Modeling Spatially Varying Physical Dynamics for Spatiotemporal Predictive Learning

Yijun Lin
lin00786@umn.edu
University of Minnesota, Twin Cities

Yao-Yi Chiang
yaoyi@umn.edu
University of Minnesota, Twin Cities

ABSTRACT

Recent advances in incorporating physical knowledge into deep neural networks can estimate previously unknown, governing partial differential equations (PDEs) in a data-driven way, which have shown promising results in spatiotemporal predictive learning. However, these methods typically assume universal governing PDEs across space, which is impractical for modeling complex spatiotemporal phenomena with high spatial variability (e.g., climate). Also, they cannot effectively model the evolution of potential errors in estimating the physical dynamics over time. This paper introduces a physics-guided neural network, SVPNet, that learns effective physical representations by estimating the error evolution in the physical states for updating and modeling spatially varying physical dynamics to predict the next state. Experiments conducted on four scenarios, including benchmarks and real-world datasets, show that SVPNet outperforms the state-of-the-art methods on spatiotemporal prediction tasks for natural processes and significantly improves the prediction when the training data are limited. Ablation studies also highlight that SVPNet is more powerful in capturing physical dynamics in complex physical systems than other physics-enabled models, e.g., PhyDNet.

KEYWORDS

spatiotemporal prediction, physics-guided machine learning, spatial AI

ACM Reference Format:
Yijun Lin and Yao-Yi Chiang. 2023. Modeling Spatially Varying Physical Dynamics for Spatiotemporal Predictive Learning. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Spatiotemporal predictive learning aims to predict future frames conditioned on previous frames for modeling complex spatiotemporal phenomena, such as weather conditions [34, 51], climate change [31], and traffic flow [14, 52, 55]. In particular, data-driven methods typically model spatiotemporal dependencies using a specific architecture design, e.g., recurrent neural networks (RNNs) [10, 17] integrated with the 2D convolution [46, 51], 3D convolution [44, 54], and attention mechanism [7, 8, 15, 23, 38]. These data-driven methods can work well, but they do not model the underlying physical processes essential for predicting spatiotemporal phenomena, e.g., climate, making them sensitive to out-of-domain scenarios, requiring a large amount of training data, and may produce physically inconsistent results (see [18, 48]).

Recent work has incorporated prior physical knowledge, e.g., formalized by partial differential equations (PDEs) [12, 13, 16, 33, 53], into deep neural networks, to overcome the challenges in data-driven methods and have shown promising results for spatiotemporal prediction tasks. Some approaches rely on expert knowledge about the target phenomena to generate well-defined underlying dynamics to guide the design of neural networks and the learning process, e.g., Burgers' equation and the harmonic oscillator [32], advection-diffusion [12], and energy exchanges [19].

A promising direction, which does not require expert knowledge, is to learn the underlying dynamics of a physical system by estimating the PDEs in a data-driven way, either directly learning from the data [22, 24, 25] or a PDE-governed physics space disentangled from residual factors [16, 53, 56]. There are two major challenges to this type of approach. The first challenge is that estimating PDEs directly from data typically introduces noise and thus requires further correction, e.g., data assimilation [2] incorporating observations in process modeling [3, 6]. However, existing approaches either ignore correction [24, 25] or fail to capture the underlying noise evolution over time in the predicted states [16, 56]. The second challenge is that in many natural spatiotemporal processes, the governing PDEs can vary across space, including spatially varying PDE parameters and multiple physical systems described by different forms of PDEs. For example, a global climate model cannot capture potentially diverse climate change patterns across local regions, resulting in poor predictions [41]. Existing work typically learns spatial irrelevant parameters for the governing PDEs [16, 56] or assumes a universal physical system, i.e., there exists a consistent form of the governing PDEs [24, 25].

This paper presents a physics-guided machine learning approach, SVPNet, for accurate spatiotemporal predictive learning (or called future frame prediction). SVPNet aims to learn an effective physical representation by capturing noise evolution in the physical states to update the current state and modeling spatially varying physical dynamics to predict the next state. In particular, SVPNet starts with a randomly initialized physical state and utilizes an *update module* to correct the current state with the observed data. This *update module* combines the current state and observations by learning a gating factor, incorporating the previous updating information recurrently to reduce the underlying noise. The updated state is an improved representation of the current physical system that facilitates predicting the next state. Following the update module,

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

a *predict module* calculates the evolution of the updated physical state by modeling spatially varying physical dynamics. The *predict module* learns spatially varying governing PDEs, including their forms and parameters, by capturing the variation of physical states over time using an attention mechanism. Then SVPNet uses the learned governing PDEs to compute the physical state evolution to predict the next state. Experimental results show that SVPNet outperforms other state-of-the-art methods in complex spatiotemporal prediction tasks with its novel *update* and *predict module*.

The main paper contribution is a new physics-guided spatiotemporal predictive model that learns physical representation to describe the underlying spatially varying physical dynamics. The new architecture contains a novel *update module* that effectively corrects the predicted physical state from observation and a novel *predict module* that learns spatially varying governing PDEs for predicting the next state. The proposed approach does not require expert knowledge, produces physically consistent results, and can handle the important spatial variability problem that commonly exists in various real-world spatiotemporal phenomena.

2 RELATED WORK

Non-physics-based deep learning models. One popular framework for spatiotemporal predictive learning is the stacked ConvRNNs (convolutional recurrent neural networks), including ConvLSTM [51] and its variants [7, 23, 26, 27, 35, 54], due to their excellent capability of jointly modeling spatial and temporal dependencies. Recent approaches propose to use additional memory cells for propagating information across space, time, and the model layers [37, 43, 45, 46, 52] for long-term prediction. E3D-LSTM [44] effectively recalls the previous memory states in an attentive way and includes 3D convolutions that learn feature representations over a window to enhance the performance. Memory in Memory (MIM) [46] introduces more memory cells to handle stationary and non-stationary information while requiring more matrix multiplications and memory usage. MAU [8] learns an attention map to aggregate historical temporal states, aiming to broaden the temporal receptive field. SimVP [15] applies blocks of Inception modules with a UNet architecture to learn the temporal evolution. TAU [38] replaces the Inception-UNet with efficient attention modules that promote performance in video prediction tasks. These methods typically focus on effectively capturing spatial and temporal dependencies by adding, stacking, and connecting various types of memory cells, which work well when large amounts of training data are available, but their results could be physically inconsistent [18, 48].

Physics-guided deep learning models. One type of incorporation of prior physical knowledge into deep learning assumes the physical system is known. For example, some approaches explicitly model the interactions between objects in an observed scene following several basic physical laws, e.g., drift, gravity, spring [28, 50], which require explicitly modeling objects and are not suitable for complex spatiotemporal prediction tasks, e.g., climate prediction. Other approaches capture physical evolution by approximating the solution and response function of PDEs [29, 30]. Most of them address specific PDEs or a family of PDEs for a particular physical system, e.g., advection-diffusion [12], fluid dynamics [22, 42], which might not be easily extendable to other domains and are not practical for modeling complex spatiotemporal processes.

Another type of physics-guided deep learning model intends to approximate the governing PDEs in a data-driven way [1, 16, 22, 24, 25, 49, 53, 56]. For example, PDE-Net [24, 25] uses a set of convolution filters to approximate partial derivatives to form a broad class of PDEs. PhyDNet [16] further disentangles a physical space from residual factors for learning the governing PDEs and allows state correction to update state evolution. However, these methods cannot handle spatial variability in the governing PDEs, leading to poor performance when the governing PDEs are not consistent over space. STMoE [1] learns physical feature representations that can be different by location but are limited to three known physical laws. Other approaches factorize frame content into various meaningful spaces, such as content/motion decomposition [39, 40], spatiotemporal disentanglement [13], to improve the prediction tasks. They do not explicitly model the underlying physical systems.

Our paper aims to solve a real-world scenario where the PDE parameters and systems can vary across space (i.e., spatial variability) by disentangling a physical space governed by spatially varying physical dynamics. Our proposed update and predict modules can effectively learn physical representations to model the system evolution and hence learn accurate physical dynamics for spatiotemporal prediction.

