Are LLMs Ready for Neural-integrated Mechanistic Modeling? A Benchmark and Agentic Framework

Zihan Guan[*,1], **Rituparna Datta**[*,1], **Mengxuan Hu**[1], **Shunshun Liu**[1], **Aiying Zhang**[1], **Prasanna Balachandran**[1], **Sheng Li**[1], **Anil Vullikanti**[1]
[*] Equal Contributions, [1]University of Virginia

Abstract

Large language models (LLMs) have shown promise in constructing mechanistic models from data. However, existing evaluations largely focus on simplified settings and fail to capture the complexity of real-world scientific modeling. In practice, such modeling often involves neural-integrated formulations, where a mechanistic model component and a neural network component are jointly constructed, leading to a significantly more complex search space. Motivated by this gap, we introduce the Neural-Integrated Mechanistic Modeling (NIMM) benchmark, which evaluates LLM-generated neural-integrated mechanistic models across three scientific domains. Experiments on NIMM reveal that existing LLM-based approaches struggle to effectively explore this complex space, resulting in limited search stability and solution quality. To address this challenge, we propose NIMMGen, a tree-guided agentic framework that enables diversified exploration via branch-level search and improves solutions through atomic model refinement. Extensive experiments demonstrate that NIMMGen achieves state-of-the-art performance on NIMM, significantly improving search stability and solution quality.

1 Introduction

Mechanistic models, constructed based on domain knowledge of physical, biological, or chemical processes, provide a principled framework for understanding and predicting complex systems and have been widely used across scientific domains, including medicine [29, 22], power systems [32, 40], and computational epidemiology [2, 28, 10, 8]. However, constructing such models typically requires substantial domain expertise and manual effort, making the process time-consuming and difficult to scale. Motivated by recent advances in large language models (LLMs) for code generation and agentic optimization [34, 39, 45], a growing line of work [19, 3] has begun to explore LLM-driven frameworks that learn and refine mechanistic models directly from data.

Among various forms of mechanistic models, neural-integrated mechanistic models [8, 35, 33, 17, 10, 11, 26], which augment mechanistic formulations with neural components, have recently gained significant attention. These models achieve stronger performance by enabling the representation of complex system dynamics. However, this added flexibility also introduces substantial challenges, as both the model structure and neural components must be *jointly constructed and refined*, leading to a significantly more complex modeling and search space.

However, existing evaluation environments in the LLM-based approaches [19, 3] are overly simplified and fail to capture the complexity of neural-integrated modeling. In particular, these works either focus on purely mechanistic models or restrict hybrid models to narrowly defined forms (e.g., additive combinations of neural and mechanistic components), which represent only a limited subset of the broader neural-integrated modeling space. Therefore, this gap limits progress toward automated scientific model discovery, where effective mechanistic models must be constructed for complex real-world scientific systems.

Preprint.

To address this gap, we introduce the *Neural-Integrated Mechanistic Modeling* (NIMM) benchmark, which is designed to systematically evaluate whether LLMs can construct and refine neural-integrated mechanistic models under realistic scientific settings. We use datasets from three diversified domains, including public health, clinical health, and materials science. In contrast to prior work, NIMM additionally incorporates key characteristics of real-world modeling: *partial observability*. For example, in epidemiological datasets, only reported infection counts are observed, while latent compartments such as susceptible or exposed populations remain unobserved. We evaluate a range of baselines, including black-box neural networks, existing mechanistic models, and LLM-driven approaches, under the NIMM framework. Our results show that current LLM-based methods exhibit limited search stability and solution quality when constructing neural-integrated mechanistic models, as evidenced by high root mean squared error (RMSE) and low execution success rates (ESR).

Guided by these observations, we propose **NIMMGen**, a tree-structured agentic framework for neural-integrated mechanistic modeling. NIMMGen organizes model construction as a branching search process, where planning, model generation, execution feedback, and refinement are orchestrated along the search tree. This structure promotes exploration diversity through branch-level search while improving stability via fine-grained, atomic model refinements. Empirically, NIMMGen achieves up to **95.1%** reduction in RMSE on the public health subset, **92.6%** reduction in RMSE on the clinical health subset, and **24.5%** on the materials science subset, compared to prior SOTA LLM-based baselines (Table 1). In terms of stability, NIMMGen further improves executability, yielding up to **76.8%** higher ESR on the public health subset, **20.8%** higher ESR on the clinical health subset, and **18.9%** on the materials science subset (Table 1). Beyond predictive accuracy, we demonstrate that the learned models support counterfactual intervention analysis (e.g., social distancing policies), highlighting their practical utility for decision-making (Figure 4). Further analysis shows that the tree structure helps the LLM progressively refine the model (Figure 3) and introduce meaningful structural updates that improve model adaptability over iterations (Figure 8).

In summary, this paper makes three contributions:

(1) Neural-integrated mechanistic modeling as a new evaluation setting. We identify neural-integrated mechanistic modeling as an important yet underexplored setting for evaluating LLM-based mechanistic modeling, where neural and mechanistic components are jointly constructed. To study this setting, we introduce NIMM, a benchmark that systematically evaluates model construction and refinement under realistic constraints; **(2) Revealing Limitations from the Existing LLM-based approaches.** We show that current methods struggle in this setting, exhibiting unstable search behavior and low-quality solutions, highlighting fundamental challenges in neural-integrated mechanistic modeling; **(3) A tree-guided agentic framework for model construction.** We propose NIMMGen, a tree-guided framework that improves neural-integrated model generation through structured exploration and fine-grained refinement, achieving state-of-the-art performance.

2 Background

Let $\mathcal{S} := (\mathcal{X}, \mathcal{U}, \Phi)$ denote a dynamical system, which consists of $d_{\mathcal{X}}$-dimensional states from its state space $\mathcal{X} \subseteq \mathbb{R}^{d_{\mathcal{X}}}$, the (optional) $d_{\mathcal{U}}$-dimensional action space $\mathcal{U} \subseteq \mathbb{R}^{d_{\mathcal{U}}}$, and a dynamical model Φ (following notation from [19]). At a given time $t \in \mathcal{T} \subseteq \mathbb{R}_+$, the state of the system is denoted as $x(t) \in \mathcal{X}$, while the action of the system is $u(t) \in \mathcal{U}$. In continuous time, the evolution of the system $\mathcal{S}$ is given by

$$\frac{dx(t)}{d_t} = \Phi(x(t), u(t), t) \tag{1}$$

with $\Phi : \mathcal{X} \times \mathcal{U} \times \mathcal{T} \to \mathcal{X}$.

For example, in an epidemic dynamic system, the state of the system $x(t)$ may represent the number of susceptible, infected, and recovered individuals.

Mechanistic Models. We usually approximate Φ using a computational model $f_\theta \in \mathcal{F}$ informed by data, where $\theta \in \Omega(\theta)$ denotes the parameters of the model. Here, the sets $\mathcal{F}, \Omega(\theta)$ denote the space of candidate model classes and parameter space, respectively. The parameters θ are usually calibrated manually [23, 13] or informed from data using statistical methods [4, 46].

In scientific domains, the system dynamics is not purely statistical but constructed grounded with domain knowledge derived from known physical, biological, or chemical processes [29, 22, 32, 10].

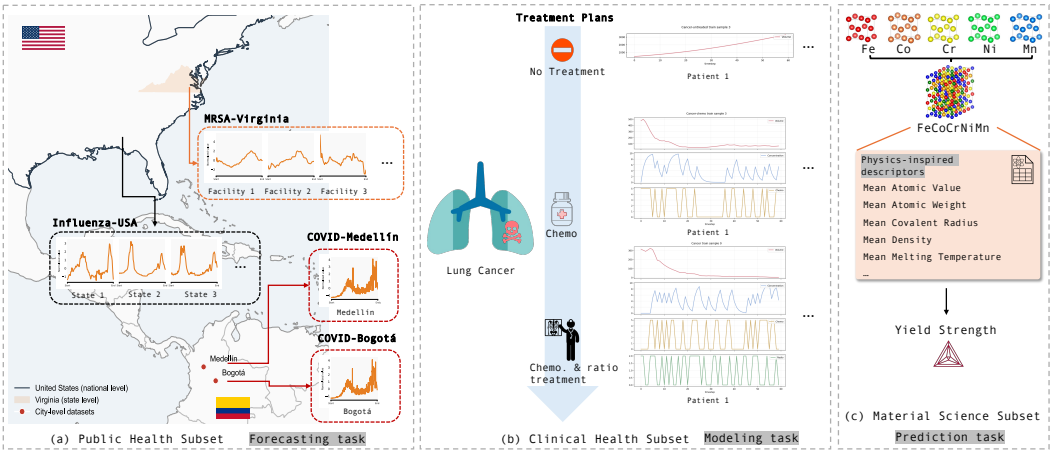


Figure 1: Overview of the NIMM benchmark across three domains: (a) Public Health, (b) Clinical Health, and (c) Materials Science. The benchmark includes epidemic forecasting tasks, lung cancer modeling tasks, and alloy yield strength prediction. NIMM evaluates whether LLMs can construct effective *neural-integrated mechanistic models* across diversified scientific tasks.

This structure requires each component in f to have a clear interpretation and supports reasoning about interventions and counterfactuals.

Neural-integrated Mechanistic Models. Recent work [8, 35, 33, 17, 10, 11, 26] has shown that black-box neural networks can be effective for calibrating mechanistic models. Specifically, let $f_\beta^{NN} : \mathcal{D} \to \Omega(\theta)$ be a neural network parameterized by β that maps the input data domain $\mathcal{D}$ to the mechanistic model parameters space. The resulting mechanistic model is then written as f_θ^{mech}, where the parameter vector θ is directly provided by the f_β^{NN}, i.e.,

$$\theta = f_\beta^{NN}(D), D \in \mathcal{D}. \tag{2}$$

This yields a compositional model f in which the neural network predicts the parameters for the mechanistic component:

$$f(D) = f^{mech} \circ f^{NN}(D) = f^{mech}(f_\beta^{NN}(D)). \tag{3}$$

3 The NIMM benchmark.

3.1 Motivation

Despite preliminary efforts to apply LLMs to mechanistic modeling [19, 3], we argue that their evaluation environment is oversimplified and cannot capture the complexity of neural-integrated modeling. Specifically, they either focus on purely mechanistic models or assume that the target model f takes the form of an *additive combination* of a neural network and a mechanistic component, which is a special case of the general neural-integrated mechanistic model we consider in the paper.

Apart from that, we additionally argue that the exisiting evaluation environments fail to align with the real-world scientific modeling: ❶ Unlike the existing problem setting [19, 3], where mechanistic models are used under *full observation* (i.e., all $d_\mathcal{X}$-dimensional states are observable in the data), a more realistic setting is *partial observation*, where only a subset of the states are observed. For example, in epidemiological datasets, only reported infection counts are observed, while latent compartments, such as susceptible or exposed populations, cannot be easily observed by the surveillance system. ❷ In many scientific applications, the goal is not merely to learn mechanistic models, but also to apply these models for diversified tasks. A desired problem setting should therefore evaluate models in diversified scenarios, rather than focusing on modeling in isolation.

