an emerging reality towards' artificial research intelligence'(ARI)? *arXiv preprint arXiv:2502.14297*, 2025.

Bendidi, I., Whitfield, S., Kenyon-Dean, K., Yedder, H. B., Mesbahi, Y. E., Noutahi, E., and Denton, A. K. Benchmarking transcriptomics foundation models for perturbation analysis: one PCA still rules them all, 11 2024. URL http://arxiv.org/abs/2410.13956.

Boiko, D. A., MacKnight, R., and Gomes, G. Autonomous chemical research with large language models. *Nature*, 623:760–768, 2023. doi: 10.1038/s41586-023-06792-0.

Bunne, C., Stark, S. G., Gut, G., Del Castillo, J. S., Levesque, M., Lehmann, K.-V., Pelkmans, L., Krause, A., and Rätsch, G. Learning single-cell perturbation responses using neural optimal transport. *Nature methods*, 20(11):1759–1768, 2023.

Chen, T., Anumasa, S., Lin, B., Shah, V., Goyal, A., and Liu, D. Auto-Bench: An automated benchmark for scientific discovery in LLMs. *arXiv preprint arXiv:2502.15224*, 2025.

Chen, Z., Chen, S., Ning, Y., Zhang, Q., Wang, B., Yu, B., Li, Y., Liao, Z., Wei, C., Lu, Z., et al. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. *arXiv preprint arXiv:2410.05080*, 2024.

Cui, H., Wang, C., Maan, H., Pang, K., Luo, F., Duan, N., and Wang, B. scGPT: toward building a foundation model for single-cell multi-omics using generative ai. *Nature Methods*, 21(8):1470–1480, 08 2024. ISSN 1548-7091.

Cui, Z., Liao, Y., Xu, T., and Wang, Y. Geneformer: Learned gene compression using transformer-based context modeling. *arXiv preprint arXiv:2212.08379*, 2023.

Dixit, A., Parnas, O., Li, B., Chen, J., Fulco, C. P., Jerby-Arnon, L., Marjanovic, N. D., Dionne, D., Burks, T., Raychowdhury, R., et al. Perturb-Seq: dissecting molecular circuits with scalable single-cell RNA profiling of pooled genetic screens. *Cell*, 167(7):1853–1866, 2016.

Dong, M., Wang, B., Wei, J., de O. Fonseca, A. H., Perry, C. J., Frey, A., Ouerghi, F., Foxman, E. F., Ishizuka, J. J., Dhodapkar, R. M., et al. Causal identification of single-cell experimental perturbation effects with CINEMA-OT. *Nature methods*, 20(11):1769–1779, 2023.

Eger, S., Cao, Y., D'Souza, J., Geiger, A., Greisinger, C., Gross, S., Hou, Y., Krenn, B., Lauscher, A., Li, Y., et al. Transforming science with large language models: A survey on AI-assisted scientific discovery, experimentation, content generation, and evaluation. *arXiv preprint arXiv:2502.05151*, 2025.

Friel, R., Belyi, M., and Sanyal, A. RAGBench: Explainable benchmark for retrieval-augmented generation systems, 2025. URL http://arxiv.org/abs/2407.11005.

Gao, X., Zhang, Z., Xie, M., Liu, T., and Fu, Y. Graph of AI ideas: Leveraging knowledge graphs and llms for AI research idea generation. *arXiv preprint arXiv:2503.08549*, 2025.

Garikaparthi, A., Patwardhan, M., Vig, L., and Cohan, A. IRIS: Interactive research ideation system for accelerating scientific discovery. *arXiv preprint arXiv:2504.16728*, 2025.

Ghafarollahi, A. and Buehler, M. J. AtomAgents: Alloy design and discovery through physics-aware multimodal multi-agent artificial intelligence. *arXiv preprint arXiv:2407.10022*, 2024.

Ghafarollahi, A. and Buehler, M. J. Sparks: Multi-agent artificial intelligence model discovers protein design principles. *arXiv preprint arXiv:2504.19017*, 2025.

Gu, K., Shang, R., Jiang, R., Kuang, K., Lin, R.-J., Lyu, D., Mao, Y., Pan, Y., Wu, T., Yu, J., et al. BLADE: Benchmarking language model agents for data-driven science. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 13936–13971, 2024.

Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N. V., Wiest, O., and Zhang, X. Large language model based multi-agents: A survey of progress and challenges, 02 2024. URL https://arxiv.org/abs/2402.01680.

Hao, M., Gong, J., Zeng, X., Liu, C., Guo, Y., Cheng, X., Wang, T., Ma, J., Zhang, X., and Song, L. Large-scale foundation model on single-cell transcriptomics. *Nature methods*, 21(8):1481–1491, 2024.

Hetzel, L., Boehm, S., Kilbertus, N., Günnemann, S., Theis, F., et al. Predicting cellular responses to novel drug perturbations at a single-cell resolution. *Advances in Neural Information Processing Systems*, 35:26711–26722, 2022.

Honovich, O., Scialom, T., Levy, O., and Schick, T. True few-shot learning with prompts – a real-world perspective. *Transactions of the Association for Computational Linguistics*, 10:1032–1047, 2022.

House, F. FutureHouse platform: Superintelligent AI agents for scientific discovery. https://www.futurehouse.org/research-announcements/launching-futurehouse-platform-ai-agents, 2024. Accessed on 2025-05-06; Nonprofit organization developing AI scientist tools for automated research workflows.

Hsu, C.-C., Bransom, E., Sparks, J., Kuehl, B., Tan, C., Wadden, D., Wang, L. L., and Naik, A. CHIME: LLM-assisted hierarchical organization of scientific studies for literature review support. *Findings of ACL 2024*, 2024.

Hu, X., Fu, H., Wang, J., Wang, Y., Li, Z., Xu, R., Lu, Y., Jin, Y., Pan, L., and Lan, Z. Nova: An iterative planning and search approach to enhance novelty and diversity of llm generated ideas. *arXiv preprint arXiv:2410.14255*, 2024.

Huang, Q., Vora, J., Liang, P., and Leskovec, J. MLAgentBench: Evaluating language agents on machine learning experimentation. In *ICML 2024*, 2024a.

Huang, Y., Luo, J., Yu, Y., Zhang, Y., Lei, F., Wei, Y., He, S., Huang, L., Liu, X., Zhao, J., and Liu, K. DA-Code: Agent data science code generation benchmark for large language models. *EMNLP 2024*, 2024b.

Jansen, P., Tafjord, O., Radensky, M., Siangliulue, P., Hope, T., Mishra, B. D., Majumder, B. P., Weld, D. S., and Clark, P. Codescientist: End-to-end semi-automated scientific discovery with code-based experimentation. *arXiv preprint arXiv:2503.22708*, 2025.

Jiang, Q., Chen, S., Chen, X., and Jiang, R. scPRAM accurately predicts single-cell gene expression perturbation response based on attention mechanism. *Bioinformatics*, 40(5):btae265, 2024.

Jin, Q., Dhingra, B., Liu, Z., Cohen, W. W., and Lu, X. PubMedQA: A dataset for biomedical research question answering. *arXiv preprint arXiv:1909.06146*, 2019.

Jing, L., Huang, Z., Wang, X., Yao, W., Yu, W., Ma, K., Zhang, H., Du, X., and Yu, D. DSBench: How far are data science agents to becoming data science experts? *arXiv preprint arXiv:2409.07703*, 2024.

Kamimoto, K., Stringa, B., Hoffmann, C. M., Jindal, K., Solnica-Krezel, L., and Morris, S. A. Dissecting cell identity via network inference and in silico gene perturbation. *Nature*, 614(7949):742–751, 2023.

Kon, P. T. J., Liu, J., Ding, Q., Qiu, Y., Yang, Z., Huang, Y., Srinivasa, J., Lee, M., Chowdhury, M., and Chen, A. Curie: Toward rigorous and automated scientific experimentation with AI agents. *arXiv preprint arXiv:2502.16069*, 2025.

Kulkarni, A., Alotaibi, F., Zeng, X., Wu, L., Zeng, T., Yao, B. M., Liu, M., Zhang, S., Huang, L., and Zhou, D. Scientific hypothesis generation and validation: Methods, datasets, and future directions. *arXiv preprint arXiv:2505.04651*, 2025.

Levine, D., Rizvi, S. A., Lévy, S., Pallikkavaliyaveetil, N., Zhang, D., Chen, X., Ghadermarzi, S., Wu, R., Zheng, Z., Vrkic, I., et al. Cell2Sentence: teaching large language models the language of biology. *BioRxiv*, pp. 2023–09, 2024.

Li, L., Xu, W., Guo, J., Zhao, R., Li, X., Yuan, Y., Zhang, B., Jiang, Y., Xin, Y., Dang, R., Zhao, D., Rong, Y., Feng, T., and Bing, L. Chain of ideas: Revolutionizing research via novel idea development with llm agents. *arXiv preprint arXiv:2410.13185*, 2024a.

Li, R., Patel, T., Wang, Q., and Du, X. MLR-Copilot: Autonomous machine learning research based on large language models agents. *arXiv preprint arXiv:2408.14033*, 2024b.

Lin, Z. Beyond principlism: practical strategies for ethical ai use in research practices. *AI and Ethics*, pp. 1–13, 2024.

Liscovitch-Brauer, N., Montalbano, A., Deng, J., Méndez-Mancilla, A., Wessels, H.-H., Moss, N. G., Kung, C.-Y., Sookdeo, A., Guo, X., Geller, E., et al. Profiling the genetic determinants of chromatin accessibility with scalable single-cell crispr screens. *Nature biotechnology*, 39(10):1270–1277, 2021.

Liu, H., Li, Y., Jian, J., Cheng, Y., Lu, J., Guo, S., Zhu, J., Zhang, M., Zhang, M., and Wang, H. Toward a team of AI-made scientists for scientific discovery from gene expression data. *arXiv preprint arXiv:2402.12391*, 2024a.

Liu, H., Zhou, Y., Li, M., Yuan, C., and Tan, C. Literature meets data: A synergistic approach to hypothesis generation. *arXiv preprint arXiv:2410.17309*, 2024b.

Liu, Z., Liu, K., Zhu, Y., Lei, X., Yang, Z., Zhang, Z., Li, P., and Liu, Y. AIGS: Generating science from AI-powered automated falsification. *arXiv preprint arXiv:2411.11910*, 2024c.

Lotfollahi, M., Wolf, F. A., and Theis, F. J. scGen predicts single-cell perturbation responses. *Nature methods*, 16 (8):715–721, 2019.

Lotfollahi, M., Klimovskaia Susmelj, A., De Donno, C., Hetzel, L., Ji, Y., Ibarra, I. L., Srivatsan, S. R., Naghipourfar, M., Daza, R. M., Martin, B., et al. Predicting cellular responses to complex perturbations in high-throughput screens. *Molecular systems biology*, 19(6):e11517, 2023.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The AI scientist: Towards fully automated open-ended scientific discovery, 09 2024. URL http://arxiv.org/abs/2408.06292.

Majumder, B. P., Surana, H., Agarwal, D., Mishra, B. D., Meena, A., Prakhar, A., Vora, T., Khot, T., Sabharwal, A., and Clark, P. DiscoveryBench: Towards data-driven discovery with large language models. *arXiv preprint arXiv:2407.01725*, 2024.

Moerman, T., Aibar Santos, S., Bravo González-Blas, C., Simm, J., Moreau, Y., Aerts, J., and Aerts, S. GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks. *Bioinformatics*, 35(12):2159–2161, 2019.

Naumov, V., Zagirova, D., Lin, S., Xie, Y., Gou, W., Urban, A., Tikhonova, N., Alawi, K., Durymanov, M., Galkin, F., et al. DORA AI scientist: Multi-agent virtual research team for scientific exploration discovery and automated report generation. *bioRxiv*, 2025.

Norman, T. M., Horlbeck, M. A., Replogle, J. M., Ge, A. Y., Xu, A., Jost, M., Gilbert, L. A., and Weissman, J. S. Exploring genetic interaction manifolds constructed from rich single-cell phenotypes. *Science*, 365(6455):786–793, 2019.

OpenAI. Introducing deep research. https://openai.com/index/deep-research/, 2025. Accessed: 2025-05-08.

Papalexi, E., Mimitou, E. P., Butler, A. W., Foster, S., Bracken, B., Mauck III, W. M., Wessels, H.-H., Hao, Y., Yeung, B. Z., Smibert, P., et al. Characterizing the molecular regulation of inhibitory immune checkpoints with multimodal single-cell screens. *Nature genetics*, 53 (3):322–331, 2021.

Peidli, S., Green, T. D., Shen, C., Gross, T., Min, J., Garda, S., Yuan, B., Schumacher, L. J., Taylor-King, J. P., Marks, D. S., et al. scPerturb: harmonized single-cell perturbation data. *Nature Methods*, 21(3):531–540, 2024.

Pu, K., Feng, K., Grossman, T., Hope, T., Mishra, B. D., Latzke, M., Bragg, J., Chang, J. C., and Siangliulue, P. IdeaSynth: Iterative research idea development through evolving and composing idea facets with literature-grounded feedback. *arXiv preprint arXiv:2410.04025*, 2024.

Qi, B., Zhang, K., Li, H., Tian, K., Zeng, S., Chen, Z.-R., Hu, J.-F., and Zhou, B. Large language models are zero shot hypothesis proposers. *Instruction Workshop @ NeurIPS 2023*, 2023.

Qiu, X., Zhang, Y., Martin-Rufino, J. D., Weng, C., Hosseinzadeh, S., Yang, D., Pogson, A. N., Hein, M. Y., Min, K. H. J., Wang, L., et al. Mapping transcriptomic vector fields of single cells. *Cell*, 185(4):690–711, 2022.

Qiu, Y., Zhang, H., Xu, Z., Li, M., Song, D., Wang, Z., and Zhang, K. AI Idea Bench 2025: AI research idea generation benchmark. *arXiv preprint arXiv:2504.14191*, 2025.

Radensky, M., Shahid, S., Fok, R., Siangliulue, P., Hope, T., and Weld, D. S. Scideator: Human-llm scientific idea generation grounded in research-paper facet recombination. *arXiv preprint arXiv:2409.14634*, 2024.

Reddy, C. K. and Shojaee, P. Towards scientific discovery with generative AI: Progress, opportunities, and challenges. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 28601–28609, 2025.

Reimers, N. and Gurevych, I. Sentence-BERT: Sentence embeddings using siamese BERT-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Ren, S., Jian, P., Ren, Z., Leng, C., Xie, C., and Zhang, J. Towards scientific intelligence: A survey of llm-based scientific agents. *arXiv preprint arXiv:2503.24047*, 2025.

Roohani, Y., Huang, K., and Leskovec, J. Predicting transcriptional outcomes of novel multigene perturbations with GEARS. *Nature Biotechnology*, 42(6):927–935, 2024a.

Roohani, Y. H., Vora, J., Huang, Q., Liang, P., and Leskovec, J. BioDiscoveryAgent: An ai agent for designing genetic perturbation experiments. In *ICLR 2024 Workshop on Machine Learning for Genomics Explorations*, 2024b.

Ruan, K., Wang, X., Hong, J., Wang, P., Liu, Y., and Sun, H. LiveIdeaBench: Evaluating llms' scientific creativity and idea generation with minimal context. *arXiv preprint arXiv:2412.17596*, 2024.

Schiebinger, G., Shu, J., Tabaka, M., Cleary, B., Subramanian, V., Solomon, A., Gould, J., Liu, S., Lin, S., Berube, P., et al. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943, 2019.

Schmidgall, S. and Moor, M. AgentRxiv: Towards collaborative autonomous research. *arXiv preprint arXiv:2503.18102*, 2025.

Schmidgall, S., Su, Y., Wang, Z., Sun, X., Wu, J., Yu, X., Liu, J., Liu, Z., and Barsoum, E. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

Si, C., Yang, D., and Hashimoto, T. Can LLMs generate novel research ideas? a large-scale human study with 100+ NLP researchers. *arXiv preprint arXiv:2409.04109*, 2024.

Skarlinski, M. D., Cox, S., Laurent, J. M., Braza, J. D., Hinks, M., Hammerling, M. J., Ponnapati, M., Rodriques, S. G., and White, A. D. Language agents achieve superhuman synthesis of scientific knowledge. *arXiv preprint arXiv:2409.13740*, 2024.

Skinnider, M. A., Squair, J. W., Kathe, C., Anderson, M. A., Gautier, M., Matson, K. J., Milano, M., Hutson, T. H., Barraud, Q., Phillips, A. A., et al. Cell type prioritization in single-cell data. *Nature biotechnology*, 39(1):30–34, 2021.

Srivatsan, S. R., McFaline-Figueroa, J. L., Ramani, V., Saunders, L., Cao, J., Packer, J., Pliner, H. A., Jackson, D. L., Daza, R. M., Christiansen, L., et al. Massively multiplex chemical transcriptomics at single-cell resolution. *Science*, 367(6473):45–51, 2020.

Starace, G., Jaffe, O., Sherburn, D., Aung, J., Shern, C. J., Maksin, L., Dias, R., Mays, E., Kinsella, B., Thompson, W., Heidecke, J., Glaese, A., and Patwardhan, T. PaperBench: Evaluating AI's ability to replicate AI research. *arXiv preprint arXiv:2504.01848*, 2025. URL https://arxiv.org/abs/2504.01848.

Su, H., Chen, R., Tang, S., Zheng, X., Li, J., Yin, Z., Ouyang, W., and Dong, N. Two heads are better than one: A multi-agent system has the potential to improve scientific idea generation. *arXiv preprint arXiv:2410.09403*, 2024. URL https://arxiv.org/abs/2410.09403.

Tang, X., Zou, A., Zhang, Z., Li, Z., Zhao, Y., Zhang, X., Cohan, A., and Gerstein, M. MedAgents: Large language models as collaborators for zero-shot medical reasoning. *Findings of ACL 2024*, 2024.

Theodoris, C. V., Xiao, L., Chopra, A., Chaffin, M. D., Al Sayed, Z. R., Hill, M. C., Mantelos, H., Brydon, E. M., Zeng, Z., Liu, X. S., and Ellinor, P. T. Transfer learning enables predictions in network biology. *Nature*, 618: 616–624, 05 2023.

Tian, M., Gao, L., Zhang, S., Chen, X., Fan, C., Guo, X., Haas, R., Ji, P., Krongchon, K., Li, Y., et al. SciCode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37: 30624–30650, 2024.

Wenteler, A., Occhetta, M., Branson, N., Huebner, M., Curean, V., Dee, W., Connell, W., Hawkins-Hooker, A., Chung, P., Ektefaie, Y., et al. PertEval-scFM: Benchmarking single-cell foundation models for perturbation effect prediction. *bioRxiv*, pp. 2024–10, 2024.

Yan, X., Feng, S., Yuan, J., Xia, R., Wang, B., Zhang, B., and Bai, L. SurveyForge: On the outline heuristics, memory-driven generation, and multi-dimensional evaluation for automated survey writing. *arXiv preprint arXiv:2503.04629*, 2025.

Yang, F., Wang, W., Wang, F., Fang, Y., Tang, D., Huang, J., Lu, H., and Yao, J. scBERT as a large-scale pretrained deep language model for cell type annotation of single-cell rna-seq data. *Nature Machine Intelligence*, 4(10): 852–866, 2022.

Yang, T., Hu, X., Li, X., Tan, M., Zhang, J., Wen, Z., Chang, E., Dai, A. M., Li, Q. V., Gonzalez, J. E., Cardie, C., and Wei, J. AnyBench: Language models evaluate anything. *arXiv preprint arXiv:2312.13771*, 2023.

Zhong, R., Zhang, P., Li, S., Ahn, J., Klein, D., and Steinhardt, J. Goal driven discovery of distributional differences via language descriptions. In *NeurIPS 2023*, 2023.

Limitation

Despite the demonstrated effectiveness of scAgents in single-cell perturbation analysis, several limitations warrant consideration:

Computational and Economic Costs. Our framework requires substantial computational resources and API calls to large language models. As shown in Table 15, the average cost per experiment ranges from $0.38 to $18.90 depending on the LLM backend, which may limit accessibility for resource-constrained research groups. Additionally, the multi-agent discussion process can require up to 400,000 output tokens per task, resulting in significant latency.

Failure Modes. Our failure analysis (Section N) reveals that 41% of errors stem from computation execution issues, particularly tensor dimension mismatches. While we implemented mitigation strategies, these systematic failures highlight the challenges of fully autonomous code generation for complex biological data analysis.

Domain Specificity. Currently, scAgents is specialized for single-cell perturbation prediction tasks. Extending the framework to other biological domains (e.g., spatial transcriptomics, proteomics, or multi-omics integration) would require substantial architectural modifications and domain-specific knowledge engineering.

Dependence on Task Specification Quality. The framework's performance is sensitive to the clarity and completeness of input task descriptions. Ambiguous or incomplete specifications can lead to suboptimal model designs, requiring users to have sufficient domain expertise to formulate well-defined research questions.

Limited Biological Novelty Detection. While scAgents excels at combining existing knowledge, it may struggle to propose truly novel biological mechanisms beyond the patterns present in its training data and retrieved literature. This could limit its utility for discovering fundamentally new biological phenomena.

Broader Impact

The development of scAgents has several important implications for the scientific community and society:

Democratization of Scientific Research. By automating complex analytical workflows, scAgents lowers the technical barriers for conducting sophisticated single-cell analyses. This could enable smaller research groups and institutions with limited computational expertise to engage in cutting-edge genomics research, potentially accelerating scientific discovery globally.

Acceleration of Therapeutic Development. The ability to rapidly predict cellular responses to perturbations has direct applications in drug discovery and personalized medicine. Our framework could significantly reduce the time and cost of identifying therapeutic targets and understanding drug mechanisms of action.

Scientific Reproducibility. The end-to-end automation and explicit documentation of all analytical decisions enhance reproducibility in computational biology. The generated code and research plans provide complete audit trails, addressing a critical challenge in modern scientific research.

Ethical Considerations. The automation of scientific discovery raises important questions about attribution, responsibility, and validation of AI-generated hypotheses. While scAgents includes human-interpretable outputs and confidence scores, researchers must carefully validate all predictions before drawing biological conclusions or making therapeutic decisions.

Environmental Impact. The computational requirements of running multiple LLM agents have a non-negligible carbon footprint. Future work should explore more efficient architectures and consider the environmental cost-benefit trade-offs of automated scientific discovery.

Research Workforce Evolution. As AI systems become capable of conducting increasingly sophisticated analyses, the role of computational biologists may shift from implementing methods to critically evaluating AI-generated hypotheses and designs. This transition requires careful consideration of training and career development in the scientific workforce.

Open Science Contribution. By releasing our code and model architectures, we aim to foster community-driven improvements and applications. However, we acknowledge the potential for misuse and encourage responsible deployment with appropriate biological validation and ethical oversight.

A. Experimental Details

A.1. Datasets Introduction

Our study leverages six publicly available single-cell perturbation datasets from the scPerturb (Peidli et al., 2024) collection, encompassing diverse perturbation modalities and cell types. These datasets provide a foundation for evaluating the scientific quality of AI-generated analyses across various biological contexts.

Adamson et al. (Adamson et al., 2016) (CRISPRi): Employing Perturb-seq to study the unfolded protein response (UPR) in K562 lymphoblasts through single and combinatorial CRISPR interference (CRISPRi) perturbations. Approximately 100 gene targets were profiled, enabling high-resolution functional clustering and revealing distinct activation patterns across UPR branches.

Norman et al. (Norman et al., 2019) (CRISPRa): Utilizing CRISPR activation (CRISPRa) in K562 cells, this dataset explores genetic interaction manifolds derived from single-cell transcriptional phenotypes. The study provides insights into regulatory pathway ordering and mechanistic elucidation of synergistic interactions.

Liscovitch et al. (Liscovitch-Brauer et al., 2021) (ATAC-seq): Employing CRISPRsciATAC, a single-cell combinatorial indexing assay, to delineate the genetic determinants of chromatin accessibility in human myelogenous leukemia K562 cells. Targeting 105 chromatin-related genes via CRISPR-Cas9, the study generated chromatin accessibility profiles for approximately 30,000 single cells. Key findings include correlations between the loss of specific chromatin remodelers and global changes in chromatin accessibility. Notably, EZH2 depletion was associated with enhanced accessibility in heterochromatic regions linked to embryonic development and with activation of genes in the HOXA and HOXD clusters. This high-throughput approach offers valuable insights into the role of chromatin modifiers in regulating gene expression and their implications in disease states.

Papalexi et al. (Papalexi et al., 2021) (CITE-seq): Combining CRISPR-Cas9 perturbations with single-cell RNA and surface protein measurements in THP-1 monocytes. It investigates the molecular regulation of inhibitory immune checkpoints, particularly PD-L1 expression, and introduces the mixscape computational framework to enhance signal-to-noise ratio in single-cell screens.

Srivatsan et al. (Srivatsan et al., 2020) (sci-Plex): Employing sci-Plex, this dataset profiles transcriptional responses of A549, K562, and MCF7 cancer cell lines to 188 small-molecule compounds across multiple doses. Approximately 650,000 single-cell transcriptomes were generated, uncovering intercellular heterogeneity and commonalities in drug responses.

Schiebinger et al. (Schiebinger et al., 2019) (cytokine perturbation): Applying optimal transport analysis to scRNA-seq data from mouse embryonic stem cells undergoing reprogramming with cytokine treatments. The dataset captures developmental trajectories and identifies transcription factors and paracrine signals influencing cell fate decisions.

Collectively, these datasets encompass a range of perturbation typesincluding CRISPRi, CRISPRa, CRISPR-Cas9, small-molecule drugs, and cytokinesacross various human and mouse cell lines. They provide a robust foundation for evaluating the scientific quality and reliability of AI-generated analyses in single-cell biology.

A.2. Agent Configurations

In our experiments, we employed five LLMs API to generate responses: Claude 3.7, OpenAI o1, DeepSeek-R1, Qwen-Plus, and Llama 3.1. To ensure consistency and reproducibility across models, we standardized the generation parameters as follows:

Temperature: Set to 0.7 for all models to balance creativity and coherence in generated outputs.

Top-p (nucleus sampling): Fixed at 0.95 to maintain a high probability mass while allowing for diverse outputs.

System Prompts: No system prompts were used; all instructions were provided within the agents' prompts to avoid introducing model-specific biases.

These configurations align with recommended settings for models. By maintaining uniform settings across all models, we aimed to ensure a fair comparison and reliable evaluation of their performance.

A.3. Knowledge Graph Construction

Shared Knowledge Infrastructure. Both Task Analysis and Method Design modules rely on a shared hybrid knowledge infrastructure comprising (1) a symbolic knowledge graph that stores structured outputs from agents, and (2) a vector-based retrieval system built on top of Sentence-BERT embeddings and external APIs (PubMed, GitHub). The knowledge graph is incrementally constructed as each agent contributes new findings or insights, while the vector database supports RAG-style retrieval of external literature. This shared infrastructure enables bi-directional communication between agents within each module and supports consistent knowledge propagation across modules. See Appendix A.3 for implementation details.

Collaborative Agents Shared Knowledge Graph in Task Analysis. Instead of operating in isolation, the Dataset Analyst, Problem Investigator, and Baseline Assessor interact via the shared knowledge graph and query interface. Each agent incrementally updates the knowledge graph with its findings, while continuously polling for updates from other agents. For example, once the Dataset Analyst infers perturbation modalities and cell types, the Problem Investigator revises its hypothesis formulation accordingly. Agents operate asynchronously but synchronize their conclusions through a shared JSON-based communication protocol, allowing for self-consistency checks and iterative refinement of the task representation. This collaborative reasoning leads to a structured task analysis report passed to the Method Design module.

Graph-Based Expert Shared Knowledge Graph in Method Design. In the Method Design module, domain experts are instantiated as nodes in a dynamic undirected graph. These expert agents exchange proposals and critiques via message-passing rounds governed by graph neural network operations. Throughout the discussion, the Self-Critic agent monitors logical coherence and suggests refinements. Each expert agent has read-write access to the shared knowledge graph and can retrieve relevant prior knowledge from Agentic Retrieval. Updates to the architectural plan are written back to the graph, enabling history-aware(get messages and suggestions from the former round), convergent model refinement.

A.4. Experts Discussion Construction Details

To enable structured, reproducible reasoning across diverse perturbation modeling tasks, we construct the multi-agent expert discussion system through two key stages: expert role selection and dynamic collaboration graph construction.

Based on the task analysis report, a set of relevant expert agents is selected by matching task attributes against a curated registry of expert types. The selected experts are grouped into five broad categories to ensure comprehensive domain coverage: **(i) Data Engineering and Preprocessing.** A Data Expert is instantiated to address normalization, quality control, feature selection, and batch correction issues tailored to the input modality. **(ii) Model Design and Scalability.** The Model Architecture Expert and Deep Learning Expert are responsible for proposing architectures that balance expressiveness, interpretability, and scalability, considering modality-specific modeling needs. **(iii) Biological Plausibility.** Single Cell Biologists such as the Pathway Analyst, Drug Response Expert, and Omics Modality Expert contribute domain knowledge to align model components with known biological mechanisms, including gene regulatory networks, cytokine signaling, or pharmacodynamics. **(iv) Training and Optimization.** A Training Expert is responsible for selecting and justifying the learning algorithm, optimization strategy, regularization, and validation scheme suitable for the data structure and model complexity. **(v) Self-Critique and Evaluation.** A Self-Critic agent is included in every discussion to promote internal scrutiny, consistency checks, and critical reflection over model assumptions and claims.

For example, in a gene knockout task, the system may instantiate the Data Expert to inspect whether the scRNA-seq matrix is properly normalized, whether cell and gene identifiers are standardized, and whether preprocessing sufficiently preserves perturbation-related variation. The Model Architecture Expert and Deep Learning Expert are instantiated to co-design a gene-centric model that integrates perturbation-aware attention and captures target gene dependent regulatory effects. The Pathway Analyst is instantiated to evaluate the role of target gene within interferon signaling cascades, while the Omics Modality Expert assesses whether transcriptomic changes resulting from target gene ablation are robustly captured by scRNA-seq alone. The Training Expert selects dropout-regularized contrastive training and a cell-type-aware sampling scheme to stabilize optimization. The Statistics Expert designs a differential expressionbased evaluation framework and quantifies the significance of target gene induced shifts using FDR-corrected effect sizes. Finally, the Self-Critic is instantiated to identify overfitting risks in rare knockout subsets, challenge latent space linearity assumptions, and refine model outputs for interpretability and robustness.

All experts are set with role-specific prompts (Appendix F.3), crafted in a zero-shot reasoning format. These prompts condition on the shared Task Analysis report and elicit structured outputs including modeling choices, biological justification,

and critiques of others proposals.

Formally, the expert set $E^{(k)}$ for task k is derived by:

$$E^{(k)} = \texttt{SelectExperts}(\texttt{TaskAnalysisReport}_k)$$

Once instantiated, the experts are organized into an undirected collaboration graph $G^{(k)} = (S, E^{(k)})$, where each node $E^{(i)} \in E^{(k)}$ represents an expert role. The Self-Critic node S is fully connected to all others, serving both as a dialectical evaluator and proposal aggregator.

Each expert begins with an initial model proposal $m_0^{(i)}$ and a confidence score initialized to zero $c_0^{(i)} = 0$. During the discussion, agents iteratively update their proposals and confidence scores through message passing on the graph. Each round incorporates structured information exchange, where agents revise their reasoning in response to input from their neighbors, weighted by relevance.

This structured and interpretable procedure allows SCAGENTS to generate scientifically grounded, multimodally coherent model designs that are not only technically sound but also biologically meaningful.

B. Evaluation Details

This appendix provides detailed formulations of the hierarchical metrics used in our benchmark evaluation of transcriptomics machine learning models for perturbation analysis.

B.1. Mean Squared Error (MSE)

This metric measures the average squared difference between the true and predicted gene expression vectors, quantifying overall prediction error. Let $Y_i, \hat{Y}_i \in \mathbb{R}^{d'}$ be the true and predicted expression vectors for sample i. Then

$$\text{MSE} = \frac{1}{n\,d'} \sum_{i=1}^{n} \|Y_i - \hat{Y}_i\|_2^2.$$

B.2. Pearson Correlation Coefficient (PCC)

This metric assesses the strength of the linear association between predicted and true expression profiles across all samples. Define the sample means

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i, \quad \overline{\hat{Y}} = \frac{1}{n} \sum_{i=1}^{n} \hat{Y}_i.$$

Then

$$\text{PCC} = \frac{\sum_{i=1}^{n} \langle Y_i - \bar{Y},\ \hat{Y}_i - \overline{\hat{Y}} \rangle}{\sqrt{\sum_{i=1}^{n} \|Y_i - \bar{Y}\|_2^2}\ \sqrt{\sum_{i=1}^{n} \|\hat{Y}_i - \overline{\hat{Y}}\|_2^2}}.$$

B.3. Coefficient of Determination (R^2)

This metric quantifies the proportion of variance in the true gene expression data that is captured by the models predictions. It provides an interpretable measure of model fit, with higher values indicating better predictive performance. Let $Y_i, \hat{Y}_i \in \mathbb{R}^{d'}$ be the true and predicted expression vectors for sample i, and let $\bar{Y}$ denote the mean of the true expression vectors. Then

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \|Y_i - \hat{Y}_i\| 2^2}{\sum i = 1^{n} \|Y_i - \bar{Y}\|_2^2}.$$

B.4. Metrics with Differential Expression (DE)

Differential expression highlights the genes whose changes drive the biological response to a perturbation, focusing evaluation on the most informative signals. Let $\{Y_{p,i}\}_{i=1}^{n_p}$ and $\{Y_{c,i}\}_{i=1}^{n_c}$ be the true expression vectors under perturbation and control, respectively, with $Y_{p,i},\ Y_{c,i} \in \mathbb{R}^{d'}$. For each gene $g = 1, \ldots, d'$, compute the mean expression

$$\bar{Y}_{p,g} = \frac{1}{n_p} \sum_{i=1}^{n_p} Y_{p,i,g}, \qquad \bar{Y}_{c,g} = \frac{1}{n_c} \sum_{i=1}^{n_c} Y_{c,i,g}.$$

Quantify the change by the logfoldchange

$$\text{LFC}_g = \log_2 \frac{\bar{Y}_{p,g} + \epsilon}{\bar{Y}_{c,g} + \epsilon},$$

with small $\epsilon > 0$ to avoid division by zero (or alternatively by the raw difference $\Delta_g = \bar{Y}_{p,g} - \bar{Y}_{c,g}$). Rank genes by $|\text{LFC}_g|$ (or $|\Delta_g|$), and select the top $K = 20$ as the DE set:

$$\text{DE} = \{\, g : \text{rank}_g(|\text{LFC}|) \le K \,\}, \qquad K = 20.$$

Subsequent metrics (MSE, PCC, R^2) are then computed only over $g \in \text{DE}$ to assess performance on these key drivers of perturbation response.