3 METHODOLOGY: SVPNET

3.1 Problem Statement

This section presents the proposed approach, SVPNet, for spatiotemporal prediction. The model takes a sequence of frames (i.e., raster grids) $\{v_1, \ldots, v_T\}$ as an input, where $v_i \in \mathbb{R}^{C \times H \times W}$, C is the number of frame channels, H and W are the frame height and width, and T is the sequence length. SVPNet aims to predict a future sequence $\{\hat{v}_{T+1}, \ldots, \hat{v}_{T'}\}$ that minimizes the difference between true frames and predicted ones, where $\hat{v}_t$ denotes the predicted frame at time t and T' is the forecasting horizon. Specifically, at each target step, SVPNet takes an input frame v_t and predicts the next frame $\hat{v}_t$. During the inference period, the predicted frame serves as the input to the next iteration.

3.2 SVPNet: Overall Architecture

Figure 1 shows the overall architecture of SVPNet. Inspired by recent work [16, 53], we assume that there exists a latent space $\mathcal{H}$ that disentangles physical dynamics $\mathcal{H}^p$ and residual factors $\mathcal{H}^r$. The physical representation $h^p \in \mathcal{H}^p$ is governed by spatially varying PDEs, and the residual representation $h^r \in \mathcal{H}^r$ captures the unknown factors that cannot be described by physical dynamics $\mathcal{H}^p$. SVPNet leverages two parallel branches to model the latent spaces, i.e., $\mathcal{H}^p$ and $\mathcal{H}^r$.

Specifically, given an input frame v_t, SVPNet first embeds v_t with an encoder, denoted as $E(v_t)$. Then, SVPNet uses a physics extractor E_p to extract physical dynamics relevant features from v_t, denoted as $u_t^p = E_p(E(v_t))$. SVPNet directly computes the subtraction between $E(v_t)$ and physical features, i.e., $u_t^r = E(v_t) - u_t^p$, as the residual input, enabling the two branches to process

completely separate information with fewer parameters than using a second deep encoder to extract residual features (e.g., [16]).[1]

For the physical branch (yellow, Figure 1), SVPNet learns physical representation h^p_{t+1} from u^p_t and h^p_t using a novel module called SVPCell. SVPCell assumes that the physical representation is governed by some unknown PDEs, e.g., the heat equation, which can vary across space, e.g., the latent states at location a might evolve following a physical law differently from location b. For the residual branch (gray, Figure 1), SVPNet adopts convolutional recurrent neural networks, e.g., ConvLSTM [51], to predict h^r_{t+1} from u^r_t and h^r_t [16, 21, 56]. Lastly, two decoders D_p and D_r in SVPCell and ConvRNN, respectively, map their output into a common latent space so that their addition is the combined representation used to forecast the next frame $\hat{v}^{t+1}$ with decoder D[2], i.e.,

$$\hat{v}^{t+1} = D(D_p(h^p_{t+1}) + D_r(h^r_{t+1}))$$

The remainder of this section focuses on SVPCell, including its novel update and predict modules to learn effective physical representation to describe the spatially varying physical system for predicting physical state evolution.

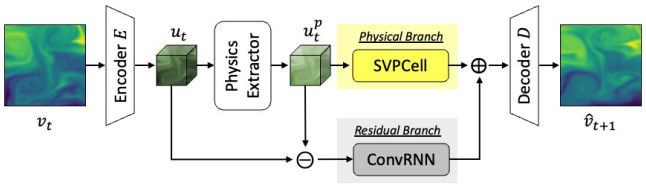


Figure 1: A two-branch architecture for disentangling physical space and residual space, where SVPCell captures spatially varying physical dynamics.

3.3 SVPCell: Learning Physical Representation

Before any observed frame, SVPCell initializes a random latent physical state, denoted as $h_{1|0}$.[3] Given the first observed physical features u_1, SVPCell produces an updated physical state, denoted as $h_{1|1}$, with a learnable gating factor that selectively incorporates relevant information from the observation (the update module, Section 3.3.1). Then SVPCell predicts the next state, denoted as $h_{2|1}$, from $h_{1|1}$ by estimating the underlying spatially varying physical system (the predict module, Section 3.3.2). Given the second observation, SVPCell updates the predicted physical state with the new input, and so forth. The process runs recurrently over frames until the last frame. Similarly, we denote u_t as the observation at time t, $h_{t|t-1}$ as the predicted prior state for time t that was made at time $t-1$, $h_{t|t}$ as the posterior state at time t after updating with the observation u_t, and $h_{t+1|t}$ as the predicted state used to forecast the $(t+1)$ frame.

[1]The appendix contains the implementation details for $E(\cdot)$ and $E_p(\cdot)$.
[2]The appendix contains the implementation details for $D(\cdot)$, $D_p(\cdot)$, and $D_r(\cdot)$.
[3]Here we drop p in h^p for simplicity

3.3.1 *The Update Module for Correcting Current Physical State.*

Figure 2 shows the architecture of the update module. The current physical state $h_{t|t-1}$ is the predicted state from the last step. The update module aims to update the predicted state given the observation u_t to compute the posterior state $h_{t|t}$ as follows,

$$h_{t|t} = (1-\mathcal{K}_t)\cdot h_{t|t-1} + \mathcal{K}_t \cdot u_t = h_{t|t-1} + \mathcal{K}_t \cdot (u_t - h_{t|t-1}) \quad (1)$$

where $\mathcal{K}_t$ is a gating factor that controls the trade-off between the predicted state and the observation.

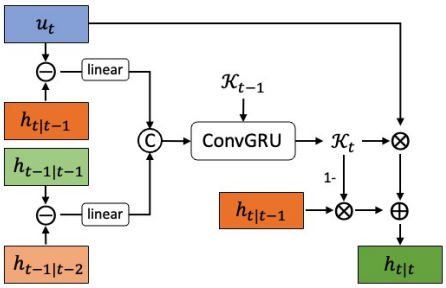


Figure 2: The detailed architecture of the update module that takes in the observation (blue), the previous (light orange) and current prior states (orange), and the previous posterior state (light green) to compute the new posterior state (green).

The update module computes $\mathcal{K}_t$ by considering how the system updates the physical states in the previous steps (i.e., from $h_{t-2|t-1}$ to $h_{t-1|t-1}$) and modeling the underlying noise evolution generated from the predict module to update the current state $h_{t|t-1}$ appropriately. The update module exploits ConvGRU [4] to learn $\mathcal{K}_t$ by taking into account the current system and previous updating information from the recurrent memory. The update module computes the following features as the input to the ConvGRU,

$$\begin{aligned} \Delta u_t &= u_t - h_{t|t-1} \\ \Delta h_{t-1} &= h_{t-1|t-1} - h_{t-1|t-2} \end{aligned} \quad (2)$$

where Δu_t is the difference between the observation and the predicted prior state, aiming to describe the information that can be potentially incorporated into the state. Δh_{t-1} is the difference between the posterior state and the prior state from the last step $t-1$, which indicates the information added to the state in the last step.

The update module first linearly maps the two features into another latent space, denoted as $\Delta\tilde{u}_t$ and $\Delta\tilde{h}_{t-1}$. Then ConvGRU takes in the two features and previous hidden state $h^{\mathcal{K}}_{t-1}$ to calculate $h^{\mathcal{K}}_t$ and $\mathcal{K}_t$,

$$\begin{aligned} h^{\mathcal{K}}_t &= \text{ConvGRU}([\Delta\tilde{u}_t, \Delta\tilde{h}_{t-1}], h^{\mathcal{K}}_{t-1}) \\ \mathcal{K}_t &= \sigma(\mathbf{W}_{\mathcal{K}} * h^{\mathcal{K}}_{t-1}) \end{aligned} \quad (3)$$

where σ is the sigmoid function and $\mathbf{W}_{\mathcal{K}}$ is the convolution kernels to output $\mathcal{K}_t$. Then the update module computes the current posterior state $h_{t|t}$ as the input to the predict module using Eq 1. The posterior state serves as an improved physical state that can be used for predicting the next state.

