How Do Large Language Models Perform in Dynamical System Modeling

Xiao Luo[♡], **Binqi Chen**[♠], **Haixin Wang**[♡], **Zhiping Xiao**[♢], **Ming Zhang**[♠], **Yizhou Sun**[♡]
[♡] University of California, Los Angeles [♠] Peking University
[♢] University of Washington
{xiaoluo, whx, yzsun}@cs.ucla.edu, cbq@stu.pku.edu.cn, mzhang_cs@pku.edu.cn, patxiao@uw.edu

Abstract

This paper studies the problem of dynamical system modeling, which involves the evolution of multiple interacting objects. Recent data-driven methods often utilize graph neural networks (GNNs) to learn these interactions by optimizing the neural network in an end-to-end fashion. Although large language models (LLMs) have shown superior zero-shot performance in various applications, their potential for modeling dynamical systems has not been extensively explored. In this work, we design prompting techniques for dynamical system modeling and systematically evaluate the capabilities of LLMs on two tasks, including dynamic forecasting and relational reasoning. An extensive benchmark LLM4DS across nine datasets is built for performance comparison. Our experimental results yield several key findings: (1) LLMs demonstrate competitive performance without training compared to state-of-the-art methods in dynamical system modeling. (2) LLMs effectively infer complex interactions among objects to capture system evolution. (3) Prompt engineering plays a crucial role in enabling LLMs to accurately understand and predict the evolution of systems.

1 Introduction

Dynamical system modeling is a critical field with wide-ranging applications, from physical simulations (Pfaff et al., 2021; Rajani et al., 2020) to epidemiological tracking (Cury et al., 2021; Mutuvi et al., 2020). These systems typically involve multiple interacting agents. To model such systems, researchers have developed various data-driven approaches (Kipf and Welling, 2017; Xu et al., 2019; Zheng et al., 2022; Li et al., 2022a; He et al., 2022), often leveraging graph neural networks (GNNs). GNN-based methods commonly employ a message passing mechanism to iteratively update node representations, allowing them to predict the state of the system at the next time step. By repeatedly feeding the output back into the model as input, these approaches can generate entire system trajectories in an autoregressive manner (Pfaff et al., 2021).

Despite significant progress in dynamical system modeling, recent approaches often face serious limitations. In particular, these methods (Pfaff et al., 2021; Huang et al., 2020) typically require extensive data for end-to-end training and struggle with poor generalization performance when applied to new scenarios. In contrast, large language models (LLMs) have shown strong capabilities in zero-shot and few-shot prompting scenarios across various domains (Wu et al., 2024). For instance, Liang et al. (2024) has shown the capacity of LLMs with in-context learning in bioinformatics. Gruver et al. (2024) have explored their zero-shot performance in time series forecasting by considering time series as numerical digit strings. However, the evaluation of LLMs specifically for dynamical system modeling remains underexplored, which presents a significant gap in our understanding of LLMs' capabilities in this crucial field.

In comparison to time series analysis (Gruver et al., 2024; Yu et al., 2023b; Jin et al., 2024), applying LLMs to dynamical system modeling consists of three challenges. First, dynamical systems typically involve multiple agents whose interactions have to be accurately understood to enable reliable dynamic forecasting (Xu et al., 2023). Thus, it is crucial to determine whether LLMs can effectively capture these relationships for modeling purposes. Second, dynamical systems often include more complex data structures and edge information, such as three-dimensional positional data, which are significantly more challenging to handle compared to simpler one-dimensional time series data. Third, long-term prediction (Méndez et al., 2023; Pfaff et al., 2021) is a major challenge in dynamical system modeling, as it typically requires generating intermediate states to forecast the evolution. Given these three significant challenges, we expect careful design and evaluation for integrating LLMs into dynamical system modeling.

In this paper, we introduce LLM for Dynamical Systems (LLM4DS), an extensive benchmark for comparing LLMs with existing non-LLM approaches for dynamical system modeling. To effectively combine LLMs with dynamical systems, we introduce prompt engineering beginning by defining the system and the task. Next, we convert the adjacency matrix into a sequence of object pairs and represent the historical observation matrix as a string of tokens. For long-term predictions, we adopt an iterative rollout strategy, where the output of LLMs, i.e., the next-time-step state, is iteratively fed back into the model. To further enhance performance, we incorporate one example as a guidance during in-context learning. Beyond dynamic forecasting, we also evaluate LLM performance on the relational reasoning task, which involves predicting whether two objects are interacting based on historical trajectories. We conduct extensive experiments with pre-trained parameters and supervised fine-tuning on nine datasets across physical simulations, pedestrian trajectories and opinion migration with three key observations. *Firstly*, without additional training, LLMs can achieve competitive performance in dynamical system modeling in comparison to state-of-the-art approaches, especially for short-term predictions. *Secondly*, LLMs are capable of inferring complex interactions between different objects, which are crucial to understanding the evolution of systems. *Thirdly*, prompt engineering (e.g., one-shot prompting) is essential for helping LLMs accurately comprehend and predict the dynamics of systems. In addition, our work highlights important characteristics and certain limitations of LLMs in dynamical system modeling, which can provide valuable direction for future research in the related areas.

The contribution of this work is summarized as follows: (1) We study the problem of integrating LLMs in dynamical system modeling and attempt extensive prompt engineering techniques to adapt LLMs to different tasks in this domain. (2) We build a comprehensive benchmark called LLM4DS consisting of nine popular datasets for comparing LLMs with existing non-LLM approaches on dynamical system modeling tasks. (3) Extensive experiments on dynamic forecasting and relational reasoning demonstrate that LLMs can achieve competitive performance on dynamical system modeling. We further provide eight important observations as a guidance for future research.

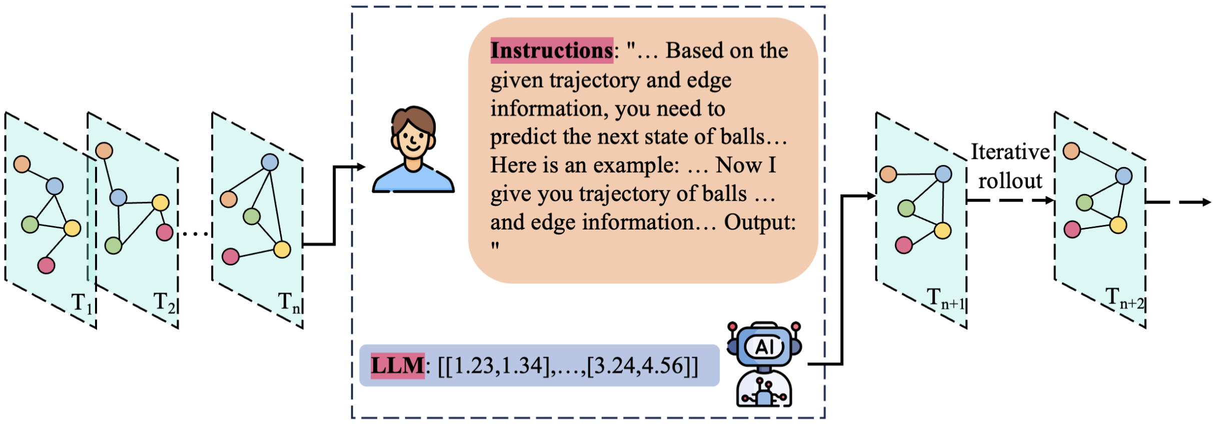
2 Related Work

2.1 Dynamical System Modeling

Interacting dynamical system modeling (Cong et al., 2023; Yu et al., 2023a; Huang et al., 2024; Gastinger et al., 2024) has gained significant attention across various fields, including computational fluid dynamics and molecular biology (Lan et al., 2022; Li et al., 2022b; Bishnoi et al., 2022; Sun et al., 2023; Yu et al., 2024). Early works in this area employ convolutional neural networks to model dynamics on regular grids (Peng et al., 2020). Recent approaches (Wu et al., 2023; Deng et al., 2023; Pfaff et al., 2021) typically use graph neural networks (GNNs) to capture spatio-temporal relationships across multi-agent systems via the message passing mechanism. In addition, several graph ODE approaches (Luo et al., 2023; Huang et al., 2020) have been developed to model dynamic systems in a continuous manner, which can be applied to systems with missing observations. However, these approaches generally require large amounts of training data, which can be expensive to generate using simulation software (Pfaff et al., 2021). Motivated by the advancements of LLMs in zero-shot and few-shot learning scenarios (Gruver et al., 2024; Kojima et al., 2022; Zhang et al., 2024a,c), this paper explores the potential of LLMs for dynamical system modeling, which aims to close the gap between two worlds.

