Agentic AI Systems in Gene Research: Small and Large Models in Action

Introduction

Recent advances in **agentic AI** have enabled systems where multiple AI components – both **small, specialized models** and **large language models (LLMs)** – collaborate to tackle complex tasks in gene research. Instead of relying on one monolithic model, these agent-based frameworks divide work among **"expert" agents** (e.g. a tool-using database query agent, a code-writing agent, a verifier) orchestrated by a higher-level reasoning model. This design leverages the efficiency and domain knowledge of small models alongside the reasoning power of large models. Crucially, such systems can interface with bioinformatics tools and databases to ground their outputs in factual data, reducing errors like hallucinations. Below, we survey several **practically useful agentic AI systems** already emerging in gene and genome research, highlighting the roles of their small and large model components in real workflows.

Gene Set and Pathway Analysis Agents

Identifying pathways or functional modules from a list of genes is a common task in genomics. **GeneAgent**, developed at NIH, exemplifies an LLM-driven agent for gene set enrichment analysis. GeneAgent uses a GPT-4 based planner to generate candidate descriptions for the biological processes that connect an input gene set, then employs a self-verification **agent** to cross-check each claim against expert-curated databases (such as Gene Ontology, Enrichr, and others). This automated fact-checking loop flags unsupported statements and prompts the model to correct them. By iterating generation → verification → modification, GeneAgent was able to **mitigate hallucinations** and produce accurate pathway descriptions far more reliably than GPT-4 alone. In tests on 1,106 benchmark gene sets, it outperformed a standard GPT-4 workflow, with human experts judging **92% of GeneAgent's self-verification decisions correct**. This agent system ultimately helps researchers interpret high-throughput data by pinpointing relevant biological pathways and functions for gene groups. Notably, GeneAgent can even suggest insights for novel gene sets (e.g. from melanoma experiments), hinting at potential new disease mechanisms or drug targets that might have been missed by manual analysis.

GeneAgent builds on earlier efforts to combine LLMs with genomic databases. For example, **GeneGPT** (Jin *et al.*, 2024) taught a code-oriented LLM to call NCBI web APIs for answering genomics questions. By retrieving gene information via tools (NCBI Entrez queries, BLAST, etc.), GeneGPT grounded its answers in real data and was shown to surpass vanilla GPT-3.5/ChatGPT on specialized queries. Similarly, the **llm2geneset** framework (Wagner *et al.*, 2024) proposes using LLMs to dynamically **generate custom gene sets** from natural language descriptions. In practice, this means if a researcher describes a biological process or condition, the LLM agent can suggest a set of genes involved, effectively

creating on-the-fly pathway databases for downstream enrichment analysis. These applications illustrate how *small agent modules* (for database retrieval or ontology lookup) together with LLM reasoning can enhance pathway analysis: the LLM proposes hypotheses, a tool-using agent fetches authoritative data, and together they produce verifiable insights.

Another promising direction is using LLMs to **mine gene interaction networks directly from literature**. For instance, one recent evaluation compared 21 models on extracting gene–gene regulatory relations and KEGG pathway components from text. GPT-4 and Claude (large closed models) achieved the best precision/recall (F1 ≈0.44 on regulatory events), while the top open-source model (Falcon-180B) reached F1 ≈0.28. This study concluded that *LLMs can significantly aid automated pathway curation* – for example, by reading papers and building a knowledge graph of who-regulates-whom – but careful model selection and domain tuning are needed for high accuracy. It's easy to imagine an agentic system where a large model "reader" agent scans new publications and a smaller **knowledge-graph builder** agent updates a network of gene interactions. In fact, emerging **agentic graph** frameworks explicitly consider this: an "Agentic Graph" system can coordinate multiple agents and tools in a directed acyclic graph of tasks, e.g. querying a knowledge graph, doing multi-hop reasoning, and expanding the graph with new findings. In a **biotech R&D context**, such a system might start from a few known protein interactions and autonomously grow a complex gene network, potentially uncovering **novel pathway connections** that warrant experimental follow-up.

