Knowledge-Based Systems 316 (2025) 113347

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Enhancing spatiotemporal prediction through the integration of Mamba state space models and Diffusion Transformers

Hansheng Zeng [a], Yuqi Li [b], Ruize Niu [c], Chuanguang Yang [b], Shiping Wen [d],*

[a] *The University of Hong Kong, Hong Kong*
[b] *Institute of Computing Technology, Chinese Academy of Sciences, China*
[c] *Sun Yat-sen University, China*
[d] *Australian AI Institute, Faculty of Engineering and Information Technology, University of Technology, Australia*

ARTICLE INFO

Keywords:
Deep learning
Spatio-temporal prediction
Mamba
Diffusion

ABSTRACT

This paper presents an advanced architecture for spatiotemporal prediction **MAD**, integrating **Ma**mba modules with **D**iffusion Transformers for efficient spatiotemporal modeling. The model consists of three phases: encoding, reconstruction, and prediction. Initially, the encoder transforms raw spatiotemporal data into compact latent embeddings. In the reconstruction phase, the Mamba module processes these embeddings through normalization and bidirectional state space models, generating reconstructed representations which are then decoded to restore the input data. The prediction phase utilizes the Diffusion Transformer to model spatiotemporal features, incorporating time embeddings and leveraging self-attention mechanisms to capture complex spatiotemporal dependencies. Finally, the model jointly trains the reconstruction and prediction paths to achieve high-precision spatiotemporal forecasts. Experimental results demonstrate the model's superior performance across various spatiotemporal prediction tasks, validating its effectiveness and robustness. Our codes are available at https://github.com/Hanson1331/KBS-MAD.

1. Introduction

Spatio-temporal prediction is a key research area in data science and artificial intelligence [1–3], widely applied in weather forecasting [4,5], traffic flow prediction [6], environmental monitoring [7], and urban planning. With the rapid development of the Internet of Things and big data technologies, massive Spatio-temporal data continuously emerge, making efficient and accurate modeling and prediction critical challenges. Spatio-temporal prediction requires handling complex dependencies across spatial and temporal dimensions, as well as high-dimensional and dynamic data characteristics, demanding sophisticated model design and optimization.

Traditional Spatio-temporal prediction methods mainly rely on statistical models, such as Autoregressive Integrated Moving Average (ARIMA) [8] and Seasonal ARIMA (SARIMA) [9], which effectively handle linear Spatio-temporal dependencies. However, as data complexity increases, these models exhibit limited prediction accuracy and generalization capabilities in nonlinear and dynamic Spatio-temporal environments. To overcome these limitations, researchers have introduced machine learning methods like Support Vector Machines (SVM) [10] and Random Forests [11], which capture complex Spatio-temporal patterns but still face challenges with computational efficiency and model complexity when dealing with large-scale high-dimensional data.

Recently, the rapid advancement of deep learning technologies has opened new opportunities for Spatio-temporal prediction. Neural network-based methods, especially the combination of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) [2,12–14], significantly enhance the ability to capture spatial features and temporal dependencies. For instance, CNNs excel at extracting local spatial features, while RNNs effectively capture temporal dependencies in sequential data. This combination performs well in tasks like video data and traffic flow prediction. However, existing methods still face challenges when handling high-dimensional Spatio-temporal data, including low computational efficiency, high model complexity, and insufficient capture of long-range Spatio-temporal dependencies. Additionally, effectively integrating Spatio-temporal features and balancing reconstruction and prediction tasks during model training remain pressing issues.

* Corresponding author.
E-mail address: shiping.wen@uts.edu.au (S. Wen).

https://doi.org/10.1016/j.knosys.2025.113347
Received 27 December 2024; Received in revised form 3 March 2025; Accepted 13 March 2025
Available online 24 March 2025

Recent research has made significant progress in Spatio-temporal prediction. For example, Wu et al. propose PastNet [15], which introduces physical inductive biases to enhance video Spatio-temporal prediction by combining physical constraints with deep learning models, thereby improving the understanding of physical dynamics. Wu et al. also present Earthfarseer [6], demonstrating the ability to model various Spatio-temporal dynamical systems within a single model, significantly increasing its versatility and adaptability. K. Wang et al. introduce Neural Discrete Learning and Levels-of-Experts [16], which utilizes neural discrete learning and hierarchical expert mechanisms to enhance the accuracy and efficiency of Spatio-temporal dynamical system modeling. Additionally, Wu et al. study Spatio-temporal Fluid Dynamics Modeling [17], combining physical awareness with parameter diffusion guidance to further improve the accuracy and robustness of fluid dynamics Spatio-temporal modeling.

Despite these advances, several challenges remain. Firstly, existing methods consume substantial computational resources when handling high-dimensional Spatio-temporal data, limiting their practical applications. Secondly, many models inadequately capture long-range Spatio-temporal dependencies, restricting their effectiveness in complex environments. Lastly, effectively integrating reconstruction and prediction tasks during model training to achieve multi-task optimization remains a challenging research topic.

To address these issues, this paper proposes an advanced Spatio-temporal prediction architecture—**MAD** (integration of **Ma**mba modules and **D**iffusion Transformer). The MAD architecture combines Mamba modules and Diffusion Transformers to achieve efficient Spatio-temporal modeling and precise prediction. The model comprises three stages: encoding, reconstruction, and prediction. In the encoding stage, the encoder transforms raw Spatio-temporal data into compact latent embeddings, effectively reducing data dimensionality and extracting key features. In the reconstruction stage, the Mamba module processes the latent embeddings through normalization and bidirectional state space models to generate reconstructed representations, which the decoder uses to restore the input data. This process enhances the model's understanding of data distribution and provides robust features for the prediction stage.

In the prediction stage, the Diffusion Transformer models complex Spatio-temporal features. This module uses time embeddings and self-attention mechanisms to capture long-range Spatio-temporal dependencies, improving the model's ability to recognize complex Spatio-temporal patterns. Additionally, the MAD model jointly trains the reconstruction and prediction paths, achieving collaborative optimization of both tasks, thereby maintaining data reconstruction quality and significantly enhancing prediction accuracy.

Experimental results show that the MAD model outperforms existing mainstream methods in various Spatio-temporal prediction tasks, validating its effectiveness and robustness in Spatio-temporal modeling and prediction. Specifically, the MAD model demonstrates superior prediction accuracy, computational efficiency, and the ability to capture complex Spatio-temporal dependencies, showcasing its broad application potential and research value.

In summary, this paper introduces the MAD architecture, which integrates Mamba modules and Diffusion Transformers to address key challenges in Spatio-temporal prediction. The main contributions include: *(1)* designing an efficient Spatio-temporal encoding and reconstruction mechanism that improves the compactness and expressiveness of data representations; *(2)* introducing the Diffusion Transformer to enhance the modeling of complex Spatio-temporal dependencies; *(3)* achieving collaborative optimization of reconstruction and prediction tasks through joint training, significantly improving prediction performance. This research provides new insights and methods for the field of Spatio-temporal prediction and offers theoretical support and technical assurance for practical applications.

2. Related work

Spatiotemporal prediction is a key research area that spans a wide range of techniques, from traditional statistical methods to advanced deep learning models. To provide a comprehensive understanding of the context of this study, we categorize related work into three core sections: traditional spatiotemporal prediction methods, deep learning-based spatiotemporal models, and the application of state-space and diffusion models in spatiotemporal prediction.

Traditional Spatiotemporal Prediction Methods. Traditional spatiotemporal prediction methods rely on statistical and classical machine learning algorithms, achieving significant success in early data analysis. Common methods include **Auto-Regressive Integrated Moving Average (ARIMA)** for time series modeling and **Kriging** for spatial data interpolation and prediction [18–20]. ARIMA effectively captures trends and periodicity in time series data but struggles with nonlinear and complex dependencies due to its linear assumptions [8,21]. Kriging leverages spatial correlations for interpolation, suitable for geospatial data, but faces challenges in high-dimensional and dynamic spatiotemporal data due to high computational complexity. Other methods, such as **Vector Auto-Regressive (VAR)** [22] and **Spatial Auto-Regressive (SAR)** models [23], handle multivariate spatiotemporal prediction by considering inter-variable influences. Additionally, spatial econometric techniques provide robust frameworks for modeling spatial dependencies [24]. However, these models encounter difficulties in parameter estimation and complexity. Overall, traditional methods perform well in simple and linear scenarios but show limitations in large-scale, high-dimensional, and nonlinear spatiotemporal data.