Distinguish from other models: The idea of incorporating observations to correct the model state is close to data assimilation,

e.g., with Kalman filter [20] that corrects the predicted system state with some measurements. However, one cannot directly apply the Kalman filter in a spatiotemporal prediction task because we do not have the measurements at time $t+1$, and the observation at time t does not contain enough information to correct the predicted state at time $t+1$. PhyDNet [16] is an example that corrects the predicted state $h_{t+1|t}$ with the observation at time t, which might not properly model the evolution process. Instead, our update module models the underlying error evolution from current and previous updating information recurrently and corrects the predicted prior state $h_{t|t-1}$ with u_t, serving as the input to the predict module.

3.3.2 Predict module for predicting the next physical states. We assume that the evolution of the physical state $h^p(t, \mathbf{s})$ is governed by some PDE that takes the following form,

$$\frac{\partial h^p(t,\mathbf{s})}{\partial t} = F_{\mathbf{s}}(h, \frac{\partial h}{\partial x}, \frac{\partial h}{\partial y}, \frac{\partial^2 h}{\partial x \partial y}, \frac{\partial^2 h}{\partial x^2}, \frac{\partial^2 h}{\partial y^2}, ..., \frac{\partial^{i+j} h}{\partial x^i \partial y^j}, ...) \quad (4)$$

where $\mathbf{s} = (x, y)$ represents spatial coordinates. The function $F_{\mathbf{s}}$ is a combination of the spatial partial derivatives up to a certain differential order q, i.e., $i + j < q$ and varies by location (i.e., $\mathbf{s}$).

Figure 3 shows the architecture of the predict module. The predict module first learns a set of convolutional filters to approximate each spatial partial derivative in Eq 4 [24, 25], parameterized by $\mathbf{W^k}$ with k^2 filters of size $k \times k$. Each filter $\mathbf{W}^{\mathbf{k}}_{i,j}$ approximates $\frac{\partial^{i+j} h}{\partial x^i \partial y^j}$, constrained by a kernel moment loss. We follow [16, 25] to compute a moment matrix from the filters $\mathcal{M}(\mathbf{W^k})$ and compare it to a target moment matrix $\mathbf{M}$ (see the computations for $\mathcal{M}$ and $\mathbf{M}$ in the Appendix), with the following equation:

$$\mathcal{L}_{\mathcal{M}}(\mathbf{W}^k) = \sum_l \sum_{i \le k} \sum_{j \le k} \mathcal{L}^2(\mathcal{M}(w^k_{l,i,j}), \mathbf{M}^{\mathbf{k}}_{i,j}) \quad (5)$$

where $\mathcal{L}^2$ is the square error and l is the number of input channels in the filter. We call these partial derivatives (each input for F_s) "physical evolution candidates". Next, our model will learn the combination function that can vary across space, i.e., enabling spatial variability in $F_{\mathbf{s}}$.

The predict module learns spatially varying $F_{\mathbf{s}}$ to form the governing PDEs by leveraging the multi-head attention mechanism to compute the coefficients in $F_{\mathbf{s}}$, denoted as $\mathbf{C_s} = [\mathbf{c}^{0,0}_{\mathbf{s}}, \mathbf{c}^{i,j}_{\mathbf{s}}, ..., \mathbf{c}^{i,j}_{\mathbf{s}}]$, for combining the physical evolution candidates. The predict module computes the coefficients, $\mathbf{C_s}$, by interacting the current physical state ($h_{t|t}$) with the historical physical states ($h_{t-\delta|t-\delta}, ..., h_{t-1|t-1}$) that describe the underlying evolution. With attention, the model learns data-dependent coefficients that vary across locations to combine these physical evolution candidates,

$$\frac{\partial h^p(t,\mathbf{s})}{\partial t} = \mathbf{c}^{0,0}_{\mathbf{s}} h + \mathbf{c}^{1,0}_{\mathbf{s}} \times \frac{\partial h}{\partial x} + \mathbf{c}^{1,0}_{\mathbf{s}} \times \frac{\partial h}{\partial y} + ..., \mathbf{c}^{i,j}_{\mathbf{s}} \frac{\partial^{i+j} h}{\partial x^i \partial y^j} \quad (6)$$

Specifically, the predict module maps the current physical state $h_{t|t} \in \mathbb{R}^{H' \times W'}$ into query $\mathbf{Q}$ and maps the previous physical states $h_{t-\delta|t-\delta} : h_{t|t}$ into key $\mathbf{K} \in \mathbb{R}^{\delta \times H' \times W'}$ and value $\mathbf{V}$ via 1×1 convolutions. The variable δ is a hyper-parameter describing the number of previous steps to consider for learning the coefficients in the governing PDEs. We add 2D spatial encoding to $\mathbf{Q}$, and 3D spatial encoding to $\mathbf{K}$ and $\mathbf{V}$. Then the predict module computes the attention score between each component q_i in $\mathbf{Q}$ and each component

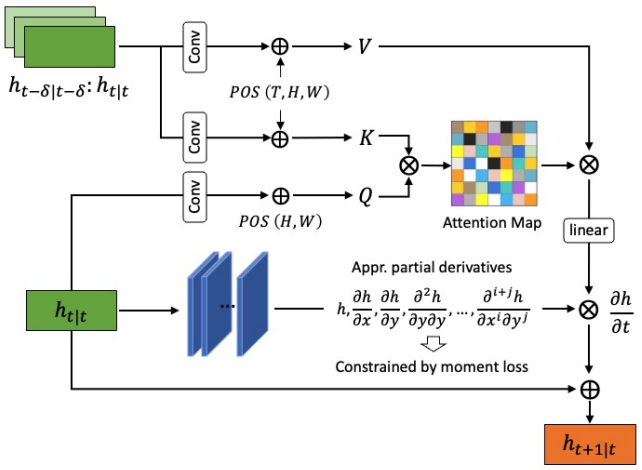


Figure 3: The detailed architecture of predict module that predicts $h_{t+1|t}$ from $h_{t|t}$ by estimating spatially varying governing PDEs.

k_j in $\mathbf{K}$, denoted as $\alpha_{i,j}$,

$$\alpha_{i,j} = \frac{\exp \mathbf{e}_{i,j}}{\sum_{k=1}^{\delta \times H' \times W'} \exp \mathbf{e}_{i,k}}$$

where $\mathbf{e} = \mathbf{Q}^T \mathbf{K} / \sqrt{d_K} \in \mathbb{R}^{[H' \times W'] \times [\delta \times H' \times W']}$, d_K is the hidden size, and $H' \times W'$ are the spatial size. Next, the predict module calculates the coefficients for location $\mathbf{s}$ by multiplying the attention with value $\mathbf{V}$,

$$\mathbf{C_s} = Linear(\sum_k \alpha_{\mathbf{s},k} \times \mathbf{V}_k)$$

where $k \in \{1, 2, ..., \delta \times H' \times W'\}$. Finally, the predict module calculates the state evolution using Eq 6 and predict the next physical state as the output of physical branch,

$$h_{t+1|t} = h_{t|t} + \frac{\partial h^p}{\partial t}$$

Distinguish from other models: One significant difference between SVPNet and others is that our predict module learns spatially varying governing PDEs to compute state evolution. Other methods are not as powerful and flexible for describing the physical dynamics as SVPNet. For example, PhyDNet [16] only learns a spatial irrelevant physical system, and PDE-Net [25] directly works on raw data space without updates from observations. Also, the flow model [12] approximates the advection-diffusion equation but is limited to this specific PDE.

3.4 Training

Let us denote SVPNet parameters as $\mathbf{W}$ and $\mathbf{W^k}$ are the filters approximating k^2 partial derivatives in Section 3.3.1. Given a training data sample $\mathcal{D} = \{v_1, \ldots, v_T\}$, our goal is to minimize the following objective function:

$$\mathcal{L} = \mathcal{L}^2(\mathcal{D}, \mathbf{W}) + \lambda \mathcal{L}_{\mathcal{M}}(\mathbf{W^k})$$

where $\mathcal{L}^2$ is the pixel-wise error between the predicted and ground truth frames, and $\mathcal{L}_{\mathcal{M}}(\mathbf{W^k})$ is the moment loss in Eq 5.

4 EXPERIMENTS AND RESULTS

4.1 Datasets

We evaluate SVPNet on three commonly-used public real-world datasets from various domains, including two for natural processes and one for human activities. We also develop one synthetic dataset with evolution potentially governed by spatial-varying physical dynamics. We summarize the statistics of the datasets in Table 1.

