Forecasting Long-term Spatial-temporal Dynamics with Generative Transformer Networks

Donggeun Park, Hugon Lee, and Seunghwa Ryu*

[a] *Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea*
[b] ** Corresponding author: ryush@kaist.ac.kr*

Abstract

Recent advances in deep learning have aimed to address the limitations of traditional numerical simulations, which, although precise, are computationally intensive and often impractical for real-time applications. Current models, however, may have challenge in obtaining high predictive accuracy and long-term stability while obeying physical principles for spatiotemporal prediction problems. We introduce DynamicGPT, a Vision Transformer-based generative model specifically designed for spatiotemporal prediction. This model operates without explicit physical constraints, preserving critical spatial features and effectively capturing dependencies across varying time scales. The model integrates a multi-scale embedding network to preserve critical spatial features and a tailored temporal modeling network to effectively capture dependencies across varying time scales. This combination enables DynamicGPT to maintain predictive accuracy and stability over long-term forecasts, as validated by its performance in diverse real-world scenarios—including composite material stress and crack analysis, global sea surface temperature prediction, and 3D reaction-diffusion simulations—demonstrating its capability to handle out-of-distribution data, extended time horizons, and complex 3D structures. Importantly, DynamicGPT can adhere to physical laws, excels in partial differential equation parameter estimation, and optimizes its architecture for reduced computational load. This work positions DynamicGPT as a scalable, data-driven alternative bridging traditional simulations and modern AI, paving the way for advancement in real-time spatiotemporal modeling.

Introduction

Spatiotemporal physical processes are foundational to both natural and engineered systems, governing phenomena across diverse fields. Historically, mathematical modeling has served as a key approach to understanding and predicting these complex processes. Advances in computational power and numerical techniques have enabled solutions to sophisticated physical models that lack analytical solutions[1-3]. Such models provide invaluable insights into areas like atomic- and microscale material design[4,5], intricate fluid dynamics[6,7], and weather forecasting[8], offering an idealized framework for exploring physical behaviors. However, modeling realistic spatiotemporal dynamics through numerical simulations often incurs substantial computational costs, presenting significant challenge in advancing our understanding and discovery of fundamental physical principles.

Recent advancements in artificial intelligence (AI) have opened up new possibilities for overcoming these challenges across multiple scientific and industrial domains. AI-driven approaches enable rapid, real-time prediction of spatiotemporal dynamics under varying input conditions, finding success in fields like brittle crack propagation[9-11], fluid machinery optimization[12-15], and enhanced weather forecasting through nowcasting[16,17]. A common approach involves coupling Convolutional Neural Networks (CNN)-based autoencoders for spatial modeling with Recurrent Neural Networks (RNNs) for temporal dynamics[18]. However, while effective in many applications, these methods still have limitations in capturing long-term dependencies, restricting their predictive accuracy when extended sequences are involved.

To address these limitations, transformer architectures have gained attention for their ability to handle long-term dependencies more effectively than traditional RNNs. Unlike RNNs, which process data sequentially, transformers leverage self-attention mechanisms to process sequences in parallel, making them highly efficient in modeling long-term temporal

relationships. This paradigm shift has revolutionized various challenging tasks like natural language processing, enabling state-of-the-art performance among the existing spatiotemporal dynamic tasks[19,20]. However, applying transformers to spatiotemporal forecasting presents unique challenges, particularly when critical spatial information is lost during dimensionality reduction. Successfully adapting transformers to these tasks requires careful preservation and integration of spatial features in addition to temporal dependencies.

Recently, Physics-Informed Neural Networks (PINNs) have emerged as an innovative approach for embedding physical laws into neural network architectures[21-23]. By incorporating boundary conditions (BCs), initial conditions (ICs), and conservation laws, PINNs enforce physical consistency within AI-driven prediction frameworks. While PINNs offer valuable advantages, they have limitations. They require a solid understanding of the underlying physics, which can sometimes limit their broader applicability – particularly in cases where the governing physical laws are not yet fully understood[24]. Additionally, they struggle with scalability in high-dimensional systems and often encounter training instability in nonlinear and stochastic environments.

Given these challenges, we introduce a novel Vision Transformer (ViT)-based neural network framework designed to predict long-term spatiotemporal dynamics directly from data, without the need for explicit physical constraints or prior knowledge of the underlying system. This makes it particularly suitable for real-world challenges where such constraints may be unknown or impractical. Unlike traditional transformers, which often lose critical multi-scale spatial features by compressing high-dimensional spatiotemporal fields into 1D vector tokens, our approach preserves these features through a multi-scale kernel-based autoencoder[25,26] enhanced by a conditional Generative Adversarial Network (cGAN)[27] that embeds high-dimensional spatiotemporal data into low-dimensional latent patches. These latent patches are

then passed through a ViT[28] specifically designed to capture temporal interdependencies across short-, mid-, and long-term regimes. The transformed patches are decoded to reconstruct future spatiotemporal fields, enabling accurate predictions of complex dynamics. Therefore, we refer to this algorithm as ***DynamicGPT*** from this point forward, as it is specialized for **Dynamic** modeling through the integration of **G**enerative learning with a **P**re-trained multi-scale kernel-based autoencoder and vision **T**ransformer framework.

DynamicGPT effectively addresses the shortcomings of traditional methods by preserving and modeling both spatial and temporal features across scales, resulting in enhanced predictive accuracy. We validate its performance across a range of real-world applications, including composite material behavior, nonlinear air pollution dispersion, sea surface temperature forecasting, and three-dimensional reaction-diffusion systems. We validate DynamicGPT across challenging scenarios, including out-of-distribution spatiotemporal dynamics, extended temporal predictions, and complex three-dimensional spatial domains. Additionally, by applying DynamicGPT to real-world datasets, we highlight its robustness and versatility of the model in addressing practical prediction challenges. This research not only establishes the effectiveness of DynamicGPT but also provides insights in optimizing neural network architectures for diverse spatiotemporal dynamics, offering valuable guidance for future studies in this field.

Results

DynamicGPT: A Novel Approach to Spatiotemporal Dynamics Prediction

Predicting complex spatiotemporal dynamics is a critical challenge across various disciplines. Traditional models, which often rely on explicit physical constraints or transformer architectures that suffer from loss of spatial information during dimensionality reduction, frequently fail to capture the intricacies of these patterns. Many of these dynamics are governed by partial differential equations (PDEs) of the form $\frac{\partial \phi}{\partial t} = F(x, \nabla x\phi, \nabla x^2\phi, \phi \cdot \nabla x\phi, \ldots)$, which describe how a system evolves over time and space. These equations are typically solved numerically by discretizing both spatial and temporal domains into sequences: $\phi_{1:T} = \{\phi_1, \phi_2, \ldots, \phi_T\}$, where T is the total discretized time step and $\phi_i \in R^d$ represents the system's state at time t_i, discretized across d spatial points. The challenge lies in constructing a model that can accurately generate future solution sets based on past data, while preserving both multi-scale spatial features and complex temporal dynamics.

To address this, DynamicGPT, a ViT-based neural network specifically designed for long-term spatiotemporal prediction, utilizes framework structured into three main stages: **embedding**, **temporal modeling**, and **prediction**. It directly operates on data, avoiding the need for explicit physical constraints or prior knowledge of the system's underlying physics.

First, in the **embedding** stage (Green box in **Figure 1a**), the **Multi-Spatial Embedding Network (E)** takes high-dimensional spatiotemporal solution sets $\phi_{1:T}$ as input and transforms them into low-dimensional latent patches $\xi_{1:T}$ (Eq. 1). This network captures both local and global spatial features by employing multi-scale kernels. Small-scale kernels detect localized phenomena (e.g., pollution hotspots), while large-scale kernels identify broader, long-range patterns (e.g., cross-border pollutant transport). By integrating these multi-scale spatial features into a unified feature map, the embedding network generates latent patches $\xi_{1:T}$ that retain essential spatial information while significantly reducing the complexity of the original solution sets (Left dotted box in **Figure 1a**). Unlike traditional transformers, which

reduce spatial information into 1D vectors, often leading to the loss of spatial information, our model preserves a 2D latent space representation, better preserving complex spatial patterns. Next, the resulting latent patches are then processed with a variational sampling technique, modeling the inherent uncertainty present in real-world spatiotemporal dynamics. Additionally, cGAN is incorporated to further strengthens the model's ability to generate future latent patches under the distributional conditions of the past data. cGAN has been widely recognized in many studies for its ability to enhance the prediction of spatiotemporal dynamics by modeling their probabilistic characteristics, such as engineering and real-world problem[29-36].

After the latent patches $\xi_{1:T}$ are generated, **Multi-Scale Temporal Network (*D*)** handles the **temporal modeling** stage (Gray box in **Figure 1a**). This network processes the latent patches to predict the patch closest to the target future time step (Eq. 2). It leverages variable time-length sequences of short-term patches ξ_{T+L}, mid-term patches $\xi_{\lfloor \frac{T+L}{2} \rfloor : T+L}$, and long-term patches $\xi_{1:T+L}$, allowing it to capture temporal dependencies over different time scales. Padding is applied to align these patches in length. Temporal position embeddings are applied to ensure the sequence of the events is maintained, enabling the model to learn temporal patterns. These embedded patches pass through a self-attention mechanism, capturing interactions across different time scales—from short-term fluctuations to long-term trends. Unlike RNNs that process sequences step-by-step and struggle with capturing long-term dependencies, the ViT processes variable time-length inputs in parallel by padding and efficiently learns correlations between the prediction target and temporal patterns of varying time-lengths. To further enhance temporal modeling, a ConvGRU[37] layer refines the interactions between predicted patches, improving the model's capacity to capture complex temporal dependencies and leading to more accurate predictions.

In the **prediction** stage, **Multi-Spatial Decoder (*F*)** reconstructs the high-dimensional

solution sets from the predicted latent patches, enabling accurate forecasting of future spatiotemporal dynamics (Blue box in **Figure 1a**). The decoder takes the predicted latent patch ξ_{T+L+1} as an input and reconstructs the corresponding high-dimensional solution set ϕ_{T+L+1} (Eq. 3). The predicted solution is then used in an autoregressive fashion, where the output of one prediction step is used as input for the next. This autoregressive process allows the network to recursively forecast multiple future time steps. By capturing both spatial and temporal dynamics effectively, this inference structure ensures robust and reliable predictions across various spatiotemporal tasks. For more detailed information on data processing, the optimized architecture and training strategies, readers may refer to the **Methods** and **Supplementary Information B** section. The formalized workflow is summarized as follows:

$$\xi_{1:T} = \boldsymbol{E}(\phi_1, \phi_2, \ldots, \phi_T), \tag{1}$$

$$\xi_{T+L+1} = \boldsymbol{D}(\xi_{1:T+L}, \xi_{(T+L)/2:T+L}, \xi_{T+L}), \tag{2}$$

$$\phi_{T+L+1} = \boldsymbol{F}(\xi_{T+L+1}). \tag{3}$$

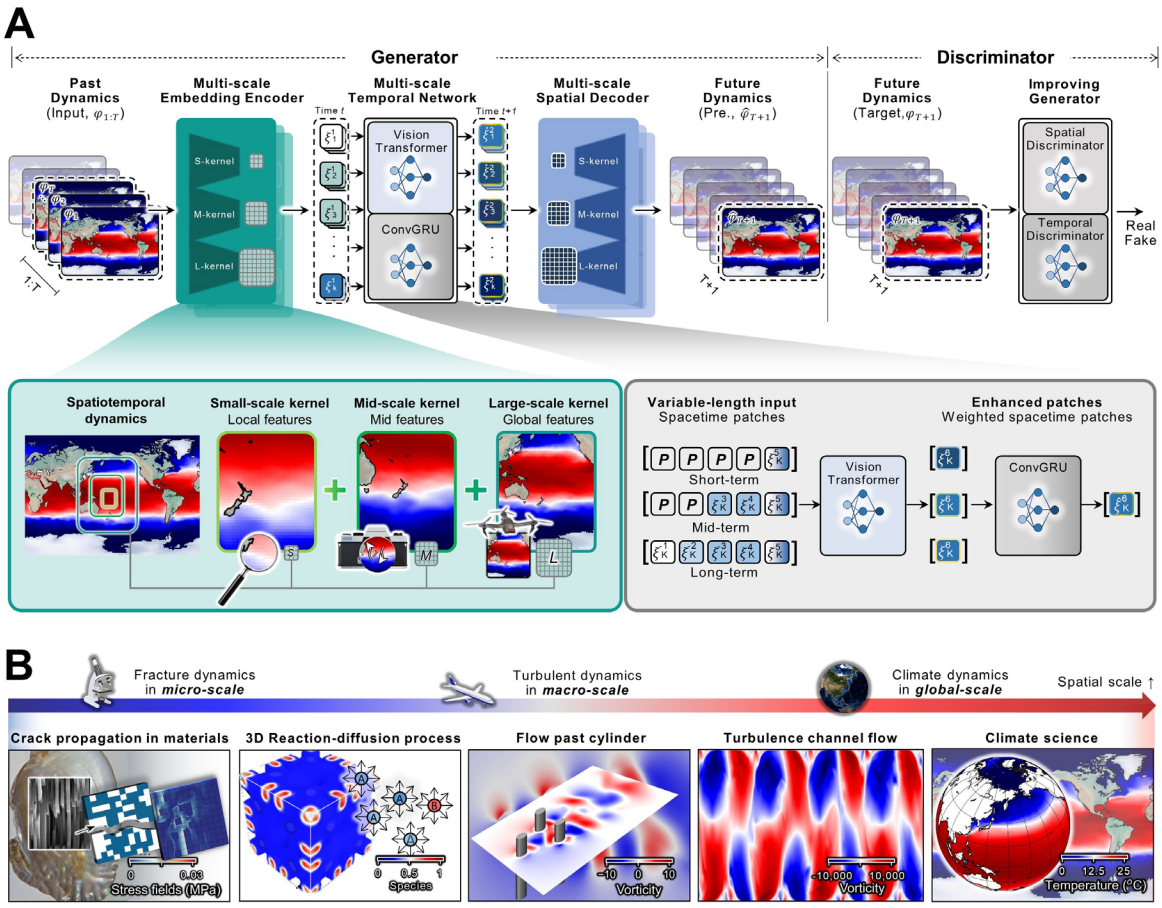


Figure 1. Overview of DynamicGPT's Multi-Scale Spatiotemporal Network and Validation Scenarios. (**a**) Multi-Scale Spatial Network: Combines small- and large-scale kernels to capture both detailed local and broad global spatial features, critical for accurate predictions across varying spatial contexts, such as in air pollution modeling. (**b**) Model Architecture: a multi-scale spatial encoder, temporal network, and spatial decoder, integrated with cGAN discriminators. The temporal network processes variable time-length spacetime patches (short-, mid-, and long-term) using ViT and ConvGRU layers for robust predictions. (**c**) Multi-Scale Temporal Network: Integrates short-, mid-, and long-term dependencies, enabling the model to capture both immediate and gradual changes crucial for long-term forecasting. *P* represents padding for variable time-length inputs in parallel (**d**) Validation Scenarios: The model is validated across four scenarios, spanning different scales and complexities—micro-scale crack propagation, macro-scale fluid flow, and global-scale air pollution and climate dynamics—highlighting its versatility across diverse spatiotemporal tasks.

Real-World Problems for Validating DynamicGPT

DynamicGPT is designed to tackle the inherent complexities of spatiotemporal dynamics with the goal of providing robust predictions across diverse real-world applications. For effective deployment, it is crucial that the model generalizes beyond its training data. Real-world problems often present unpredictable conditions, requiring consistent performance across a variety of circumstances.

To rigorously evaluate DynamicGPT's generalization capabilities, we assess its performance across five key scenarios that represent critical real-world challenges: **1. Out-of-Distribution Behavior**: Managing unexpected scenarios, such as fracture characteristics in engineering materials that differ significantly from the training data. **2. Future Sequences**: Forecasting future states based solely on past training data, such as predicting future weather conditions. **3. Larger/Different Domain Adaptation**: Adapting to new or expanded spatial domains, such as turbulence in different resolutions or larger geographical areas. **4. Three-Dimensional Data Handling**: Effectively processing complex 3D data, vital for applications like anisotropic reaction-diffusion systems. **5. Experimental Data Applicability**: Ensuring reliable performance on real experimental data beyond idealized simulations.

Addressing these cases is essential for any deep learning model aiming to replace traditional simulations and deliver reliable real-time predictions in spatiotemporal dynamics. Therefore, to validate DynamicGPT’s robustness, we curated a series of real-world validation datasets that reflect these challenges. The datasets consist of both numerical simulations and experimental measurements, covering a broad range of temporal and spatial scales (**Figure 1b**). These datasets are not just theoretical benchmarks but practical representations of critical issues impacting everyday life, various industries, and natural phenomena. Each scenario tests a unique aspect of DynamicGPT's capabilities, ensuring adaptability across a diverse range of

conditions. The validation scenarios include:

- **Scenario I: Catastrophic Fracture in Structural Materials** – Predicting material failures is critical to ensuring the safety of buildings and transportation systems. In this scenario, DynamicGPT is tested on its ability to predict stress distribution and crack propagation in composite materials[38] commonly used in various industries. The scenario also involves **Out-of-Distribution Behavior**, where fracture characteristics not seen during training are introduced. This evaluation is vital for assessing whether deep learning can replace traditional simulations in material design.

- **Scenario II: Unsteady Flow in Fluid Machinery** - Accurate predictions on complex fluid flows are vital for safety and efficiency in engineering systems[39]. Here, DynamicGPT is assessed on its ability to adapt to unseen **Future Sequences** and **larger or different spatial domains** by simulating the flow dynamics perturbed by three bodies and Rayleigh-Bénard convection flow. This scenario assesses the model's capability to manage complex fluid behaviors in varied conditions.

- **Scenario III: Global-scale Sea Surface Temperature Variations** - Monitoring and predicting sea surface temperature (SST) variations are crucial for understanding global climate trends[40]. In this scenario, DynamicGPT is evaluated on its ability to predict **Future Sequences** of SST data, including **Out-of-Distribution** values from **real-world Experimental Data** collected via ship-mounted sensors. This assessment tests the model's applicability in handling real-world data and predicting large-scale environmental changes.

- **Scenario IV: Three-Dimensional Chemical Species Reaction and Diffusion** – Managing chemical processes in 3D spaces is essential for fields such as environmental modeling and reaction-diffusion systems[41]. This scenario examines

DynamicGPT's ability to capture intricate spatiotemporal relationships within a 3D context, showcasing the model's versatility in handling **complex 3D data** across various applications.

Through these diverse and challenging scenarios, we aim to demonstrate DynamicGPT's effectiveness in solving real-world problems critical to both industry and human well-being. This validation process underscores DynamicGPT's potential to replace traditional simulations, providing reliable and scalable predictions. For more details on simulation setup and dataset generation, please refer to the **Methods** section.

Baseline Model Selection and Rationale

To comprehensively evaluate DynamicGPT's performance, we selected three widely used baseline models in spatiotemporal dynamic prediction tasks: PINN[42], Transformer[43], and Deep Generative Model (DGM)[44]. These models were chosen for their distinct approaches to spatiotemporal modeling and their relevance to the types of dynamics addressed by DynamicGPT. These models were chosen for their distinct approaches to spatiotemporal modeling and their relevance to the types of dynamics addressed by DynamicGPT.