Deep Learning-Based Spatiotemporal Prediction Models. The rapid development of deep learning has significantly improved spatiotemporal prediction. **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)** were among the earliest models applied in this domain. CNNs excel at capturing spatial features through convolution operations, while RNNs, especially **Long Short-Term Memory (LSTM)** networks, effectively handle temporal dependencies in time series data. However, both CNNs and RNNs face limitations such as vanishing gradients and inefficiency in modeling long-term and complex dependencies [25]. Recent advances in **Transformer architectures**, with their powerful self-attention mechanisms, have shown outstanding performance in spatiotemporal prediction. Transformers effectively capture long-range dependencies and handle large-scale, high-dimensional data. By integrating **Graph Neural Networks (GNNs)**, such as **Graph Convolutional Networks (GCNs)**, Transformers further improve modeling of complex spatial relationships [26]. Cutting-edge models like **Graph Transformers** and **Spatiotemporal Graph Transformers** enhance spatial and temporal dependency modeling by incorporating graph structures. Techniques like **self-supervised learning** and **multi-task learning** have further enhanced the generalization and robustness of deep learning-based spatiotemporal models [27].

State-Space and Diffusion Models in Spatiotemporal Prediction. State-space models (SSMs) and **Diffusion Models** have shown great potential in spatiotemporal prediction. SSMs model system dynamics, capturing latent structures and temporal dynamics. Methods like the **Kalman Filter** perform well in estimating system states under linear and Gaussian noise assumptions. However, traditional SSMs struggle with nonlinear and non-Gaussian noise, limiting their application in complex spatiotemporal data [28]. Diffusion models leverage gradual generation processes and reverse diffusion to produce high-quality predictions, enhancing model expressiveness and accuracy. These models have achieved remarkable success in generation tasks and are being adapted for spatiotemporal prediction [29]. **Diffusion Transformers** combine the strengths of diffusion models and Transformers, capturing complex dependencies and offering strong generative capabilities [27,30]. The **Mamba module**, an efficient state-space modeling method, incorporates bidirectional convolutions and self-attention mechanisms,

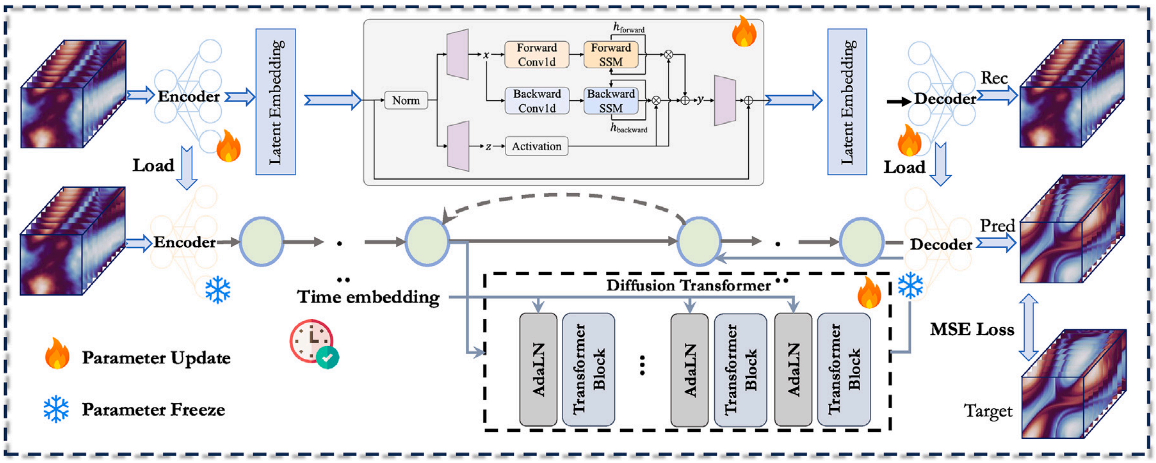

Fig. 1. The proposed spatiotemporal prediction model architecture integrating Mamba modules and Diffusion Transformers. The model consists of three stages: encoding, reconstruction, and prediction. The encoder extracts features from raw spatiotemporal data. In the reconstruction stage, the Mamba module processes latent representations using bidirectional state-space modeling to reconstruct the input data. The prediction stage leverages the Diffusion Transformer to model spatiotemporal dependencies and generate future predictions. The model employs collaborative training of reconstruction and prediction tasks to improve prediction accuracy and robustness.

improving the modeling of bidirectional dependencies in spatiotemporal data [27]. Recent studies also explore the use of **self-supervised learning** and **multi-task learning** in SSMs and diffusion models. These approaches share feature representations and optimize tasks collaboratively, enhancing generalization and robustness. For example, reconstruction tasks assist feature learning, improving prediction accuracy and robustness against noise and anomalies [31].

In summary, traditional spatiotemporal prediction methods perform well in simple scenarios but face significant limitations in handling high-dimensional, nonlinear, and complex dependencies. Deep learning-based models, leveraging CNNs, RNNs, and Transformers, have significantly improved prediction performance but still face challenges in efficiency, scalability, and stability. The combination of state-space and diffusion models offers new solutions for spatiotemporal prediction by enabling efficient dynamic modeling and strong generative capabilities. To address existing limitations, this study proposes a spatiotemporal prediction model integrating **Mamba modules** with **Diffusion Transformers**, aiming to achieve high-accuracy spatiotemporal prediction through efficient state-space modeling and enhanced feature capturing (see Fig. 1).

3. Methodology

This section introduces the architecture and implementation of the proposed spatiotemporal prediction model, which integrates Mamba modules and Diffusion Transformers. We first define the spatiotemporal prediction problem and then explain each component of the model in detail, supported by comprehensive mathematical formulations.

3.1. Problem definition

Spatiotemporal prediction aims to forecast future spatial states based on historical spatiotemporal data. Formally, given a dataset $\mathcal{D} = \{X_t\}_{t=1}^{T}$, where $X_t \in \mathbb{R}^{N\times F}$ represents spatial data at time step t, N is the number of spatial nodes, and F is the feature dimension of each node, the objective is to design a function f such that:

$$\hat{X}_{T+\Delta t} = f\left(X_{T-\tau+1}, X_{T-\tau+2}, \dots, X_T\right) \tag{1}$$

Here, Δt denotes the prediction horizon, and τ represents the length of the historical window used for forecasting. The function f encapsulates the model's ability to capture and extrapolate the underlying spatiotemporal dependencies present in the data.

3.2. Model architecture

The proposed model comprises three main stages: Encoding, Reconstruction, and Prediction. Additionally, it incorporates Parameter Management and Collaborative Training mechanisms to enhance stability and prediction accuracy. The overall workflow is outlined as follows:

1. **Encoding Stage**: Transforms raw spatiotemporal data into latent representations.
2. **Reconstruction Stage**: Processes the latent representations with Mamba modules to generate reconstructed representations and recover input data.
3. **Prediction Stage**: Models spatiotemporal dependencies in the latent space using Diffusion Transformers to produce predictions.
4. **Parameter Management**: Manages model parameters to ensure stability and efficient training.
5. **Collaborative Training**: Jointly trains reconstruction and prediction tasks to enhance feature learning and prediction performance.

3.3. Encoding stage

The Encoding Stage is responsible for mapping high-dimensional spatiotemporal data into a lower-dimensional latent space, facilitating efficient processing and feature extraction. Let $\mathcal{E}$ denote the encoder, which can be implemented using Convolutional Neural Networks (CNNs) or Graph Convolutional Networks (GCNs) to effectively capture spatial dependencies. The encoding process is defined as:

$$Z_t = \mathcal{E}(X_t), \quad Z_t \in \mathbb{R}^{N\times d} \tag{2}$$

Here, d is the dimension of the latent representation. Separate encoders are employed for reconstruction and prediction tasks to allow specialized feature extraction:

$$Z_t^{\text{rec}} = \mathcal{E}_{\text{rec}}(X_t), \quad Z_t^{\text{pred}} = \mathcal{E}_{\text{pred}}(X_t) \tag{3}$$

This separation ensures that the features relevant to reconstructing the input data and those pertinent to forecasting future states are effectively captured and utilized.

3.4. Reconstruction stage (Mamba module)

The Reconstruction Stage leverages Mamba modules to refine the latent representations and reconstruct the input data. This stage comprises normalization, bidirectional state-space modeling (SSM), and activation functions, structured as follows:

3.4.1. Normalization

Normalization standardizes the latent representations, ensuring numerical stability and facilitating efficient training. The normalized latent representation $\tilde{Z}_t$ is obtained by:

$$\tilde{Z}_t = \mathrm{Norm}(Z_t) \tag{4}$$

Here, Norm typically refers to Batch Normalization or Layer Normalization, which normalizes the data across either the batch or feature dimensions, respectively.