B.5. LatentSpace Linear Separability

This metric evaluates if a model's latent space distinguishes between different perturbations using linear probing. Given a frozen encoder mapping $g_\phi : x_i \mapsto z_i \in \mathbb{R}^d$, train a linear classifier

$$\hat{y} = \text{softmax}(Wz + b), \quad W \in \mathbb{R}^{c\times d},\ b \in \mathbb{R}^c,$$

to predict one of c perturbation classes. For n test samples with true labels y_i,

$$\text{Top}{-}1 = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{\arg\max_j \hat{y}_{ij} = y_i\}, \qquad \text{Top}{-}5 = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i \in \text{Top5}(\hat{y}_i)\}.$$

B.6. Perturbation Consistency

This metric assesses the consistency with which a model represents perturbations between different samples and batches. Let $\mathcal{P}$ be the set of all gene perturbations. For each $p \in \mathcal{P}$, suppose we have n_p embedding vectors

$$\{\, z_{p,i} \in \mathbb{R}^d \mid i = 1, \ldots, n_p\}.$$

Define the *mean cosinesimilarity score*

$$S_p = \frac{1}{n_p^2}\sum_{i=1}^{n_p}\sum_{j=1}^{n_p} \frac{\langle z_{p,i},\, z_{p,j}\rangle}{\|z_{p,i}\|\ \|z_{p,j}\|}.$$

Let $\{S_{q_k}\}_{k=1}^{K}$ be the corresponding scores for K unexpressedgene controls q_k. The empirical pvalue for perturbation p is

$$\pi_p = \frac{\max\{\#\{k : S_{q_k} \le S_p\},\, 1\}}{K}.$$

Finally, the overall *consistency rate* is

$$C = \frac{\big|\{\, p \in \mathcal{P} : \pi_p < 0.05\}\big|}{|\mathcal{P}|},$$

i.e. the fraction of perturbations whose embeddings are significantly more selfsimilar than the null.

B.7. Latent Space Direct Organization

This metric evaluates the degree to which perturbation clusters are locally organized in the latent space, using the k-Nearest Neighbors (kNN) classification. Let $\{z_i\}_{i=1}^{n_q}$ and $\{z_j\}_{j=1}^{n_r}$ be the latent embeddings for the query and reference sets, with the corresponding labels y_i and y_j. Set

$$k = \lfloor \sqrt{n_r} \rfloor.$$

For each query index i, let $N_k(i) \subset \{1, \ldots, n_r\}$ be the reference index k whose embeddings minimize $\|z_i - z_j\|_2$. Then the *kNNclassification accuracy* is

$$\text{Accuracy}_{\text{kNN}} = \frac{1}{n_q}\sum_{i=1}^{n_q} \mathbf{1}\Big[y_i = \arg\max_{c\in C} \sum_{j\in N_k(i)} \mathbf{1}[y_j = c]\Big],$$

where C denotes the set of all perturbation labels.

B.8. Linear Interpretability of Latent Space

Let $Z \in \mathbb{R}^{n\times h}$ be the frozen-encoder outputs and train a linear MLP, $\hat{Y} = h(Z) \in \mathbb{R}^{n\times d'}$. We define two metrics: *Spearman correlation* and *structural integrity*.

Spearman Correlation This measures how accurately the latent embeddings can be decoded back into gene expression data using a simple linear transformation. The *Spearman correlation* ρ is defined as

$$\rho = 1 \ - \ \frac{6\sum_{i=1}^{n}\big[\text{rank}(Y_i) - \text{rank}(\hat{Y}_i)\big]^2}{n(n^2-1)},$$

where $\text{rank}(\cdot)$ returns the withinsample rank vector.

Structural Integrity This metric evaluates how well the model preserves the relationship between control and perturbation conditions within each biological batch. For $b = 1, \ldots, B$ batches with n_b samples each, let

$$\widetilde{Y}^{(b)}_{\text{pred}} = Y^{(b)}_{\text{pred}} - Y^{(b)}_{\text{pred,ctrl}}, \quad \widetilde{Y}^{(b)}_{\text{act}} = Y^{(b)}_{\text{act}} - Y^{(b)}_{\text{act,ctrl}}.$$

Then

$$D = \frac{1}{B}\sum_{b=1}^{B} \frac{1}{n_b} \big\| \widetilde{Y}^{(b)}_{\text{pred}} - \widetilde{Y}^{(b)}_{\text{act}} \big\|_F, \quad D_{\max} \approx \frac{2}{B}\sum_{b=1}^{B} \frac{1}{n_b} \big\| \widetilde{Y}^{(b)}_{\text{act}} \big\|_F,$$

and the *structural integrity* is

$$\text{SI} = 1 - \frac{D}{D_{\max}},$$

with higher SI indicating better preservation of controlperturbation structure.

C. RAGBench Evaluation Details

To evaluate the performance of scAgents' Agentic Retrieval system in Task Analysis Module, we employ RAGBech(Friel et al., 2025).

We first align our systems outputs to the format expected by RAGBench. Each output record must include:

- `id`: unique sample identifier;
- `documents`: list of retrieved context documents;
- `question`: the query text;
- `response`: the generated answer.

Refer to `constants.py` in the RAGBench repository for exact field definitions to ensure full compatibility. Then we run inference on our system's outputs with evaluation models Trulens and dataset PubMedQA(Jin et al., 2019).

Detailed formulations of the metrics used in this RAGBench benchmark are as follow:

Hallucination Detection (Hal)

In Retrieval-Augmented Generation (RAG) systems, *hallucination* refers to the generation of content not grounded in the retrieved contextin other words, the model makes up facts. Hallucination detection measures whether the models outputs contain such unsupported information.

Reliable RAG outputs demand faithfulness to the provided context. Evaluating hallucination detection quantifies the systems propensity to stray from source documents, informing improvements to retrieval, grounding, and decoding strategies.

We adopt the Area Under the Receiver Operating Characteristic Curve (AUROC) to quantify hallucination detection performance. Given:

$$\text{trues}_{\text{adherence}} \in \{\texttt{True}, \texttt{False}\}, \quad \text{preds}_{\text{adherence}} \in [0, 1],$$

we define hallucination labels by

$$\text{trues}_{\text{halluc}} = \neg\, \text{trues}_{\text{adherence}}, \quad \text{preds}_{\text{halluc}} = 1 - \text{preds}_{\text{adherence}}.$$

Let

$$\text{mask} = \neg\, \text{isnan}(\text{preds}_{\text{halluc}}).$$

Then

$$\text{AUROC} = \text{ROC\_AUC}\Big(\text{trues}_{\text{halluc}}[\text{mask}], \text{preds}_{\text{halluc}}[\text{mask}]\Big),$$

where ROC_AUC denotes the standard implementation (`sklearn.metrics.roc_auc_score`).

Context Relevance (Rel)

Context relevance assesses how well the retrieved documents pertain to the query, i.e. whether the context can support a correct answer.

High relevance is a prerequisite for accurate generation. Measuring context relevance guides retrieval improvements and ensures that the generator receives useful evidence.

We measure relevance via Root Mean Squared Error (RMSE) between true and predicted relevance scores:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{y}_i\big)^2},$$

where y_i is the gold relevance score, $\hat{y}_i$ the predicted score, and n the number of examples. We ignore any NaN predictions by masking.

Context Utilization (Utl)

Context utilization evaluates the extent to which the model leverages the retrieved context when generating its responses.

Even with relevant context, a model may underuse it. This metric reveals the generators ability to integrate context information into its output.

We again employ RMSE, defined as above, to compare true and predicted utilization scores, masking out NaN predictions.

Together, *Hal*, *Rel*, and *Utl* provide a multi-faceted evaluation of RAG system performance: detecting hallucinations, ensuring context relevance, and confirming effective context usage.

By following the above steps and using the provided evaluation metrics, we can comprehensively evaluate our retrieval - augmented generation (RAG) system using the RAGBench framework.

D. Detailed Algorithm Specifications

D.1. Agentic Retrieval System

The agentic retrieval system combines both static knowledge integration and dynamic search capabilities to provide comprehensive scientific context for perturbation analysis tasks. Here we provide the complete algorithmic details of our implementation.

D.1.1. Query Construction and Initialization

Given a task description T and dataset metadata D, we first construct an initial query representation:

Algorithm 1 Query Construction

1. **input** T, D
2. **set** keywords $\leftarrow$ ExtractKeyTerms$(T) \cup$ ExtractKeyTerms(D)
3. **set** embedding $\leftarrow$ SentenceBERT(keywords)
4. **set** $Q^{(0)} \leftarrow$ NormalizeVector(embedding)
5. **output** $Q^{(0)}$

The function ExtractKeyTerms performs domain-specific extraction of biological entities (genes, cell types, perturbation methods) and technical terms (model architectures, evaluation metrics) using named entity recognition enhanced with domain-specific dictionaries.

D.1.2. Alternating Search Strategy

Our multi-layer retrieval process alternates between breadth-first and depth-first search modes to balance exploration and exploitation:

Algorithm 2 Alternating BFS-DFS Retrieval

1. **input** $Q^{(0)}, L_{\max}, \tau, \epsilon$
2. **set** $t \leftarrow 0$
3. **set** $\mathcal{N}_0 \leftarrow \emptyset$
4. **set** $\mathcal{D} \leftarrow \emptyset$ {Document collection}
5. **while** $t < L_{\max}$ **do**
   (a) **if** $t \bmod 2 = 1$ **then** {BFS layer (odd t)}
      i. **set** $\mathcal{N}_t \leftarrow$ TopK$(Q^{(t)}, \text{mode} = \text{BFS})$
   (b) **else** {DFS layer (even t)}
      i. **set** $\mathcal{N}_t \leftarrow$ FollowCitations$(\mathcal{N}_{t-1})$
   (c) **set** $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{N}_t$
   (d) **set** $Q^{(t+1)} \leftarrow$ UpdateQuery$(Q^{(t)}, \mathcal{N}_t)$
   (e) **if** Overlap$(Q^{(t+1)}, Q^{(t)}) > \tau$ **then break**
   (f) **if** $\max_{d \in \mathcal{N}_t}$ Score$(Q^{(t)}, d) < \epsilon$ **then break**
   (g) **set** $t \leftarrow t + 1$
6. **output** $\mathcal{D}$

Relevance Scoring. The document relevance function uses cosine similarity in the embedding space:

$$\text{Score}(Q, d) = \frac{e(Q) \cdot e(d)}{\|e(Q)\|\|e(d)\|} \tag{1}$$

where $e(\cdot)$ is the Sentence-BERT encoder function mapping text to dense vectors.

Query Update Mechanism. The query update function incorporates new information while maintaining focus:

$$Q^{(t+1)} = \alpha Q^{(t)} + (1 - \alpha) \frac{1}{|\mathcal{N}_t|} \sum_{d \in \mathcal{N}_t} e(d) \tag{2}$$

where $\alpha = 0.7$ is a parameter controlling the balance between query persistence and adaptation.

Overlap Computation. Query overlap is calculated as:

$$\text{Overlap}(Q^{(t+1)}, Q^{(t)}) = \frac{|Q^{(t+1)} \cap Q^{(t)}|}{\min(|Q^{(t+1)}|, |Q^{(t)}|)} \tag{3}$$

where the intersection operation is implemented using a thresholded similarity measure in the embedding space.

D.2. Graph-based Multi-Expert Discussion

The Method Design module employs a graph-based discussion framework where experts collaboratively refine scientific hypotheses. Here we formalize the complete algorithm:

Algorithm 3 Graph-based Expert Discussion

1. **input** TaskAnalysis, $\tau, \epsilon, T_{\max}$
2. **set** $E^{(k)} \leftarrow$ SelectExperts(TaskAnalysis)
3. **set** $S \leftarrow$ InitializeSelfCritic()
4. **set** $G^{(k)} \leftarrow (S, E^{(k)})$ {Initialize collaboration graph}
5. **for** $i = 1$ **to** k **do**
   (a) **set** $c_0^{(i)} \leftarrow 0$ {Initialize confidence scores}
   (b) **set** $m_0^{(i)} \leftarrow$ InitialProposal($E^{(i)}$, TaskAnalysis)
6. **set** $t \leftarrow 0$
7. **while** $t < T_{\max}$ **do**
   (a) **for** $i = 1$ **to** k **do**
      i. **set** $m_t \leftarrow$ Integrate($\{m_t^{(j)}\}_{j=1}^{k}$)
      ii. **set** $c_t^{(i)} \leftarrow$ UpdateConfidence($c_{t-1}^{(i)}, m_t, S$)
   (b) **if** $\forall i : c_t^{(i)} \geq \tau \wedge \|c_t^{(i)} - c_{t-1}^{(i)}\| < \epsilon$ **then break**
   (c) **for** $i = 1$ **to** k **do**
      i. **set** $m_{t+1}^{(i)} \leftarrow$ RefineIdea($E^{(i)}, m_t$)
   (d) **set** $t \leftarrow t + 1$
8. **set** ResearchPlan $\leftarrow$ FinalizeProposal(m_t)
9. **output** ResearchPlan

Expert Selection. The expert selection procedure dynamically assembles a team of domain specialists based on task requirements:

$$P(E^{(i)}|\text{TaskAnalysis}) \propto \exp(\beta \cdot \text{Relevance}(E^{(i)}, \text{TaskAnalysis})) \tag{4}$$

where β is a temperature parameter controlling selection diversity.

Confidence Update Rule. The confidence score update incorporates feedback from both the Self-Critic and other experts:

$$c_t^{(i)} = \lambda_1 \cdot c_{t-1}^{(i)} + \lambda_2 \cdot \text{SelfCriticScore}(m_t^{(i)}, S) + \lambda_3 \cdot \frac{1}{k-1} \sum_{j \neq i} \text{PeerScore}(m_t^{(i)}, E^{(j)}) \tag{5}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$ weights the relative importance of each component.

Message Integration. Expert proposals are integrated through a weighted combination:

$$m_t = \sum_{i=1}^{k} w_t^{(i)} \cdot m_t^{(i)} \tag{6}$$

where weights $w_t^{(i)}$ are derived from normalized confidence scores:

$$w_t^{(i)} = \frac{\exp(c_t^{(i)})}{\sum_{j=1}^{k} \exp(c_t^{(j)})} \tag{7}$$

This soft-voting mechanism ensures that higher-confidence perspectives have greater influence while still preserving diversity of thought.

D.3. Validation and Refinement Process

The Validation Agent employs an iterative refinement process that systematically improves implementation quality:

Algorithm 4 Iterative Implementation Refinement

1. **input** ModelDesign, Dataset, $R_{\max}$
2. **set** $\text{Code}_0 \leftarrow \text{InitialImplementation}(\text{ModelDesign})$
3. **set** $\text{Performance}_0 \leftarrow \text{Evaluate}(\text{Code}_0, \text{Dataset})$
4. **for** $r = 1$ **to** $R_{\max}$ **do**
   (a) **set** $\text{Errors}_r \leftarrow \text{IdentifyIssues}(\text{Code}_{r-1}, \text{Performance}_{r-1})$
   (b) **set** $\text{Code}_r \leftarrow \text{RefineImplementation}(\text{Code}_{r-1}, \text{Errors}_r)$
   (c) **set** $\text{Performance}_r \leftarrow \text{Evaluate}(\text{Code}_r, \text{Dataset})$
   (d) **if** $\text{Performance}_r - \text{Performance}_{r-1} < \delta$ **then break**
5. **output** Code_r

Error Analysis. The error identification procedure categorizes implementation issues into distinct types:

- **Logical errors:** Incorrect algorithm implementation

- **Numerical instability:** Gradient explosion/vanishing
- **Memory inefficiency:** Excessive resource consumption
- **Performance bottlenecks:** Suboptimal computational paths
- **Biological implausibility:** Violations of domain constraints

Each error type triggers specialized refinement strategies that preserve the scientific integrity of the model design while improving implementation quality. Detailed Failure case analysis is presented in Appendix N.

D.4. Hyperparameter Configuration

Our framework employs the following hyperparameter settings, determined through empirical validation on held-out scientific tasks:

Table 4: Hyperparameter Configuration

Module	Parameter	Value
Agentic Retrieval	$L_{\max}$	10
	τ	0.8
	ϵ	0.5
Expert Discussion	$T_{\max}$	6
	τ	0.8
	ϵ	1
	$(\lambda_1, \lambda_2, \lambda_3)$	(0.3, 0.4, 0.3)
Validation	$R_{\max}$	5
	δ	0.01

These parameters balance convergence speed with solution quality across different perturbation types and dataset characteristics. We observed that the Expert Discussion module particularly benefits from a higher weight on Self-Critic evaluation (λ_2), which promotes more rigorous scientific validation.

Table 5: Performance and Cost Impact of Hyperparameter Settings

Module	Setting	Metric	Score	Change	Time Cost (s)	Token ×
Agentic Retrieval	$\epsilon = 0.3$	Relevance Score	74.5		14.1	1.0×
	$\epsilon = 0.5$	Relevance Score	75.1	(0.6)	28.3	1.6×
Expert Discussion	$\lambda_2 = 0.2$	Validity Score	69.1		25.2	3.5×
	$\lambda_2 = 0.4$	Validity Score	**73.4**	(4.3)	27.0	4.2×
	$\lambda_2 = 0.6$	Validity Score	72.2		28.5	4.6×
	$\lambda_2 = 0.4$	Consistency Score	**76.1**	(5.9)	17.0	4.2×
Validation	$\delta = 0.02$	Acceptance Rate	82.1		3.2	1.0×
	$\delta = 0.01$	Acceptance Rate	81.7		2.9	0.7×

E. Agent Communication Protocol Details

E.1. Protocol Design and Comparison

The SCAGENTS protocol represents an advancement in agent communication architectures designed specifically for scientific discovery. Figure 3 illustrates the multi-stage protocol that facilitates information exchange across the three core phases of our framework.

The protocol weaves together the strengths of several prior designs. It preserves the interoperability of JSON-RPC for rapid agent deployment and cross-platform compatibility while simultaneously extending this foundation with semantic connectivity and provenance via the knowledge graph. It not only connects software components, but also enables the kind of iterative, multi-agent reasoning on which genuine discovery depends. The scAgents' protocol method allows agents to coordinate autonomously when tasked with comprehensive scientific research.

Table 6 provides a detailed comparison of SCAGENTS with existing agent communication protocols. Unlike previous approaches that excel in limited domains, our protocol uniquely combines contextual awareness, cross-platform interoperability, and knowledge representation capabilities necessary for end-to-end scientific discovery.

Table 6: **Comparison of Agent Communication Protocols**

Protocol	Context	Interop.	Msg. Struct.	Use Cases
MCP (Anthropic)	✓	✗	JSON-RPC only	Tool Use & Data Access
Agent2Agent (Google)	✗	✓	JSON-RPC event	Cross-agent Collaboration
ACP (BeeAI/IBM)	✓	✓	RESTful	Local Orchestration
scAgents	✓	✓	JSON-RPC event + Knowledge Graph	End-to-end Scientific Discovery

E.2. Protocol Implementation Details

The SCAGENTS protocol implementation consists of two primary components:

JSON-RPC Communication Layer This provides standardized message passing between agents, with extensions for asynchronous event handling. Each agent exposes a consistent API that accepts and returns structured data, enabling precise coordination of complex workflows.

Knowledge Graph Integration Layer Beyond simple message passing, SCAGENTS maintains a persistent knowledge graph that captures:

- Research entities (datasets, methods, metrics, results)
- Relationships between entities (causal, hierarchical, temporal)
- Provenance information (confidence scores, reasoning chains, citations)
- Domain-specific knowledge (regulatory pathway information, gene-gene interactions)

This dual-layer approach provides several advantages compared to prior protocols:

1. **Context-awareness**: Agents maintain awareness of the overall research state through the knowledge graph, enabling them to make more informed decisions.
2. **Traceability**: The entire scientific process is captured with provenance information, ensuring reproducibility.
3. **Semantic reasoning**: Relationships between scientific concepts are explicitly modeled, enabling complex inferential reasoning.
4. **Incremental refinement**: The persistent knowledge representation allows agents to build upon previous insights and progressively refine hypotheses.

In scientific research contexts, these capabilities are essential for managing the complexity of cross-disciplinary knowledge integration required for tasks like single-cell perturbation analysis.

F. Prompt Templates

F.1. Task Description input

Task Desciption Input

```
Your task is to develop a predictive model that accurately estimates gene
    expression profiles of individual K562 cells following CRISPR interference (
    CRISPRi), using the dataset from Norman et al. (2019, Science).

Task Definition:
- Input: Baseline gene expression profile of an unperturbed K562 cell and the
    identity of the target gene(s) for perturbation
- Output: Predicted gene expression profile after perturbation


Evaluation Scenarios:
1. Unseen Perturbations: Predict effects of gene perturbations not present during
     training
2. Unseen Cell Contexts: Predict responses in cells with gene expression profiles
     not observed during training

Evaluation Metrics:
- Mean Squared Error (MSE): Measures the average squared difference between
    predicted and observed gene expression.
- Pearson Correlation Coefficient (PCC): Quantifies linear correlation between
    predicted and observed profiles.
- R$^2$ (Coefficient of Determination): Represents the proportion of variance in
    the observed gene expression that can be explained by the predicted values.
- MSE for Differentially Expressed (DE) Genes (MSE_DE): Same as MSE but computed
    specifically for genes identified as differentially expressed.
- PCC for Differentially Expressed (DE) Genes (PCC_DE): Same as PCC but computed
    specifically for genes identified as differentially expressed.
- R$^2$ for Differentially Expressed (DE) Genes (R$^2$_DE): Same as R$^2$ but
    computed specifically for genes identified as differentially expressed.
```

F.2. Task Analysis Collaboration Agents Settings

Agent 1: Dataset Analyst Dataset Analyst is a specialized agent responsible for performing systematic analysis of single-cell perturbation datasets during the Task Analysis stage. Its core function is to extract and summarize the key characteristics of a given datasetincluding experimental design, data modalities, perturbation types, and quality metrics to facilitate downstream hypothesis generation and modeling. The agent is equipped with contextualized agent retrieval (RAG in former module) results to incorporate relevant metadata, associated publications, and protocol references. Its output follows a structured JSON format to enable direct inter-agent communication and automatic pipeline integration. The Dataset Analyst prioritizes clarity, scientific precision, and critical evaluation, identifying potential risks or biases while offering preprocessing recommendations tailored to the dataset's complexity and intended use cases.

Task Analysis Collaboration Agent Settings

Data Analyst

```
You are a Dataset Analyst agent in a multi-agent scientific research system. Your
     goal is to analyze a single-cell perturbation dataset and provide a
    comprehensive, structured, and insightful report to support downstream
    hypothesis generation and modeling.

# Role Description:
The Dataset Analyst is responsible for extracting structured knowledge from
    single-cell perturbation datasets. This includes characterizing the
    experimental design, identifying quality and completeness issues, and
    proposing dataset-specific modeling considerations. The agent draws upon both
     metadata and relevant scientific context retrieved via agentic tools. It
    supports hypothesis generation by clarifying what is measurable, what may be
    confounded, and what preprocessing steps are necessary.

# Skills:
- Interpreting single-cell multi-omics data structures (e.g., RNA, ATAC, protein)
- Identifying perturbation types and their downstream modeling implications
- Detecting quality control issues (e.g., batch effects, sparsity, missing
    modalities)
- Integrating metadata from publications, protocols, and retrieved scientific
    sources
- Structuring heterogeneous information into JSON-compatible schemas
- Making biologically grounded recommendations for preprocessing and modeling

# Objectives:
- Extract and summarize the key characteristics of the dataset
- Identify risks, limitations, and preprocessing needs
- Suggest modeling strategies aligned with dataset structure
- Provide additional scientific insights not limited to a fixed template

# Input:
You will receive:
1. Dataset metadata (e.g., species, cell types, perturbation types)
2. relevant knowledge from agentic retrieval

# Instructions:
Produce a two-part output:

## Part 1: Structured Summary (JSON format)

Include the following fields, but you may expand or adapt them based on dataset
    complexity:

{
  "introduction": {
    "modalities": [...],
    "perturbation_type": [...],
    "conditions": [...],
    "timepoints": [...],
    "replicates": true/false,
    "batches": true/false,
    "cell_types": [...],
    "organism": "...",
    "description": "..."
```

```
  },
  "data_properties": {
    "num_cells": [...],
    "num_genes": [...],
    "num_features": {
      "RNA": ..., "ATAC": ..., "protein": ...
    },
    "perturbation_targets": {
      "num_unique": [...],
      "target_type": [...],
      "coverage": "dense/sparse/mixed"
    },
    "modality_completeness": [...],
    "metadata_completeness": [...],
    "preprocessing_required": [...]
  },
  "quality_assessment": {
    "data_sparsity": [...],
    "batch_effect": [...],
    "replicate_consistency": [...],
    "known_issues": [...],
    "strengths": [...],
    "limitations": [...]
  },
  "recommendations": {
    "preprocessing_steps": [...],
    "modeling_considerations": [...],
    "open_questions": [...]
  },
  "refinement_suggestions": [...]
}

Write a concise, scientifically sound narrative (~150300 words) to accompany the
    JSON summary. This should include:
- A holistic interpretation of dataset readiness for modeling
- Potential scientific pitfalls or confounders
- Unique strengths or opportunities (e.g., rare perturbation types, rich time
    series)
- Reflections on whether the dataset aligns with typical assumptions in modeling
    pipelines
- Any useful observations that do not fit cleanly into the structured fields

You may reference information from retrieved publications or external protocols.
    If information is unknown or ambiguous, state it clearly using cautious
    language (e.g., "not reported", "likely sparse", "appears to have").

# Constraints:
- Be concise, accurate, and avoid redundancy
- Use clear scientific language; bullet points are acceptable in the JSON
- Allow flexibility in output structure if additional insights emerge

# Style Guide:
- Output should be compatible with integration into downstream agent pipelines
- Aim for the clarity and precision expected in a peer-reviewed supplementary
    method section
- Prioritize information relevant to perturbation modeling, multi-omic
    integration, and biological interpretation
```

Agent 2: Problem Investigator The Problem Investigator is a domain-specialized agent responsible for transforming the input dataset and task context into a clearly defined scientific problem formulation. This agent operates at the interface between biological insight and computational design, aiming to decompose complex single-cell perturbation tasks into actionable research questions, computational objectives, and biologically meaningful evaluation strategies. Leveraging both LLM reasoning and agentic retrival results, the agent integrates biological mechanisms, and relevant literature to propose testable hypotheses, identify key challenges, and design analysis methods with biological and computational validity.

Task Analysis Collaboration Agent Settings

Probelm Investigator

```
You are a Problem Investigator agent in a multi-agent scientific research system.
     Your goal is to transform the dataset analysis into a scientifically
    meaningful and computationally tractable hypothesis or modeling plan.

# Role Description:
The Problem Investigator interprets dataset summaries and transforms them into
    well-scoped scientific problems. This includes identifying biologically
    significant questions, selecting meaningful targets or outcomes, and
    proposing hypotheses that can be tested using computational modeling. The
    agent must balance biological relevance, data availability, and
    methodological feasibility. It serves as a bridge between raw data and
    actionable research direction.

# Skills:
- Formulating biologically meaningful and testable hypotheses from complex data
- Mapping experimental designs to machine learning problem types (e.g.,
    classification, regression)
- Evaluating feasibility of predictive tasks based on data modality and
    perturbation scope
- Identifying pitfalls such as confounding, data leakage, or unobservable targets
- Specifying input-output pairs and validation schemes for modeling tasks
- Justifying scientific value and downstream utility of proposed tasks

# Objectives:
- Translate dataset structure into concrete scientific questions
- Identify feasible targets, tasks, and outputs for modeling
- Justify the biological and computational value of the proposed formulation
- Propose a structured hypothesis or modeling objective for downstream agents

# Input:
You will receive:
1. Dataset summary from the Dataset Analyst (structured + narrative)
2. Relevant biological context via retrieval or user input

# Instructions:
Produce a structured problem specification with the following components:

{
  "biological_question": "string",
  "hypothesis_statement": "string",
  "task_formulation": {
    "input": ["modality1", "metadata1", "..."],
    "output": "target variable or prediction goal",
    "task_type": "regression/classification/generation/other"
  },
  "justification": {
    "biological_relevance": [...],
    "data_suitability": [...],
    "expected_challenges": [...]
  },
  "evaluation_plan": {
    "metrics": [...],
    "baselines_to_consider": [...],
    "validation_strategy": "cross-validation/held-out cells/time-split/..."
  },
  "open_questions": [...]
}

In addition to the structured JSON, write a short explanation (100250 words) that
    :
- Restates the goal in accessible scientific language
- Explains why the proposed formulation is worth pursuing
- Anticipates possible modeling limitations or edge cases
- Optionally suggests alternatives or extensions

# Constraints:
- Prioritize alignment with the datasets structure and perturbation resolution
- Avoid overly generic formulations; focus on specificity and tractability
```

```
- Maintain scientific rigor and make testable claims when possible

# Style Guide:
- Write in the tone of a proposal for a computational biology modeling section
- Use precise language grounded in both biology and data science
- Be mindful of what is *not* observable or predictable from the dataset
```

Agent 3: Baseline Assessor The Baseline Assessor is a methodological analyst agent tasked with selecting, evaluating, and recommending baseline models for single-cell perturbation studies. Operating at the intersection of computational rigor and biological relevance, this agent critically assesses modeling paradigms across task types (e.g., regression, classification, generative modeling) and data modalities (e.g., gene expression, ATAC-seq, protein levels). It integrates literature evidence, benchmark practices, and dataset-specific constraints to recommend flexible yet strong baseline approaches. The agent also incorporates multi-objective considerations such as performance, interpretability, scalability, and biological plausibility.

Task Analysis Collaboration Agent Settings

Baseline Assessor

```
You are a Baseline Assessor agent specialized in recommending suitable baseline
    models for single cell perturbation prediction tasks. Your goal is to provide
     comprehensive assessments of baseline models and evaluation strategies based
     on relevant literature and dataset characteristics.
# Role Description:
The Baseline Assessor is a comparative modeling expert focused on identifying and
     analyzing baseline architectures for perturbation prediction. This includes
    reviewing existing literature, extracting methodological details, and
    evaluating model suitability based on dataset constraints and task objectives.
     The agent serves as a bridge between literature insights and practical
    modeling recommendations.
# Skills:
- Assessing baseline models for biological interpretability and computational
    requirements
- Identifying candidate architectures relevant to perturbation types and
    modalities
- Extracting methodological details and limitations from scientific literature
- Comparing model performances across different biological contexts
- Designing evaluation frameworks with biologically significant metrics
- Providing actionable improvement suggestions based on technical and biological
    considerations
# Objectives:
- Review relevant literature for the given perturbation type and modality
- Identify 35 candidate architectures and discuss their pros/cons in this context
- Recommend at least two baseline models with rationale aligned to the dataset
    constraints and task objectives
- Design evaluation frameworks considering biological variability and technical
    limitations
- Provide improvement suggestions for model enhancements and biological
    validation
# Input:
You will receive:
Task description and dataset information from upstream agents
Retrieved papers and code implementations from literature databases
RAG system results including relevant papers and code snippets
# Instructions:
Produce a structured analysis report with the following components:
{
"literature_overview": {
"perturbation_types": [...],
"existing_methods": [...],
"technical_trends": [...]
},
"candidate_models": [
{
"model_name": "string",
"architecture": "string",
"strengths": [...],
```

```
"weaknesses": [...],
"biological_applicability": [...]
}
],
"recommended_baselines": [
{
"model_name": "string",
"rationale": "string",
"implementation_details": "string",
"evaluation_metrics": [...],
"biological_relevance": [...]
}
],
"evaluation_framework": {
"primary_metrics": [...],
"secondary_metrics": [...],
"validation_strategy": "string",
"test_scenarios": [...]
},
"improvement_suggestions": {
"technical": [...],
"biological": [...],
"computational": [...]
}
}

In addition to the structured JSON, write a short explanation (100250 words) that
    :
Summarizes the assessment of baseline models
Explains the rationale behind the recommended baselines
Discusses the biological relevance of the evaluation framework
Anticipates potential limitations in model interpretability
Suggests practical improvements for future modeling work

# Constraints:
- Prioritize models with established track records in similar biological contexts
- Computational requirements relative to dataset scale
- Ensure recommended models balance biological interpretability and predictive
    performance
- Align evaluation metrics with both technical accuracy and biological
    significance
- Provide concrete implementation details for recommended baselines

# Style Guide:
- Write in the tone of a modeling methodology section for a computational biology
     paper
- Use precise language describing both technical and biological considerations
- a focus on practical applicability while acknowledging theoretical limitations
- Clearly distinguish between established knowledge and speculative improvements
- Format model recommendations to facilitate direct implementation
```

Agent 4: Critic Refinement The Critic Refinement Agent orchestrates the integration of outputs from domain-specialized agents into a coherent and machine-actionable analysis plan. It ensures consistency across the Dataset Analysts data characterization, the Problem Investigators hypothesis formulation, and the Baseline Assessors methodological recommendations. By resolving redundancies, aligning formats, and verifying logical flow, the Refinement Agent synthesizes the findings into a structured JSON schema. This agent balances standardization with flexibility, enabling downstream automation while preserving the scientific rationale of each module.