AI Co-Pilots for Experimental Design

Beyond data analysis, agentic AI is being applied to **plan and execute laboratory experiments**. A cutting-edge example is **CRISPR-GPT**, an autonomous LLM-driven agent for **gene-editing experiment design**. Developed by gene researchers, CRISPR-GPT assists scientists through every step of a CRISPR experiment: selecting the appropriate CRISPR-Cas system, planning the overall experiment, designing guide RNAs for the target genes, choosing delivery methods, drafting the lab protocols, and even analyzing the results of the gene edit. To achieve this breadth, the system combines a powerful reasoning LLM with several specialized components in an agentic architecture. A high-level **LLM Planner agent** breaks down the complex task into sub-tasks and makes decisions (often via a chain-of-thought prompting strategy). A **Task Executor agent** then carries out each step in a structured *state machine* workflow, prompting the human for any needed inputs and invoking external tools at the right junctures. CRISPR-GPT also includes a **"User Proxy" agent**, an LLM that essentially simulates the user's perspective during automated runs (allowing the system to iterate and self-dialogue when the human is not guiding it step by step). Finally, a suite of **Tool-using agents** are integrated – these are plugins that connect to domain-specific software and databases via APIs. For example, when designing guide RNAs, CRISPR-GPT can call out to established gRNA design tools like **CRISPick or ChopChop** to retrieve candidate guide sequences. The LLM then uses reasoning (and even a method akin to a "chain-of-tables" lookup) to identify the most relevant guides from those results. Similarly, for primer design and assay planning, the agent incorporates the Primer3 tool, and for analyzing gene-editing outcomes from sequencing data, it automatically runs the CRISPResso2 pipeline – even emailing the user a report of the mutation outcomes in a recent demonstration.

To ensure domain expertise, CRISPR-GPT includes a **specialized LLM fine-tuned on scientific discussions** (e.g. Q&A threads among CRISPR experts) which provides more knowledgeable responses in biology-specific contexts. This fine-tuned model works in tandem with the general LLM: essentially acting as a **domain expert agent** supplementing the planner. By incorporating **external knowledge retrieval, tool use, and a domain-trained mini-LLM**, the system overcame many limitations of a standalone GPT. The researchers validated CRISPR-GPT by actually following its AI-crafted plans in the lab: they successfully knocked out four genes in human lung cells and activated two genes in melanoma cells under the agent's guidance. This real-world trial showed that a well-designed agent can serve as an **AI**

co-pilot for genome engineering, automating tedious design steps and offering suggestions, while the human focuses on high-level decisions. It's worth noting that CRISPR-GPT's architecture is modular – the authors describe distinct **Planner, Tool Provider, Task Executor,** and **User-Proxy agents**, each an LLM-powered module with a defined role. This reflects a general design pattern: use an LLM for what it's best at (reasoning, flexible decision-making) and delegate domain-specific tasks to either smaller trained models or deterministic tools. As the authors note, such a system could be extended further by plugging in new bioinformatics models – for instance, connecting to the latest **genome or protein foundation models** for structure prediction, or specialized plasmid design software, to broaden the agent's capabilities.

Multi-Agent Systems with Specialized Small Models

One motivation for using **multiple smaller models** instead of a single giant model is efficiency and local deployability. **BioAgents** (Mehandru *et al.*, 2025) is a project that explicitly demonstrates a multi-agent bioinformatics assistant built entirely on **small language models**. Rather than calling OpenAI GPT-4 or another large API, BioAgents uses a locally-run model called **Phi-3** (from Microsoft's *Phi* series) as its engine for each agent role. Phi-3 is on the order of only a few billion parameters, yet fine-tuning and quantization tricks (like QLoRA with low-rank adapters) were applied to maximize its domain performance within a modest footprint. The BioAgents system spawns **multiple specialized agents** from this base model: for example, one agent was fine-tuned on documentation of bioinformatics tools and ontologies, making it an expert in "conceptual genomics" questions (e.g. explaining analysis concepts, suggesting which tool to use). Another agent was configured with **retrieval-augmented generation (RAG)** over workflow documentation and tutorials, specializing in helping with coding and pipeline assembly tasks. These two specialists operate in parallel and feed into a third **Reasoning agent** (also a Phi-3 model) that integrates their outputs and formulates the final answer or solution for the user.

By dividing knowledge this way, BioAgents achieves performance **comparable to human experts** on many bioinformatics Q&A tasks. In an evaluation using real questions from the Biostars forum (ranging from easy conceptual queries to hard pipeline problems), the multi-agent system's answers on *conceptual genomics questions* were on par with expert-written answers. For programming tasks (writing analysis code), BioAgents matched experts on simpler cases, though it still fell short on the most complex coding challenges. This highlights both the promise and current limitations of small-model agents: with focused training, even 100-300M or few-billion-parameter models can achieve **high utility in niche tasks** (often with much lower cost and latency than GPT-4), but difficult tasks may require further improvements or human-in-the-loop feedback. On the upside, because all models are local, BioAgents allows use of **proprietary data and customization**, and avoids the need for costly cloud APIs. This is important in many research settings where patient data or unpublished results must remain secure. The BioAgents work demonstrates that **knowledge distillation and fine-tuning** can produce small models that act as effective agents for specific duties (in this case, one "consultant" agent for genomics knowledge and one "coder" agent for workflows). The orchestration is handled by another small model, showing that *large reasoning capacity can sometimes be approximated by a team of cooperating smaller models*. Indeed, the authors reported that their fine-tuned Phi-3 model's answers were not far off from GPT-4's on the test questions, despite GPT-4 being orders of magnitude larger. This aligns with the emerging view that "Small Language Models are the Future of Agentic AI," whereby networks of task-specific mini-LLMs could replace a single super-LLM for many applications.