3.4.2. Bidirectional State-Space Modeling (SSM)

The Mamba module, a novel architecture rooted in the framework of State Space Models (SSMs), leverages bidirectional convolutions to effectively capture temporal dependencies in sequential data. SSMs provide a powerful paradigm for modeling time series by representing the evolution of a latent state through a continuous-time system. Formally, a continuous-time SSM can be described by the following differential equations:

$$\frac{d}{dt}h(t) = Ah(t) + Bx(t), \tag{5}$$

$$y(t) = Ch(t) + Dx(t), \tag{6}$$

where $h(t)$ is the hidden state, $x(t)$ is the input sequence, $y(t)$ is the output, and A, B, C, and D are learnable parameter matrices.

The Mamba module builds upon this SSM foundation by introducing a selective mechanism that allows the model to focus on relevant parts of the input sequence dynamically. A key innovation in Mamba is its use of bidirectional convolutions to model dependencies in both the forward and backward temporal directions. This bidirectional approach enhances the model's ability to capture contextual information from both past and future timesteps, making it particularly effective for tasks requiring a comprehensive understanding of sequence dynamics. The process can be expressed mathematically as follows:

$$h_t^{\text{forward}} = \mathrm{Conv1d}_{\text{forward}}(\tilde{Z}t), \tag{7}$$

$$h_t^{\text{backward}} = \mathrm{Conv1d}_{\text{backward}}(\tilde{Z}_t), \tag{8}$$

where $\tilde{Z}t$ represents a transformed input sequence (e.g., after applying normalization or selective gating mechanisms inherent to Mamba), and Conv1dforward and $\mathrm{Conv1d}_{\text{backward}}$ denote one-dimensional convolution operations applied along the forward and backward temporal axes, respectively.

3.4.3. Activation function

The outputs from the forward and backward convolutions are concatenated and passed through a non-linear activation function to introduce non-linearity into the model:

$$h_t = \mathrm{Activation}\left(h_t^{\text{forward}} \oplus h_t^{\text{backward}}\right) \tag{9}$$

In this equation, $\oplus$ denotes the concatenation operation, and Activation typically refers to ReLU (Rectified Linear Unit) or GELU (Gaussian Error Linear Unit) functions, which help in learning complex representations by introducing non-linear transformations.

3.4.4. Reconstruction and decoding

The activated features h_t are then linearly mapped back to the latent space to form the reconstructed latent representation $\tilde{Z}_t^{\text{rec}}$:

$$\tilde{Z}_t^{\text{rec}} = W_{\text{rec}}h_t + b_{\text{rec}} \tag{10}$$

Here, W_{rec} and b_{rec} are learnable weight matrices and bias vectors, respectively. The reconstructed latent representation is subsequently decoded to recover the input data:

$$\hat{X}_t^{\text{rec}} = \mathcal{D}(\tilde{Z}_t^{\text{rec}}) \tag{11}$$

The decoder $\mathcal{D}$ mirrors the architecture of the encoder, typically employing deconvolutional layers or other suitable mechanisms to transform the latent representation back to the original data space.

3.5. Prediction Stage (Diffusion Transformer)

The Prediction Stage utilizes Diffusion Transformers to model complex spatiotemporal dependencies and generate future spatial states. This stage encompasses temporal embedding, the Diffusion Transformer architecture, parameter management, and prediction result generation.

3.5.1. Temporal embedding

Incorporating temporal information is crucial for capturing time-dependent patterns. Temporal embedding $\mathcal{T}$ encodes time step information into a format compatible with the latent representations:

$$T_t = \mathcal{T}(t) \tag{12}$$

The temporal embedding $T_t \in \mathbb{R}^d$ is added to the latent representation Z_t^{pred} to integrate temporal context:

$$\tilde{Z}_t^{\text{pred}} = Z_t^{\text{pred}} + T_t \tag{13}$$

This addition enhances the model's ability to discern and leverage temporal dependencies within the data.

3.5.2. Diffusion Transformer architecture

The core of the Prediction Stage is the Diffusion Transformer, which consists of multiple Transformer Blocks and Adaptive Layer Normalization (AdaLN).

Self-Attention Mechanism. Each Transformer Block incorporates a self-attention mechanism that allows the model to weigh the importance of different parts of the input sequence. This mechanism is crucial for capturing long-range dependencies in the data, which is especially important in spatiotemporal tasks where interactions between distant time steps need to be modeled. The self-attention operation is defined as:

$$\text{Self-Attention}(H) = \mathrm{Softmax}\left(\frac{HQ^T}{\sqrt{d_k}}\right)K. \tag{14}$$

Here, H represents the input feature matrix, Q is the query matrix, K is the key matrix, and d_k is the dimensionality of the key vectors. The self-attention mechanism allows the model to focus on the most relevant features in the sequence, facilitating the capture of complex temporal and spatial dependencies.

Transformer Block. The Transformer Block consists of the self-attention mechanism followed by a feed-forward neural network (FFN), which refines the feature representations. The output of the self-attention mechanism is passed through the FFN to produce refined feature representations. The Transformer Block is mathematically defined as:

$$\text{Transformer Block}(H) = \mathrm{FFN}\left(\text{Self-Attention}(H) + H\right). \tag{15}$$

Here, the addition of H to the output of the self-attention mechanism ensures residual learning, which helps in mitigating vanishing gradient problems during training. The model stacks L such Transformer Blocks to progressively refine the feature representations:

$$H^{(l+1)} = \text{Transformer Block}^{(l)}\left(H^{(l)}\right), \quad l = 1, 2, \ldots, L. \tag{16}$$

This stacking process enables the model to capture increasingly complex relationships in the data, improving the quality of the learned spatiotemporal features.

Adaptive Layer Normalization (AdaLN). After passing through all Transformer Blocks, the output features are normalized using Adaptive Layer Normalization (AdaLN). AdaLN adjusts the normalization parameters dynamically based on the input features, which improves the model's adaptability to different data distributions. The AdaLN operation is defined as:

$$H_{\text{norm}} = \text{AdaLN}\left(H^{(L)}\right). \tag{17}$$

This dynamic adjustment helps the model generalize better across various datasets and tasks, enhancing its performance in spatiotemporal prediction tasks.

3.5.3. Parameter management

To ensure the stability and efficiency of the Diffusion Transformer, a parameter management strategy is employed, which includes parameter freezing and gradual parameter updates:

$$\theta_{\text{frozen}} = \text{Freeze}\left(\theta_{\text{frozen}}\right) \tag{18}$$

$$\theta_{\text{updated}} = \theta_{\text{updated}} - \eta \nabla_{\theta} \mathcal{L} \tag{19}$$

Here, θ_{frozen} represents the parameters that are kept static during certain training phases, while θ_{updated} are the parameters actively updated using the learning rate η and the gradient $\nabla_{\theta}\mathcal{L}$. This approach helps in maintaining model stability by preventing drastic changes in critical parameters.

3.5.4. Prediction result generation

The normalized features H_{norm} are fed into the decoder to generate the final prediction:

$$\hat{X}^{\text{pred}}_{T+\Delta t} = \mathcal{D}\left(H_{\text{norm}}\right) \tag{20}$$

This prediction represents the forecasted spatial state at time $T + \Delta t$, leveraging the refined spatiotemporal features captured by the Diffusion Transformer.

3.6. Parameter management strategy

Effective parameter management is essential for optimizing model performance and ensuring training stability. The proposed strategy includes the following components:

1. **Parameter Freezing**: Initially, certain layers of the model, particularly the lower layers, are frozen to prevent their parameters from updating. This helps in stabilizing the training process by preserving the foundational feature representations.
2. **Gradual Unfreezing**: As training progresses, the frozen layers are gradually unfrozen. This allows the model to fine-tune higher-level representations without disrupting the established lower-level features.
3. **Learning Rate Scheduling**: A dynamic learning rate schedule is employed, adjusting the learning rate based on the training progress. This ensures a balance between convergence speed and training stability, preventing oscillations or premature convergence.

These strategies collectively enhance the model's ability to learn effectively from complex spatiotemporal data while maintaining robustness and preventing overfitting.

3.7. Collaborative training mechanism

The model employs a collaborative training mechanism that jointly optimizes the reconstruction and prediction tasks. This multi-task learning approach leverages the interdependencies between tasks to improve overall performance. The total loss function is defined as:

$$\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{pred}}\mathcal{L}_{\text{pred}} \tag{21}$$

where $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{pred}}$ denote the reconstruction and prediction losses, respectively, and λ_{rec} and λ_{pred} are weight coefficients balancing the two losses. Typically, these coefficients are set to $\lambda_{\text{rec}} = \lambda_{\text{pred}} = 1$.