2.2 Large Language Models

Among various foundation models (Feng et al., 2024; Liu et al., 2024a), large language models (LLMs) such as GPT (Achiam et al., 2023) and LLaMA (Touvron et al., 2023) have shown effectiveness across a variety of tasks, including question answering (Kamalloo et al., 2023; Nguyen et al., 2023), knowledge graphs (Yang et al., 2024) and mathematical reasoning (Ahn et al., 2024; Srivastava et al., 2024). To further improve performance, several in-context learning techniques such as chain-of-thought prompting (Wei et al., 2022) and few-shot prompting (Ma et al., 2023) have been developed. These prompt engineering methods enable LLMs to achieve strong performance without additional training. While numerous studies have focused on evaluating LLMs on different tasks including time-series forecasting (Gruver et al., 2024) and creative writing (Gómez-Rodríguez and



Instructions: "… Based on the given trajectory and edge information, you need to predict the next state of balls… Here is an example: … Now I give you trajectory of balls … and edge information… Output: "
LLM: [[1.23,1.34],…,[3.24,4.56]]
AI
Iterative rollout
T1
T2
Tn
Tn+1
Tn+2

Figure 1: An overview of our procedure for LLM-based dynamic forecasting. We feed the prompts into LLMs and then require LLMs to output the predictions at the next timestamp in the matrix form. To generate long-term predictions, we utilize an iterative rollout strategy. More details of prompts can be found in Appendix D.

Williams, 2023) and rule discovery (Liu et al., 2024b), their capabilities in dynamical system modeling remain underexplored. To address this gap, we incorporate LLMs into dynamical system modeling and introduce a comprehensive benchmark LLM4DS that facilitates extensive comparisons and demonstrates the effectiveness of LLMs for dynamical system modeling.

3 Our Benchmark: LLM4DS

3.1 Problem Definition

In this work, we mainly consider two tasks of dynamical system modeling, i.e., dynamic forecasting and relational reasoning.

Dynamic Forecasting. We first utilize graphs to describe a multi-agent dynamical system, i.e., $G^t = (\mathcal{V}, \mathcal{E}^t, \boldsymbol{X}^t)$, where $\mathcal{V}$ denotes the node set and $\mathcal{E}^t$ denotes the edge set at the timestamp t. $\boldsymbol{X}^t$ records the states of different objects at the timestamp t. Given the historical information $\boldsymbol{G}^{1:T_{obs}} = \{G^1, \cdots, G^{T_{obs}}\}$, our target is to predict the states in the future states, i.e., $\boldsymbol{X}^{T_{obs}+1:T_{end}} = \{\boldsymbol{X}^{T_{obs}+1}, \cdots, \boldsymbol{X}^{T_{end}}\}$.

Relational Reasoning (Xu et al., 2023). Given the trajectories of multi-agent dynamical systems, i.e., $\boldsymbol{X}^{1:T} = \{\boldsymbol{X}^1, \cdots, \boldsymbol{X}^{T_{end}}\}$, we aim to infer the relationship between different objects, i.e., $\mathcal{E} = \{e_{ij}\}_{i \neq j \in \mathcal{V}}$. where $e_{ij} = 1$ indicates the interaction between object i and object j. This task can provide a direction evaluation on whether LLMs can understand interaction across different objects.

3.2 Prompt Engineering

To incorporate LLMs into dynamical system modeling, we aim to carefully design prompts with extensive contexts. We first provide the context to describe the dynamical system and then introduce the goal of tasks. The historical observations $\boldsymbol{X}^{1:T_{obs}}$ are converted into a tensor as the input. To enhance the understanding of LLMs, edge information is depicted using a list of positive pairs for dynamic forecasting. Due to the limit of token lengths for LLMs, we utilize a rollout strategy, which requires LLMs to output the state at the next timestamp and include it in the input as historical information in an iterative manner. An overview of the framework can be seen in Figure 1. As for relational reasoning, we feed the trajectories into LLMs and require LLMs to generate binary outputs for each pair, i.e., interact or isolate. To regularize the output format of LLMs, we adopt the one-shot prompting strategy, which includes an example as the guidance of output. The examples of prompts for both tasks can be found in Appendix D.

3.3 Compared Models

We adopt two popular LLMs, i.e., GPT-3.5 (Achiam et al., 2023) and Llama3-70B (Touvron et al., 2023) for performance comparison. GPT-3.5 is a closed-source pre-trained LLM from OpenAI, and there we utilize its API to output the results. Llama3-70B is an open-source pre-trained LLM, which consists of 70 billion parameters. For the dynamic forecasting task, we compare LLMs with extensive state-of-the-art non-LLM approaches including LSTM (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014), NODE (Chen et al., 2018), NRI (Kipf et al., 2018), and EGNN (Satorras et al., 2021). We also include LLMTime (Gruver et al., 2024) as the baseline,

Dataset	Method	2-step		4-step		6-step		8-step		10-step		12-step	
		MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
Springs	LSTM	12.928	2.584	13.169	2.672	13.393	2.755	13.627	2.843	13.868	2.937	14.114	3.033
	GRU	9.477	1.388	10.008	1.529	10.507	1.670	10.931	1.795	11.304	1.907	11.645	2.012
	NODE	10.051	1.531	10.344	1.607	10.639	1.684	10.940	1.764	11.247	1.848	11.553	1.936
	NRI	0.890	0.036	1.832	0.069	2.835	0.163	3.897	0.304	5.024	0.497	6.216	0.747
	EGNN	1.394	**0.034**	1.610	**0.047**	1.830	**0.062**	**2.052**	**0.080**	**2.275**	**0.100**	**2.601**	**0.137**
	TimeLLM	24.811	3.855	24.989	3.867	25.072	3.878	25.558	3.981	27.690	4.393	29.635	4.737
	GPT-3.5	2.589	1.391	4.355	1.509	6.383	1.692	7.925	1.889	9.293	2.074	10.639	2.473
	GPT-3.5 (SFT)	1.421	0.396	3.023	1.329	4.551	1.512	5.993	1.629	7.361	1.894	8.707	1.993
	Llama3-70B	**0.368**	0.040	**0.915**	0.057	**1.550**	0.498	2.360	1.016	3.296	1.236	4.352	1.453
Charged	LSTM	21.345	7.273	22.078	7.800	22.773	8.315	23.464	8.844	24.147	9.384	24.821	9.934
	GRU	19.293	6.003	20.303	6.684	21.221	7.328	22.058	7.941	22.838	8.531	23.589	9.114
	NODE	20.214	6.605	20.872	7.060	21.518	7.522	22.178	8.010	22.845	8.518	23.519	9.044
	NRI	5.141	1.547	7.583	**2.202**	**9.188**	2.471	11.094	3.158	12.913	3.921	14.770	4.955
	EGNN	8.517	1.504	9.072	1.623	9.393	**1.680**	**9.636**	**1.735**	**9.837**	**1.792**	**10.500**	**2.085**
	TimeLLM	22.703	9.828	25.269	9.939	33.710	11.429	41.271	12.145	55.798	13.870	60.913	14.671
	GPT-3.5	4.254	0.977	10.411	3.117	15.010	5.860	19.044	7.454	23.226	8.042	27.442	9.638
	GPT-3.5 (SFT)	3.725	0.823	7.167	2.897	11.406	3.214	15.653	6.072	20.254	7.973	25.441	9.028
	Llama3-70B	**2.973**	**0.724**	**6.566**	2.681	10.340	3.501	14.857	5.510	20.033	7.569	25.652	8.658

Table 1: The MSE and MAE ($\times 10^{-2}$) of compared methods on *Springs* and *Charged*. **Bold** numbers indicate the best results while underline numbers imply the second best performance.