- **PINNs**[42] integrate physical governing equations directly into the neural network's loss function, enabling the model to enforce known physical laws during training. This allows PINNs to model spatiotemporal phenomena based on partial differential equations (PDEs) that describe system dynamics. While PINNs provide high physical consistency and interpretability, their reliance on accurate physical descriptions and the computational cost associated with solving PDEs limits their scalability, especially in real-time predictions[24].
- **Transformers**[43], with their self-attention mechanisms, excel at modeling long-range temporal dependencies within sequential data. In spatiotemporal dynamic predictions, Transformers are often used with autoencoder architectures that reduce

high-dimensional spatiotemporal data into lower-dimensional latent vectors. While effective at capturing temporal patterns, Transformers often suffer from spatial information loss when dealing with high-dimensional data due to their focus on one-dimensional representations and single-scale processing[28].

- **DGMs**[44], particularly those using cGANs, are well-known for their ability to model complex distributions and generate realistic samples. In spatiotemporal prediction tasks, DGMs use encoder-ConvLSTM-decoder architectures to capture temporal dependencies. However, the sequential nature of this structure can lead to error propagation and instability in predictions, particularly for large or high-dimensional datasets. Additionally, the high computational cost associated with training DGMs makes them less suitable for real-time applications.

Since illustrating the prediction results of all baseline models across spatiotemporal dynamics (i.e., changes over time and local spatial positions) would require extensive figure sets that occupy significant space and increase visual complexity, we focus on comparing DynamicGPT's outcomes with the best-performing baseline model—DGM (based on RMSE performance, see **Table 1**). This approach ensures that the visual presentation remains clear and concise, highlighting the most relevant comparisons. For a more comprehensive evaluation, all baseline models, including PINNs and Transformers, are analyzed using physical evaluation criteria that form the basis of the simulation. Specifically, we compare: (i) *Conservation law evaluation*, (ii) *Estimation of unknown PDE coefficients*. These evaluations are critical for assessing the ability of each model to not only predict spatiotemporal dynamics accurately but also adhere to the underlying physical principles governing the system. By focusing on these criteria, we ensure that all baseline models are evaluated on their capacity to maintain physical consistency and provide meaningful insights into the system's dynamics under various scenarios. These evaluations will be discussed in detail in the subsequent sections.

Method	Scenario 1 Unit: (MPa)	Scenario 2a Unit: (m/s)	Scenario 2b Unit: (m/s)	Scenario 3 Unit: $(°C)$	Scenario 4 Unit: (ppm)
Field variables	σ_{UTS}	**u**	**u**	T	c
DynamicGPT	$\mathbf{1.81 \times 10^{-4}}$	$\mathbf{1.183 \times 10^{-1}}$	$\mathbf{1.24 \times 10^{3}}$	$\mathbf{3.92 \times 10^{-1}}$	$\mathbf{3.62 \times 10^{-2}}$
Deep generative model	2.31×10^{-4}	1.702×10^{-1}	1.65×10^{3}	5.21×10^{-1}	1.65×10^{-1}
Transformer	2.68×10^{-4}	1.842×10^{-1}	2.07×10^{3}	9.21×10^{-1}	2.21×10^{-1}
Physics-informed neural network	5.51×10^{-4}	2.211×10^{-1}	1.91×10^{3}	N/A	4.62×10^{-1}

Table 1. Root mean squared error for each test scenario. (1) predicting catastrophic failure of composite materials, (2a,b) modeling complex unsteady hydrodynamics, (3) accurate long-term prediction of global sea surface temperature change, and (4) simulating scaling-reaction-diffusion systems in 3D. The best performing algorithms in each scenario are highlighted in bold. PINN is not used as a baseline model in Scenario 3 because the result is a realistic estimation dataset, with temperature fields only that cannot satisfy the governing equations.

Scenario I: Predicting Catastrophic Fracture in Composite Materials

In this scenario, we evaluated DynamicGPT's ability to predict the brittle failure behavior of composite materials under tensile loading, particularly for configurations that fall outside the distribution of the training data according to the strength criteria. Accurately predicting both strength and fracture behavior is crucial in material design, as these properties determine a material's ability to withstand external loads before failure[38]. **Figure 2a** represents the mechanical responses of various composite configurations. DynamicGPT was trained on 4,000 different configurations (represented by thin white lines) and tested on 1,000 configurations with higher strength values (represented by thin blue lines), which have ultimate tensile strengths significantly higher than the training data, indicating out-of-distribution (OOD) cases. The thick gray line represents the test configuration with the highest strength (0.052 MPa), more than twice the average strength of the training set (around 0.02 MPa).

When evaluated based on toughness (area below the stress-strain curve), the baseline model (DGM) achieved an R^2 of 0.91, while DynamicGPT achieved a much higher R^2 of 0.97,

showing a substantial improvement in performance. Unlike the baseline model, which struggles to maintain accuracy beyond 0.1% strain due to the accumulation of error, DynamicGPT (blue line) closely follows the finite element method (FEM) curve for a full strain range (**Figure 2a**).

Figures 2b and **2c** further examine the model's performance on the highest strength configuration (Top 1). Using the initial three stress field steps and the material's geometric layout as inputs, the model predicts both stress and crack fields at different strain levels. **Figure 2b** shows the stress distribution at different strain levels (0.025%, 0.05%, and 0.2%). At all strain levels, DynamicGPT closely matches the FEM results, accurately capturing localized stress concentrations associated with crack initiation and propagation—critical indicators of material failure. In contrast, the baseline model accumulates significant errors at higher strain levels, particularly when the material is on the verge of catastrophic failure. Accurately predicting the stress field at this critical stage is vital for evaluating material safety in real-world applications, further underscoring DynamicGPT's potential as a reliable tool for composite material design.

In addition, we compare the crack phase fields predicted by FEM, DynamicGPT, and the baseline model (**Figure 2c**). Similar to the stress distribution results, DynamicGPT closely aligns with FEM, accurately predicting crack initiation and growth. The baseline model, however, struggles to predict crack behavior at higher strains, leading to significant deviations and errors. This inability to model crack behavior accurately is a critical limitation in material design, where predicting fracture patterns is as important as forecasting overall mechanical performance.

These results collectively highlight DynamicGPT's superior ability to model both stress and crack propagation with high precision, even in configurations beyond the training

data. Its robustness in handling complex spatiotemporal dynamics positions it as a powerful alternative to traditional simulation methods, particularly in high-performance material design and structural safety analysis.

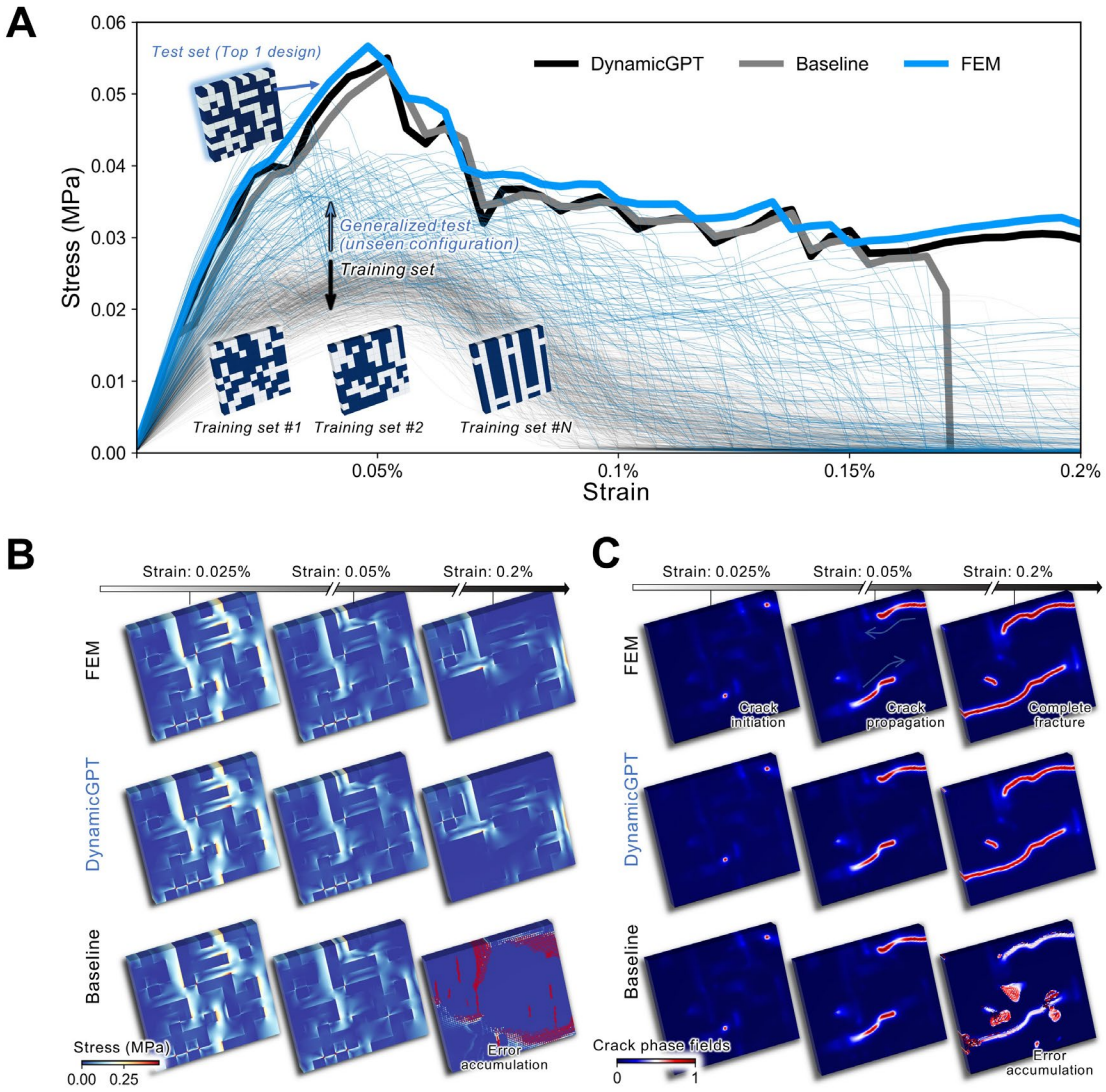


Figure 2. Performance of DynamicGPT in Predicting Composite Material Behavior Under Tensile Loading. (**a**) Stress-strain curves comparing DynamicGPT (black), baseline (white), and FEM (blue) across different composite configurations. Training data includes 4,000 configurations (thin white lines), and test data includes 1,000 stronger configurations (thin blue lines). The thick gray line represents the highest-strength configuration in the test set. DynamicGPT closely aligns with FEM, outperforming the baseline in OOD scenarios. (**b**) Stress distribution maps at different strain levels (0.025%, 0.05%, and 0.2%) for FEM, DynamicGPT, and the baseline model. (**c**) Crack phase fields at the various strain levels.

Scenario II: Modeling Complex Unsteady Fluid Dynamics for Engineering Safety and Efficiency

The validation in this scenario centers on DynamicGPT's ability to predict unsteady flow patterns, particularly when confronted with unseen geometries and boundary conditions. Accurate flow prediction is crucial for real-world applications such as fluid machinery, building design, and turbine engineering, where even minor inaccuracies can lead to inefficiencies or catastrophic failures[39].

Figure 3a illustrates the setup for both training and testing, where fluid flows past different bodies with varying geometries. The challenge here lies in predicting how these flow patterns evolve as the fluid encounters new obstacles—a key task in optimizing engineering systems like airflow control. DynamicGPT's ability to generalize and maintain accuracy in these unseen configurations demonstrates its potential as a robust predictive tool for such applications. Moreover, **Figure 3b** shifts the focus to turbulent channel flow, where the training dataset is collected from smaller, less turbulent domains. The test set, however, includes larger domains with intense turbulence, similar to Rayleigh-Bénard convection, where fluid is heated and cooled on bottom and top surfaces, respectively, forming intricate convection cells. Capturing such complex structures is essential for applications ranging from climate modeling to industrial design, as it directly impacts system performance and safety.

We carry out a comparative analysis between CFD simulations, DynamicGPT predictions, and a baseline model for the two scenarios—unsteady flow past bodies (**Figure 3c**) and turbulent channel flow (**Figure 3d**). DynamicGPT's predictions closely align with CFD results, accurately capturing key flow structures and velocity distributions over time in both unsteady flow and Rayleigh-Bénard convection problems. In contrast, the baseline model begins to accumulate errors, particularly after around 60-time steps, deviating significantly

from the CFD results at the long-term time snapshot (100-time steps). This issue, known as error accumulation, is a critical challenge in long-term spatiotemporal dynamics modeling by "sub-optimal neural network architecture". The baseline model's predictions diverge significantly over time due to its limitations in preserving spatial coherence and handling complex interactions, especially in unseen spatial domains. DynamicGPT's superior performance, with an R^2 of 0.974 compared to 0.86 for the baseline, underscores its robustness in maintaining accuracy across various conditions.

Furthermore, we analyze the energy spectrum across different wavenumbers, validating how well the models capture turbulence dynamics for the Rayleigh-Bénard convection problem (**Figure 3e)**. The energy spectrum is crucial for understanding how energy cascades from large-scale eddies (low wavenumbers) to small-scale eddies (high wavenumbers), where it eventually dissipates as heat and flow. In a well-resolved turbulent flow, the inertial subrange—characterized by the -5/3 slope—indicates a steady transfer of energy between scales, maintaining a balance between energy generation and dissipation. DynamicGPT closely follows the Kolmogorov -5/3 law, reflecting its ability to effectively model multi-scale turbulence—a key requirement for accurate real-world predictions. This alignment suggests that DynamicGPT successfully captures both the energy generation at larger scales and its dissipation at smaller scales, accurately modeling the full turbulence cascade. The baseline model, however, deviates from this expected behavior, struggling to represent the energy distribution across scales, indicating its limitations in capturing realistic turbulence dynamics. These results demonstrate DynamicGPT's robustness in generalizing complex multi-scale flow dynamics across both seen and unseen scenarios. This capability positions DynamicGPT as a powerful tool for predicting fluid behavior in challenging engineering applications, offering a significant advancement over conventional methods.

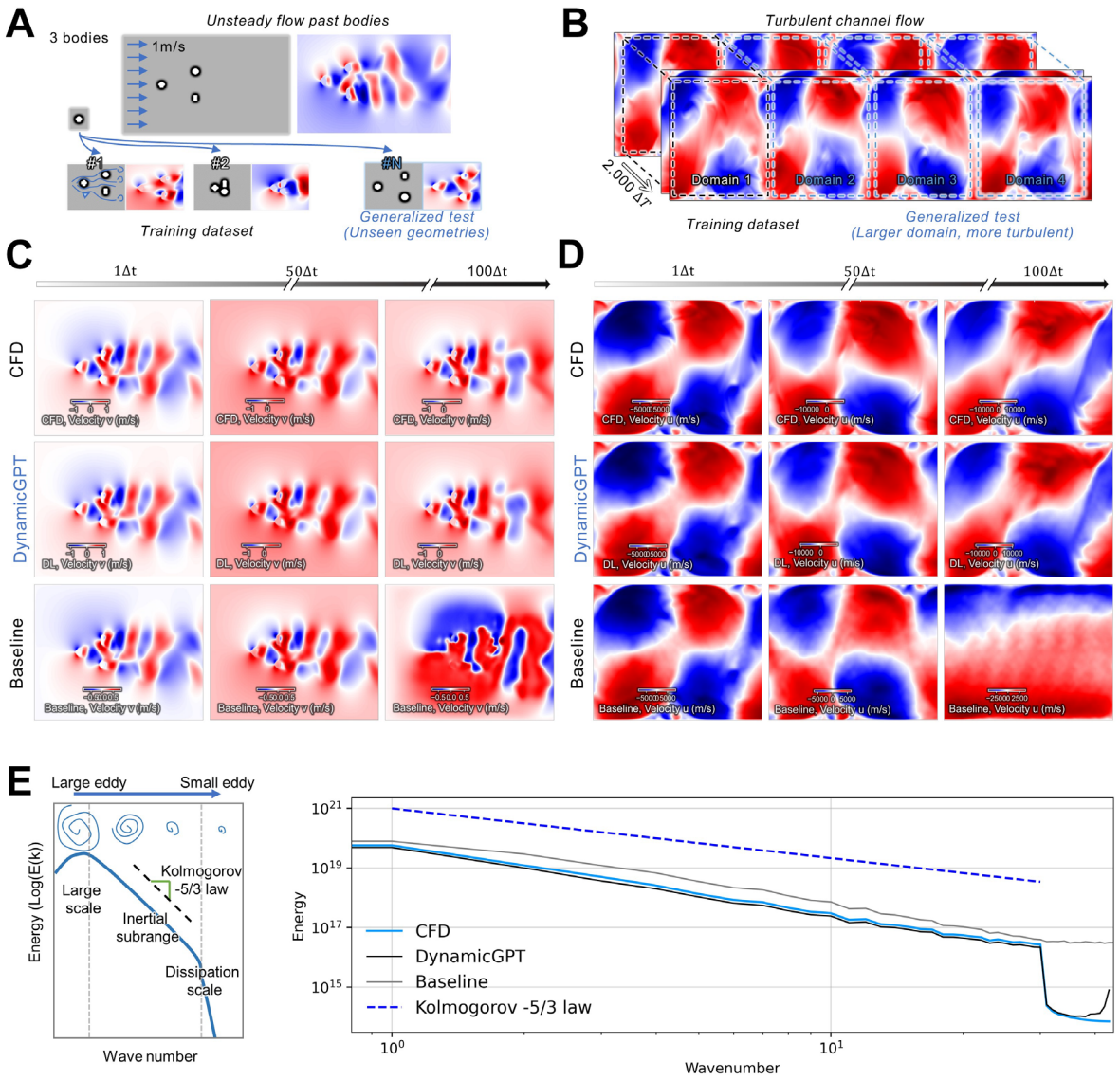


Figure 3. DynamicGPT's Performance in Predicting Unsteady Flow and Turbulent Channel Flow. (**a**) Training and testing setup for unsteady flow past three bodies with varying geometries. The generalized test cases involve unseen configurations to assess the model's generalization ability. (**b**) Turbulent channel flow scenario highlighting the transition from smaller, less turbulent domains in training to larger, more chaotic domains in testing, similar to Rayleigh-Bénard convection. (**c**) Velocity field comparison at different time steps ($1\Delta t$, $50\Delta t$, and $100\Delta t$) for unsteady flow past bodies. (**d**) Velocity field comparison for turbulent channel flow. (**e**) Energy spectrum analysis comparing CFD, DynamicGPT, and the baseline model across different wavenumbers.

Scenario III: Accurate Long-Term Predictions of Global Sea Surface Temperature Variations

This scenario evaluates DynamicGPT's capability to predict global-scale SST variations, a crucial component for understanding and forecasting climate dynamics, including phenomena like El Niño and La Niña. Accurate SST predictions are essential for climate modeling, disaster preparedness, and policy planning[40,45].

Figure 4a illustrates historical SST data trends from 1940 to 2023, with training data spanning from 1990 to 2010 and test data covering 2010 onwards. The data were collected using temperature sensors installed on ships, emphasizing the real-world measurement context and the model's ability to generalize from experimental data. This setup is crucial for assessing the model's ability to predict future sequences—a core requirement for effective long-term climate forecasting. The SST patterns reveal both seasonal cycles and long-term trends, providing a foundation for evaluating the model's ability to capture these dynamics when extrapolating beyond the training period.

Figure 4b compares SST anomaly predictions across different months between observed measurements, DynamicGPT, and a baseline model. While DynamicGPT consistently aligns more closely with observed data, particularly in capturing spatial temperature distributions over large geographic regions, the baseline model also performs relatively well due to the repetitive, linear growth pattern in both the training and test sets. These consistent cycles help the baseline model achieve decent accuracy in this specific scenario, a point worth noting given that such conditions are not always present in real-world applications.