Beyond BioAgents, other projects have also explored multi-agent or role-playing setups for bioinformatics. The **GenoAgents** (Liu *et al.*, 2024) is a "team of LLM-based agents" designed to autonomously perform a **gene expression data analysis pipeline**, as introduced with the GenoTEX benchmark. In this scenario, an agent system needs to choose a relevant public dataset, filter and preprocess the data, run statistical tests to find differentially expressed genes, and interpret the results – a series of steps that a human bioinformatician would normally carry out via coding in R/Python. The GenoAgents architecture was given modules for **context-aware planning, iterative error correction, and even a 'domain expert' consultation loop**. For example, one agent might draft a piece of analysis code, another agent (in a reviewer role) checks and debugs that code if it fails, and a third agent can be consulted for domain knowledge (such as whether a certain clinical variable is relevant as a covariate). This collaborative agent approach substantially increases reliability: if the coding agent makes a mistake (common with LLMs writing code), the reviewer agent can catch it and prompt a fix, mimicking a human-in-the-loop debugging session. Experiments with GenoAgents showed that they could indeed automate several steps of real microarray/RNA-seq analyses, though not perfectly – the authors' error analysis pointed out that complex statistical reasoning is still challenging for current models without human guidance. Nonetheless, the development of such benchmarks and prototypes is **pushing agentic AI closer to practical use in data-heavy genomics workflows**. Researchers can now start to measure how well an AI agent performs end-to-end analysis (with human-level criteria for correctness), and identify which specialized agents or tool integrations would most improve the system.

Integrating Domain-Specific Models and Tools

A recurring theme in these systems is the integration of **domain-specific AI models as modular agents**. In gene research, one often needs specialized predictive models – for example, to analyze sequences or structures – which can be plugged into an agent's reasoning loop. By combining these with LLM-based controllers, we get the best of both worlds: *domain rigor* plus *flexible reasoning*. A case in point is the emerging class of **protein language models** and structure prediction models, which can serve as oracles within a larger agent. For instance, Meta AI's **ESMFold** (2022) and DeepMind's **AlphaFold2** are large pretrained models that predict 3D protein structure from amino acid sequences. In a recent study, researchers evaluated how well AlphaFold2, ESMFold, and the sequence-design model **ProteinMPNN** could **predict the success of protein designs in a zero-shot way**. The idea is that an agent trying to design a new protein could use these models to filter or evaluate its designs: AlphaFold or ESMFold can be asked to fold the candidate sequence *in silico* – if the predicted structure is implausible or the confidence (pLDDT) is low, the agent knows the design is likely a failure. ProteinMPNN, on the other hand, can estimate how "natural" or energetically favorable a sequence is for a given target fold. The study found that each model has strengths and weaknesses in discriminating successful vs. failed designs, implying that an ensemble of such tools might be most reliable. In an agentic framework, one could envisage a **design loop**: a large-language-model agent proposes a new protein sequence (perhaps guided by functional requirements provided by the user), then calls AlphaFold/ESMFold (as a *structure agent*) and ProteinMPNN (as a *sequence-evaluator agent*) to score the proposal. If the sequence is predicted to fold well and looks stable, the agent accepts it; if not, the agent refines the design and tries again. Early work in this vein (e.g. IBM's *AlphaDesign* and other autonomous protein designers) underscores that **LLMs plus protein-model tools can automate creative bioengineering tasks** that used to require expert intuition.

Another example of a specialized small model improving efficiency is **MSA Pairformer**, a new protein language model introduced in 2025. MSA Pairformer achieves high accuracy in protein sequence understanding **using only 111 million parameters** – orders of magnitude smaller than models like Meta's 15B-parameter ESM-2. It does so by leveraging **multiple sequence alignments (MSAs)** as input, thereby extracting rich evolutionary signals without brute-force parameter count. For agent designers, this is exciting because it means tasks like predicting protein function or contacts can potentially be offloaded to a lightweight agent model that runs faster and on local hardware. Instead of calling a huge server-side model, an agent could call MSA Pairformer to analyze a sequence (given an MSA from a database) and quickly return insights. This kind of *model miniaturization through domain insight* (in this case, using alignments) is precisely what makes it feasible to incorporate advanced analytics into daily research workflows via agents.