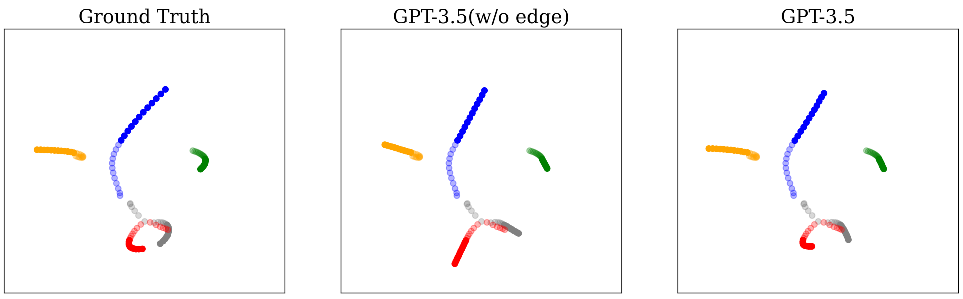


Figure 2: Visualization of the compared methods on *Charged*. Semi-transparent paths represent observed trajectories and solid paths indicate the predictions. From the results, we can find that without edge information, GPT-3.5 cannot accurately predict the trajectory in red.

which considers each object independently. For relational reasoning, we include a random baseline, which selects one of the answers in a uniform fashion. By comparing the performance with random guessing to test whether LLMs have explicit reasoning ability for the interaction between objects.

3.4 Datasets

We adopt nine datasets based on physical simulations, pedestrian trajectories and opinion migration to evaluate the performance of LLMs in comparison with competitive baselines. In detail, we involve two physical simulation datasets, i.e., *Springs* and *Charged* (Kipf et al., 2018). *Springs* contains the dynamic trajectories of interconnected springs following Hooke's law. *Charged* is made up of the trajectories of electronics in electromagnetic phenomena. We also involve a pedestrian trajectory benchmark *ETH-UCY* (Lerner et al., 2007; Pellegrini et al., 2009) with five datasets, i.e., *ETH*, *HOTEL*, *UNIV*, *ZARA1*, and *ZARA2*. The *Social* dataset (Gu et al., 2017) models the opinion migration of different people in a social network. We conduct dynamic forecasting on all nine datasets and relational reasoning on *Springs* and *Charged* following previous works.

4 Experiment

4.1 Experimental Settings

We utilize API to access GPT-3.5 and download the weight of Llama3-70B for evaluation. The prediction length varies among $\{2, 4, 6, 8, 10, 12\}$ and both mean square error (MSE) and the mean absolute error (MAE) are reported on these datasets. For the LLaMa-70B, we leverage 4-bit quantization to perform different tasks on an A100 GPU. We also fine-tune the two datasets Springs and Charged on GPT-3.5, using the training data.

For the non-LLM approaches, we set the number

Dataset	Method	2-step		4-step		6-step		8-step		10-step		12-step	
		MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
ETH	LSTM	3.492	3.166	6.749	10.377	10.536	25.087	14.251	44.475	17.685	66.342	20.700	88.432
	GRU	4.420	4.355	8.940	16.403	11.832	28.167	13.920	38.837	15.712	49.625	17.429	61.298
	NODE	3.153	2.804	4.415	4.739	5.748	8.128	7.117	13.019	8.506	19.402	9.905	27.250
	NRI	2.744	2.861	5.297	10.687	7.694	22.234	9.929	36.404	11.992	52.566	13.459	78.476
	EGNN	**1.119**	**0.216**	**1.884**	**0.629**	2.782	**1.463**	3.815	**2.763**	4.576	**4.136**	**5.494**	**6.184**
	TimeLLM	2.491	6.759	2.741	7.173	3.607	10.199	4.156	12.609	4.724	15.502	5.637	21.402
	GPT-3.5	1.395	0.691	1.950	1.695	**2.752**	4.084	3.756	8.375	4.746	14.483	5.907	22.729
	Llama3-70B	1.175	0.495	2.037	1.606	2.794	2.602	**3.463**	3.441	**4.295**	4.881	5.523	7.892
Hotel	LSTM	5.208	6.145	10.386	22.768	14.419	42.689	17.343	59.509	19.582	73.294	21.455	85.512
	GRU	4.950	7.436	10.926	19.554	15.983	33.668	20.700	43.763	24.97	54.839	28.677	67.902
	NODE	3.192	3.117	4.458	4.972	5.798	8.281	7.178	13.083	8.581	19.361	9.999	27.088
	NRI	1.624	0.876	3.207	3.459	4.752	7.687	6.274	13.541	7.77	20.916	9.274	29.235
	EGNN	0.271	**0.027**	0.465	**0.086**	0.732	0.207	1.036	0.476	1.443	0.984	1.769	1.554
	TimeLLM	0.649	0.132	0.805	0.196	0.967	0.294	1.090	0.389	1.194	0.439	1.351	0.537
	GPT-3.5	0.318	0.041	0.47	0.089	**0.633**	**0.136**	**0.815**	**0.211**	**1.025**	**0.319**	1.253	**0.460**
	Llama3-70B	**0.195**	0.030	**0.431**	0.178	0.675	0.435	0.874	0.745	1.027	1.133	**1.180**	1.609
Univ	LSTM	6.201	8.698	13.788	36.511	19.633	68.742	24.18	99.047	27.898	126.937	31.021	152.504
	GRU	4.366	5.206	8.405	16.792	11.164	27.673	13.49	39.012	15.627	51.319	17.699	65.144
	NODE	2.881	2.817	4.206	4.812	5.581	8.373	6.981	13.541	8.395	20.308	9.816	28.644
	NRI	3.357	2.276	6.416	8.170	9.219	16.637	11.826	27.054	14.288	39.110	16.793	50.479
	EGNN	0.705	**0.103**	1.186	0.318	1.864	0.822	2.609	1.483	3.518	2.994	4.335	4.776
	TimeLLM	2.205	6.283	2.842	6.732	3.774	7.350	5.009	9.122	6.756	15.221	8.562	21.839
	GPT-3.5	0.278	0.022	0.664	**0.136**	1.014	**0.322**	1.404	**0.625**	1.832	**1.099**	2.696	5.526
	Llama3-70B	**0.014**	0.173	**0.101**	0.491	**0.290**	0.835	**0.575**	1.234	**1.039**	1.711	**1.787**	**2.24**
Zara1	LSTM	9.412	18.487	20.214	69.558	27.541	118.793	32.892	162.16	36.946	200.082	40.064	232.986
	GRU	4.736	4.97	8.918	16.829	11.742	28.674	13.964	40.275	15.853	51.785	17.533	63.178
	NODE	3.341	3.009	4.295	4.414	5.319	6.857	6.387	10.398	7.479	15.036	8.586	20.748
	NRI	2.759	1.756	5.396	6.859	7.929	15.112	10.373	26.368	12.736	40.51	15.023	59.286
	EGNN	1.976	0.791	1.951	0.687	2.974	1.710	3.297	2.509	3.944	3.529	5.472	6.834
	TimeLLM	4.123	10.866	5.435	12.665	7.334	18.187	9.47	25.554	11.784	34.877	14.22	46.54
	GPT-3.5	**0.230**	**0.021**	**0.590**	**0.122**	**0.949**	0.360	1.340	0.775	1.785	1.405	2.253	2.203
	Llama3-70B	0.239	0.022	0.627	0.129	0.954	**0.308**	**1.275**	**0.534**	**1.632**	**0.806**	**2.013**	**1.164**
Zara2	LSTM	4.363	4.212	8.521	13.763	13.024	31.864	17.096	54.053	20.468	75.553	23.272	95.317
	GRU	4.664	5.603	8.862	18.114	11.785	31.088	14.003	43.309	15.864	55.223	17.541	67.197
	NODE	3.427	3.222	4.687	5.099	6.02	8.188	7.387	12.552	8.775	18.193	10.175	25.098
	NRI	1.715	1.101	3.414	4.364	5.103	9.74	6.788	17.194	8.474	26.682	10.179	39.729
	EGNN	0.431	0.058	0.933	0.276	1.443	0.752	2.579	2.22	3.042	3.038	3.737	4.559
	TimeLLM	5.572	18.151	7.174	18.638	10.586	33.922	13.895	48.17	17.366	68.378	21.068	94.65
	GPT-3.5	0.533	0.089	1.131	0.424	1.695	0.895	2.221	1.487	2.702	2.198	**3.142**	3.02
	Llama3-70B	**0.386**	**0.042**	**0.882**	**0.179**	**1.412**	**0.467**	**2.021**	**0.974**	**2.683**	**1.742**	3.365	**2.74**

Table 2: The MSE and MAE ($\times 10^{-1}$) of compared methods on *ETH*, *Hotel*, *Univ*, *Zara1* and *Zara2*.

of samples for train/valid/test to a 1:1:1 ratio, train 50 epochs for every model, and set the learning rate to 5e-4. For TimeLLM, since it only supports input time-series data, we split the 2D data of multiple objects into multiple independent time-series data for prediction. For example, the 2D data of 5 objects in a sample will be converted into 10 independent time-series data.