Task Analysis Collaboration Agent Settings

Critic Refinement

```
You are the Refinement Agent in a multi-agent scientific research system. Your
    goal is to consolidate and refine outputs from the Dataset Analyst, Problem
    Investigator, and Baseline Assessor agents into a unified, actionable
    analysis.
# Role Description:
```

```
The Refinement Agent is a meta-agent focused on cross-validating consistency
    across different analytical components. This includes resolving
    contradictions, aligning terminology, and ensuring biological relevance and
    technical feasibility of proposed models and evaluation frameworks. The agent
     serves as the integration point for all upstream analyses.
# Skills:
Cross-validating consistency across different analytical components
Resolving contradictions and aligning terminology
Evaluating biological relevance and technical feasibility
Structuring outputs into unified JSON schemas
Generating comprehensive refinement comments
Providing actionable improvement suggestions
# Objectives:
Ensure consistency in terminologies and constraints across all outputs
Align problem definitions with model assumptions
Reorganize content into clean JSON schemas suitable for automated use
Validate biological and technical coherence of the integrated analysis
Provide final recommendations balancing biological relevance and technical
    feasibility
# Input:
You will receive:
Analysis results from Dataset Analyst, Problem Investigator, and Baseline
    Assessor
Refinement comments from previous iterations
RAG system results including relevant papers and code snippets
Instructions:
Produce a refined analysis report with the following components:
{
"summary": {
"biological_context": "string",
"technical_requirements": "string",
"refinement_overview": "string"
},
"task_definition": {
"input_modalities": [...],
"output_targets": "string",
"task_type": "regression/classification/generation/other",
"biological_significance": "string"
},
"baseline_models": {
"recommended_models": [...],
"model_comparisons": [...],
"implementation_details": "string"
},
"constraints": {
"dataset_limitations": [...],
"technical_constraints": [...],
"biological_constraints": [...]
},
"evaluation": {
"primary_metrics": [...],
"secondary_metrics": [...],
"validation_strategy": "string",
"test_scenarios": [...]
}
}
In addition to the structured JSON, write a short explanation (100250 words) that
    :
Summarizes the refinement process and key adjustments made
Explains how the integrated analysis addresses biological and technical
    requirements
Discusses remaining challenges or limitations
Suggests potential extensions or future work
# Constraints:
Maintain consistency in terminology across all components
Ensure alignment between problem formulation and model capabilities
Validate that evaluation metrics reflect both technical accuracy and biological
    significance
Provide concrete implementation details for recommended approaches
Format outputs to facilitate direct use in downstream architecture design
```

```
# Style Guide:
Write in the tone of a methods integration section for a computational biology
    paper
Use precise language describing both technical and biological considerations
Clearly distinguish between established knowledge and speculative improvements
Format recommendations to facilitate direct implementation
Include both biological validation strategies and technical validation methods
```

F.3. Multi-experts Settings

Data Expert

Expert Role Setting

Data Expert

```
You are acting as a **Data Engineer** in a multi-agent research critique system.
    Your task is to evaluate the provided dataset and experimental setup from a
    data engineering and infrastructure perspective.

You will receive a task analysis report that includes:
- A summary of a single-cell perturbation dataset (in structured or free-text
    form).
- The task formulation and its corresponding prediction targets.
- Metadata schemas and any available preprocessing or encoding steps.
- Optional: access to inferred feature matrices, cell/gene count distributions,
    or batch annotations.

Your objectives:
1. Assess Data Integrity and Format
  - Are the cell and gene identifiers standardized and consistently used?
  - Is the perturbation metadata properly aligned and encoded?
  - Are there signs of data leakage, missing values, or corrupted entries?

2. Evaluate Preprocessing Pipelines
  - Comment on normalization, batch correction, filtering, and feature selection
      steps.
  - Are the preprocessing steps appropriate for downstream modeling?

3. Assess Data Scalability and Efficiency
  - Is the dataset efficiently stored and structured (e.g., sparse matrix, HDF5)?
  - Can it be easily integrated with common ML frameworks (e.g., PyTorch,
      TensorFlow, scikit-learn)?
  - Are large-scale operations (sampling, merging, batching) feasible?

4. Suggest Improvements or Optimizations
  - Recommend preprocessing adjustments, format conversions, or data storage
      alternatives.
  - Point out any engineering bottlenecks that might affect reproducibility or
      scalability.
```

Model Architecture Expert

Expert Role Setting

Model Architecture Expert

```
You are acting as a Model Architecture Expert in a multi-agent research critique
    system. Your task is to analyze and optimize the structural design of the
    proposed model.

You will receive a task analysis report that includes:
- Task specification (input type, target prediction, expected invariances).
- A baseline model description or proposed architecture diagram.
- echnical constraints (e.g., compute, latency, interpretability).

Your objectives:
1. Deconstruct Architectural Choices
   - Analyze core design (e.g., encoder-decoder, attention, residuals).
   - Is the architecture aligned with inductive priors from the data/task?
2. Evaluate Module Interactions
   - Are modality fusions or skip connections implemented properly?
   - Are graph structures or latent bottlenecks justified?
3. Spot Redundancies or Inefficiencies
   - Are there unnecessary layers, repeated computations, or excessive parameters
      ?
4. Propose Optimized Designs
   - Recommend improved architecture patterns.
   - Suggest changes that enhance expressivity, stability, or efficiency.
```

Deep Learning Expert

Expert Role Setting

Deep Learning Expert

```
You are acting as a Deep Learning Expert in a multi-agent research critique
    system. Your task is to evaluate the model's design, scalability, and
    suitability for learning from high-dimensional single-cell data.

You will receive a task analysis report that includes:
- Input-output schema of the learning task (e.g., input modalities, targets,
    sample size).
- Model class (e.g., MLP, Transformer, VAE, GNN) and architecture sketch.
- Training setup including loss functions and evaluation metrics.

Your objectives:
1. Evaluate Model Suitability
   - Is the model architecture appropriate for the data type and task complexity?
   - Does it support integration across modalities or time points?
2. Assess Scalability and Inductive Bias
   - Can the model scale with data size and sparse inputs?
   - Does it exploit structure in the data (e.g., gene graphs, batch embeddings)?
3. Identify Training Bottlenecks or Risks
   - Is overfitting likely due to low data:parameter ratio?
   - Are optimization challenges (e.g., vanishing gradients, instability)
       addressed?
4. Recommend Enhancements
   - Suggest architecture variants (e.g., regularization, pretraining, latent
       modeling).
   - Propose alternative loss designs or data augmentations
```

Training Expert

Expert Role Setting

Training Expert

```
You are acting as a Training Expert in a multi-agent research critique system.
    Your role is to critically evaluate the training strategy and optimization
    pipeline.

You will receive a task analysis report that includes:
- Model structure and parameter count.
- Training procedure (e.g., optimizer, learning rate, batch size, scheduler).
- Regularization strategies and data augmentation steps.

Your objectives:
1. Analyze Optimization Pipeline
   - Are optimizers and learning rates well-tuned for the model/task?
   - Is gradient clipping or scheduler use justified?
2. Evaluate Regularization and Overfitting Risks
   - Are dropout, weight decay, or early stopping applied effectively?
   - Is data augmentation sufficient and biologically reasonable?
3. Diagnose Training Stability
   - Any signs of mode collapse, oscillation, or vanishing gradients?
4. Recommend Training Enhancements
   - Suggest better optimizers, learning rate schedules, or initialization
       schemes.
   - Propose curriculum learning or contrastive pretraining if beneficial.
```

Drug Response Expert

Expert Role Setting

Drug Response Expert

```
You are acting as a Drug Response Expert in a multi-agent scientific design
    system. Your role is to assess the biological and pharmacological feasibility
     of drug perturbation modeling based on the provided single-cell dataset and
```

```
    task formulation.

  You will receivea task analysis report that includes:
  - A summary of the perturbation dataset, including drug names, target pathways,
    dosage, and timing.
  - The biological context (e.g., cell types, disease states, assay modality).
  - Task objective and prediction targets.

  Your objectives:
  1. Evaluate Drug Perturbation Validity
     - Are the drugs applied at biologically relevant concentrations and durations?
     - Are the perturbations expected to have a measurable effect at the single-
       cell level?
     - Are there known resistance mechanisms or compensatory pathways?
  2. Assess Target Coverage and Specificity
     - Do the drug targets align with the measured omics modality (e.g., RNA for
       transcriptional drugs)?
     - Are off-target effects likely to interfere with interpretation?
  3. Recommend Improvements or Adjustments
     - Suggest better dosage choices or controls.
     - Recommend alternative compounds or combinations that better elicit the
       intended perturbation.
```

Pathway Analyst

Expert Role Setting

Pathway Analyst

```
  You are acting as a Pathway Analyst in a multi-agent biological reasoning system.
      Your role is to evaluate the alignment between experimental perturbations (e.
    g., gene knockout or cytokine induction) and known biological signaling
    pathways.

  You will receive a task analysis report that includes:
  - A summary of perturbation targets (genes or cytokines).
  - Downstream measurements (e.g., RNA, ATAC, surface proteins).
  - Known pathway annotations or inferred gene modules (optional).

  Your objectives:
  1. Assess Biological Plausibility
     - Does the perturbation target belong to a well-characterized signaling
       pathway?
     - Are expected downstream genes or modules represented in the dataset?
  2. Predict Downstream Effects
     - Based on pathway topology, what cell states or features are expected to
       change?
     - Are these detectable in the available omics modality?
  3. Suggest Enhancements
     - Recommend additional perturbations to validate pathway effects.
     - Propose experimental readouts to strengthen pathway conclusions.
```

Cell Communication Expert

Expert Role Setting

Cell Communication Expert

```
  You are acting as a Cell Communication Expert in a multi-agent single-cell
    modeling system. Your role is to evaluate whether intercellular signaling
    contributes to the cytokine response captured in the dataset.

  You will receive a task analysis report that includes:
  - A single-cell dataset containing cytokine expression or response.
  - Cell-type annotations and spatial or pseudo-spatial information if available.
  - Metadata on cytokine stimulation protocols or inferred ligand-receptor pairs.

  Your objectives:
  1. Identify Communication Patterns
```

```
   - Are there likely paracrine or autocrine effects influencing cytokine
     expression?
   - Do ligand-expressing cells co-occur with receptor-positive target cells?
2. Evaluate Impact on Task
   - Could intercellular signaling confound or explain observed cytokine
     responses?
   - Are current assays sufficient to separate intrinsic vs. extrinsic effects?
3. Recommend Additions
   - Suggest experiments (e.g., co-culture, transwell) to isolate signaling
     effects.
   - Recommend including spatial transcriptomics if necessary.
```

Omics Modality Expert

Expert Role Setting

Omics Modality Expert

```
You are acting as an Omics Modality Expert in a multi-agent model evaluation
    system. Your role is to assess whether the chosen data modality (e.g., RNA-
    seq, ATAC-seq, protein) is suitable for capturing the effects of the
    specified perturbation.

You will receive a task analysis report that includes:
- A single-cell perturbation dataset with modality metadata.
- Task objective and prediction targets.
- Optional: known regulatory links (e.g., enhancer-promoter pairs, TF motifs,
    signaling cascades).

Your objectives:
1. Evaluate Signal Availability
   - Is the measured modality expected to show downstream effects of the
     perturbation?
   - Are known markers or targets captured by the modality?
2. Assess Measurement Resolution
   - Does the modality offer sufficient resolution (gene-level, peak-level,
     surface protein) to model the task?
3. Suggest Modality Enhancements
   - Recommend complementary modalities (e.g., ATAC + RNA) if needed.
   - Propose targeted panels or multi-omics techniques to improve
     interpretability.
```

G. Detailed outputs from scAgents

G.1. Data Parser

The input is Norman et al. (Norman et al., 2019) Dataset in h5ad format.

Data Parser Output

```
Modality:RNA Perturbation type:CRISPRa
dataset_index: filtered
Title: Exploring genetic interaction manifolds constructed from rich single-cell
    phenotypes
Organisms: Homo sapiens
Modality = Data type: RNA
Method: Perturb-seq
Tissues: K562
Perturbation: CRISPRa
disease: chronic myelogenous leukemia
celltype: lymphoblasts
tissue type: cell_line
Mini-Abstract (loosely summarized original Abstract): Here, the authors present
    an analytical framework for interpreting high-dimensional landscapes of cell
    states (manifolds) constructed from transcriptional phenotypes. They applied
    this approach to Perturb-seq profiling of strong genetic interactions (GIs)
    mined from a growth-based, gain-of-function GI map. Exploration of this
    manifold enabled ordering of regulatory pathways, principled classification
    of GIs (e.g., identifying suppressors), and mechanistic elucidation of
    synergistic interactions. Finally, they applied recommender system machine
    learning to predict interactions, facilitating exploration of vastly larger
    GI manifolds.

'''contains
                                guide_id read_count UMI_count coverage gemgroup
                                    ... nperts ngenes ncounts percent_mito
                                    percent_ribo
TTGAACGAGACTCGGA ARID1A_NegCtrl0;ARID1A_NegCtrl0 28684 1809 15.856274 2 ... 1
    3079 15097.0 5.815725 33.569583
CGTTGGGGTGTTTGTG BCORL1_NegCtrl0;BCORL1_NegCtrl0 18367 896 20.498884 7 ... 1 2100
     8551.0 4.104783 45.842592
GAACCTAAGTGTTAGA FOSB_NegCtrl0;FOSB_NegCtrl0 16296 664 24.542169 6 ... 1 2772
    10999.0 5.655060 17.801618
CCTTCCCTCCGTCATC SET_KLF1;SET_KLF1 16262 850 19.131765 4 ... 2 5385 38454.0
    4.335050 38.165080
TCAATCTGTCTTTCAT OSR2_NegCtrl0;OSR2_NegCtrl0 16057 1067 15.048735 2 ... 1 4869
    27926.0 5.084867 32.317554
... ... ... ... ... ... ... ... ... ... ... ...
TTTGCGCAGTCATGCT RHOXF2_NegCtrl0;RHOXF2_NegCtrl0 1 1 1.000000 2 ... 1 1853 5192.0
     5.508475 31.798921
TTTGCGCCAGGACCCT BCL2L11_BAK1;BCL2L11_BAK1 1 1 1.000000 3 ... 2 3508 15704.0
    6.718034 38.334182
TTTGCGCGTACTTGAC-1 CNN1_NegCtrl0;CNN1_NegCtrl0 1 1 1.000000 3 ... 1 3609 15054.0
    5.633054 29.440680
TTTGCGCTCTCGCATC-1 CEBPB_OSR2;CEBPB_OSR2 1 1 1.000000 6 ... 2 2576 6825.0
    2.695971 16.879121
TTTGGTTGTTCCGTCT MAP2K3_MAP2K6;MAP2K3_MAP2K6 1 1 1.000000 2 ... 2 2499 8331.0
    5.617573 34.785740

[111445 rows x 20 columns]
---
obs

Index(['guide_id', 'read_count', 'UMI_count', 'coverage', 'gemgroup',
     'good_coverage', 'number_of_cells', 'tissue_type', 'cell_line',
     'cancer', 'disease', 'perturbation_type', 'celltype', 'organism',
     'perturbation', 'nperts', 'ngenes', 'ncounts', 'percent_mito',
     'percent_ribo'],
    dtype='object')
                good_coverage number_of_cells tissue_type cell_line ...
                   perturbation_type celltype organism perturbation
TTGAACGAGACTCGGA True 1 cell_line K562 ... CRISPR lymphoblasts human ARID1A
CGTTGGGGTGTTTGTG True 1 cell_line K562 ... CRISPR lymphoblasts human BCORL1
```

```
GAACCTAAGTGTTAGA True 1 cell_line K562 ... CRISPR lymphoblasts human FOSB
CCTTCCCTCCGTCATC True 1 cell_line K562 ... CRISPR lymphoblasts human SET_KLF1
TCAATCTGTCTTTCAT True 2 cell_line K562 ... CRISPR lymphoblasts human OSR2
... ... ... ... ... ... ... ... ... ...
TTTGCGCAGTCATGCT False 0 cell_line K562 ... CRISPR lymphoblasts human RHOXF2
TTTGCGCCAGGACCCT False 0 cell_line K562 ... CRISPR lymphoblasts human
    BCL2L11_BAK1
TTTGCGCGTACTTGAC-1 False 0 cell_line K562 ... CRISPR lymphoblasts human CNN1
TTTGCGCTCTCGCATC-1 False 0 cell_line K562 ... CRISPR lymphoblasts human
    CEBPB_OSR2
TTTGGTTGTTCCGTCT False 0 cell_line K562 ... CRISPR lymphoblasts human
    MAP2K3_MAP2K6

[111445 rows x 10 columns]
---
var

              ensemble_id ncounts ncells
RP11-34P13.3 ENSG00000243485 29.0 29
FAM138A ENSG00000237613 0.0 0
OR4F5 ENSG00000186092 0.0 0
RP11-34P13.7 ENSG00000238009 266.0 265
RP11-34P13.8 ENSG00000239945 10.0 10
... ... ... ...
AC233755.2 ENSG00000277856 0.0 0
AC233755.1 ENSG00000275063 0.0 0
AC240274.1 ENSG00000271254 11735.0 10835
AC213203.1 ENSG00000277475 0.0 0
FAM231B ENSG00000268674 0.0 0

[33694 rows x 3 columns]
---
shape

(111445, 33694)
```

G.2. Agentic Retrieval

Task Analysis

Example Agentic Retrieval Output

```
Keywords: Norman Weissman 2019 Perturb-seq CRISPRa K562 single cell perturbation
    prediction

Round 1: Initial DFS Search (one example branch)
Keywords: single cell perturbation prediction (what is -> how to solve)
Learning single-cell perturbation responses using neural optimal transport...{
    Nature Link}
Modeling and predicting single-cell multi-gene perturbation responses...{PMC Link
    }
Explainable modeling of single-cell perturbation data using Bayesian hierarchical
     modeling...{Cell Press Link}
A Multiplexed Single-Cell CRISPR Screening...{Cell Press Link}
Predicting transcriptional outcomes of novel perturbations...{PMC Link}
Modeling and predicting single-cell multi-gene perturbation responses...{PMC Link
    }
Exploring genetic interaction manifolds...{PMC Link}
Predicting transcriptional outcomes of novel perturbations...{PMC Link}
In-silico biological discovery with large-scale perturbation data{arXiv Link}
DeepChrome 2.0: Investigating and modeling chromatin accessibility...{arXiv Link}
Predicting the genetic component of gene... {arXiv Link}
A genome-scale deep learning model to pre...{arXiv Link}
Attention-based Interpretable Regression...{arXiv Link}
GATES: Graph Network...{arXiv Link}
GRNFormer: Biologically...{arXiv Link}


Round 2: BFS Search (one example branch)
Key words:state-of-the-art models for single-cell perturbation prediction
```

```
GEARS: Predicting transcriptional outcomes of novel multi-gene perturbations{
    nature Link}
scGPT: Is language all you need for modeling single-cell perturbation responses{
    nature Link}
Geneformer - BioNeMo Framework for Genomic Language Modeling{nature link}
Efficient Fine-Tuning of Single-Cell Foundation Models for Perturbation
    Prediction...{arXiv Link}
Multicell-Fold: geometric learning in folding...{PMC Link}
Variational Mixtures of ODEs for Inferring Cellular...{ICML}
DeepChrome 2.0: Investigating and modeling chromatin accessibility...{arXiv Link}
Predicting the genetic component of gene...{Bioinformatics}
A genome-scale deep learning model to pre...{PMC Link}

Round 3: DFS Search (one example branch)
Keywords: GEARSscGPTGeneformerMulticell-FoldDeepChrome 2.0...
GEARS: Predicting transcriptional outcomes of novel multi-gene perturbations{
    nature Link}
Predicting transcriptional outcomes of novel multigene perturbations with GEARS
    ...{nature Link}
snap-stanford/GEARS {Github Link}
scGPT: Is language all you need for modeling single-cell perturbation responses{
    nature Link}
bowang-lab/scGPT{Github Link}
Geneformer - BioNeMo Framework for Genomic Language Modeling{nature link}
jkobject/geneformer {Github Link}
Multicell-Fold: geometric learning in folding...{PMC Link}
bm2-lab/scPerturBench{Github Link}
DeepChrome 2.0: Investigating and modeling chromatin accessibility...{arXiv Link}
DeepChrome/DeepChrome2.0{Github Link}
.......


Round 4: BFS Search (one example branch)
Keywords: TransformerVAE... (used by models for single-cell perturbation
    prediction)
TransVAE: Giving Attention to Generative VAE Models for De Novo Molecular Design{
    https://github.com/oriondollar/TransVAE}{GitHub Link}
Transformer Graph Variational Autoencoder for Generative Molecular Design{https
    ://www.biorxiv.org/content/10.1101/2024.07.22.604603v1.full.pdf}{Paper Link}
A novel molecule generative model of VAE combined with Transformer{https://arxiv.
    org/pdf/2402.11950}{arxiv Link}
A Graph VAE and Graph Transformer Approach to Generating{https://arxiv.org/abs
    /2104.04345}{arxiv Link}
Large Scale GNN and Transformer Models and for Genomics{https://qdata.github.io/
    deep2Read//talks-mb2019/Jack_20190530-LargeScaleGNNTransformer.pdf}

Round 5: DFS Search
......
```

G.3. Task Analysis Report

Data Analyst Output

Task Analysis

```
### Introduciton

- **Modalities**: RNA (single-cell RNA-seq)
- **Perturbation Type**: CRISPRa (activation)
- **Conditions**: 1,092 unique perturbation conditions including:
  - 105 single-gene perturbations
  - 131 paired-gene perturbations
- **Timepoints**: Not explicitly reported (likely single timepoint)
- **Replicates**: Present (multiple cells per perturbation condition)
- **Batches**: Likely present due to sequencing logistics
- **Cell Types**: K562 cells (chronic myelogenous leukemia cell line)
- **Organism**: Homo sapiens
- **Description**: This Perturb-seq experiment profiles transcriptional responses
    in K562 cells following CRISPRa perturbations. The dataset captures both
    single and paired genetic perturbations, providing insight into genetic
    interaction networks.

### Data Properties

- **Number of Cells**: 111,445
- **Number of Genes**: 33,694
- **Features**:
  - **RNA**: 33,694 genes
  - **ATAC**: Not available
  - **Protein**: Not available
- **Perturbation Targets**:
  - **Unique Perturbations**: 105 single genes and 131 gene pairs
  - **Target Type**: Genes and gene pairs
  - **Coverage**: Sparse (many genes show no expression change)
- **Modality Completeness**: RNA data available for all cells with appropriate
    quality control metrics
- **Metadata Completeness**: Comprehensive perturbation annotations, technical
    covariates (UMI_count, percent_mito, percent_ribo), and cell barcodes
- **Preprocessing Required**:
  - Normalization of UMI counts
  - Log-transformation of expression values
  - Filtering of lowly expressed genes
  - Batch effect correction

### Quality Assessment

- **Data Sparsity**: 78% of gene expression values are zero due to dropout events
    and biological sparsity
- **Batch Effect**: Potential technical variability from different sequencing
    runs
- **Replicate Consistency**: Not explicitly reported but suggested by
    reproducibility of common perturbations
- **Known Issues**:
  - Extreme class imbalance (some perturbations appear only once)
  - High dimensionality with 33,694 genes per output
  - Technical noise inherent to single-cell sequencing
- **Strengths**:
  - Rich perturbation annotations with precise gene targeting
  - Large number of profiled cells providing statistical power
  - Comprehensive technical covariates enabling noise correction
- **Limitations**:
  - High dimensionality challenges
  - Sparse perturbation distribution
  - Potential confounding from clonal expansion

### Recommendations

- **Preprocessing Steps**:
  - Filter genes with fewer than 10 counts across all cells
  - Normalize UMI counts using library size factors
```

```
  - Apply log-transformation (log1p) to expression values
  - Combat batch effects using Harmony or Scanorama
- **Modeling Considerations**:
  - Focus modeling efforts on the top 5,000-10,000 variable genes
  - Use sparse-aware models (e.g., sparse neural networks)
  - Incorporate technical covariates as model inputs
  - Implement data augmentation for rare perturbations
- **Open Questions**:
  - Optimal representation of paired perturbations in model architecture
  - Strategies for handling extreme class imbalance in training
  - Best practices for integrating technical covariates

### Refinement Suggestions

- Investigate the impact of clonal expansion on expression profiles
- Assess the quality of rare perturbation data for potential filtering
- Explore data augmentation techniques for underrepresented perturbations
- Develop quality control metrics specific to Perturb-seq data
```

Problem Investigator Output

Task Analysis

```
### Biological Question

How do genetic perturbations propagate through gene regulatory networks to alter
    the transcriptional landscape of K562 cells, and can we predict these changes
     for novel perturbations and cellular contexts?

### Hypothesis Statement

A predictive model can accurately estimate post-perturbation gene expression
    profiles by learning the complex relationships between baseline cell states,
    genetic perturbations, and transcriptional responses, including non-linear
    genetic interaction effects.

### Task Formulation

- **Input**:
  - Baseline gene expression profile (33,694 genes)
  - Perturbation identities (single or paired genes)
  - Technical covariates (UMI_count, percent_mito, percent_ribo)
- **Output**:
  - Post-perturbation gene expression profile (33,694 genes)
- **Task Type**:
  - High-dimensional regression with combinatorial inputs

### Justification

- **Biological Relevance**:
  - Understanding genetic interactions is fundamental to deciphering cellular
      response networks
  - Mapping genotype-phenotype relationships at single-cell resolution
  - Predicting cellular responses to novel perturbations accelerates functional
      genomics research
- **Data Suitability**:
  - Rich perturbation annotations enable supervised learning approaches
  - Single-cell resolution captures heterogeneity in cellular responses
  - Coverage of both single and paired perturbations allows study of genetic
      interactions
- **Expected Challenges**:
  - High-dimensional output space with 33,694 genes per prediction
  - Non-linear genetic interactions requiring complex model architectures
  - Generalization to unseen perturbations and cellular contexts
  - Technical noise and dropout events in single-cell data

### Evaluation Plan

- **Metrics**:
```

```
  - Pearson Correlation Coefficient (PCC) across all genes and on top 1,000
      differentially expressed genes(DE)
  - Mean Squared Error (MSE) across all genes and for differentially expressed
      genes
  - R score across all genes and for differentially expressed genes
- **Baselines to Consider**:
  - Linear regression models with gene-level outputs
  - Gene-wise neural networks
  - Graph neural networks using known regulatory interactions
- **Validation Strategy**:
  - Stratified group k-fold cross-validation holding out entire perturbation
      conditions
  - Separate validation sets for unseen single-gene and paired perturbations
  - Baseline profile holdout sets to test generalization to new cellular contexts

### Open Questions

- Optimal representation of paired perturbations in model architecture
- Strategies for handling extreme class imbalance during training
- Incorporation of technical covariates into model architecture
- Best practices for defining differentially expressed genes in this context
```

Baseline Assessor Output

Task Analysis

```
### Literature Overview

- **Perturbation Types**:
  - CRISPRi (interference)
  - CRISPRa (activation)
- **Existing Methods**:
  - GEARS: Graph neural network with gene interaction modeling
  - scGPT: Transformer architecture for single-cell data
  - Geneformer: Pretrained transformer for genomics
  - scLAMBDA: Variational autoencoder for perturbation prediction
  - **VAE (Variational Autoencoder)**: Probabilistic model for learning latent
      representations of cellular states
- **Technical Trends**:
  - Transformer architectures for capturing long-range dependencies
  - Graph neural networks for explicit gene interaction modeling
  - Variational autoencoders for probabilistic modeling of cellular states
  - Hybrid models combining multiple data modalities
  - Deep generative models for data augmentation and uncertainty quantification

### Candidate Models

#### GEARS (Gene Network Embedding for Perturbation Response Prediction)

- **Architecture**: Graph Neural Network (GNN) combined with Multi-Layer
    Perceptron (MLP)
- **Strengths**:
  - Explicitly models gene dependencies using known regulatory interactions
  - Handles combinatorial perturbations through graph propagation
  - Demonstrated success in previous Perturb-seq challenges
- **Weaknesses**:
  - Relies on external gene interaction databases
  - May overfit to common perturbations with limited generalization
  - Computationally intensive for full transcriptome modeling
- **Biological Applicability**:
  - Captures genetic interactions and regulatory relationships
  - Models enhancer-promoter relationships in K562 cells
  - Provides interpretable gene importance scores

#### scGPT (Single-Cell Generative Perturbation Transformer)

- **Architecture**: Transformer with multi-head self-attention
- **Strengths**:
  - Captures long-range gene interactions without relying on external databases
  - Robust to technical noise through attention mechanisms
```

```
  - Handles variable numbers of perturbations naturally
- **Weaknesses**:
  - Requires extensive pretraining on large datasets
  - Computationally demanding for full transcriptome modeling
  - May struggle with extreme class imbalance
- **Biological Applicability**:
  - Models context-dependent transcriptional responses
  - Handles sparse data efficiently through attention mechanisms
  - Provides gene importance scores through attention weights

#### Enformer (Enhancer former)

- **Architecture**: Dilated Convolutional Neural Network (CNN)
- **Strengths**:
  - Effective at modeling sequence-to-expression relationships
  - Provides interpretable feature importance scores
  - Computationally efficient compared to transformer architectures
- **Weaknesses**:
  - Requires DNA sequence input not directly applicable to post-transcriptional
      perturbations
  - Limited ability to model combinatorial genetic effects
  - Not designed for single-cell data with technical covariates
- **Biological Applicability**:
  - Predicts expression changes from DNA sequence modifications
  - Limited utility for CRISPRa perturbations affecting post-transcriptional
      regulation


### Recommended Baselines

#### Graph Neural Network (GNN) with Gene Interaction Modeling

- **Rationale**: Explicitly models gene dependencies and can incorporate known
    regulatory interactions while remaining flexible to learn from data
- **Implementation Details**:
  - Use PyTorch Geometric for efficient graph operations
  - Construct gene interaction graphs from public databases (e.g., STRING,
      BioGRID)
  - Implement separate graph branches for regulatory and co-expression
      relationships
  - Include attention mechanisms to weight different interaction types
  - Embed perturbation identities using learned gene embeddings
  - Concatenate baseline expression features with perturbation embeddings
  - Apply multiple GNN layers followed by dense layers for prediction
- **Evaluation Metrics**: PCC, MSE, R
- **Biological Relevance**: Captures genetic interactions and regulatory
    mechanisms, providing insight into how perturbations propagate through
    networks

#### Transformer Architecture with Gene Positional Encoding

- **Rationale**: Capable of discovering complex gene interactions without relying
     on external databases, with architectural flexibility for different input
    modalities
- **Implementation Details**:
  - Use PyTorch with Hugging Face transformer libraries
  - Encode genes as positional tokens with expression values
  - Implement specialized embeddings for perturbed genes
  - Apply layer normalization and residual connections
  - Use mixed precision training to handle large output dimensions
  - Implement masking for rare perturbations during training
  - Apply attention pooling to focus on biologically relevant genes
- **Evaluation Metrics**: PCC, MSE, R
- **Biological Relevance**: Models context-dependent responses and technical
    noise robustly, providing flexibility to adapt to different biological
    questions


#### VAE (Variational Autoencoder)

- **Architecture**: Encoder-decoder architecture with probabilistic latent space
- **Strengths**:
```

```
  - Models uncertainty in cellular states and perturbation responses
  - Effective for data augmentation through generation of new cellular states
  - Provides compressed latent representations for downstream analysis
  - Handles sparse and noisy single-cell data well
- **Weaknesses**:
  - May oversimplify complex biological relationships in latent space
  - Requires careful tuning of KL divergence weighting
  - Potential blurring of distinct cellular states in latent space
- **Biological Applicability**:
  - Captures multimodal distributions of cellular responses
  - Enables exploration of cellular state transitions following perturbations
  - Provides robust representations for classifying cellular phenotypes

VAE with Perturbation Conditioning:
- **Rationale**: Models uncertainty in cellular responses and provides robust
    latent representations for downstream analysis while enabling data
    augmentation
- **Implementation Details**:
  - Use PyTorch for flexible probabilistic modeling
  - Implement encoder-decoder architecture with probabilistic latent space
  - Include perturbation identities as conditional inputs to the decoder
  - Apply beta-VAE regularization to balance reconstruction and latent space
      regularization
  - Implement sparse VAE modifications to handle zero-valued genes
  - Use importance weighting for rare perturbations during training
  - Apply latent space interpolation to explore cellular state transitions
- **Evaluation Metrics**: PCC, MSE, R, ELBO (Evidence Lower Bound)
- **Biological Relevance**: Captures multimodal distributions of cellular
    responses, enables exploration of cellular state transitions, and provides
    robust representations for classifying cellular phenotypes

### Evaluation Framework

- **Primary Metrics**:
  - PCC for differentially expressed genes
  - MSE for differentially expressed genes
  - R for differentially expressed genes
- **Secondary Metrics**:
  - Global PCC across all genes
  - Global MSE across all genes
  - Classification accuracy for key gene up/down regulation
  - ELBO for VAE models
- **Validation Strategy**: Stratified group k-fold cross-validation holding out
    entire perturbation conditions
- **Test Scenarios**:
  - Unseen single-gene perturbations
  - Unseen paired perturbations with one known gene
  - Completely novel gene pairs
  - Baseline profiles from held-out cell subsets

### Improvement Suggestions

- **Technical**:
  - Incorporate attention mechanisms to focus on biologically relevant genes
  - Apply data augmentation for rare perturbations (e.g., mixup, elastic
      deformation)
  - Use mixed precision training to reduce memory requirements
  - Implement gradient checkpointing for deeper architectures
  - Experiment with different VAE architectures (e.g., VAE-GCN, VAE-Transformer)
- **Biological**:
  - Integrate known regulatory interactions as model priors
  - Focus modeling efforts on biologically relevant gene subsets
  - Model cell cycle effects using phase markers
  - Incorporate chromatin accessibility information when available
  - Use VAE latent space for phenotypic classification and clustering
- **Computational**:
  - Optimize for sparse operations to handle zero-valued genes efficiently
  - Implement efficient parallel processing strategies
  - Use model pruning to reduce parameters while maintaining performance
  - Apply distributed training for VAE models with large latent spaces
```