These considerations motivate the need for a benchmark that formalizes the new problem setting and systematically evaluates an LLM's ability to perform neural-integrated mechanistic modeling.

3.2 NIMM benchmark

Datasets. Neural-integrated mechanistic modeling has been widely applied across a variety of domains, including public health [8, 17, 11, 10], clinical health [16, 19], and materials science [26]. Accordingly, in the NIMM benchmark, we include datasets from all three domains. An overview of the datasets is shown in Figure 1.

The three subsets have different tasks (Point ❷). In this section, we describe the **Public Health** subset. The **Clinical Health** and **Materials Science** subsets are described in detail in Appendix B and Appendix C, respectively. For the **Public Health** domain, we use four widely used datasets from prior work: influenza data from the United States (Influenza-USA), methicillin-resistant *Staphylococcus aureus* data from Virginia (MRSA-Virginia), and COVID-19 data from two South American cities—Bogota (COVID-Bogota) and Medellin (COVID-Medellin).

Task Formulation. For public health, we adopt spatial-temporal forecasting as the main task for evaluating the effectiveness of the generated mechanistic models, which is one of the most popular tasks in epidemics. Accordingly, we consider D as a spatial-temporal dataset, where $D_{l,t}$ denotes the signals for the location $l \in \{1, ..., L\}$ at timestamp t. The predicted parameters θ are also *spatially-* and *time-varying*, where $\theta_{l,t}$ denotes the predicted mechanistic parameter for the location l at the timestamp t. This fine-grained formulation enables the mechanistic model to better adapt to heterogeneous dynamics across both space and time. We also note that the signals $D_{l,t} \in \mathbb{R}^d$ may be multivariate, containing a *subset* of the $d_{\mathcal{X}}$-dimensional states (e.g., infection counts) together with other heterogeneous contextual features (Point ❶).

Formally, the problem goal is to use data till time T, namely $\{D_{:,t}\}_{t=1}^{T}$, to predict $\{\hat{y}_{:,t}\}_{t=T+1}^{T+H}$, e.g., the number of infections and deaths during the time period $[T+1, T+H]$, where H is the forecast horizon. More specifically, the historical data $\{D_{:,t}\}_{t=1}^{T}$ is used to train the compositional model f as in Equation 6. In the inference stage, given the historical data, the neural network component f^{NN} first generates spatially- and time-varying parameters $\theta_{l,t}, l \in \{1, ..., L\}, t \in \{1, ..., T+H\}$. Then the mechanistic component f^{mech} is run $T+H$ steps under these predicted parameters, where the last H simulation steps serve as our forecasts.

Neural-integrated Setups. In a typical neural-integrated mechanistic modeling pipeline, both a neural component f^{NN} and a mechanistic component f^{mech} are required. To evaluate the LLM's capabilities across different practical settings, we define the following scenarios: (1) **Mechanistic Mode**: completing a mechanistic model when the neural network component is already provided in the environment; (2) **Hybrid Mode**: jointly completing both the mechanistic model and the neural network model. The hybrid mode is intuitively more challenging than the mechanistic mode, as it involves a substantially larger search space.

Metrics. We aim to evaluate the *effectiveness* and *executability* of the generated model. For effectiveness, we employ popular metrics such as root mean square error (RMSE) for the forecasting task. Formally, the RMSE is defined as $\sqrt{\frac{1}{L \cdot H} \sum_{l,t=T+1}^{t=T+H} \|\hat{y}_{l,t} - y_{l,t}\|^2}$. Following the prior work [8], we adopt a more practical but challenging *real-time* evaluation setup, where the training and testing period shifts over time, to avoid the cherry-picking fallacy. More details about the setup can be found in the Appendix A.4. For executability evaluations, we use the execution success rate (ESR) [7] as the metric, where the ESR evaluates the percentage of code samples that not only compile but also execute without runtime errors. Formally, ESR is defined as $\frac{\text{\# Exectuable Codes}}{\text{Total \# Generations}} \times 100$.

Baselines. The LLM-based baselines include zero-shot prompting and HDTwinGen [19]. In addition, we consider purely black-box models such as Transformers [41] and LSTMs [18]. We also include prior neural–integrated mechanistic models for comparison, including Grad-Metapopulation [17].

3.3 Evaluation Results

Table 1 reports the results of the baseline methods on the NIMM benchmark. We run each method three times and report the average RMSE values together with the ESR. Our results reveal several key observations.

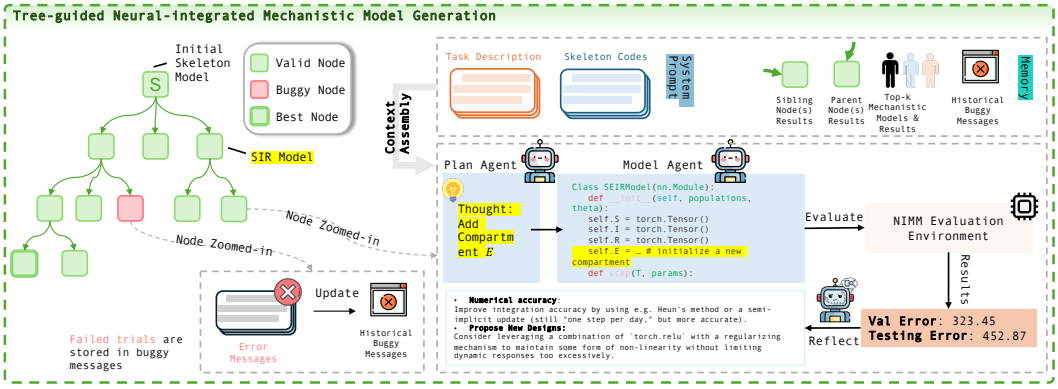


Figure 2: Overview of the NIMMGen framework. The initialized skeleton model serves as the root node, and the model is iteratively refined through a search tree. For each valid node, NIMMGen uses the assembled contextual information to perform three steps: model generation by a model agent, model evaluation in the NIMM environment, and result reflection by a reflection agent. For buggy nodes whose execution fails, the resulting error message is stored to update the contextual memory.

Table 1: Comparison of the effectiveness and executability in the NIMM benchmark.

Dataset ↓	Mode	LSTM	Transformer	Grad-Metapopulation	Zero-shot		HDTwinGen		Ours	
		RMSE ↓	RMSE ↓	RMSE ↓	RMSE ↓	ESR[#]↑	RMSE ↓	ESR ↑	RMSE ↓	ESR ↑
COVID-Bogota	Mechanistic	633.88$_{\pm 158.06}$	901.96$_{\pm 284.58}$	1362.89$_{\pm 165.26}$	1102.62$_{\pm 476.82}$	0.80	339.13$_{\pm 86.53}$	0.69$_{\pm 0.18}$	**325.22**$_{\pm 82.52}$	**0.98**$_{\pm 0.01}$
	Hybrid				1358.24$_{\pm 266.25}$	0.40	765.27$_{\pm 64.58}$	0.57$_{\pm 0.13}$	**398.62**$_{\pm 131.57}$	**0.98**$_{\pm 0.03}$
COVID-Medellin	Mechanistic	586.88$_{\pm 16.96}$	552.39$_{\pm 46.89}$	743.25$_{\pm 25.79}$	908.19$_{\pm 357.17}$	0.80	652.72$_{\pm 102.79}$	0.62$_{\pm 0.10}$	**480.51**$_{\pm 172.47}$	**0.85**$_{\pm 0.09}$
	Hybrid				1026.25$_{\pm 262.64}$	0.75	675.82$_{\pm 190.89}$	0.80$_{\pm 0.13}$	**529.65**$_{\pm 207.86}$	**0.84**$_{\pm 0.11}$
Influenza-USA	Mechanistic	86.69$_{\pm 1.73}$	30.72$_{\pm 0.27}$	117.90$_{\pm 10.93}$	20.92$_{\pm 5.38}$	0.80	8.78$_{\pm 1.49}$	0.89$_{\pm 0.11}$	**7.57**$_{\pm 2.05}$	**0.98**$_{\pm 0.01}$
	Hybrid				783.49$_{\pm 1240.58}$	0.70	163.69$_{\pm 40.75}$	0.60$_{\pm 0.08}$	**8.09**$_{\pm 3.27}$	**0.89**$_{\pm 0.13}$
MRSA-Virginia	Mechanistic	167.08$_{\pm 8.52}$	127.53$_{\pm 45.59}$	134.05$_{\pm 16.29}$	516.36$_{\pm 646.61}$	0.70	93.23$_{\pm 5.19}$	0.76$_{\pm 0.04}$	**33.74**$_{\pm 0.46}$	**0.99**$_{\pm 0.01}$
	Hybrid				809.99$_{\pm 302.54}$	0.65	132.67$_{\pm 42.82}$	0.56$_{\pm 6.44}$	**38.58**$_{\pm 14.62}$	**0.99**$_{\pm 0.01}$
Cancer (NT)	Mechanistic	3.48$_{\pm 1.46}$	4.24$_{\pm 1.74}$	/	73.25$_{\pm 45.1}$	0.70	1.69$_{\pm 2.05}$	0.73$_{\pm 0.04}$	**0.22**$_{\pm 0.01}$	**0.88**$_{\pm 0.04}$
	Hybrid				89.65$_{\pm 28.72}$	0.70	1.97$_{\pm 1.62}$	0.74$_{\pm 0.06}$	**0.35**$_{\pm 0.17}$	**0.89**$_{\pm 0.11}$
Cancer (w/ C)	Mechanistic	0.62$_{\pm 0.17}$	0.58$_{\pm 0.29}$	/	53.52$_{\pm 16.22}$	0.65	1.23$_{\pm 0.04}$	0.77$_{\pm 0.04}$	**0.10**$_{\pm 0.05}$	**0.93**$_{\pm 0.03}$
	Hybrid				69.87$_{\pm 5.82}$	0.65	1.63$_{\pm 0.62}$	0.72$_{\pm 0.06}$	**0.12**$_{\pm 0.01}$	**0.85**$_{\pm 0.07}$
Cancer (w/ C & R)	Mechanistic	0.93$_{\pm 0.34}$	1.76$_{\pm 0.28}$	/	109.25$_{\pm 25.64}$	0.70	0.78$_{\pm 0.46}$	0.82$_{\pm 0.12}$	**0.32**$_{\pm 0.11}$	**0.95**$_{\pm 0.04}$
	Hybrid				129.20$_{\pm 15.97}$	0.65	1.47$_{\pm 0.64}$	0.79$_{\pm 0.03}$	**0.23**$_{\pm 0.17}$	**0.89**$_{\pm 0.06}$
Materials (FCC)	Mechanistic	213.13$_{\pm 58.51}$[*]	/	/	268.24$_{\pm 12.90}$	0.75	192.67$_{\pm 11.30}$	0.74$_{\pm 0.07}$	**145.43**$_{\pm 25.14}$	**0.85**$_{\pm 0.03}$
	Hybrid				274.30$_{\pm 16.72}$	0.65	208.66$_{\pm 25.62}$	0.80$_{\pm 0.06}$	**176.04**$_{\pm 10.52}$	**0.87**$_{\pm 0.02}$
Materials (BCC)	Mechanistic	235.10$_{\pm 15.55}$[*]	/	/	267.84$_{\pm 14.42}$	0.65	192.65$_{\pm 10.51}$	0.79$_{\pm 0.04}$	**166.01**$_{\pm 29.37}$	**0.89**$_{\pm 0.04}$
	Hybrid				277.17$_{\pm 74.21}$	0.80	212.65$_{\pm 14.35}$	0.74$_{\pm 0.03}$	**189.92**$_{\pm 17.25}$	**0.88**$_{\pm 0.01}$

[*] We use an MLP-based neural network rather than LSTM for the Materials task.
[#] The metric is intentionally abused. We report the ESR obtained by running the zero-shot method 20 times.