4.2 Performance Comparison

We first report the results of LLMs and competitive non-LLM baselines on all nine datasets for the dynamic forecasting task. The results on *Springs* and *Charged* are reported in Table 1. The results on *ETH-UCY* and *Social* are reported in Table 2 and Table 3, respectively. From the results, we have the following observations.

Observation 1. LLMs have competitive capacities in dynamical forecasting. With one-shot prompting, Llama3-70B generally performs better than non-LLM approaches when prediction lengths are small (e.g., 2 and 4). In particular, the performance increasement of Llama3-70B on *Springs* (prediction length < 7) is 71.2% in terms of MAE

Dataset	2-step		4-step		6-step		8-step		10-step		12-step	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
LSTM	22.508	40.540	22.704	40.702	22.896	40.849	23.081	40.985	23.264	41.115	23.447	41.242
GRU	38.199	19.703	38.724	20.274	39.206	20.821	39.585	21.268	39.886	21.638	40.134	21.956
NODE	38.919	20.400	38.911	20.428	38.960	20.523	39.030	20.643	39.115	20.781	39.210	20.932
NRI	**0.925**	**0.014**	**1.813**	**0.056**	2.673	**0.122**	3.511	0.211	4.322	0.322	5.114	0.454
EGNN	2.704	0.121	2.887	0.140	3.052	0.158	3.200	**0.176**	3.337	**0.193**	3.463	**0.209**
TimeLLM	25.791	19.942	26.083	20.746	27.063	21.278	27.698	21.647	28.024	23.064	28.926	23.847
GPT-3.5	21.808	16.557	21.051	15.758	20.999	15.527	21.126	15.449	22.745	16.576	24.482	17.813
Llama3-70B	1.535	1.194	1.902	1.275	**2.220**	1.364	**2.451**	1.365	**2.702**	1.369	**3.033**	1.405

Table 3: The MSE and MAE ($\times 10^{-2}$) of compared methods on the *Social* dataset.

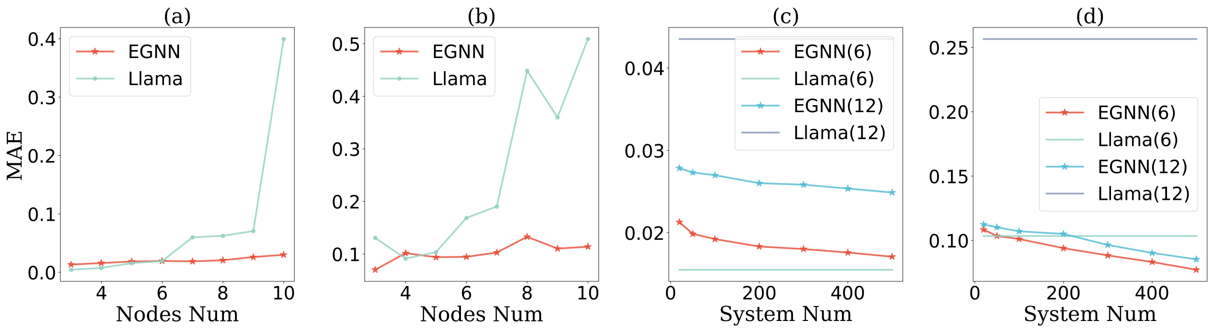


Figure 3: (a), (b) The performance of EGNN and Llama with respect to different numbers of nodes on *Springs* and *Charged*. (c), (d) The performance with respect to different training samples on *Springs* and *Charged* at different prediction lengths (6 and 12).

when compared with NRI, which demonstrates the potential ability of LLMs on dynamical system modeling. Note that non-LLM approaches are trained using a large number of samples, while LLMs achieve the performance with only one example to provide the format, which validates the high generalization ability of LLMs on dynamical system modeling. With supervised fine-tuning (SFT), LLMs can achieve even better performance on *Springs* and *Charged*.

Observation 2. Relation information is the key of LLMs to dynamical system modeling. Although TimeLLM has a competitive performance compared with state-of-the-art time-series forecasting approaches (Gruver et al., 2024), it performs much worse than Llama3-70B and GPT-3.5. In addition, TimeLLM performs much worse than non-LLM approaches. Note that TimeLLM only considers the trajectories of each agent individually, which can validate that relation information is crucial for LLMs to understand the system with accurate dynamic forecasting. We also conduct visualization of generated trajectories on *Charged*. The results are shown in Figure 2. From the results, we can observe that our LLM implementation can generate worse trajectories when removing edge information, which validates the importance of incorporating relation information.

Observation 3. The long-term forecasting performance of LLMs could be limited without any training in some cases. Both Llama3-70B and GPT-3.5 generally perform worse than EGNN when the prediction length is over 7 on *Charged*. The potential reason is that long-term forecasting through rollout could suffer from serious error accumulation. In contrast, EGNN does not need iterative rollout with extensive training signals, which can achieve less prediction error compared with training-free LLMs in some cases.

Observation 4. LLMs perform better when the number of objects is limited. Figure 3 (a) and (b) record the performance of Llama and EGNN with respect to different numbers of objects on *Springs* and *Charged*, respectively. From the results, we can find that when the number of objects is 4, Llama3-70B performs better than EGNN. However, when the number of objects is over 6, EGNN performs much better. The potential reason is that more objects bring in a more complicated graph structure, which is hard to understand for LLMs without any training.

Observation 5. LLMs suffer less performance

Dataset	Method	2-step		4-step		6-step		8-step		10-step		12-step	
		MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
Springs	LSTM	12.563	2.443	12.916	2.572	13.213	2.684	13.495	2.792	13.769	2.898	14.038	3.004
	GRU	9.100	1.278	9.760	1.456	10.331	1.621	10.807	1.763	11.218	1.888	11.586	2.003
	NODE	9.860	1.472	10.137	1.541	10.423	1.614	10.717	1.691	11.016	1.772	11.318	1.858
	NRI	1.550	0.062	2.941	**0.192**	4.260	**0.358**	5.557	**0.560**	6.858	**0.807**	8.176	**1.108**
	EGNN	12.646	2.493	12.936	2.638	13.204	2.811	13.474	2.970	13.798	3.188	15.004	3.852
	TimeLLM	24.811	3.855	24.989	3.867	25.072	3.878	25.558	3.981	27.690	4.393	29.635	4.737
	GPT-3.5	2.889	1.437	4.658	1.678	6.848	1.962	9.920	2.065	9.649	2.302	12.060	3.116
	Llama3-70B	**0.542**	**0.049**	**1.273**	0.365	**1.706**	0.887	**2.132**	1.566	**2.899**	1.812	**4.300**	1.842
Charged	LSTM	21.638	7.477	22.290	7.949	22.930	8.426	23.585	8.930	24.243	9.452	24.900	9.990
	GRU	20.235	6.594	21.041	7.164	21.803	7.718	22.533	8.263	23.241	8.804	23.939	9.350
	NODE	20.538	6.809	21.201	7.276	21.850	7.743	22.506	8.237	23.170	8.749	23.840	9.279
	NRI	4.976	1.859	8.543	2.617	13.600	4.029	18.829	7.540	24.559	12.481	30.319	18.811
	EGNN	10.637	2.002	10.888	**2.079**	**11.056**	**2.124**	**11.191**	**2.164**	**11.305**	**2.205**	**12.072**	**2.609**
	TimeLLM	22.703	9.828	25.269	9.939	33.710	11.429	41.271	12.145	55.798	13.870	60.913	14.671
	GPT-3.5	4.072	1.804	10.891	4.100	15.007	6.089	19.253	7.782	23.205	8.628	28.061	9.740
	Llama3-70B	**3.766**	**1.010**	**7.138**	3.420	11.131	3.891	15.555	6.057	20.429	8.170	25.943	9.085

Table 4: The MSE and MAE ($\times 10^{-2}$) of compared methods in the cross-domain settings.