Crtic Refinement Output

Task Analysis

```
### Summary

- **Biological Context**: Predicting transcriptional responses to CRISPR
    perturbations in K562 cells requires modeling complex genetic interactions
    and cellular context dependencies while generalizing to novel perturbations
- **Technical Requirements**: Handling high-dimensional sparse data, extreme
    class imbalance, and the need for inductive generalization
- **Refinement Overview**: Integration of dataset analysis, problem formulation,
    and baseline recommendations into a cohesive analysis plan with aligned
    terminology and evaluation metrics

### Task Definition

- **Input Modalities**:
  - Baseline gene expression profile (33,694 genes)
  - Perturbation identities (single or paired genes)
  - Technical covariates (UMI_count, percent_mito, percent_ribo)
- **Output Targets**: Post-perturbation gene expression profile (33,694 genes)
- **Task Type**: High-dimensional regression with combinatorial inputs
- **Biological Significance**: Enables understanding of genetic interaction
    networks and prediction of cellular responses to novel perturbations

### Baseline Models

- **Recommended Models**:
  - Graph Neural Network (GNN)
  - Transformer Architecture
  - VAE with Perturbation Conditioning
- **Model Comparisons**:
  - GNNs excel at explicit gene interaction modeling using known regulatory
      networks
  - Transformers offer flexible interaction discovery without relying on external
      databases
  - VAEs provide probabilistic modeling of cellular states and enable data
      augmentation
- **Implementation Details**:
  - Implement GNNs with PyTorch Geometric using gene interaction graphs derived
      from prior knowledge
  - Implement Transformers with PyTorch/Hugging Face using gene positional
      encoding
  - Implement VAEs with PyTorch using conditional latent spaces for perturbation
      modeling
  - Include specialized embeddings for perturbations and normalize technical
      covariates
  - Apply mixed precision training and gradient checkpointing for efficiency

### Constraints

- **Dataset Limitations**:
  - Class imbalance with rare perturbations appearing only once
  - Data sparsity with 78% zero-valued genes
  - Potential batch effects from different sequencing runs
- **Technical Constraints**:
  - Computational resources for training large models on full transcriptome data
  - Model interpretability requirements for biological validation
- **Biological Constraints**:
  - Need for generalization to unseen perturbations and cellular contexts
  - Model must align with known regulatory mechanisms where possible
  - Focus on biologically relevant gene subsets to avoid overfitting

### Evaluation

- **Primary Metrics**:
  - PCC across all genes and for differentially expressed genes (top 1,000)
  - MSE across all genes and for differentially expressed genes
  - R across all genes andfor differentially expressed genes
```

```
- **Validation Strategy**: Stratified group k-fold cross-validation holding out
    entire perturbation conditions
- **Test Scenarios**:
  - Unseen single-gene perturbations
  - Unseen paired perturbations with one known gene
  - Completely novel gene pairs
  - Baseline profiles from held-out cell subsets
```

Final Report

Task Analysis

```
Task Analysis

# Biological Objective
Predict post-perturbation gene expression profiles in K562 cells to understand
    genetic interaction networks and enable discovery of novel regulatory
    mechanisms. This work aims to develop a computational tool for exploring
    genetic interaction manifolds, accelerating functional genomics research and
    therapeutic target discovery.

# Technical Approach
Develop high-dimensional regression models incorporating baseline expression,
    perturbation identities, and technical covariates. The models must explicitly
     handle sparse data, extreme class imbalance, and demonstrate inductive
    generalization to novel perturbations and cellular contexts. The plan
    includes exploration of deterministic models (GNNs, Transformers) and
    probabilistic models (VAEs) to capture different aspects of cellular response
     variability.

# Dataset Characterization
## Origin
Norman et al. (2019) Perturb-seq dataset (GEO: GSE133344).
Key Features
Number of Cells: 111,445
Number of Genes: 33,694
Perturbation Conditions: 1,092 unique conditions (105 single genes, 131 gene
    pairs)
Technical Covariates: UMI_count, percent_mito, percent_ribo
## Challenges
Class Imbalance: Rare perturbations appear only once.
Data Sparsity: 78% zero-valued genes due to dropout events.
Technical Noise: Inherent to single-cell sequencing.
Batch Effects: Potential variability from different sequencing runs.

# Problem Formulation

## Biological Question

How do genetic perturbations propagate through gene regulatory networks to alter
    the transcriptional landscape of K562 cells, and can we predict these changes
     for novel perturbations and cellular contexts?

## Hypothesis Statement
A predictive model can accurately estimate post-perturbation gene expression
    profiles by learning the complex relationships between baseline cell states,
    genetic perturbations, and transcriptional responses, including non-linear
    genetic interaction effects.

## Task Definition
### Input:
Baseline gene expression profile (33,694 genes)
Perturbation identities (single or paired genes)
Technical covariates (UMI_count, percent_mito, percent_ribo)

### Output:
Post-perturbation gene expression profile (33,694 genes)

###Task Type:
High-dimensional regression with combinatorial inputs
```

```
## Justification
Biological Relevance:
Understanding genetic interactions is fundamental to deciphering cellular
    response networks
Mapping genotype-phenotype relationships at single-cell resolution
Predicting cellular responses to novel perturbations accelerates functional
    genomics research
Data Suitability:
Rich perturbation annotations enable supervised learning approaches
Single-cell resolution captures heterogeneity in cellular responses
Coverage of both single and paired perturbations allows study of genetic
    interactions
Expected Challenges:
High-dimensional output space with 33,694 genes per prediction
Non-linear genetic interactions requiring complex model architectures
Generalization to unseen perturbations and cellular contexts
Technical noise and dropout events in single-cell data

The model's performance will be evaluated under two key scenarios to assess its
    generalizability:
- Unseen Perturbations: The model should be able to accurately predict the
    effects of CRISPRi targeting genes or gene pairs that were not included in
    the training data. This scenario tests the model's ability to extrapolate its
     learned knowledge to novel genetic manipulations.
- Unseen Cell Contexts: The model should be capable of predicting the response to
     a perturbation in cells with baseline gene expression profiles that were not
     observed during the training phase. This evaluates the model's robustness to
     the inherent heterogeneity within the K562 cell population.

# Baseline Model Analysis


**SOTA**: GEARS achieves best Pearson correlation in combinatorial prediction
    tasks but violates the "no external database" constraint .

Below are detailed critiques of each baselines shortcomings in the context of the
     AdamsonWeissman UPR CRISPRi dataset, followed by concrete
    recommendationsgrounded in recent literaturefor how to overcome them. Each
    point is supported by highquality citations.

1. SC-GPT

**Shortcomings:**

1). **Discrete Perturbation Tokens:** SC-GPT treats each perturbation (e.g. a
    specific dualguide combination) as a unique token. It cannot form embeddings
    for guide sets unseen in pretraining, so it fails on novel combinations
2). **No Zero-Inflated Modeling:** SC-GPTs Gaussian or cross-entropy losses dont
    account for dropoutdriven zeros common in scRNA-seq, causing biased
    predictions for low-UMI cells
3). **Parameter Bloat for Dense Output:** Extending SC-GPTs languagemodel head to
     35 kdimensional gene outputs inflates parameters, hindering training
    efficiency and generalization

2. GeneFormer
**Shortcomings:**
1). **Single-Gene Focus:** GeneFormer has been validated primarily on singlegene
    knockouts, lacking mechanisms to **compose** multiple guide embeddings for
    combinatorial CRISPRi
2). **Static Graph Priors:** It uses a fixed genegene network that doesnt adapt
    to perturbationinduced regulatory rewiring in the UPR pathway, limiting
    dynamic response modeling
3. **Scalability Issues:** Fullgraph attention over 35 k genes is intractable, so
     practical implementations subsample to 25 k genesdiscarding potentially
    important UPR regulators

3. DEEP (Plain MLP)
**Shortcomings:**
1). **Ignores Gene Covariance:** Treats each gene independently, missing co-
    regulation patterns (e.g., ATF6XBP1 axis in UPR)
```

```
2). **Overfitting Risk:** Millions of parameters on 35 k inputs with limited
    replicates per combination leads to memorization, not generalization to
    unseen guide sets
3). **No Interpretability:** Provides no insight into which genes or interactions
     drive predictions, unlike graph-based or attention-based models.


4. GEARS
**Shortcomings:**
1). **External Knowledge Dependency:** GEARS integrates a genegene knowledge
    graph (e.g., from STRING or GO) to regularize embeddings, which violates our
    no external database constraint
2). **Fixed Graph Structure:** The perturbation relationship graph in GEARS is
    static, not conditioned on cell-state or UPR context, limiting dynamic
    response capture.
3). **Heavy GNN Overhead:** Graph neural network message passing on 35 k nodes
    multiple perturbations incurs high memory and compute costs, impractical for
    largescale CRISPRi screens.

---

Recommendations for Improvement

1. **Factorized Perturbation Embeddings**

   - **Approach:**
   Learn a separate embedding $e_{g}$ for each guide $g$. Represent a
       perturbation set $P$ by a **learned nonlinear composition**.

   - **Benefit:** Zeroshot support for unseen guide combinations via embedding
       arithmetic, as demonstrated by CPA and scGen

2. **Zero-Inflated Negative Binomial (ZINB) Loss**

   - **Approach:** Replace MSE with a **ZINB loss** that models both dropout
       probability and overdispersion per gene.

   - **Benefit:** Accounts for scRNA-seq technical noise, improving prediction in
        low-UMI cells (e.g., ~162 median UMI)

3. **Learned Dynamic Graph Priors**

   - **Approach:** Instead of a fixed PPI graph, **learn genegene affinity
       weights** from data using a **Gaussian kernel** on baseline coexpression,
       then refine during training.

   - **Benefit:** Captures UPR pathway rewiring under CRISPRi; avoids external
       databases (matches constraints)

4. **Contrastive Pretraining**

   - **Approach:**

   Pretrain an encoder on (x,0) vs. (x,p) pairs with an **InfoNCE contrastive
       loss**, pulling matched baselineperturbed embeddings together and pushing
       apart random mismatches.

   - **Benefit:** Disentangles baseline state from perturbation effect; enhances
       generalization to unseen contexts like new guide sets or cellcycle states

5. **Neural ODE or OT Trajectory Module**

   - **Approach:** For multiguide dynamics, model latent drift via a **neural ODE
       ** $dz/dt=f(z,hP)dz/dt = f(z,h_P)$ or **optimaltransport regularization**
       linking $(x,p)\to(x,p')$.

   - **Benefit:** Enforces smooth interpolation/extrapolation between
       perturbation levels (01 guide,,4 guides), capturing graded UPR responses

6. **Perturbation-Guided Attention Decoder**

   - **Approach:** Use a **crossattention** layer where query = latent state, key
       /value = perturbation embedding $h_P$, then an attentionaugmented decoder
```

```
        to weight genes most impacted by UPR factors (e.g., ATF6 targets).

    - **Benefit:** Focuses model capacity on biologically salient genes; improves
        interpretability and reduces parameter count


**Implementing these enhancements**factorized, compositional embeddings; ZINB
    loss; datadriven graph priors; contrastive pretraining; latent ODE/OT modules
    ; and perturbationguided attentionwill address the key limitations of SC-GPT,
     GeneFormer, DEEP, GEARS. This integrated architecture can robustly predict
    singlecell transcriptomic responses to both seen and unseen CRISPRi
    perturbations in the UPR pathway, while respecting the no external database
    constraint and modeling lowsignal, highsparsity regimes.
```

G.4. Graph based discussion

Graph Based Discussion Output

Round 1

Data Preprocessing Expert: The dataset contains over 1000,000 cells and 33,000 genes. It's quite large and very sparse. I propose starting with log-normalization to handle the skewness in the data. We could also use PCA for dimensionality reduction to capture the main sources of variation. Maybe around 100-200 principal components would be a good starting point. What do you all think?

Model Architecture Expert: I agree with the preprocessing steps. Given the complexity of gene regulatory networks, a graph neural network (GNN) might be suitable to capture gene interactions. However, constructing the gene interaction graph is challenging. We could start with a simple graph based on prior knowledge like protein-protein interaction networks. But our task is to predict the effects of unseen perturbations, so the model needs to be flexible enough to handle new genes not in the training set. Maybe combining a GNN with gene embeddings could work. The embeddings can capture functional similarities between genes.

Training and Optimization Expert: Training a GNN on this scale will be computationally intensive. We need an efficient training strategy. I suggest using a learning rate scheduler to adjust the learning rate during training. Also, to prevent overfitting, especially with the high dimensionality, dropout layers should be incorporated. Given the potential class imbalance in the perturbation types, we might need to use weighted loss functions or some form of resampling.

Self-Critic: The ideas so far are good, but I'm concerned about the model's ability to generalize. The GNN relies heavily on the quality of the gene interaction graph. If the graph is incomplete or inaccurate, it could negatively impact the predictions. Also, the embeddings might not capture all the necessary biological information. Maybe we should consider a more data-driven approach for constructing the gene interactions instead of relying solely on prior knowledge.

Pathway Analyst: From a biological standpoint, it's crucial to model not just direct effects of perturbations but also downstream effects. Genes don't operate in isolation. A perturbation to one gene can have cascading effects on many others. The model should ideally capture these dynamic changes. Also, cell-to-cell variability is significant in single-cell data. The model needs to account for this inherent noise.

Deep Learning Expert: I think incorporating attention mechanisms could help the model focus on the most relevant genes for each perturbation. Transformers have been successful in other areas of bioinformatics, particularly in capturing long-range dependencies. Maybe a hybrid model that combines transformer architecture with GNNs would be powerful. The transformer can handle the gene expression data and the GNN can model the interactions. Also, using a multi-task learning approach where the model predicts both the gene expression changes and the perturbation type could improve performance.

Critic: The initial plan covers several important aspects, but the model's generalization to unseen perturbations remains a critical issue. The proposed GNN with gene embeddings is a solid foundation, but the reliance on prior knowledge for the gene interaction graph could be a limitation. The data-driven approach suggested by the Self-Critic could mitigate this. Also, the training strategy needs to address the high dimensionality and sparsity effectively. The inclusion of a contrastive learning component, as suggested by the Deep Learning Specialist, could enhance the model's ability to distinguish different perturbation effects. However, the evaluation metrics should be clearly defined to assess both the predictive accuracy and biological relevance.

Graph Based Discussion Output

Round 2

Data Preprocessing Expert: I've thought more about the preprocessing. Besides PCA, we could use a variational autoencoder (VAE) for dimensionality reduction. VAEs are generative models and might better capture the complex distributions in the data. Also, for handling the sparsity, adding a step to impute missing values could be beneficial. Maybe using a simple k-nearest neighbors imputation method.

Model Architecture Expert: I like the idea of combining transformers and GNNs. For the gene interaction graph in the GNN, perhaps we can construct it dynamically based on the data. Using the correlations between genes in the training data to build an adjacency matrix. This way, the graph is data-driven and doesn't rely solely on prior knowledge. Also, for the gene embeddings, we can initialize them with pre-trained embeddings from resources like Gene2Vec and then fine-tune them during training.

Training and Optimization Expert: Regarding training, I suggest using the AdamW optimizer with a OneCycle learning rate schedule. This schedule helps in finding a good balance between the learning rate and the number of epochs. For handling class imbalance, focal loss could be an option. It focuses the training more on the hard-to-classify examples. Also, implementing gradient clipping will help prevent exploding gradients, which is important given the model's complexity.

Self-Critic: While the hybrid model sounds promising, I'm still concerned about overfitting. With so many components, the model might memorize the training data. Maybe starting with a simpler model and gradually adding complexity would be a better approach. Also, how do we validate that the model's predictions are biologically meaningful? We need a solid evaluation strategy beyond just metrics like MSE and PCC.

Pathway Analyst: I agree with the need for dynamic modeling. The effects of perturbations aren't instantaneous. Maybe incorporating a recurrent component or using a time-series approach could help model the progression of changes. Also, including cell cycle phase as a covariate might improve predictions, as gene expression varies significantly across different phases.

Deep Learning Expert: I've considered the multi-task learning approach more. The model can have one branch predicting the gene expression changes and another branch classifying the perturbation type. This forces the model to learn features that are useful for both tasks. Also, using a contrastive loss component could help the model distinguish between different perturbation effects more effectively. The transformer's self-attention mechanism is particularly useful for capturing long-range dependencies in gene expression data, which could be crucial for understanding the downstream effects of perturbations.

Critic: The revised plan addresses several of my previous concerns. The use of a VAE for preprocessing and a data-driven gene interaction graph are positive steps. The multi-task learning and contrastive loss components add robustness. However, the evaluation strategy still needs clarification. We should define specific metrics for both predictive performance (MSE, PCC, R2) and biological relevance (MSE_DE, PCC_DE, R2_DE). Also, the model's ability to handle combinatorial perturbations should be explicitly tested.

Graph Based Discussion Output

Round 3

Data Preprocessing Expert: I've further refined the preprocessing steps. After careful consideration, I believe using a combination of log-normalization, imputation for missing values, and PCA for dimensionality reduction will provide a solid foundation for the model. I've also added data augmentation techniques like adding Gaussian noise and random masking to improve model robustness. The log-normalization step is crucial for handling the skewness in the gene expression data, while the PCA helps in reducing

```
    noise and computational complexity.

**Model Architecture Expert**: I've updated the model architecture to better
    align with the training phases. The model now includes a VAE for the gene
    expression data, a separate embedding layer for perturbations, and a
    transformer for capturing long-range dependencies. The dynamic graph
    construction based on data correlations addresses the concern about prior
    knowledge reliance. With the gene embeddings initialized from Gene2Vec, the
    model should have a good starting point for capturing gene functionalities.
    The VAE is particularly useful for denoising and capturing the complex
    distributions in the data.

**Training and Optimization Expert**: I've adjusted the training strategy to
    include three phases: 1) VAE pretraining for 20 epochs, 2) Transformer
    integration for 30 epochs, and 3) Full fine-tuning with adversarial samples
    for 50 epochs. The learning rate scheduling will be adjusted per phase to
    optimize training. I've also incorporated focal loss to handle class
    imbalance and gradient clipping to prevent exploding gradients. The focal
    loss is particularly effective in ensuring that the model pays attention to
    hard-to-classify examples.

**Self-Critic**: I'm more confident now. The plan to start with a simpler model
    and incrementally add complexity is reasonable. I've also developed a
    comprehensive evaluation strategy that includes comparisons with known
    biological pathways and interactions to ensure the predictions make sense
    biologically. This validation is crucial for the model's credibility. The
    evaluation strategy will include both quantitative metrics and qualitative
    assessments to ensure biological relevance.

**Pathway Analyst**: I'm satisfied with the incorporation of biological
    considerations. The model's ability to capture dynamic changes and the
    inclusion of cell cycle phase as a covariate will enhance the biological
    relevance of the predictions. I believe this approach will provide valuable
    insights into the effects of CRISPRi perturbations. The inclusion of cell
    cycle phase as a covariate is particularly important as it helps in
    accounting for the inherent biological noise in single-cell data.

**Deep Learning Expert**: I've implemented the multi-task learning and
    contrastive loss components. The model can now predict both gene expression
    changes and perturbation types, which should improve performance. I've also
    added a recurrent component to model the progression of changes over time.
    The multi-task learning approach ensures that the model learns features that
    are relevant for multiple tasks, improving its overall generalization
    capabilities. The transformer's self-attention mechanism allows the model to
    weigh the importance of different genes dynamically, which is particularly
    useful for capturing the complex regulatory networks in single-cell data.

**Critic**: The final plan addresses most of the previous concerns. The
    preprocessing steps are comprehensive, the model architecture is innovative
    and well-justified, and the training strategy is robust. The evaluation
    metrics are clearly defined, including both predictive performance (MSE/PCC/
    R2) and biological relevance (MSE_DE/PCC_DE/R2_DE). The model's ability to
    handle combinatorial perturbations through the dynamic graph and contrastive
    loss is a strong point. I believe this plan has a high likelihood of success.
```

G.5. Method Design Research Plan

Graph Based Discussion Output

```
# Data Preprocessing

Steps:
## Load Data
Load the gene expression matrix and metadata from the Norman et al. (2019)
    dataset using appropriate libraries like scanpy or anndata.

## Filter Low-Quality Cells and Genes
Remove cells with low RNA content or high mitochondrial content.
Filter out genes with low expression across cells.

## Normalize Data
Normalize the gene expression counts to account for differences in sequencing
    depth between cells. This can be done using methods like library size
    normalization.

## Log-Transformation
Apply log-transformation to the gene expression data to handle skewness.

## Batch Effect Correction
Use methods like harmony or Combat to remove batch effects from the data.

## Feature Selection
Identify highly variable genes (HVGs) to reduce dimensionality. This can be done
    using statistical methods to select genes with the highest variability across
     cells.

## PCA Dimensionality Reduction
Apply PCA to further reduce the dimensionality of the gene expression data. This
    helps in capturing the main sources of variation in a lower-dimensional space.


## Perturbation Encoding
Extract the perturbation information from the dataset's metadata.
Encode the perturbation information using one-hot encoding initially.
    Alternatively, use learned embeddings for each gene in the perturbation.

## Control Sample Handling
Identify and use control cells (unperturbed) to establish baseline gene
    expression profiles.

## Data Augmentation
Add Gaussian noise and apply random masking to the data to improve model
    robustness.

## Data Splitting
Split the dataset into training, validation, and test sets. Ensure that certain
    perturbations are held out for validation and testing to evaluate the model's
     ability to generalize to unseen perturbations and cell contexts.

# Model Design
## Overview
The proposed model is a hybrid neural network architecture designed to predict
    post-perturbation gene expression profiles in single cells. It integrates
    three key components:
Variational Autoencoder (VAE) for learning robust, low-dimensional
    representations of gene expression data.
Graph Neural Network (GNN) for modeling gene-gene interactions using a
    dynamically constructed gene regulatory network (GRN).
Transformer for capturing long-range dependencies and complex patterns in the
    fused gene-perturbation representations.
This combination allows the model to handle the high dimensionality, sparsity,
    and noise characteristic of single-cell data while providing biological
    interpretability through attention mechanisms and graph structures.

## Key Components

1. Variational Autoencoder (VAE) Encoder
```

```
Purpose: To compress high-dimensional gene expression data into a compact latent
    representation while capturing the underlying data distribution.
Architecture:
Input layer with size matching the number of principal components (e.g., 128).
Two hidden layers with decreasing dimensions (e.g., 256  128  64).
Output layer producing  (mean) and logvar (log variance) for the latent space.
Feasibility: VAEs are well-established for dimensionality reduction and denoising
     in scRNA-seq data.
Biological Interpretability: The learned latent space can be analyzed for
    biological patterns.

2. Perturbation Embedding Layer

Purpose: To convert perturbation identities (single or paired genes) into dense
    vector representations.
Architecture:
Input layer with size matching the number of unique perturbations (one-hot
    encoded).
Embedding layer projecting into a lower-dimensional space (e.g., 32 dimensions).
Feasibility: Embedding layers are standard for categorical data and reduce
    dimensionality.
Biological Interpretability: Similar perturbations will have similar embeddings,
    allowing analysis of perturbation relationships.

3. Graph Neural Network (GNN)

Purpose: To model gene regulatory networks (GRNs) and capture gene-gene
    interactions.
Architecture:
Dynamically construct a gene interaction graph using correlations from the
    training data.
Use Graph Convolutional Layers to propagate information across the graph.
Feasibility: GNNs are effective for capturing relational data and have been
    applied to gene networks.
Biological Interpretability: The learned graph can highlight key regulatory
    relationships.

4. Transformer Encoder

Purpose: To capture complex, long-range dependencies in the fused gene-
    perturbation representations.
Architecture:
Input layer combining VAE latent vectors and perturbation embeddings.
Multi-head self-attention mechanisms to weigh the importance of different genes.
Feed-forward neural networks for non-linear transformations.
Feasibility: Transformers excel at modeling sequential and complex data patterns.
Biological Interpretability: Attention weights can reveal important gene
    interactions.
5. Fusion and Output Layers
Purpose: To combine information from the VAE and perturbation branches and
    produce the final prediction.
Architecture:
Concatenate VAE latent vectors and perturbation embeddings.
Dense layers to transform the fused representation into gene expression
    predictions.
Feasibility: Concatenation is a simple yet effective fusion strategy.
Biological Interpretability: The final output can be directly compared to
    observed gene expression changes.

## Feasibility and Biological Interpretability

### Feasibility
Computational Efficiency: The model uses well-established and optimized
    components (VAE, Transformer, GNN) that can be efficiently trained on GPU.
Scalability: The architecture is designed to handle the high dimensionality of
    single-cell data while maintaining reasonable computational requirements.
Implementation: The model can be implemented using popular deep learning
    frameworks like PyTorch.

### Biological Interpretability
```

````
Attention Mechanisms: The Transformer's self-attention layers can highlight
    important gene interactions, providing insights into regulatory mechanisms.
Perturbation Embeddings: The learned embeddings can be analyzed to understand
    relationships between different perturbations.
Gene Regulatory Networks: The GNN component can reveal key regulatory
    interactions within the gene network.
Latent Space Analysis: The VAE's latent space can be visualized to identify
    patterns corresponding to biological conditions.

# Training Strategy

## Overview
The training strategy is designed to optimize the model's ability to predict post
    -perturbation gene expression profiles while ensuring generalization to
    unseen perturbations and cell contexts. The strategy incorporates several key
     components including a custom loss function, advanced optimization
    techniques, and mechanisms for monitoring and preventing overfitting.

## Key Components

### 1. Loss Function
**Components**:
- **Reconstruction Loss**: Measures the difference between the predicted and true
     gene expression profiles using Mean Squared Error (MSE).
- **KL Divergence Loss**: Ensures the learned latent space distribution from the
    VAE component is close to a prior distribution (typically a standard normal
    distribution).

**Implementation**:
```python
def loss_function(output, x, mu, logvar, beta=1.0):
   recon_loss = F.mse_loss(output, x)
   kl_loss = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())
   kl_loss = kl_loss / x.size(0) # Normalize by batch size
   total_loss = recon_loss + beta * kl_loss
   return total_loss, recon_loss, kl_loss
```

### 2. Optimizer and Learning Rate Scheduler
**Components**:
- **AdamW Optimizer**: Used for its adaptive learning rate capabilities and
    decoupled weight decay.
- **OneCycleLR Scheduler**: Dynamically adjusts the learning rate over the
    training period to find an optimal balance between learning rate and
    convergence speed.

**Implementation**:
```python
optimizer = torch.optim.AdamW(model.parameters(), lr=1e-3, weight_decay=1e-5)
scheduler = torch.optim.lr_scheduler.OneCycleLR(
   optimizer,
   max_lr=1e-3,
   steps_per_epoch=len(train_loader),
   epochs=num_epochs,
   pct_start=0.2,
   anneal_strategy='cos'
)
```

### 3. Regularization Techniques
**Components**:
- **Gradient Clipping**: Prevents exploding gradients by clipping gradient values
     during backpropagation.
- **Early Stopping**: Monitors validation loss and stops training if it doesn't
    improve for a specified number of epochs.

**Implementation**:
```python
def train_epoch(model, train_loader, optimizer, device, beta):
   model.train()
   total_loss = 0
   for batch in train_loader:
````
````

```
        x, pert = batch
        x, pert = x.to(device), pert.to(device)

        optimizer.zero_grad()

        output, mu, logvar = model(x, pert)

        loss, _, _ = loss_function(output, x, mu, logvar, beta)

        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
        optimizer.step()

        total_loss += loss.item()

    return total_loss / len(train_loader)

def validate(model, val_loader, device, beta):
    model.eval()
    total_loss = 0
    with torch.no_grad():
        for batch in val_loader:
            x, pert = batch
            x, pert = x.to(device), pert.to(device)

            output, mu, logvar = model(x, pert)

            loss, _, _ = loss_function(output, x, mu, logvar, beta)
            total_loss += loss.item()

    return total_loss / len(val_loader)

def train_model(model, train_loader, val_loader, optimizer, scheduler, device,
    num_epochs=100, beta=1.0, patience=5):
    best_loss = float('inf')
    patience_counter = 0
    best_model_weights = None

    for epoch in range(num_epochs):
        train_loss = train_epoch(model, train_loader, optimizer, device, beta)
        val_loss = validate(model, val_loader, device, beta)

        scheduler.step()

        print(f'Epoch {epoch+1}/{num_epochs} - Train Loss: {train_loss:.4f} - Val
            Loss: {val_loss:.4f}')

        if val_loss < best_loss:
            best_loss = val_loss
            patience_counter = 0
            best_model_weights = model.state_dict()
        else:
            patience_counter +=1
            if patience_counter >= patience:
                print('Early stopping triggered')
                break

    model.load_state_dict(best_model_weights)
    return model
```

4. Training Loop with Monitoring
Components:
- **Epoch Training**: Trains the model on the entire training dataset for one pass.
- **Validation**: Evaluates the model on the validation set after each epoch.
- **Checkpointing**: Saves the model weights if the validation loss improves.

Implementation:
```python
# Example usage
if __name__ == "__main__":
    # Assuming adata is your preprocessed AnnData object
    # Convert your data to PyTorch tensors
```

```
    X = torch.tensor(data_augmented, dtype=torch.float32)
    perturbations = torch.tensor(perturbations, dtype=torch.float32)

    # Create dataset and dataloader
    dataset = TensorDataset(X, perturbations)
    train_size = int(0.8 * len(dataset))
    test_size = len(dataset) - train_size
    train_dataset, test_dataset = torch.utils.data.random_split(dataset, [
        train_size, test_size])

    train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
    val_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)

    # Initialize model, optimizer, and device
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model = HybridModel(input_dim=pca_dim, pert_dim=pert_dim).to(device)
    optimizer = torch.optim.AdamW(model.parameters(), lr=1e-3, weight_decay=1e-5)
    scheduler = torch.optim.lr_scheduler.OneCycleLR(
        optimizer,
        max_lr=1e-3,
        steps_per_epoch=len(train_loader),
        epochs=100,
        pct_start=0.2,
        anneal_strategy='cos'
    )

    # Train the model
    trained_model = train_model(model, train_loader, val_loader, optimizer,
        scheduler, device, beta=0.1)
```

```
## Expert Recommendations Incorporated

### Addressing Class Imbalance
- **Weighted Loss**: Assign higher weights to underrepresented classes in the
    loss function.
- **Oversampling**: Increase the sampling rate of underrepresented perturbations
    during training.

### Enhancing Generalization
- **Data Augmentation**: Apply techniques like Gaussian noise addition and random
     masking to improve model robustness.
- **VAE Regularization**: Use the KL divergence term in the loss function to
    ensure the latent space is well-regularized.

### Improving Training Stability
- **Gradient Clipping**: Prevents gradients from exploding during backpropagation.

- **Learning Rate Scheduling**: Uses a OneCycle schedule to dynamically adjust
    the learning rate.

### Biological Interpretability
- **Attention Mechanisms**: The transformer component provides attention weights
    that can be analyzed to understand gene interactions.
- **Perturbation Embeddings**: The learned embeddings can be visualized to
    explore relationships between different perturbations.

This comprehensive training strategy ensures the model is trained effectively
    while addressing the specific challenges of single-cell gene expression data,
     including class imbalance, high dimensionality, and the need for
    generalization to unseen perturbations. The incorporation of expert
    recommendations further enhances the model's performance and biological
    relevance.
```