> *Observation* 1 (**Search Stability**). Existing LLM-based approaches frequently generate mechanistic models that fail due to run-time errors.

The average ESR values for the zero-shot and the HDTwinGen are 0.69 and 0.73, respectively. For the zero-shot method, the low ESR primarily stems from the difficulty of the NIMM tasks: the LLM may fail to fully understand the task requirements and generate runnable models in a single attempt. For HDTwinGen, we hypothesize that the low ESR arises from the combination of its sequential search process and its coarse-grained model revision strategy. On the one hand, HDTwinGen adopts a sequential refinement strategy, in which each candidate is generated by holistically revising models from the previous round. This makes the search highly dependent on a single trajectory, implicitly allowing early errors or suboptimal structures to propagate across iterations. On the other hand, HDTwinGen might unexpectedly employ large and entangled revisions, making the search difficult to control, as failures cannot be easily attributed to a specific change, and useful components may be unintentionally broken.

> *Observation* 2 (**Solution Quality**). Existing LLM-based approaches fail to generate effective models that fit well with the NIMM evaluation environments.

Even among successful trials, the resulting models often still exhibit high RMSE. For example, across the nine evaluated datasets, HDTwinGen yields higher RMSE than deep learning–based approaches (e.g., LSTM) in five cases under the hybrid mode. This suggests that the search process is not only

unstable but also limited in its capacity to discover accurate mechanistic structures. We argue that this weakness also arises from the reliance on a single search trajectory, which restricts the diversity of exploration.

4 Methodology

4.1 Overview

This section presents NIMMGen, a tree-guided agentic framework, designed to address the two observations above. In contrast to existing sequential search strategies, we adopt a tree-guided search strategy [27, 47] that combines branch-level exploration with atomic model refinement. The branching structure prevents flawed intermediate solutions from dominating the entire search process while also increasing search diversity. Atomic edits make each refinement step more localized and controllable. Together, these design choices reduce error propagation and increase search diversity, leading to higher ESR and lower RMSE. An overview of the NIMMGen is provided in Figure 2.

Given the system prompt C, NIMMGen performs $G \in \mathbb{N}^+$ iterations of optimization under a tree-guided search framework. The search starts from an initialized model, which serves as the root node of the search tree. Each node in the tree corresponds to a concrete candidate neural-integrated mechanistic model, and the edges correspond to the atomic refinement applied to the model. In this formulation, a child node is obtained by applying a structural modification (e.g., adding a new compartment or transition) to its parent model.

Instead of sequentially refining a single trajectory, NIMMGen iteratively expands valid nodes in the tree, enabling branch-level exploration of multiple models. For each selected node, the framework assembles contextual information from the task description, skeleton codes, system prompt, parent and sibling node results, top-k historical mechanistic models, and stored buggy messages. Conditioned on this context, a plan agent first proposes an atomic refinement action (e.g., adding a new compartment E to the parent model), and a model agent implements the corresponding structural modification to produce a new candidate model.

The generated model is then evaluated in the NIMM environment, and the resulting feedback is summarized by a reflection agent to update the search memory. If the candidate is buggy, its error message is recorded in the buggy-memory bank, which would be revisited only if there are no valid expandable nodes in the search tree. By combining branch-level exploration with atomic refinement, NIMMGen makes the search process more controllable, preserves diversity across candidate trajectories, and reduces error propagation during model generation.

4.2 Main Loop

System Prompt. The system prompt C consists of two components: (1) T, a task description in natural language that specifies the input and output variables as well as the modeling objective; and (2) S, skeleton code that defines the template of the neural-integrated mechanistic model. The skeleton code S provides essential structural information, such as function signatures and required code formats, which guides the generation of executable model implementations in a predefined format. For illustration, examples of the skeleton code are provided in Appendix F.

Tree-guided Search. Starting from a single model f^0 initialized from the skeleton S, the framework builds a search tree by iteratively expanding one parent node at a time. At iteration g, NIMMGen first constructs an expandable set under depth, child-capacity, and validation-quality constraints. Formally,

$$\mathcal{F}^g_{\text{valid}} = \left\{ v \in \mathcal{T}^{g-1}_{\text{valid}} : d(v) < D_{\max},\ c(v) < C_{\max},\ L_{\text{val}}(v) \leq \tau_{\text{val}} \right\} \tag{4}$$

where $\mathcal{T}^{g-1}_{\text{valid}}$ denotes the valid nodes in the tree at step $g-1$, $d(v)$ denotes the depth of the node v, $c(v)$ denotes the child capacity of the node v, and $L_{\text{val}}(v)$ denotes the validation loss evaluated with the model in node v.

Then, we select the node to expand by lexicographically minimizing the following tuple, where the priority is given in the listed order:

$$v_p^g = \arg \min_{v \in \mathcal{F}_{\text{valid}}^g} \Big(L_{\text{val}}(v),\ -d(v),\ c(v),\ \text{id}(v) \Big),$$

Context Assembly. At each generation step g, NIMMGen assembles contextual information from multiple sources. Specifically, the context includes: (1) the task description and skeleton code in the system prompt; (2) the parent node result and sibling node results associated with the currently expanded node v; (3) a population of top-k historical mechanistic models, along with their evaluation results; and (4) historical error messages from previously failed models. Together, these sources provide rich contextual signals for the agent to reason over prior successes and failures, support branch-level exploration, and guide subsequent model refinement.

Plan Agent. Before model generation, the plan agent proposes a localized refinement plan for the selected node based on the assembled context. Rather than making a holistic revision to the entire model, the plan agent identifies an atomic modification, such as adding a new term, revising an equation, or adjusting a dependency between variables. In Figure 2, the plan agent proposes adding an exposed compartment E to the parent susceptible-infected-recovered (SIR) model, yielding a susceptible-exposed-infected-recovered ($SEIR$) model. This step makes the search process more structured and controllable by refining the model step-by-step with localized modifications.

Modeling Agent. Conditioned on the system prompt C, the assembled context at step g, and the refinement plan proposed by the plan agent, the modeling agent generates a new neural-integrated mechanistic model:

$$f = f^{mech}(f_\beta^{NN}) \sim \texttt{LLM}_m(C, M^g),$$

where M^g denotes the assembled context associated with the selected tree node.

Model Evaluation. The generated model interacts with the NIMM environment for evaluation. Specifically, the objective is to fit the neural-integrated model to the training data, so as to obtain the optimized parameter set β^*. The learnable hyperparameters may vary across tasks; following prior work [19], we provide the full details in the Appendix.

Reflection Agent. After evaluation, the reflection agent summarizes the feedback for the newly generated candidate from multiple perspectives, including both model specification and implementation quality. These reflections are written in natural language and added back to memory to guide future node expansions. Formally, the reflection process can be written as

$$R^g \leftarrow \texttt{LLM}_r(v^g),$$

where v^g denotes the evaluation feedback at step g.

Buggy Nodes. If the generated candidate fails to execute, its error message is stored in the buggy-memory bank, and the corresponding branch is terminated rather than further expanded. After completing the G iterations, NIMMGen selects the best-performing model across all explored nodes as the final output. The buggy nodes can only be revisited for debugging if there are no valid nodes in the expandable set.

Due to the space limit, more detailed descriptions of the NIMMGen algorithm are in the Appendix D.

5 Evaluation

5.1 Main Evaluations

Experimental Setup. To demonstrate the effectiveness of the NIMMGen, we evaluate its performance on the three domains in the NIMM benchmark. The iteration step is chosen as 40, $D_{\max}$ and $C_{\max}$ are chosen as 6, by default. All these hyperparameters are evaluated with sensitivity studies later.

Effectiveness. Table 1 reports the performance comparison between the NIMMGen and the baselines. Overall, the NIMMGen achieves substantially lower RMSE across all datasets. Compared to the existing LLM-based baselines such as zero-shot prompting and HDTwinGen, NIMMGen achieves up to **95.1%** reduction in RMSE on the public health subset, **92.6%** reduction in RMSE on the clinical health subset, and **24.5%** on the materials science subset.

Searching Stability. In terms of stability, NIMMGen further improves executability of the constructed models, yielding up to **76.8%** higher ESR on the public health subset, **20.8%** higher ESR on the clinical health subset, and **18.9%** on the materials science subset.

Comparison between the Mechanistic and the Hybrid Models. We also observe that the hybrid mode generally shows a slightly worse performance than the mechanistic mode. Although this may appear counterintuitive, it aligns with our expectations: In the hybrid mode, the LLM is expected to tune both the mechanistic part and the neural network part, which might be harder to synchronize. The challenge also comes from the fact that the searching space of the hybrid space is considerably larger compared to the mechanistic mode. Therefore, given the same budget, the hybrid mode shows a slightly worse performance.

5.2 Further Analysis

What has been changed during the Optimization? Figure 3 visualizes the loss curve over the course of optimization. We plot both the average RMSE validation loss of all historically generated mechanistic models and the best (lowest) RMSE validation loss achieved so far. As shown, the mean loss exhibits a decreasing trend, suggesting that models generated at later iterations progressively improve the population through guided optimization rather than purely stochastic trial-and-error. The best loss also steadily decreases with iterations, demonstrating that the NIMMGen can continuously refine the model during the search process.

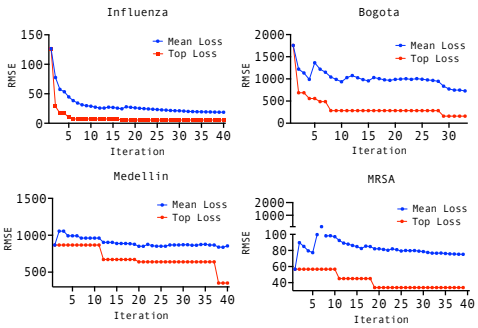


Figure 3: Loss curve over iterations.

Case Study of the Expansion Tree. Figure 8 shows an example of the expansion tree in a typical run in the Influenza-USA dataset. As shown, the expansion tree reveals that NIMMGen explores multiple candidate models through branch-level search, where each node corresponds to a refined model and is evaluated by its validation loss. The best-performing model is reached through a sequence of iterative refinements from the root. To further understand what changes in the mechanistic code contribute to these improvements, we qualitatively analyze the mechanistic models generated by LLMs throughout the path to the best node, with the key modifications summarized in Table 5.

Sensitivity of the Hyper-parameters. In Table 2, we compare the performance of the NIMMGen with different base LLMs, including proprietary models such as `GPT-4.1`, `GPT-4o`, and `GPT-5-mini`, and two frontier open-source coding models, including `Qwen2.5-32B-Coder` and `Qwen3-30B-Coder`. The detailed setups are provided in Appendix E.1. In Table 4, we compare the performance of the NIMMGen under different iteration numbers, ranging from 10 to 80. In Figure 7, we compare the performance of the NIMMGen with different D_{max} and C_{max} choices.