4.1.1 Sea Surface Temperature (SST). The SST dataset describes the sea surface temperatures of the Atlantic Ocean generated using the state-of-the-art simulation engine NEMO and then reanalyzed by ERA5 for recent years.[4] We collect the SST data containing the daily mean temperature of the North Atlantic Ocean region from 2008 to 2017, whose evolution is governed by the physical laws of fluid dynamics. The entire acquisition region covers an area of 301×661 grid cells. We extract 33 sub-regions of size 64×64 cells and remove the areas covered by lands or with missing values. We normalize the data in the same way as in previous work [12, 13]. Each sub-region is first normalized by the mean and standard deviation of the temperatures in a sub-region computed from all days with the same date of the available training years. This removes the seasonal component in the SST data. For example, the SST of sub-region "A" on May 19th, 2010, is normalized using all the May 19th available acquisitions in "A". Then each sub-region is normalized into the scale between 0 and 1 and fed to the tested models. The predictions are reverted to the original temperature measurement space for evaluation, similar to [13]. We use the data from 2008 to 2015 for training and from 2016 to 2017 for testing. The task is to use the past 4 frames to predict the next 6 frames, following the previous work [12, 16].

4.1.2 Turbulent [42]. The turbulent dataset describes the turbulent flow velocity along the x and y direction simulated using the Lattice Boltzmann Method [9]. Following [42], we crop the original image (1792×256) to seven square sub-regions of size 256×256 and downsample them into the size of 64×64. The task is to predict the velocity fields up to 60 steps ahead given 10 initial frames [42].

4.1.3 Taxi BJ [55]. The traffic flow dataset [55] describes the inflow and outflow of taxis in Beijing from July 2013 to April 2016, where each image represents a square region (of 32×32 pixels) of the city. Each pixel contains the traffic flow entering and leaving that district in 30 minutes. This dataset represents a complex system, which might be governed by multiple physical systems but can be highly impacted by frequent ad-hoc events (e.g., traffic accidents). We split the data from 2013 to 2015 for training and the year 2016 for testing. We perform a min-max normalization on the data to the [0, 1] range. The evaluation scores are computed in the [0,1]-normalized space, consistent with [13, 16]. We use 4 known frames (2 hours) to predict the next 4 frames (2 hours) [16, 23, 55].

4.1.4 Moving MNIST+. The Moving MNIST benchmark [36] is a synthetic video dataset generated from a simple physical system (only involving translation operation), consisting of two random digits bouncing from the walls with the same translation speed in a 64×64 grid. To examine the model performance in capturing spatially varying physical laws, we create Moving MNIST+ containing three random digits where two digits translate with different speeds (no bouncing) and one digit rotates. We predict 8 future frames given 4 past frames for Moving MNIST+ since digits might move out of the canvas after several steps. Training sequences are generated on the fly, and the testing set is fixed.

[4] https://resources.marine.copernicus.eu/product-detail/GLOBAL_MULTIYEAR_PHY_001_030/INFORMATION

Table 1: Dataset statistics. N_{train} and N_{test} are the number of training and testing samples, (C, H, W) is the input frame shape, T is the input sequence length, and T' is the forecasting horizon.

	N_{train}	N_{test}	(C, H, W)	T	T'
SST	65,373	4,000	(1,64,64)	4	6
Turbulent	6,000	2,100	(2,64,64)	10	60
TaxiBJ	19,617	1,344	(2,32,32)	4	4
Moving MNIST+	10,000	5,000	(1,64,64)	4	8

4.2 Experimental Setup

4.2.1 Baselines and Network Design. We compare SVPNet against the state-of-the-art methods, including non-physics-based methods: ConvLSTM [51], MIM [46], MAU [8], SimVP [15], TAU [38], and a physics-guided method: PhyDNet [16]. SVPNet and baseline models (ConvLSTM, MIM, MAU, and PhyDNet) share the same encoder structure to extract deep features from frames and the same decoder structure to generate predicted frames from the latent space. We keep the encoder and decoder of SimVP and TAU the same as their original paper since they directly input and output a block of frames, while other methods process images and generate predictions frame by frame. For a fair comparison, all tested models use three layers of their backbone network with the same hidden size. The physical branch of SVPNet and PhyDNet contains one layer that uses a 7×7 kernel filter for approximating 49 spatial partial derivatives. More details on network design are in the Appendix.

4.2.2 Training settings. We train all models using one RTX A5000 GPU with 32G memory. We adopt the scheduled sampling strategy [5] and early stopping on validation sets with the patience of 20 epochs. We set the batch size as 32, the initial learning rate as 0.001, the hyper-parameter λ as 1 [16], and use the Adam optimizer.

4.2.3 Evaluation metrics. We use the evaluation metrics commonly adopted in the state-of-the-art predictive methods: the Mean Squared Error (MSE), Mean Absolute Error (MAE), and Structure Similarity (SSIM) [47]. MSE and MAE estimate the absolute pixel-wise errors, and SSIM measures the similarity of structural information within the spatial neighborhoods. The lower MSE and MAE and higher SSIM indicate better performance. We compute the average metrics for each frame of the output sequence.

4.3 Overall Performance

Table 2 shows that SVPNet outperforms non-physics-based methods with an improvement of at least 6% MSE on SST and 2% on Turbulent, demonstrating the effectiveness of incorporating physics in modeling nature processes. Figure 4 presents examples of visual

Table 2: Quantitative result comparison. The numbers with * are from the original or cited papers. The SST predictions are reverted to the original temperature measurement space while Turbulent and TaxiBJ predictions remain in the scale between 0 and 1 for easy comparison since the original data values are large.

	Natural Processes						Human Activities			Synthetic Systems		
	SST			Turbulent			TaxiBJ			Moving MNIST+		
Method	MSE/10	MAE/10	SSIM	MSE	MAE	SSIM	MSE*100	MAE	SSIM	MSE*100	MAE	SSIM
ConvLSTM [51]	34.2	81.9	0.949	17.5	275.6	0.940	48.5*	19.7*	0.972*	46.7	114.1	0.886
MIM [46]	32.9	79.0	0.955	17.1	271.4	0.941	42.9*	16.6*	0.971*	40.5	110.8	0.906
MAU [8]	31.5	77.2	0.971	16.9	274.7	0.942	42.1	16.3	0.982	36.7	100.9	0.916
SimVP [15]	32.0	78.6	0.969	15.2	255.4	0.948	43.4*	16.2*	0.982*	34.7	95.4	0.920
TAU [38]	31.8	77.6	0.972	14.2	247.5	0.950	**40.4**	**16.1**	**0.983**	33.6	94.9	0.923
PhyDNet [16]	31.2	76.8	0.972	16.6	266.6	0.943	41.9*	16.2*	0.982*	35.8	98.9	0.918
SVPNet	**29.8**	**75.1**	**0.976**	**13.9**	**244.4**	**0.951**	41.6	16.2	0.982	**31.8**	**90.0**	**0.929**
↓ Error (%)	4.8	2.3	-	2.2	1.3	-	-2.8	-0.6	-	5.7	5.4	-

comparison of the predicted frames between SVPNet and TAU on SST and Turbulent. We observe that TAU tends to have blurry results and fails to capture spatial details in the early frames for both datasets, which is a typical problem for non-physics-based methods. For example, TAU does not correctly predict the movement of the dark region in 4(a), and the bright spot at the top in 4(b) is misaligned with the ground truth at t=16. In contrast, SVPNet accurately predicts frames by modeling the underlying physical dynamics from data. Furthermore, SVPNet consistently performs better across all datasets than PhyDNet, demonstrating its effectiveness in modeling spatial-varying physical dynamics. Figure 5 visually depicts the enhanced prediction accuracy of SVPNet compared to PhyDNet, e.g., SVPNet successfully captures the evolution of the bright spot on the top-left.