Dataset	Method	Precision	Recall
Springs	Random	0.500	0.500
	GPT-3.5	0.579	0.800
	GPT-3.5 (SFT)	**0.973**	**0.914**
Charged	Random	0.500	0.500
	GPT-3.5	0.520	0.480
	GPT-3.5 (SFT)	**0.975**	**0.811**

Table 5: The relational reasoning performance of GPT-3.5 and random guessing on *Springs* and *Charged*. SFT denotes supervised fine-tuning.

degradation when it comes to zero-shot scenarios. Table 4 reports the zero-shot performance of different approaches. In particular, we first train the model in one domain and test the performance on a different domain. For LLMs, their one-shot example is from the different domains accordingly. From the results, we can observe that LLM methods always rank first or second among all the compared methods, which can validate the zero-shot capacity of LLMs. In addition, we compare the performance of Llama with EGNN with different numbers of systems during training. The results are shown in Figure 3 (c) and (d). From the results, we can find that when there is limited training data, LLMs can achieve better performance in some cases. With the increasing training data, data-driven non-LLM models would fit the data better.

Then, we study the performance of relational reasoning. In particular, the results of compared approaches on *Springs* and *Charged* in terms of precision and recall are shown in Table 5. From the results, we have the following observation.

Observation 6. LLMs have competitive capacities in relational reasoning with fine-tuning. In particular, with fine-tuning, GPT-3.5 achieves much better performance in comparison to random guessing, which validates that LLMs have the ability of inferring the interaction in dynamical systems. However, without fine-tuning, the performance increasement of GPT-3.5 in comparison to random guessing is limited, which validates that supervised fine-tuning can basically improve the performance of LLMs.

4.3 Ablation Study of Prompt Engineering

We further conduct ablation studies of prompt engineering in our implementation of LLMs. In particular, we introduce six model variants: (1) GPT w/o Context, which removes the context information related to the background; (2) GPT w/o Edge, which removes the edge information in our prompts; (3) GPT w/o Rollout, which skips the rollout process and directly outputs the target samples; (4) GPT w/o Example, which removes the example in the prompt; (5) GPT w/ One, which include one example in the prompts; (6) GPT w/ Two, which include two examples in the prompts. The compared results are recorded in Table 6. From the results, we have the following observation.

Observation 7. Prompt engineering is crucial for LLMs to tackle dynamical system modeling. Firstly, by comparing the performance between

Dataset	Method	2-step MAE	2-step MSE	4-step MAE	4-step MSE	6-step MAE	6-step MSE	8-step MAE	8-step MSE	10-step MAE	10-step MSE	12-step MAE	12-step MSE
Springs	No-Context	8.489	1.899	17.888	2.867	21.899	3.074	24.957	3.742	27.548	3.926	29.982	4.325
	No-Edge	3.821	1.645	5.685	1.727	7.936	2.047	10.268	2.279	12.651	2.62	15.051	3.042
	No-Rollout	3.698	1.432	5.298	1.664	8.925	2.156	11.772	2.489	15.673	3.215	21.623	3.822
	No-Example	14.037	2.529	22.981	3.525	25.156	3.748	33.644	4.679	38.725	4.894	50.519	5.726
	1-Example	**2.589**	**1.391**	**4.355**	**1.509**	**6.383**	1.692	7.925	1.889	9.293	2.074	**10.639**	**2.473**
	2-Example	3.275	1.449	4.513	1.513	6.414	**1.614**	**7.534**	**1.734**	**8.540**	**2.040**	10.822	2.822
Charged	No-Context	11.322	2.245	14.967	3.899	18.687	5.458	22.679	6.828	26.766	8.125	30.858	9.769
	No-Edge	15.136	4.027	18.566	5.328	22.523	6.417	27.255	8.367	32.858	9.875	38.349	10.872
	No-Rollout	6.278	1.367	12.346	3.526	17.782	4.670	23.929	7.017	30.527	9.214	36.251	10.021
	No-Example	23.677	6.728	28.866	8.672	33.543	9.884	37.75	10.27	42.009	11.387	46.384	12.672
	1-Example	**4.254**	**0.977**	10.411	3.117	15.010	5.860	19.044	7.454	23.226	8.042	27.442	9.638
	2-Example	5.434	1.105	**9.144**	**2.088**	**13.093**	**3.788**	**17.665**	**4.192**	**22.014**	**7.253**	**26.606**	**9.121**

Table 6: Ablation study on *Springs* and *Charged* (10^{-2}) with varying prediction length.

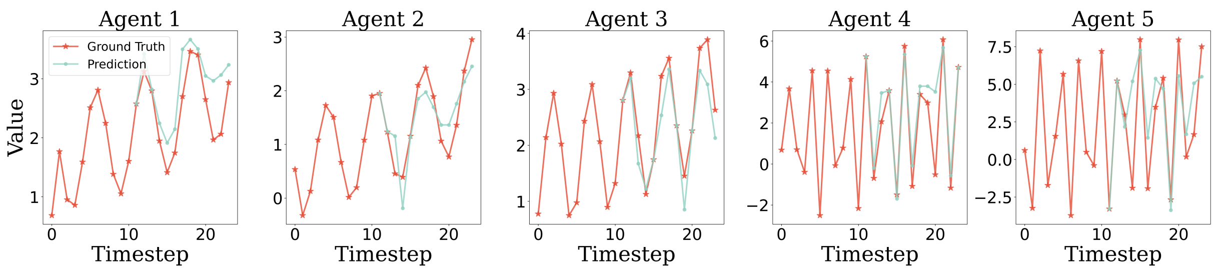


Figure 4: The comparison of predictions from GPT-3.5 and ground truth on periodicity-related dynamical systems

GPT w/ One and GPT w/o Context, we can find that context information is indispensable for the forecasting ability of LLMs. Secondly, GPT w/ One outperforms GPT w/o Edge consistently, which validates that relation information can be understood by LLMs to enhance dynamical system modeling. Thirdly, GPT w/o Rollout performs much worse than GPT w/ One. The potential reason is that it is challenging to generate long-term predictions directly without supervised training. Fourthly, without the one-shot prompting, GPT w/o Example cannot achieve reliable performance. However, one more example cannot bring the performance increasement as well from the comparison between GPT w/ One and GPT w/ Two.

4.4 Case Study

In the end, we provide a case study with periodicity-related dynamical systems. In particular, each agent has a basic underlying periodic function such as trigonometric functions, and their values are influenced by the neighbors from the predefined graph. Given the observation of different agents at the first 10 steps, we aim to predict the state at the next 10 steps. The predictions of GPT-3.5 and ground truth can be found in Figure 4. From the results, we can have the following observation.

Observation 8. LLMs have the ability to learn from periodicity-related dynamical systems. GPT-3.5 can successfully model periodicity-related dynamical systems. In particular, we can observe that LLMs can not only capture the periodic nature of the signal and approximate the general trend, even when the patterns are highly complicated for agents 4 and 5. In addition, GPT-3.5 identifies the peaks of agents accurately, which validates the strong ability of LLMs in dynamical system modeling.

5 Conclusion

In this paper, we show that LLMs can achieve comparable performance to non-LLM methods, which demonstrates the potential of LLMs in dynamical system modeling. Surprisingly, LLMs can effectively understand the relation to improve performance in dynamic forecasting tasks. Moreover, prompt engineering is crucial for LLMs to tackle dynamical system modeling tasks and the combination of context, edge information, and few-shot examples would benefit LLMs. We believe that our work provides an extensive benchmark to test more ways of blending LLMs and dynamical system modeling, e.g., LLM alignment in physics and multi-agent collaboration for dynamic forecasting.

6 Limitations

Despite the extensive progress, we should note that our benchmark does not involve complicated scientific scenarios with extensive domain knowledge such as molecular dynamical simulation scenarios. In future work, we will try to incorporate more external knowledge with domain-based prompting engineering to solve more realistic scientific problems. In addition, due to the limitations of computation resources, we do not utilize more LLMs such as Claude, which we leave in our future work.