Table 2: Performance comparison under different LLMs.

Model Type	Model Name	Influenza-USA	MRSA-Virginia
Proprietary LLM	`GPT-4.1`	$7.57_{\pm 2.05}$	$33.74_{\pm 0.46}$
	`GPT-4o`	$7.20_{\pm 1.56}$	$38.15_{\pm 5.25}$
	`GPT-5-mini`	$10.20_{\pm 0.37}$	$39.12_{\pm 1.07}$
Open-source LLM	`Qwen2.5-32B-Coder`	$24.26_{\pm 3.12}$	$57.45_{\pm 12.98}$
	`Qwen3-30B-Coder`	$20.24_{\pm 10.08}$	$58.25_{\pm 16.91}$

5.3 Counterfactual Intervention Simulations

In this section, we show the potential validity of the models generated by NIMMGen with a counterfactual intervention simulation experiment.

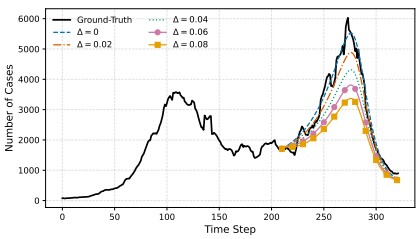


Figure 4: Observed and simulated epidemic trajectories under social distancing interventions.

Counterfactual intervention simulation is a widely used application of mechanistic models, aiming to quantify how hypothetical policies (e.g., social distancing) alter epidemic trajectories relative to the observed baseline [15, 36, 43]. In particular, social distancing policies primarily affect disease transmission by reducing effective contact rates. We therefore simulate social distancing by adjusting the transmission-related component of the model after a given time T'. Let $\theta_{l,t,\tilde{\gamma}}$ denote the effective transmission rates $\tilde{\gamma}$ produced by the neural component

f_β^{NN} at time t for location l. We define

$$\theta_{l,t,\bar{\gamma}} = \begin{cases} \theta_{l,t,\bar{\gamma}}, & t < T' \\ (1-\Delta) \cdot \theta_{l,t,\bar{\gamma}}, & t \geq T' \end{cases}, \forall 1 < l \leq L \tag{5}$$

where Δ denotes the strength of social distancing intervention and Δ a larger Δ corresponds to stronger reductions in population contact.

Figure 4 shows the intervention experiments on the COVID-Bogota dataset ($L = 1$) with $T' = 270$. The neural-integrated mechanistic model is chosen as the best one generated by the NIMMGen. As expected, increasing the strength of the simulated social distancing intervention results in systematic reductions in epidemic peak magnitude and cumulative case counts. Notably, this experiment does not aim to assess the real-world effectiveness of specific policies, as counterfactual ground truth is inherently unavailable. Instead, the objective is to evaluate whether the generated mechanistic models respond to intervention parameters in a manner consistent with epidemiological principles.

5.4 Limitations and Future Works

Despite the great effectiveness we show in the evaluation section, there are several main limitations that can be improved in future work: (1) The definition of the atomic operation is still coarse-grained. A more interesting direction is to use an additional ontology-guided knowledge graph to construct the domain-specific atomic sets. This might potentially further enhance the agentic system with the domain knowledge; (2) We mainly focus on the effectiveness (i.e., RMSE) of these models. Although we show the potential validity of the constructed model by using the counterfactual analysis experiment, we still think it is important to design approaches to verify the semantic correctness of the generated model (e.g., whether each transition is physically grounded in the real world).

6 Related Works

LLMs for Mechanistic Modeling. Large language models (LLMs) have been extensively used for scientific coding tasks [39, 45, 49, 24]. However, most existing approaches focus on one-shot code generation, rather than iterative optimization of model structures. Recently, AlphaEvolve [34] proposed an evolutionary coding framework that automatically generates, evaluates, and refines computer programs for scientific and algorithmic discovery. Nevertheless, it is designed as a general-purpose system, whereas our work focuses specifically on mechanistic modeling, a subfield of scientific coding that places stronger emphasis on validity and knowledge grounding. EpiAgent [12] proposed an agentic framework for epidemic modeling, where the problem settings differ from ours.

The most closely related work to ours is HDTwinGen [19]. However, their problem setting only considers the simple hybrid model, which is a special case of the neural-integrated mechanistic modeling we consider in the NIMM benchmark.

Sequential Deep Neural Networks. Sequential Deep Neural Networks (DNNs) can be adapted to the mechanistic modeling task. Specifically, rather than directly learning a single function f^{mech}, the sequential DNNs aim to learn a black-box function f that aims to directly predict the system state at the next time step based on a window of historical states. Typical architectures such as RNNs [14], LSTMs [18], and Transformers [41] have been widely applied in this setting. Although sequential DNNs show promising performance in capturing complex dynamics in the data, they are heavily relying on training data for generalization and lack substantive interpretability.

7 Conclusion

We first propose NIMM, a benchmark with the novel neural-integrated mechanistic modeling problem setting. Based on the evaluation results on the baselines, we identify some key challenges in the current baseline methods. Motivated by these findings, we design a tree-guided agentic framework, NIMMGen, that explicitly improves search stability and solution quality during iterative refinement. Experiments across three domains demonstrate the effectiveness of the NIMMGen. We believe the paper can contribute to the community by providing a practical benchmark and an agentic system in neural-integrated mechanistic modeling. We hope the paper can serve as a call for future research in developing better methods in this direction.

References

[1] Bijaya Adhikari, Xinfeng Xu, Naren Ramakrishnan, and B Aditya Prakash. Epideep: Exploiting embeddings for epidemic forecasting. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 577–586, 2019.

[2] Abhijin Adiga, Chris J. Kuhlman, Madhav V. Marathe, Henning S. Mortveit, S. S. Ravi, and Anil Vullikanti. Graphical dynamical systems and their applications to bio-social systems. In *International Journal of Advances in Engineering Sciences and Applied Mathematics*, pages 1–19. Springer, 2018. Online version appeared in Dec. 2018.

[3] Harry Amad, Nicolás Astorga, and Mihaela van der Schaar. Continuously updating digital twins using large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[4] Jeffrey L Anderson. An ensemble adjustment kalman filter for data assimilation. *Monthly weather review*, 129(12):2884–2903, 2001.

[5] Ioana Bica, Ahmed M Alaa, James Jordon, and Mihaela Van Der Schaar. Estimating counterfactual treatment outcomes over time through adversarially balanced representations. *arXiv preprint arXiv:2002.04083*, 2020.

[6] Christopher KH Borg, Carolina Frey, Jasper Moh, Tresa M Pollock, Stéphane Gorsse, Daniel B Miracle, Oleg N Senkov, Bryce Meredig, and James E Saal. Expanded dataset of mechanical properties and observed phases of multi-principal element alloys. *Scientific Data*, 7(1):430, 2020.

[7] Le Chen, Nesreen Ahmed, Mihai Capotă, Ted Willke, Niranjan Hasabnis, and Ali Jannesari. Pcebench: A multi-dimensional benchmark for evaluating large language models in parallel code generation. In *2025 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 546–557. IEEE, 2025.

[8] Ayush Chopra, Alexander Rodríguez, Jayakumar Subramanian, Arnau Quera-Bofarull, Balaji Krishnamurthy, B Aditya Prakash, and Ramesh Raskar. Differentiable agent-based epidemiology. *arXiv preprint arXiv:2207.09714*, 2022.

[9] Estee Y Cramer, Evan L Ray, Velma K Lopez, Johannes Bracher, Andrea Brennen, Alvaro J Castro Rivadeneira, Aaron Gerding, Tilmann Gneiting, Katie H House, Yuxin Huang, et al. Evaluation of individual and ensemble probabilistic forecasts of covid-19 mortality in the united states. *Proceedings of the National Academy of Sciences*, 119(15):e2113561119, 2022.

[10] Jiaming Cui, Jack Heavey, Eili Klein, Gregory R Madden, Costi D Sifri, Anil Vullikanti, and B Aditya Prakash. Identifying and forecasting importation and asymptomatic spreaders of multi-drug resistant organisms in hospital settings. *NPJ Digital Medicine*, 8(1):147, 2025.

[11] Rituparna Datta, Jiaming Cui, Gregory R Madden, and Anil Vullikanti. Calypso: Forecasting and analyzing mrsa infection patterns with community and healthcare transmission dynamics. *arXiv preprint arXiv:2508.13548*, 2025.

[12] Rituparna Datta, Zihan Guan, Baltazar Espinoza, Yiqi Su, Priya Pitre, Srini Venkatramanan, Naren Ramakrishnan, and Anil Vullikanti. Agentic framework for epidemiological modeling. *arXiv preprint arXiv:2602.00299*, 2026.

[13] Allan Davids, Gideon du Rand, Co-Pierre Georg, Tina Koziol, and Joeri Schasfoort. Sabcom: A spatial agent-based covid-19 model. *MedRxiv*, pages 2020–07, 2020.

[14] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[15] Yang Ge, Zhiping Chen, Andreas Handel, Leonardo Martinez, Qian Xiao, Changwei Li, Enfu Chen, Jinren Pan, Yang Li, Feng Ling, et al. The impact of social distancing, contact tracing, and case isolation interventions to suppress the covid-19 epidemic: A modeling study. *Epidemics*, 36:100483, 2021.

[16] Changran Geng, Harald Paganetti, and Clemens Grassberger. Prediction of treatment response for combined chemo-and radiation therapy for non-small cell lung cancer patients using a bio-mathematical model. *Scientific reports*, 7(1):13542, 2017.

[17] Zihan Guan, Zhiyuan Zhao, Fengwei Tian, Dung Nguyen, Payel Bhattacharjee, Ravi Tandon, B Aditya Prakash, and Anil Vullikanti. A framework for multi-source privacy preserving epidemic analysis. *arXiv preprint arXiv:2506.22342*, 2025.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Samuel Holt, Tennison Liu, and Mihaela van der Schaar. Automatically learning hybrid digital twins of dynamical systems. *Advances in Neural Information Processing Systems*, 37:72170–72218, 2024.

[20] Harshavardhan Kamarthi, Lingkai Kong, Alexander Rodríguez, Chao Zhang, and B Aditya Prakash. Camul: calibrated and accurate multi-view time-series forecasting. In *Proceedings of the ACM Web Conference 2022*, pages 3174–3185, 2022.

[21] Harshavardhan Kamarthi, Alexander Rodríguez, and B Aditya Prakash. Back2future: Leveraging backfill dynamics for improving real-time predictions in future. *arXiv preprint arXiv:2106.04420*, 2021.

[22] Evangelia Katsoulakis, Qi Wang, Huanmei Wu, Leili Shahriyari, Richard Fletcher, Jinwei Liu, Luke Achenie, Hongfang Liu, Pamela Jackson, Ying Xiao, et al. Digital twins for health: a scoping review. *NPJ digital medicine*, 7(1):77, 2024.