However, when focusing on SST anomalies in the Niño3.4 region—a critical indicator for monitoring El Niño and La Niña events (**Figure 4c**)—DynamicGPT clearly outperforms

the baseline. Accurate predictions in the Niño3.4 region is essential for climate monitoring and response, as this region significantly influences global weather patterns and climate phenomena. DynamicGPT tracks the observed anomaly trends closely, capturing both the timing and magnitude of these oscillations. In contrast, the baseline model deviates significantly, misrepresenting key peaks and troughs, which could lead to inaccurate forecasts and suboptimal climate response strategies. DynamicGPT's ability to follow the observed trends more accurately highlights its strength in generalizing complex climate patterns beyond the training data. Overall, these results demonstrate that DynamicGPT is a robust and reliable tool for large-scale climate prediction, offering significant improvements over conventional models in both short-term and long-term forecasts. Its superior performance in predicting SST anomalies and capturing global climate dynamics underscores its potential for applicability in real-world climate monitoring and forecasting systems, enabling more informed and effective decision-making in environmental and policy contexts.

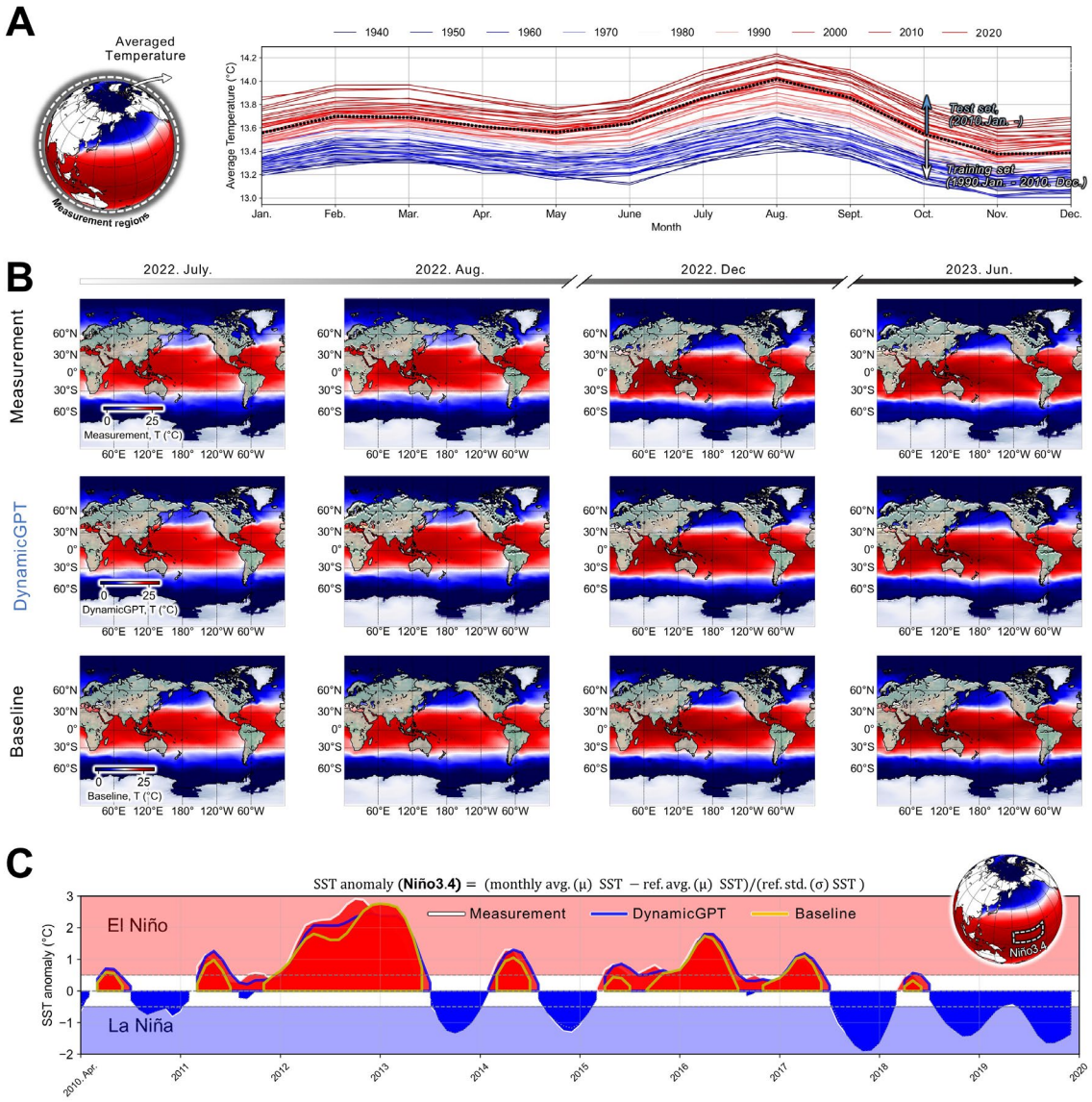


Figure 4. Global Sea Surface Temperature (SST) Prediction and Validation. (**a**) Historical and Future SST Trends: Displays averaged temperature trends from 1940 to 2023, with training data from 1990 to 2010 and test data from 2010 onwards. (**b**) Prediction Accuracy: Comparison of measured SST, DynamicGPT predictions, and baseline model predictions for key months between 2022 and 2023. DynamicGPT shows closer alignment with actual measurements, particularly in capturing spatial patterns and temperature anomalies. (**c**) El Niño and La Niña SST Anomaly Prediction: Time series analysis of SST anomalies in the Niño3.4 region, highlighting the superior performance of DynamicGPT in forecasting climate oscillations compared to the baseline model.

Scenario IV: Extending to 3D - Simulating Reaction-Diffusion Systems for Chemical Processes

In this scenario, DynamicGPT's capabilities are extended to handle 3D spatiotemporal dynamics, a crucial advancement for broader scientific and engineering applications. Beyond validating the model's performance in 2D simulations and real-world experiments, its ability to accurately simulate 3D processes is essential for tackling more complex challenges. To evaluate this, we adapted DynamicGPT by replacing its 2D convolutional layers with 3D convolutions and retrained the model on a 3D Gray-Scott reaction-diffusion system dataset, which is commonly used to model intricate chemical reactions and diffusion processes[41]. This dataset serves as an ideal testbed for evaluating the model's 3D prediction capabilities.

Figure 5a illustrates the Gray-Scott mechanism, where chemical species undergo diffusion and reaction driven by intricate boundary conditions and equations central to PINN However, such physics-based approaches often struggle with long-term predictions due to instability issues, limiting their effectiveness in real-world scenarios. Our results demonstrate that DynamicGPT can successfully generalize to more complex 3D problems. The model was trained on varied initial source distributions and then tested on generalized scenarios with unseen geometries (**Figure 5a**). **Figure 5b** shows the model's long-term prediction accuracy across different time steps ($10\Delta t$ to $160\Delta t$). DynamicGPT closely matches ground-truth simulations for all time steps, capturing both diffusion patterns and reaction fronts with high fidelity in 3D space. Crucially, this performance is achieved entirely via data-driven learning, without relying on embedded physical laws, highlighting the model's capacity to generalize solely from spatiotemporal data.

In contrast, the baseline model exhibits significant inconsistencies, with unexpected diffusion behavior and severe error accumulation over time, leading to distorted predictions. These errors underscore the limitations of traditional models when scaling to higher dimensions

and complex dynamics. Overall, these findings highlight DynamicGPT's adaptability and robustness when transitioning from 2D to 3D applications, maintaining accuracy even in high-dimensional spatiotemporal systems. This extension solidifies DynamicGPT's potential for real-world 3D simulations across various scientific fields, offering a scalable alternative to traditional physics-driven models.

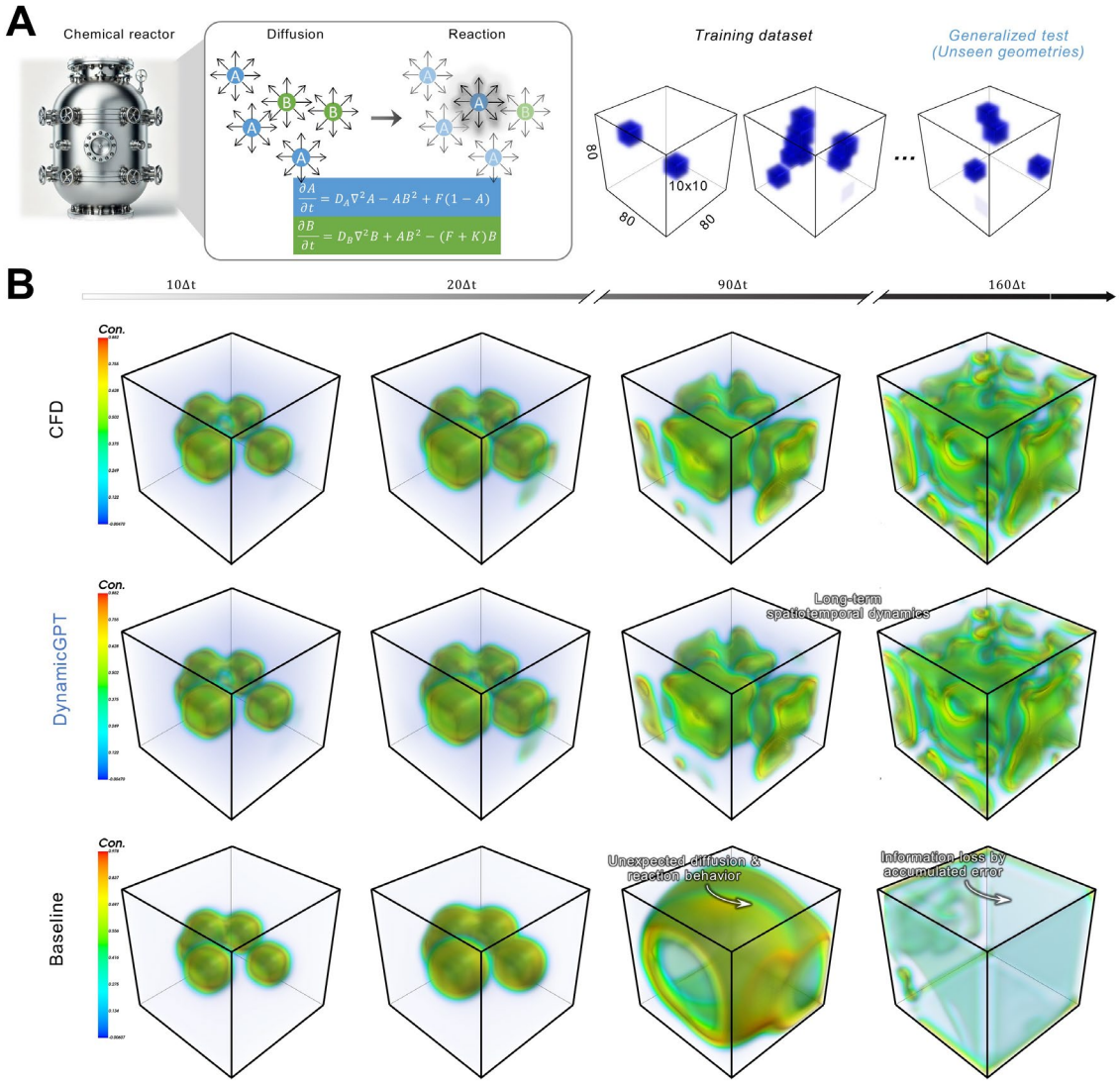


Figure 5 Extending DynamicGPT to 3D Reaction-Diffusion Systems. (**a**) Overview of the reaction-diffusion mechanism, where chemical species A and B diffuse and react within a 3D chemical reactor, governed by nonlinear equations. The training dataset comprises various initial source distributions, while the generalized test involves unseen geometries, challenging the model's ability to predict

complex spatiotemporal dynamics. (**b**) Comparison of concentration fields at different time steps ($10\Delta t$, $20\Delta t$, $90\Delta t$, and $160\Delta t$) with simulation, DynamicGPT, and the baseline model. These results highlight DynamicGPT's scalability and robustness in 3D spatiotemporal system

Performance Assessment of DynamicGPT: Physical Consistency and Computational Efficiency

Understanding the importance of targeted evaluations is crucial for demonstrating a model's applicability to real-world scenarios. These assessments confirm whether models can be reliably used in practical applications, where adherence to physical principles and computational efficiency are critical. Here, we evaluate DynamicGPT's performance through physical conservation law assessments and inverse PDE parameter estimation, emphasizing its predictive accuracy, physical consistency, and computational efficiency.

Assessing physical conservation laws, such as force equilibrium and mass balance, is essential to ensure that predictive models align with fundamental physical principles. Failure to adhere to these laws can lead to unreliable or unsafe outcomes in engineering and scientific applications. Additionally, accurate parameter estimation bridges theoretical models and data-driven approaches, enabling models to adjust and fine-tune coefficients critical for predictive reliability.

DynamicGPT's compliance with physical conservation laws is tested through scenarios involving stress evolution and unsteady flow (as shown in **Figure 6a**). This involves calculating the deviations between simulation benchmarks and model predictions, ensuring models adhere to physical constraints. The complete methodology for this assessment is provided in **Supplementary Information D.1**. For parameter estimation, **Figure 6b** illustrates how DynamicGPT and baseline models infer unknown coefficients (D_u, D_v, F, k) within the governing equations for the reaction-diffusion model: $u_t = D_u \nabla^2 u - uv^2 + F(1-u)$, $v_t = D_v \nabla^2 v$

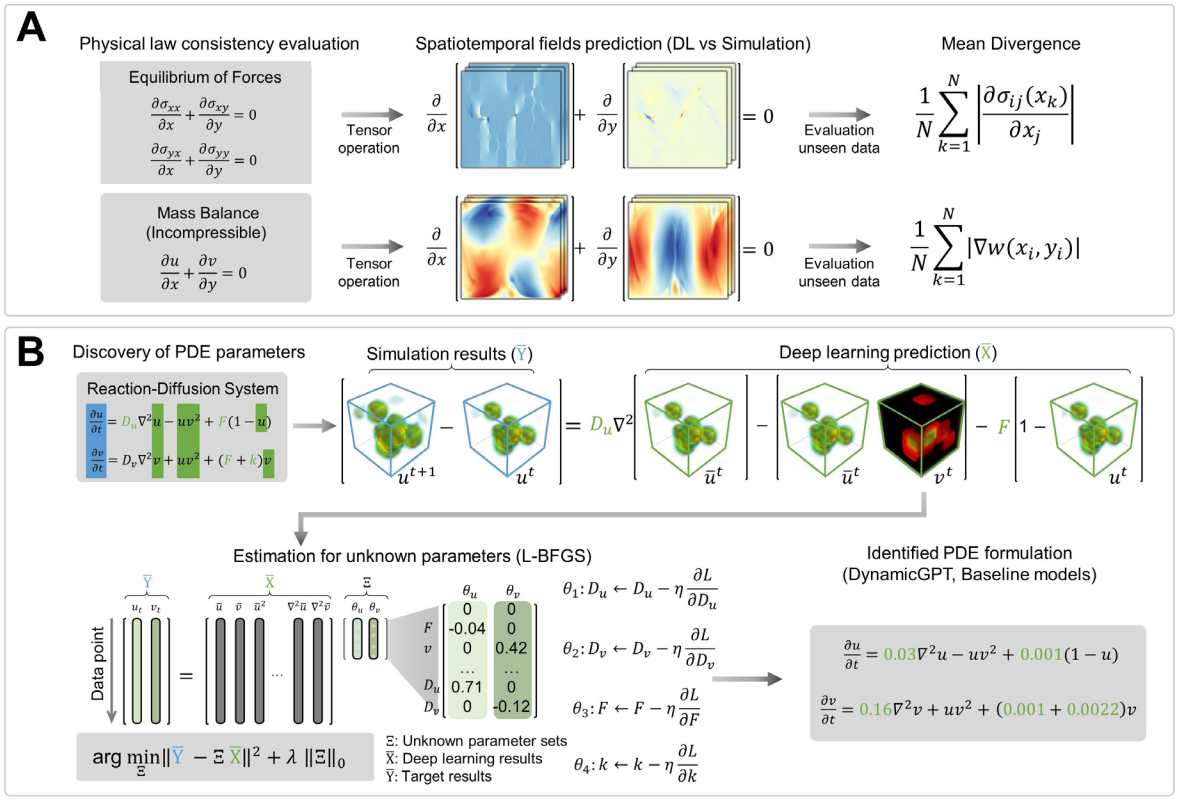
$+ uv^2 - (F + \kappa)v$. The models adjust these parameters using optimization techniques to minimize the error between the predicted and true values, demonstrating their capability to capture complex system behaviors. Detailed procedures are available in **Supplementary Information D.4**.

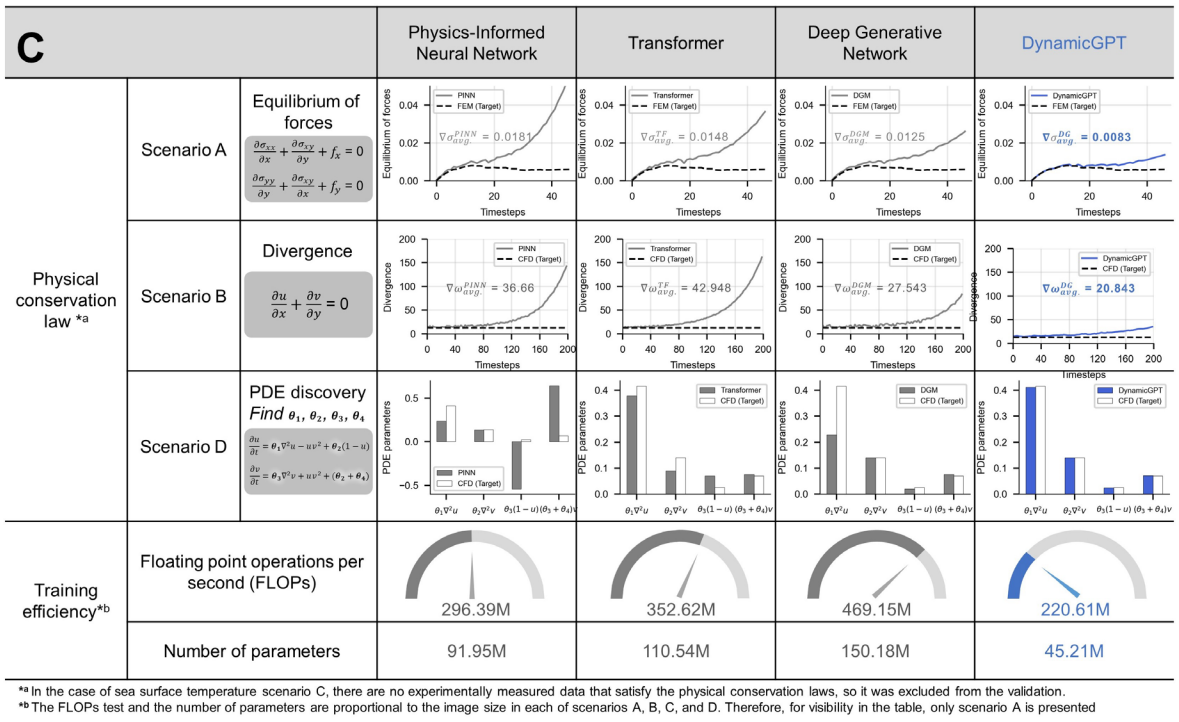
The results, summarized in **Figure 6c** and further detailed in **Figures S9** and **S10**, highlight DynamicGPT's superior performance in physical consistency and parameter estimation. In conservation law adherence, DynamicGPT achieves a mean force equilibrium deviation of 0.0083, closely matching the simulation benchmark of 0.0072. This performance surpasses that of PINNs (0.0181), Transformers (0.0148), and DGMs (0.0125), showcasing DynamicGPT's robustness even in unseen high-strength configurations with properties up to 32% stronger than the training data. This ability is crucial for applications requiring long-term stability and precise predictions, especially in engineering tasks where physical consistency is paramount.