Since TaxiBJ's underlying physical system is not explicit and can be disrupted by ad-hoc events, capturing spatiotemporal dependencies without modeling potential physical phenomena might be enough to produce good prediction results when the training samples are sufficient, and the training and testing samples are in the same distribution. Thus, the advantages of SVPNet are not obvious, but SVPNet still outperforms the majority of the baselines in Table 2. Figure 6 shows that SVPNet can produce reliable frames even when the input frames significantly differ from the target frames. While there are minor differences, particularly along the diagonal main road, the overall trend is closely aligned with the ground truth. Also, the prediction errors of SVPNet are higher than TAU in the first frame but lower in the third and fourth frames, showing the success of the update module in SVPNet for effectively correcting the physical state.

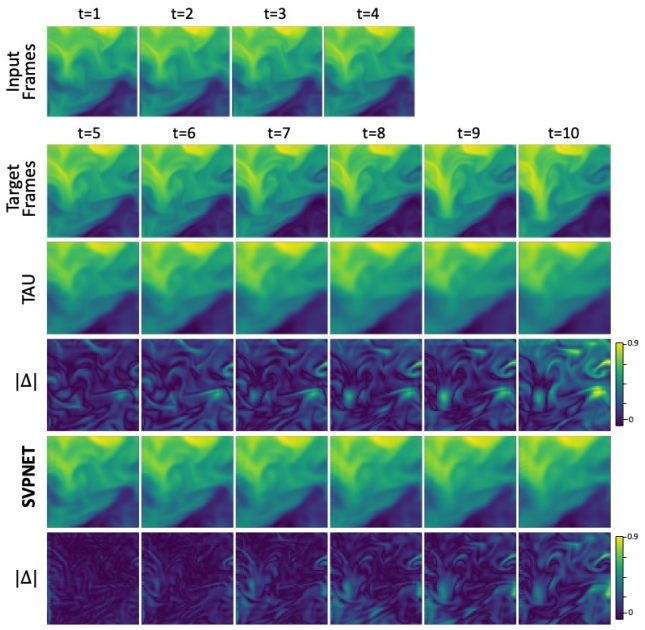
For Moving MNIST+, SVPNet performs best since the physical system contains significantly different physical laws varying across space. Specifically, SVPNet significantly outperforms PhyDNet since PhyDNet learns universal governing PDEs over space and cannot handle well the physical systems in Moving MNIST+. SVPNet overcomes the problem by learning spatially varying governing PDEs, resulting in the best performance. Figure 7 shows the qualitative visualizations of the predicted results. SVPNet successfully predicts the moving behaviors of the digits (for example, "6" and "7" are translating and "8" is rotating) even when the digits are overlapping at the beginning. In comparison, the predicted digits in TAU become blurry in complex scenarios, (for example, "9" in the second example fades quickly when separating from other digits) .

4.4 Effect of Number of Training Samples

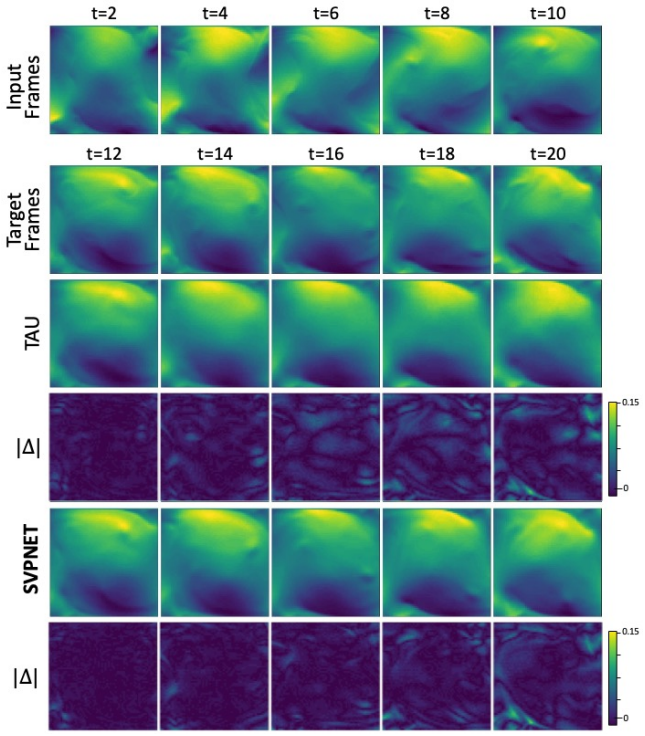
We examine the model performance using a real-world scenario when only limited training data are available. Specifically, we randomly select varying percentages of the original TaxiBJ training samples to create new training sets, namely 15% (3,000 samples), 25% (5,000 samples), 35% (5,000 samples), and 51% (10,000 samples). Table 3 compares SVPNet , PhyDNet, and TAU, considering the various training sample sizes. As expected, the results show that TAU performs much worse than the physics-guided methods when the training set is relatively small (15%-35%). However, as the number of training samples increases (>50%), TAU performs better in capturing spatiotemporal dependencies in a data-driven manner. These findings demonstrate the requirement of a substantial volume of training data for the data-driven methods without modeling the underlying physical processes and the benefits of physics-guided approaches even with limited training data.

4.5 Performance at Varying Horizons

Figure 8 shows the variation of MSE as the prediction horizon grows from 0 to 60 steps ahead in the case of Turbulent. We observe that at the early prediction stage (t < 5), all the models have close performance as the frame variation might be small in the turbulent flow. When the horizon increases, the forecasting errors of baseline methods generally grow faster than SVPNet . For example, the MSE values increase more than 15This observation highlights that capturing spatially varying physical dynamics in SVPNet enables more stable prediction than non-physics-based methods or PhyDNet that learns spatial irrelevant governing PDEs. Figure 9 visualizes the predicted frames at different forecasting horizons. Initially, there are minimal differences between the predicted and ground truth frames. As time passes, the differences gradually increase due to the loss of certain details in the predicted frames. Nevertheless, the overall trend of the predicted frames is aligned with the ground truth frames. For instance, the movement of the bright region is accurately captured, illustrating the robustness of SVPNet in predicting the underlying physical dynamics.



(a) SST Prediction



(b) Turbulent Prediction

Figure 4: Qualitative visualization of predicted results from SVPNet and TAU on SST and Turbulent

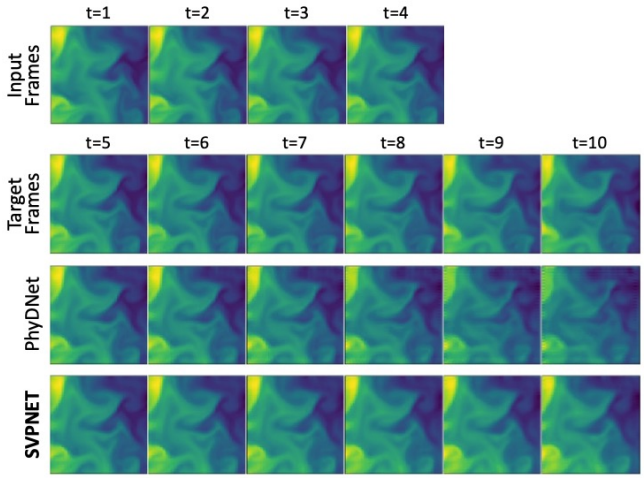


Figure 5: Qualitative visualization of predicted results from SVPNet and PhyDNet on SST.

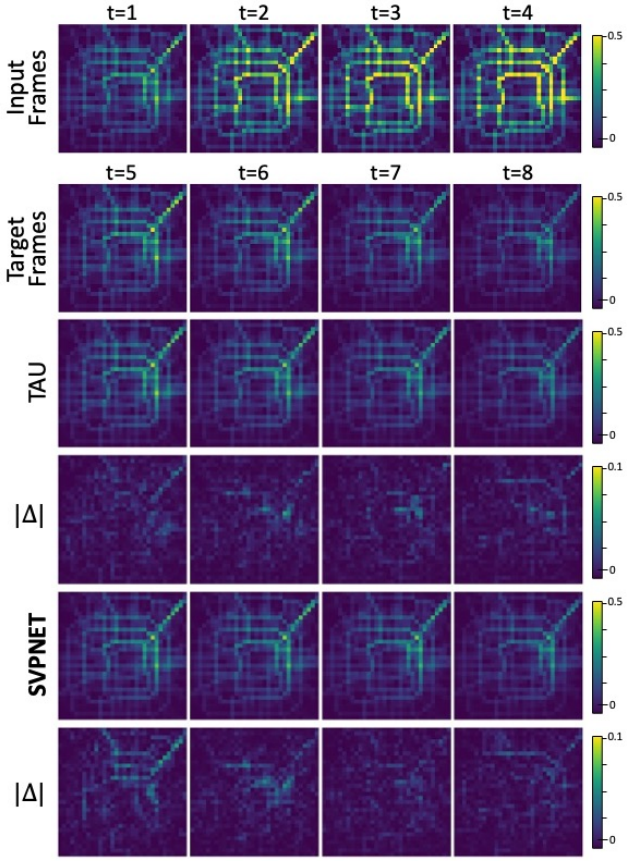


Figure 6: Qualitative visualization of predicted results on TaxiBJ dataset. The last two row presents the differences between the ground truth and the predicted frames in SVPNet and TAU.