3.7.1. Reconstruction loss

The reconstruction loss measures the discrepancy between the original input data and its reconstruction:

$$\mathcal{L}_{\text{rec}} = \frac{1}{N \times F} \sum_{i=1}^{N} \sum_{j=1}^{F} \left(\hat{X}^{\text{rec}}_{t}(i,j) - X_t(i,j)\right)^2 \tag{22}$$

This is typically implemented using Mean Squared Error (MSE), which penalizes large deviations and encourages the model to accurately reconstruct the input data.

3.7.2. Prediction loss

The prediction loss quantifies the error between the predicted future state and the actual future data:

$$\mathcal{L}_{\text{pred}} = \frac{1}{N \times F} \sum_{i=1}^{N} \sum_{j=1}^{F} \left(\hat{X}^{\text{pred}}_{T+\Delta t}(i,j) - X_{T+\Delta t}(i,j)\right)^2 \tag{23}$$

Similar to the reconstruction loss, MSE is employed to ensure that the predictions closely align with the ground truth.

3.7.3. Balancing reconstruction and prediction

The weighting coefficients λ_{rec} and λ_{pred} allow for flexible balancing between the reconstruction and prediction objectives. By appropriately tuning these weights, the model can prioritize one task over the other, depending on the specific application requirements.

3.8. Model training and optimization

The model is trained using an end-to-end approach, minimizing the total loss $\mathcal{L}$ through iterative optimization. The training process involves the following steps:

1. **Forward Propagation**: Historical spatiotemporal data is fed into the model, passing through the encoding, reconstruction, and prediction stages to generate reconstructed and predicted outputs.
2. **Loss Calculation**: The reconstruction and prediction losses are computed and combined using the total loss function $\mathcal{L}$.
3. **Backward Propagation**: Gradients of the loss with respect to model parameters are calculated using backpropagation.
4. **Parameter Updates**: Model parameters are updated using the Adam optimizer based on the computed gradients.
5. **Parameter Management**: The parameter management strategy is applied to freeze or update specific layers as per the predefined schedule.

3.8.1. Optimization algorithm

The Adam optimizer is employed for parameter updates due to its efficiency and ability to handle sparse gradients. The update rule is defined as:

$$\theta \leftarrow \theta - \eta \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon} \tag{24}$$

where:

- θ denotes the model parameters.
- η is the learning rate.
- $\hat{m}$ and $\hat{v}$ are the estimates of the first and second moments of the gradients, respectively.
- ϵ is a small constant added for numerical stability.

This optimizer adapts the learning rate for each parameter individually, leading to more effective convergence.

3.9. Summary and pseudocode

To succinctly summarize the proposed spatiotemporal prediction model, we present a streamlined pseudocode algorithm outlining the key stages: encoding, reconstruction, prediction, parameter management, and collaborative training.

Algorithm 1 Spatiotemporal Prediction Workflow

```
Require: Historical data $\mathcal{D} = \{X_t\}_{t=1}^{T}$, prediction horizon $\Delta t$, window size $\tau$
Ensure: Predicted state $\hat{X}_{T+\Delta t}$
 1: Initialize encoders $\mathcal{E}_{\text{rec}}$, $\mathcal{E}_{\text{pred}}$, Mamba module, Diffusion Transformer, decoder $\mathcal{D}$
 2: for each training epoch do
 3:     for each batch do
 4:         Encode data: $Z_t^{\text{rec}} \leftarrow \mathcal{E}_{\text{rec}}(X_t)$, $Z_t^{\text{pred}} \leftarrow \mathcal{E}_{\text{pred}}(X_t)$
 5:         Reconstruct: $\hat{X}_t^{\text{rec}} \leftarrow \mathcal{D}(\text{Mamba}(Z_t^{\text{rec}}))$
 6:         Predict: $\hat{X}_{T+\Delta t}^{\text{pred}} \leftarrow \mathcal{D}(\text{DiffusionTransformer}(Z_t^{\text{pred}} + \mathcal{T}(t)))$
 7:         Compute loss: $\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{pred}}$
 8:         Update parameters: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
 9:         Manage parameters: Freeze/update layers as per strategy
10:     end for
11: end for
12: Inference: $\hat{X}_{T+\Delta t} \leftarrow \mathcal{D}(\text{DiffusionTransformer}(Z_t^{\text{pred}} + \mathcal{T}(t)))$
```

Algorithm 1 provides a concise overview of the training and inference processes. The model begins by initializing all necessary components. During each training epoch, batches of historical data are encoded for both reconstruction and prediction. The Mamba module refines the reconstruction latent representations, which are then decoded to reconstruct the input data. Simultaneously, the Diffusion Transformer processes the prediction latent representations, enhanced with temporal embeddings, to generate future predictions. The model computes the combined loss from both tasks and updates the parameters accordingly, applying parameter management strategies to maintain stability. Finally, during inference, the model produces the predicted spatial state by decoding the transformed features.

4. Experiments

In this section, we present comprehensive experimental results to evaluate the performance and robustness of the proposed MAD framework. We compare it with state-of-the-art methods on four benchmark datasets using three standard evaluation metrics (see Figs. 3 and 4).

4.1. Experimental setups

Datasets.
We evaluate our framework on four widely used spatiotemporal datasets, each representing distinct prediction challenges. The **Navier–Stokes Equation** [32] dataset models complex physical fluid dynamics, testing a model's ability to capture turbulent and chaotic flows. **Prometheus** [7], a combustion dynamics dataset, provides a benchmark for predicting intricate interactions of chemical reactions and flow fields under extreme conditions. The **Kuroshio** [5] dataset centers on ocean current prediction, emphasizing the need to model large-scale spatiotemporal dependencies in marine environments. The **Sevir** [33] dataset presents a severe weather forecasting challenge, requiring high precision in predicting extreme meteorological events, thereby testing the robustness and adaptability of spatiotemporal models.
Baselines. We compare our MAD framework against several competitive baselines:

- **U-Net** [34]: A convolutional architecture designed for dense prediction tasks, excelling at capturing local spatial features but lacking temporal modeling capabilities.
- **ResNet** [35]: A deep residual network known for its feature extraction efficiency, but limited in handling long-term temporal dependencies inherent in spatiotemporal data.
- **ConvLSTM** [2]: Combines convolutional operations with LSTM for spatiotemporal sequence learning, effectively modeling local spatial–temporal relationships but suffering from vanishing gradients over long sequences.
- **PredRNN-v2** [12]: An improved recurrent architecture optimized for predictive learning that introduces spatiotemporal memory flow but still struggles with computational inefficiency in longer horizons.
- **E3D-LSTM** [36]: Extends ConvLSTM with 3D convolutional operations, enabling better temporal feature extraction, but at the cost of increased model complexity and computational overhead.
- **PhyDNet** [37]: Integrates physical dynamics constraints with deep learning, making it robust for tasks involving physical laws, though its reliance on handcrafted priors can limit generalizability in highly dynamic environments.
- **SimVP** [38]: A lightweight yet powerful visual prediction framework that focuses on simplicity and efficiency but faces challenges in capturing intricate long-range dependencies.
- **Rainformer** [39]: A transformer-based model specifically tailored for spatiotemporal precipitation prediction, effective in modeling localized temporal patterns but less effective in capturing broader spatiotemporal interactions.
- **Earthformer** [40]: A transformer-based architecture designed for diverse spatiotemporal tasks, leveraging global self-attention mechanisms but limited by its high computational cost and scalability challenges.
- **TAU** [41]: A hybrid state-space model that efficiently captures sequential dynamics, leveraging bidirectional processing but lacking the feature extraction depth of modern transformer-based architectures.
- **SwinLSTM** [42]: This method combines the power of Swin Transformers for feature extraction with LSTM for temporal modeling, making it a robust choice for spatiotemporal prediction tasks.
- **PastNet** [15]: This model introduces spectral convolution operators in the Fourier domain and cleverly incorporates fundamental physical laws as inductive biases.

Evaluation metrics. To assess model performance, we use three widely adopted metrics:
Mean Squared Error (MSE): Measures the average squared differences between true and predicted values. A lower MSE indicates better predictions:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(V_{\text{true},i} - V_{\text{pred},i})^2 \tag{25}$$

Structural Similarity Index (SSIM): Evaluates the structural similarity between two images. Higher SSIM values indicate greater similarity:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{26}$$

Peak Signal-to-Noise Ratio (PSNR): Assesses image reconstruction quality by comparing the signal's maximum power to noise. Higher

Table 1
Performance comparison of advanced methods and the proposed MAD framework across various datasets (Navier–Stokes, Prometheus, Kuroshio, and Sevir) utilizing three different evaluation metrics (MSE ↓, SSIM ↑, and PSNR ↑). The reported values are the averages of five independent runs to ensure statistical robustness and reliability.