In the inverse estimation task, DynamicGPT demonstrates its capability to accurately estimate target parameters, showcasing its utility for scenarios demanding precise physical calibration. Unlike baseline models that often face parameter instability or convergence issues, DynamicGPT maintains stability and accurate estimates throughout the training process (see **Figure 6c**, third row). This reliability underlines its robustness for complex spatiotemporal applications. For full parameter estimation analysis, refer to **Supplementary Information D.5** This finding is particularly significant for scenarios involving extreme mechanical properties where maintaining force equilibrium is crucial. Conventional models like PINNs, using a ResNet backbone, rely on penalized loss functions to address physical constraints but often show diminishing performance over extended sequences due to architectural limitations. State-of-the-art models like Transformers and DGMs, while optimized for temporal modeling, face

challenges such as spatial information loss and gradient vanishing during long-term forecasts. DynamicGPT's Vision Transformer (ViT)-based architecture resolves these issues by preserving multi-scale spatial features and enabling reliable long-term predictions (see **Figure S9** for detailed comparisons). In addition, DynamicGPT's capabilities also extend effectively to unsteady flow assessments, reinforcing its applicability across various engineering challenges (Second row in **Figure 6c**). These results highlight the potential of DynamicGPT to bridge the gap between traditional computational simulations and data-driven models. Achieving near-simulation-level accuracy in long-term spatiotemporal predictions, DynamicGPT showcases its capability for reliable prediction.

In addition, from a computer science perspective (**Figure 6c**), we assess the computational efficiency of each model by analyzing floating-point operations per second (FLOPs). PINNs, with their simple backbones, achieve low FLOPs but at the expense of predictive accuracy. State-of-the-art models, although achieving high accuracy, incur substantial computational costs due to the complex neural network architecture, limiting their scalability for real-world applications. DynamicGPT overcomes these challenges by incorporating multi-scale kernels in its convolution operations, significantly reducing parameter counts while optimizing FLOPs without compromising accuracy. This design not only ensures computational efficiency but also enhances the model's applicability across diverse spatiotemporal tasks, including real-time monitoring systems and deployment on edge devices with limited processing power. In summary, these results underscore DynamicGPT's balanced performance in terms of physical fidelity and computational efficiency, making it a powerful tool for accurately and efficiently predicting spatiotemporal dynamics in complex systems. DynamicGPT's ability to manage both long-term predictions and computational efficiency represents a significant advancement over traditional models, offering robust and versatile predictions suitable for a wide range of scientific and industrial applications.





*a In the case of sea surface temperature scenario C, there are no experimentally measured data that satisfy the physical conservation laws, so it was excluded from the validation.
*b The FLOPs test and the number of parameters are proportional to the image size in each of scenarios A, B, C, and D. Therefore, for visibility in the table, only scenario A is presented

Figure 6. Evaluation of DynamicGPT for Physical Law Consistency and Parameter Estimation. (**a**) Physical law consistency assessment for stress evolution and unsteady flow. (**b**) Inverse estimation of unknown parameters in a 3D reaction-diffusion system. (c) Comparative analysis of DynamicGPT with baseline models in physical law adherence and parameter estimation.

Guidelines for Tuning DynamicGPT Across Diverse Spatiotemporal Dynamics

DynamicGPT consistently delivers high accuracy across multiple real-world scenarios without requiring physical knowledge or boundary condition constraints. Compared to existing baseline models like deep generative model, PINNs and transformer-based architectures, DynamicGPT demonstrated superior generalization, as indicated by RSME metrics across all test sets (**Table 1**). For a detailed description of the baseline models, see **Supplementary Information C**. Importantly, DynamicGPT excelled in capturing long-term spatiotemporal dynamics—a key challenge for conventional approaches.

While DynamicGPT's adaptability and effectiveness have been proven, the next critical step is providing clear guidelines for modelers and scientists aiming to design neural networks for novel spatiotemporal phenomena. The key question becomes: *How can a model be optimally designed based solely on the nature of the data it encounters?* **Extended Data Fig. 1** outlines essential guidelines for neural network design based on the diverse physical characteristics of datasets.

- **β-Parameter Tuning**: The β-value controls the balance between generative capabilities and reconstruction accuracy in the variational autoencoder (VAE) framework. For systems with higher levels of nonlinearity and stochasticity—such as air pollution dispersion and reaction-diffusion processes—a higher β-value is essential. This setting enhances the model's ability to account for multi-scale spatial embedding, prioritizing generative diversity. For instance, in the air quality prediction scenario, increasing the β-value was crucial in capturing the stochastic nature of pollutant dispersion, leading to a significant R^2 improvement compared to the baseline configuration. Conversely, for more linear and periodic data like SST, lowering the β-value optimizes reconstruction from existing data distributions, leading to improved

prediction accuracy. Hence, adjusting the β-parameter according to the physical properties of the problem is key to achieving optimal performance.

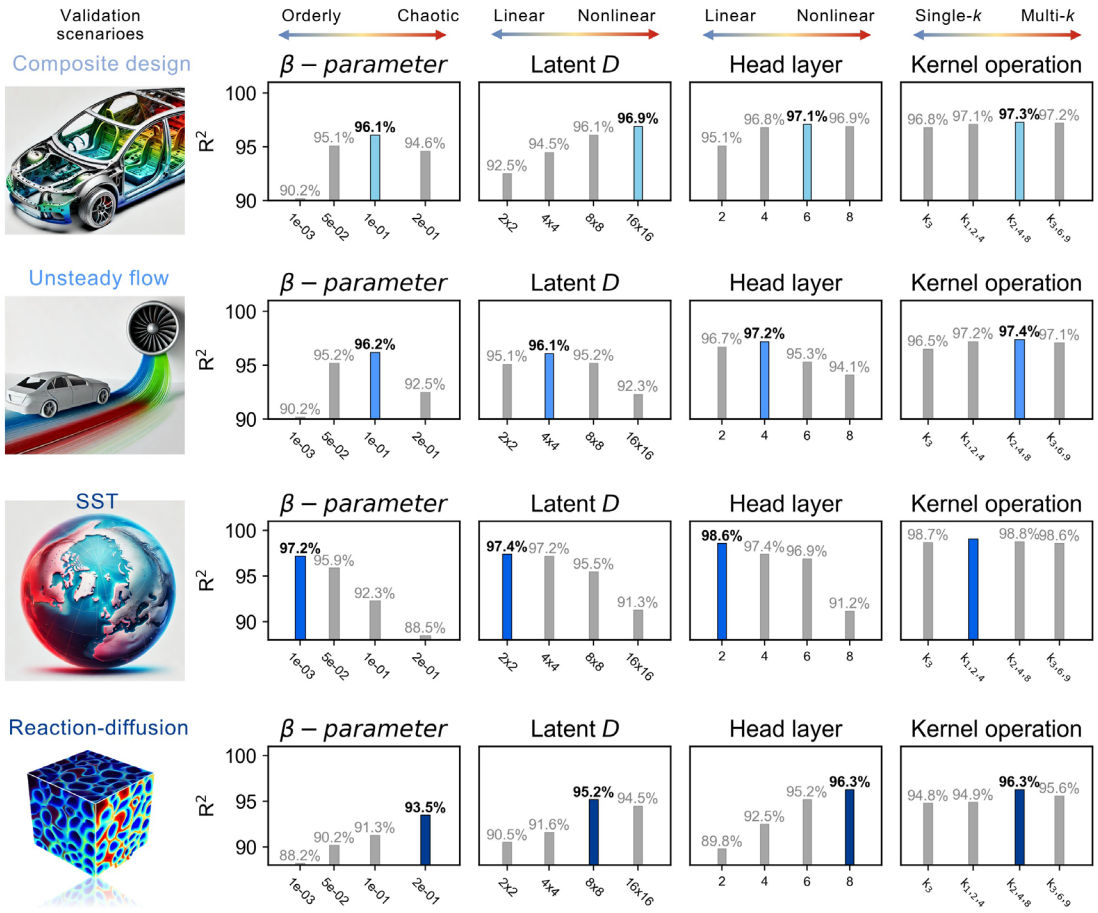
- **Latent Dimension (Latent *D*)**: The latent dimension refers to the length of the square patches used as inputs for the tailored ViT. Larger latent dimensions retain more spatial information, which is crucial for complex systems. A dimension of 1 represents the typical transformer input-output connection. Our findings indicate that a latent dimension of 16 provides a balanced capacity to model diverse dynamics. Increasing this dimension beyond 16 introduces inefficiencies in ViT computation. For example, in predicting crack propagation in materials, a higher latent dimension was essential for accurately modeling abrupt stress concentration and global stress pattern changes. While smaller latent dimensions may suffice for simpler, periodic phenomena, systems with abrupt changes—like rapid material fracture or localized stress concentration—require larger latent dimensions to adequately represent intricate patterns.

- **Head Layer Configuration**: The number of head layers in the tailored ViT directly influences the integration of multi-scale temporal features. Deeper self-attention layers are particularly effective at learning multi-scale temporal dynamics, capturing both short- and long-term dependencies. For most scenarios, 4 to 6 head layers provided optimal results, especially for non-linear dynamics where capturing multi-scale dependencies is crucial. This aligns closely with the behavior observed when tuning the latent dimension. For instance, in fluid dynamics scenarios like unsteady flow past bodies, the optimal head layer configuration was found to be 6, which effectively captured both localized vortices and global flow structures.

- **Kernel Operations**: Incorporating multi-scale kernels is vital for handling both localized and global patterns. Our results show that using a combination of kernels of

varying sizes (e.g., 2, 4, and 8) leads to more accurate predictions across different scales, especially in highly complex systems like turbulent flows and global climate dynamics. The initial recommendation is to begin with this combination for optimal extraction of both local and global features. For example, when predicting turbulent channel flow, this multi-kernel approach effectively balanced small-scale eddy formations with larger-scale flow structures, leading to significantly improved R^2 scores.

These tuning guidelines not only enhance DynamicGPT's flexibility and robustness but also provide a clear framework for modelers working with novel spatiotemporal phenomena. Beyond the physical phenomena explored in this study, these principles have broader relevance, guiding neural network design for various applications ranging from medical imaging to financial market predictions. By following these principles, researchers and engineers can confidently apply DynamicGPT to a wide array of scientific and engineering problems, ensuring the model's reliability and accuracy across diverse data-driven scenarios.



Extended Data Fig. 1. DynamicGPT Neural Network Tuning Guidelines Across Diverse Spatiotemporal Phenomena. The figure outlines tuning strategies based on varying levels of nonlinearity, periodicity, and complexity in the datasets. From top to bottom, the scenarios cover composite design (orderly systems), unsteady fluid flow (chaotic systems), air quality prediction (linear systems), and SST prediction (periodic systems). Four key parameters were examined: (1) β-parameter, which controls the balance between generative capacity and reconstruction accuracy in the VAE framework; (2) Latent Dimension (Latent D), representing the input patch size in the Vision Transformer (ViT); (3) Head Layer Configuration, which affects the integration of multi-scale temporal features; and (4) Kernel Operations, emphasizing the importance of multi-scale kernel combinations for capturing both local and global patterns.

Conclusions

This study presents DynamicGPT, a versatile generative transformer model specifically optimized for long-term spatiotemporal dynamics across diverse real-world applications. Through comprehensive evaluations, DynamicGPT demonstrated superior performance in accurately predicting complex systems without relying on explicit physical knowledge or boundary condition constraints. Its consistent success in handling unseen future sequences, out-of-distribution data, and three-dimensional spatial domains underscores its adaptability and robustness, setting it apart from conventional models such as PINNs and transformer-based architectures.

A key innovation of DynamicGPT lies in its multi-scale architecture, integrating both spatial and temporal features while preserving essential spatial information. By employing multi-scale kernels, the model captures intricate dynamics, from composite material failures to global sea surface temperature changes and 3D reaction-diffusion patterns. These results illustrate DynamicGPT's potential as a scalable and efficient alternative to traditional simulations, enabling significant advancements in real-time predictive capabilities across scientific and engineering fields.

DynamicGPT's reliable predictions of stress propagation and fracture patterns in composite materials imply promising applications in material safety analysis, allowing for better-informed decisions in structural design and failure prevention. Its capability to model complex fluid flows accurately can improve engineering solutions in aerodynamics and environmental monitoring, ensuring safer and more efficient designs. Moreover, precise SST forecasting highlights its potential role in climate research, providing valuable data for predicting climate anomalies such as El Niño events, which have far-reaching implications for disaster response and global weather patterns. The model's successful application to 3D

reaction-diffusion systems paves the way for improved simulations in chemical engineering, potentially accelerating research in reaction kinetics and catalyst design.

Additionally, we provide critical guidelines for designing neural networks tailored to novel spatiotemporal phenomena. The outlined tuning strategies—including β-parameter adjustments, latent dimension optimization, and multi-scale kernel configurations—offer practical insights for researchers and engineers adapting DynamicGPT to specific tasks. These recommendations ensure that the model can be applied confidently to a wide range of applications, from environmental monitoring to industrial process optimization, achieving high accuracy and reliability.

While DynamicGPT sets a new benchmark for spatiotemporal modeling, it also presents opportunities for further exploration. Extending this framework to tackle even more complex challenges, such as high-dimensional chaotic systems or highly irregular and stochastic patterns, is a promising avenue for future work. By advancing the boundaries of what can be achieved with generative transformer networks, DynamicGPT provides a flexible and powerful foundation for meeting the growing demand for accurate, data-driven predictions in various scientific and engineering domains. These advancements could lead to transformative impacts on how industries approach real-time decision-making and predictive modeling.

Methods

DynamicGPT Implementation Details and Training Process

DynamicGPT is designed to predict long-term spatiotemporal dynamics by leveraging a combination of the Multi-Spatial Embedding Network (***E***), Multi-Scale Temporal Network (***D***), and Multi-Spatial Decoder (***F***), which together form the generator. To enhance the generator's performance, DynamicGPT also incorporates spatial and temporal discriminators

that refine the generated predictions, allowing the model to effectively capture complex, multi-scale features in spatiotemporal data without relying on explicit physical constraints.

The process begins with high-dimensional spatiotemporal data being input into the Multi-Spatial Embedding Network (***E***). This network uses a multi-scale kernel approach, where various kernel sizes—adapted to the specific problem—are employed to capture both local and global spatial features. Smaller kernels focus on fine-grained details, essential for capturing localized phenomena such as small-scale turbulence or pollution hotspots, while larger kernels capture broader, long-range patterns like large-scale atmospheric dynamics. This combination effectively encodes spatial information across different scales, which is crucial for accurate predictions.

The embedding network is structured with multiple layers of convolutional operations, skip connections, and feature fusion techniques to maintain spatial resolution while reducing computational complexity. Skip connections prevent information loss during training, while feature fusion integrates multi-scale features into a single, comprehensive latent representation. Dimensionality reduction through pointwise convolution ensures that the model preserves essential spatial features in a lower-dimensional latent space. Additionally, a Convolutional Variational Autoencoder (CVAE) is employed within this network, capturing the probabilistic and non-linear dynamics inherent in the data by assigning means and variances to the latent patches. The CVAE balances generative diversity and reconstruction accuracy through the β-parameter, which is adjusted depending on the specific nature of the spatiotemporal data.

After embedding, the latent space is reshaped to restore necessary dimensionality before being passed to the Multi-Scale Temporal Network (***D***). This network uses a Vision Transformer (ViT) architecture to process sequences of varying lengths, capturing both short-term fluctuations and long-term trends. The ViT's self-attention mechanism excels at learning

temporal dependencies across different scales, handling relationships between input and output in parallel—unlike RNNs, which process sequences step-by-step. Positional embeddings preserve the temporal order of latent patches, and a ConvGRU layer further refines dynamic interactions, integrating multi-scale temporal patterns into predictions. Finally, the Multi-Spatial Decoder (***F***) reconstructs the high-dimensional solution sets from the predicted latent patches, generating accurate forecasts of future spatiotemporal fields. For more comprehensive details on the mechanisms behind the architecture of DynamicGPT and the specific final configurations used across various spatiotemporal scenarios, please refer to the **Supplementary Information Section D**.

Explanation of Discriminators and Training Dynamics

To ensure the generated fields are both realistic and temporally consistent, DynamicGPT incorporates two critical components: The Spatial Discriminator and the Temporal Discriminator. The Spatial Discriminator, implemented as a CNN, evaluates the spatial consistency of generated fields by distinguishing real spatial patches from generated ones. It operates on random crops of both the input and predicted fields to ensure sharp and realistic spatial details. The Temporal Discriminator employs 3D convolutions to assess the temporal coherence of sequences, ensuring that the generated sequences maintain consistent temporal dynamics by penalizing any abrupt or unrealistic transitions between time steps. This approach is particularly effective at enforcing temporal smoothness across long sequences, which is crucial for maintaining the accuracy of long-term spatiotemporal predictions. During training, the generator and discriminators are updated in an alternating fashion. The discriminators provide feedback to the generator, challenging it to produce more realistic outputs. Typically, the discriminators are updated more frequently than the generator to ensure they remain effective adversaries, constantly driving improvements in the generator's

performance. The detailed inference process was represented in Supplementary Information.

Mathematical Formulations and Loss Functions

The training of DynamicGPT is guided by a combination of adversarial and reconstruction losses, which together optimize the model's ability to generate accurate and realistic predictions. The total loss function, L_{total}, is defined as:

$$L_{total} = \lambda_D L_D + \lambda_T L_T + \lambda_R L_R + \lambda_{reg} L_{reg}$$

- L_D is the **Spatial Adversarial Loss** derived from the Spatial Discriminator, formulated as a hinge loss. This loss ensures that the spatial structures within the generated predictions closely mimic those found in the real data.
- L_T is the **Temporal Adversarial Loss** derived from the Temporal Discriminator, also formulated as a hinge loss. It penalizes inconsistencies in the temporal sequences of the generated data, ensuring temporal coherence.
- L_R is the **Reconstruction Loss**, implemented as an L2 loss (mean squared error), which ensures that the generated predictions are accurate in terms of actual values across all grid cells and time steps.
- L_{reg} is the **Regularization Term** applied to stabilize training and prevent overfitting, particularly by ensuring that the mean prediction remains close to the ground truth.

These loss functions collectively drive the generator to produce predictions that are not only realistic in terms of spatial and temporal dynamics but also precise in their numerical accuracy.

Hyperparameter Optimization and Implementation Details

The model's hyperparameters are carefully fine-tuned to optimize performance across different scenarios. The learning rate is set to 1×10^{-4} and optimized using the Adam optimizer, selected for its adaptive learning rate capabilities and effectiveness in training deep neural networks. Batch sizes are adjusted based on dataset complexity, ranging from 16 for simpler scenarios to 64 for more intricate 3D simulations. The number of epochs is 20,000 for all scenarios. Regularization techniques, including dropout layers (with a rate of 0.2) and L2 weight decay, are employed to prevent overfitting, particularly in scenarios involving large datasets and complex 3D simulations. Early stopping, based on validation loss, ensures that the model does not overfitting and maintains generalization capability.

Detailed Validation Datasets

Case 1: Composite Material Fracture Analysis

Validation Case 1 focuses on spatiotemporal stress fields and crack propagation in composite materials with differing mechanical properties. In such materials, local stress concentrations and the resulting failure behavior can vary significantly depending on the material configuration. Therefore, this study aims to validate a deep learning model that can predict crack propagation in out-of-distribution scenarios, where the failure characteristics are stronger than those present in the training set.