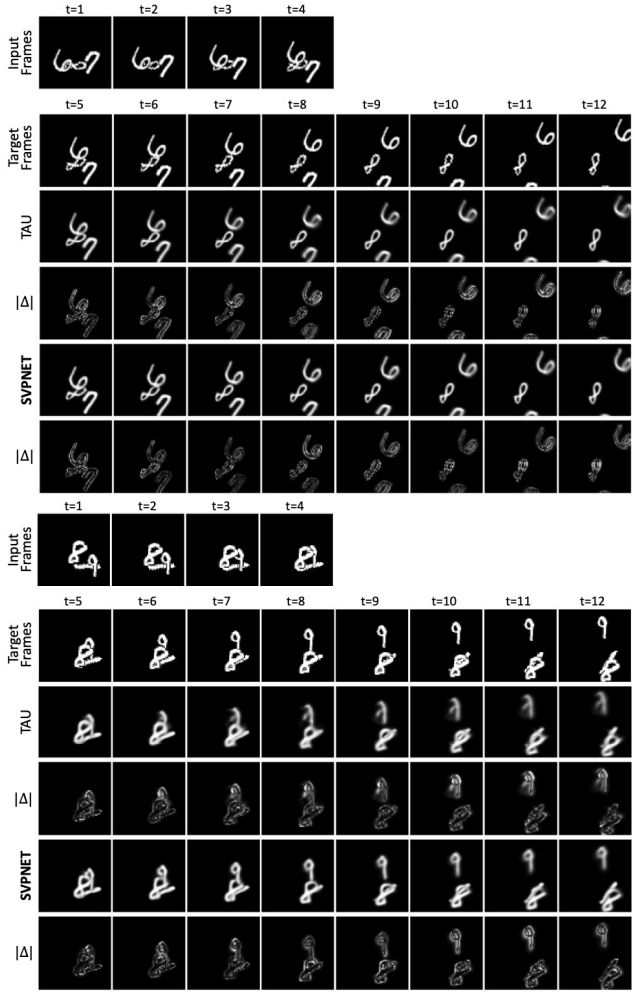


Figure 7: Qualitative visualization of predicted results on Moving MNIST+.

Table 3: Quantitative results (MSE×100) on examining the influence of the amount of training samples using TaxiBJ.

	15%	25%	35%	51%
PhyDNet [16]	49.6	46.3	46.1	45.6
TAU [38]	52.3	47.6	47.1	**43.4**
SVPNet	**49.5**	**45.8**	**45.8**	43.7

4.6 Effect of Modeling Spatial-Varying Physics

We analyze the physics and residual output from the physical and residual branches in SVPNet and PhyDNet. We use Moving MNIST+ since the physical laws are explicit and controlled in this synthetic dataset. Figure 10 presents the reconstructed frames from the two branches of SVPNet and PhyDNet. The physical branch aims to capture the physical dynamics of the digits, while the residual branch compensates for some information about the digits. Theoretically, we expect the physical branch contains as much information as possible if the physical laws governing the digit movement can be fully captured. We observe that the physical branch of SVPNet mostly predicts the physical dynamics of the translating digits, while the residual branch only shows the skeleton of the rotating digit "0". In comparison, PhyDNet requires the residual branch to retain additional details about the digits since the physical branch predicts the range of digit movements. This experiment shows that the physical branch of SVPNet can capture better physical dynamics (i.e., spatially varying PDEs) than PhyDNet, even though they produce similar predictions in this sample.

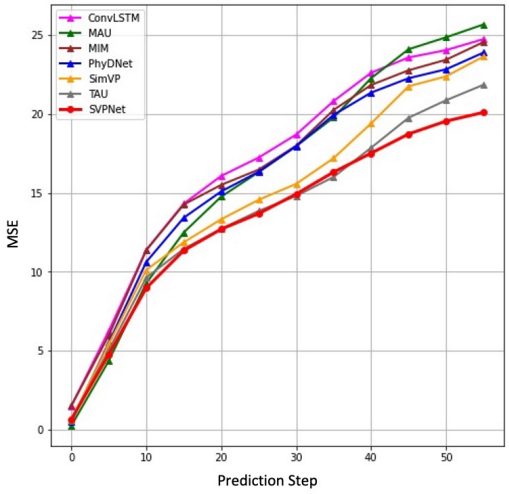


Figure 8: MSE of SVPNet and baselines' predictions at varying forecasting horizons

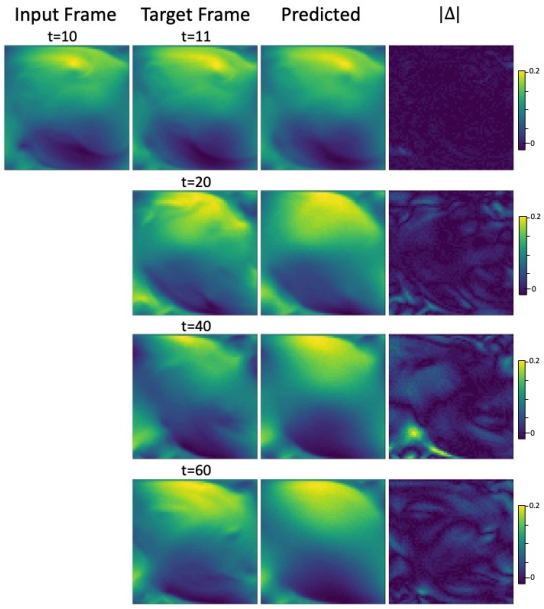


Figure 9: Qualitative visualization of predicted results at varying forecasting horizons. The last column presents the differences between the ground truth and the predicted frames.

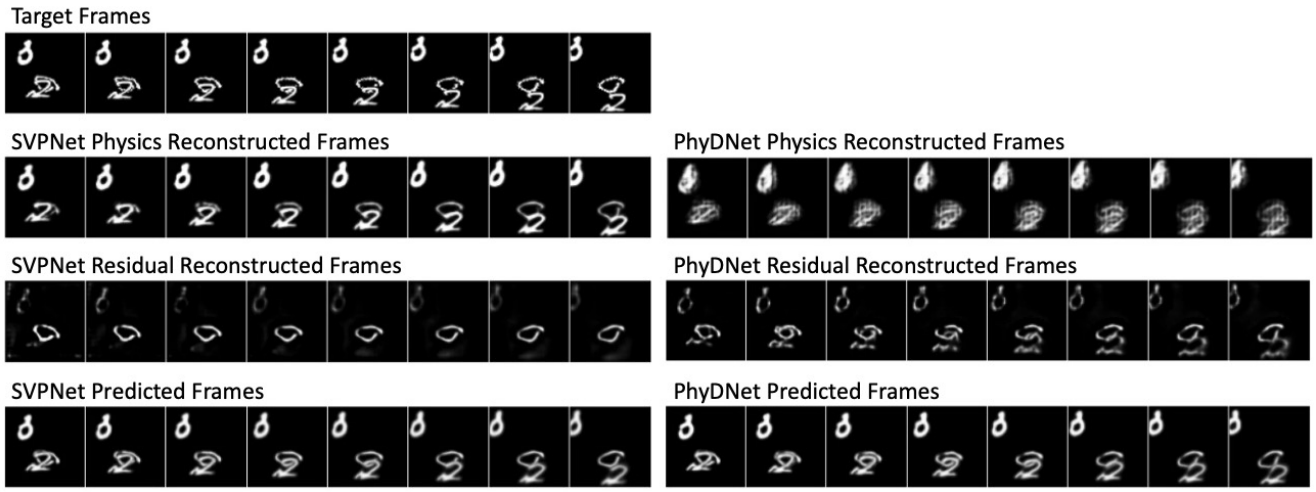


Figure 10: Visual comparison between reconstructions from physical and residual branches