Acknowledgment

This work was partially supported by NSF 2211557, NSF 1937599, NSF 2119643, NSF 2303037, NSF 2312501, NSF 2319449, NSF 2312502, NASA, SRC JUMP 2.0 Center, Amazon Research Awards, and Snapchat Gifts.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237, St. Julian's, Malta. Association for Computational Linguistics.

Suresh Bishnoi, Ravinder Bhattoo, Sayan Ranu, and NM Krishnan. 2022. Enhancing the inductive biases of graph neural ode for modeling dynamical systems. *arXiv preprint arXiv:2209.10740*.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *NeurIPS*.

Kyunghyun Cho et al. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. 2023. Do we really need complicated model architectures for temporal networks? *arXiv preprint arXiv:2302.11636*.

Ricardo C Cury, Istvan Megyeri, Tony Lindsey, Robson Macedo, Juan Batlle, Shwan Kim, Brian Baker, Robert Harris, and Reese H Clark. 2021. Natural language processing and machine learning for detection of respiratory illness by chest ct imaging and tracking of covid-19 pandemic in the united states. *Radiology: Cardiothoracic Imaging*, 3(1):e200596.

Yitong Deng, Hong-Xing Yu, Jiajun Wu, and Bo Zhu. 2023. Learning vortex dynamics for fluid inference and prediction. *arXiv preprint arXiv:2301.11494*.

Bin Feng, Zequn Liu, Nanlan Huang, Zhiping Xiao, Haomiao Zhang, Srbuhi Mirzoyan, Hanwen Xu, Jiaran Hao, Yinghui Xu, Ming Zhang, et al. 2024. A bioactivity foundation model using pairwise metalearning. *Nature Machine Intelligence*, 6(8):962–974.

Julia Gastinger, Shenyang Huang, Mikhail Galkin, Erfan Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, et al. 2024. Tgb 2.0: A benchmark for learning on temporal knowledge graphs and heterogeneous graphs. *arXiv preprint arXiv:2406.09639*.

Carlos Gómez-Rodríguez and Paul Williams. 2023. A confederacy of models: a comprehensive evaluation of LLMs on creative writing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14504–14528, Singapore. Association for Computational Linguistics.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. 2024. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36.

Yupeng Gu, Yizhou Sun, and Jianxi Gao. 2017. The co-evolution model for social network evolving and opinion migration. In *KDD*, pages 175–184.

Mingguo He, Zhewei Wei, and Ji-Rong Wen. 2022. Convolutional neural networks on graphs with chebyshev approximation, revisited. In *NeurIPS*.

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

Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. 2024. Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems*, 36.

Zijie Huang, Yizhou Sun, and Wei Wang. 2020. Learning continuous system dynamics from irregularly-sampled partial observations. In *NeurIPS*, pages 16177–16187.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. 2024. Time-llm: Time series forecasting by reprogramming large language models. *ICLR*.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606, Toronto, Canada. Association for Computational Linguistics.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. 2018. Neural relational inference for interacting systems. In *International conference on machine learning*, pages 2688–2697. PMLR.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Shiyong Lan, Yitong Ma, Weikang Huang, Wenwu Wang, Hongyu Yang, and Pyang Li. 2022. Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *ICML*, pages 11906–11917.

Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. 2007. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664. Wiley Online Library.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. 2022a. Finding global homophily in graph neural networks when meeting heterophily. In *ICML*, pages 13242–13256.

Zijie Li, Kazem Meidani, Prakarsh Yadav, and Amir Barati Farimani. 2022b. Graph neural networks accelerated molecular dynamics. *The Journal of Chemical Physics*, 156(14).

Xiao Liang, Di Wang, Haodi Zhong, Quan Wang, Ronghan Li, Rui Jia, and Bo Wan. 2024. Candidate-heuristic in-context learning: A new framework for enhancing medical visual question answering with llms. *Information Processing & Management*, 61(5):103805.

Tianyu Liu, Tinyi Chu, Xiao Luo, and Hongyu Zhao. 2024a. Baitsao: Building a foundation model for drug synergy analysis powered by language models. *bioRxiv*, pages 2024–04.

Toni JB Liu, Nicolas Boullé, Raphaël Sarfati, and Christopher J Earls. 2024b. Llms learn governing principles of dynamical systems, revealing an in-context neural scaling law. *arXiv preprint arXiv:2402.00795*.

Xiao Luo, Jingyang Yuan, Zijie Huang, Huiyu Jiang, Yifang Qin, Wei Ju, Ming Zhang, and Yizhou Sun. 2023. Hope: High-order graph ode for modeling interacting dynamics.

Huan Ma, Changqing Zhang, Yatao Bian, Lemao Liu, Zhirui Zhang, Peilin Zhao, Shu Zhang, Huazhu Fu, Qinghua Hu, and Bingzhe Wu. 2023. Fairness-guided few-shot prompting for large language models. *Advances in Neural Information Processing Systems*, 36:43136–43155.

Manuel Méndez, Mercedes G Merayo, and Manuel Núñez. 2023. Long-term traffic flow forecasting using a hybrid cnn-bilstm model. *Engineering Applications of Artificial Intelligence*, 121:106041.

Stephen Mutuvi, Antoine Doucet, Gaël Lejeune, and Moses Odeo. 2020. A dataset for multi-lingual epidemiological event extraction. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 4139–4144.

Minh Thuan Nguyen, Khanh Tung Tran, Nhu Van Nguyen, and Xuan-Son Vu. 2023. ViGPTQA - state-of-the-art LLMs for Vietnamese question answering: System overview, core models training, and evaluations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 754–764, Singapore. Association for Computational Linguistics.

Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. 2009. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th international conference on computer vision*, pages 261–268. IEEE.

Jiang-Zhou Peng, Siheng Chen, Nadine Aubry, Zhihua Chen, and Wei-Tao Wu. 2020. Unsteady reduced-order model of flow over cylinders based on convolutional and deconvolutional neural network structure. *Physics of Fluids*, 32(12):123609.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. 2021. Learning mesh-based simulation with graph networks. In *ICLR*.

Nazneen Fatema Rajani, Rui Zhang, Yi Chern Tan, Stephan Zheng, Jeremy Weiss, Aadit Vyas, Abhijit Gupta, Caiming Xiong, Richard Socher, and Dragomir Radev. 2020. Esprit: Explaining solutions to physical reasoning tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7906–7917.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. 2021. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR.

Pragya Srivastava, Manuj Malik, Vivek Gupta, Tanuja Ganu, and Dan Roth. 2024. Evaluating LLMs' mathematical reasoning in financial document question answering. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3853–3878, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Luning Sun, Xu Han, Han Gao, Jian-Xun Wang, and Liping Liu. 2023. Unifying predictions of deterministic and stochastic physics in mesh-reduced space with sequential flow generative model. In *NeurIPS*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tailin Wu, Takashi Maruyama, Qingqing Zhao, Gordon Wetzstein, and Jure Leskovec. 2023. Learning controllable adaptive simulation for multi-resolution physics. In *ICLR*.

Yunshu Wu, Hayate Iso, Pouya Pezeshkpour, Nikita Bhutani, and Estevam Hruschka. 2024. Less is more for long document summary evaluation by LLMs. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 330–343, St. Julian's, Malta. Association for Computational Linguistics.

Chenxin Xu, Robby T Tan, Yuhong Tan, Siheng Chen, Yu Guang Wang, Xinchao Wang, and Yanfeng Wang. 2023. Eqmotion: Equivariant multi-agent motion prediction with invariant interaction reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1410–1420.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *ICLR*.

Junwei Yang, Hanwen Xu, Srbuhi Mirzoyan, Tong Chen, Zixuan Liu, Zequn Liu, Wei Ju, Luchen Liu, Zhiping Xiao, Ming Zhang, et al. 2024. Poisoning medical knowledge using large language models. *Nature Machine Intelligence*, 6(10):1156–1168.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2023a. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36:67686–67700.

Xinli Yu, Zheng Chen, and Yanbin Lu. 2023b. Harnessing llms for temporal data-a study on explainable financial time series forecasting. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 739–753.

Youn-Yeol Yu, Jeongwhan Choi, Woojin Cho, Kookjin Lee, Nayong Kim, Kiseok Chang, ChangSeung Woo, Ilho Kim, SeokWoo Lee, Joon Young Yang, et al. 2024. Learning flexible body collision dynamics with hierarchical contact mesh transformer. In *ICLR*.