[23] Françoise Kemp, Daniele Proverbio, Atte Aalto, Laurent Mombaerts, Aymeric Fouquier d'Hérouël, Andreas Husch, Christophe Ley, Jorge Gonçalves, Alexander Skupin, and Stefano Magni. Modelling covid-19 dynamics and potential for herd immunity by vaccination in austria, luxembourg and sweden. *Journal of Theoretical Biology*, 530:110874, 2021.

[24] Kin On Kwok, Tom Huynh, Wan In Wei, Samuel YS Wong, Steven Riley, and Arthur Tang. Utilizing large language models in infectious disease transmission modelling for public health preparedness. *Computational and Structural Biotechnology Journal*, 23:3254–3257, 2024.

[25] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

[26] Shunshun Liu, Kyungtae Lee, and Prasanna V Balachandran. Integrating machine learning with mechanistic models for predicting the yield strength of high entropy alloys. *Journal of Applied Physics*, 132(10):105105, 2022.

[27] Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919, 2026.

[28] Madhav Marathe and Anil Vullikanti. Computational epidemiology. *Communications of the ACM*, 56(7):88–96, 2013.

[29] Iris Marchal. Applications of digital twins in medicine. *nature biotechnology*, 43(10):1606–1612, 2025.

[30] Francesco Maresca and William A Curtin. Mechanistic origin of high strength in refractory bcc high entropy alloys up to 1900k. *Acta Materialia*, 182:235–249, 2020.

[31] Valentyn Melnychuk, Dennis Frauen, and Stefan Feuerriegel. Causal transformer for estimating counterfactual outcomes. In *International conference on machine learning*, pages 15293–15329. PMLR, 2022.

[32] Rounak Meyur, Anil Vullikanti, Samarth Swarup, Henning S Mortveit, Virgilio Centeno, Arun Phadke, H Vincent Poor, and Madhav V Marathe. Ensembles of realistic power distribution networks. *Proceedings of the National Academy of Sciences*, 119(42):e2205772119, 2022.

[33] Xiao Ning, Linlin Jia, Yongyue Wei, Xi-An Li, and Feng Chen. Epi-dnns: Epidemiological priors informed deep neural networks for modeling covid-19 dynamics. *Computers in biology and medicine*, 158:106693, 2023.

[34] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[35] Matías Núñez, Nadia L Barreiro, Rafael A Barrio, and Christopher Rackauckas. Forecasting virus outbreaks with social media data via neural ordinary differential equations. *Scientific Reports*, 13(1):10870, 2023.

[36] Anca Rădulescu, Cassandra Williams, and Kieran Cavanagh. Management strategies in a seir-type model of covid 19 community spread. *Scientific reports*, 10(1):21256, 2020.

[37] Nabeel Seedat, Fergus Imrie, Alexis Bellot, Zhaozhi Qian, and Mihaela van der Schaar. Continuous-time modeling of counterfactual outcomes using neural controlled differential equations. *arXiv preprint arXiv:2206.08311*, 2022.

[38] Jeffrey Shaman, Alicia Karspeck, Wan Yang, James Tamerius, and Marc Lipsitch. Real-time influenza forecasts during the 2012–2013 season. *Nature communications*, 4(1):2837, 2013.

[39] Xiangru Tang, Bill Qian, Rick Gao, Jiakang Chen, Xinyun Chen, and Mark B Gerstein. Biocoder: a benchmark for bioinformatics code generation with large language models. *Bioinformatics*, 40(Supplement_1):i266–i276, 2024.

[40] Swapna Thorve, Young Yun Baek, Samarth Swarup, Henning Mortveit, Achla Marathe, Anil Vullikanti, and Madhav Marathe. High resolution synthetic residential energy use profiles for the united states. *Scientific Data*, 10(1):76, 2023.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[42] Srinivasan Venkatramanan, Jiangzhuo Chen, Arindam Fadikar, Sandeep Gupta, Dave Higdon, Bryan Lewis, Madhav Marathe, Henning Mortveit, and Anil Vullikanti. Optimizing spatial allocation of seasonal influenza vaccine under temporal constraints. *PLoS computational biology*, 15(9):e1007111, 2019.

[43] Paulo Cesar Ventura, Alberto Aleta, Francisco Aparecido Rodrigues, and Yamir Moreno. Modeling the effects of social distancing on the large-scale spreading of diseases. *Epidemics*, 38:100544, 2022.

[44] Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2(1):16028, 2016.

[45] Andrew D White, Glen M Hocky, Heta A Gandhi, Mehrad Ansari, Sam Cox, Geemi P Wellawatte, Subarna Sasmal, Ziyue Yang, Kangxin Liu, Yuvraj Singh, et al. Assessment of chemistry knowledge in large language models that generate code. *Digital Discovery*, 2(2):368–376, 2023.

[46] Wan Yang, Marc Lipsitch, and Jeffrey Shaman. Inference of seasonal and pandemic influenza transmission dynamics. *Proceedings of the National Academy of Sciences*, 112(9):2723–2728, 2015.

[47] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[48] Binglun Yin and William A Curtin. First-principles-based prediction of yield strength in the rhirpdptnicu high-entropy alloy. *npj Computational Materials*, 5(1):14, 2019.

[49] Yu Zhang, Yang Han, Shuai Chen, Ruijie Yu, Xin Zhao, Xianbin Liu, Kaipeng Zeng, Mengdi Yu, Jidong Tian, Feng Zhu, et al. Large language models to accelerate organic chemistry synthesis. *Nature Machine Intelligence*, pages 1–13, 2025.

Table 3: Statistical Description of the datasets used in the NIMM benchmark (Public Health).

Dataset Name	Data Source	Observation Period	Spatial Granularity	# Locations	Temporal Granularity	# Timestamps	Disease	# Features	Target
COVID-Bogota	Colombia NIH	2020-04-12 - 2021-01-30	City-level	1	Daily	343 (d)	COVID	11	Infection Counts
COVID-Medellin	Colombia NIH	2020-04-12 - 2021-01-30	City-level	1	Daily	343 (d)	COVID	11	Infection Counts
Influenza-USA	CDC	2017-W34 - 2021-W15[1]	State-level	51	Weekly	191 (w)	Influenza	15	Infection Counts
MRSA-Virginia	Virginia APCD	2016-W01 - 2021-W04	Facility-level	644	Weekly	244 (w)	MRSA	20	Infection Counts

A NIMM benchmark (Public Health)

A.1 Overview

The NIMM benchmark consists of four epidemiological datasets [10, 17, 8, 11] spanning multiple diseases, spatial resolutions, and temporal granularities. It is designed to reflect realistic challenges encountered in public health surveillance, including spatial heterogeneity and varying observation frequencies. Table 3 summarizes the datasets used to construct the NIMM benchmark, and Figure 1 visualizes the geolocations and aggregated time series for each dataset.

A.2 Dataset

The benchmark covers three infectious diseases, i.e., COVID, Influenza, and Methicillin-resistant *Staphylococcus aureus* (MRSA), and includes data from both macro-level surveillance systems and facility-level administrative records. The shape of each dataset can be described using three dimensions: (# Locations, # Timestamps, # Features).

COVID-Bogota & COVID-Medellin These are City-level COVID-19 datasets collected from the Colombia NIH[2]. The dataset contains daily observations of 343 days. Following [17], we adopt 11 features collected from heterogeneous sources, including:

- **Mobility signals**. These signals originate from records of visits to points of interest (POIs) across different regions. According to Google Mobility[3], daily changes in visits to various POI categories are collected and reported relative to the baseline period from January 3 to February 6, 2020. We extract six time-series features capturing mobility changes under different scenarios, including `retail_and_recreation` and `grocery_and_pharmacy`.
- **Google Trends signals**. Google provides search trend statistics for user-specified keywords[4]. We collect search trends for four COVID-19–related keywords, such as "covid-19 en Colombia," "covid-19 hoy," and "covid-19 Bogotá," resulting in three time-series features.
- **Infection-related context**. The Colombian National Institute of Health (NIH) reports daily COVID-19 infection and mortality counts, serving as the official data source in Colombia. From this source, we derive two time-series features.

COVID-Bogota and COVID-Medellin are both collected on the city-level. Therefore, the final shape of the two datasets is (1, 343, 11).

Influenza-USA State-level influenza datasets. Following [8], we adopt 14 signals.

- **Symptoms**. This contains 13 different time series symptoms features collected from the Google symptoms dataset[5], including Fever, Cough, and Sore throat.
- **influenza-like-illness (ILI)**. This is collected by the CDC and records the influenza-like illness across different timestamps.

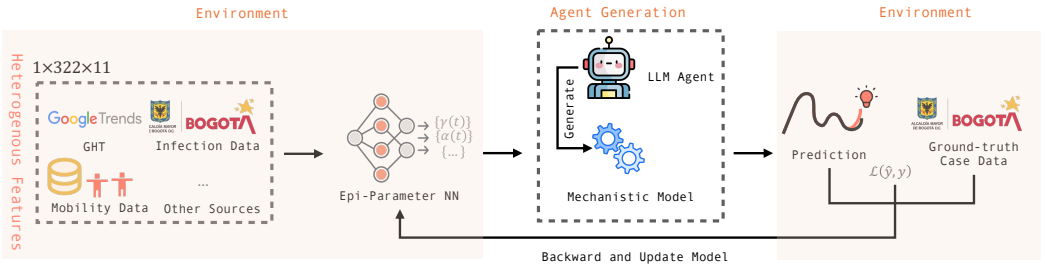
Influenza-USA is collected from 51 states in the USA. Therefore, the final shape of the Influenza-USA dataset is (51, 191, 14).

[2] `https://www.ins.gov.co/Noticias/Paginas/coronavirus-casos.aspx`

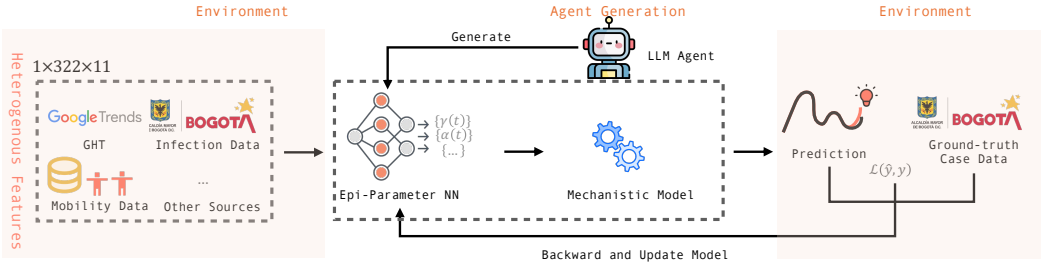
[3] `https://www.gstatic.com/covid19/mobility/Global_Mobility_Report.csv`

[4] `https://trends.google.com/trends/explore?date=today`

[5] `https://github.com/GoogleCloudPlatform/covid-19-open-data/blob/main/docs/table-search-trends.md`



(a) Mechanistic Mode



(b) Hybrid Mode

Figure 5: Experimental setup. (a) Mechanistic mode. (b) Hybrid Mode.

MRSA-Virginia Facility-level MRSA datasets are collected from the Virginia All Payers Insurance Claims dataset (APCD). Built of top of [11], we adopt four signals:

- **Demographical signals**. This includes the number of MRSA cases for each age group. (We have 15 age groups and therefore we have 15 features)
- **Prescriptions**. Number of Prescription claims listed in the insurance data, along with the timestamps.
- **MRSA Infections**. The number of infection counts corresponding to the MRSA.