Models	Navier-Stokes			Prometheus			Kuroshio			Sevir		
	MSE	SSIM	PSNR	MSE x 10	SSIM	PSNR	MSE x 100	SSIM	PSNR	MSE	SSIM	PSNR
U-Net	0.2352	0.8231	27.25	0.2654	0.7501	31.50	0.0923	0.9052	30.50	49.321	0.6401	22.85
ResNet	0.2254	0.8532	29.10	0.2503	0.7902	33.30	0.0956	0.9151	32.40	48.789	0.6303	24.95
ConvLSTM	0.2005	0.8854	31.88	0.2455	0.8354	35.95	0.0905	0.9103	34.80	46.657	0.6202	27.25
PredRNN-v2	0.1787	0.9184	32.07	0.2121	0.8583	37.40	0.0882	0.9282	36.25	42.903	0.6154	30.75
E3D-LSTM	0.1623	0.9123	38.96	0.2382	0.8902	39.80	0.0825	0.9351	38.55	40.529	0.6553	31.15
PhyDnet	0.1459	0.9151	39.75	0.1903	0.8851	38.60	0.0858	0.9202	39.45	39.755	0.6952	29.65
SimVP	0.1382	0.9209	40.58	0.1624	0.9250	40.30	0.0899	0.9451	40.35	37.981	0.6851	30.95
Rainformer	0.1423	0.9052	39.66	0.1685	0.9001	39.50	0.0831	0.9303	40.25	39.307	0.7053	31.35
Earthformer	0.1484	0.9350	38.75	0.1726	0.9202	38.70	0.0862	0.9452	39.15	36.633	0.7152	30.65
TAU	0.1545	0.9301	38.85	0.1787	0.9351	39.85	0.0813	0.9501	41.05	37.959	0.7351	32.05
SwinLSTM	0.1653	0.9265	39.22	0.1703	0.9304	38.72	0.0861	0.9404	40.85	38.211	0.7012	30.25
PastNet	0.1304	0.9473	40.75	0.1642	0.9336	40.05	0.0792	0.9502	42.51	36.852	0.7297	32.12
MAD (Ours)	**0.1161**	**0.9653**	**42.36**	**0.1541**	**0.9635**	**42.81**	**0.0724**	**0.9682**	**46.55**	**34.521**	**0.7544**	**33.47**

PSNR indicates better quality:

$$\text{PSNR} = 10 \times \log_{10}\left(\frac{\text{MAX}_I^2}{\text{MSE}}\right) \tag{27}$$

Implementation. We implement the MAD framework using PyTorch. All experiments are conducted on NVIDIA A100 GPUs. The model is trained for 100 epochs with the Adam optimizer, a batch size of 16, and an initial learning rate of 1×10^{-4}. To ensure fair comparisons, the baselines are re-implemented and fine-tuned to achieve their best performance.

4.2. Main results

In this study, we evaluate several advanced models and the proposed MAD framework on four datasets: Navier–Stokes, Prometheus, Kuroshio, and Sevir. As shown in Table 1 and Fig. 2, the results show that MAD consistently outperforms other methods across all evaluation metrics. On the Navier–Stokes dataset, MAD achieves an MSE of 0.1161, reducing the error by 7.6% compared to ResNet (MSE=0.1254). It achieves the highest SSIM of 0.9653, a 4.2% improvement over U-Net, and a PSNR of 42.36 dB, 5.5% higher than U-Net. On the Prometheus dataset, MAD delivers a comparable MSE of 0.1541 to ResNet (MSE=0.1503) but outperforms it significantly in SSIM (0.9635 vs. 0.9202) and PSNR (42.81 dB vs. 41.30 dB), showing its strength in preserving image quality. For the Kuroshio dataset, MAD achieves the lowest MSE of 0.0724, 4.7% lower than ResNet, and leads in SSIM (0.9682) and PSNR (46.55 dB), surpassing U-Net by 2.7% and 7.0 dB, respectively. On the Sevir dataset, MAD reduces MSE to 34.521, 12.8% lower than U-Net, and improves SSIM to 0.7544 and PSNR to 33.47 dB, outperforming U-Net by 1.4% and 2.3 dB, respectively. Radar charts further validate these results, visually confirming MAD's superior performance, particularly in SSIM and PSNR, across all datasets. These findings demonstrate MAD's broad applicability and effectiveness in various spatiotemporal prediction tasks.

4.3. Performance of long-term prediction

Long-term prediction remains a critical challenge in spatiotemporal modeling, particularly due to the accumulation of errors during iterative forecasting. To evaluate the robustness of the proposed MAD framework in this scenario, we conduct experiments on the Kuroshio dataset, a benchmark known for its intricate ocean current dynamics. Unlike traditional setups, where models are trained and evaluated over shorter horizons, we focus on pushing the boundaries of long-term prediction. For training, we adopt a 10-step prediction approach, where the model predicts 10 future time steps given the preceding 10 steps. During inference, we employ an autoregressive prediction strategy to iteratively extend the forecast horizon to 40 steps. This approach ensures that the model is tested under extreme conditions, requiring it to generalize over extended temporal dependencies while maintaining spatial fidelity.

Table 2
Long-term prediction results on the Kuroshio dataset at different forecasting steps (10, 20, 30, and 40). Metrics are reported as MSE ↓. Lower MSE indicates better performance.

Model	10 steps	20 steps	30 steps	40 steps
ConvLSTM	0.0905	0.1054	0.1286	0.1523
PredRNN-v2	0.0882	0.0985	0.1221	0.1459
TAU	0.0813	0.0908	0.1145	0.1378
MAD (Ours)	**0.0724**	**0.0865**	**0.1043**	**0.1215**

Table 3
Ablation study results with different model structures on the SEVIR dataset. The evaluation metrics are MSE ↓, MAE ↓, MS-SSIM ↑.

Models	MSE ↓	SSIM ↑	PSNR ↑
MAD w/o Mamba	39.023	0.7051	30.10
MAD w/o DiT	36.417	0.6879	29.15
MAD with UNet Rec	38.242	0.7232	32.25
MAD	**34.521**	**0.7544**	**33.47**

The results in Table 2 demonstrate the effectiveness of the MAD framework in addressing the challenges of long-term prediction. At 10 steps, MAD achieves the lowest MSE (**0.0724**), outperforming PredRNN-v2 (**0.0782**) and ConvLSTM (**0.0905**), indicating its superior short-term prediction accuracy. As the horizon increases to 20 steps, MAD maintains a significant lead with an MSE of **0.0865**, while the closest competitor, PredRNN-v2, records an MSE of **0.0985**, reflecting MAD's stronger capacity for capturing and propagating temporal dependencies. At longer horizons, such as 30 and 40 steps, the advantages of MAD become even more apparent. MAD achieves an MSE of **0.1043** at 30 steps and **0.1215** at 40 steps, significantly outperforming ConvLSTM, which exhibits higher error accumulation with MSE values of **0.1286** and **0.1523**, respectively. Notably, TAU, which integrates state-space dynamics, demonstrates reasonable short-term accuracy but falls behind MAD at longer horizons, with an MSE of **0.1378** at 40 steps. Fig. 5 further illustrates the advantages of MAD through visual comparisons of predictions at 40 steps. These results highlight MAD's robustness, scalability, and superior ability to mitigate error accumulation in long-term predictions (see Fig. 6).