Jing Zhang, Hui Gao, Peng Zhang, Boda Feng, Wenmin Deng, and Yuexian Hou. 2024a. La-ucl: Llm-augmented unsupervised contrastive learning framework for few-shot text classification. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10198–10207.

Jingqing Zhang, Kai Sun, Akshay Jagadeesh, Parastoo Falakaflaki, Elena Kayayan, Guanyu Tao, Mahta Haghighat Ghahfarokhi, Deepa Gupta, Ashok Gupta, Vibhor Gupta, et al. 2024b. The potential and pitfalls of using a large language model such as chatgpt, gpt-4, or llama as a clinical assistant. *Journal of the American Medical Informatics Association*, 31(9):1884–1891.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. 2024c. Llm4dyg: can large language models solve spatial-temporal problems on dynamic graphs? In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4350–4361.

Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. 2022. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*.

A Details of Datasets

In this paper, we include nine datasets, and their details are shown as follows:

- *Springs* (Kipf et al., 2018) simulates the system containing a range of interconnected springs. The whole system is driven by Hooke's law, which indicates the situation of forces from springs. Each object has an initial position and its movement is influenced by its connected objects. Interaction strength can determine the force from its neighboring objects. Different systems could have different initial positions and interaction patterns.

- *Charged* (Kipf et al., 2018) is a popular dataset in electromagnetic. It studies the movement of charged particles within a box, which could attract or repel one other by Coulomb's law. There are equal probabilities for attraction and repulsion. This dataset can be applied to understand plasma physics and astrophysics.

- *ETH-UCY* (Lerner et al., 2007; Pellegrini et al., 2009) is a benchmark collected by ETH Zurich and the University of Cyprus, which contains five datasets. These datasets aim to study how people move in public spaces in different scenarios such as urban hotels and the University of Cyprus. The datasets have been popular in computer vision and autonomous driving.

- *Social* (Gu et al., 2017) is a dataset that models the opinion migration of people based on the social network. It simulates how people's opinions evolve along with time because of the interaction and influence of people nearby. Here, each node denotes a person and an edge is constructed if two people are related.

B Details of Compared Methods

In this paper, we compare nine approaches, which are introduced in detail as below:

- LSTM (Hochreiter and Schmidhuber, 1997) is a popular RNN for sequence prediction and modeling. It consists of three important gates, i.e., the forget gate, the input gate, and the output gate, which jointly contribute to capturing long-term dependency in the system.

- GRU (Cho et al., 2014) is a different version of RNN, which is comprised of two gates, i.e., the update gate and the reset gate. Through removing one gate, GRU is known to have better efficiency compared with LSTM.

- NODE (Chen et al., 2018) is a continuous neural network approach to model sequence data, which models the derivates of hidden states using a learnable function.

- NRI (Kipf et al., 2018) is a graph neural network method, which follows a variational encoder-decoder architecture. The encoder conducts the message passing mechanism to update node representations and the decoder is adopted to output the position change.

- EGNN (Satorras et al., 2021) is a graph neural network which considers the equivalence of the system on 3-dimensional space and contains a well-designed updating rule for the message passing procedure.

- LLMTime (Gruver et al., 2024) is a recent method for incorporating LLMs into time series forecasting. It directly encodes digits into tokens and considers the forecasting tasks as extrapolation.

- GPT-3.5 (Achiam et al., 2023) is a closed-source large language model released by OpenAI, which has shown effectiveness on a wide variety of tasks such as text generation and text summarization.

- Llama3-70B (Touvron et al., 2023) is an open-source large language model designed by Meta, which contains 70 billion parameters. From the previous works (Zhang et al., 2024b), Llama3-70B usually outperforms GPT-3.5 on most language-based tasks.

C More Visualization

In the section, we provide additional visualization results of compared methods. The compared performance on *Charged* and *ZARA1* are shown in Figure 5 and Figure 7, respectively. From the results, we can validate the competitive performance of LLMs on dynamical system modeling. Besides, we notice that LLMs perform poorly in some complicated cases as in Figure 6. The potential reason is that the system is highly complicated and thus LLMs cannot fully model it without any tuning.

D Examples of Prompts

We provide the full prompts in our benchmark. The prompts for dynamic forecasting on *Springs*, *Charged*, *ETC-ETH* and *Social* are shown in Figure 8, Figure 9, Figure 10, and Figure 11, respectively. The prompts for relational reasoning on *Springs* and *Charged* are shown in Figure 12 and Figure 13, respectively.

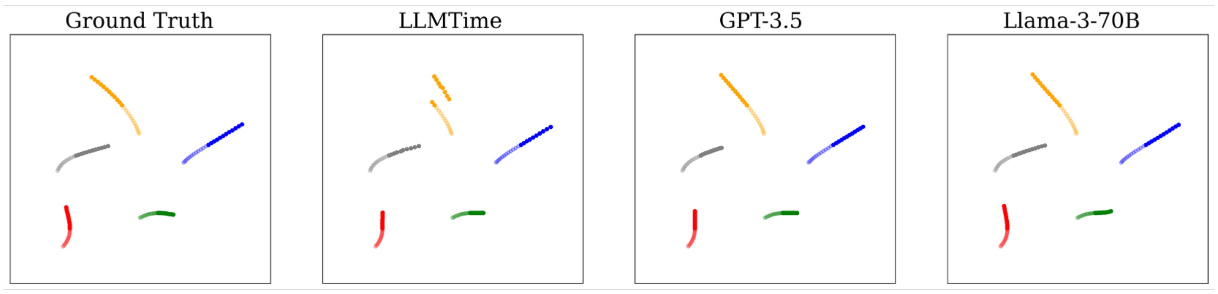


Figure 5: Visualization of the compared methods on *Charged*. Semi-transparent paths represent observed trajectories and solid paths indicate the predictions.

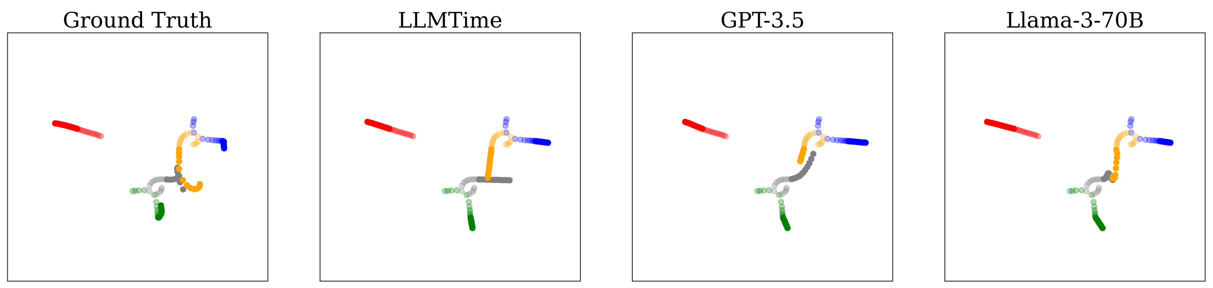


Figure 6: Visualization of a bad example on *Charged*.