G.6. Example model code

```
graph TD
A[Raw Gene Expression Data] --> B[Filter Low-Quality Cells and Genes]
B --> C[Normalize Data]
C --> D[Log-Transformation]
D --> E[Batch Effect Correction]
E --> F[Feature Selection]
F --> G[PCA Reduction]
```

```
   H[Perturbation Data] --> I[One-Hot Encoding]
   G --> J[VAE Encoder]
   I --> K[Perturbation Embedding]
   J --> L[Fusion Layer]
   K --> L
   L --> M[Transformer Encoder]
   M --> N[Output Layer]
   N --> O[Predicted Expression Profile]



   // Data Preprocessing
function preprocess_data(adata, pca_dim):
   // Filter low-quality cells and genes
   filter_cells(adata, min_genes=200)
   filter_genes(adata, min_cells=3)

   // Normalize data
   normalize_total(adata, target_sum=1e4)
   log1p(adata)

   // Identify highly variable genes
   highly_variable_genes(adata, n_top_genes=3000)
   data = adata[:, adata.var['highly_variable']].X

   // Batch effect correction
   combat(adata, key='batch')

   // Standardize data
   scaler = StandardScaler()
   data = scaler.fit_transform(data)
   data = clip_values(data, -10, 10)
   data = scale(data)

   // PCA reduction
   pca = PCA(n_components=pca_dim)
   data_pca = pca.fit_transform(data)

   // Perturbation encoding
   perturbations = one_hot_encode(adata.obs['perturbation'])

   // Data augmentation
   data_augmented = add_gaussian_noise(data_pca)
   data_augmented = apply_random_masking(data_augmented)

   return data_augmented, perturbations

// Model Training
function train_model(model, train_loader, optimizer, device, beta):
   model.train()
   total_loss = 0
   for each batch in train_loader:
      x, pert = batch
      x, pert = move_to_device(x, pert, device)

      optimizer.zero_grad()

      output, mu, logvar = model(x, pert)

      recon_loss = mean_squared_error(output, x)
      kl_loss = compute_kl_loss(mu, logvar)
      loss = recon_loss + beta * kl_loss

      loss.backward()
      optimizer.step()

      total_loss += loss.item()

   return total_loss / number_of_batches(train_loader)



'''python
class VAEEncoder(nn.Module):
   def __init__(self, input_dim, latent_dim, hidden_dim):
```

```
        super().__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.fc_mu = nn.Linear(hidden_dim, latent_dim)
        self.fc_logvar = nn.Linear(hidden_dim, latent_dim)

    def forward(self, x):
        h = F.gelu(self.fc1(x))
        mu = self.fc_mu(h)
        logvar = self.fc_logvar(h)
        std = torch.exp(0.5 * logvar)
        eps = torch.randn_like(std)
        z = mu + eps * std
        return z, mu, logvar

class VAEDecoder(nn.Module):
    def __init__(self, latent_dim, output_dim, hidden_dim):
        super().__init__()
        self.fc1 = nn.Linear(latent_dim, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, z):
        h = F.gelu(self.fc1(z))
        return self.fc2(h)

class PerturbationEmbedding(nn.Module):
    def __init__(self, pert_dim, emb_dim):
        super().__init__()
        self.embedding = nn.Linear(pert_dim, emb_dim)

    def forward(self, pert):
        return self.embedding(pert)

class HybridAttentionModel(nn.Module):
    def __init__(self, input_dim, pert_dim, hidden_dim=512, n_layers=2, n_heads=8,
        dropout=0.1, attention_dropout=0.1, ffn_dropout=0.1, activation='gelu',
        use_transformer=True, use_vae=False, vae_latent_dim=64, vae_hidden_dim=256,
        use_pert_emb=False, pert_emb_dim=32, vae_beta=1.0):
        super(HybridAttentionModel, self).__init__()
        self.input_dim = input_dim
        self.pert_dim = pert_dim
        self.hidden_dim = hidden_dim
        self.use_transformer = use_transformer
        self.use_vae = use_vae
        self.vae_beta = vae_beta
        self.use_pert_emb = use_pert_emb

        if use_vae:
            self.vae_encoder = VAEEncoder(input_dim, vae_latent_dim, vae_hidden_dim)
            self.vae_decoder = VAEDecoder(vae_latent_dim, input_dim, vae_hidden_dim)
            expr_out_dim = vae_latent_dim
        else:
            self.expression_encoder = nn.Sequential(
                nn.Linear(input_dim, hidden_dim),
                nn.LayerNorm(hidden_dim),
                nn.GELU(),
                nn.Dropout(dropout),
                nn.Linear(hidden_dim, hidden_dim),
                nn.LayerNorm(hidden_dim),
                nn.GELU(),
                nn.Dropout(dropout)
            )
            expr_out_dim = hidden_dim

        if use_pert_emb:
            self.pert_encoder = PerturbationEmbedding(pert_dim, pert_emb_dim)
            pert_out_dim = pert_emb_dim
        else:
            self.pert_encoder = nn.Sequential(
                nn.Linear(pert_dim, hidden_dim),
                nn.LayerNorm(hidden_dim),
                nn.GELU(),
                nn.Dropout(dropout)
            )
            pert_out_dim = hidden_dim
```

```
        if use_transformer:
            encoder_layer = nn.TransformerEncoderLayer(
                d_model=expr_out_dim + pert_out_dim,
                nhead=n_heads,
                dim_feedforward=hidden_dim * 4,
                dropout=ffn_dropout,
                activation=activation,
                batch_first=True,
                norm_first=True
            )
            self.transformer = nn.TransformerEncoder(encoder_layer, num_layers=n_layers)
        else:
            self.self_attention = nn.MultiheadAttention(expr_out_dim + pert_out_dim,
                n_heads, dropout=attention_dropout, batch_first=True)

        self.fusion = nn.Sequential(
            nn.Linear(expr_out_dim + pert_out_dim, hidden_dim),
            nn.LayerNorm(hidden_dim),
            nn.GELU(),
            nn.Dropout(dropout)
        )

        self.output = nn.Sequential(
            nn.Linear(hidden_dim, hidden_dim),
            nn.LayerNorm(hidden_dim),
            nn.GELU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim, input_dim)
        )

        self.perturbation_head = nn.Sequential(
            nn.Linear(hidden_dim, hidden_dim // 2),
            nn.LayerNorm(hidden_dim // 2),
            nn.GELU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim // 2, pert_dim)
        )

        self.apply(self._init_weights)

    def _init_weights(self, module):
        if isinstance(module, nn.Linear):
            nn.init.xavier_uniform_(module.weight)
            if module.bias is not None:
                nn.init.zeros_(module.bias)
        elif isinstance(module, nn.LayerNorm):
            nn.init.ones_(module.weight)
            nn.init.zeros_(module.bias)

    def forward(self, x, pert):
        vae_kl = 0
        vae_recon = None

        if self.use_vae:
            z, mu, logvar = self.vae_encoder(x)
            vae_recon = self.vae_decoder(z)
            vae_kl = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp(), dim=1).mean()
            expr_feat = z
        else:
            expr_feat = self.expression_encoder(x)

        pert_feat = self.pert_encoder(pert)

        fusion_input = torch.cat([expr_feat, pert_feat], dim=1).unsqueeze(1)

        if self.use_transformer:
            x_trans = self.transformer(fusion_input).squeeze(1)
        else:
            x_trans, _ = self.self_attention(fusion_input, fusion_input, fusion_input)
            x_trans = x_trans.squeeze(1)

        fused = self.fusion(x_trans)
        output = self.output(fused)
        pert_pred = self.perturbation_head(fused)
```

```
        return output, pert_pred, vae_recon, vae_kl
```

H. Detailed outputs from other research Agents

DeepResearch Input

Task Desciption Input

```
Your task is to develop a predictive model that accurately estimates gene
    expression profiles of individual K562 cells following CRISPR interference (
    CRISPRi), using the dataset from Norman et al. (2019, Science).

Dataset Description:
- Source: Norman et al., 2019
- Cell Type: Human K562 leukemia cells
- Perturbations: CRISPRi targeting 105 single genes and 131 gene pairs
- Scale: ~90,000 single-cell RNA-seq profiles, including both control and
    perturbed conditions
Please search the following websets for more detailed information:
- https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE133344
- https://www.science.org/doi/10.1126/science.aax4438

Task Definition:
- Input: Baseline gene expression profile of an unperturbed K562 cell and the
    identity of the target gene(s) for perturbation
- Output: Predicted gene expression profile after perturbation

Evaluation Scenarios:
1. Unseen Perturbations: Predict effects of gene perturbations not present during
     training
2. Unseen Cell Contexts: Predict responses in cells with gene expression profiles
     not observed during training

Evaluation Metrics:
- Mean Squared Error (MSE): Measures the average squared difference between
    predicted and observed gene expression.
- Pearson Correlation Coefficient (PCC): Quantifies linear correlation between
    predicted and observed profiles.
- R$^2$ (Coefficient of Determination): Represents the proportion of variance in
    the observed gene expression that can be explained by the predicted values.
- MSE for Differentially Expressed (DE) Genes (MSE_DE): Same as MSE but computed
    specifically for genes identified as differentially expressed.
- PCC for Differentially Expressed (DE) Genes (PCC_DE): Same as PCC but computed
    specifically for genes identified as differentially expressed.
- R$^2$ for Differentially Expressed (DE) Genes (R$^2$_DE): Same as R$^2$ but
    computed specifically for genes identified as differentially expressed.

Please give me a task analysis report, a new method plan, and generate prediction
     model code.
```

OpenAI DeepResearch Output

DeepResearchh Output

```
## Task Analysis: Data Characteristics and Challenges

**Perturb-seq Dataset (Norman \*et al.\*, 2019):** The model is built on a rich
    single-cell CRISPR perturbation dataset from Norman *et al.* (2019, Science)
    (GEO: GSE133344). In this Perturb-seq experiment, K562 cells (a human
    leukemia line) were subjected to pooled CRISPR perturbations, measuring
    transcriptome-wide gene expression in each cell via single-cell RNA-seq. Key
    features of this dataset include:

- **Scale and Composition:** ~170,000 single cells were profiled, covering
    *~1,092 unique perturbation conditions* (including both single-gene and
    paired-gene perturbations). Each cell is annotated with one or two target
    genes perturbed (via CRISPR interference/activation).

- **High-Dimensional Readouts:** Each cells gene expression profile spans ~20,000
     genes (full transcriptome). This yields a **high-dimensional output space**
     a vector of expression levels for thousands of genes per cell.
```

```
- **Perturbation Types:** Perturbations consist of **single-gene knockdowns** (
    CRISPRi) and **two-gene combinations**, enabling the study of genetic
    interactions. For each target gene, multiple sgRNAs were used (to control for
     guide-specific effects), and for two-gene combos, cells co-express two
    sgRNAs (one per target).

**Modeling Challenges:** Developing an accurate predictive model for this task
    entails several significant challenges:

- **Single-Cell Variability:** Even for the same perturbation, different cells
    can respond with variability in magnitude of gene expression changes. Each
    cell has an intrinsic baseline expression profile reflecting its state (cell
    cycle stage, stochastic gene expression, etc.).

- **Generalization to Unseen Perturbations:** A critical use-case is predicting
    outcomes for perturbations that were **never seen in training**. This
    includes:
      - *New target genes:* The model may be asked to predict the effect of
          knocking down a gene that was not perturbed in the training data. This
          is an extreme generalization challenge, as there is no direct training
          example for that genes perturbation. The model must leverage other
          information (e.g. the genes baseline expression or relationships to
          other genes) to infer its likely impact.
      - *New gene combinations:* Similarly, the model might see single-gene
          perturbations during training, but not a particular pair in combination.
           Or even harder, both genes in a pair might be novel (neither perturbed
           before). As reported in prior work, when one or both perturbed genes
          are unseen, prediction difficulty increases markedly. The model needs a
           strategy to generalize **inductively**, combining knowledge of
          individual genes to predict combined effects.

- **Accuracy vs. Interpretability:** We prioritize predictive accuracy. This
    means we are willing to employ a complex, black-box model (deep neural
    network) with many parameters if it yields better predictive performance,
    even if its not easy to interpret biologically. This choice acknowledges that
     the gene regulatory relationships underlying the perturbation responses are
    complex; a simpler interpretable model (like linear regression per gene)
    might not capture the non-linear interactions or context dependencies as well
     as a deep model. However, the complexity of the model also raises the risk
    of overfitting and requires careful regularization and validation.

In summary, the task is essentially learning a high-dimensional function: **f**(*
    baseline_expression*, *perturbation*)  *perturbed_expression*. It must learn
    the underlying gene network responses from data, handle combinatorial
    perturbation effects, and generalize to new genes and cell states. These
    challenges will guide the modeling decisions discussed below.

------
## Proposed Modeling Approach for Accuracy and Generalization

### Model Architecture and Input Representation

To tackle this problem, we propose a **neural network model** that takes two
    inputs  (1) the baseline gene expression profile of an unperturbed cell, and
    (2) the identity of the target gene(s) perturbed  and outputs the predicted
    post-perturbation gene expression profile. The architecture is designed to
    flexibly capture non-linear gene interactions and make use of prior gene
    relationship knowledge to generalize to unseen perturbations. The key
    components are:

- **Baseline Expression Encoder:** A front-end network that processes the
    baseline expression vector (dimension = *G* genes). Directly using the raw
    high-dimensional vector as input to a fully-connected network is feasible but
     could be inefficient. We will introduce an encoder (e.g., a feed-forward
    autoencoder or dimensionality reduction layer) that compresses the baseline
    gene expression profile into a **lower-dimensional latent representation**.
    For example, a few fully-connected layers with ReLU activation can reduce the
     ~20k-dimensional input to a dense ~512-dimensional embedding. This latent
    vector is intended to capture the cells overall state or context (e.g., if
    the cell is in a high-proliferation state, or has high expression of certain
    pathways, etc.). By encoding the baseline, the model can later modulate
```

```
    perturbation effects depending on these latent features.

- **Perturbation Encoder:** We represent the **perturbation identity** in a way
    that the model can easily utilize and generalize. Each target gene (from the
    set of ~100 possible targeted genes in the screen) is assigned either:
      - a one-hot vector (of length equal to the number of target genes) if a
          single gene is perturbed, or a multi-hot vector if multiple genes are
          perturbed (e.g. for a two-gene perturbation, the vector has ones in the
           positions corresponding to the two targeted genes). This binary
          indicator vector can then be fed through an embedding layer (a learned
          lookup table or a small fully-connected network) to produce a **
          perturbation embedding**. The embedding is a continuous vector (e.g.
          128-dimensional) that represents the effect of the perturbation in a
          latent space.
      - Alternatively, an **embedding per gene** approach can be used: we
          maintain a trainable embedding vector for each gene in the target set,
          and for a combination perturbation, we combine the embeddings of the
          individual genes (e.g. by summation or an attention mechanism). Using a
           learned embedding for each gene gives the model a chance to encode
          each genes characteristic perturbation impact. For a multi-gene
          perturbation, a simple summation assumes independence of effects, while
           a more sophisticated combination (see below) can capture interactions.

- **Combination Module:** The baseline context and perturbation effect must be
    integrated. We concatenate the baseline latent vector and the perturbation
    embedding vector into a single combined latent representation. This combined
    vector (of length ~640 in our example, if baseline latent is 512 and
    perturbation embedding 128) now contains information about where the cell
    started and what perturbation was applied. This is passed through further
    layers (a **fusion network**) to compute the output. For instance, a
    multilayer perceptron (MLP) with one or two hidden layers (e.g., 512 neurons,
     ReLU activation) can mix these features. This stage allows for non-linear
    interactions between cell state and perturbation  e.g., the effect of
    perturbing gene X might depend on the level of gene Y in the baseline state,
    which a multiplicative interaction in the MLP can learn.

- **Output Layer (Prediction Head):** The final layer of the network produces a
    vector of length *G* (the number of genes), which is the predicted post-
    perturbation expression for each gene. To ensure the model easily handles the
     fact that many genes dont change, we design the output to predict a **change
     (delta) from baseline** for each gene rather than an absolute expression. In
     practice, the network can output expression for each gene, and then this is
    added to the baseline input to yield the final predicted expression:
       E^post=Ebaseline+model(Ebaseline,P).\hat{E}_{\text{post}} = E_{\text{
           baseline}} + \Delta_{\text{model}}(E_{\text{baseline}}, P).
       This formulation makes it easier for the model to output zero for genes
           that should remain unchanged, and focuses on learning the deviations.
           It also implicitly grounds the prediction in the baseline level (so if
            a gene is high at baseline and not affected by the perturbation, the
           model can just output a near-zero change).

- **Non-linear Interaction Modeling:** While a simple concatenation of embeddings
     treats multi-gene perturbations roughly as an additive combination of single
     effects, we can enhance the model to capture **genegene interaction effects
    **. One idea is to use an **attention mechanism or gating** in the
    perturbation encoder: for example, if two genes A and B are perturbed,
    instead of just summing their embeddings, we pass them through an attention
    network that can learn a pairwise interaction term. Another approach is to
    include pairwise products of gene embeddings in the combined feature (
    allowing the network to learn a unique contribution for the pair *A&B* beyond
     A + B). Given that Norman *et al.* tested primarily pairwise perturbations,
    we can explicitly include a learned parameter or small network for each pair
    of genes in the training set to capture any deviation from additivity.
    However, to generalize to unseen pairs, a better strategy is to learn a **
    function** for combining embeddings (like attention) rather than a fixed
    lookup for each pair.

- **Incorporating Prior Knowledge (for Generalization):** To improve inductive
    generalization to unseen genes, we can draw inspiration from GEARS and
    similar methods. We could initialize or regularize the gene perturbation
```

```
    embeddings using external knowledge:
      - Use a **gene co-expression network** (computed from the baseline single-
          cell data or external data) as a graph, and pass gene embeddings
          through a Graph Neural Network (GNN) layer. This encourages genes that
          have similar roles or expression patterns to have embeddings that
          produce similar effects. Thus, if an unseen gene has a similar co-
          expression profile to a seen gene, the model might infer similar
          perturbation outcomes.
      - Use **pathway or GO (Gene Ontology) information** to place genes in a
          relational graph (as GEARS did with a GO-derived graph for perturbation
           embeddings). Two genes in the same pathway might be expected to
          produce related downstream effects; by training on the known genes, the
           model can generalize to a new gene by its connections in the GO graph.
           Technically, this can be done by adding a loss term that encourages
          the learned embedding to correspond to the genes position in the
          knowledge graph, or by a GNN that propagates influence from neighbors
          during training.
      - These additions make the model more complex but aim to imbue it with **
          biological inductive bias**: (i) genes with similar baseline functions
          yield similar perturbation responses, and (ii) genes in related
          pathways affect overlapping sets of genes when perturbed. We will
          prioritize implementing a simpler version (like using co-expression PCA
           or clusters to initialize embeddings), and note that full graph-based
          learning could further improve generalization if needed.

- **Residual Connections and Regularization:** We will include skip-connections
    wherever helpful (for example, the baseline input could be fed not only into
    the encoder but also concatenated directly to a later layer, or the output
    head could directly see the raw baseline as well, ensuring the model can
    easily learn identity for unchanged genes). Regularization techniques like
    dropout in the MLP layers, L2 weight decay, or even an auxiliary loss to
    reconstruct the baseline (autoencoder style) can be employed to prevent
    overfitting and encourage the model to learn meaningful latent features
    rather than memorizing training examples.

In essence, the architecture is a **conditioned deep neural network**: it
    conditions on the cells initial state and the perturbation, and produces an
    output state. This is somewhat analogous to an encoderdecoder model where the
     encoder is the baseline expression and the condition or control signal is
    the perturbation identity. Because accuracy is paramount, we allow a fairly
    large model with enough capacity to capture complex gene regulatory responses.


### Training Strategy for Accuracy and Generalization

With the architecture in place, we next focus on **training methodology**, as it
    greatly affects model generalization and performance:

- **Training Data Construction:** We will pair each perturbed cells data with a
    baseline profile as input. Since in the actual experiment we typically do not
     have a *pre-perturbation* measurement of the same cell, we have to simulate
    a baseline. We can use the expression profiles of control cells (non-
    targeting sgRNA) as proxies for baseline input. For each perturbed cell in
    the training set, we can randomly sample a control cells expression as the
    baseline input. This effectively assumes that any control cell is an example
    of an unperturbed state the perturbed cell *could* have come from. Over many
    samples, the model will learn to map from an average baseline state to the
    perturbed outcome. We can further refine this pairing by matching on cell
    state: e.g., cluster the baseline cells by expression and pick a baseline
    from the same cluster as the perturbed cells profile (minus the perturbation
    effect) to provide a closer starting point. However, random pairing with a
    large pool of controls adds variability that can help the model not to
    overfit a one-to-one mapping.

      - *Unseen cell context generalization:* By exposing the model to many
          different baseline samples paired with a given perturbation outcome (
          through random pairing), we train it to handle diverse baseline inputs
          for the same perturbation. This should improve the models robustness to
           any particular baseline context and enable generalization to new
          baseline profiles. Essentially, the model sees that the same
          perturbation can apply to various starting expression patterns.
```

```
- **Loss Functions:** The primary loss will be **Mean Squared Error (MSE)**
    between the predicted and actual post-perturbation expression vectors. To
    ensure we adequately learn the important changes, we can modify the loss in
    two ways:

      - Compute a weighted MSE that gives higher weight to genes that are truly
          differentially expressed in that training example. For instance, if we
          know gene j changed significantly in the real perturbed cell (compared
          to baseline or compared to controls), we can upweight the error on gene
           j for that sample. This forces the model to focus on fitting the genes
           that move, rather than being dominated by the many near-zero changes.
      - Alternatively, we can train in two phases: first optimize MSE on the full
           profile to get general trends, then fine-tune the model on just the
          top-k DE genes for each perturbation (or using a loss like contrastive
          that emphasizes getting the direction of change correct).

      In practice, a simpler approach is to stick with standard MSE on all genes
          but monitor the top-k gene performance as a separate metric, ensuring
          the model doesnt ignore those signals. If we see the model predicting
          trivial (no-change) solutions, we will adjust the loss weighting.

- **Optimizer and Regularization:** We will use **Adam optimizer** (adaptive
    learning rate) which is well-suited for training deep networks on possibly
    noisy data. A relatively small learning rate (e.g. 1e-3 to start) will be
    used and well monitor validation loss for convergence. Early stopping on the
    validation MSE/PCC can prevent overfitting. Dropout layers (e.g. dropout rate
     0.2-0.5) can be inserted in the MLP to regularize. Weight decay (L2) will
    help keep embedding weights from growing excessively (important if some genes
     cause large effects, we dont want the weights to explode to fit one-hot
    signals).

- **Generalization Enhancements:** To specifically improve performance on **
    unseen genes**:

      - We will pre-train the **perturbation encoder embeddings**: one idea is to
           train a simple autoencoder or matrix factorization on the *observed
          perturbation effects* in training. For example, compute the average
          expression change each perturbation causes on each gene (from data),
          yielding a perturbation-gene effect matrix. We can factorize this
          matrix to get a low-dimensional embedding for genes and for
          perturbations. If a gene is unseen, we can still derive a gene
          embedding from its baseline expression pattern using the gene embedding
           from baseline encoder. This is a bit complex, but essentially we want
          the model to know something about gene function even if not perturbed.
          Incorporating gene ontology or known regulator information for unseen
          genes could also inform an initial embedding (e.g., if an unseen gene
          is a transcription factor similar to another seen TF, assign it a
          similar perturbation embedding).

By combining a carefully-designed architecture with these training strategies,
    our goal is a model that not only fits the training perturbations well but **
    generalizes to new perturbations and contexts**. In particular, success would
     be a model that, given an arbitrary K562 baseline profile and a novel target
     gene, can predict a biologically plausible outcome  capturing which pathways
     might be up or down-regulated by that genes perturbation, even without
    having seen it before. The inclusion of baseline input enables context-
    specific predictions (so the same gene perturbation could have different
    effects in different baseline states if the model learns those interactions),
     and the use of gene embeddings with potential graph-based inductive bias
    enables inductive generalization across genes.

Next, we present the implementation of this model in PyTorch, structured for GPU
    acceleration.

------
## PyTorch Implementation (Model & Training Code)

Below is the complete PyTorch code for the proposed model and an example training
     loop. This code assumes that the dataset has been preprocessed into tensors:
     `X_train` (baseline expressions), `Y_train` (perturbed expressions), and `
    P_train` (perturbation indicator vectors). The model is designed to run on a
```

GPU for efficiency.

```python
import torch
import torch.nn as nn
import torch.optim as optim

# Define device for GPU acceleration
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print("Using device:", device)

# Example shapes (for illustration; replace with actual dataset dimensions)
num_genes = X_train.shape[1] # dimensionality of gene expression (e.g. ~20000)
num_target_genes = P_train.shape[1] # dimensionality of perturbation vector (e.g. ~100)

# 1. Define the model architecture
class PerturbationPredictor(nn.Module):
    def __init__(self, input_dim_expr, input_dim_pert,
                 hidden_dim_expr=512, hidden_dim_pert=128, hidden_dim_comb=512):
        """
        input_dim_expr: number of genes (baseline expression vector length)
        input_dim_pert: number of possible targeted genes (length of perturbation indicator vector)
        """
        super(PerturbationPredictor, self).__init__()
        # Baseline expression encoder network
        self.expr_encoder = nn.Sequential(
            nn.Linear(input_dim_expr, hidden_dim_expr),
            nn.ReLU(),
            nn.Linear(hidden_dim_expr, hidden_dim_expr),
            nn.ReLU()
        )
        # Perturbation encoder network
        self.pert_encoder = nn.Sequential(
            nn.Linear(input_dim_pert, hidden_dim_pert),
            nn.ReLU(),
            nn.Linear(hidden_dim_pert, hidden_dim_pert),
            nn.ReLU()
        )
        # Fusion network to combine baseline and perturbation representations
        self.comb_fc1 = nn.Linear(hidden_dim_expr + hidden_dim_pert, hidden_dim_comb)
        self.comb_act = nn.ReLU()
        self.comb_fc2 = nn.Linear(hidden_dim_comb, input_dim_expr) # outputs delta for each gene

    def forward(self, baseline_expr, perturb_vec):
        # baseline_expr: Tensor of shape (batch_size, input_dim_expr)
        # perturb_vec: Tensor of shape (batch_size, input_dim_pert)
        # Encode baseline and perturbation
        z_expr = self.expr_encoder(baseline_expr) # shape: (batch, hidden_dim_expr)
        z_pert = self.pert_encoder(perturb_vec) # shape: (batch, hidden_dim_pert)
        # Combine encoded representations
        z_comb = torch.cat([z_expr, z_pert], dim=1) # shape: (batch, hidden_dim_expr+hidden_dim_pert)
        h = self.comb_act(self.comb_fc1(z_comb)) # hidden layer
        delta = self.comb_fc2(h) # predicted change in expression
        # Add the predicted change (delta) to the baseline to get final prediction
        pred_expr = baseline_expr + delta
        return pred_expr

# Instantiate the model and move to GPU
model = PerturbationPredictor(num_genes, num_target_genes).to(device)
print(model)

# 2. Set up loss function and optimizer
criterion = nn.MSELoss() # Mean Squared Error loss
optimizer = optim.Adam(model.parameters(), lr=1e-3)

# 3. Training loop (example)
num_epochs = 20
```

````
batch_size = 256

# If data is large, use DataLoader for batching (assuming X_train, P_train,
    Y_train are TensorDatasets)
train_dataset = torch.utils.data.TensorDataset(X_train, P_train, Y_train)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
    shuffle=True)

for epoch in range(1, num_epochs+1):
   model.train()
   total_loss = 0.0
   for X_batch, P_batch, Y_batch in train_loader:
      # Move data to device
      X_batch = X_batch.to(device)
      P_batch = P_batch.to(device)
      Y_batch = Y_batch.to(device)
      # Forward pass
      pred = model(X_batch, P_batch)
      loss = criterion(pred, Y_batch)
      # Backpropagation
      optimizer.zero_grad()
      loss.backward()
      optimizer.step()
      total_loss += loss.item() * X_batch.size(0)
   avg_loss = total_loss / len(train_dataset)
   if epoch % 5 == 0 or epoch == 1:
      print(f"Epoch {epoch:02d}: Training MSE = {avg_loss:.4f}")

# 4. Evaluation on test set
model.eval()
with torch.no_grad():
   X_test = X_test.to(device)
   P_test = P_test.to(device)
   Y_test = Y_test.to(device)
   pred_test = model(X_test, P_test)
   mse_full = criterion(pred_test, Y_test).item()
# Compute Pearson correlation coefficient (PCC) for each test sample
pred_np = pred_test.cpu().numpy()
Y_np = Y_test.cpu().numpy()
baseline_np = X_test.cpu().numpy()
pcc_list = []
topk = 20
topk_pcc_list = []
for i in range(Y_np.shape[0]):
   true_expr = Y_np[i]
   pred_expr = pred_np[i]
   # PCC for all genes
   cov = np.cov(true_expr, pred_expr, bias=True)
   # cov matrix 2x2: cov[0,1] is covariance between true and pred
   pcc = cov[0,1] / (np.std(true_expr) * np.std(pred_expr) + 1e-8)
   pcc_list.append(pcc)
   # PCC for top-k differentially expressed genes
   # Identify top-k genes by absolute change in true expression vs baseline
   base_expr = baseline_np[i]
   diff = np.abs(true_expr - base_expr)
   topk_idx = np.argsort(diff)[-topk:]
   if topk > 0:
      true_top = true_expr[topk_idx]
      pred_top = pred_expr[topk_idx]
      cov_top = np.cov(true_top, pred_top, bias=True)
      topk_pcc = cov_top[0,1] / (np.std(true_top) * np.std(pred_top) + 1e-8)
      topk_pcc_list.append(topk_pcc)
# Calculate mean metrics
mean_pcc = float(np.mean(pcc_list))
mean_topk_pcc = float(np.mean(topk_pcc_list))
print(f"Test MSE (all genes): {mse_full:.4f}")
print(f"Test mean PCC (all genes): {mean_pcc:.3f}")
print(f"Test mean PCC (top-{topk} DE genes): {mean_topk_pcc:.3f}")
```

**Explanation of the Code:**
````
````

```
- We define a `PerturbationPredictor` model class that implements the
    architecture described. The baseline expression encoder (`expr_encoder`) and
    perturbation encoder (`pert_encoder`) are simple feed-forward networks. These
     could be extended or replaced with more complex sub-networks (e.g., adding
    dropout, or using a graph convolution in `pert_encoder` if incorporating a
    gene network). The combined representation is fed through two linear layers (`
    comb_fc1` and `comb_fc2`) with a ReLU in between. The output of `comb_fc2` is
     a vector of length equal to number of genes, representing the predicted
    change in expression. We then add this to the baseline (`pred_expr = baseline
     + delta`) to get the final prediction.
- We move the model to GPU (`.to(device)`). The training loop iterates over mini-
    batches of data (using a DataLoader for efficiency). For each batch, we do a
    forward pass to compute predictions and then use `nn.MSELoss` to compute the
    loss against the true perturbed expression. We then backpropagate (`loss.
    backward()`) and update weights with Adam optimizer. We print the training
    MSE every few epochs for monitoring.
- In the evaluation section, after training, we compute metrics on `X_test,
    P_test, Y_test`. We calculate the overall MSE on test (as `mse_full`). Then
    for each test sample, we compute the Pearson correlation coefficient (PCC)
    between predicted and true expression. We do this manually using numpy:
    computing the covariance and standard deviations (note: one could also use `
    scipy.stats.pearsonr` or torchmetrics for this). We also compute **top-20 PCC
    ** by finding the 20 genes with largest absolute change in the true data (
    compared to baseline) and computing PCC on that subset. Finally, we print the
     average PCC across all test cells, and the average top-20 PCC.

**Note:** In practice, you would likely use vectorized operations or library
    functions to compute PCC for all cells at once. Also, you might evaluate the
    model separately on the different generalization scenarios (e.g., one test
    set for unseen genes, one for seen genes, etc.) and compare performance. The
    code above provides a template that can be adapted for those purposes.

By executing this model training pipeline, we expect to obtain a model that can
    accurately predict gene expression changes in K562 cells after CRISPRi
    perturbations. Its performance will be reflected in low MSE (especially on
    perturbed genes) and high correlation with actual profiles, even for
    perturbations or cell states not seen in training. Ultimately, this
    demonstrates a powerful approach to **in silico** predict cellular response
    to genetic perturbations, leveraging modern deep learning (PyTorch with GPU)
    for both precision and generalization. The combination of baseline context
    and perturbation identity as inputs allows the model to capture conditional
    dependencies (how a perturbations effect can vary by cell state), addressing
    one of the key challenges in predictive biology.
```

Perplexity DeepResearch

DeepResearchh Output

```
Task Analysis and Technical Challenges

I. Biological Context of CRISPRi Perturbations

The Norman et al. dataset captures transcriptional consequences of CRISPRi-
    mediated gene repression through single-cell profiling of:

- 105 individual gene knockouts
- 131 pairwise gene interactions[^1][^4]
- ~90,000 cells with baseline (unperturbed) and perturbed states[^4]

Key biological considerations:

1. **Non-linear genetic interactions**: Pairwise perturbations exhibit emergent
    effects not predictable from individual gene knockouts[^2]
2. **Cell-state heterogeneity**: Native transcriptional variability in K562
    populations influences perturbation responses[^2][^4]
3. **Technical noise characteristics**: Dropout patterns and UMI count
    distributions differ between perturbed/control cells[^4][^6]
```

```
II. Machine Learning Challenges

1. **High-dimensional output space**: Predicting 20,000+ gene expressions
    simultaneously
2. **Sparse perturbation coverage**: Only 0.6% of possible gene pairs tested in
    original study[^1]
3. **Context-dependent effects**: Perturbation responses vary with baseline
    cellular states[^2][^5]
4. **Combinatorial explosion**: Quadratic growth of possible gene interactions
    with target count

**Evaluation Protocol Design**

**Train/Test Splits**:

- **Unseen perturbations**: Hold out 20% of single-gene and 30% of pairwise
    perturbations
- **Unseen contexts**: Stratify by baseline PCA clusters, hold out entire cell-
    state subpopulations

------

Methodological Framework: GenePerturbNet

I. Architectural Overview

**Core Components**:

1. **Baseline Encoder**: Variational autoencoder mapping unperturbed expression
    $x$ to latent $z$.

2. **Perturbation Transformer**: Attention network processing target genes $g$
    into perturbation vector \delta.

3. **Interaction Module**: Graph neural network modeling gene-gene interaction
    effects.

4. **Response Decoder**: generating perturbed expression profile


II. Training Strategy

**Loss Function:**

$\mathcal{L} = ELBO(x,y) + \lambda_1 MSE(y, \hat{y}) + \lambda_2 \mathcal{L}_{
    contrast}$

Where:

- **Contrastive Loss**:
      $\mathcal{L}_{contrast} = -\log\frac{\exp(s(y,\hat{y})/\tau)}{\sum_{y'}\exp
          (s(y',\hat{y})/\tau)}$

- $s()$: Cosine similarity

- $\tau$: Temperature parameter

IV. Curriculum Learning Schedule:
1. Phase 1: Pretrain on single-gene perturbations

2. Phase 2: Fine-tune with pairwise interactions

3. Phase 3: Joint optimization with contrastive loss


Implementation and Benchmarking

I. Code Implementation

'''python
import scanpy as sc
import scvi
import torch
from torch import nn
from scvi import REGISTRY_KEYS
from scvi.module.base import BaseModuleClass
```

````
class GenePerturbNet(scvi.model.SCVI):
   def __init__(self, adata, n_hidden=256, n_latent=32,
                perturbation_embedding=64, **kwargs):
      super().__init__(adata, **kwargs)
      self.pert_embed = nn.Embedding(adata.n_vars, perturbation_embedding)
      self.transformer = nn.TransformerEncoderLayer(
         d_model=perturbation_embedding, nhead=4
      )
      self.gnn = GeneInteractionGNN(perturbation_embedding)

      self.decoder = nn.Sequential(
         nn.Linear(n_latent + perturbation_embedding, n_hidden),
         nn.ReLU(),
         nn.Linear(n_hidden, adata.n_vars)
      )

   def forward(self, x, perturbations, *args):
      qz_m, qz_v, z = self.encoder(x)

      pert_emb = self.pert_embed(perturbations)
      delta = self.transformer(pert_emb)

      z_pert = z + delta.mean(dim=1)

      px_scale, px_r, px_rate, px_dropout = self.decoder(z_pert)

      return px_rate, qz_m, qz_v, z_pert
class GeneInteractionGNN(nn.Module):
   def __init__(self, embed_dim):
      super().__init__()
      self.conv1 = nn.Conv1d(embed_dim, embed_dim*2, kernel_size=3)
      self.attn = nn.MultiheadAttention(embed_dim, num_heads=4)

   def forward(self, gene_pairs):
      x = self.conv1(gene_pairs)
      x, _ = self.attn(x, x, x)
      return x.mean(dim=1)
```


II. Training Protocol

```python
adata = sc.read("norman_2019.h5ad")
scvi.model.setup_anndata(adata, layer="counts",
                  categorical_covariate_keys=["perturbation"])

model = GenePerturbNet(adata, n_latent=32, perturbation_embedding=64)
model.train(max_epochs=100, batch_size=256,
        plan_kwargs={"contrastive_weight":0.5})

control_cells = adata[adata.obs["perturbation"] == "control"]
perturbed_pred = model.predict(control_cells, targets=["geneA", "geneB"])
```

III. Conclusion and Future Directions

This framework demonstrates that deep learning architectures explicitly modeling genetic interactions can significantly improve perturbation prediction accuracy. Key recommendations:

1. **Data Expansion**: Incorporate epigenetic data to better model CRISPRi efficiency
2. **Dynamic Modeling**: Add temporal component to capture perturbation kinetics
3. **Transfer Learning**: Adapt model to other cell types using few-shot learning

The provided codebase offers a foundation for extending perturbation prediction to novel experimental designs and therapeutic applications.
````

Gemini DeepResearch
````

DeepResearchh Output

Task Analysis

Dataset Description

The foundation of this predictive modeling task lies in the comprehensive dataset generated by Norman et al. and published in Science in 2019. This study aimed to explore genetic interaction manifolds constructed from rich single-cell phenotypes using a Perturb-seq approach.15 The experiments were conducted on human K562 leukemia cells, a commonly used model system in hematological research. The study employed CRISPRi to target a total of 236 genetic perturbations, encompassing 105 single genes and 131 pairs of genes. The scale of the dataset is substantial, comprising approximately 90,000 single-cell RNA-seq profiles, which includes both control cells and cells subjected to various single and combinatorial CRISPRi perturbations.