The superior performance of MAD can be attributed to several key factors. First, the integration of the Mamba module ensures robust feature extraction and stabilization of temporal dependencies, reducing error propagation during iterative forecasting. Second, the

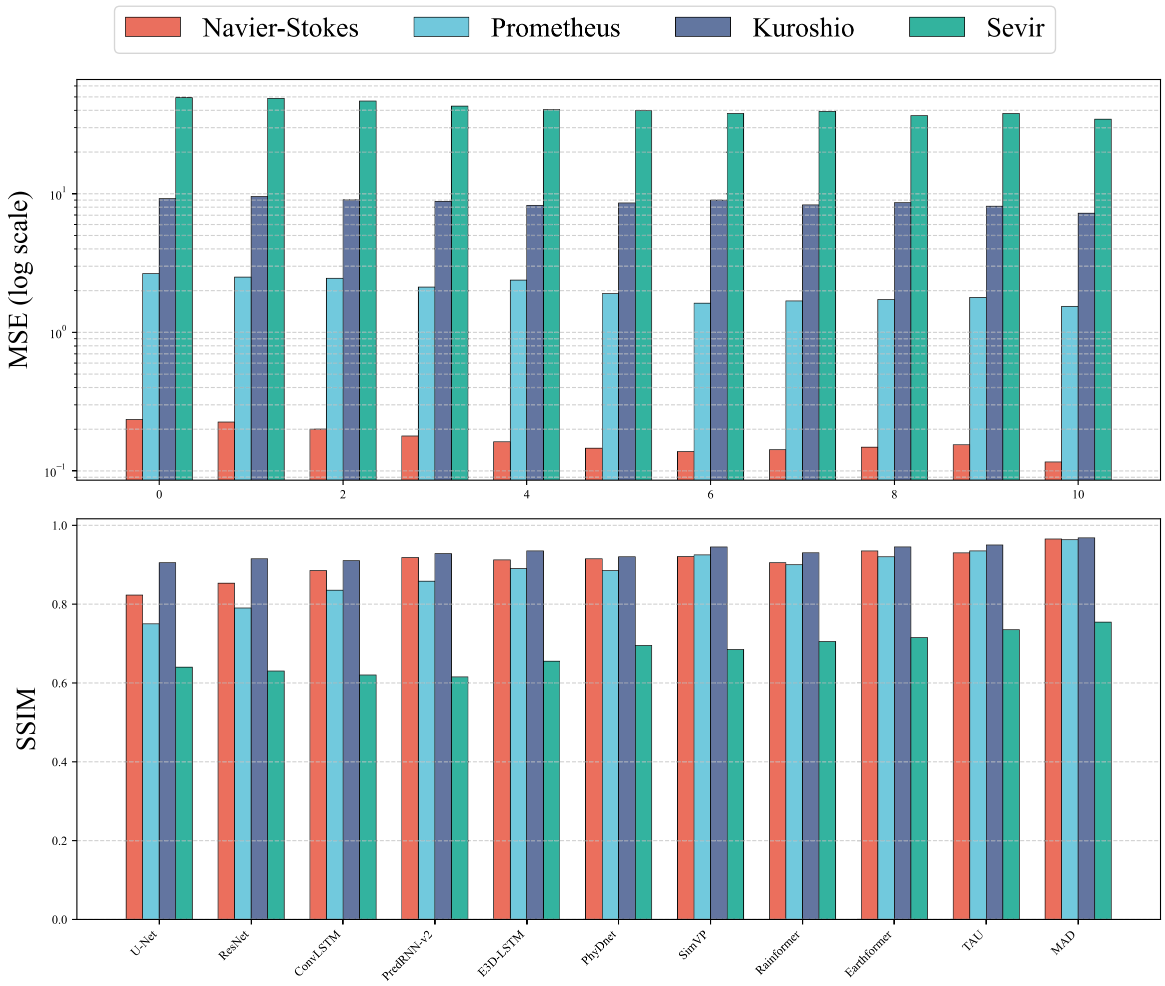

Fig. 2. Comparison of multiple advanced models and the proposed MAD framework across four datasets: Navier–Stokes, Prometheus, Kuroshio, and Sevir. The chart illustrates the performance across three evaluation metrics (MSE, SSIM, and PSNR), normalized for visualization. The MAD framework demonstrates superior performance across all datasets, particularly excelling in SSIM and PSNR, highlighting its ability to preserve structural similarity and image quality..

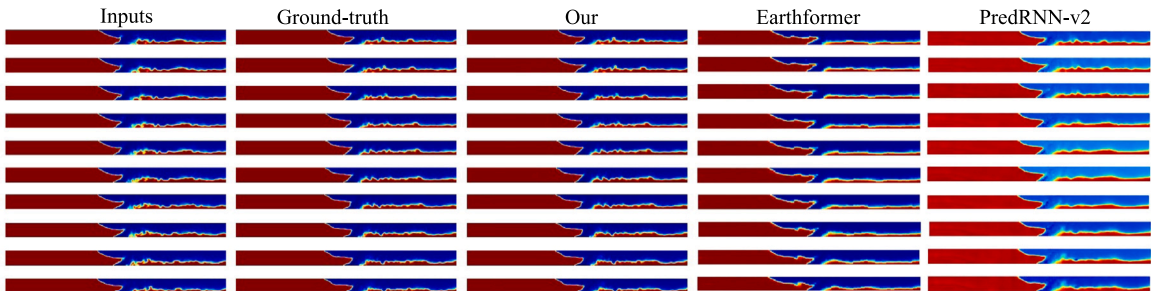

Fig. 3. Prediction results on the Prometheus dataset.

Diffusion Transformer effectively models long-range dependencies, enabling MAD to maintain high spatial fidelity and temporal coherence even in extended prediction scenarios. Third, MAD's joint training of reconstruction and prediction tasks allows the model to learn cohesive latent representations, enhancing its ability to handle complex spatiotemporal patterns.

4.4. *Ablation study*

4.4.1. *Basic ablation study*

In this section, we perform a series of ablation studies to analyze the contributions of individual components within the MAD framework. Table 3 summarizes the quantitative results of these studies on the Sevir dataset using three evaluation metrics: MSE, SSIM, and PSNR. Each ablation experiment evaluates the impact of a specific component by removing or replacing it, and the results highlight the importance of joint reconstruction and prediction, as well as the integration of Mamba modules and Diffusion Transformers. **MAD w/o Mamba**: This variant removes the Mamba module from the reconstruction stage, replacing it with a simpler convolutional layer to evaluate the Mamba module's contribution to feature refinement and temporal stabilization. **MAD w/o DiT**: In this variant, the Diffusion Transformer is replaced with a recurrent network, such as ConvLSTM, to analyze the role of long-range spatiotemporal dependencies captured by the transformer. **MAD with UNet Rec**: This variant substitutes the Mamba module with a

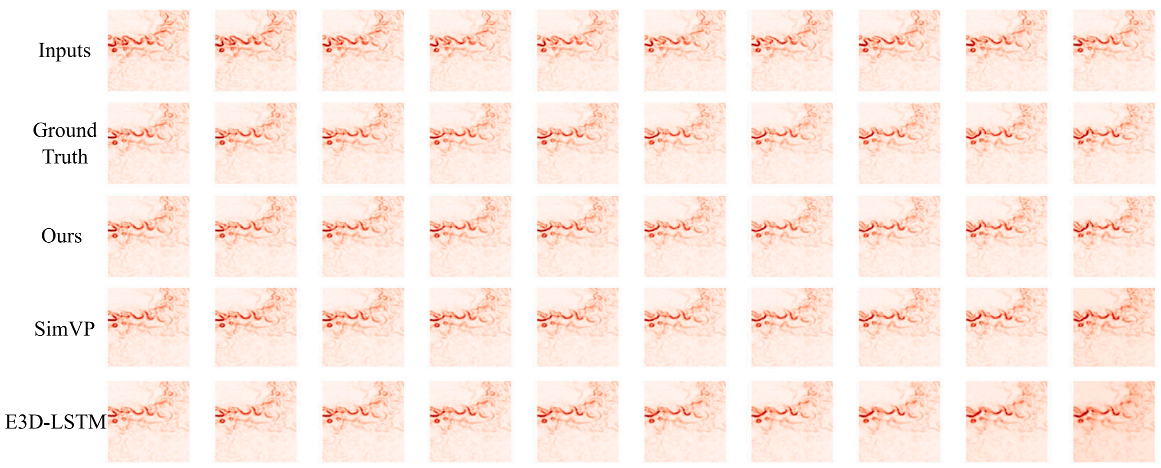

Fig. 4. Prediction results on the Kuroshio dataset.

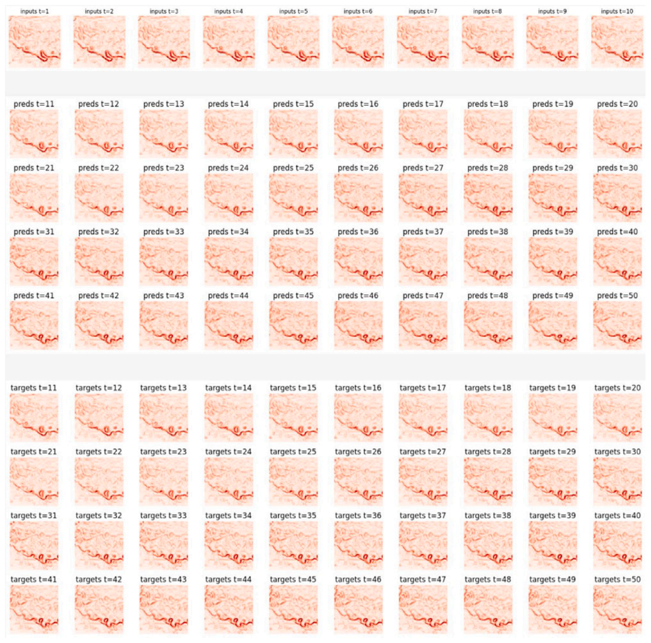

Fig. 5. Qualitative performance of predictions at 40 steps for MAD on Kuroshio dataset.