MRSA-Virginia is collected from 644 healthcare facilities in the state of Virginia. Therefore, the final shape of the MRSA-Virginia dataset is (644, 244, 20).

A.3 Code Generation Modes

We consider two types of code generations modes: *mechanistic model* and *hybrid mode*. In the mechanistic mode, the agent is responsible for generating the codes for f^{mech} only, while the f^{NN} is predefined in the environment. In particular, in the mechanistic mode, we use the CalibNN designed by [8], which has been demonstrated to be effective in extracting the information from heterogeneous data sources across multiple datasets. In the full model mode, the agent is responsible for generating the full model $f^{mech}(f^{NN}(\cdot))$.

A.4 Task & Evaluation

Task Description In the public health subset, we mainly focus on forecasting, which is one of the most popular tasks in epidemics. Formally, the goal in the forecasting problem is to use data till time T, namely $\{\mathbf{x}_t\}_{t=1}^{T}$, and predict $\{\mathbf{x}_t\}_{T+1}^{T+H}$ i.e., the number of infections and deaths during the time period $[T+1, T+H]$, where H is the forecast horizon.

Training and Inference During training, we aim to calibrate the composite model $f(\cdot) = f^{mech}(f_\beta^{NN}(\cdot))$ from the training data $\{D_{:,t}\}_{t=1}^{T}$. Specifically, the neural network $\theta = f_\beta^{NN}(\{D_{:,t}\}_{t=1}^{T})$ takes the entire training sequence as input and predicts the parameters of the mechanistic model. These predicted parameters θ are then used to instantiate the mechanistic model

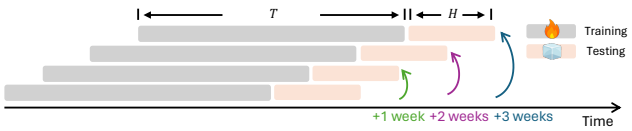


Figure 6: Real-time evaluation Setup.

f^{mech}, which is executed for T time steps to generate simulations for the target state (e.g., infection counts), i.e., $\{\hat{y}_{:,t}\}_{t=1}^{T}$. The training objective is to minimize the root mean squared error (RMSE) between the simulated target values $\{\hat{y}_{:,t}\}_{t=1}^{T}$ and the ground-truth target observations $\{y_{:,t}\}_{t=1}^{T}$:

$$\begin{aligned} \beta^* &= \arg\min_{\beta} \text{RMSE}(\{\hat{y}_{:,t}\}_{t=1}^{T}, \{y_{:,t}\}_{t=1}^{T}), \\ \text{where} \quad \{\hat{y}_{:,t}\}_{t=1}^{T} &= f^{mech}(f_{\beta}^{NN}(\{D_{:,t}\}_{t=1}^{T})). \end{aligned} \tag{6}$$

At inference time, we perform forecasting by running the calibrated composite model for $T+H$ steps. The first T steps reproduce the observed period, while the additional H steps generate predictions for the future horizon $[T+1, \ldots, T+H]$.

Real-time Evaluation Following the popular real-time evaluation pipelines in the epidemics [8, 1, 20, 38], we mainly evaluate the performance of the constructed models using the *real-time forecasting* setup. We simulate real-time forecasting by forcing the models to train only using data available until each of the prediction dates and make predictions for 4 weeks ahead in the future. Data revisions in public health data are large and may affect evaluation and conclusions [9, 21], therefore, we utilize fully revised data following previous papers on methodological advances.

Specifically, we shift the training time window to 1, 2, and 3 weeks ahead to simulate the real-time setting. Figure 6 visualizes the real-time evaluation setups.

Environment Setup The learning rate of the training is set as $5e^{-4}$ with AdamW as the optimizer. The training iteration is set to 1000 to ensure convergence. The H is set to be 8 (weeks) for Influenza-USA and MRSA-Virginia datasets, and 28 (days) for the COVID-Bogota and COVID-Medellin datasets. The T is chosen depending on different datasets. For the COVID-Bogota and COVID-Medellin, the default T is 294 days (42 weeks); For the Influenza-USA, the default T is 191 (weeks); For the MRSA-Virginia, the default T is 244 (weeks).

To use the COVID-Bogota and COVID-Bogota dataset for real-time evaluation, the training period will shift from [1,...,294] to [21,...,315], and the testing period will accordingly shift from [295,...,322] to [316,...,343].

To use the Influenza-USA dataset for real-time evaluation, the training period will shift from [1,...,179] to [4,...,182], and the testing period will accordingly shift from [180,...,188] to [183,...,191].

To use the MRSA-Virginia dataset for real-time evaluation, the training period will shift from [1,...,232] to [4,...,235], and the testing period will accordingly shift from [233,...,241] to [236,...,244].

Experimental Setup The default iteration number for the NIMMGen pipeline is set to 40. All experiments are conducted on two A100 GPUs. We use GPT-4.1 as the default backbone LLM.

A.5 Baselines

As stated in the main text, we opt to use the following three types of baselines.

A.5.1 Neural Sequence Models

LSTM Model We adopt a simple LSTM model consisting of one LSTM layer with hidden size 32, followed by a dense layer with hidden size 16, a rectified linear unit (ReLU) activation function, and a dropout layer with a dropout rate of 0.2. The output layer is a dense layer with linear activation and L_2 kernel regularization with a penalty factor of 0.01. The input dataset $\{D_t\}_{t=1}^{T}$ is first segmented into overlapping five-timestamp windows, where the first four timestamps $(D_{t-4}, \ldots, D_{t-1})$ are

used as input features and the last timestamp D_t is used as the prediction target. Formally, the model learns a mapping

$$\hat{D}_t = f^{\text{LSTM}}(D_{t-4}, D_{t-3}, D_{t-2}, D_{t-1}).$$

In the inference stage, we generate H-step-ahead forecasts in an auto-regressive manner. Starting from the last observed window $(D_{T-3}, D_{T-2}, D_{T-1}, D_T)$, the model predicts

$$\hat{D}_{T+1} = f^{\text{LSTM}}(D_{T-3}, D_{T-2}, D_{T-1}, D_T).$$

This prediction is then fed back into the input window to generate subsequent forecasts:

$$\begin{aligned} \hat{D}_{T+h} = f^{\text{LSTM}}(\hat{D}_{T+h-4}, \hat{D}_{T+h-3}, \hat{D}_{T+h-2}, \hat{D}_{T+h-1}), \\ h = 2, \ldots, H, \end{aligned} \tag{7}$$

where observed values are used when available and predicted values are used once the forecasting horizon extends beyond the training data. This iterative process yields the final H-step forecast $\{\hat{D}_{T+1}, \ldots, \hat{D}_{T+H}\}$.

Transformer Model We first encode the input dimension of the observations into an embedding vector dimension of size 64 through a linear network, followed by the addition of a standard positional encoder. Then, this is fed into the Transformer encoder layers, which had a head size of 4 and contain three layers of Transformer blocks. The final output is mapped back to the input dimension by using an additional linear layer. The training and the inference strategies are similar to those in the LSTM model.

A.5.2 Existing Neural-integrated Mechanistic Models

Grad Meta-population Model A meta-population model requires two key inputs: a contact matrix $\mathcal{C} \in \mathbb{R}^{m \times m}$ and a population vector $N \in \mathbb{R}^m$, where $m = |\mathcal{G}|$ is the number of groups. Each entry $C_{i,j} \in \mathcal{C}$ represents the contact probability between the populations in groups i and j, while each entry $N^j \in N$ records the population of group j. Following [42], the effective number of infected individuals and the effective population of group j after daily movement are defined as:

$$I_j^{\text{EFF}} = \sum_i \mathcal{C}_{i,j} I_i, \quad N_j^{\text{EFF}} = \sum_i \mathcal{C}_{i,j} N_i \tag{8}$$

where "EFF" (effective) refers to infections accounted for, including infected individuals between different subpopulations due to transmissibility and mobility. The conditional force of infection for group j is then given by:

$$\beta_j^{\text{EFF}} = \beta \frac{I_j^{\text{EFF}}}{N_j^{\text{EFF}}}. \tag{9}$$

For each group $i \in [1, ..., m]$, the epidemic evolution $\mathcal{Z}_i = [S_i, E_i, I_i, R_i, M_i]$ from day t to day $t+1$ is formulated as:

$$\begin{aligned} \mathcal{Z}_i(t+1) &= \mathcal{Z}_i(t) + \Delta\mathcal{Z}_i(t), \quad where \\ \Delta S_i(t) &= -\sum_{j=1}^{m} C_{i,j} \beta_j^{\text{EFF}} S_i(t) + \delta(t) R_i(t), \\ \Delta E_i(t) &= \sum_{j=1}^{m} C_{i,j} \beta_j^{\text{EFF}} S_i(t) - \alpha(t) E_i, \\ \Delta I_i(t) &= \alpha(t) E_i(t) - (\gamma(t) + \eta(t)) I_i(t), \\ \Delta R_i(t) &= \gamma(t) I_i(t) + (1 - \delta(t)) R_i(t), \\ \Delta M_i(t) &= \eta(t) I_i(t). \end{aligned} \tag{10}$$

where our forecasting objective $\Delta I_i(t)$ represents the number of newly infected individuals in age group i at time t. Running the meta-population model for T time steps, it outputs the aggregated new cases over all the age groups for the entire training period:

$$\hat{y}_t := \sum_{i=1}^{m} \Delta I_i(t), \forall 1 \leq t \leq T. \tag{11}$$

Following [17], we calibrate the model with a CalibNN. We follow the same implementations as outlined in GitHub[6].

A.5.3 LLM-based Generations

Zero-shot For zero-shot generation, we prompt the LLM model with the exact task description and the skeleton codes as our pipeline. However, it only generates the code once and incorporates no criticism from the reflection agent.

HDTwinGen [19] We adopt the same implementation as described in the official GitHub [7].

B NIMM (Clinical Health) Setup

B.1 Dataset

Following [19], we include a biomedical Pharmacokinetic-Pharmacodynamic (PKPD) benchmark derived from a state-of-the-art mechanistic model of lung cancer tumor growth [16], which has been widely adopted in prior studies on treatment effect modeling and counterfactual simulation [5, 37, 31]. The benchmark simulates tumor progression under different treatment strategies, including: (1) no treatment, (2) chemotherapy only, and (3) combined chemotherapy and radiotherapy. We use this environment to evaluate whether neural-integrated mechanistic models can correctly recover clinically meaningful treatment dynamics and generate stable long-term forecasts.

The underlying system models the temporal evolution of tumor volume $x(t)$ after diagnosis using a mechanistic differential equation that combines intrinsic tumor growth dynamics with treatment from chemotherapy and radiotherapy. The model has two binary treatments: (1) radiotherapy u_t^r and (2) chemotherapy u_t^c.

$$\frac{dx(t)}{dt} = \left(\rho \log\left(\frac{K}{x(t)}\right) - \beta_c c(t) - \left(\alpha_r d(t) + \beta_r d(t)^2\right)\right) x(t), \tag{12}$$

where $K = 30$ denotes the tumor carrying capacity, $\rho = 7.00 \times 10^{-5}$ controls the intrinsic growth rate, $\alpha_r = 0.0398$ denotes the linear radiotherapy cell-kill parameter, β_r denotes the quadratic radiotherapy cell-kill parameter and is set such that $\alpha_r/\beta_r = 10$, $\beta_c = 0.028$ denotes the chemotherapy cell-kill parameter, $c(t)$ represents the chemotherapy drug concentration, and $d(t)$ denotes the radiotherapy dosage.