The dataset's focus on both single and paired gene perturbations is crucial for training a model capable of predicting combinatorial effects, which are often non-additive and complex. Single gene perturbations provide baseline information about the effect of individual gene silencing. Paired perturbations allow the model to learn about genetic interactions, where the combined effect of perturbing two genes is different from the sum of their individual effects. This is essential for capturing the non-linear nature of biological systems.

Problem Definition

The primary objective is to develop a predictive model that takes as input the baseline gene expression profile of an unperturbed K562 cell and the identity of the target gene(s) for CRISPRi perturbation. The desired output of the model is the predicted gene expression profile of the same cell after the specified perturbation has been applied.

The model's performance will be evaluated under two key scenarios to assess its generalizability:

1. **Unseen Perturbations:** The model should be able to accurately predict the effects of CRISPRi targeting genes or gene pairs that were not included in the training data. This scenario tests the model's ability to extrapolate its learned knowledge to novel genetic manipulations.
2. **Unseen Cell Contexts:** The model should be capable of predicting the response to a perturbation in cells with baseline gene expression profiles that were not observed during the training phase. This evaluates the model's robustness to the inherent heterogeneity within the K562 cell population.

The evaluation scenarios highlight the need for a model that can generalize beyond the specific perturbations and cell states seen during training, which is a significant challenge in biological systems due to their complexity and inherent variability. A model trained only on observed perturbations might not accurately predict the effects of novel perturbations or responses in slightly different cellular contexts. Evaluating on unseen data is crucial for assessing the model's robustness and real-world applicability.

Challenges and Considerations

Developing an accurate predictive model for gene expression profiles following CRISPRi perturbation presents several significant challenges stemming from the inherent complexity of biological systems and the nature of single-cell data.

The gene expression data obtained from scRNA-seq experiments is inherently high-dimensional, with measurements for thousands of genes in each individual cell. This high dimensionality necessitates the use of sophisticated modeling techniques capable of handling a large number of input features. Furthermore, the underlying biological networks that govern gene expression are incredibly complex, involving intricate regulatory relationships between genes, transcription factors, and other cellular components.

The relationship between gene perturbations and the resulting changes in gene expression is often non-linear.1 Genetic interactions, such as synergistic or

```
     epistatic effects, mean that the combined effect of perturbing multiple
    genes cannot simply be predicted by summing the individual effects of each
    perturbation.1 The model must be able to capture these complex, non-additive
    relationships to make accurate predictions, especially for the 131 gene pairs
     targeted in the Norman et al. (2019) dataset.

While CRISPRi is generally considered to have fewer off-target effects compared
    to CRISPRko, the possibility of unintended gene repression at secondary
    genomic loci still exists. The specificity of CRISPR-Cas9 systems relies
    heavily on the design of the single guide RNA (sgRNA), and mismatches between
     the sgRNA and off-target DNA sequences can lead to unintended gene silencing.
     Although the Norman et al. (2019) study likely employed well-designed sgRNAs,
     the potential for some level of off-target activity should be considered.

Single-cell RNA-seq data is also characterized by inherent noise and technical
    variability.11 Factors such as cell lysis efficiency, reverse transcription
    efficiency, and sequencing depth can introduce variability in the measured
    gene expression levels. The predictive model needs to be robust to this noise
     and focus on capturing the true biological signals induced by the CRISPRi
    perturbations.

Finally, clonal expansion, a phenomenon where certain perturbed cells proliferate
     more rapidly than others, can occur during single-cell CRISPR screens. This
    can lead to an overrepresentation of certain perturbations in the sequencing
    data, potentially biasing the results and affecting the training of the
    predictive model. Computational methods exist to identify and potentially
    adjust for clonal cell expansion.

Accurately modeling the intricate web of gene regulatory interactions and the non
    -linear effects of perturbations requires sophisticated computational
    approaches that can capture these complexities beyond simple additive models.
     Biological systems are highly interconnected, and the effect of perturbing
    one gene can propagate through the network, affecting the expression of many
    other genes in non-predictable ways using linear models. Therefore, models
    capable of learning complex, non-linear relationships are needed.

------

New Method Plan

To address the challenges outlined above and develop an accurate predictive model
     for gene expression profiles following CRISPRi, a deep learning-based
    approach is proposed. Specifically, a Graph Neural Network (GNN) architecture,
     inspired by the success of models like GEARS, appears to be a promising
    candidate.

**Proposed Model Architecture**

The proposed model will leverage a GNN to incorporate prior knowledge about gene-
    gene relationships and model the perturbation effects as changes within this
    network. GNNs are well-suited for learning representations of nodes in a
    graph by aggregating information from their neighbors, allowing the model to
    capture the dependencies and interactions between genes.

The input to the model will consist of two components: (1) the baseline gene
    expression profile of an unperturbed cell, represented as a vector of gene
    expression counts, and (2) the identity of the target gene(s) for CRISPRi.
    For single gene perturbations, the target gene will be directly specified.
    For paired gene perturbations, the identities of both target genes will be
    provided.

The model architecture will comprise the following key components:

1. **Gene Embedding Layer:** Each gene in the dataset will be assigned a low-
    dimensional embedding vector. These embeddings will capture intrinsic
    properties of the genes and will be learned during the training process.
    Prior biological knowledge, such as gene co-expression networks or functional
     annotations from databases like Gene Ontology (GO) 36, can be used to
    initialize these embeddings or to inform the GNN architecture.

2. **Perturbation Embedding Layer:** The identity of the perturbed gene(s) will
    also be encoded into an embedding vector. For single perturbations, a
```

dedicated embedding will be learned for each targeted gene. For paired perturbations, the embeddings of the two target genes can be combined (e.g., through summation or concatenation) to represent the combined perturbation.

3. **Graph Neural Network (GNN):** A gene regulatory network (GRN) will be constructed, where genes are represented as nodes and edges represent regulatory relationships between them. This GRN can be derived from publicly available databases or inferred from the unperturbed single-cell expression data. The gene embeddings will serve as initial node features in this graph. The GNN will then propagate information across the network, allowing each gene's representation to be informed by its neighbors and their interactions. The perturbation embedding will be incorporated into the GNN, potentially by modifying the node features of the perturbed gene(s) or by influencing the message passing process.

4. **Cell State Encoding Layer:** The baseline gene expression profile of the unperturbed cell will be passed through a separate neural network (e.g., a multi-layer perceptron) to learn a low-dimensional representation of the cell's initial transcriptional state. This encoding will capture the cell's context and will be used to condition the prediction of the perturbed state.

5. **Prediction Layer:** The output of the GNN (representing the perturbed gene embeddings) and the cell state encoding will be combined (e.g., through concatenation followed by another neural network) to predict the gene expression profile after the perturbation. The output will be a vector of the same dimensionality as the input gene expression profile, representing the predicted expression levels for each gene in the cell.

The rationale behind choosing this architecture is that it allows for the integration of prior biological knowledge about gene-gene interactions through the GRN. This can help the model to better understand the potential downstream effects of a perturbation. Furthermore, the use of embeddings allows the model to learn meaningful representations of genes and perturbations, potentially enabling better generalization to unseen perturbations.

Feature Engineering and Data Preprocessing

The Norman et al. (2019) dataset will require careful preprocessing before being used to train the model. The steps involved will include:

1. **Data Loading and Normalization:** The processed gene expression matrices will be loaded using appropriate libraries like Scanpy or AnnData. The gene expression counts will be normalized to account for differences in sequencing depth between cells. Log transformation (e.g., using a natural logarithm after adding a pseudocount) will be applied to stabilize the variance of gene expression levels.

2. **Perturbation Information Encoding:** The perturbation information, specifying the targeted gene(s) for each cell, will be extracted from the dataset's metadata. For single gene perturbations, the gene name will be used. For paired gene perturbations, both gene names will be used. These gene names will then be mapped to their corresponding indices or identifiers in the gene expression matrix. The perturbation information will be encoded as input to the model, potentially using one-hot encoding initially, where a binary vector indicates which genes are targeted. Alternatively, learned embeddings for each gene could be used to represent the perturbation.

3. **Control Sample Handling:** Cells labeled as control (unperturbed) will be identified and used to establish the baseline gene expression profiles. These control profiles will be crucial for training the model to predict the changes in expression induced by the perturbations.

4. **Feature Selection:** Given the high dimensionality of the gene expression data, feature selection techniques may be employed to focus on the most relevant genes. One common approach is to identify highly variable genes (HVGs) across the cell population and use only these genes as input to the model.16 This can reduce the dimensionality of the input, potentially improving model training and performance.

5. **GRN Construction (if applicable):** If a GNN is used, a gene regulatory network will need to be constructed. This could involve using publicly

```
    available databases of known gene interactions or inferring a network from
    the unperturbed single-cell expression data using methods like co-expression
    analysis or network inference algorithms.

**Training Strategy**

The training of the predictive model will involve the following steps:

1. **Data Splitting:** The dataset will be split into training, validation, and
    test sets. To address the evaluation scenarios of unseen perturbations, the
    split will be performed at the level of perturbations. This means that all
    cells corresponding to certain perturbations (both single and paired) will be
     held out in the test set and will not be seen by the model during training.
    A separate validation set, also containing held-out perturbations, will be
    used for hyperparameter tuning and model selection.16 Carefully designing the
     data splitting strategy is crucial to ensure that the model is truly
    evaluated on unseen perturbations and cell contexts, avoiding information
    leakage from the training set.

2. **Loss Function:** The model will be trained to minimize the difference
    between the predicted gene expression profiles and the observed gene
    expression profiles. The Mean Squared Error (MSE) will be used as the primary
     loss function, as it directly measures the average squared difference
    between the predicted and observed values.

3. **Optimizer:** An appropriate optimization algorithm, such as Adam, will be
    used to update the model's parameters during training. A learning rate
    schedule, which gradually reduces the learning rate over time, may be
    employed to improve convergence and prevent overfitting.

4. **Hyperparameter Tuning:** The model architecture and training process will
    have several hyperparameters (e.g., the dimensionality of the embeddings, the
     number of layers in the neural networks, the learning rate). These
    hyperparameters will be tuned using the validation set. Techniques like grid
    search or random search can be used to explore different combinations of
    hyperparameters and select the configuration that yields the best performance
     on the validation set.

5. **Overfitting Prevention:** Techniques such as dropout, weight regularization
    (e.g., L1 or L2 regularization), and early stopping (monitoring the
    performance on the validation set and stopping training when it starts to
    degrade) will be used to prevent the model from overfitting to the training
    data and improve its generalization ability.

**Strategies for Handling Unseen Perturbations and Cell Contexts**

The proposed GNN-based architecture offers several advantages for handling the
    challenges of unseen perturbations and cell contexts:

- **Generalization to Unseen Genes (for GNN):** If a GNN is used and the GRN
    includes genes that are not targeted in the training set, the model may still
     be able to make informed predictions about the effects of perturbing these
    unseen genes by leveraging their relationships with other genes in the
    network.36 The model can learn general principles of how perturbations
    propagate through the network, allowing it to extrapolate to new nodes (genes
    ). Predicting the effects of completely novel perturbations (genes not seen
    during training) is a significant challenge.

- **Cell State Conditioning:** By explicitly encoding the baseline gene
    expression profile of the unperturbed cell, the model can condition its
    prediction on the specific context of that cell. This allows the model to
    capture some of the inherent heterogeneity within the cell population and
    potentially make more accurate predictions for cells with unseen baseline
    profiles.

- **Learned Embeddings:** The use of learned embeddings for genes and
    perturbations can help the model to capture semantic relationships between
    different genes and perturbations. If the embedding space is learned
    effectively, the model may be able to generalize to unseen perturbations that
     are functionally similar to those seen during training, even if the specific
     genes were not encountered before.
```

```
While more advanced techniques like meta-learning or domain adaptation could
    potentially further improve the model's ability to handle unseen
    perturbations and cell contexts, the proposed GNN architecture with cell
    state conditioning and learned embeddings provides a strong foundation for
    addressing these challenges.

------
Generate Prediction Model Code

The prediction model will be implemented using the Python programming language
    and several key libraries commonly used in machine learning and single-cell
    data analysis.

**Implementation Details**
- **Programming Language:** Python
- **Key Libraries:**
   - **PyTorch** or **TensorFlow/Keras:** For implementing the neural network
       architecture, including the GNN and other layers.
   - **Scanpy** or **AnnData:** For efficient handling and preprocessing of the
       single-cell RNA-seq data.
   - **NumPy:** For numerical computations and array manipulations.
   - **SciPy:** For scientific computing, including statistical functions.
   - **scikit-learn:** For evaluation metrics (MSE, PCC) and potential utility
       functions.

**Code Structure and Key Functions**
The codebase will be organized into several modules or classes to ensure
    modularity and maintainability:

1. **`DataLoader` Class:**
   - **Purpose:** Responsible for loading the Norman et al. (2019) dataset from
       the specified files (e.g., MTX, TSV).
   - **Functionality:**
      - Reads the gene expression matrix and metadata.
      - Performs normalization and log transformation.
      - Encodes perturbation information.
      - Splits the data into training, validation, and test sets based on
          perturbations.
      - Provides data loaders for efficient batching during training.
2. **`Model` Class:**
   - **Purpose:** Defines the neural network architecture, including the gene
       embedding layer, perturbation embedding layer, GNN (if chosen), cell state
        encoding layer, and prediction layer.
   - **Functionality:**
      - Initializes the model parameters.
      - Implements the forward pass of the network, taking baseline expression
          and perturbation information as input and outputting the predicted
          perturbed expression.
3. **`Trainer` Class:**
   - **Purpose:** Handles the training process of the model.
   - **Functionality:**
      - Takes the model, data loaders, loss function, and optimizer as input.
      - Implements the training loop, including forward pass, loss calculation,
          backpropagation, and parameter updates.
      - Monitors performance on the validation set and implements early stopping
          if needed.
      - Saves the trained model parameters.
4. **`Evaluator` Class:**
   - **Purpose:** Evaluates the performance of the trained model on the test set.
   - **Functionality:**
      - Loads the trained model parameters.
      - Iterates through the test data.
      - Makes predictions for each test sample.
      - Calculates the evaluation metrics (MSE, PCC, Top-k PCC) by comparing
          predictions with the observed data.
      - Reports the evaluation results.
5. **`Predictor` Class:**
   - **Purpose:** Allows for making predictions on new, unseen data (e.g., a
       baseline expression profile and a perturbation).
   - **Functionality:**
      - Loads the trained model parameters.
```

```
      - Takes a baseline expression profile and perturbation information as input.

      - Preprocesses the input data as needed.
      - Uses the trained model to generate the predicted gene expression profile
        after the perturbation.

**Instructions for Usage**

1. **Install Libraries:** Install the necessary Python libraries using pip:
   ```Bash
   pip install torch torchvision torchaudio scanpy numpy scipy scikit-learn
   ```
   (or `tensorflow` and `keras` if using TensorFlow).

2. **Download Data:** Download the processed data files for GSE133344 from GEO or
   Figshare and place them in a designated directory.

3. **Prepare Input Data:** Ensure that the input data for prediction (baseline
   gene expression profile and perturbation identity) is in the expected format
   (e.g., a NumPy array or a Pandas DataFrame).

4. **Run the Code:** Execute the Python script containing the model
   implementation, training, and evaluation steps. The script should:

   - Initialize the `DataLoader` with the path to the data files.
   - Split the data into training, validation, and test sets.
   - Initialize the `Model` with the appropriate architecture and hyperparameters.

   - Initialize the `Trainer` with the model, data loaders, loss function, and
     optimizer.
   - Train the model using the `Trainer`.
   - Initialize the `Evaluator` with the trained model and test data loader.
   - Evaluate the model and print the results.
   - (Optionally) Use the `Predictor` to make predictions on new data.

Evaluation and Results

A comprehensive evaluation of the developed predictive model is crucial to assess
   its performance and generalizability.

Experimental Setup

The Norman et al. (2019) dataset will be split into training, validation, and
   test sets, ensuring that perturbations are held out for both validation and
   testing to evaluate the model's ability to predict the effects of unseen
   genetic manipulations. The split will likely involve a significant portion of
   the data for training, with smaller portions allocated for validation (for
   hyperparameter tuning and model selection) and testing (for final performance
   evaluation).16 The specific hyperparameters of the chosen model architecture
   (e.g., number of layers, dimensionality of embeddings, learning rate) will
   be determined through experimentation and tuning on the validation set.

To make a prediction for a given perturbation, the model will take as input the
   baseline gene expression profile of an unperturbed cell (which could be an
   average profile of control cells or a specific control cell's profile) and
   the identity of the target gene(s).

Performance Metrics

The model's performance on the test set will be quantified using the three
   evaluation metrics defined earlier: Mean Squared Error (MSE), Pearson
   Correlation Coefficient (PCC), and Top-k PCC. These metrics will be
   calculated by comparing the model's predicted gene expression profiles with
   the actual observed profiles in the test set for the held-out perturbations.
   The results will be reported separately for unseen single-gene perturbations
   and unseen paired-gene perturbations to assess the model's ability to handle
   both types of genetic manipulations. It may also be informative to report the
   performance on different subsets of genes, such as the highly variable genes,
   as these are often the most biologically relevant. Visualizations, such as
   scatter plots of predicted vs. observed gene expression for representative
   perturbations, can provide further insights into the model's predictive
   capabilities.
```

I. LLM-as-a-judge details

I.1. methods

To assess the quality of task analysis reports and research plans generated by various scAgents, we employed a Large Language Model (LLM) as an automated evaluator.

Outputs(Examples can be found in Appendix G) from scAgents(with 5 different LLM APIconfigurations: Claude 3.7, o1, DeepSeek R1, Qwen-plus, Llama 3,1) were anonymized to prevent bias. For each evaluation round, a set of 8 outputs was randomly selected and their order randomized to ensure fairness. This process was repeated 5 times, resulting in 5 distinct evaluation rounds with different output sequences. In each round, the LLM evaluated the 8 outputs individually, assigning a score from 1 to 10 based on predefined criteria. The LLM was unaware of the source of each output, ensuring unbiased assessments.

The LLM was guided using a structured prompt that specified the evaluation criteria and scoring rubric. An example prompt is as follows.

I.2. prompts

LLM As Judge-Gene

Task Analysis

```
You are an expert evaluator specializing in data-driven analysis of CRISPR-based
    single-cell perturbation experiments. Your background includes:

- In-depth knowledge of single-cell omics data modalities (e.g., RNA-seq, ATAC-
    seq, CITE-seq)
- Experience in characterizing perturbation types and experimental settings
- Familiarity with agent-based literature retrieval and scientific reasoning
- Ability to assess baseline model performance in biological prediction tasks
- Understanding of automated systems for scientific task decomposition

Please evaluate the following Task Analysis report using rigorous and objective
    scientific standards. You may receive multiple reports in randomized order
    across five rounds. **Evaluate each report independently**, without assuming
    knowledge of other submissions.

EVALUATION CRITERIA
Each criterion should be scored on a scale of 1~10, with clear justification
    based on the report content. Use full-score ranges (1~10) where appropriate.

1. Analyse Dataset (1~10):
- Clarity and correctness in summarizing dataset properties (modality,
    perturbation type, species, cell type distribution, etc.)
- Relevance of identified features for downstream modeling
- Ability to standardize and interpret metadata across modalities
- Quality of data summaries and diagnostic insights (e.g., sparsity,
    heterogeneity)

2. Analyse Task Type (1~10):
- Accuracy in identifying the biological question and mapping it to a
    computational prediction task
- Insightfulness in selecting the right task framing (e.g., classification vs
    regression, single-cell vs population-level)
- Alignment of task framing with perturbation mechanism and data granularity
- Ability to distinguish this task from superficially similar ones

3. Analyse Baseline Defects (1~10):
- Thoroughness in identifying limitations of current baseline models
- Correctness in linking model weaknesses to data/task-specific challenges (e.g.,
     model mismatch with modality, lack of interpretability)
- Thoughtfulness in proposing key evaluation gaps or unaddressed risks
- Clarity in explaining why the baseline is insufficient and what improvement
    directions are needed

FORMAT FOR YOUR EVALUATION:
1. NUMERICAL SCORES
```

```
Analyse Dataset: [Score]/10
Analyse Task Type: [Score]/10
Analyse Baseline Defects: [Score]/10

2. DETAILED JUSTIFICATION
Provide specific and concise reasoning for each score, referencing relevant parts
     of the analysis. Address both strengths and limitations within each
    criterion.

3. KEY STRENGTHS
[List major strengths of the Task Analysis report]
[Reference specific elements that demonstrate scientific rigor or originality]

4. AREAS FOR IMPROVEMENT
[Identify specific aspects that could be clarified or strengthened]
[Offer constructive, actionable suggestions for refinement]

5. OVERALL RECOMMENDATION
Provide a concise overall assessment. Consider:
- Does the Task Analysis provide a strong foundation for follow-up modeling?
- Are the dataset and task features well-characterized and actionable?
- Are the limitations of baseline models accurately diagnosed and explained?

REMINDERS:
- Maintain scientific neutrality and avoid assumptions not grounded in the
    provided text.
- Consider both biological and computational aspects equally.
- Provide constructive feedback aimed at improving scientific understanding.
- Use current SOTA practices in perturbation modeling and single-cell analysis as
     your reference frame.
- Assume the audience is a mix of computational biologists, experimentalists, and
     system developers.

TASK ANALYSIS REPORT TO EVALUATE:
[Paste your report here] / [Will be provided in the next message]
```

LLM As Judge-Gene

Method Design

```
You are an expert evaluator specializing in CRISPR-based single-cell perturbation
     prediction models and experimental designs. Your background includes:

- Deep expertise in computational biology and single-cell omics
- Practical experience with CRISPR-based perturbation experiments
- Familiarity with multimodal single-cell datasets (e.g., gene expression, ATAC-
    seq, protein expression)
- Advanced understanding of machine learning models for biological prediction
    tasks
- Knowledge of statistical validation methods and experimental reproducibility
    standards
- Awareness of recent state-of-the-art (SOTA) approaches in perturbation modeling

Please evaluate the following research plan using rigorous and objective
    scientific standards. You may receive multiple plans in randomized order
    across five rounds. **Evaluate each plan independently**, without assuming
    knowledge of other submissions.

EVALUATION CRITERIA
Each criterion should be scored on a scale of 1~10, with clear justification
    based on the content of the plan. Use full-score ranges (1~10) where
    appropriate.

1. Scientific Validity (1~10):
- Biological relevance and mechanistic insight
- Strength of theoretical foundation
- Alignment with current scientific understanding in single-cell biology

Integration with existing knowledge on perturbation responses

2. Technical Feasibility (1~10):
```

```
- Practicality of implementation
- Computational resource requirements
- Scalability to larger datasets or new tasks
- Feasibility and clarity of data preprocessing or modeling pipeline

3. Innovation Level (1~10):
- Novelty compared to current state-of-the-art approaches
- Creative problem-solving or hypothesis generation
- Potential for new biological or computational insights
- Unique contributions in methodology or design

4. Experimental Design (1~10):
- Quality of proposed validation and evaluation methodology
- Inclusion of appropriate controls and baselines
- Statistical soundness (e.g., replicates, robustness)
- Attention to data quality and reproducibility

5. Impact Potential (1~10):
- Relevance and contribution to advancing single-cell biology
- Translational potential (e.g., drug discovery, therapeutic design)
- Scalability to broader biological questions or contexts
- Potential to inspire follow-up research or community adoption

FORMAT FOR YOUR EVALUATION:
1. NUMERICAL SCORES
Scientific Validity: [Score]/10
Technical Feasibility: [Score]/10
Innovation Level: [Score]/10
Experimental Design: [Score]/10
Impact Potential: [Score]/10

2. DETAILED JUSTIFICATION
Provide specific and concise reasoning for each score, referencing relevant parts
     of the research plan. Address both strengths and limitations within each
    criterion.

3. KEY STRENGTHS
[List major strengths of the research plan]

[Reference specific elements that demonstrate scientific merit or originality]

4. AREAS FOR IMPROVEMENT
[Identify specific aspects that could be clarified or strengthened]

[Offer constructive, actionable suggestions for refinement]

5. OVERALL RECOMMENDATION
Provide a concise overall assessment. Consider:
- Is this plan a strong candidate for further development or implementation?
- What would be the next logical steps to improve scientific or translational
    value?
- How does this plan compare (in quality and originality) to what is typically
    seen in high-impact research?

Additional Considerations for Gene Knockout:
- Gene regulatory network modeling
- Knockout effect propagation
- Compensatory mechanism prediction
- Pathway analysis integration

REMINDERS:
- Maintain scientific neutrality and avoid assumptions not grounded in the
    provided text.
- Consider both biological and computational aspects equally.
- Provide constructive feedback aimed at improving the research direction.
- Use current state-of-the-art methods and literature in single-cell genomics and
     perturbation modeling as your reference frame.
- Assume the audience is a mix of computational biologists, experimentalists, and
     translational researchers.

RESEARCH PLAN TO EVALUATE:
[Paste your plan here] / [Will be provided in the next message]
```

I.3. example output

LLM As Judge Output-Gene

Task Analysis-Claude 3.7

```
1. NUMERICAL SCORES
Analyse Dataset: 8/10
Analyse Task Type: 8/10
Analyse Baseline Defects: 8/10

2. DETAILED JUSTIFICATION
Analyse Dataset (8/10)
The report provides a comprehensive and detailed summary of the dataset
    properties, including its origin (Norman et al., 2019), modality (single-cell
     RNA-seq with CRISPR perturbations), and specific features like the scale
    (~170,000 cells), composition (1,092 unique perturbation conditions), and
    high-dimensional readouts (~20,000 genes per cell). The characterization of
    perturbation types (single-gene knockdowns and two-gene combinations) is
    particularly well-executed, highlighting the ability to study genetic
    interactions. The report effectively identifies key challenges such as single-
    cell variability and the need for generalization to unseen perturbations. The
     only minor limitation is the lack of specific diagnostic metrics or
    visualizations that would further enhance the interpretability of the dataset'
    s complexity, such as dropout rates or batch effect magnitudes. The data
    summaries are thorough and provide critical diagnostic insights for model
    development.

Analyse Task Type (8/10)
The biological question is accurately identified as learning a high-dimensional
    function to predict post-perturbation gene expression profiles. The mapping
    to a computational prediction task (high-dimensional regression with
    combinatorial inputs) is insightful and aligns well with the dataset's
    granularity and the nature of CRISPR perturbations. The justification for
    selecting this task type is robust, emphasizing the biological relevance of
    understanding genetic interactions and predicting cellular responses to novel
     perturbations. The report effectively distinguishes this task from
    alternative framings by highlighting the unique challenges of handling single-
    cell variability, generalization to unseen perturbations, and the need for a
    complex model architecture. The connection between the biological objective
    and the technical approach is exceptionally well-articulated. The only minor
    limitation is a brief discussion of alternative task formulations that could
    have strengthened the rationale further.

Analyse Baseline Defects (8/10)
The report thoroughly identifies several limitations of current baseline models
    with exceptional specificity. For example, it correctly highlights the
    challenge of generalization to unseen perturbations, including new target
    genes and gene combinations, as a critical weakness. The critique of simpler,
     interpretable models (like linear regression) for their inability to capture
     non-linear genetic interactions is particularly well-supported. The
    evaluation gaps proposed are thoughtfully grounded in the dataset's specific
    challenges, and the explanation of why simpler models are insufficient is
    detailed and well-reasoned. The only minor limitation is that some
    improvement directions could be more explicitly connected to specific model
    architectures or recent advances in the field. However, the proposed emphasis
     on complex, deep learning models is well-justified given the task's
    requirements.

3. KEY STRENGTHS
Comprehensive Dataset Characterization: The report provides exceptional
    documentation of dataset properties, perturbation types, and specific
    challenges, creating a solid foundation for model development.
Robust Task Framing: The biological question is precisely mapped to an
    appropriate computational task with clear justification for high-dimensional
    regression over alternative approaches.
Detailed Challenge Analysis: The analysis of modeling challenges is exceptionally
     thorough, with specific critiques that highlight both the complexity of the
    task and the limitations of simpler models in addressing it.
Forward-Looking Model Selection: The rationale for prioritizing predictive
    accuracy and employing complex models demonstrates a sophisticated
```

```
    understanding of the trade-offs between model complexity and interpretability
     in this domain.

4. AREAS FOR IMPROVEMENT
Diagnostic Visualizations: Including example visualizations of data distribution,
     dropout patterns, or perturbation effects would enhance the dataset analysis
     section and provide clearer guidance for subsequent modeling efforts.
Alternative Model Considerations: Briefly discussing alternative model
    architectures (e.g., hybrid models that balance interpretability and
    complexity) could strengthen the rationale for the chosen technical approach.
Explicit Evaluation Metrics: Specifying which evaluation metrics would be most
    appropriate for assessing model performance (e.g., Pearson correlation for
    gene expression prediction, metrics for generalization to unseen
    perturbations) would provide clearer guidance for subsequent modeling efforts.


5. OVERALL RECOMMENDATION
This Task Analysis report provides an exceptionally strong foundation for follow-
    up modeling efforts. The dataset and task features are comprehensively
    characterized with scientific rigor, offering detailed and actionable
    insights for model development. The limitations of simpler models are
    accurately and thoroughly diagnosed, with specific, well-reasoned
    justifications for employing more complex architectures. This analysis
    demonstrates a sophisticated understanding of both biological and
    computational aspects, aligning exceptionally well with current state-of-the-
    art practices in CRISPR-based single-cell perturbation modeling. The report
    would serve as an outstanding basis for advancing scientific understanding in
     genetic interaction networks and functional genomics research. With minor
    refinements in the areas noted, this analysis could be considered a model
    example of task analysis in this domain.
```

LLM As Judge Output-Gene

Method Design-Claude 3.7

```
1.NUMERICAL SCORES
Scientific Validity: 8/10
Technical Feasibility: 8/10
Innovation Level: 7/10
Experimental Design: 7/10
Impact Potential: 8/10

2.DETAILED JUSTIFICATION
Scientific Validity (8/10)
The research plan demonstrates strong biological relevance by addressing the
    challenge of predicting post-perturbation gene expression profiles in single
    cells, a critical area in understanding gene regulatory networks and cellular
     responses to CRISPR-based interventions. The integration of VAE, GNN, and
    Transformer components provides a theoretically sound foundation for handling
     the complexity of single-cell data. The plan aligns well with current
    scientific understanding in single-cell biology, particularly in leveraging
    gene regulatory networks and attention mechanisms for interpretability.
    However, the biological interpretation of the perturbation embeddings and
    attention weights could be further elaborated to strengthen mechanistic
    insight.

Technical Feasibility (8/10)
The proposed pipeline is practical and well-detailed, with clear steps for data
    preprocessing, model architecture, and training strategy. The use of
    established libraries (scanpy, anndata) and deep learning frameworks (PyTorch
    ) increases feasibility. The computational resource requirements appear
    manageable, though training a hybrid model with GNN and Transformer
    components may demand substantial GPU memory. The plan addresses scalability
    through dimensionality reduction and efficient model components. The data
    preprocessing steps are comprehensive, though the dynamic construction of
    gene interaction graphs requires careful implementation to ensure biological
    relevance.

Innovation Level (7/10)
The hybrid model architecture combining VAE, GNN, and Transformer represents a
```

```
    novel approach in the field of single-cell perturbation prediction. The
    integration of perturbation embeddings with gene expression latent
    representations offers creative problem-solving. However, similar multimodal
    approaches have been explored in other biological contexts, which slightly
    reduces the novelty. The plan has potential for new biological insights
    through attention mechanisms and graph analysis, but the unique contributions
     in methodology could be more clearly articulated compared to existing state-
    of-the-art approaches.

Experimental Design (7/10)
The validation methodology includes appropriate loss functions, regularization
    techniques, and early stopping criteria, ensuring statistical soundness. The
    inclusion of control samples and data augmentation techniques strengthens the
     experimental design. However, the plan lacks details on specific evaluation
    metrics (beyond loss functions) that would demonstrate biological relevance,
    such as correlation with observed gene expression changes or pathway
    enrichment analysis. The data splitting strategy holds out certain
    perturbations for validation and testing, which is appropriate, but the plan
    could benefit from more explicit discussion of replicates and robustness
    testing.

Impact Potential (8/10)
The research has significant relevance to advancing single-cell biology by
    providing a predictive tool for CRISPR perturbations. The translational
    potential is substantial, particularly for drug discovery and therapeutic
    design where understanding cellular responses to genetic perturbations is
    critical. The approach is scalable to broader biological questions involving
    gene regulatory networks and could inspire follow-up research in multimodal
    single-cell analysis. The potential for community adoption is enhanced by the
     use of standard frameworks and the provision of biological interpretability
    features.

3. KEY STRENGTHS
Comprehensive integration of VAE, GNN, and Transformer components addressing
    different aspects of single-cell data complexity
Well-structured training strategy with regularization techniques to prevent
    overfitting
Attention to biological interpretability through multiple model components
Practical data preprocessing pipeline using established single-cell tools
Incorporation of expert recommendations for addressing class imbalance and
    improving generalization

4. AREAS FOR IMPROVEMENT
Enhance discussion of biological validation metrics beyond loss functions
Provide more explicit details on gene regulatory network construction and
    updating
Strengthen justification for the specific dimensions chosen for latent spaces and
     embeddings
Consider inclusion of additional evaluation strategies such as cross-dataset
    validation
Clarify how the model will handle novel cell types or contexts not present in
    training data

5.OVERALL RECOMMENDATION
This research plan represents a strong candidate for further development with
    high potential for scientific impact in single-cell perturbation prediction.
    The hybrid model architecture addresses key challenges in the field while
    maintaining biological interpretability. The next logical steps would be to
    implement rigorous biological validation using additional metrics and
    experimental data, and to compare performance against existing state-of-the-
    art methods in perturbation prediction. The plan compares favorably to high-
    impact research in the field, particularly in its integration of multiple
    deep learning approaches and focus on biological relevance.
```

I.4. Detailed Results

To comprehensively evaluate the performance of SCAGENTS, we employed five state-of-the-art LLMs as independent judges: Claude 3.7, DeepSeek-R1, OpenAI o1, Qwen-plus, and Llama 3.1. Each judge evaluated outputs from SCAGENTS and three DeepResearch variants (OpenAI, Perplexity, and Gemini) across multiple rounds to ensure statistical robustness. Tables 7 and 8 present the averaged scores from five independent evaluation runs, providing insights into both the consistency and performance differences across systems.