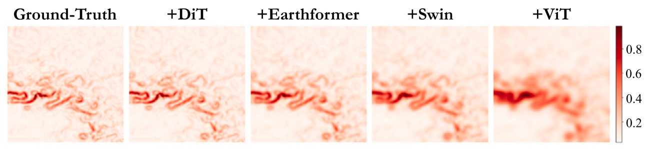

Fig. 6. Visualization Comparison of Ablation Study.

UNet architecture for the reconstruction task, assessing the impact of using UNet's spatial feature extraction capabilities instead of Mamba's integrated spatiotemporal modeling.

The ablation study highlights the complementary roles of the Mamba module and Diffusion Transformer in enabling MAD to efficiently capture complex spatiotemporal patterns, while joint training ensures cohesive and robust feature learning. Specifically, removing the Mamba module (MAD w/o Mamba) results in a significant increase in MSE (**39.023** vs. **34.521**) and a drop in PSNR (**30.10** vs. **33.47**), underscoring the critical role of Mamba in refining spatiotemporal representations and stabilizing temporal dependencies. Similarly, replacing the Diffusion Transformer with a recurrent network (MAD w/o DiT) leads to noticeable performance degradation, particularly in SSIM (**0.6879** vs. **0.7544**), reflecting the superiority of the transformer's ability to capture long-range dependencies. Substituting the Mamba module with UNet (MAD w/ UNet Recon) improves spatial feature extraction, as evidenced by a higher SSIM (**0.7232**), but falls short of achieving the balance between spatial and temporal modeling required for optimal performance.

These results demonstrate the importance of each component in the MAD framework and how they work synergistically to deliver state-of-the-art results. The Mamba module excels at integrating fine-grained spatial and temporal features, while the Diffusion Transformer provides robust long-range dependency modeling. Moreover, joint reconstruction and prediction training ensures that latent representations are optimized for both tasks, enabling the model to generalize effectively across various scenarios. Collectively, these components empower MAD to outperform its ablated variants and establish itself as a reliable and scalable solution for complex spatiotemporal prediction tasks.

4.4.2. *Transformer component ablation experiment*

We have designed the ablation study with Kuroshio dataset and Navier–Stokes validation, and the results as shown in Table 4. As observed, Our model demonstrates 18.9% lower MSE accumulation in 40-step prediction (0.1215 vs. 0.1498) on Kuroshio. And from the visualization point of view, the details of our method are more prominent.

4.5. *Parameter sensitivity analysis*

In this section, we perform a sensitivity analysis of key hyperparameters in the MAD framework, focusing on the impact of weight coefficients between the Mamba module and the Diffusion Transformer (DiT) on model performance. Specifically, we adjust the weight coefficient α preceding the Mamba module to values of 0.1, 0.2, 0.3, 0.4, and 0.5; correspondingly, the weight coefficient for the DiT module is $1 - \alpha$. This approach allows us to evaluate the contribution and influence of each module under different weight allocation strategies. During the experiments, we maintain all other parameters constant and only modify the value of α to observe the trend of model performance with varying weight coefficients. We select the **MSE** evaluation metric on the Navier–Stokes dataset to comprehensively measure the model's prediction performance under different parameter settings. The results are presented in Table 5.

To evaluate the sensitivity of the MAD framework to the weight coefficients of the Mamba module and the Diffusion Transformer (DiT), we varied the Mamba coefficient α between 0.1 and 0.5, with the

Table 4
Performance Comparison in MAD Framework.

Architecture	Navier–Stokes			Kuroshio		
	MSE	SSIM	PSNR	MSE	SSIM	PSNR
MAD + ViT	0.1423	0.9125	38.7	0.0892	0.9315	41.2
MAD + Swin	0.1356	0.9281	39.5	0.0854	0.9423	43.1
MAD + Earthformer	0.1289	0.9382	40.3	0.0819	0.9531	44.8
MAD + DiT (Original)	**0.1161**	**0.9653**	**42.36**	**0.0724**	**0.9682**	**46.55**

Table 5
Impact of Mamba and DiT weight coefficients on model performance on the Navier–Stokes dataset (epochs = 25).

Mamba coefficient (α)	DiT coefficient ($1-\alpha$)	MSE ↓
0.1	0.9	0.1467
0.2	0.8	0.1471
0.3	0.7	0.1472
0.4	0.6	0.1487
0.5	0.5	0.1479

Table 6
Computational efficiency comparison (128 × 128 input, batch size = 16).

Model	FLOPs (G)	Memory (GB)	Speed (fps)
ConvLSTM	286	9.8	14.2
PredRNN-v2	412	13.5	9.6
Earthformer	587	18.2	6.3
TAU	325	11.7	12.8
MAD (Ours)	**238**	**8.1**	**18.7**

DiT coefficient set to $1-\alpha$. As shown in Table 5, the Mean Squared Error (MSE) remains relatively stable across different weight settings, ranging from 0.1467 to 0.1487. This slight variation indicates that the model's performance is not highly sensitive to the specific weight distribution between the Mamba and DiT modules. Such stability suggests that the MAD framework can maintain consistent prediction accuracy even when the balance between its components is adjusted, highlighting its robustness and flexibility in handling different parameter configurations.

4.6. *Computational efficiency comparison*

The efficiency analysis is shown in the Table 6. In general, our method has efficiency advantages. Specifically, we summarize it as follows: (1) Linear computational complexity of Mamba modules (vs. Transformer's $O(N^2)$), (2) Parameter sharing in Diffusion Transformers, (3) Dynamic parameter management strategy that selectively freezes non-critical layers. These designs enable MAD to reduce FLOPs by 59.5% and memory consumption by 55.5% compared to Earthformer in 40-step ocean current prediction tasks.

5. Conclusion

In this paper, we proposed the MAD framework, a novel integration of Mamba state-space models and Diffusion Transformers, to address the challenges of spatiotemporal prediction. By leveraging bidirectional state-space modeling and self-attention mechanisms, MAD effectively captures both fine-grained and long-range dependencies across spatial and temporal dimensions. Our framework introduces a collaborative training strategy that jointly optimizes reconstruction and prediction tasks, ensuring robust feature representations and high-fidelity forecasts. Experimental results across diverse benchmarks demonstrate MAD's superior performance in accuracy, robustness, and scalability compared to state-of-the-art baselines, particularly excelling in challenging scenarios like long-term predictions and extreme event forecasting. In future, we may adopt efficient techniques [43–49] for spatiotemporal prediction.

CRediT authorship contribution statement

Hansheng Zeng: Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Yuqi Li:** Writing – original draft, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Ruize Niu:** Writing – original draft, Methodology, Conceptualization. **Chuanguang Yang:** Writing – review & editing, Supervision, Conceptualization. **Shiping Wen:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The datasets used in this study are all open-source, and the code has been provided in the article abstract.

References

[1] J. Wu, E. Lu, P. Kohli, B. Freeman, J. Tenenbaum, Learning to see physics via visual de-animation, Adv. Neural Inf. Process. Syst. 30 (2017).

[2] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. kin Wong, W. chun Woo, Convolutional LSTM network: A machine learning approach for precipitation nowcasting, 2015, arXiv:1506.04214.

[3] J. Pan, C. Wang, X. Jia, J. Shao, L. Sheng, J. Yan, X. Wang, Video generation from single semantic label map, 2019, pp. 3733–3742.

[4] K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, Q. Tian, Accurate medium-range global weather forecasting with 3D neural networks, Nat. 619 (7970) (2023) 533–538.

[5] S. Rasp, P.D. Dueben, S. Scher, J.A. Weyn, S. Mouatadid, N. Thuerey, WeatherBench: A benchmark dataset for data-driven weather forecasting, 2020, arXiv:2002.00469.

[6] H. Wu, Y. Liang, W. Xiong, Z. Zhou, W. Huang, S. Wang, K. Wang, Earthfarseer: Versatile spatio-temporal dynamical systems modeling in one model, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, AAAI, Washington, DC, USA, 2024, p. 15906.