In the realm of genome informatics, **GeneWhisperer** (McNeil *et al.*, 2025) is a system that shows how an LLM agent can interface with classic bioinformatics tools to assist human curators. Genome annotation often involves integrating evidence from various sources (BLAST searches for homology, protein domain finders, gene expression clues) to decide what a gene's function is. GeneWhisperer uses an LLM-based conversational agent to automatically call the right tools for each subtask of manual annotation. For example, if a curator needs to find similar genes in other species, the agent can trigger a BLAST search; if they need functional domain predictions, the agent fetches results from a PFAM or InterPro scan. The LLM consolidates these results and suggests an annotation, which the human can then verify or refine. This significantly speeds up the curation process while keeping a human in control. GeneWhisperer's approach is a prototype of **human-AI collaboration** in genomics: the AI agent handles the laborious data-gathering and preliminary analysis by interacting with specialized tools, and the expert focuses on final interpretation. This system again underscores the value of *small, task-specific agents* within a larger workflow – each external tool effectively acts as an "agent" that the LLM can query, and the LLM itself is fine-tuned to know **which tool to use for which problem** during the conversation.

Finally, even outside these specific examples, there is a general pattern in agentic system design for gene research: **verifier or critic agents** are used to ensure quality control on the outputs of generator agents. GeneAgent's self-verification module is one such example, functioning almost like an internal "fact-checker" agent that scrutinizes the primary LLM's claims. More broadly, researchers have proposed **Prover–Verifier setups** where one model generates an answer or hypothesis and another model (often smaller and specialized) checks its validity. In some configurations, there can even be a "sneaky prover" introducing incorrect statements and a "helpful prover" with correct reasoning, to stress-test the verifier's ability to discern truth. While this red-team/blue-team scenario comes more from AI safety research, the core idea is highly relevant for scientific applications: having an independent verification agent dramatically improves reliability. As we saw, GeneAgent's integration of a verifier led to substantially more trustworthy gene set analyses. We can expect future gene-AI systems to routinely include such internal audit agents – for instance, a small model trained to flag implausible gene interactions, or an agent that cross-checks reported p-values and statistics against the data.

Conclusion

Agentic AI systems are quickly moving from theory to practice in gene research. **Real prototypes and tools now exist** that embed LLMs into multi-step workflows for genomic data analysis, pathway discovery, and even experiment design. These systems employ **hybrid model architectures**: large generalist models (like GPT-4 or fine-tuned LLaMA variants) handle reasoning and natural language interfacing, while **small specialized agents** contribute domain knowledge, run bioinformatics algorithms, retrieve database facts, or verify results. By orchestrating these components, researchers have achieved performance on par with human experts in certain tasks (e.g. answering genomics questions) and even

demonstrated fully AI-devised laboratory experiments. Importantly, many of these agent systems are available for others to use or extend: for example, **GeneAgent** is described in *Nature Methods* (with likely code or data released via NLM), **BioAgents** and **GenoTEX/GenoAgents** have open benchmarks and code published, and the CRISPR-GPT team has shared detailed methodology in *Nature Biomedical Engineering*. This means gene researchers can start experimenting with these AI co-pilots, potentially plugging them into their own pipelines or laboratory routines.

In designing an agentic system for genomics, one should clearly define each agent's role in the **overall architecture** – be it a planner, a tool executor, a data retriever, an analyzer, or a validator. The case studies above show that when each component is optimized for a specific function (often via fine-tuning on relevant data or integrating a proven domain tool), the **combined system is far more powerful and reliable** than a single large model prompt. As the field progresses, we anticipate even tighter integration between LLM agents and bioinformatics platforms: for example, lab informatics systems might come with built-in LLM agents that automatically write data processing scripts, query pathway knowledge bases, or control lab robots based on high-level user goals. By leveraging both the **scale of large models** and the **precision of small models**, agentic AI is poised to accelerate gene research – helping scientists make sense of big biological data, design experiments, and explore hypotheses with an intelligent, context-aware assistant at their side.

Sources: Recent literature and preprints on domain-specific AI agents in biology (GeneAgent, CRISPR-GPT), multi-agent small-LM systems (BioAgents), and benchmarks for genomics AI (GenoTEX/GenoAgents), as well as news releases and evaluations highlighting the effectiveness of these approaches. These examples underscore the practical utility of agentic AI – from accurate pathway analysis to automated experiment planning – in real gene research workflows.