Specifically, the simulation uses a hybrid scheme-based crack phase field model[46,47] solved via FEM to generate data on how stress and crack patterns evolve based on the composite material configuration. Unlike the extended finite element method (XFEM) or cohesive zone modeling, the crack phase field method does not require special tracking or insertion of cohesive elements. This makes it more suitable for cases involving evolving or branching cracks in composite materials. As a test bed, the composite is composed of stiff and

soft blocks arranged in an 11 × 11 array, with the stiff-to-soft block ratio fixed at 71:50 (58.7% stiff blocks). The material's moduli exhibit a 100-fold difference. A single edge crack was introduced to initiate crack propagation, with the crack phase field model implemented in Abaqus. For more details on the simulation parameters, please refer to the referenced study. The 11 × 11 array provides a vast design space, with approximately 10^{34} possible combinations when fixing the volume ratio of the two materials. Even with the fixed ratio of 71:50, this still offers a large enough test bed to explore diverse crack propagation scenarios.

We generated 4,000 stress/crack field datasets through random sampling, with 2,900 used for training, 100 for validation, and 1,000 for testing. For each configuration, stress and crack phase field data were processed over 50-time steps, up to the point of material failure. The initial three-time steps were input into DynamicGPT, and an autoregressive method (see **Supplementary Information B**) was used to predict the following 47-time steps. This dataset covers a wide range of crack propagation scenarios in composite materials. Conventional deep learning models often struggle to accurately predict unseen configurations, particularly those with superior properties not present in the training set, when trained on small datasets. To evaluate the generalizability of our framework, the test dataset was designed with average strength and toughness values 35% and 42% higher, respectively, than those of the training dataset. For further details and access to the full dataset, please refer to the dataset available at *https://github.com/DonggeunPark/MaterialsHorisonsSTGNet*.

Case 2-i: Unsteady Fluid Flow – Wake Dynamics Around Multiple Bodies

Validation case 2 consists of a two-dimensional unsteady flow around multiple bodies that generates periodic wakes and chaotic wakes. The interaction between multiple bodies significantly increases the complexity of the flow field compared to a single body. This complexity presents practical challenges that play a crucial role in engineering applications,

especially in aircraft design and marine structures.

The three bodies are arranged in a triangular arrangement, each assigned a different configuration—rectangular, circular, or semicircular. These configurations are simulated using unsteady CFD to obtain spatiotemporal dynamic data as the 100 configuration changes. The distance between the bodies is defined by the l/d (Spacing Ratio), which ranges from 1.2 to 5.5, allowing observations of changes in boundary layer flow and vortex formation based on shape variations. The inlet velocity U was set at 1 m/s from the left boundary, with a Reynolds number (Re) of 100.

Half of these configurations (with a number of 50) are used for training, and the remaining 50 data were used for testing. The computational domain size is 384 × 256 pixels, with a time step of $\Delta t = 0.1$s. Simulations generate for 300 seconds of physical time (with 3,000 snapshots), and the first 150-second snapshots are discarded to ensure that only the statistically stable flow regime was used, eliminating transient effects. From the remaining data, a sliding window approach is used to create 46-second snapshots for training DynamicGPT and 3-second snapshots for validation. After training, DynamicGPT is tested autoregressively, using 10-second snapshots to forecast the convection phenomena over the following 100 seconds. This split is designed to validate the model's performance in extrapolating more challenging out-of-distribution sequences (future sequences). For detailed data and further analysis, please refer to the dataset available at *https://github.com/tum-pbs/DiffPhys-CylinderWakeFlow*.

Case 2-ii: Unsteady Fluid Flow – Rayleigh-Bénard Convection

Validation Case 3 focuses on the prediction of spatiotemporal velocity fields within a two-dimensional turbulent flow regime, simulated using the Lattice Boltzmann Method (LBM) to capture the dynamics of Rayleigh-Bénard Convection (RBC). This phenomenon arises when a fluid layer is heated from below and cooled from above, generating complex flow patterns

driven by buoyancy forces.

The simulation generates the velocity fields across a computational domain represented by 1792 × 256-pixel snapshots, with two channels for the x and y velocity components. The key physical parameters include a Prandtl number of 0.71, a Rayleigh number of 2.5×10^8, and a maximum Mach number of 0.1. In total, 1,500 snapshots are generated. The sliding window approach is used to create 1,200-timestep snapshots for training DynamicGPT and 190-timestep snapshots for validation. After training, DynamicGPT is tested autoregressively, using 10-timestep snapshots as input to predict the convection phenomena over the next 100 timesteps.

The key point is that the model is trained on the left 256 × 256 region, and its performance is tested on the 1,536 × 256 region. The larger spatial domain for testing was selected to demonstrate the model's ability to generalize to unseen, larger domains, which is crucial for evaluating how well the deep learning model can handle extrapolation to larger computational domain, especially in real-world applications. For further details and access to the full dataset, please refer to *https://github.com/Rose-STL-Lab/Turbulent-Flow-Net*.

Case 3: Global Sea Surface Temperature (SST) Prediction

Validation Case 5 utilizes SST data from the National Oceanic and Atmospheric Administration (NOAA), collected through satellite and ship-based measurements (*http://www.esrl.noaa.gov/psd/).* Accurate SST prediction is crucial as it plays a significant role in refining climate models and is vital for understanding long-term climate variability and interactions within ocean-atmosphere systems.

The dataset comprises 1,742 temperature fields. The training period covers January 1, 1990, to June 4, 2010, while the testing period extends from June 5, 2010, to June 30, 2023. The original data have been interpolated onto a 360 × 180-pixel grid, with temperature readings

extracted from approximately 10,300 locations per sample.

For this study, 12 sequential time steps are input into DynamicGPT, and the model predicts the next 36-time steps. This long-term forecasting approach allows for a comprehensive evaluation of SST prediction performance, which is fundamental to enhancing climate model accuracy and deepening our understanding of complex oceanic processes.

Case 4: 3D Gray-Scott Reaction-Diffusion System

Validation Case 5 expands previous 2D cases into a 3D spatiotemporal dynamic system, modeled by the Gray-Scott Reaction-Diffusion system. This system simulates the interaction and reaction between two species, A and B, where the species react and consume each other. This phenomenon is critical for understanding pattern formation in reaction-diffusion systems, with applications across biology, chemistry, and other scientific fields. The dataset is generated by initializing 4,000 random species distributions, followed by numerical simulations. For each simulation, the initial conditions are set by introducing random perturbations in the domain. Of these, 3,800 samples are used for training, and 200 samples are reserved for testing. The governing equations of the reaction-diffusion system are discretized using the finite differencing method and solved within an 80 × 80 × 80 computational domains. In this domain, 1 to 5 species distributions, each measuring 10 × 10 × 10, are randomly positioned as initial conditions. The simulations track the diffusion and interaction of these species over time. The Laplacian of the species concentration fields is calculated to model diffusion, while the reaction dynamics follow the Gray-Scott kinetics. In this setup, the feed and kill rates for species A and B are set to 0.037 and 0.06, respectively, with diffusion rates D_u = 0.16 and D_v = 0.1. The simulation proceeds for 200-time steps with a time increment of dt = 1. DynamicGPT predicts 170 future time steps based on 30 observed steps, providing a comprehensive evaluation of model performance over a long-term prediction horizon.

Data availability

All datasets used in this study are openly available in Zenodo, accessible at: https://doi.org/10.5281/zenodo.14184896.

Code availability

The codes used for this work are available at: https://github.com/KAIST-M4/Forecasting-Long-term-Spatial-temporal-Dynamics-with-Generative-Transformer-Networks.

Acknowledgements

This research was supported by the Basic Science Research Program (2022R1A2B5B02002365) and the Engineering Research Center program (RS-2023-00222166), funded by the National Research Foundation of Korea, as well as a grant from the Ministry of Food and Drug Safety of Korea (RS-2023-00215667).

Contributions

DGP: Conceptualization, Methodology, Investigation, Data curation, Software, Writing – original draft, Writing – review & editing. **HGL**: Investigation, Data curation. **SHR**: Conceptualization, Investigation, Validation, Supervision, Writing – review & editing.

References

1. Teixeira, F.L., Sarris, C., Zhang, Y., Na, D.-Y., Berenger, J.-P., Su, Y., Okoniewski, M., Chew, W.C., Backman, V. & Simpson, J.J. Finite-difference time-domain methods. *Nat. Rev. Methods Primers* **3**, 75 (2023).
2. d'Esposito, A., Sweeney, P.W., Ali, M., Saleh, M., Ramasawmy, R., Roberts, T.A., Agliardi, G., Desjardins, A., Lythgoe, M.F., Pedley, R.B., Shipley, R. & Walker-Samuel, S. Computational fluid dynamics with imaging of cleared tissue and of in vivo perfusion predicts drug uptake and treatment responses in tumours. *Nat. Biomed. Eng.* **2**, 773–787 (2018) .
3. Bousige, C., Levitz, P. & Coasne, B. Bridging scales in disordered porous media by mapping molecular dynamics onto intermittent Brownian motion. *Nat. Commun.* **12**, 1043 (2021).
4. Louie, S.G., Chan, Y.-H., da Jornada, F.H., Li, Z. & Qiu, D.Y. Discovering and understanding materials

through computation. *Nat. Mater.* **20**, 728–735 (2021).

5. Lin, K. & Wang, Z. Multiscale mechanics and molecular dynamics simulations of the durability of fiber-reinforced polymer composites. *Commun. Mater.* **4**, 66 (2023).

6. Li, Y. & Fuhrman, D.R. On the turbulence modelling of waves breaking on a vertical pile. *J. Fluid Mech.* **953**, A3 (2022)([6]_).

7. Magionesi, F., Dubbioso, G. & Muscari, R. Contribution of tip and hub vortex to the structural response of a marine rudder in the propeller slipstream. *J. Fluid Mech.* **946**, A23 (2022).

8. Bauer, P., Thorpe, A. & Brunet, G. The quiet revolution of numerical weather prediction. *Nature* **525**, 47–55 (2015).

9. He, Y., Tan, Y., Yang, M., Wang, Y., Xu, Y., Yuan, J., Li, X., Chen, W. & Kang, G. Accurate prediction of discontinuous crack paths in random porous media via a generative deep learning model. *Proc. Natl. Acad. Sci. USA* **121**, e2413462121 (2024).

10. Park, D., Lee, J., Lee, H., Gu, G.X. & Ryu, S. Deep generative spatiotemporal learning for integrating fracture mechanics in composite materials: inverse design, discovery, and optimization. *Mater. Horiz.* **11**, 3048–3065 (2024).

11. Yan, H. *et al.* Machine learning based framework for rapid forecasting of the crack propagation. *Eng. Fract. Mech.* **307**, 110278 (2024).

12. Bao, K. *et al.* Deep learning method for super-resolution reconstruction of the spatio-temporal flow field. *Adv. Aerodyn.* **5**, 19 (2023).

13. Vinuesa, R. & Brunton, S.L. Enhancing computational fluid dynamics with machine learning. *Nat. Comput. Sci.* **2**, 358–366 (2022).

14. Kochkov, D. *et al.* Machine learning–accelerated computational fluid dynamics. *PNAS* **119**, e2101784119 (2021).

15. Brunton, S.L. & Kutz, J.N. Promising directions of machine learning for partial differential equations. *Nat. Comput. Sci.* **4**, 483–494 (2024).

16. Wu, H. *et al.* Interpretable weather forecasting for worldwide stations with a unified deep model. *Nat. Mach. Intell.* **5**, 602–611 (2023).

17. Zhang, Y. *et al.* Skilful nowcasting of extreme precipitation with NowcastNet. *Nature* **619**, 526–537 (2023).

18. Lew, A.J. *et al.* Deep learning model to predict fracture mechanisms of graphene. *npj 2D Mater. Appl.* **5**, 48 (2021).

19. Geneva, N. & Zabaras, N. Transformers for modeling physical systems. *Neural Netw.* **146**, 272–289 (2022).

20. Yousif, M.Z. *et al.* A transformer-based synthetic-inflow generator for spatially developing turbulent boundary layers. *J. Fluid Mech.* **957**, A6 (2023).

21. Raissi, M. *et al.* Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **367**, 1026–1030 (2020).

22. Chen, Z., Liu, Y. & Sun, H. Physics-informed learning of governing equations from scarce data. *Nat. Commun.* **12**, 6136 (2021).

23. Cavanagh, H. *et al.* Physics-informed deep learning characterizes morphodynamics of Asian soybean rust disease. *Nat. Commun.* **12**, 6424 (2021).

24. Champion, K., Lusch, B., Kutz, J.N. & Brunton, S.L. Data-driven discovery of coordinates and

governing equations. *PNAS* **117**, 22445–22451 (2020).

25. Park, D., Jung, J., Gu, G.X. & Ryu, S. A generalizable and interpretable deep learning model to improve the prediction accuracy of strain fields in grid composites. *Mater. Des.* **223**, 111192 (2022).

26. Lee, J. *et al.* Machine learning-based inverse design methods considering data characteristics and design space size in materials design and manufacturing: a review. *Mater. Horiz.* **10**, 5436–5456 (2023).

27. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A.A. Image-to-image translation with conditional adversarial networks. *CVPR* **2017**, 1125–1134 (2017).

28. Dosovitskiy, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR* (2021).

29. Yang, Z. *et al.* Deep learning model to predict complex stress and strain fields in hierarchical composites. *Sci. Adv.* **7**, eabd7416 (2021).

30. Milla-Val, J., Montañés, C. & Fueyo, N. Adversarial image-to-image model to obtain highly detailed wind fields from mesoscale simulations in urban environments. *Build. Environ.* **266**, 112123 (2024).

31. Wang, Q., Yang, L. & Huang, K. Fast prediction and sensitivity analysis of gas turbine cooling performance using supervised learning approaches. *Energy* **246**, 123373 (2022).

32. Choi, S. & Kim, Y. Rad-cGAN v1.0: Radar-based precipitation nowcasting model with conditional generative adversarial networks for multiple dam domains. *Geosci. Model Dev.* **15**, 5967–5985 (2022).

33. Wang, Z. *et al.* Pore-scale modeling of multiphase flow in porous media using a conditional generative adversarial network (cGAN). *Phys. Fluids* **34**, 123325 (2022).

34. Koch, R. & Lado, J.L. Designing quantum many-body matter with conditional generative adversarial networks. *Phys. Rev. Res.* **4**, 033223 (2022).

35. Kim, J., Kim, J. & Lee, C. Prediction and control of two-dimensional decaying turbulence using generative adversarial networks. *J. Fluid Mech.* **981**, A19 (2024).

36. Chen, C.-H., Chen, K.-Y. & Shu, Y.-C. Data-driven bio-mimetic composite design: Direct prediction of stress–strain curves from structures using cGANs. *J. Mech. Phys. Solids* **193**, 105857 (2024).

37. Ballas, N., Yao, L., Pal, C. & Courville, A. Delving deeper into convolutional networks for learning video representations. *ICLR* (2016).

38. Zhang, M. *et al.* On the damage tolerance of 3-D printed Mg-Ti interpenetrating-phase composites with bioinspired architectures. *Nat. Commun.* **13**, 3247 (2022).

39. Park, J. *et al.* CFD-based design optimization of ducted hydrokinetic turbines. *Sci. Rep.* **13**, 17968 (2023).

40. Cael, B.B. *et al.* Historical and future maximum sea surface temperatures. *Sci. Adv.* **10**, eadj5569 (2024).

41. Fromenteze, T. *et al.* Morphogenetic metasurfaces: unlocking the potential of Turing patterns. *Nat. Commun.* **14**, 6249 (2023).

42. Nguyen, P.C.H. *et al.* PARC: Physics-aware recurrent convolutional neural networks to assimilate meso scale reactive mechanics of energetic materials. *Sci. Adv.* **9**, eadd6868 (2023).

43. Solera-Rico, A. *et al.* β-Variational autoencoders and transformers for reduced-order modelling of fluid flows. *Nat. Commun.* **15**, 1361 (2024).

44. Ravuri, S. *et al.* Skilful precipitation nowcasting using deep generative models of radar. *Nature* **597**, 672–677 (2021).

45. Liu, Y. *et al.* Enhanced multi-year predictability after El Niño and La Niña events. *Nat. Commun.* **14**,

6387 (2023).

46. Ambati, M., Gerasimov, T. & De Lorenzis, L. A review on phase-field models of brittle fracture and a new fast hybrid formulation. *Comput. Mech.* **55**, 383–405 (2015).

47. Jeong, H., Signetti, S., Han, T.-S. & Ryu, S. Phase field modeling of crack propagation under combined shear and tensile loading with hybrid formulation. *Comput. Mater. Sci.* **155**, 483–492 (2018).

Contents

Forecasting Long-term Spatial-temporal Dynamics with Generative Transformer Networks

Donggeun Park, Hugon Lee, and Seunghwa Ryu*

Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea

*Corresponding author: ryush@kaist.ac.kr

A Background

Spatiotemporal dynamics are at the heart of modern science and engineering, shaping our understanding of complex systems and their behavior over time and space. From climate simulations to the intricate analysis of turbulent flows and material failure mechanisms, computational simulations and sensing technologies have become essential tools. These approaches provide invaluable insights into specific scenarios; however, they face significant challenges when it comes to real-time forecasting. The vast amount of data, the substantial computational costs, and the inherent uncertainties associated with dynamic systems make real-time prediction a formidable task.

These limitations are especially pronounced in real-world applications, where it is impractical to perfectly model every aspect of a system. Real-world data often exhibits nonlinear and chaotic interactions among numerous variables, making precise prediction a challenging endeavor. To address these issues, deep learning has emerged as a game-changing approach. With its capacity to learn complex, nonlinear relationships and process data at multiple scales, deep learning has shown remarkable potential in overcoming many of the limitations faced by traditional methods.

Over time, the integration of AI and simulation techniques has evolved significantly, as illustrated in **Figure S1**. This evolution can be divided into four distinct generations, each building on the strengths of its predecessors while attempting to address their weaknesses:

- **Generation 1** introduced Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for processing spatiotemporal data. While these models were effective for learning

simpler patterns, they struggled to capture long-term dependencies, which limited their predictive capabilities.

- **Generation 2** saw the rise of more sophisticated models, including Physics-informed Neural Networks (PINNs), ConvLSTM (a combination of CNNs and RNNs), and Transformers.
  - PINNs integrated governing physical equations into their training processes, enhancing the realism of simulations. However, they required a deep understanding of these governing equations, which limited their flexibility for diverse real-world problems.
  - ConvLSTM brought a hybrid approach, combining spatial feature extraction and temporal sequence modeling to improve spatiotemporal dependency handling. Despite this, it faced limitations in managing ultra-long term dynamics, constraining its use in more extended predictive tasks.
  - Transformers provided a breakthrough by handling long-term dependencies more effectively than prior architectures. However, they still encountered issues with spatial information loss, particularly when processing complex spatiotemporal data due to their reliance on single-scale processing.
- **Generation 3** featured the development of Deep Generative Models, offering unprecedented capabilities for handling intricate spatiotemporal patterns and uncertainties. Despite their promise, challenges related to computational expense and the complexity of the models persisted, hindering their application in real-time scenarios.
- **Generation 4** emerged to tackle these remaining obstacles. In this context, we introduce DynamicGPT, a model built upon a Vision Transformer (ViT)-based architecture that preserves multi-scale spatial features and effectively models long-term temporal dependencies. This design strikes an optimal balance between high-accuracy predictions and real-time feasibility.

In this supplementary information, we will provide an in-depth look at the mechanisms of DynamicGPT, including its inference process and architectural components. Additionally, we will outline how we selected and utilized baseline models to benchmark DynamicGPT's performance in a fair and rigorous manner.