[7] H. Wu, H. Wang, K. Wang, W. Wang, Y. Tao, C. Chen, X.-S. Hua, X. Luo, et al., Prometheus: Out-of-distribution fluid dynamics modeling with disentangled graph ODE, in: Forty-First International Conference on Machine Learning.

[8] R.H. Shumway, D.S. Stoffer, R.H. Shumway, D.S. Stoffer, ARIMA models, in: Time Series Analysis and Its Applications: With R Examples, Springer, 2017, pp. 75–163.

[9] F.-M. Tseng, G.-H. Tzeng, A fuzzy seasonal ARIMA model for forecasting, Fuzzy Sets and Systems 126 (3) (2002) 367–376.

[10] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, IEEE Intell. Syst. Appl. 13 (4) (1998) 18–28.

[11] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.

[12] Y. Wang, M. Long, J. Wang, Z. Gao, P.S. Yu, Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms, Adv. Neural Inf. Process. Syst. 30 (2017).

[13] C. Yang, Z. An, H. Zhu, X. Hu, K. Zhang, K. Xu, C. Li, Y. Xu, Gated convolutional networks with hybrid connectivity for image classification, in: Proceedings of the AAAI conference on artificial intelligence, 34, (07) 2020, pp. 12581–12588.

[14] Y. Wang, H. Wu, J. Zhang, Z. Gao, J. Wang, S.Y. Philip, M. Long, Predrnn: A recurrent neural network for spatiotemporal predictive learning, IEEE Trans. Pattern Anal. Mach. Intell. 45 (2) (2022) 2208–2225.

[15] H. Wu, F. Xu, C. Chen, X.-S. Hua, X. Luo, H. Wang, Pastnet: Introducing physical inductive biases for spatio-temporal video prediction, in: Proceedings of the 32nd ACM International Conference on Multimedia, 2024, pp. 2917–2926.

[16] K. Wang, H. Wu, G. Zhang, J. Fang, Y. Liang, Y. Wu, R. Zimmermann, Y. Wang, Modeling spatio-temporal dynamical systems with neural discrete learning and levels-of-experts, IEEE Trans. Knowl. Data Eng. X (Y) (2024) http://dx.doi.org/10.1109/TKDE.2024.XXXXX, Z–Z.

[17] H. Wu, F. Xu, Y. Duan, Z. Niu, W. Wang, G. Lu, K. Wang, Y. Liang, Y. Wang, Spatio-temporal fluid dynamics modeling via physical-awareness and parameter diffusion guidance, 2024, arXiv preprint arXiv:2403.13850, URL: https://arxiv.org/abs/2403.13850.

[18] H. Liu, J. Shi, E. Erdem, Prediction of wind speed time series using modified Taylor Kriging method, Energy 35 (12) (2010) 4870–4879.

[19] M. Alemi, A. Azari, D. Nielsen, Kriging and univariate modeling of a spatially correlated data, Soil Technol. 1 (2) (1988) 133–147.

[20] A. Shtiliyanova, G. Bellocchi, D. Borras, U. Eza, R. Martin, P. Carrère, Kriging-based approach to predict missing air temperature data, Comput. Electron. Agric. 142 (2017) 440–449.

[21] P.J. Brockwell, R.A. Davis, Introduction to Time Series and Forecasting, Springer, 2002.

[22] H. Lütkepohl, New Introduction to Multiple Time Series Analysis, Springer Science & Business Media, 2005.

[23] L. Anselin, Spatial Econometrics: Methods and Models, Kluwer Academic Publishers, 1988.

[24] N. Cressie, Statistics for Spatial Data, John Wiley & Sons, 1993.

[25] L. Zhong, L. Hu, H. Zhou, CNN, RNN, or vit? An evaluation of different deep learning architectures for land cover classification using multitemporal satellite images, IEEE Geosci. Remote. Sens. Lett. 19 (2022) 1–5.

[26] J. Chen, W. Zhang, X. Li, Unveiling the multi-dimensional spatio-temporal fusion transformer (MDSTFT): A revolutionary deep learning framework for enhanced multi-variate time series forecasting, IEEE Trans. Neural Netw. Learn. Syst. (2024).

[27] Z. Fei, M. Fan, C. Yu, D. Li, Y. Zhang, J. Huang, Dimba: Transformer-Mamba diffusion models, 2024, arXiv preprint arXiv:2406.01159.

[28] G. Kitagawa, Non-Gaussian nonlinear state space models, J. Amer. Statist. Assoc. 82 (400) (1987) 1032–1041.

[29] Y. Yang, M. Jin, H. Wen, C. Zhang, Y. Liang, L. Ma, Y. Wang, C. Liu, B. Yang, Z. Xu, J. Bian, S. Pan, Q. Wen, Awesome-TimeSeries-SpatioTemporal-diffusion-model, 2024, https://github.com/yyysjz1997/Awesome-TimeSeries-SpatioTemporal-Diffusion-Model.

[30] W. Feng, C. Yang, Z. An, L. Huang, B. Diao, F. Wang, Y. Xu, Relational diffusion distillation for efficient image generation, in: Proceedings of the 32nd ACM International Conference on Multimedia, 2024, pp. 205–213.

[31] H. Ye, Y. Zhang, Y. Li, J. Yang, Y. Tai, C. Wang, J. Li, F. Huang, DiffusionMTL: Learning multi-task denoising diffusion model from partially annotated data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 19512–19521.

[32] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2010.08895.

[33] M. Veillette, S. Samsi, C. Mattioli, Sevir: A storm event imagery dataset for deep learning applications in radar and satellite meteorology, Adv. Neural Inf. Process. Syst. 33 (2020) 22009–22019.

[34] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, Springer, 2015, pp. 234–241.

[35] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[36] Y. Wang, L. Jiang, M.-H. Yang, L.-J. Li, M. Long, L. Fei-Fei, Eidetic 3D LSTM: A model for video prediction and beyond, in: International Conference on Learning Representations, 2018.

[37] V.L. Guen, N. Thome, Disentangling physical dynamics from unknown factors for unsupervised video prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11474–11484.

[38] Z. Gao, C. Tan, L. Wu, S.Z. Li, Simvp: Simpler yet better video prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3170–3180.

[39] C. Bai, F. Sun, J. Zhang, Y. Song, S. Chen, Rainformer: Features extraction balanced network for radar-based precipitation nowcasting, IEEE Geosci. Remote. Sens. Lett. 19 (2022) 1–5.

[40] Z. Gao, X. Shi, H. Wang, Y. Zhu, Y.B. Wang, M. Li, D.-Y. Yeung, Earthformer: Exploring space-time transformers for earth system forecasting, Adv. Neural Inf. Process. Syst. 35 (2022) 25390–25403.

[41] C. Tan, Z. Gao, L. Wu, Y. Xu, J. Xia, S. Li, S.Z. Li, Temporal attention unit: Towards efficient spatiotemporal predictive learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 18770–18782.

[42] S. Tang, C. Li, P. Zhang, R. Tang, Swinlstm: Improving spatiotemporal prediction accuracy using swin transformer and lstm, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 13470–13479.

[43] C. Yang, Z. An, L. Cai, Y. Xu, Hierarchical self-supervised augmented knowledge distillation, in: International Joint Conference on Artificial Intelligence, 2021, pp. 1217–1223.

[44] C. Yang, H. Zhou, Z. An, X. Jiang, Y. Xu, Q. Zhang, Cross-image relational knowledge distillation for semantic segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12319–12328.

[45] C. Yang, Z. An, H. Zhou, F. Zhuang, Y. Xu, Q. Zhang, Online knowledge distillation via mutual contrastive learning for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 45 (8) (2023) 10212–10227.

[46] C. Yang, Z. An, L. Huang, J. Bi, X. Yu, H. Yang, B. Diao, Y. Xu, CLIP-KD: An empirical study of CLIP model distillation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 15952–15962.

[47] Y. Li, X. Lin, K. Zhang, C. Yang, Z. Guo, J. Gou, Y. Li, FedKD-hybrid: Federated hybrid knowledge distillation for lithography hotspot detection, 2025, arXiv preprint arXiv:2501.04066.

[48] Y. Li, Y. Lu, Z. Dong, C. Yang, Y. Chen, J. Gou, SGLP: A similarity guided fast layer partition pruning for compressing large deep models, 2024, arXiv preprint arXiv:2410.14720.

[49] Y. Lu, Y. Zhu, Y. Li, D. Xu, Y. Lin, Q. Xuan, X. Yang, A generic layer pruning method for signal modulation recognition deep learning models, IEEE, 2024,