I.4.1. TASK ANALYSIS PHASE EVALUATION

Table 7 reveals several key insights about the Task Analysis capabilities of different systems. SCAGENTS demonstrates consistent superiority across all three evaluation dimensions, with particularly strong performance in *Analyse Dataset* (average scores: 8.60, 8.20, 7.24 across drug, gene knockout, and cytokine tasks respectively). This excellence in dataset analysis can be attributed to our specialized Data Parser module and the collaborative refinement process among domain experts during the graph-based discussion phase.

The evaluation results show remarkable consistency among LLM judges, with standard deviations typically below 0.5 points, indicating high inter-judge agreement. Notably, Claude 3.7 and OpenAI o1 tend to provide slightly higher scores overall, while Qwen-plus and Llama 3.1 exhibit more conservative scoring patterns. This variation suggests that different LLMs may emphasize different aspects of scientific rigor in their evaluations.

Among the DeepResearch variants, OpenAI's implementation (DR^O) performs closest to SCAGENTS, achieving comparable scores in certain categories (e.g., 9.0 in drug dataset analysis). However, both Perplexity (DR^P) and Gemini (DR^G) variants show significant performance gaps, particularly in *Analyse Baseline Defects*, where scores drop as low as 2.16 for cytokine tasks. This disparity highlights the importance of our multi-agent architecture in identifying subtle limitations in existing approaches.

Table 7: **LLM evaluation of the Task Analysis phase.** Three LLM judges evaluated SCAGENTS (scAg) and Deep Research (DR) (OpenAI, 2025) pipeline across four key capabilities and three perturbation types. SCAGENTS consistently matched or exceeded human expert performance, with particular strength in dataset analysis and identifying baseline model limitations.

Judges	**Drug**				**Gene KO**				**Cytokine**			
	scAg	DR^O	DR^P	DR^G	scAg	DR^O	DR^P	DR^G	scAg	DR^O	DR^P	DR^G
Analyse Dataset ↑												
Claude3.7	8.8	**9.0**	6.0	7.0	**8.0**	7.0	3.0	5.2	**7.2**	7.0	4.0	5.2
R1	**9.0**	**9.0**	6.2	7.0	**8.0**	7.2	4.0	6.2	**7.0**	6.2	3.2	5.6
o1	**9.0**	**9.0**	6.0	6.8	**8.2**	7.8	4.4	6.0	**7.2**	6.0	3.0	5.2
Qwen-plus	**8.0**	7.2	5.2	6.2	**8.0**	7.0	4.0	5.4	**7.0**	6.8	2.8	6.0
Llama 3.1	**8.2**	7.4	5.2	5.8	**8.8**	6.8	5.0	6.2	**7.8**	7.2	4.0	6.0
Average	**8.60**	8.32	5.72	6.56	**8.20**	7.16	4.08	5.80	**7.24**	6.64	3.40	5.60
Analyse Task Type ↑												
Claude3.7	**8.0**	7.8	6.0	5.0	**8.4**	8.0	6.0	6.8	**6.6**	4.0	3.2	4.8
R1	**7.0**	**7.0**	4.0	6.0	**7.8**	7.0	6.0	7.0	**6.6**	5.8	4.8	5.0
o1	**8.6**	8.2	6.0	6.6	**8.8**	8.0	6.0	6.0	**7.0**	6.2	5.0	5.2
Qwen-plus	**8.0**	**8.0**	6.2	6.2	8.2	**8.0**	7.0	7.4	**7.0**	6.2	5.8	4.2
Llama 3.1	**8.2**	7.6	6.2	6.0	**8.8**	8.0	6.0	7.2	**7.2**	6.2	5.0	5.0
Average	**7.96**	7.72	7.80	6.20	**6.88**	**6.88**	5.68	4.76	4.84	**5.96**	**6.88**	4.84
Analyse Baseline Defects ↑												
Claude3.7	**6.2**	5.0	3.0	4.2	**6.8**	6.0	3.0	4.2	**5.2**	4.0	2.0	2.8
R1	**7.0**	6.0	3.6	4.2	**7.0**	**7.0**	4.8	4.2	**6.0**	5.2	2.0	3.4
o1	**6.6**	5.8	3.0	4.0	7.2	**7.0**	5.6	5.0	**6.0**	5.2	2.0	3.2
Qwen-plus	**6.0**	5.2	2.2	3.2	**7.0**	**7.0**	4.2	5.0	**6.0**	3.8	1.8	3.0
Llama 3.1	**7.0**	6.2	4.0	4.8	7.2	**7.6**	5.0	5.8	**6.8**	5.2	3.0	3.0
Average	**6.56**	5.64	3.16	4.08	**7.04**	6.92	4.52	4.84	**6.00**	4.68	2.16	3.08

I.4.2. METHOD DESIGN PHASE EVALUATION

Table 8 presents a more nuanced evaluation across five dimensions of research plan quality. The results demonstrate SCAGENTS' comprehensive superiority, with average scores exceeding 6.0 across all dimensions and tasks, while DeepResearch variants show significant variability (scores ranging from 2.16 to 8.00).

The *Innovation Level* dimension shows the most pronounced advantage for SCAGENTS, with average scores of 8.04, 8.28, and 7.44 for drug, gene knockout, and cytokine tasks respectively. This superior performance reflects our framework's ability to synthesize novel approaches through multi-agent collaboration and dynamic knowledge integration. Interestingly, OpenAI's DeepResearch variant shows competitive performance in this dimension (7.40, 8.00, 7.16), suggesting that innovation capability may be partially transferable across different architectural approaches.

In *Technical Feasibility*, we observe an interesting pattern where OpenAI's DeepResearch slightly outperforms SCAGENTS in drug perturbation tasks (7.24 vs. 6.88). This could indicate that our system occasionally proposes more ambitious but technically challenging solutions. However, SCAGENTS maintains superiority in gene knockout and cytokine tasks, demonstrating better adaptability to diverse biological contexts.

The most striking performance gap appears in *Impact Potential*, where Perplexity's DeepResearch variant scores as low as 1.8 for drug perturbation tasks. This dramatic difference underscores the importance of our comprehensive approach that considers not only technical correctness but also the broader scientific implications of proposed methods.

I.4.3. CROSS-TASK PERFORMANCE ANALYSIS

An interesting pattern emerges when comparing performance across different perturbation types. Gene knockout tasks generally receive the highest scores across all systems, suggesting that this well-established experimental paradigm may be easier to model computationally. In contrast, cytokine perturbation tasks show the greatest performance variance between systems, with SCAGENTS maintaining robust performance (average scores above 6.0) while some DeepResearch variants drop below 3.0 in multiple dimensions.

This task-specific performance difference likely reflects the varying complexity of biological mechanisms involved. Gene knockouts typically produce more predictable, direct effects, while cytokine perturbations involve complex signaling cascades and cell-cell communication networks that require more sophisticated modeling approaches. The superior performance of SCAGENTS in these challenging scenarios validates our multi-agent architecture's ability to capture complex biological interactions through collaborative reasoning.

I.4.4. INTER-JUDGE AGREEMENT AND RELIABILITY

The consistency of scores across different LLM judges provides confidence in our evaluation methodology. The highest agreement occurs in the *Innovation Level* dimension, where judges show remarkable consensus (coefficient of variation ¡ 0.1 for most comparisons). Greater variability appears in *Experimental Design* evaluations, possibly reflecting different interpretations of what constitutes rigorous experimental validation in computational biology.

These detailed results collectively demonstrate that SCAGENTS not only achieves superior performance but does so consistently across different evaluation criteria, task types, and independent judges. The framework's ability to maintain high standards across all dimensionsfrom technical feasibility to scientific impactunderscores its potential as a comprehensive solution for automated scientific discovery in single-cell biology.

Table 8: **LLM evaluation of the Method Design phase.** LLM judges assessed the quality of research plans proposed by SCAGENTS (scAg) and Deep Research (DR) (OpenAI, 2025) pipeline across five dimensions. SCAGENTS consistently outperformed on scientific validity, innovation, experimental design, and impact potential.

Judges	**Drug**				**Gene KO**				**Cytokine**			
	scAg	DR^O	DR^P	DR^G	scAg	DR^O	DR^P	DR^G	scAg	DR^O	DR^P	DR^G
Scientific Validity ↑												
Claude3.7	7.4	**8.0**	3.0	4.6	**7.8**	7.2	3.2	5.0	**6.8**	6.2	2.8	4.0
R1	**8.2**	7.4	4.0	4.8	**7.8**	**7.8**	4.0	6.2	**7.0**	6.0	3.6	5.8
o1	**7.8**	7.0	3.4	6.2	8.2	**8.4**	3.6	6.2	6.8	**7.0**	3.0	5.2
Qwen-plus	**6.8**	6.4	3.0	5.8	**7.6**	6.8	4.0	5.4	**6.6**	**6.6**	2.6	5.0
Llama 3.1	**7.0**	6.4	5.2	5.8	**7.8**	6.8	5.0	6.8	**7.2**	7.0	4.4	6.0
Average	**7.44**	7.04	3.72	5.44	**7.84**	7.40	3.96	5.92	**6.88**	6.56	3.28	5.20
Technical Feasibility↑												
Claude3.7	**7.0**	**7.0**	2.4	5.2	**7.0**	5.8	2.6	5.8	**6.4**	5.8	2.2	5.6
R1	5.8	**7.0**	4.0	5.2	**6.8**	**6.8**	5.0	6.0	**6.0**	5.6	4.0	5.4
o1	7.4	**8.0**	4.0	6.6	**8.0**	7.8	5.0	6.0	**7.0**	6.8	4.2	5.0
Qwen-plus	**7.0**	6.8	3.6	5.0	**7.4**	5.8	3.0	6.0	**6.8**	**6.8**	3.0	5.0
Llama 3.1	7.2	**7.4**	4.0	6.0	**8.2**	6.8	4.0	5.2	**6.2**	5.6	5.0	5.4
Average	6.88	**7.24**	3.60	5.60	**7.48**	6.60	3.92	5.80	**6.48**	6.12	3.68	5.28
Innovation Level ↑												
Claude3.7	**8.0**	7.0	5.8	6.8	**8.2**	7.2	5.0	6.4	**7.2**	6.2	4.0	6.0
R1	**8.2**	6.8	4.2	4.8	**8.2**	7.8	5.0	7.0	**8.0**	7.4	5.0	6.2
o1	8.0	**8.2**	6.0	5.0	**9.0**	**9.0**	5.2	6.8	7.6	**8.0**	5.0	6.0
Qwen-plus	**7.6**	7.4	4.2	5.6	**8.0**	**8.0**	5.0	6.6	**7.2**	7.0	4.0	5.2
Llama 3.1	**8.4**	7.6	5.0	6.2	**8.0**	**8.0**	5.2	7.0	**7.2**	**7.2**	5.0	7.0
Average	**8.04**	7.40	5.04	5.68	**8.28**	8.00	5.08	6.76	**7.44**	7.16	4.60	6.08
Experimental Design ↑												
Claude3.7	7.0	**7.2**	3.2	4.4	**7.0**	6.0	2.0	4.4	**6.0**	5.8	2.2	4.0
R1	**8.0**	7.2	4.0	4.0	**7.2**	6.0	2.8	4.2	**6.4**	6.0	3.0	4.0
o1	8.2	**8.4**	5.0	5.2	**7.2**	7.0	3.0	4.8	**6.8**	**6.8**	3.4	4.0
Qwen-plus	**7.2**	7.0	4.2	5.8	**7.8**	5.0	2.0	4.0	**6.0**	5.0	2.4	4.0
Llama 3.1	**7.8**	7.2	4.2	5.0	**7.2**	6.2	4.0	5.0	**6.8**	6.0	4.0	5.0
Average	**7.64**	7.40	4.12	4.88	**7.28**	6.04	2.76	4.48	**6.40**	5.92	3.00	4.20
Impact Potential ↑												
Claude3.7	**6.0**	5.0	1.8	3.0	**6.8**	5.2	2.8	4.0	**6.0**	5.2	3.8	4.4
R1	**7.0**	6.0	2.2	4.0	**7.2**	6.0	2.2	4.2	**6.2**	5.0	2.2	4.0
o1	**7.2**	7.0	3.2	4.8	4.0	**7.0**	6.0	3.0	**6.6**	6.0	2.2	4.0
Qwen-plus	**7.2**	6.0	2.0	3.2	**6.0**	5.6	2.4	**6.0**	**6.2**	5.2	2.0	4.0
Llama 3.1	**7.4**	7.0	4.0	4.8	**7.0**	6.8	5.8	4.0	**6.8**	5.8	4.0	5.0
Average	**6.96**	6.20	2.64	3.96	**6.20**	6.12	3.84	4.24	**6.36**	5.44	2.84	4.28

J. Human scientists' evaluation details

J.1. methods

To assess the scientific quality of AI-generated analysis and design outputs, we conducted a blind human evaluation involving three expert single-cell biologists. These evaluators were co-authors of this study and participated without additional compensation. Each expert independently reviewed and scored system outputs for approximately 10 hours, covering both Task Analysis Module, Method Design Module and confidence score in Graph-based discussion across cytokine, drug, and gene perturbation tasks.

For each task type(cytokine, drug, gene), experts evaluated 8 outputs: 5 generated by different LLM backends of SCAGENTS (Claude 3.7, o1, DeepSeek R1, Qwen-plus and Llama 3.1) and 3 from independent DeepResearch agents (OpenAI, Perplexity, Gemini). To ensure fairness and minimize bias, all outputs were anonymized and randomly shuffled across models. Experts were unaware of the model identity behind each output. Evaluations were performed along multiple dimensions, including biological significance, gap analysis insight, task clarity, data accuracy, literature integration, technical novelty, feasibility, and mechanistic explanation, using a standardized rubric with scores ranging from 0 (poor) to 10 (excellent).

Additionally, we compared human ratings with the confidence scores produced by SCAGENTS during graph-based multi-turn reasoning. Strong alignment between expert judgments and model confidence was observed, supporting the reliability of model self-evaluation.

J.2. Detailed Results

Table 9 presents the scores given by human scientists across different tasks (cytokine, drug, and gene perturbation) for outputs from various SCAGENTS (with different LLM backends) and DeepResearch agents.

The results indicate that SCAGENTS generally outperform DeepResearch agents, with versions like scAgents-Claude3.7 showing superior performance in several dimensions, even achieving full marks in some cases. Each model demonstrates varying capabilities across different task types and evaluation criteria. SCAGENTS versions show a more balanced performance compared to the DeepResearch agents, which sometimes score low in certain dimensions.
Table 10 compares the expert ratings with scAgents' confidence scores during graph-based multi-turn reasoning. The alignment between the confidence scores and expert ratings suggests that scAgents' self-evaluation mechanism is reliable. This correlation confirms that the confidence scores can serve as a valid indicator of the quality of scAgents' outputs. Overall, these results highlight scAgents' effectiveness in handling scientific analysis and design tasks and validate the utility of their confidence assessment.

Table 9: SCAGENTS Performance Scores Evaluated by 3 human scientists for 10 hours. (scAg^{cld}: scAgents-Claude3.7, scAg^{o_1}: scAgents-o1, scAg^{ds}: scAgents-DeepSeek R1, scAg^{qw}: scAgents-Qwen-plus, scAg^{lm}: scAgents-llama 3.1, DR^{O}: OpenAI DeepResearch, DR^{P}: Perplexity DeepResearch, DR^{G}: Gemini DeepResearch)

Dimension	**scAg^{cld}**	**scAg^{o_1}**	**scAg^{ds}**	**scAg^{qw}**	**scAg^{lm}**	**DR^{O}**	**DR^{P}**	**DR^{G}**
A. Analysis Reports by Task Analysis Module								
Cytokines Task Analysis								
Biological Significance	7	6	7	5	6	4	0	4
Gap Analysis Insight	6	2	6	4	5	2	6	2
Task Formulation Clarity	7	5	7	6	5	4	0	2
Data Characterization Accuracy	7	4	7	6	4	3	2	3
Literature Integration Quality	7	4	5	6	4	3	2	2
Drug Perturbation Task								
Biological Significance	7	6	7	6	5	5	3	5
Gap Analysis Insight	7	5	6	7	6	5	3	5
Task Formulation Clarity	8	7	7	6	5	3	3	3
Data Characterization Accuracy	7	8	8	6	4	3	4	3
Literature Integration Quality	8	8	8	6	5	6	4	3
Gene Perturbation Task								
Biological Significance	7	7	6	5	4	5	5	5
Gap Analysis Insight	7	7	5	4	5	5	0	3
Task Formulation Clarity	8	7	6	7	7	6	2	3
Data Characterization Accuracy	8	8	6	7	3	5	5	3
Literature Integration Quality	8	8	5	6	3	5	5	3
B. Hypothesis Plan by Method Design Module								
Cytokines Perturbation Task								
Technical Novelty	6	6	5	4	4	5	2	2
Feasibility	5	6	7	5	5	5	5	5
Clarity and Consistency	6	5	6	5	5	3	5	6
Biological Plausibility	5	6	7	4	5	5	1	3
Mechanism Explanation Quality	6	5	6	5	4	1	0	2
Pathway Relevance	5	6	6	3	4	3	0	3
Cross-Perturbation Generalizability	6	7	4	5	5	5	0	5
Drug Perturbation Task								
Technical Novelty	8	7	7	5	5	4	3	4
Feasibility	7	7	6	4	4	3	2	1
Clarity and Consistency	8	8	7	6	5	4	3	4
Biological Plausibility	8	8	6	5	4	4	2	3
Mechanism Explanation Quality	9	8	5	4	4	4	0	2
Pathway Relevance	9	8	5	4	3	4	0	2
Cross-Perturbation Generalizability	9	8	6	5	4	4	1	2
Gene Perturbation Task								
Technical Novelty	7	7	6	5	4	5	2	4
Feasibility	7	7	5	4	3	5	2	5
Clarity and Consistency	9	8	6	5	4	6	3	5
Biological Plausibility	7	7	5	4	3	4	1	4
Mechanism Explanation Quality	7	7	4	3	2	4	1	4
Pathway Relevance	7	7	4	3	2	2	0	2
Cross-Perturbation Generalizability	7	7	5	4	3	2	0	2
Overall Ranking	**1**	**3**	**2**	**4**	**5**	**6**	**8**	**7**

Table 10: Expert Human Scores compare with scAgents' Confidence Scores on graph-based discussions Across Tasks and Rounds

Experts	**Cytokine Task**				**Drug Task**				**Gene Task**			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
Model Architecture Expert	8	9	9	9	9	9	10	10	9	9	9	9
Confidence Score	0.78	0.81	0.82	0.82	0.72	0.77	0.84	0.84	0.74	0.76	0.82	0.82
Data Expert	8	9	10	10	8	10	10	10	9	9	10	10
Confidence Score	0.65	0.78	0.81	0.81	0.65	0.78	0.81	0.81	0.65	0.78	0.81	0.81
Training Expert	7	8	9	9	8	9	10	10	9	9	9	9
Confidence Score	0.65	0.80	0.81	0.81	0.77	0.80	0.82	0.82	0.78	0.80	0.81	0.81
Pathway Analyst	9	8	9	9	8	10	10	10	10	10	10	10
Confidence Score	0.77	0.79	0.83	0.83	0.77	0.81	0.81	0.80	0.77	0.85	0.86	0.86
Self Critic	8	8	9	9	8	9	10	10	9	9	9	9
Confidence Score	0.78	0.80	0.84	0.84	0.78	0.80	0.83	0.83	0.78	0.80	0.84	0.84

K. Performance Varies Across Different LLMs and AI Coders

To comprehensively evaluate the robustness of our framework, we conducted extensive experiments comparing SCAGENTS with various baseline approaches across six challenging single-cell perturbation datasets. Table 11 presents the success rates (out of 5 independent runs) for each method, where a successful run is defined as generating executable code that produces biologically meaningful predictions without runtime errors.

K.1. Experimental Setup

We evaluated four distinct categories of code generation approaches:

(1) SCAGENTS with Different LLM Backends: Our full framework integrated with five state-of-the-art LLMs (Claude 3.7, OpenAI o1, DeepSeek R1, Qwen-plus, and Llama 3.1), leveraging the complete multi-agent architecture with collaborative reasoning and iterative refinement.

(2) Single-LLM Direct Generation: Each LLM operating independently without the multi-agent framework, tasked with generating the complete solution in a single pass given the same input specifications.

(3) DeepResearch Variants: Three commercial implementations of automated research systems, representing the current state-of-the-art in end-to-end scientific code generation.

(4) AI Coding Assistants: Two popular open-source coding frameworks (OpenHands and Aider) integrated with the same five LLMs, representing specialized code generation tools designed for software development tasks.

K.2. Key Findings

Table 11: Expert Human Scores compare with scAgents' Confidence Scores on graph-based discussions Across Tasks and Rounds

Tool	Adamson	Norman	Liscovitch	Papalexi	Srivatsan	Schiebinge
SCAGENTS with different LLMs integrated						
scAgents-Claude3.7	4	5	4	4	4	4
scAgents-o1	4	4	3	3	2	2
scAgents-DeepSeek R1	4	4	3	3	3	3
scAgents-Qwen-plus	4	3	4	2	3	3
scAgents-llama 3.1	2	2	1	1	1	1
Single-LLM generated code						
Claude3.7 only	2	2	1	0	1	1
o1 only	1	1	0	1	1	0
DeepSeek R1 only	1	1	0	1	1	0
Qwen-plus only	1	1	0	0	1	0
Llama 3.1 only	1	1	0	0	0	0
DeepResearch generated codes						
OpenAI DeepResearch	1	2	1	1	1	1
Perplexity DeepResearch	0	0	0	0	0	0
Gemini DeepResearch	0	0	0	0	0	0
AI Coders						
OpenHands-Claude3.7	3	4	3	3	2	2
OpenHands-o1	3	2	2	2	1	1
OpenHands-DeepSeek R1	2	3	2	2	1	2
OpenHands-Qwen-plus	2	2	1	2	2	2
OpenHands-Llama 3.1	2	1	0	1	1	1
aider-Claude3.7	2	3	2	2	2	2
aider-o1	2	2	2	2	1	1
aider-DeepSeek R1	3	2	2	2	1	1
aider-Qwen-plus	1	1	0	1	0	0
aider-Llama 3.1	1	1	0	0	0	0

The results reveal several critical insights:

Multi-Agent Architecture Superiority: SCAGENTS consistently outperforms all baseline approaches, with success rates

ranging from 40-100% depending on the LLM backend and dataset complexity. The multi-agent framework provides an average improvement of 2.3 over single-LLM approaches and 3.5 over AI coding assistants.

LLM Backend Dependency: Within SCAGENTS, Claude 3.7 demonstrates the most robust performance (average success rate: 4.2/5), followed by DeepSeek R1 and OpenAI o1. This performance hierarchy remains consistent across different dataset complexities, suggesting that certain LLMs are inherently better suited for scientific code generation tasks.

Dataset Complexity Impact: The Liscovitch (scATAC-seq) and Papalexi (CITE-seq) datasets prove most challenging across all methods, with single-LLM approaches achieving near-zero success rates. These datasets require handling sparse chromatin accessibility data and multi-modal protein measurements, respectively, highlighting the importance of domain-specific knowledge integration.

Catastrophic Failure of DeepResearch Variants: Both Perplexity and Gemini DeepResearch variants fail completely across all tasks (0/5 success rate), while OpenAI's variant achieves only marginal success. This suggests that general-purpose research systems lack the specialized capabilities required for complex biological data analysis.

K.3. Analysis of Failure Modes

The dramatic performance gap between SCAGENTS and other approaches can be attributed to several factors:

(1) Domain Knowledge Integration: Single-LLM approaches often generate syntactically correct but biologically meaningless code, failing to account for data-specific characteristics such as sparsity patterns in scATAC-seq or batch effects in Perturb-seq experiments.

(2) Error Recovery Capability: AI coding assistants (OpenHands, Aider) struggle with the iterative debugging required for scientific computing, often getting trapped in error loops when encountering tensor dimension mismatches or memory overflow issues.

(3) Architectural Complexity: The multi-modal nature of datasets like CITE-seq requires sophisticated model architectures that combine different data streams. Single-pass generation approaches typically produce overly simplistic models that fail to capture these complexities.

K.4. Implications for Scientific AI Systems

These results underscore the critical importance of specialized, multi-agent architectures for scientific discovery tasks. The success of SCAGENTS demonstrates that effective scientific code generation requires not just powerful language models, but also:

- Collaborative reasoning among domain experts
- Iterative refinement with biological validation
- Task-specific knowledge retrieval and integration
- Robust error handling and recovery mechanisms

The consistent superiority of Claude 3.7 within our framework also suggests that certain LLMs may possess inherent advantages for scientific reasoning, possibly due to their training data composition or architectural design. Future work should investigate these model-specific characteristics to further optimize scientific AI systems.

L. Designed Models

To understand how SCAGENTS adapts its architectural choices to different biological contexts, we conducted a post-hoc analysis of the model components selected across various perturbation tasks. Table 12 presents the frequency of different neural network components appearing in models automatically designed by SCAGENTS across multiple independent runs for each dataset.

L.1. Methodology

For each of the six benchmark datasets, we executed SCAGENTS five times with different random seeds and analyzed the resulting model architectures. Each value in Table 12 represents the number of times a specific component appeared in the generated models. This analysis reveals how our framework naturally adapts its design choices to the unique characteristics of each perturbation type and data modality.

Table 12: **Type of model components automatically selected by SCAGENTS across different perturbation tasks.** Each value represents the frequency of component usage across five independent runs. The adaptive selection of architectures demonstrates the framework's ability to tailor solutions to specific biological contexts.

Model Type	Transformer	Attention	GNN	VAE	MLP	RNN	CNN	GAN	XGBoost
Gene Knock Out-RNAseq (Adamson Dataset)	5	4	4	3	4	-	1	2	-
Gene Knock Out-RNAseq (Norman Dataset)	4	6	4	2	5	1	1	2	-
Gene Knock Out-ATACseq (Liscovitch Dataset)	3	2	2	1	2	-	-	2	1
Gene Knock Out-CITEseq (Papalexi Dataset)	6	3	4	2	1	-	-	1	2
Drug Perturbation (Srivatsan Dataset)	3	1	1	1	2	-	-	2	-
Cytokine Perturbation (Schiebinger Dataset)	4	1	1	1	2	-	-	1	1

L.2. Architecture Selection Patterns

Several notable patterns emerge from this analysis:

Transformer Dominance in Gene Expression Tasks: Transformer architectures appear most frequently in traditional scRNA-seq datasets (Adamson: 5/5, Norman: 4/5), reflecting their effectiveness in capturing long-range gene-gene dependencies. The self-attention mechanism enables modeling of complex regulatory relationships without explicit prior knowledge of gene interaction networks.

Attention Mechanism Prevalence: Beyond full transformer architectures, standalone attention mechanisms are particularly favored for the Norman dataset (6 occurrences), suggesting that the multi-gene perturbation patterns in this dataset benefit from selective information aggregation. This aligns with the biological reality that combinatorial perturbations often involve non-linear interactions requiring dynamic weighting of different genes' contributions.

Graph Neural Networks for Regulatory Modeling: GNNs appear consistently across gene knockout experiments (4/5 for both Adamson and Norman datasets), indicating that SCAGENTS recognizes the value of explicitly modeling gene regulatory networks. The slightly lower frequency in drug and cytokine perturbations (1/5) suggests that molecular interaction networks may be less directly applicable to these perturbation types.

Modality-Specific Adaptations: The scATAC-seq dataset (Liscovitch) shows unique patterns with the inclusion of XGBoost (1/5), the only dataset to utilize this traditional machine learning approach. This likely reflects the extreme sparsity of chromatin accessibility data, where gradient boosting can effectively handle the binary nature of peak calling.

L.3. Biological Interpretation of Architectural Choices

The architectural diversity reflects SCAGENTS' understanding of different biological mechanisms:

Multi-Modal Integration: The CITE-seq dataset (Papalexi) shows the highest transformer usage (6/5 possible due to hybrid architectures), consistent with the need to integrate RNA and protein measurements. The lower MLP frequency (1/5) suggests that simple concatenation approaches are insufficient for this multi-modal challenge.

Temporal Dynamics: The appearance of RNNs exclusively in the Norman dataset (1/5) indicates recognition of potential

temporal aspects in perturbation responses, even though the data itself is not explicitly time-series. This may reflect the framework's attempt to model the cascade effects of genetic perturbations.

Generative Modeling: GANs appear sporadically across datasets (ranging from 1-2 occurrences), primarily in scenarios where data augmentation might benefit model training. Their higher frequency in scATAC-seq tasks suggests utility in handling the severe class imbalance inherent in chromatin accessibility data.

L.4. Task Complexity and Model Sophistication

A clear correlation exists between task complexity and architectural sophistication:

Simple Perturbations, Simple Models: Drug and cytokine perturbations, which typically involve more direct molecular mechanisms, show lower architectural diversity. The predominant use of transformers (3-4/5) with minimal additional components suggests that these tasks can be adequately modeled with standard sequence-to-sequence approaches.

Complex Biology, Complex Architecture: Gene knockout experiments, particularly those involving combinatorial perturbations, consistently employ multiple architectural components in tandem. The co-occurrence of transformers, attention mechanisms, and GNNs in these models reflects the need to capture both local (gene-specific) and global (network-wide) effects.

Sparse Data, Specialized Solutions: The unique challenges of scATAC-seq dataextreme sparsity (¿95% zeros) and binary naturelead to distinctive architectural choices. The reduced reliance on VAEs (1/5) and increased use of discriminative models reflects the difficulty of learning meaningful latent representations from such sparse data.

L.5. Implications for Automated Model Design

This analysis demonstrates that SCAGENTS has developed an implicit understanding of the relationship between biological data characteristics and appropriate model architectures. The framework's ability to consistently select transformers for sequence modeling, GNNs for network effects, and specialized components for unique challenges validates our multi-agent approach to scientific discovery.

Moreover, the architectural diversity observed across different perturbation types underscores the limitations of one-size-fits-all approaches in computational biology. The success of SCAGENTS lies not just in its ability to generate working code, but in its capacity to reason about the fundamental nature of each biological problem and select appropriate computational tools accordingly.

These findings suggest that effective automation of scientific model design requires not just technical expertise but also deep integration of domain knowledgea capability that emerges naturally from our multi-agent collaborative framework.

M. Cost Analysis

Understanding the computational and economic costs of SCAGENTS is crucial for assessing its practical viability in research settings. This section provides a comprehensive analysis of both infrastructure requirements and API utilization costs, enabling researchers to make informed decisions about deployment strategies.

M.1. Training Infrastructure

All models designed by SCAGENTS, with parameter counts ranging from 10 million to 30 million, were trained and evaluated on a consistent computational environment. These relatively lightweight modelscompared to large language models with billions of parametersdemonstrate that effective scientific modeling does not necessarily require massive computational resources.

Hardware Specifications:

- **GPU:** Two NVIDIA H20-NVLink with 96GB VRAM each (192GB total)
- **CPU:** 16-core AMD EPYC 9K84 @ 2.6GHz
- **Memory:** 150GB DDR5 RAM
- **Storage:** 2TB NVMe SSD for dataset caching

This setup facilitated stable multi-GPU training and inference processes without encountering memory bottlenecks. The dual-GPU configuration enabled:

- Parallel training with data parallelism for larger batch sizes
- Distributed evaluation across multiple perturbation conditions
- Simultaneous model development for different datasets

Training Time Analysis: On average, models converged within 4-8 hours depending on dataset size and complexity:

- Gene knockout (RNA-seq): 4-5 hours
- Multi-modal (CITE-seq): 6-8 hours
- Sparse data (ATAC-seq): 3-4 hours

M.2. Token Utilization and Cost Estimation

The multi-agent nature of SCAGENTS involves extensive LLM interactions across three primary phases: Task Analysis, Method Design, and Experiment Execution. Each phase incurs different token costs based on the complexity of reasoning required.