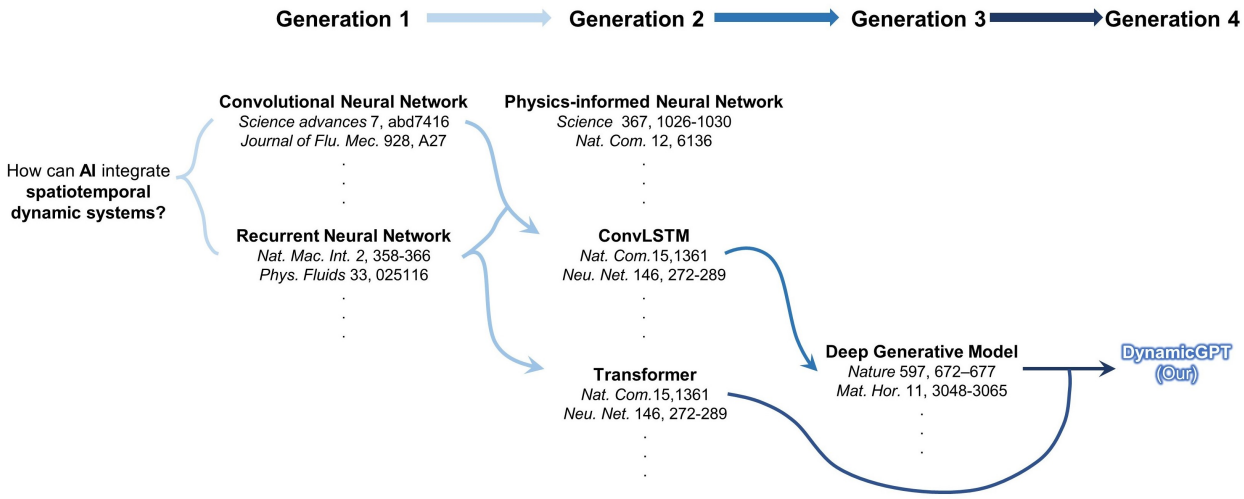


Figure S1: Evolution of neural network architectures for integrating spatiotemporal dynamic systems, progressing from basic models (Generation 1) to advanced models like Deep Generative Models and DynamicGPT (Generation 4).

B DynamicGPT

In this section, we provide detailed insights aimed at reinforcing the explanations presented in **DynamicGPT Implementation Details and Training Process** of the **Main Text**. We describe the data preprocessing required for training and inference in DynamicGPT, delve into the architecture's key mechanisms, and explain its training strategy, including the rationale behind specific design choices. Additionally, we outline the optimized hyperparameters used for validating DynamicGPT on real-world problem datasets, adding further depth to understanding its performance and applicability.

B.1 DynamicGPT Data process

As illustrated in **Figure S2**, DynamicGPT's training strategy begins with processing spatiotemporal data using a sliding window approach. The input consists of past dynamic fields with a length $\boldsymbol{T}$, and the target is the subsequent frame. This preprocessing method effectively augments the dataset by sliding the window frame-by-frame, ensuring that the model learns from various temporal contexts across the entire dataset. Once trained, DynamicGPT iteratively predicts frames, where each predicted frame is fed back into the model as input for subsequent predictions. This process allows for accurate long-term spatiotemporal forecasting, providing continuity and consistency over extended periods.

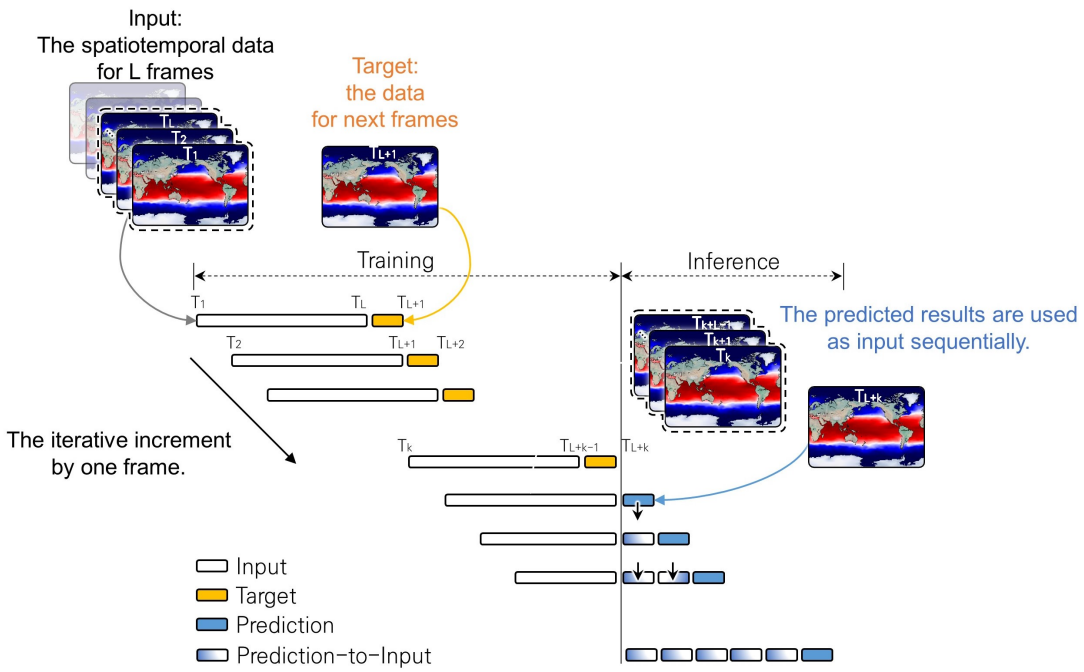


Figure S2: The process of training and inference in DynamicGPT. The input consists of spatiotemporal data for 1-L frames, where the target is the data for the next frame L+1. During training, the model learns to predict the target from the input frames. In the inference phase, the predicted results are used sequentially as input to forecast subsequent frames, demonstrating the iterative increment by one frame.

B.2 DynamicGPT Architecture

DynamicGPT's architecture features powerful components that address the limitations of previous **generations 2** and **3**.

- **First Key Mechanism – Multi-spatial Feature Extraction**: As seen in **Figure S3**, the first step of DynamicGPT involves extracting comprehensive spatial features from preprocessed spatiotemporal data pairs using a multi-kernel encoder. The encoder applies multiple kernel sizes (e.g., 2x2, 4x4, 8x8) to capture both fine local details and broader global patterns. This multi-scale approach ensures that the model maintains rich spatial information throughout the feature extraction process. For each input X and kernel F_k , the feature maps F_k are generated as follows (Eq. 1):

$$F_k = X * K_k \quad \forall k \in \{2, 4, 8\} \tag{1}$$

The extracted features F_2, F_4, F_8 are fused through a 1x1 convolutional layer, which combines them efficiently while reducing learning parameters (Eq. 2):

$$F_{\text{fused}} = \sigma\left(W_{\text{fusion}} \cdot [F_2; F_4; F_8]\right) \tag{2}$$

where W_{fusion} denotes the weights of the 1×1 convolution, σ is a non-linear activation function (e.g., ReLU), and $[\cdot]$ represents concatenation of feature maps. This convolution operation not only enhances model performance but also maintains computational efficiency. Next, the resulting latent patches are then processed with a variational sampling technique, modeling the inherent uncertainty present in real-world spatiotemporal dynamics, where the mean μ and variance σ are calculated as follows:

$$\mu = W_\mu F_{\text{fused}} + b_\mu \tag{3}$$

$$\sigma = \exp\left(W_\sigma F_{\text{fused}} + b_\sigma\right) \tag{4}$$

The final latent vector z is sampled using the reparameterization trick:

$$z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{5}$$

where $\odot$ denotes element-wise multiplication. This probabilistic approach ensures that the model can generalize effectively, making the method robust in handling unseen scenarios.

- **Second Key Mechanism – Temporal Modeling with Latent Patches**: **Figure S4** illustrates how DynamicGPT enhances temporal modeling using the latent patches produced by the multi-kernel encoder. Unlike Generations 2 and 3, which struggled to preserve spatiotemporal characteristics, DynamicGPT employs a Vision Transformer (ViT) to strengthen temporal connections between latent patches. To capture varying temporal dependencies (short, medium, and long-term), a padding technique aligns latent patches of different input lengths $\boldsymbol{T}$. The core of this ViT is the Multi-Head Self-Attention (MHSA) mechanism, which allows the model to focus on different temporal aspects by processing each latent patch independently across multiple attention heads.

For each latent patch z_i, linear projections are created to form the query (Q), key (K), and value (V) matrices:

$$Q_i = W_Q z_i, \quad K_i = W_K z_i, \quad V_i = W_V z_i \tag{6}$$

where W_Q, W_K, and W_V are learned weights. The attention score for each head is computed as a scaled dot-product between queries and keys, followed by a softmax operation to ensure probabilities:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{7}$$

Here, d_k is the dimension of the key vectors, used to scale the dot-product for stable gradient updates. The outputs from each attention head are concatenated and passed through a final linear layer to integrate the multi-head outputs:

$$\text{MHSA\_output} = W_O \cdot [\text{head}_1, \text{head}_2, \ldots, \text{head}_h] \tag{8}$$

where W_O is the learned weight of the output projection. This MHSA mechanism enables the model to capture short, medium, and long-term temporal dependencies, effectively maintaining temporal integrity across varying input lengths $\boldsymbol{T}$.

Furthermore, to enhance the model's temporal modeling capability, DynamicGPT incorporates a ConvGRU module, which is integrated into the multi-scale temporal network. This ConvGRU helps capture sequential dependencies more effectively within the temporal sequence, reinforcing the latent patch representations. The ConvGRU operates as follows:

- Update Gate:

$$z_t = \sigma\left(W_{xz} * z_{t-1} + U_{hz} * h_{t-1} + b_z\right) \tag{9}$$

- Reset Gate:

$$r_t = \sigma\left(W_{xr} * z_{t-1} + U_{hr} * h_{t-1} + b_r\right) \tag{10}$$

- New Memory Content:

$$\tilde{h}_t = \tanh\left(W_{x\tilde{h}} * z_{t-1} + r_t \odot (U_{h\tilde{h}} * h_{t-1}) + b_{\tilde{h}}\right) \tag{11}$$

- Final Output:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{12}$$

where h_t represents the final output of the ConvGRU, reinforcing the temporal dependencies among latent patches. This combined approach enables DynamicGPT to model complex temporal sequences while preserving spatial integrity and incorporating uncertainty—a significant advancement over previous approaches.

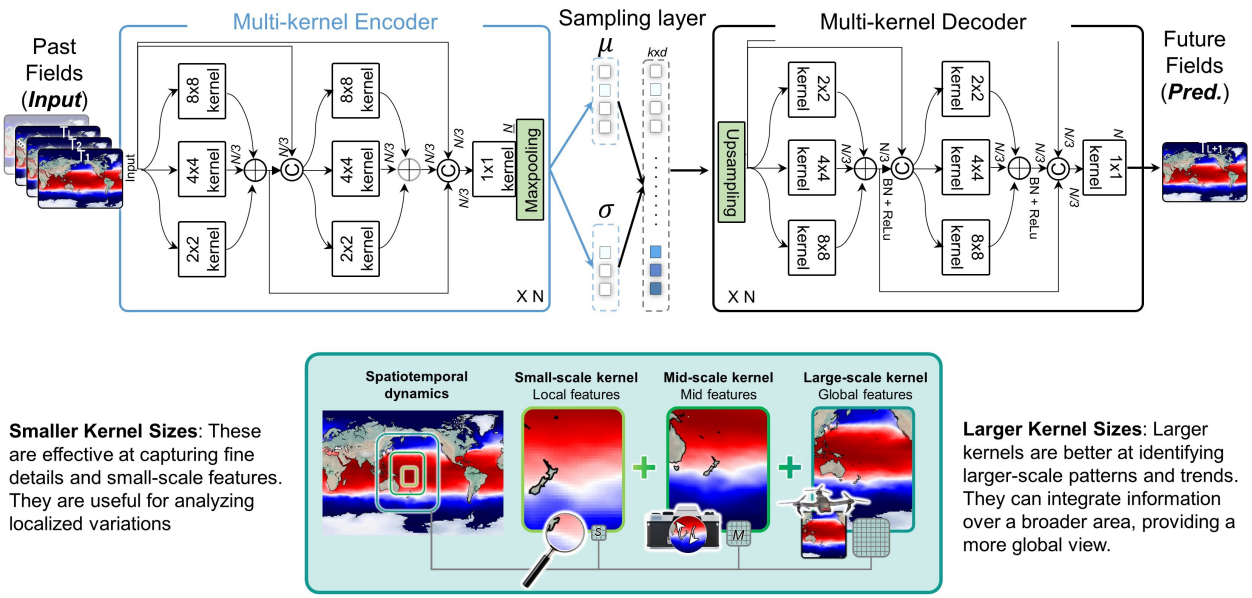


Figure S3: Architecture of the Multi-Kernel Encoder-Decoder model for spatiotemporal data processing. The model utilizes various kernel sizes to capture features at different scales: smaller kernels for local features, mid-scale kernels for intermediate features, and larger kernels for global patterns. The encoding process analyzes past fields (input) to derive latent representations, while the decoding process predicts future fields. This architecture enhances the model's ability to integrate detailed and large-scale information effectively.

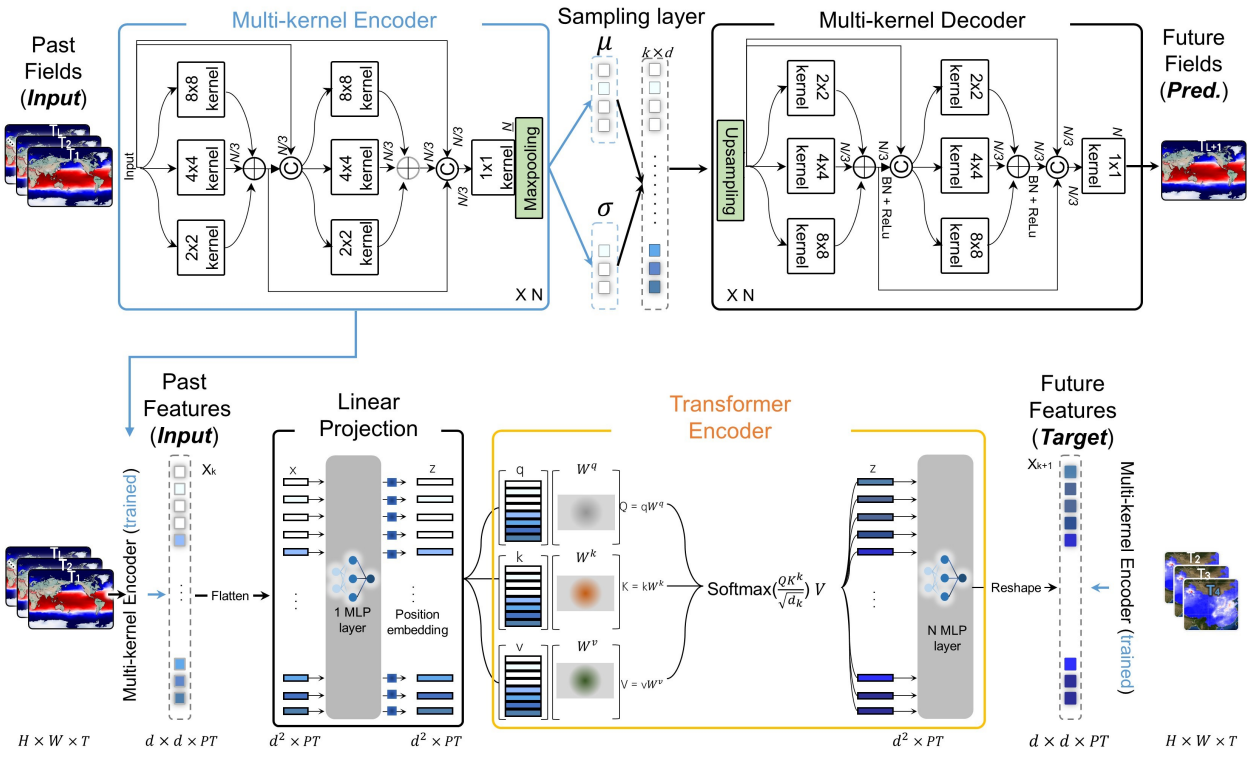


Figure S4: Architecture of the Multi-Kernel Encoder-Decoder model with Transformer integration for spatiotemporal data processing. The model consists of a multi-kernel encoder that extracts features from past fields (input) using various kernel sizes, followed by a sampling layer that captures latent representations. The Transformer encoder processes these features through a linear projection, allowing for effective learning of relationships. The multi-kernel decoder then generates future fields (predictions) based on the encoded information, enhancing the model's ability to predict future dynamics.

B.3 DynamicGPT Training Strategy

As shown in **Figure S5**, DynamicGPT's training employs a split network training approach. This approach separates the training of the multi-kernel encoder and the temporal modeling network, addressing the challenge of scale differences between components that can complicate convergence and hinder learning efficiency. By training these components independently, DynamicGPT achieves better convergence rates and optimizes parameter learning more effectively. This strategy is grounded in the need for stability during training and the goal of utilizing model parameters more efficiently. In other words, this split training method is not only practical computaional efficiency but also essential for maximizing model performance.

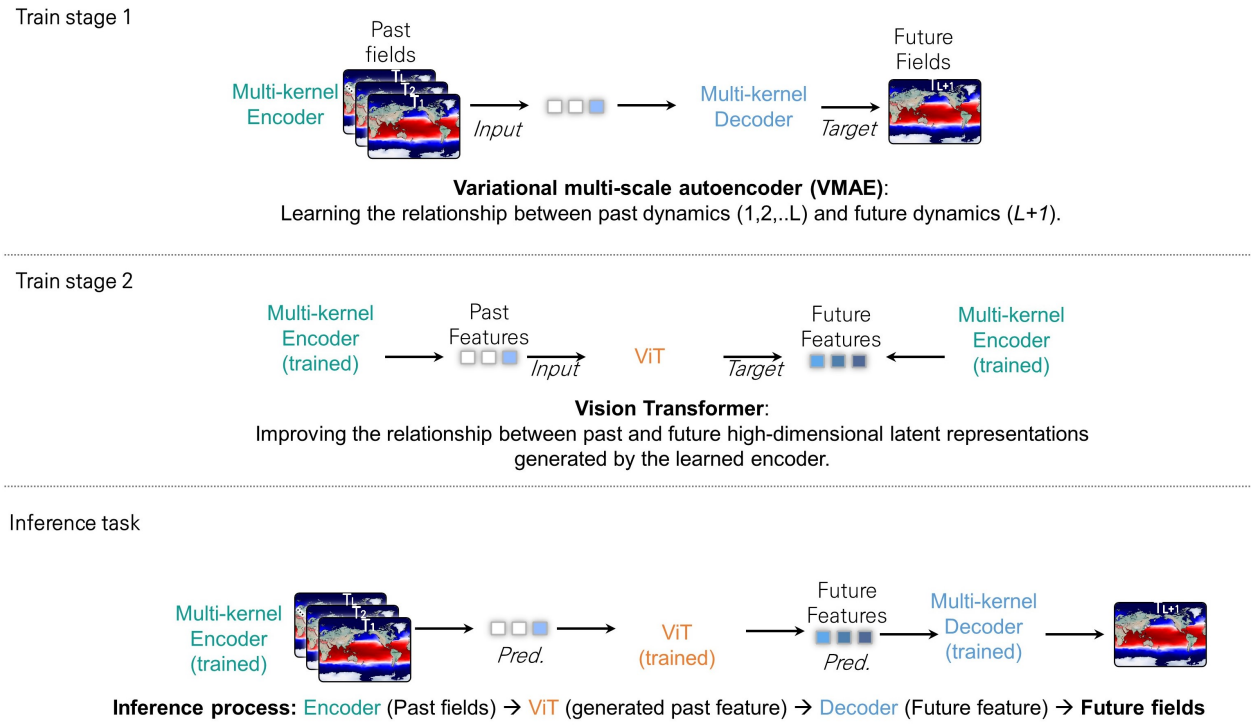


Figure S5: Overview of the training stages and inference process in the model architecture. Train Stage 1 involves the Variational Multi-Scale Autoencoder (VMAE), which learns the relationship between past dynamics (T_1, T_2, ..., T_L) and future dynamics (T_{L+1}) using a multi-kernel encoder and decoder. Train Stage 2 utilizes the Vision Transformer (ViT) to improve the relationship between high-dimensional past and future features generated by the trained multi-kernel encoder. The Inference Task illustrates how the trained model processes past fields to generate predictions, utilizing the ViT and multi-kernel decoder to output future fields.