The chemotherapy concentration follows an exponential decay process with a one-day half-life:

$$\frac{dc(t)}{dt} = -0.5c(t). \tag{13}$$

The chemotherapy action is modeled as a binary intervention that increases the chemotherapy concentration $c(t)$ by $5.0\,\mathrm{mg/m^3}$ of Vinblastine when administered. The radiotherapy action corresponds to delivering a $2.0\,\mathrm{Gy}$ fraction of ionizing radiation at timestep t, where Gy denotes the Gray radiation dose unit.

The treatment assignment process additionally incorporates time-varying confounding. Specifically, chemotherapy and radiotherapy administrations are sampled as Bernoulli random variables whose probabilities depend on the current tumor diameter. The associated probabilities, $p_c(t)$ and $p_r(t)$, represent the probability for the two actions:

$$p_c(t) = \sigma\left(\frac{\gamma_c}{D_{\max}}(\bar{D}(t) - \delta_c)\right), \quad p_r(t) = \sigma\left(\frac{\gamma_r}{D_{\max}}(\bar{D}(t) - \delta_r)\right), \tag{14}$$

where $D_{\max} = 13\,\mathrm{cm}$ denotes the largest tumor diameter, $\delta_c = \delta_r = D_{\max}/2$, $\bar{D}(t)$ denotes the mean tumor diameter, and $\gamma_c = \gamma_r = 2$ controls the extent of time-varying confounding.

[6] https://github.com/GuanZihan/GradMetapopulation.git
[7] https://github.com/samholt/HDTwinGen

Following the original simulation protocol, we generate patient trajectories using Euler-based forward simulation over 60 days. Each trajectory contains sequential observations of tumor progression together with treatment interventions. The benchmark is particularly challenging for mechanistic modeling because the observed outcomes emerge from nonlinear interactions between intrinsic tumor growth, pharmacokinetic decay, radiotherapy response, and time-dependent treatment allocation policies. Consequently, successful modeling requires both accurate dynamical forecasting and robust handling of treatment-dependent temporal confounding.

Using the above Cancer PKPD model, we sample $N = 1000$ trajectories where we sample their initial tumor volumes from a uniform distribution $x(0) \sim \mathcal{U}(0, 1149)$ and use the Cancer PKPD Equation 12 along with the action policy in Equation 14 to forward simulate patient trajectories for 60 days with an Euler stepwise solver. At each timestamp, the treatment action $u_t = (u_t^c, u_t^r)$ consists of the chemotherapy and radiotherapy interventions sampled from the treatment assignment policy.

This results in a trajectory dataset

$$\mathcal{D} = \left\{\tau^{(i)}\right\}_{i=1}^{N}, \quad \tau^{(i)} = \left\{(x_t^{(i)}, u_t^{(i)}, x_{t+1}^{(i)})\right\}_{t=0}^{T-1}. \tag{15}$$

Following the prior work, we independently repeat the above process three times for generating training, validation, and testing datasets, denoted as D_{train}, D_{val}, and D_{test}.

B.2 Task Formulation

Given the current tumor state $x_t \in \mathbb{R}^{d_\mathcal{X}}$ and treatment actions u_t , the objective is to train a neural-integrated mechanistic model $f(\cdot) = f^{mech}(f_\beta^{NN}(\cdot))$ to predict the next state x_{t+1}.

Formally, we model the system dynamics as a composite neural-integrated mechanism:

$$\hat{x}_{t+1} = f^{\text{mech}}\big(x_t, u_t; f_\beta^{\text{NN}}(x_t, u_t)\big), \tag{16}$$

where the neural network component $f_\beta^{\text{NN}}(x_t, u_t)$ is used to predict the parameters for the mechanistic component f^{mech}, and the mechanistic component then evolves the tumor dynamics according to the inferred parameters and treatment conditions.

The model is trained by minimizing the loss function given below:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\left\|\hat{x}_{t+1}^{(i)} - x_{t+1}^{(i)}\right\|_2^2, \tag{17}$$

where N denotes the number of samples in the training dataset, the upper script i denote the sample index $i \leq N$, x_{t+1}^i denotes the ground-truth value, and $\hat{x}_{t+1}^i$ denotes the predicted value by the neural-integrated model $f(\cdot)$.

B.3 Code Generation Modes

As in the public health subset, we also consider two types of code generation modes: *mechanistic model* and *hybrid mode*. In the mechanistic mode, the agent is responsible for generating the codes for f^{mech} only, while the f^{NN} is predefined in the environment. In particular, in the mechanistic mode, we use a simple 2-layer MLP as f^{NN}. In the full model mode, the agent is responsible for generating the full model $f^{mech}(f^{NN}(\cdot))$.

C NIMM (Materials Science) Setup

C.1 Task Formulation

The considered task is to predict the temperature-dependent yield strength of single-phase Face-Centered Cubic (FCC) / Body-Centered Cubic (BCC) high-entropy alloys (HEAs) by integrating a neural network with a physics-based mechanistic model [48, 30]. Similar to the setup in NIMM, the

neural network component is used to predict key mechanistic parameters from alloy composition data D. Then these predicted parameters are used to propagate through a mechanistic model, yielding the final yield strength prediction.

The Materials subset follows the setup of [26], where the task is to train a neural-integrated mechanistic model $f(\cdot) = f^{mech}(f_\beta^{NN}(\cdot))$ that predicts alloy yield strength $\hat{y}_i$ for the given descriptor features x_i of the alloy i. The descriptors include features such as valence electron concentration and atomic volume. Following the neural-integrated mechanistic modeling framework, the prediction $\hat{y}_i$ is given by:

$$\hat{y}_i = f^{mech}(f_\beta^{NN}(x_i)), \tag{18}$$

where f_β^{NN} is parameterized by β, and f^{mech} is a physics-inspired mechanistic model that computes the alloy yield strength.

C.2 Datasets

For FCC alloys, the dataset D contains 29 HEAs and room temperature yield strength, where each input feature characterizes composition-based descriptors derived from elemental properties [44, 26] (e.g., valence electrons, atomic volume). For BCC alloys, the dataset D contains 130 HEAs with temperature-dependent yield strength data, where each input feature also characterizes composition-based descriptors derived from elemental properties [6] (e.g., valence electrons, atomic volume).

C.3 Code Generation Modes

We also consider two types of code generation modes: *mechanistic model* and *hybrid mode*. In the mechanistic mode, the agent is responsible for generating the codes for f^{mech} only, while the f^{NN} is predefined in the environment. In particular, in the mechanistic mode, we use a simple 2-layer MLP as f^{NN}. In the full model mode, the agent is responsible for generating the full model $f^{mech}(f^{NN}(\cdot))$.

D Algorithm

Initialization from Skeleton Code. NIMMGen takes as input a skeleton code template S^0, which specifies only the function signatures and structural interface of the target model, without providing a concrete implementation. Before tree expansion begins, an initialization LLM generates a simple executable model f^0 conditioned on the system prompt C and the skeleton S^0. This initialization step is constrained to produce a minimal and non-complex implementation, so that the search starts from a valid but lightweight baseline model. The resulting model f^0 is used as the root node of the search tree.

Tree-Guided Search. Algorithm 1 summarizes the outer-loop optimization procedure of NIMMGen. Starting from a single initialized skeleton model f^0, the framework builds a search tree by iteratively expanding one parent node at a time. At iteration g, NIMMGen first constructs the valid and buggy frontiers under depth, child-capacity, and validation-quality constraints. It then selects an expandable parent node according to the tree policy: valid nodes are prioritized by predictive performance, while buggy nodes may be revisited during designated debug rounds or when no valid node remains. After a new child model is generated and evaluated, the tree and memory are updated, and the current best valid model $f^\star$ is refreshed accordingly.

Expansion and Evaluation. Algorithm 2 describes the inner-loop expansion step for a selected parent node. Given the system prompt C, the selected node v_p^g, and the accumulated tree memory, NIMMGen first assembles a node-specific context $\mathcal{M}^g$ containing the relevant historical information. A plan agent then proposes an atomic refinement action E^g, such as adding a compartment, revising a dependency, or modifying a mechanistic term. Conditioned on C and $\mathcal{M}^g$, the modeling agent generates a new neural-integrated mechanistic model $f^g = f^{\mathrm{mech}}(f_\beta^{\mathrm{NN}})$. The generated model is subsequently trained and evaluated in the benchmark environment, producing its objective value $J(f^g)$, validation loss $L_{\mathrm{val}}(f^g)$, and bug indicator $\mathrm{bug}(f^g)$, which are returned to the outer search loop.

Algorithm 1 Tree-Guided Search in NIMMGen

Require: system prompt C, skeleton code S, iteration budget G, maximum depth $D_{\max}$, maximum children per node $C_{\max}$, validation threshold τ_{val}, debug period K
Ensure: best model $f^\star$

Generate an initial simple implementation

$$f^0 \sim \text{LLM}_{\text{init}}(C, S)$$

$\mathcal{T}^0 \leftarrow \{f^0\}$
Evaluate f^0 and initialize memory $\mathcal{M}^0$
$\mathcal{T}^0_{\text{valid}} \leftarrow \{f \in \mathcal{T}^0 : \text{bug}(f) = 0\}$; $\mathcal{T}^0_{\text{buggy}} \leftarrow \{f \in \mathcal{T}^0 : \text{bug}(f) = 1\}$
$f^\star \leftarrow \arg\min_{f \in \mathcal{T}^0_{\text{valid}}} J(f)$
for $g = 1, \ldots, G$ **do**

$$\mathcal{F}^g_{\text{valid}} = \left\{ v \in \mathcal{T}^{g-1}_{\text{valid}} : d(v) < D_{\max},\ c(v) < C_{\max},\ L_{\text{val}}(v) \le \tau_{\text{val}} \right\}$$

$$\mathcal{F}^g_{\text{buggy}} = \left\{ v \in \mathcal{T}^{g-1}_{\text{buggy}} : d(v) < D_{\max},\ c(v) < C_{\max} \right\}$$

  if $\mathcal{F}^g_{\text{valid}} \neq \emptyset$ and $g \not\equiv 0 \pmod K$ **then**

$$v^g_p = \arg \min_{v \in \mathcal{F}^g_{\text{valid}}} \big(J(v), -d(v), c(v), \text{id}(v) \big)$$

  else

$$v^g_p = \arg \min_{v \in \mathcal{F}^g_{\text{buggy}}} \big(d(v), c(v), \text{id}(v) \big)$$

  end if
  $(f^g, J(f^g), L_{\text{val}}(f^g), \text{bug}(f^g)) \leftarrow \text{ExpandAndEvaluate}(C, v^g_p, \mathcal{T}^{g-1}, \mathcal{M}^{g-1})$
  $\mathcal{T}^g \leftarrow \mathcal{T}^{g-1} \cup \{f^g\}$
  $\mathcal{M}^g \leftarrow \Psi(\mathcal{M}^{g-1}, f^g)$
  $\mathcal{T}^g_{\text{valid}} \leftarrow \{f \in \mathcal{T}^g : \text{bug}(f) = 0\}$
  $\mathcal{T}^g_{\text{buggy}} \leftarrow \{f \in \mathcal{T}^g : \text{bug}(f) = 1\}$
  if $\text{bug}(f^g) = 0$ and $L_{\text{val}}(f^g) < L_{\text{val}}(f^\star)$ **then**
    $f^\star \leftarrow f^g$
  end if
end for
return $f^\star$