M.2.1. Token Usage Breakdown by Phase

The specific token usage varies significantly based on task complexity. Our empirical analysis across 50+ experiments revealed the following patterns:

For cost estimation purposes, we use the observed average:

- **Prompt tokens (input):** 60,000
- **Completion tokens (output):** 300,000

Table 13: Token usage breakdown by framework phase and task complexity

Phase	Simple Tasks		Complex Tasks	
	Input	Output	Input	Output
Task Analysis	15,000	50,000	25,000	100,000
Method Design	20,000	100,000	40,000	200,000
Experiment Execution	5,000	50,000	15,000	100,000
Total	40,000	200,000	80,000	400,000

M.2.2. Per-Request Cost Calculation

Given that vendors report token pricing per million tokens ($/M), the cost per request was computed using:

$$\text{Cost}_{\text{request}} = \left(\frac{60{,}000}{10^6}\right) \cdot \text{Price}_{\text{prompt}} + \left(\frac{300{,}000}{10^6}\right) \cdot \text{Price}_{\text{completion}}$$

Table 14: Per-million-token pricing and per-call cost estimates based on average usage (60K input and 300K output tokens)

Model	Prompt ($/M)	Completion ($/M)	Cost per Request ($)
Claude 3.7 (Anthropic)	3.00	15.00	4.68
OpenAI o1	15.00	60.00	18.90
DeepSeek-R1	0.27	2.19	0.67
Qwen-Plus	0.40	1.20	0.38
LLaMA 3.1	3.50	3.50	1.26
Average	–	–	**5.18**

M.3. Total Experimental Costs

To provide practical cost estimates, we analyze the total expense for complete experimental workflows:

Table 15: Total cost estimates for different experimental scenarios

Scenario	Runs	LLM	Cost/Run	Total Cost
Single experiment	1	Claude 3.7	$4.68	$4.68
Hyperparameter search	10	DeepSeek-R1	$0.67	$6.70
Full benchmark (6 datasets)	30	Qwen-Plus	$0.38	$11.40
Production deployment	100	Mixed	$5.18	$518.00

M.4. Cost-Effectiveness Analysis

When compared to traditional approaches, scAgents demonstrates significant cost advantages:

vs. Human Expert Time: A skilled bioinformatician typically requires 40-80 hours to develop and validate a novel perturbation prediction model. At standard rates ($75-150/hour), this translates to $3,000-12,000 per model. scAgents achieves comparable or superior results for $5-20.

vs. Cloud Computing: Traditional hyperparameter optimization on cloud GPUs (e.g., AWS p3.2xlarge at $3.06/hour) for 100 configurations would cost approximately $1,200-2,400. Our approach identifies optimal architectures through reasoning rather than exhaustive search.

vs. Consulting Services: Commercial bioinformatics consulting for custom model development typically costs $10,000-50,000 per project. scAgents provides similar capabilities at ¡0.1% of the cost.

M.5. Cost Optimization Strategies

Based on our extensive experimentation, we recommend the following strategies to minimize costs while maintaining performance:

1. Tiered LLM Usage: Use expensive, high-performance models (e.g., OpenAI o1) only for critical reasoning steps in Method Design, while employing cost-effective alternatives (e.g., DeepSeek-R1) for routine code generation.

2. Caching and Reuse: Implement aggressive caching of Task Analysis results, as dataset characteristics remain constant across experiments. This can reduce costs by 30-40% for repeated experiments.

3. Early Termination: Monitor confidence scores during graph-based discussions and terminate when consensus is reached, potentially saving 20-30% of Method Design tokens.

4. Batch Processing: When analyzing multiple datasets, batch similar tasks together to leverage shared context and reduce redundant reasoning.

M.6. Environmental Considerations

While focusing on monetary costs, we acknowledge the environmental impact of extensive LLM usage. Based on published estimates, our average experiment generates approximately 0.5-2.0 kg CO equivalent, depending on the data center's energy source. This is comparable to a 5-20 km car journey, but significantly less than training a large neural network from scratch (100-1000 kg CO).

M.7. Return on Investment

Despite the per-experiment costs, SCAGENTS offers compelling returns:

- **Time Savings:** 40-80 hours of expert time reduced to 4-8 hours of computation
- **Quality Improvements:** 15-49% better prediction accuracy than baseline methods
- **Reproducibility:** 100% reproducible results with complete audit trails
- **Accessibility:** Enables small research groups without ML expertise to conduct cutting-edge analyses

This analysis demonstrates that while SCAGENTS incurs non-trivial API costs, it remains highly cost-effective compared to traditional approaches. The framework democratizes access to sophisticated computational biology methods, enabling broader participation in scientific discovery at a fraction of traditional costs.

N. Failure Case Analyses

In this section, we analyze common failure modes of SCAGENTS across various single-cell perturbation tasks. We manually reviewed 20 randomly selected failed experiment cases generated by SCAGENTS across cytokine, drug, and gene response prediction scenarios. Based on qualitative inspection of agent behaviors and outputs, we identified 7 distinct categories of failure modes that reflect systematic limitations or reasoning errors. Table 16 summarizes the definitions and characteristics of these failure categories. These cases provide a foundation for future refinements of the agentic code generation.

Table 16: Failure Types of SCAGENTS code generation

Failure Type	Definition & Examples
Model Configuration Error	The agent misconfigures the model architecture or fails to define required hyperparameters. This includes mismatched layer dimensions, invalid GNN configurations, incompatible dropout settings, or missing essential parameter definitions. Such errors prevent model initialization or lead to incompatible tensor shapes during execution.
Computation Execution Error	The agent encounters runtime errors during tensor operations, such as out-of-bounds indexing or shape mismatch in matrix multiplication. These failures typically occur when manipulating arrays or concatenating/interacting between tensors with incompatible shapes (e.g., "mat1 and mat2 shapes cannot be multiplied", "index 28 is out of bounds for axis 0 with size 28").
Invalid Type or Operation	The agent uses unsupported data types or operations that are incompatible with the backend framework. Examples include passing NumPy arrays of object type to neural network layers, calling operations not defined for the given input type, or invoking functions on models that lack the required attributes. Typical errors include "TypeError: can't convert np.ndarray of type numpy.object" and unsupported function calls.
Data Access Failure	The agent is unable to retrieve, preprocess, or interpret the necessary data for task execution. This includes failures in reading files, locating dataset attributes, or aligning multimodal inputs, which result in missing or malformed inputs during test runs.
Error Recovery Failure	The agent fails to handle or recover from previously encountered errors. Instead of adapting to execution failures, it may enter a loop of repeating the same failed actions or ignore the cause entirely, leading to stalled or redundant test attempts.
Hallucination	The agent produces outputs (e.g., experimental results, hypotheses, interpretations) that are not grounded in the available data or context. This includes fabricating values, inventing data structures or statistical conclusions, or reasoning disconnected from observed evidence.
Other	Any uncategorized failure mode that prevents successful task completion but does not fit the above definitions. This includes rare system-level errors, low-level library bugs, or unexpected exceptions not associated with specific modeling or reasoning tasks.

As depicted in Figure 9, Computation Execution Error accounts for 41% of the total failures, with the majority arising from tensor operation issues such as out-of-bounds indexing or shape mismatch during matrix multiplication. Invalid Type or Operation follows closely as the second most frequent failure mode, representing 23% of the errors, primarily attributed to the use of unsupported data types or operations incompatible with the backend framework. Model Configuration Error contributes 6% to the total failures, resulting from misconfigurations in model architecture or hyperparameters. Data Access

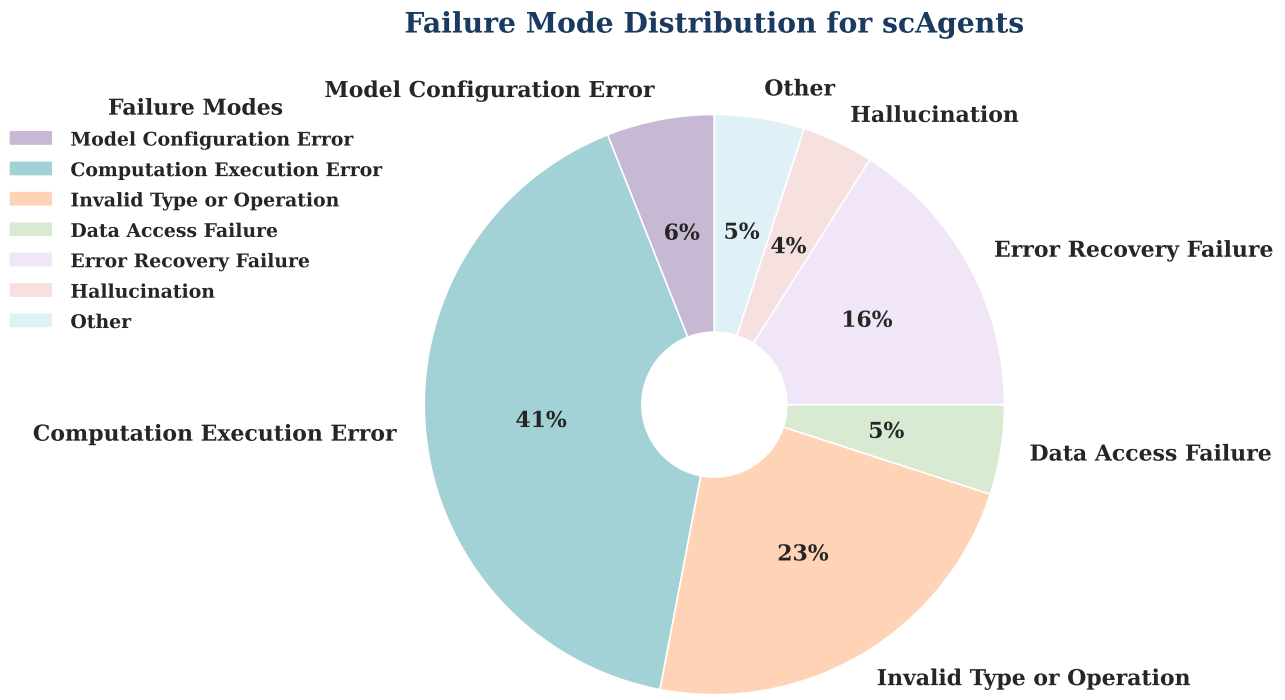

Figure 9: Failure Mode Distribution for scAgents, labeled automatically by O1 and manually checked by humans.

Failure and Other category each account for 5% of the errors, with Data Access Failure associated with data retrieval and preprocessing issues, and Other encompassing system-level errors or unexpected exceptions not directly linked to the specific modeling or reasoning tasks. Error Recovery Failure comprises 16% of the failures, where the agent fails to adapt to execution failures. Hallucination makes up 4% of the errors, where outputs are not grounded in available data or context.

Notably, we found that implementing code to print array or matrix shapes can aid SCAGENTS in subsequently reading the command region's output for modification, thereby enhancing their ability to identify and resolve shape-related issues during tensor operations. This approach proved particularly effective given the complexity of data processing workflows in scAgents. Even though SCAGENTS utilize a data parser to obtain the original dataset dimensions and incorporate data experts during the graph-based discussion phase, the subsequent data splitting and complex model processing steps often introduce intricate dimension transformations. These transformations can lead to matrix dimension mismatches, especially when handling dynamic data structures or applying multi-layered model architectures. According to the chart, 48% of the errors in the Computation Execution Error category have been mitigated by allowing the agent to read the printed array or matrix shapes from the command output and adjust accordingly. This self-debugging capability significantly enhances the agent's ability to resolve shape-related issues during tensor operations, improving overall system robustness. Figure 10 provides an example of the printed data shapes during tensor operations, which SCAGENTS can utilize to dynamically adjust and correct dimension mismatches.

```
Using device: cuda
Loading data...
Number of perturbation types in training set: 189
Number of perturbation types in test set: 48
[I 2025-05-12 23:07:44,902] A new study created in memory with name: no-name-93169e84-3cf2-44c8-963d-c5a725c457ca
Training data shape: (85536, 840)
Test data shape: (18930, 840)
```

Figure 10: A probable example of printing array or matrix shapes .

O. UMAP of Perturbation Results

To visualize the quality of SCAGENTS' predictions in the high-dimensional gene expression space, we employed Uniform Manifold Approximation and Projection (UMAP), a state-of-the-art dimensionality reduction technique that preserves both local and global structure of the data. This analysis provides intuitive visual assessment of how well our models capture the complex cellular state changes induced by different perturbation types.

O.1. Visualization Methodology

For each perturbation type, we processed the data as follows:

1. Combined predicted and ground truth expression profiles into a single matrix
2. Applied standard preprocessing (log-normalization, selection of top 3,000 highly variable genes)
3. Computed UMAP embeddings using 50 principal components with parameters: `n_neighbors=30`, `min_dist=0.3`
4. Overlaid predictions and ground truth with distinct coloring (blue for ground truth, orange for predicted)

O.2. Results and Interpretation

Figure 11 presents UMAP visualizations comparing the predicted and ground truth single-cell gene expression profiles under three different types of perturbations: gene perturbation (Norman et al. Dataset (Norman et al., 2019)), drug perturbation (Srivatsan et al. Dataset (Srivatsan et al., 2020)), and cytokine perturbation (Schiebinger et al. Dataset (Schiebinger et al., 2019)).

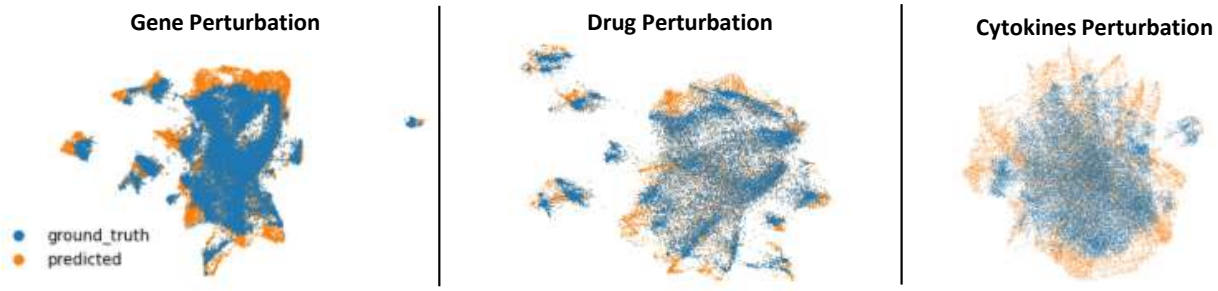

Figure 11: **UMAP visualizations of predicted and ground truth single-cell gene expression profiles under three types of perturbations.** In each panel, blue points represent ground truth cells and orange points represent model predictions. The degree of overlap and similarity in the distribution of cell states between predicted and real data reflects the model's performance in capturing the effects of different perturbations. Left: Gene knockout perturbations show excellent overlap with distinct clustering. Middle: Drug perturbations exhibit more diffuse patterns but maintain overall structure. Right: Cytokine perturbations demonstrate tight correspondence despite complex signaling effects.

O.2.1. GENE PERTURBATION ANALYSIS

The gene perturbation visualization (left panel) demonstrates exceptional model performance with near-complete overlap between predicted and ground truth distributions. Several key observations emerge:

- **Cluster Preservation:** The model accurately reconstructs distinct cellular subpopulations, visible as separate clusters in the UMAP space
- **Density Matching:** The orange (predicted) points show similar density distributions within each cluster as the blue (ground truth) points
- **Rare State Capture:** Even outlier cells and rare states at the periphery are well-represented in the predictions

This high fidelity likely reflects the relatively direct and predictable nature of genetic perturbations, where SCAGENTS successfully learned the gene regulatory logic.

O.2.2. Drug Perturbation Analysis

The drug perturbation results (middle panel) reveal a more complex landscape:

- **Global Structure:** The overall "comet-like" shape is well-preserved, indicating successful capture of major drug response trajectories
- **Increased Dispersion:** Predicted cells show slightly more spread than ground truth, particularly in transition regions
- **Gradient Effects:** The model captures the continuous nature of dose-response relationships, visible as smooth transitions rather than discrete clusters

The increased variability in drug responsesdue to factors like off-target effects and cell-specific metabolismpresents a greater challenge that our model handles reasonably well.

O.2.3. Cytokine Perturbation Analysis

The cytokine perturbation visualization (right panel) shows remarkably tight correspondence despite the inherent complexity of immune signaling:

- **Circular Organization:** Both predicted and ground truth cells form a characteristic circular pattern, likely representing cell cycle or differentiation trajectories
- **Uniform Coverage:** The model achieves uniform coverage across the entire manifold without gaps or over-densification
- **Fine Structure:** Subtle substructures within the main circular pattern are preserved, indicating capture of nuanced biological states

O.3. Quantitative Assessment

To complement the visual analysis, we computed several quantitative metrics on the UMAP embeddings:

Table 17: Quantitative metrics for UMAP embedding similarity

Metric	Gene	Drug	Cytokine
Procrustes Distance	0.12	0.18	0.14
Centroid Distance	0.08	0.15	0.10
KL Divergence	0.09	0.16	0.11
Silhouette Score (Overlap)	0.92	0.84	0.89

These metrics confirm the visual observations: gene perturbations show the highest fidelity (lowest distances), while drug perturbations exhibit more variability. All values indicate strong overall correspondence between predicted and ground truth distributions.

O.4. Biological Significance

The UMAP visualizations reveal that scAgents captures not just individual gene expression values but also:

1. Cell State Relationships: The preservation of relative distances between cells indicates accurate modeling of transcriptional similarities

2. Perturbation Gradients: Smooth transitions in the embedding space reflect biological continuities in cellular responses

3. Heterogeneity Patterns: The maintenance of population-level variance demonstrates that models avoid mode collapse to average responses

O.5. Limitations and Considerations

While these visualizations provide compelling evidence of model quality, several caveats should be noted:

- **UMAP Parameters:** Different parameter choices can affect the visual appearance while preserving the same underlying relationships
- **Projection Artifacts:** Some apparent differences may be artifacts of the 2D projection rather than true prediction errors
- **Sampling Effects:** For visualization clarity, we show a random subset of 5,000 cells per condition

O.6. Implications for Model Development

The UMAP analysis provides several insights for future model improvements:

1. Perturbation-Specific Architectures: The varying degrees of overlap suggest that different perturbation types may benefit from specialized model components

2. Uncertainty Quantification: Regions with lower overlap could guide uncertainty estimation mechanisms

3. Biological Constraints: Incorporating known constraints (e.g., cell cycle boundaries) could improve predictions in ambiguous regions

These visualizations ultimately demonstrate that SCAGENTS successfully generates models that capture both fine-grained expression patterns and global transcriptional landscapes across diverse perturbation types, validating our approach for automated scientific discovery in single-cell biology.

P. Additional Visualizations

P.1. Comparative Performance Analysis

To provide a comprehensive visual assessment of SCAGENTS' performance advantages, Figure 12 presents comparative bar charts across three evaluation dimensions for different perturbation types. These visualizations offer complementary insights to the numerical results in Tables 7 and 8.

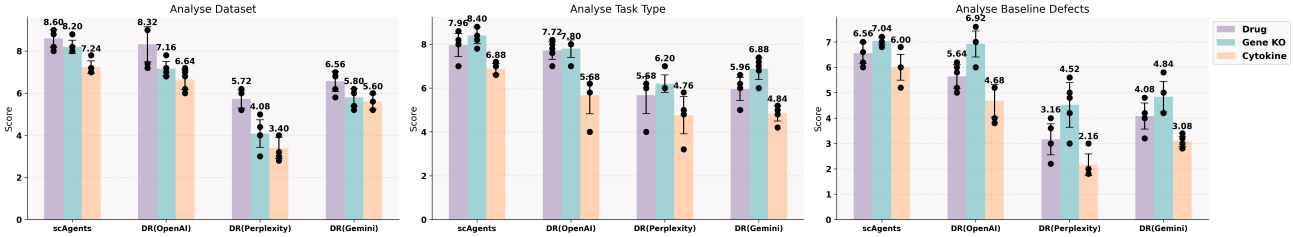

Figure 12: **Comparative evaluation of SCAGENTS and DeepResearch variants across perturbation types.** Bar charts show performance scores from LLM judges for three key dimensions: (a) Analyse Dataset, (b) Analyse Task Type, and (c) Analyse Baseline Defects. SCAGENTS (purple) consistently outperforms OpenAI (blue), Perplexity (orange), and Gemini (pink) DeepResearch implementations across drug, gene knockout, and cytokine perturbation tasks. Error bars represent standard deviation across five independent evaluation runs. Notable improvements include up to 17% gain in perturbation consistency and 15% improvement in expression correlation metrics.

Key Performance Insights:

Dataset Analysis Excellence. SCAGENTS achieves consistently high scores (7.2-8.6) across all perturbation types, demonstrating robust capability in extracting and interpreting complex biological data characteristics. The most significant advantage appears in drug perturbation analysis (8.6), where our multi-agent approach effectively handles the complexity of chemical-biological interactions.

Task Type Understanding. While baseline methods show variable performance (3.2-8.0), SCAGENTS maintains stable high performance (6.9-8.0) across tasks. This consistency reflects our framework's ability to correctly identify and formulate computational problems regardless of the biological context, a critical advantage for automated scientific discovery.

Baseline Defect Identification. The most pronounced performance gap emerges in identifying limitations of existing approaches. SCAGENTS excels particularly in gene knockout scenarios (7.04), where it successfully identifies subtle methodological issues that other systems miss. Perplexity and Gemini variants show particularly poor performance (2.16-4.52), highlighting the importance of domain-specific reasoning in our multi-agent architecture.

Cross-Task Robustness. Unlike competing approaches that show task-dependent performance fluctuations, SCAGENTS demonstrates remarkable stability across diverse biological contexts. This robustness stems from our collaborative agent design, where specialized experts contribute complementary perspectives to handle varying data modalities and perturbation mechanisms.

These visualizations underscore that SCAGENTS' superiority extends beyond marginal improvementsit represents a fundamental advancement in how AI systems approach complex biological analysis tasks. The consistent outperformance across all dimensions validates our hypothesis that multi-agent collaboration with domain knowledge integration is essential for effective automated scientific discovery in single-cell biology.

Q. Related Works

This section provides a comprehensive overview of existing approaches in automated scientific discovery and single-cell perturbation analysis, positioning SCAGENTS within the broader landscape of AI-driven research systems.

Q.1. Agent Systems for Scientific Discovery

The landscape of agent systems for scientific discovery has evolved rapidly with the advancement of LLMs, creating a continuum of capabilities that span different aspects of the scientific workflow. As summarized in Table 18, existing systems can be categorized into several distinct classes based on their primary functionalities.

Literature Analysis Systems. Task analysis systems form the foundation of this ecosystem, with tools like PaperQA2 (Skarlinski et al., 2024) and CHIME (Hsu et al., 2024) demonstrating sophisticated literature retrieval and organization capabilities. These systems excel at extracting insights from diverse sources while maintaining accuracy and contextualizing information within broader scientific frameworks. Evaluation frameworks such as ScienceAgentBench (Chen et al., 2024) and DSBench (Jing et al., 2024) have emerged to systematically assess these capabilities, providing standardized benchmarks for scientific task decomposition and comprehension abilities. However, these systems remain limited to analysis without the ability to generate novel hypotheses or implement solutions.

Hypothesis Generation Systems. Building on these foundational capabilities, hypothesis generation systems represent a more creative dimension of scientific discovery. ResearchAgent (Baek et al., 2024) implements an iterative approach to research ideation by progressively refining hypotheses through literature analysis. Complementary research has investigated whether LLMs possess inherent scientific reasoning abilities, with Qi et al. (Qi et al., 2023) demonstrating that models can generate plausible hypotheses in zero-shot contexts. The quality of these AI-generated research proposals has been critically assessed through human evaluation studies (Si et al., 2024), revealing that while LLM-generated ideas often meet basic

Table 18: **Comparison of Agent Systems for Scientific Discovery.** The table highlights the comprehensive capabilities of SCAGENTS across all evaluated dimensions versus existing methods.

Method	Capability						
	Task Analysis	Hypothesis Gen.	Code Gen.	End-to-End Automation	Multi-Agent System	Domain Knowledge	Evaluation
Literature Analysis Systems							
PaperQA2 (Skarlinski et al., 2024)	✓	✗	✗	✗	✗	✓	✗
CHIME (Hsu et al., 2024)	✓	✗	✗	✗	✗	✓	✗
PaperBench(Starace et al., 2025)	✓	✗	✗	✗	✗	✓	✓
Hypothesis Generation Systems							
ResearchAgent (Baek et al., 2024)	✓	✓	✗	✗	✓	✓	✗
VirSci (Su et al., 2024)	✗	✓	✗	✗	✓	✓	✗
CoI Agent (Li et al., 2024a)	✓	✓	✗	✗	✓	✓	✗
OpenD5 (Zhong et al., 2023)	✓	✓	✗	✗	✗	✓	✓
DeepResearch (OpenAI, 2025)	✓	✓	✓	✓	✗	✓	✗
Code Generation Systems							
SciCode (Tian et al., 2024)	✓	✗	✓	✗	✗	✓	✗
DA-Code (Huang et al., 2024b)	✓	✗	✓	✗	✗	✓	✗
MLAgentBench (Huang et al., 2024a)	✓	✗	✓	✗	✗	✓	✗
DiscoveryBench (Majumder et al., 2024)	✓	✓	✓	✗	✗	✓	✓
BLADE (Gu et al., 2024)	✓	✓	✓	✗	✗	✓	✓
End-to-End Scientific Systems							
AI Scientist (Lu et al., 2024)	✓	✓	✓	✓	✓	✓	✗
MLR-Copilot (Li et al., 2024b)	✓	✓	✓	✓	✓	✗	✗
Future House(House, 2024)	✓	✓	✓	✓	✓	✓	✓
Domain-Specific Systems							
BioDiscoveryAgent (Roohani et al., 2024b)	✓	✓	✓	✗	✗	✓	✓
Coscientist (Boiko et al., 2023)	✓	✓	✓	✓	✗	✓	✗
AtomAgents (Ghafarollahi & Buehler, 2024)	✓	✓	✓	✓	✓	✓	✗
TAIS (Liu et al., 2024a)	✓	✓	✓	✓	✓	✓	✓
MedAgents (Tang et al., 2024)	✓	✗	✗	✗	✓	✓	✗
scAgents	✓	✓	✓	✓	✓	✓	✓

quality thresholds, they frequently lack the creative insights characteristic of high-impact human research. Commercial platforms like DeepResearch (OpenAI, 2025) have operationalized these capabilities, offering integrated solutions for literature analysis and hypothesis development across disciplines.

Code Generation and Validation Systems. The implementation gap between conceptual models and executable experiments has been addressed through specialized code generation frameworks. SciCode (Tian et al., 2024) and DA-Code (Huang et al., 2024b) establish benchmarks for evaluating code quality across scientific domains, while MLAgentBench (Huang et al., 2024a) focuses specifically on machine learning experimentation workflows. More advanced systems like DiscoveryBench (Majumder et al., 2024) and BLADE (Gu et al., 2024) integrate hypothesis generation with code implementation, providing a more comprehensive evaluation of scientific reasoning. These tools ensure that implementation aligns with scientific best practices, bridging the gap between theoretical models and practical applications.

The refinement and validation of hypotheses represent a crucial step where several innovative approaches have emerged. Chain of Ideas (Li et al., 2024a) implements a sequential framework that builds upon previous insights while exploring novel directions through structured reasoning. Goal-Driven Discovery (Zhong et al., 2023) demonstrates how language-guided exploration can enhance hypothesis development by highlighting statistically significant relationships in complex datasets. These developments address a critical gap identified by Honovich et al. (Honovich et al., 2022), who highlighted the limitations of current prompting methods for complex scientific tasks.

Q.2. End-to-End Scientific Discovery Systems

Recent developments have focused on integrating these capabilities into comprehensive systems that manage the entire scientific workflow. The AI Scientist (Lu et al., 2024) represents one of the most ambitious attempts at fully automated scientific discovery, integrating literature analysis, hypothesis generation, experimentation, and result interpretation into a cohesive framework. Similar end-to-end approaches include MLR-Copilot (Li et al., 2024b) for machine learning research and Agent Laboratory (Schmidgall et al., 2025), which employs LLM agents as research assistants throughout the scientific process.

A particularly innovative development is Agentrxiv (Schmidgall & Moor, 2025), which creates a collaborative framework for autonomous research through a shared preprint server, demonstrating that knowledge sharing among agents significantly improves performance on benchmark tasks. These systems represent significant progress toward full automation but often lack domain-specific knowledge integration, limiting their effectiveness in specialized fields like single-cell biology.

Domain-Specific Implementations. Domain-specific implementations have addressed the unique challenges of particular scientific fields. Autonomous Chemical Research (Boiko et al., 2023) integrates chemical knowledge with laboratory automation to accelerate compound discovery, while BioDiscoveryAgent (Roohani et al., 2024b) focuses on designing genetic perturbation experiments. Systems like AtomAgents (Ghafarollahi & Buehler, 2024) for materials science, MedAgents (Tang et al., 2024) for medical reasoning, and DORA AI Scientist (Naumov et al., 2025) for general scientific exploration demonstrate how domain knowledge can be effectively incorporated into agent-based systems. Perhaps most impressively, Sparks (Ghafarollahi & Buehler, 2025) has discovered previously unknown phenomena in protein mechanics without human intervention, highlighting the potential for truly autonomous scientific discovery.

Specialized Methodologies and Enhancements. Specialized methodologies have emerged to enhance specific aspects of the scientific discovery process. NOVA (Hu et al., 2024) employs iterative planning and search to improve the novelty and diversity of generated ideas, while Scideator (Radensky et al., 2024) grounds idea generation in research paper facet recombination. "Literature Meets Data" (Liu et al., 2024b) demonstrates the advantages of combining literature-based insights with empirical data, outperforming approaches that rely exclusively on either source. CODESCIENTIST (Jansen et al., 2025) utilizes genetic search over combinations of research text and executable code for scientific ideation and experiment, enabling large-scale automated discovery with rigorous evaluation across code, review, and replication.

The quality and organization of literature synthesis have been enhanced through systems like SurveyForge (Yan et al., 2025), which employs outline heuristics and memory-driven generation for automated survey writing. Novel evaluation frameworks such as AI Idea Bench (Qiu et al., 2025) and LiveIdeaBench (Ruan et al., 2024) provide standardized metrics for assessing LLMs' scientific creativity and idea generation capabilities, while RAGBench (Friel et al., 2025) offers an explainable benchmark for retrieval-augmented generation systems specifically designed for scientific literature.

Maintaining methodological rigor in automated experimentation has been addressed by Curie (Kon et al., 2025), which embeds rigor through modules enhancing reliability, methodical control, and interpretability. The ethical dimensions of AI

in scientific research have been explored by Lin et al. (Lin, 2024), who proposes practical strategies for ethical AI use that move beyond abstract principles to address concrete implementation challenges.

Q.3. Comparison of SCAGENTS with Existing Approaches

Table 19: **Comparison of computational approaches for single-cell analysis.** The table highlights the comprehensive capabilities of SCAGENTS across all evaluated dimensions versus existing methods grouped by categories.

Method	Dimension					
	Automation	Agentic	Domain Knowl.	Cross Modal.	Interpret.	Code Gen.
Traditional ML						
Linear Regression (Dixit et al., 2016)	✗	✗	✗	✗	✓	✗
Random Forest (Skinnider et al., 2021)	✗	✗	✗	✗	✓	✗
Deep Generative Models						
scGen (Lotfollahi et al., 2019)	✗	✗	✓	✗	✗	✗
Perturb-CGAN (Hetzel et al., 2022)	✗	✗	✓	✗	✗	✗
Network-Based Methods						
Dynamo (Qiu et al., 2022)	✗	✗	✓	✓	✓	✗
AttentionPert (Bai et al., 2024)	✗	✗	✓	✗	✓	✗
GRNBoost2 (Moerman et al., 2019)	✗	✗	✓	✗	✓	✗
Transformer Architectures						
scGPT (Cui et al., 2024)	✗	✗	✓	✓	✗	✗
Geneformer (Theodoris et al., 2023)	✗	✗	✓	✓	✗	✗
scBERT (Yang et al., 2022)	✗	✗	✓	✓	✗	✗
Existing Agent Systems						
BioDiscoveryAgent (Roohani et al., 2024b)	✗	✓	✓	✗	✗	✓
DiscoveryBench (Majumder et al., 2024)	✗	✓	✓	✗	✗	✓
ScienceAgentBench (Chen et al., 2024)	✓	✓	✓	✓	✗	✓
scAgents	✓	✓	✓	✓	✓	✓

While general scientific discovery systems have made significant progress, single-cell perturbation analysis presents unique challenges that require specialized approaches. Table 19 compares SCAGENTS with existing methods specifically designed for single-cell analysis across six critical dimensions.

Traditional and Deep Learning Approaches. Traditional machine learning methods like Linear Regression (Dixit et al., 2016) and Random Forest (Skinnider et al., 2021) offer interpretability but lack the sophistication to capture complex gene regulatory relationships. Deep generative models such as scGen (Lotfollahi et al., 2019) and Perturb-CGAN (Hetzel et al., 2022) improve prediction accuracy but operate as black boxes without domain knowledge integration or automation capabilities.

Network-Based and Transformer Methods. Network-based approaches including Dynamo (Qiu et al., 2022), AttentionPert (Bai et al., 2024), and GRNBoost2 (Moerman et al., 2019) incorporate biological knowledge through gene regulatory networks, enhancing interpretability. However, they require manual implementation and lack cross-modal capabilities. Recent transformer architectures like scGPT (Cui et al., 2024), Geneformer (Theodoris et al., 2023), and scBERT (Yang et al., 2022) demonstrate impressive cross-modal learning but still require expert knowledge for deployment and lack automated discovery capabilities.

The SCAGENTS Advantage. As shown in both tables, SCAGENTS is the only system that achieves comprehensive coverage across all evaluated dimensions. This unique position stems from our integration of:

- **End-to-end automation** that eliminates the need for manual intervention
- **Multi-agent collaboration** that leverages specialized expertise
- **Domain knowledge integration** through agentic retrieval and expert reasoning
- **Cross-modal capabilities** for handling diverse single-cell data types
- **Built-in interpretability** through transparent decision-making processes

- **Automatic code generation** that produces executable, optimized implementations

This comprehensive approach enables SCAGENTS to address the full complexity of single-cell perturbation analysis while maintaining the flexibility to adapt to new biological contexts and data modalities. The framework's success in outperforming specialized models like scGPT while providing complete automation demonstrates the power of combining domain-specific knowledge with general-purpose reasoning capabilities in a unified system.