B.4 Summary of the optimized DynamicGPT's archtectures

To provide a comprehensive understanding of how DynamicGPT apply to various real-world problems, we highlighted the impact of different hyperparameters (e.g. beta-Parameter, Latent dimension, Head layer, Kernel operation) on model performancein the section **Guidelines for Tuning DynamicGPT Across Diverse Spatiotemporal Dynamics** of the **Main Text**. In this supplementary information, we summarize the final architecture details, including the layers and feature maps, for each validation scenario (**Figure S6**). This overview is essential for demonstrating the adaptability and robustness of the model across different applications.

In the context of validating DynamicGPT, we investigated the performance across four representative real-world problems: crack propagation in materials, 3D reaction-diffusion processes, flow past a cylinder, and climate science predictions. Each scenario presents unique challenges and requires tailored network configurations to achieve optimal performance. By systematically adjusting and analyzing the hyperparameters

in the multi-spatial embedding network ($\boldsymbol{E}$), multi-scale temporal network ($\boldsymbol{D}$), multi-scale spatial decoder ($\boldsymbol{F}$), and the discriminator, we identified the configurations that yielded the most accurate predictions.

The provided supplementary figure serves as a detailed reference, illustrating how each network component and feature map size is specifically configured for the scenarios tested. For example, the multi-spatial embedding network ($\boldsymbol{E}$) is tailored with varying numbers of encoding layers and feature dimensions depending on the input data structure and problem complexity. Similarly, the multi-scale temporal network ($\boldsymbol{D}$) incorporates self-attention mechanisms and position layers to handle different temporal dependencies effectively.

By presenting this structured summary, we aim to make it clear why these particular architectural choices were made and how they contribute to the performance of DynamicGPT in diverse spatiotemporal prediction tasks. This supplementary information underscores the thorough experimentation process and provides clarity on the adjustments made for each validation dataset

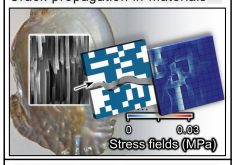
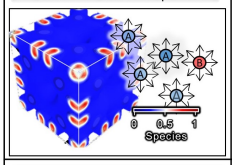
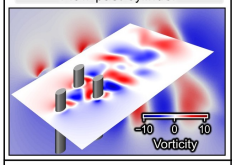
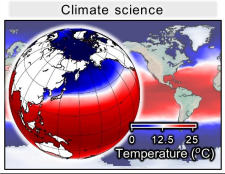
Validation Scenarios	Multi-spatial embedding network (E)	Multi-scale temporal network (D)	Multi-scale spatial decoder (F)	Discriminator
Crack propagation in materials 	*Input layer* (None, 132, 132, 3) *Encode layer 1* (None, 66, 66, 32) *Encode layer 2* (None, 33, 33, 32) *Encode layer 3* (None, 16, 16, 16)	*Input* (None, 16, 16, 16, 3) *Embed layer* (None, 3, 2048) *Position layer* (None, 3, 2048) *Self-attention* (None, 3, 2048) *Forward layer* (None, 3, 2048) *Forward layer* (None, 1, 2048) *Reshape layer* (None, 16, 16, 16)	*Decode layer 1* (None, 16, 16, 16) *Decode layer 2* (None, 33, 33, 32) *Decode layer 3* (None, 66, 66, 32) *Decode layer 4* (None, 132, 132, 32) *Output layer* (None, 132, 132, 1)	*Concatenate* (None, 132, 132, 2) *Conv2D* (None, 66, 66, 64) *Conv2D* (None, 33, 33, 128) *Conv2D* (None, 16, 16, 256) *Conv2D* (None, 15, 15, 512) *Output layer* (None, 14, 14, 1)
3D Reaction-diffusion process 	*Input layer* (None, 80, 80, 80, 30) *Encode layer 1* (None, 40, 40, 40, 64) *Encode layer 2* (None, 20, 20, 20, 32) *Encode layer 3* (None, 10, 10, 10, 16) *Encode layer 4* (None, 8, 8, 8, 16) *Average pooling* (None, 8, 8, 16)	*Input* (None, 8, 8, 16, 30) *Embed layer* (None, 30, 1024) *Position layer* (None, 30, 1024) *Self-attention* (None, 30, 1024) *Forward layer* (None, 30, 1024) *Forward layer* (None, 1, 1024) *Reshape layer* (None, 8, 8, 16)	*Dense + Reshape* (None, 8, 8, 8, 16) *Upsampling* (None, 10, 10, 10, 16) *Decode layer 1* (None, 20, 20, 20, 32) *Decode layer 2* (None, 40, 40, 40, 64) *Decode layer 3* (None, 80, 80, 80, 64) *Output layer* (None, 80, 80, 80, 1)	*Concatenate* (None, 80, 80, 80, 2) *Conv3D* (None, 40, 40, 40, 64) *Conv3D* (None, 20, 20, 20, 128) *Conv3D* (None, 10, 10, 10, 256) *Conv3D* (None, 9, 9, 9, 512) *Output layer* (None, 8, 8, 8, 1)
Flow past cylinder 	*Input layer* (None, 256, 384, 10) *Encode layer 1* (None, 128, 192, 64) *Encode layer 2* (None, 64, 96, 64) *Encode layer 3* (None, 32, 48, 64) *Encode layer 4* (None, 16, 24, 64) *Encode layer 5* (None, 8, 12, 64) *Encode layer 6* (None, 4, 4, 64)	*Input* (None, 4, 4, 64, 10) *Embed layer* (None, 10, 1024) *Position layer* (None, 10, 1024) *Self-attention* (None, 10, 1024) *Forward layer* (None, 10, 1024) *Forward layer* (None, 1, 1024) *Reshape layer* (None, 4, 4, 64)	*Decode layer 1* (None, 8, 12, 64) *Decode layer 2* (None, 16, 24, 64) *Decode layer 3* (None, 32, 48, 64) *Decode layer 4* (None, 64, 96, 64) *Decode layer 5* (None, 128, 192, 64) *Output layer* (None, 256, 384, 1)	*Concatenate* (None, 256, 384, 2) *Conv2D* (None, 128, 192, 64) *Conv2D* (None, 64, 96, 128) *Conv2D* (None, 32, 48, 256) *Conv2D* (None, 31, 47, 512) *Output layer* (None, 30, 46, 1)
Climate science 	*Input layer* (None, 180, 360, 12) *Encode layer 1* (None, 90, 180, 64) *Encode layer 2* (None, 45, 90, 64) *Encode layer 3* (None, 23, 45, 64) *Encode layer 4* (None, 12, 23, 64) *Encode layer 5* (None, 6, 12, 64) *Encode layer 6* (None, 3, 6, 64) *Encode layer 7* (None, 2, 2, 64)	*Input* (None, 2, 2, 64, 12) *Embed layer* (None, 10, 256) *Position layer* (None, 10, 256) *Self-attention* (None, 10, 256) *Forward layer* (None, 10, 256) *Forward layer* (None, 1, 256) *Reshape layer* (None, 2, 2, 64)	*Decode layer 1* (None, 3, 6, 64) *Decode layer 2* (None, 6, 12, 64) *Decode layer 3* (None, 12, 23, 64) *Decode layer 4* (None, 23, 45, 64) *Decode layer 5* (None, 45, 90, 64) *Decode layer 6* (None, 90, 180, 64) *Output layer* (None, 180, 360, 1)	*Concatenate* (None, 180, 360, 2) *Conv2D* (None, 90, 180, 64) *Conv2D* (None, 45, 90, 128) *Conv2D* (None, 23, 45, 256) *Conv2D* (None, 22, 44, 512) *Output layer* (None, 21, 43, 1)

Figure S6: DynamicGPT's architecture for various validation scenarios. This table summarizes the layers and dimensions used in the Multi-spatial Embedding Network ($\boldsymbol{E}$), Multi-scale Temporal Network ($\boldsymbol{D}$), Multi-scale Spatial Decoder ($\boldsymbol{F}$), and Discriminator across different tasks, including crack propagation in materials, the 3D reaction-diffusion process, flow past a cylinder, and climate science. Each scenario is detailed with the input and output dimensions, highlighting the complexity and design of the neural network components tailored for specific spatiotemporal challenges.

C Validation Against Baseline Models

To thoroughly assess the performance of DynamicGPT, we conducted a comprehensive comparison against established baseline models, including Physics-Informed Neural Networks (PINNs)[1], Transformers[2], and Deep Generative Models (DGMs)[3]. Each of these models has been widely utilized in spatiotemporal dynamic prediction tasks, and their selection as baselines allows for a robust evaluation of DynamicGPT's advantages.

C.1 Physics-Informed Neural Networks (PINNs)

PINNs integrate physical governing equations directly into the neural network's loss function (Left in **Figure S7**), enabling the model to enforce known physical laws during training. This mechanism allows PINNs to model spatiotemporal phenomena by embedding partial differential equations (PDEs) that describe system dynamics. The advantage of PINNs lies in their ability to maintain physical consistency and provide interpretability. However, their performance is often limited by the need for accurate mathematical descriptions of complex systems and high computational costs, which can hinder real-time prediction capabilities.

Why use PINNs as a baseline? PINNs are included as a baseline to highlight DynamicGPT's ability to achieve comparable or superior accuracy without the constraints of embedding explicit physical equations. This comparison underscores DynamicGPT's flexibility and efficiency when applied to complex systems where exact physical models are difficult to derive or computationally intensive to implement.

C.2 Transformer

Transformers, known for their self-attention mechanisms, excel at modeling long-range dependencies within sequential data. In spatiotemporal dynamic predictions, they provide a way to capture relationships across time steps while processing spatial features. A common approach involves using an autoencoder architecture to effectively reduce the high-dimensional spatiotemporal data into lower-dimensional latent vectors (Center in **Figure S7**). The encoder compresses the input data into a compact 1-dimensional vector representation, preserving essential temporal and spatial features. This dimensionality reduction is critical for making the subsequent dynamic modeling more computationally efficient. The self-attention mechanism within the Transformer then processes these 1-dimensional vectors, enabling the model to weigh different temporal relationships and capture diverse patterns. This allows for sophisticated dynamic modeling over extended sequences. However, despite these strengths, standard Transformers can suffer from spatial infor-

mation loss when dealing with high-dimensional spatiotemporal data due to their focus on 1-dimensional representations and single-scale processing.

Why use Transformers as a baseline? Including Transformers as a baseline allows us to demonstrate DynamicGPT's ability to overcome limitations related to spatial information loss. While Transformers excel at managing long-term temporal dependencies, the comparison highlights how DynamicGPT's multi-scale feature extraction and vision-based temporal modeling can lead to more accurate and detailed spatiotemporal predictions by preserving rich spatial details alongside temporal modeling.

C.3 Deep Generative Models (DGMs)

Deep Generative Models, especially those built upon conditional GANs (cGANs), are well-known for their capability to learn complex distributions and generate realistic samples. In our DGM baseline, an encoder maps the input data to a latent space representation, followed by ConvLSTM layers that model temporal dependencies. The decoder reconstructs the final spatiotemporal output from the latent space (Right in **Figure S7**). This mechanism allows for modeling uncertainty and non-linear interactions in data. However, due to the sequential structure involving encoder-ConvLSTM-decoder modules, errors can propagate through the model, impacting the stability and accuracy of predictions. In addition, the combined use of cGAN, ConvLSTM, and multiple ecoding layers leads to significant computational expense. The high complexity can hinder real-time performance, particularly with large spatiotemporal datasets.

Why use DGMs as a baseline?: DGMs are included to demonstrate DynamicGPT's advantages in managing complex spatiotemporal dynamics without the drawbacks of excessive computational cost and instability associated with generative models. This comparison showcases how DynamicGPT's structured approach, which incorporates probabilistic modeling and multi-scale feature extraction, achieves stable training and effective inference, overcoming DGM limitations.

C.4 Hyperparameter setting

For all baseline models, hyperparameters were selected based on reported optimal configurations in peer-reviewed studies:

- **PINN**: **https://github.com/stephenbaek/parc**
- **Transformer**: **https://github.com/KTH-FlowAI/beta-Variational-autoencoders-and-transformers-for-reduced-order-modelling-of-fluid-flows**

- **DGM**: https://github.com/google-deepmind/deepmind-research /tree/master/nowcasting

This ensures that the comparison remains fair and rooted in previous validation studies that have rigorously tested these architectures. Each model was tuned to achieve its best performance in similar spatiotemporal tasks, making the comparison with DynamicGPT reflective of real-world applicability.

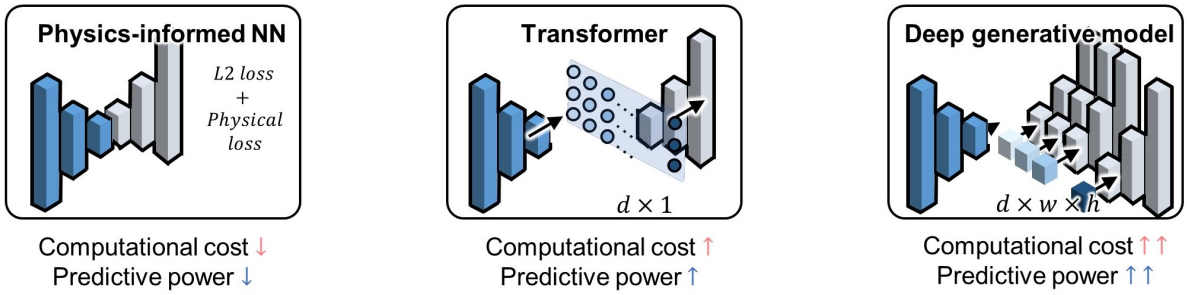


Figure S7: Baseline model architectures: Physics-informed Neural Networks (PINN), Transformer, and Deep Generative Model.

D Detailed Methods for Assessing the Physical Consistency and Computational Efficiency of DynamicGPT

In modern machine learning, evaluating the physical consistency and computational efficiency of predictive models is crucial, especially in domains that involve complex spatiotemporal dynamics. The **Performance Assessment of DynamicGPT: Physical Consistency and Computational Efficiency** section in the **Main Text** underscores the importance of these metrics and highlights DynamicGPT's superior performance compared to baseline models. This section provides a comprehensive explanation of the methods and code used for this assessment, supplementing the main text.

D.1 Methodology for Assessing Physical Consistency

Physical consistency ensures that model predictions adhere to the governing physical equations of the system. For the composite design scenario, we evaluated physical consistency by predicting the stress evolution along different directions and calculating the gradient values (∇) using finite difference methods to verify convergence to zero, satisfying the equilibrium equations of force.

To compute directional derivatives and gradients, we utilized Python libraries like Korina and Torch, which support finite difference-based differentiation. This enabled us to develop a function that calculates deviations from the force equilibrium equations for both simulation-based and model-predicted results (**Figure S8**). The following code snippet demonstrates how this function is structured:

```
def ForceEquilibriumCalculation(stress_data):
    # Input data: batch_size * output_steps * stress components * H * W
    # stress_data is the results of 'stress tensor' from a model

    sigma_xx = stress_data[:, :, 0]  # xx stress component (shape: B * T * H * W)
    sigma_yy = stress_data[:, :, 1]  # yy stress component (shape: B * T * H * W)
    sigma_xy = stress_data[:, :, 2]  # xy stress component (shape: B * T * H * W)
    sigma_yx = stress_data[:, :, 3]  # yx stress component (shape: B * T * H * W)

    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    # Convert numpy arrays to torch tensors and move to device
    sigma_xx = torch.from_numpy(sigma_xx).float().to(device)
    sigma_yy = torch.from_numpy(sigma_yy).float().to(device)
    sigma_xy = torch.from_numpy(sigma_xy).float().to(device)
    sigma_yx = torch.from_numpy(sigma_yx).float().to(device)

    # Add a batch and channel dimension -> (H, W) -> (1, 1, H, W)
    sigma_xx = sigma_xx.unsqueeze(0).unsqueeze(0)
    sigma_yy = sigma_yy.unsqueeze(0).unsqueeze(0)
    sigma_xy = sigma_xy.unsqueeze(0).unsqueeze(0)
    sigma_yx = sigma_yx.unsqueeze(0).unsqueeze(0)

    # Sobolev-function gradients (Kornia's Spatial Gradient)
    field_grad = kornia.filters.SpatialGradient()

    # Calculate gradients
    sigma_xx_grad = field_grad(sigma_xx)
    sigma_yy_grad = field_grad(sigma_yy)
    sigma_xy_grad = field_grad(sigma_xy)
    sigma_yx_grad = field_grad(sigma_yx)

    # Extract x and y components of gradients
    sigma_xx_x = sigma_xx_grad[:, :, 0]  # sigma_xx's x-gradient
    sigma_xy_y = sigma_xy_grad[:, :, 1]  # sigma_xy's y-gradient

    sigma_yy_y = sigma_yy_grad[:, :, 1]  # sigma_yy's y-gradient
    sigma_yx_x = sigma_yx_grad[:, :, 0]  # sigma_yx's x-gradient

    # Force equilibrium equations
    rho = 1.0  # Assume density is 1 for simplicity
    bx, by = 0.0, 0.0  # External forces (e.g., gravity), can be set to non-zero
        if needed

    # x-direction force equilibrium
    force_x_eq = sigma_xx_x + sigma_xy_y + rho * bx

    # y-direction force equilibrium    force_y_eq = sigma_yy_y + sigma_yx_x + rho
        * by

    # Compute mean absolute value of the violations in force equilibrium
    force_x_mean = torch.mean(torch.abs(force_x_eq)).item()
    force_y_mean = torch.mean(torch.abs(force_y_eq)).item()
    f = force_x_mean + force_y_mean
    return f
```

This method was applied to both simulation results and predictions from DynamicGPT and baseline models to assess physical consistency. To test DynamicGPT's robustness and generalizability, we evaluated its performance on 1,000 composite configurations with strength and toughness averages 1.35 and 1.42 times higher than those in the training set. This strategy addresses a common challenge where conventional models struggle to predict unseen configurations accurately, especially when trained on limited data. For the unsteady flow problem, we used a similar approach to assess mass conservation by calculating the divergence of the predicted flow field and verifying its convergence to zero (**Figure S8**). DynamicGPT was trained on a 256 × 256 spatial domain and tested on a larger 1,536 × 256 domain to assess its generalization to unseen, larger computational domains, a critical factor for real-world applications. The code snippet below outlines the divergence calculation:

```
def DivergenceCalculation(data):
    # Input data: batch_size * output_steps * flow length * H * W
    # The data is the results by 'DynamicGPT and Baseline models'

    preds_u = data[:, :, 0]  # Extract u field (shape: B * T * H * W)
    preds_v = data[:, :, 1]  # Extract v field (shape: B * T * H * W)

    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    # Convert numpy arrays to torch tensors and move to device
    u = torch.from_numpy(preds_u).float().to(device)
    v = torch.from_numpy(preds_v).float().to(device)

    # Add a batch and channel dimension -> (H, W) -> (1, 1, H, W)
    u = u.unsqueeze(0).unsqueeze(0)  # Add batch and channel dimensions
    v = v.unsqueeze(0).unsqueeze(0)  # Add batch and channel dimensions

    # Sobolev-function gradients (Kornia's Spatial Gradient)
    # Note: the operation use "Central differential method"!
    field_grad = kornia.filters.SpatialGradient()

    # Calculate gradients
    u_grad = field_grad(u)
    v_grad = field_grad(v)

    # Extract x and y components of gradients
    u_x = u_grad[:, :, 0]  # u's x-gradient
    v_y = v_grad[:, :, 1]  # v's y-gradient

    # Compute divergence (u_x + v_y)
    div = u_x + v_y

    # Convert back to numpy and compute mean of the absolute divergence
    div_mean = np.mean(np.abs(div.cpu().data.numpy()), axis=(0, 2, 3))  # Mean
        over H, W

    return div_mean
```