Algorithm 2 $\text{ExpandAndEvaluate}(C, v^g_p, \mathcal{T}^{g-1}, \mathcal{M}^{g-1})$

Require: system prompt C, selected parent node v^g_p, tree $\mathcal{T}^{g-1}$, memory $\mathcal{M}^{g-1}$
Ensure: child model f^g, objective $J(f^g)$, validation loss $L_{\text{val}}(f^g)$, bug flag $\text{bug}(f^g)$

Assemble context

$$\mathcal{X}^g = \Phi\big(C,\ \mathcal{S}(v^g_p),\ \mathcal{M}^{g-1}\big)$$

$$E^g \sim \texttt{LLM}_{\text{plan}}(\mathcal{X}^g)$$

$$f^g = f^{\text{mech}}\left(f^{\text{NN}}_\beta\right) \sim \text{LLM}_m(C, \mathcal{X}^g, E^g)$$

$$\big(J(f^g), L_{\text{val}}(f^g), \text{bug}(f^g)\big) \leftarrow \text{Eval}(f^g)$$

return $\big(f^g, J(f^g), L_{\text{val}}(f^g), \text{bug}(f^g)\big)$

E Detailed Setups

E.1 Open-source Models Setup

To improve the reproducibility of the NIMMGen, we set the temperature to 0 for all experiments using open-source models. The `max_tokens` parameter is set to 2048. We use vLLM [25] to deploy the open-source models and adopt the OpenAI API schema to communicate with them. All experiments

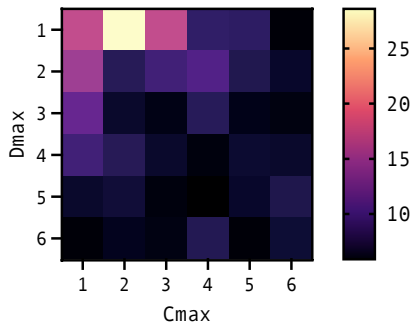


Figure 7: Performance (measured by RMSE) of NIMMGen on the Influenza-USA dataset (mechanistic mode) with different choices of D_{max} and C_{max}. The performance becomes stably good with $C_{max} \geq 3$ and $D_{max} \geq 3$.

are conducted on two A100 GPUs. The two open-source models are `Qwen/Qwen3-Coder-30B-A3B-Instruct`[8] and `Qwen/Qwen2.5-Coder-32B-Instruct`[9] respectively.

Table 4: Performance Comparison under Different Iterations with the Influenza-USA dataset in mechanistic mode.

	10	20	40	80
Influenza-USA	$13.16_{\pm 3.81}$	$11.13_{\pm 1.26}$	$7.57_{\pm 2.05}$	$6.25_{\pm 1.78}$

F Prompts

F.1 Reflection Prompt

```
def build_tree_reflection_prompt(
    *,
    env_name: str,
    iteration: int,
    global_memory: str,
    target_loss: Optional[float],
) -> str:
    target_text = "as low as possible" if target_loss is None else f"below
    ↪  {target_loss:.6f}"
    return dedent(
        f"""
        You are reviewing a mechanistic-model tree search for {env_name}.
        The objective is to drive the test loss {target_text}.

        Global tree memory:
        ```
        {global_memory}
        ```

        Call the `ReflectionSubmission` tool with short, concrete guidance for the
        ↪  next expansion.
        Focus on:
        - which structural edit classes are promising,
        - whether the current best branch should be deepened,
        - which buggy nodes are worth repairing,
        - and which repeated failure patterns should be avoided.
```

[8] https://huggingface.co/Qwen/Qwen3-Coder-30B-A3B-Instruct
[9] https://huggingface.co/Qwen/Qwen2.5-Coder-32B-Instruct

```
        This is reflection round {iteration}.
        """
    ).strip()
```

F.2 Skeleton Codes Prompt

The skeleton codes are very similar across different datasets. Here, we use Influenza-USA as an example.

```
class StateDifferential:
    def __init__(self, population, theta):
        # This section provides prefilled code that enables the model to incorporate
        ↪  meta-population dynamics.
        # Initialize the parameters
        self.device = 'cuda:0'

        # migration matrix defines a matrix describing the contact probability among
        ↪  N locations. Its size is (N, N)
        self.migration_matrix = theta

        # num_agents defines a vector describing the population of each location.
        ↪  Its size is (N)
        self.num_agents = population

        # self.num_patches is N.
        self.num_patches = self.migration_matrix.shape[0]

    def step(self, params, T):
        # TODO: Fill in the code here
        # You can use any number of parameters from ['beta', 'kappa', 'symprob',
        ↪  'epsilon', 'alpha', 'gamma', 'delta', 'mor'] to construct the
        ↪  mechanistic model.
        # The shape of `params`: (self.num_patches, T, 8)
        # Note that 'beta_t', 'kappa_t', 'symprob_t', 'epsilon_t', 'alpha_t',
        ↪  'gamma_t', 'delta_t', and 'mor_t' are all of shape (self.num_patches)
        # **You MUST NOT using any sub loops exceept inside this loop**
        for t in range(T):
            (
                beta_t,
                alpha_t,
                gamma_t,
                delta_t,
                kappa_t,
                epsilon_t,
                symprob_t,
                mor_t,
            ) = params[:, t, :].unbind(dim=-1)
            S_t = S[-1]
            E_t = E[-1]
            I_t = I[-1]
            R_t = R[-1]
            M_t = M[-1]

            # TODO: Fill in the code here
            # - Remember this is a deterministic ODE Simulator
            # - Return `I` as tensors, where `I` denotes the simulated infected
            ↪  counts along different time stamps; `I` should be the shape of
            ↪  (`T`+1, self.num_patches); params have a grad_fn, so make sure that
            ↪  `I` also remains in the computation graph.

        return I
```

Table 5: Evolution of Mechanistic Model Design and Validation Loss for the Influenza-USA dataset

Depth	Parent model	Best refinement explored at this stage	Test. loss	Val loss
1	Seed models	Best initial seed: SEIRD baseline with latent exposure, recovery, and mortality	11.79	17.03
2	SEIRD	Add external importation into the exposed compartment E to model sporadic introductions	11.63	8.95
3	SEIRD + importation	Add a cumulative incidence compartment C to track total new infections over time	8.25	7.68
4	SEIRD + patch-specific importation	Add a hospitalization compartment H to capture severe-case progression	9.06	8.89
5	SEIRHD + patch-specific importation	Make hospital mortality patch-specific to capture spatial heterogeneity in severe outcomes	8.27	6.23
6	Best depth-5 model	Further refinements explored: time-varying transmission, patch-specific hospitalization, direct out-of-hospital mortality, and patch-specific waning immunity; none improved over depth 5	7.97	7.54

Note that the 'population' and the 'theta' are two constant vectors passed by the environment. We used the ones collected from a popular source[10].

G Qualitative Analysis on the Generated Mechanistic Codes

G.1 Evolving Code Examples

Table 5 summarizes the major structural refinements explored during a run in the Influenza-USA dataset. Rather than listing every generated node, the table highlights the best refinement discovered at each effective stage of the search, together with its validation and test losses. This provides a compact view of how the search progressed from simple compartmental baselines toward more expressive neural-integrated mechanistic models.

Several findings emerge from Table 5. First, the largest early improvement came from introducing external importation into the exposed compartment, which substantially reduced the validation loss ($17.03 \rightarrow 8.95$) and suggests that sporadic external introductions were essential for explaining the observed influenza dynamics. Second, branch-level exploration was important: although one strong intermediate branch introduced cumulative incidence tracking (row 3), the final best model (row 5) emerged from a different branch that combined patch-specific importation, explicit hospitalization, and patch-specific hospital mortality. This indicates that preserving multiple mechanistic hypotheses during search was beneficial.

H Impact Statement

This paper proposed a benchmark for evaluating the LLM's capability in generating neural-integrated mechanistic models, and an agentic framework for enabling the LLMs to optimize the neural-integrated mechanistic models. Our main aim is to explore potentially effective methods for generating the neural-integrated mechanistic models.

[10] `https://github.com/NSSAC/patchflow-data/tree/main`

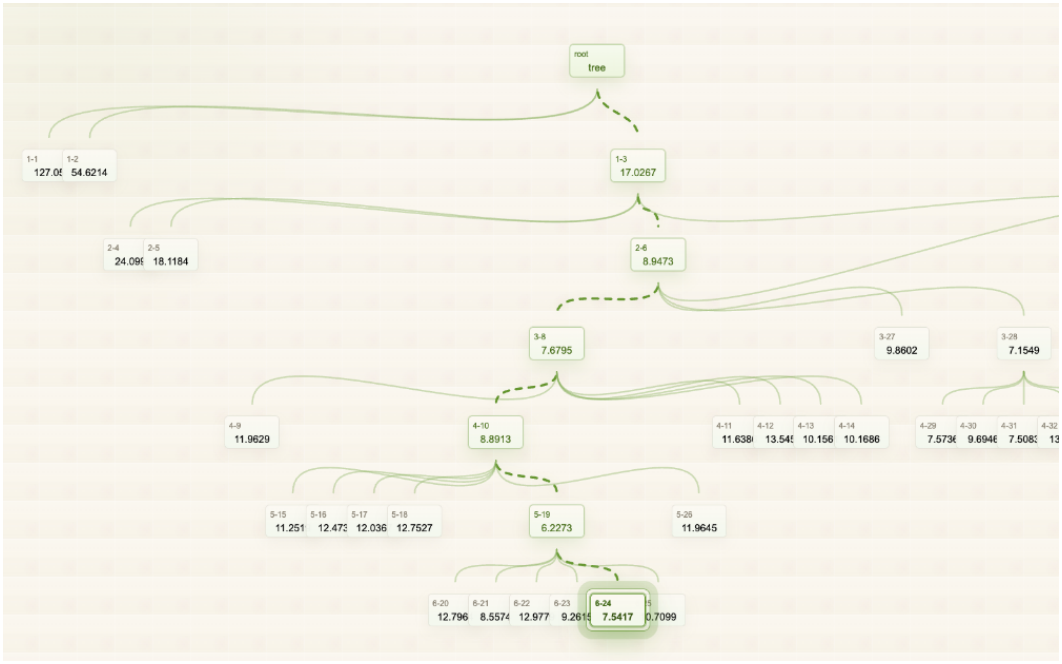


Figure 8: An example of the expansion tree on the Influenza-USA dataset. The number in each node denotes the validation loss. The best node is highlighted in green, and the path from the root node to this node is also highlighted.