4.7 Ablation Study

4.7.1 Influence of the Predict Module. We conduct an ablation study to analyze the predict module for learning the governing PDEs with spatially varying coefficients. We compare SVPNet to SVPNet-A that replaces the predict module with a linear regression layer parameterized by $c_{0,0}$, $c_{1,0}$, ..., $c_{i,j}$, and combines the partial derivatives to learn spatial irrelevant governing PDEs, $\frac{\partial h}{\partial t} = c_{0,0}h + c_{1,0}\frac{\partial h}{\partial x} + ... + c_{i,j}\frac{\partial^{i+j}h}{\partial x^i \partial y^j}$. Table 4 shows that the predict module largely improves the performance across all datasets, demonstrating the effectiveness of the proposed predict module that learns the governing PDEs with spatially varying coefficients as the underlying physical system.

4.7.2 Influence of the Update Module. We further analyze the update module for physical state correction. We compare SVPNet to (1) SVPNet-B that removes the update module, i.e., there is no update on the predicted physics state from the previous step, and (2) SVPNet-C that computes the gating factor by considering only the interaction between current physical input u_t and the last physical state $h_{t|t-1}$, i.e., learning convolution kernels $\mathbf{W}_u$, $\mathbf{W}_h$ and bias $\mathbf{b}$ to combine two variables $\mathcal{K}_t = \sigma(\mathbf{W}_u * u_t + \mathbf{W}_h * h_{t|t-1} + \mathbf{b})$. Table 4 shows that the prediction accuracy drops drastically without the update module, and SVPNet achieves consistently better performance than SVPNet-C, indicating that incorporating previous updating information can effectively correct the predicted physical states for computing the physics evolution.

Table 4: Quantitative results (MSE) on four datasets for (1) examining the influence of the predict module (SVPNet-A); (2) demonstrating the effectiveness of the update module (SVPNet-B and SVPNet-C).

	SST	Turbulent	TaxiBJ	Moving MNIST+
SVPNet -A	31.6	15.9	42.3	35.6
SVPNet -B	35.3	16.7	44.7	38.7
SVPNet -C	30.5	14.6	42.8	32.9
SVPNet	**29.8**	**13.9**	**41.6**	**31.8**

5 CONCLUSION, LIMITATIONS, AND FUTURE WORK

This paper proposes SVPNet, a spatiotemporal predictive learning method that learns spatial-varying governing PDEs for predicting physical states. Extensive experiments show that SVPNet can perform accurate frame forecasting for complex spatiotemporal prediction tasks. This work has limitations in interpreting how PDEs govern the state evolution and their changes over space and time. Thus, our future work include adding interpretability to the formation of the governing PDEs, and also extending to probabilistic forecasts with uncertainty estimation in a generative way [11]. Incorporating physics for long-term prediction would be another future work direction in spatiotemporal prediction.

REFERENCES

[1] Yuka Aoyagi, Noboru Murata, and Hidetomo Sakaino. 2021. Spatio-temporal predictive network for videos with physical properties. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2268–2278.
[2] Mark Asch, Marc Bocquet, and Maëlle Nodet. 2016. *Data assimilation: methods, algorithms, and applications*. SIAM.
[3] Heike Bach and Wolfram Mauser. 2003. Methods and examples for remote sensing data assimilation in land surface process modeling. *IEEE Transactions on Geoscience and Remote Sensing* 41, 7 (2003), 1629–1637.
[4] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. 2015. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432* (2015).
[5] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems* 28 (2015).
[6] Marc Bocquet, Julien Brajard, Alberto Carrassi, and Laurent Bertino. 2019. Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models. *Nonlinear Processes in Geophysics* 26, 3 (2019), 143–162.
[7] Zenghao Chai, Chun Yuan, Zhihui Lin, and Yunpeng Bai. 2021. CMS-LSTM: context-embedding and multi-scale spatiotemporal-expression LSTM for video prediction. *arXiv preprint arXiv:2102.03586* (2021).
[8] Zheng Chang, Xinfeng Zhang, Shanshe Wang, Siwei Ma, Yan Ye, Xiang Xinguang, and Wen Gao. 2021. MAU: A Motion-Aware Unit for Video Prediction and Beyond. *Advances in Neural Information Processing Systems* 34 (2021).

[9] Dragos Bogdan Chirila. 2018. *Towards lattice Boltzmann models for climate sciences: The GeLB programming language with applications.* Ph. D. Dissertation. Universität Bremen.

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[11] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. 2019. Addressing failure prediction by learning model confidence. *Advances in Neural Information Processing Systems* 32 (2019).

[12] Emmanuel De Bézenac, Arthur Pajot, and Patrick Gallinari. 2019. Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment* 2019, 12 (2019), 124009.

[13] Jérémie Donà, Jean-Yves Franceschi, Sylvain Lamprier, and Patrick Gallinari. 2020. Pde-driven spatiotemporal disentanglement. *arXiv preprint arXiv:2008.01352* (2020).

[14] Shen Fang, Qi Zhang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2019. GSTNet: Global Spatial-Temporal Network for Traffic Flow Prediction.. In *IJCAI*. 2286–2293.

[15] Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z Li. 2022. Simvp: Simpler yet better video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3170–3180.

[16] Vincent Le Guen and Nicolas Thome. 2020. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11474–11484.

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

[18] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan S. Read, Jacob A. Zwart, Michael Steinbach, and Vipin Kumar. 2021. Physics-Guided Machine Learning for Scientific Discovery: An Application in Simulating Lake Temperature Profiles. *ACM/IMS Trans. Data Sci.* 2, 3, Article 20 (may 2021), 26 pages. https://doi.org/10.1145/3447814

[19] Xiaowei Jia, Jacob Zwart, Jeffrey Sadler, Alison Appling, Samantha Oliver, Steven Markstrom, Jared Willard, Shaoming Xu, Michael Steinbach, Jordan Read, et al. 2021. Physics-guided recurrent graph model for predicting flow and temperature in river networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 612–620.

[20] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. (1960).

[21] Vincent Le Guen and Nicolas Thome. 2020. A deep physical model for solar irradiance forecasting with fisheye images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 630–631.

[22] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2020. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895* (2020).

[23] Zhihui Lin, Maomao Li, Zhuobin Zheng, Yangyang Cheng, and Chun Yuan. 2020. Self-attention convlstm for spatiotemporal prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 11531–11538.

[24] Zichao Long, Yiping Lu, and Bin Dong. 2019. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *J. Comput. Phys.* 399 (2019), 108925.

[25] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. 2018. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*. PMLR, 3208–3216.

[26] William Lotter, Gabriel Kreiman, and David Cox. 2016. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104* (2016).

[27] Chuyao Luo, Xutao Li, and Yunming Ye. 2020. PFST-LSTM: A spatiotemporal LSTM model with pseudoflow prediction for precipitation nowcasting. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14 (2020), 843–857.

[28] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li F Fei-Fei, Josh Tenenbaum, and Daniel L Yamins. 2018. Flexible neural representation for physics prediction. *Advances in neural information processing systems* 31 (2018).

[29] Maziar Raissi. 2018. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research* 19, 1 (2018), 932–955.

[30] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. 2017. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561* (2017).

[31] Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, and Nuno Carvalhais. 2019. Deep learning and process understanding for data-driven Earth system science. *Nature* 566, 7743 (2019), 195–204.

[32] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. 2017. Data-driven discovery of partial differential equations. *Science advances* 3, 4 (2017), e1602614.

[33] Sungyong Seo and Yan Liu. 2019. Differentiable physics-informed graph networks. *arXiv preprint arXiv:1902.02950* (2019).

[34] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. 2017. Deep learning for precipitation nowcasting: A benchmark and a new model. *Advances in neural information processing systems* 30 (2017).