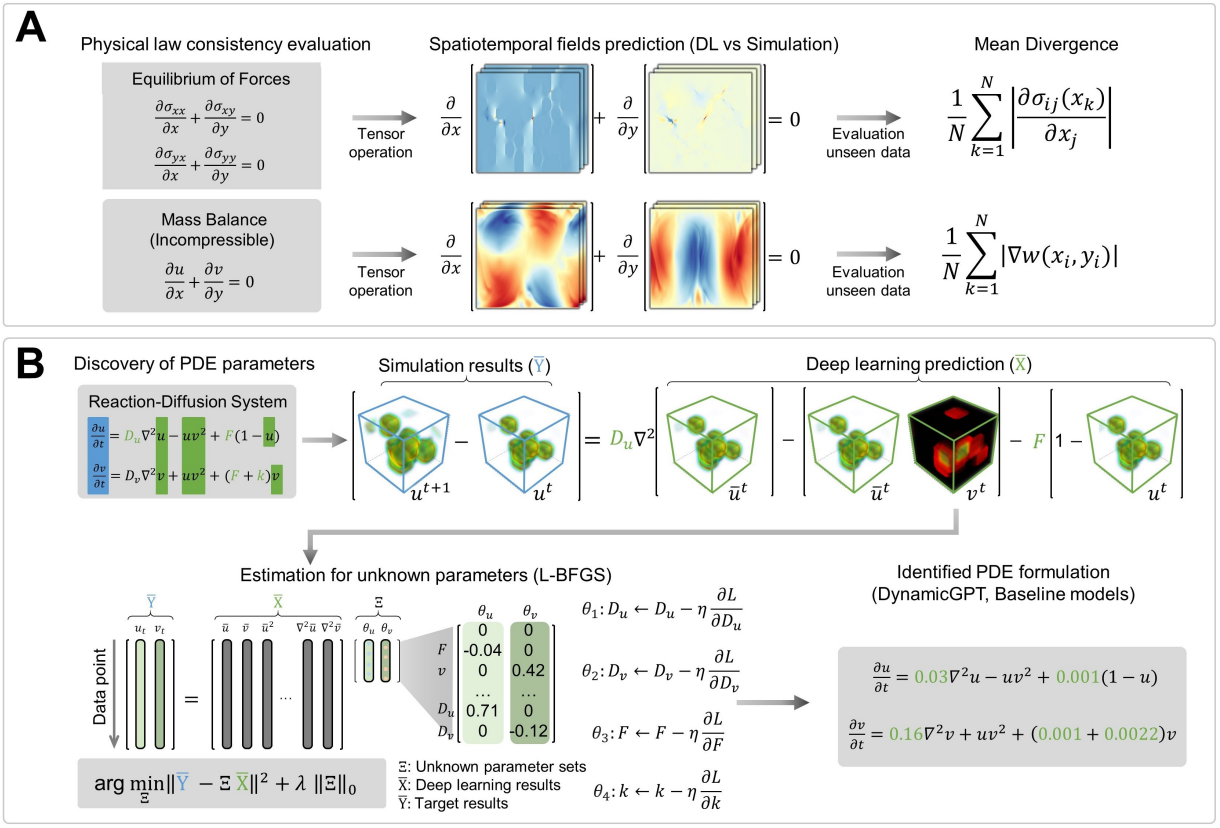


Figure S8: (**a**) Flowchart of physical Consistency evaluation, (**b**) Flowchart of PDE parameter estimation

D.2 Comprehensive Evaluation of Physical Consistency for Real-world Validation Sets

Equilibrium of Force

To evaluate the equilibrium of force, we calculated and averaged the stress evolution over 47 time steps for each model in the test set (**Supplementary Table 1**). DynamicGPT achieved a score of 0.0083, closely aligning with the simulation benchmark of 0.0072, highlighting its superior ability to maintain physical consistency in stress evolution. In comparison, conventional models such as PINN, Transformer, and DGM showed greater deviation from the simulation with scores of 0.0181, 0.0148, and 0.0125, respectively.

For a more detailed analysis, we focused on the configuration with the highest strength in the test set—representing the most challenging case and the primary objective of exploring unseen spaces in mechanical design. **Figure S9** illustrates the results for composite design, showing the equilibrium of forces over time. DynamicGPT's predictions closely follow the simulation curve, demonstrating superior physical consistency even under extreme conditions. In contrast, other models such as PINN, DGM, and Transformer exhibit more significant deviations, particularly at later time steps, indicating a loss of accuracy when deal-

ing with high-strength, unseen configurations. The ability to predict the stress evolution in configurations with unprecedented mechanical properties is crucial for advancing material design and engineering practices. These results underline the importance of having models capable of accurately extrapolating to such high-strength designs. By maintaining physical consistency and accurately modeling the long-term stress evolution, DynamicGPT proves to be not only an effective predictive tool but also a potentially transformative asset for real-world engineering, where predicting behavior under novel and extreme conditions is vital for safety and innovation. These findings suggest that while previous models can approximate physical consistency, DynamicGPT's architecture, which integrates multi-scale spatial and temporal modeling with probabilistic components, significantly enhances its ability to capture the fine details necessary for adhering to the governing force equilibrium equations.

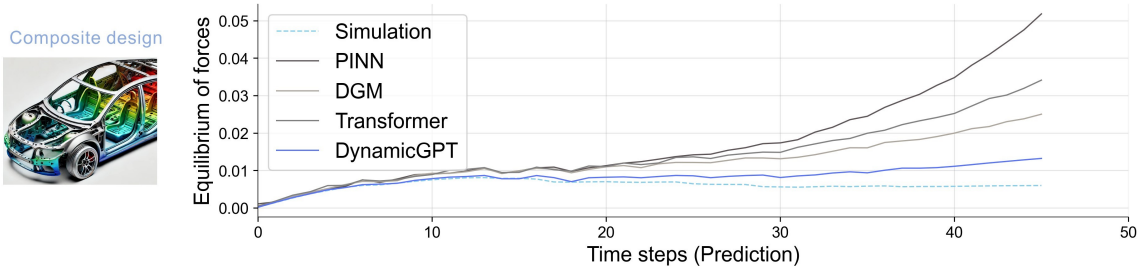


Figure S9: Comparative analysis of physical consistency in predictions made by different models. The equilibrium of forces over time steps is calculated by baseline models, DynamicGPT, and Simulation. DynamicGPT's predictions align more closely with the simulation results, showcasing its ability to maintain physical consistency even in high-strength configurations that were not present in the training data.

Mass Balance

For the mass balance assessment, DynamicGPT also demonstrates superior performance with a deviation of 20.8, compared to the benchmark simulation value of 15.5. While this result is not as close to the simulation as in the case of the force equilibrium, it is notably better than the scores of other models, such as DGM (27.5), Transformer (42.9), and PINN (36.6). This indicates that DynamicGPT is not only capable of maintaining force equilibrium but also effectively manages the conservation of mass in unsteady flow scenarios (**Figure S10**).

D.3 Comparative Analysis and Implications

The results in **Supplementary Table 1** indicate that DynamicGPT consistently outperforms baseline models in physical law adherence for both force equilibrium and mass balance. This performance is attributed to its advanced architecture, which integrates multi-scale feature extraction and temporal modeling while

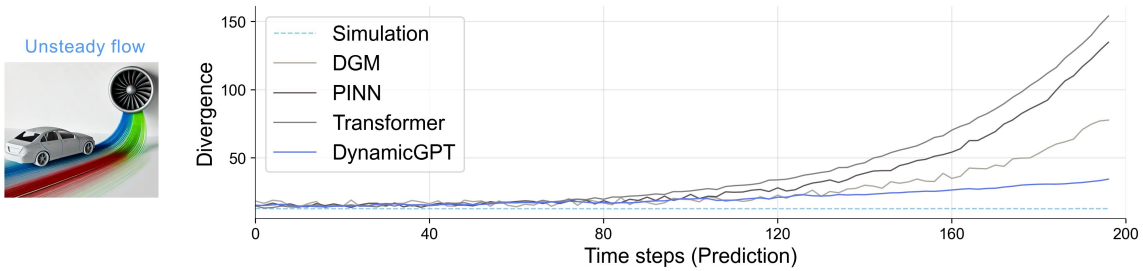


Figure S10: Comparative analysis of physical consistency in predictions made by different models. The divergence over time steps is shown for each model, including PINN, DGM, Transformer, and the proposed DynamicGPT. DynamicGPT demonstrates lower divergence and maintains closer adherence to the simulation benchmark, indicating superior mass conservation in unsteady flow predictions.

incorporating physical constraints. Unlike conventional models that struggle with extrapolating to unseen scenarios and maintaining physical consistency, DynamicGPT demonstrates a clear advantage.

These results highlight the potential of DynamicGPT to bridge the gap between traditional computational simulations and data-driven models. Achieving near-simulation-level accuracy in long-term spatiotemporal predictions, DynamicGPT showcases its capability for reliable and computationally efficient predictions, reinforcing its role as a powerful tool in applications where physical consistency is critical.

In conclusion, these findings demonstrate that DynamicGPT has been designed and optimized to overcome the limitations of previous models, maintaining physical consistency in real-world validation scenarios. This capability suggests that DynamicGPT could augment or even replace traditional simulation methods for complex physical systems, paving the way for new applications in science and engineering.

Table S1: Summary of the evaluation of physical conservation laws across different models. The table compares the performance of various models (PINN, Transformer, DGM, DynamicGPT, and Simulation) in terms of two key metrics: Equilibrium of force and Mass balance. The DynamicGPT model is highlighted in blue, showing the best performance in both metrics, with the lowest values for Equilibrium of force and Mass balance compared to the other models.

Type	PINN	Transformer	DGM	**DynamicGPT**	Simulation
Equilibrium of force	0.0181	0.0148	0.0125	**0.0083**	0.0072
Mass balance	36.6	42.9	27.5	**20.8**	15.5

D.4 Methodology for PDE Parameter Estimation

To expand on the evaluation of physical conservation laws, we employed a method for estimating PDE (Partial Differential Equation) parameters based on the results of deep learning predictions. This approach

is significant because it bridges the gap between data-driven predictions and theoretical physical models by allowing for the extraction and refinement of unknown PDE parameters, enhancing the interpretability and reliability of the model.

Process Overview

The process for PDE parameter estimation is depicted in **Figure S8(b)**, which illustrates the workflow for analyzing the reaction-diffusion system as validation scenario. The detailed process with Python code is as follow:

1. **Simulation Results (Y) and Deep Learning Predictions ($\hat{X}$)**: We compare simulation results with predictions from models like DynamicGPT. These outputs approximate time derivatives and spatial features used for parameter estimation.

2. **Deriving Time Derivatives**: The time derivatives of the predicted fields, $\frac{\partial u}{\partial t}$ and $\frac{\partial v}{\partial t}$, are calculated using finite differences based on the simulation results:

```
u_pred = np.load('Simulation_u.npy')
v_pred = np.load('Simulation_v.npy')
u_t_pred = (u_pred[:, 1:] - u_pred[:, :-1]) / dt
v_t_pred = (v_pred[:, 1:] - v_pred[:, :-1]) / dt
```

3. **Initial Parameter Setup**: Initial guesses for PDE parameters are established and refined during training:

```
ini1 = np.random.uniform(-1, 1)
ini2 = np.random.uniform(-1, 1)
ini3 = np.random.uniform(-1, 1)
ini4 = np.random.uniform(-1, 1)

D_U = torch.tensor(ini1, requires_grad=True)
D_V = torch.tensor(ini2, requires_grad=True)
F = torch.tensor(ini3, requires_grad=True)
k = torch.tensor(ini4, requires_grad=True)
```

4. **Optimization Strategy**: We use an optimizer, such as LBFGS, for parameter updates:

```
optimizer = torch.optim.LBFGS([D_U, D_V, F, k], lr=1e-4)
```

5. **Laplacian Calculation**: The Laplacian operator for 3D spatial data is defined:

```
def laplacian_3d(U):
    return (
        -6 * U +
        torch.roll(U, 1, dims=2) + torch.roll(U, -1, dims=2) +
        torch.roll(U, 1, dims=3) + torch.roll(U, -1, dims=3) +
        torch.roll(U, 1, dims=4) + torch.roll(U, -1, dims=4)
    )
```

6. **Loss Function Definition**: The loss function compares predicted time derivatives (Right side of equation) with PDE-driven time derivatives (Left side of equation):

```
def gray_scott_loss_multi(U_pred, V_pred, U_t_pred, V_t_pred, D_U, D_V, F, k,
    lambda_reg=1e-3):
    loss_total = 0
    for i in range(U_pred.shape[0]):
        lap_U = laplacian_3d(U_pred[i:i+1])
        lap_V = laplacian_3d(V_pred[i:i+1])
        lap_U = lap_U[:, :-1]
        lap_V = lap_V[:, :-1]

        U_eqn = D_U * lap_U - U_pred[i:i+1, :-1] * V_pred[i:i+1, :-1]**2 + F *
            (1 - U_pred[i:i+1, :-1])
        V_eqn = D_V * lap_V + U_pred[i:i+1, :-1] * V_pred[i:i+1, :-1]**2 - (F
            + k) * V_pred[i:i+1, :-1]

        U_t_pred_sliced = U_t_pred[i:i+1, :-1]
        V_t_pred_sliced = V_t_pred[i:i+1, :-1]

        loss_data = torch.mean((U_t_pred_sliced - U_eqn)**2) + torch.mean((
            V_t_pred_sliced - V_eqn)**2)
        loss_total += loss_data

    loss_total /= U_pred.shape[0]
    return loss_total
```

7. **Training Loop**: The training loop iteratively updates parameters to minimize the loss. The learning rate are the number of update are 1e-04 and 1,200, respectively:

```
for epoch in range(1200):
    optimizer.zero_grad()
    loss = gray_scott_loss_multi(u_pred[:, :-1], v_pred[:, :-1], u_t_pred,
        v_t_pred, D_U, D_V, F, k)
    loss.backward()
    optimizer.step()

    if epoch % 2 == 0:
        print(f'Epoch␣{epoch},␣Loss:␣{loss.item():.6f}')
        print(f'Estimated␣D_U:␣{D_U.item():.6f},␣Actual␣D_U:␣{actual_DU}')
        print(f'Estimated␣D_V:␣{D_V.item():.6f},␣Actual␣D_V:␣{actual_DV}')
        print(f'Estimated␣F:␣{F.item():.6f},␣Actual␣F:␣{actual_F}')
        print(f'Estimated␣k:␣{k.item():.6f},␣Actual␣k:␣{actual_k}')
        print('-'*50)
```

D.5 Detailed Results on PDE Parameter Estimation

Figure S11 illustrates the comparative results of parameter estimation for the reaction-diffusion system across different models: PINN, DGM, and DynamicGPT. Each subplot represents the estimated values of the diffusion coefficients D_u and D_v, feed rate F, and decay rate k over training epochs.

1. D_u **Estimation**: DynamicGPT (blue line) demonstrates rapid convergence towards the target coefficient (dashed line), achieving stability and accurate estimation earlier than both PINN and DGM.

While DGM shows significant fluctuations and a slower approach, PINN struggles to align with the true value, indicating less robust estimation in this parameter.

2. D_v **Estimation**: For the diffusion coefficient D_v, DynamicGPT again shows a more consistent trajectory towards the target compared to other models. PINN exhibits considerable instability, and DGM, although it approaches the target value, does so with a more erratic pattern. This highlights DynamicGPT's ability to estimate this parameter with greater precision and reliability.

3. F **Estimation**: In the estimation of the feed rate F, DynamicGPT displays smooth convergence and aligns closely with the actual target value after the initial training phase. DGM reaches a closer estimate but with higher variance, while PINN diverges away from the target, showing its limitations in accurately estimating this parameter.

4. k **Estimation**: The decay rate k further underscores DynamicGPT's performance advantage. The model converges effectively and stabilizes near the target coefficient. In contrast, DGM initially oscillates significantly before slowly settling, and PINN demonstrates the least effective estimation, deviating throughout the training.

These comparative results underscore DynamicGPT's strength in reliably estimating reaction-diffusion parameters, maintaining accuracy across long-term spatiotemporal dynamic. The superior performance of DynamicGPT over PINN and DGM emphasizes its robustness and precision, positioning it as a powerful tool for predictive modeling and parameter discovery in complex spatiotemporal systems.

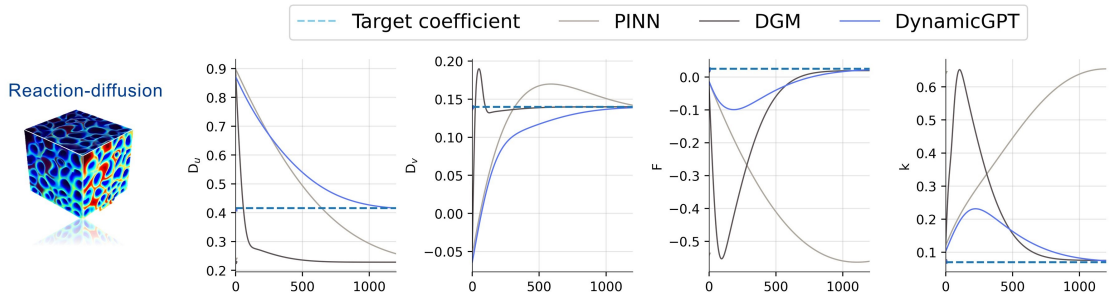


Figure S11: Comparative analysis of physical consistency in predictions made by different models. The divergence over time steps is shown for each model, including PINN, DGM, Transformer, and the proposed DynamicGPT. DynamicGPT demonstrates lower divergence and maintains closer adherence to the simulation benchmark, indicating superior mass conservation in unsteady flow predictions.

Table S2: Summary of the results of parameter estimation for partial differential equations (PDEs). The table compares the estimated values of four key components (D_u, D_v, F, and k) obtained using three different models: PINN, DGM, and DynamicGPT. The true values of the unknown parameters are also included for comparison. Notably, the results obtained with DynamicGPT, highlighted in blue, show the closest estimates to the actual unknown parameter values.

Component	D_u	D_v	F	k
PINN	0.237	0.137	-0.54	0.0642
DGM	0.228	0.02	0.0753	0.075
DynamicGPT	**0.411**	**0.14**	**0.0243**	**0.071**
Unknown Parameters	**0.416**	**0.14**	**0.025**	**0.07**

E References

[1] Nguyen, P.C.H. et al. PARC: Physics-aware recurrent convolutional neural networks to assimilate meso scale reactive mechanics of energetic materials. Sci. Adv. 9, eadd6868 (2023).

[2] Solera-Rico, A. et al. β-Variational autoencoders and transformers for reduced-order modelling of fluid flows. Nat. Commun. 15, 1361 (2024).

[3] Ravuri, S. et al. Skilful precipitation nowcasting using deep generative models of radar. Nature 597, 672–677 (2021).