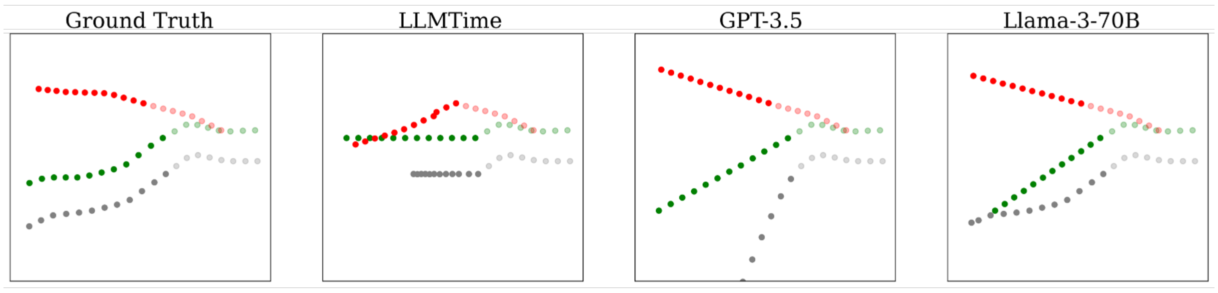
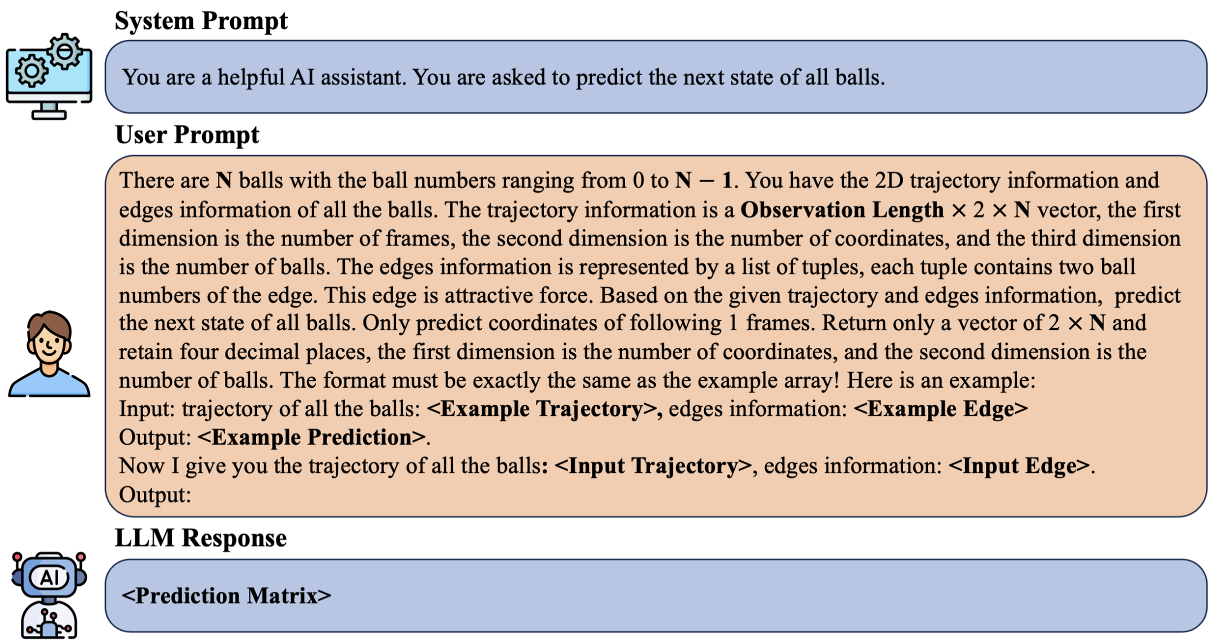


Figure 7: Visualization of the compared methods on *ZARA1*. Semi-transparent paths represent observed trajectories and solid paths indicate the predictions.



System Prompt

You are a helpful AI assistant. You are asked to predict the next state of all balls.

User Prompt

There are **N** balls with the ball numbers ranging from 0 to $\mathbf{N-1}$. You have the 2D trajectory information and edges information of all the balls. The trajectory information is a **Observation Length** × 2 × **N** vector, the first dimension is the number of frames, the second dimension is the number of coordinates, and the third dimension is the number of balls. The edges information is represented by a list of tuples, each tuple contains two ball numbers of the edge. This edge is attractive force. Based on the given trajectory and edges information, predict the next state of all balls. Only predict coordinates of following 1 frames. Return only a vector of 2 × **N** and retain four decimal places, the first dimension is the number of coordinates, and the second dimension is the number of balls. The format must be exactly the same as the example array! Here is an example:
Input: trajectory of all the balls: **<Example Trajectory>**, edges information: **<Example Edge>**
Output: **<Example Prediction>**.
Now I give you the trajectory of all the balls**: <Input Trajectory>**, edges information: **<Input Edge>**.
Output:

LLM Response

<Prediction Matrix>

Figure 8: An example of prompts for dynamic forecasting on *Springs*.

System Prompt

You are a helpful AI assistant. You are asked to predict the next state of all balls.

User Prompt

There are **N** balls with the ball numbers ranging from 0 to $\mathbf{N-1}$. You have the 2D trajectory information and edges information of all the balls. You have the 2D trajectory information and edges information of all the balls. The trajectory information is a **Observation Length** $\times 2 \times$ **N** vector, the first dimension is the number of frames, the second dimension is the number of coordinates, and the third dimension is the number of balls. The edges information is represented by a list of tuples, each tuple contains three elements: the first two elements are the ball numbers of the edge, and the third element is the edge type. There are two types of edges: 1 represents the repulsive force, and -1 represents the attractive force. Based on the given trajectory and edges information, predict the next state of all balls. Only predict coordinates of following 1 frames. Return only a vector of $2 \times$ **N** and retain four decimal places, the first dimension is the number of coordinates, and the second dimension is the number of balls. The format must be exactly the same as the example array! Here is an example:
Input: trajectory of all the balls: **<Example Trajectory>,** edges information: **<Example Edge>**
Output: **<Example Prediction>**.
Now I give you the trajectory of all the balls**: <Input Trajectory>**, edges information: **<Input Edge>**.
Output:

LLM Response

<Prediction Matrix>

Figure 9: An example of prompts for dynamic forecasting on *Charged*.

System Prompt

You are a helpful AI assistant. You are asked to predict the next state of pedestrians.

User Prompt

There are **N** pedestrian. You have the 2D trajectory information of the **N** pedestrians. The trajectory information is a **Observation Length** $\times 2 \times$ **N** vector, the first dimension is the number of frames, the second dimension is the number of coordinates, and the third dimension is the number of pedestrians. Based on the given trajectory information, predict the next state of the pedestrians. Only predict coordinates of following 1 frames. Return only a vector of $2 \times$ **N** and retain four decimal places, the first dimension is the number of coordinates. Don't give any text or return nothing! The format must be exactly the same as the example array! Here is an example:
Input: trajectory of all the pedestrians: : **<Example Trajectory>**.
Output: **<Example Prediction>**. Now I give you the trajectory of all the pedestrians: **<Input Trajectory>**.
Output:

LLM Response

<Prediction Matrix>

Figure 10: An example of prompts for dynamic forecasting on *ETC-ETH*.

System Prompt

You are a helpful AI assistant. You are asked to predict the next state of all people.

User Prompt

There are **N** person with the numbers ranging from 0 to $\mathbf{N-1}$ in a social network. You have the 2D trajectory information and popularity information of all the person. The trajectory information is a **Observation Length** $\times$ 2 $\times$ **N** vector, the first dimension is the number of frames, the second dimension is the number of coordinates, and the third dimension is the number of person. Popularity information represents the degree of popularity of a person in a social network. Based on the given trajectory and popularity information, predict the next state of all person. Only predict coordinates of following 1 frames. Return only a vector of 2 $\times$ **N** and retain four decimal places, the first dimension is the number of coordinates, and the second dimension is the number of person. The format must be exactly the same as the example array! Here is an example:
Input: trajectory of all the person: **<Example Trajectory>**, popularity information: **<Example Popularity>**
Output: **<Example Prediction>**.
Now I give you the trajectory of all the person: **<Input Trajectory>**, popularity information: **<Popularity>**.
Output:

LLM Response

<Prediction Matrix>

Figure 11: An example of prompts for dynamic forecasting on *Social*.

System Prompt

You are a helpful AI assistant for physical simulation.

User Prompt

You have 3D trajectory and velocity information of two balls. Based on the given trajectory, Infer the interaction relationship between the two balls.
Input: trajectory of ball 0: **<Trajectory 0>**, velocity of ball 0: **<Velocity 0>**. trajectory of ball 1: **<Trajectory 1>**, velocity of ball 1: **<Velocity 1>**. Are ball 0 and ball 1 interact?

LLM Response

True/False

Figure 12: An example of prompts for relational reasoning on *Springs*.

System Prompt

You are a helpful AI assistant for physical simulation.

User Prompt

You have 3D trajectory and velocity information of two charges. Based on the given trajectory, infer the interaction relationship between the two charges.
Input: trajectory of charge 0: **<Trajectory 0>**, velocity of charge 0: **<Velocity 0>**. trajectory of charge 1: **<Trajectory 1>**, velocity of charge 1: **<Velocity 1>**. Are charge 0 and charge 1 the same type?

LLM Response

True/False

Figure 13: An example of prompts for relational reasoning on *Charged*.