[35] Yania Molina Souto, Fabio Porto, Ana Maria Moura, and Eduardo Bezerra. 2018. A spatiotemporal ensemble approach to rainfall forecasting. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[36] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using lstms. In *International conference on machine learning*. PMLR, 843–852.

[37] Jiahao Su, Wonmin Byeon, Jean Kossaifi, Furong Huang, Jan Kautz, and Anima Anandkumar. 2020. Convolutional tensor-train lstm for spatio-temporal learning. *Advances in Neural Information Processing Systems* 33 (2020), 13714–13726.

[38] Cheng Tan, Zhangyang Gao, Lirong Wu, Yongjie Xu, Jun Xia, Siyuan Li, and Stan Z Li. 2023. Temporal attention unit: Towards efficient spatiotemporal predictive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18770–18782.

[39] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. 2018. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1526–1535.

[40] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. 2017. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033* (2017).

[41] Daniel Walton, Neil Berg, David Pierce, Ed Maurer, Alex Hall, Yen-Heng Lin, Stefan Rahimi, and Dan Cayan. 2020. Understanding differences in California climate projections produced by dynamical and statistical downscaling. *Journal of Geophysical Research: Atmospheres* 125, 19 (2020), e2020JD032812.

[42] Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. 2020. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1457–1466.

[43] Yunbo Wang, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and S Yu Philip. 2018. Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In *International Conference on Machine Learning*. PMLR, 5123–5132.

[44] Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. 2018. Eidetic 3d lstm: A model for video prediction and beyond. In *International conference on learning representations*.

[45] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. 2017. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. *Advances in neural information processing systems* 30 (2017).

[46] Yunbo Wang, Jianjin Zhang, Hongyu Zhu, Mingsheng Long, Jianmin Wang, and Philip S Yu. 2019. Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9154–9162.

[47] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[48] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. 2020. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919* 1, 1 (2020), 1–34.

[49] Hao Wu, Wei Xion, Fan Xu, Xiao Luo, Chong Chen, Xian-Sheng Hua, and Haixin Wang. 2023. PastNet: Introducing Physical Inductive Biases for Spatio-temporal Video Prediction. *arXiv preprint arXiv:2305.11421* (2023).

[50] Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. 2017. Learning to see physics via visual de-animation. *Advances in Neural Information Processing Systems* 30 (2017).

[51] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NIPS*. 802–810.

[52] Ziru Xu, Yunbo Wang, Mingsheng Long, Jianmin Wang, and M KLiss. 2018. PredCNN: Predictive Learning with Cascade Convolutions.. In *IJCAI*. 2940–2947.

[53] Yuan Yin, Vincent Le Guen, Jérémie Dona, Emmanuel de Bézenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. 2021. Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment* 2021, 12 (2021), 124012.

[54] Wei Yu, Yichao Lu, Steve Easterbrook, and Sanja Fidler. 2020. Efficient and information-preserving future frame prediction and beyond. (2020).

[55] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-first AAAI conference on artificial intelligence*.

[56] Nir Ben Zikri and Andrei Sharf. 2022. PhyLoNet: Physically-Constrained Long-Term Video Prediction. In *Asian Conference on Computer Vision*. Springer, 570–587.

A APPENDIX

A.1 Model Architecture

For a fair comparison, SVPNet and baseline models share the same encoder structure to extract features from frames and the same decoder structure to generate predicted frames from the latent space. We provide the architecture of the encoder and decoder for all datasets in Table 5, 6, and 7. We decide the number of convolutional layers and the hidden dimension based on the image size and its complexity. We define a conv-block, composed of an input convolution layer (Conv2d or ConvTranspose2d), GroupNorm, and LeakyRelu:

conv-block(conv-layer) = [conv-layer, GroupNorm(16), LeakyRelu(0.2)]

where Conv2d(in_channels, out_channels, kernel_size, stride, padding) and ConvTranspose2d(in_channels, out_channels, kernel_size, stride, padding, output_padding=0)

Table 5: Encoder and decoder for Moving MNIST+

Encoder
conv-block(Conv2d(1, 16, 3, 1, 1))
conv-block(Conv2d(16, 32, 3, 2, 1))
conv-block(Conv2d(32, 32, 3, 1, 1))
conv-block(Conv2d(32, 64, 3, 2, 1))
conv-block(Conv2d(64, 64, 3, 1, 1))
Decoder
conv-block(ConvTranspose2d(64, 64, 3, 1, 1))
conv-block(ConvTranspose2d(64, 32, 3, 2, 1, 1))
conv-block(ConvTranspose2d(32, 32, 3, 1, 1))
conv-block(ConvTranspose2d(32, 16, 3, 2, 1, 1))
ConvTranspose2d(16, 1, 3, 1, 1)

Table 6: Encoder and decoder for SST and Turbulent

Encoder
conv-block(Conv2d(1, 32, 3, 1, 1))
conv-block(Conv2d(32, 64, 3, 2, 1))
conv-block(Conv2d(64, 64, 3, 1, 1))
conv-block(Conv2d(64, 128, 3, 2, 1))
conv-block(Conv2d(128, 128, 3, 1, 1))
Decoder
conv-block(ConvTranspose2d(128, 128, 3, 1, 1))
conv-block(ConvTranspose2d(128, 64, 3, 2, 1, 1))
conv-block(ConvTranspose2d(64, 64, 3, 1, 1))
conv-block(ConvTranspose2d(64, 32, 3, 2, 1, 1))
ConvTranspose2d(32, 1, 3, 1, 1)

We design the physics extractor E_p as a deep encoder containing two layers of conv-block in Table 8. We design D_p and D_r with two layers of conv-block in Table 9.

A.2 Moment Matrix

To approximate partial derivatives (i.e., as the physical evolution candidates) using convolutions, we refer to [16, 25] to compute a moment matrix $\mathcal{M}(\mathbf{W^k})$ from the filters of size $k \times k$,

$$\mathcal{M}(\mathbf{W^k})_{i,j} = \frac{1}{i!j!} \sum_{u=-\frac{k-1}{2}}^{\frac{k-1}{2}} \sum_{v=-\frac{k-1}{2}}^{\frac{k-1}{2}} u^i v^j \mathbf{k}[u,v] \tag{7}$$

where $i, j = 0, 1, ..., k-1$ and $\mathcal{M}(\mathbf{W^k}) \in \mathbb{R}^{k^2 \times k \times k}$.

We can approximate the partial derivative of an order (decided by i and j) by imposing constraints on the moment matrix $\mathcal{M}(\mathbf{W^k})_{i,j}$ [16, 25]. For example, to approximate the partial derivative of $\frac{\partial^{a+b}}{\partial x^a \partial y^b}(.)$, we should impose $\mathcal{M}(\mathbf{W^k})_{i,j} = 0$ for $i \neq a$ and $j \neq b$. Thus, we denote a target moment matrix $\mathbf{M}^k_{i,j}$ of size $k \times k$, where all elements equal 0 except position (i, j) is 1. Then we compute the difference between the moment matrix and the target matrix with the Frobenius norm as the loss:

$$\mathcal{L}_{\mathcal{M}}(\mathbf{w_p}) = \sum_l \sum_{i \leq k} \sum_{j \leq k} ||(\mathcal{M}(w^k_{l,i,j}), \mathbf{M}^k_{i,j}||_F \tag{8}$$

where l is the number of input channels in the filters.

Table 7: Encoder and decoder for TaxiBJ

Encoder
conv-blockConv2d((2, 32, 3, 1, 1))
conv-block(Conv2d(32, 64, 3, 2, 1))
conv-block(Conv2d(64, 128, 3, 1, 1))
Decoder
conv-block(ConvTranspose2d(128, 64, 3, 1, 1))
conv-block(ConvTranspose2d(64, 32, 3, 2, 1, 1))
ConvTranspose2d(32, 2, 3, 1, 1)

Table 8: The structure of physics extractor E_p

conv-block(Conv2d(64, 64, 3, 1, 1))
conv-block(Conv2d(64, 64, 3, 1, 1))

Table 9: The structure of decoders, D_p and D_r

conv-block(ConvTranspose2d(64, 64, 3, 1, 1))
conv-block(ConvTranspose2d(64, 64, 3, 1, 1))