Disentangling Physical Dynamics from Unknown Factors for Unsupervised Video Prediction

Vincent Le Guen 1,2, Nicolas Thome 2
1 EDF R&D, Chatou, France
2 CEDRIC, Conservatoire National des Arts et Métiers, Paris, France

Abstract

Leveraging physical knowledge described by partial differential equations (PDEs) is an appealing way to improve unsupervised video prediction methods. Since physics is too restrictive for describing the full visual content of generic videos, we introduce PhyDNet, a two-branch deep architecture, which explicitly disentangles PDE dynamics from unknown complementary information. A second contribution is to propose a new recurrent physical cell (PhyCell), inspired from data assimilation techniques, for performing PDE-constrained prediction in latent space. Extensive experiments conducted on four various datasets show the ability of PhyDNet to outperform state-of-the-art methods. Ablation studies also highlight the important gain brought out by both disentanglement and PDE-constrained prediction. Finally, we show that PhyDNet presents interesting features for dealing with missing data and long-term forecasting.

1. Introduction

Video forecasting consists in predicting the future content of a video conditioned on previous frames. This is of crucial importance in various contexts, such as weather forecasting [73], autonomous driving [29], reinforcement learning [43], robotics [16], or action recognition [33]. In this work, we focus on unsupervised video prediction, where the absence of semantic labels to drive predictions exacerbates the challenges of the task. In this context, a key problem is to design video prediction methods able to represent the complex dynamics underlying raw data.

State-of-the-art methods for training such complex dynamical models currently rely on deep learning, with specific architectural choices based on 2D/3D convolutional [40, 62] or recurrent neural networks [66, 64, 67].

To improve predictions, recent methods use adversarial training [40, 62, 29], stochastic models [7, 41], constraint predictions by using geometric knowledge [16, 24, 75] or by disentangling factors of variation [60, 58, 12, 21].

Another appealing way to model the video dynamics is to exploit prior physical knowledge, e.g. formalized by partial differential equations (PDEs) [11, 55]. Recently, interesting connections between residual networks and PDEs have been drawn [71, 37, 8], enabling to design physically-constrained machine learning frameworks [48, 11, 55, 52]. These approaches are very successful for modeling complex natural phenomena, e.g. climate, when the underlying dynamics is well described by the physical equations in the input space [48, 52, 35]. However, such assumption is rarely fulfilled in the pixel space for predicting generalist videos.

In this work, we introduce PhyDNet, a deep model dedicated to perform accurate future frame predictions from generalist videos. In such a context, physical laws do not apply in the input pixel space; the goal of PhyDNet is to learn a semantic latent space \( \mathcal{H} \) in which they do, and are disentangled from other factors of variation required to per-
form future prediction. Prediction results of PhyDNet when trained on Moving MNIST [56] are shown in Figure 1. The left branch represents the physical dynamics in $H$; when decoded in the image space, we can see that the corresponding features encode approximate segmentation masks predicting digit positions on subsequent frames. On the other hand, the right branch extracts residual information required for prediction, here the precise appearance of the two digits. Combining both representations eventually makes accurate prediction successful.

Our contributions to the unsupervised video prediction problem with PhyDNet can be summarized as follows:

- We introduce a global sequence to sequence two-branch deep model (section 3.1) dedicated to jointly learn the latent space $H$ and to disentangle physical dynamics from residual information, the latter being modeled by a data-driven (ConvLSTM [73]) method.

- Physical dynamics is modeled by a new recurrent physical cell, PhyCell (section 3.2), discretizing a broad class of PDEs in $H$. PhyCell is based on a prediction-correction paradigm inspired from the data assimilation community [1], enabling robust training with missing data and for long-term forecasting.

- Experiments (section 4) reveal that PhyDNet outperforms state-of-the-art methods on four generalist datasets: this is, as far as we know, the first physically-constrained model able to show such capabilities. We highlight the importance of both disentanglement and physical prediction for optimal performances.

2. Related work

We review here related multi-step video prediction approaches dedicated to long-term forecasting. We also focus on unsupervised training, i.e. only using input video data and without manual supervision based on semantic labels.

Deep video prediction Deep neural networks have recently achieved state-of-the-art performances for data-driven video prediction. Seminal works include the application of sequence to sequence LSTM or Convolutional variants [56, 73], adopted in many studies [16, 36, 74]. Further works explore different architectural designs based on Recurrent Neural Networks (RNNs) [66, 64, 44, 67, 65] and 2D/3D ConvNets [40, 62, 50, 6]. Dedicated loss functions [10, 30] and Generative Adversarial Networks (GANs) have been investigated for sharper predictions [40, 62, 29]. However, the problem of conditioning GANs with prior information, such as physical models, remains an open question.

To constrain the challenging generation of high dimensional images, several methods rather predict geometric transformations between frames [16, 24, 75] or use optical flow [46, 38, 33, 32, 31]. This is very effective for short-term prediction, but degrades quickly when the video content evolves, where more complex models and memory about dynamics are required.

A promising line of work consists in disentangling independent factors of variations in order to apply the prediction model on lower-dimensional representations. A few approaches explicitly model interactions between objects inferred from an observed scene [14, 27, 76]. Relational reasoning, often implemented with graphs [2, 26, 53, 45, 59], can account for basic physical laws, e.g. drift, gravity, spring [70, 72, 42]. However, these methods are object-centric, only evaluate on controlled settings and are not suited for general real-world video forecasting. Other disentangling approaches factorize the video into independent components [60, 58, 12, 21, 19]. Several disentanglement criteria are used, such as content/motion [60] or deterministic/stochastic [12]. In specific contexts, the prediction space can be structured using additional information, e.g. with human pose [61, 63] or key points [41], which imposes a severe overhead on the annotation budget.

Physics and PDEs Exploiting prior physical knowledge is another appealing way to improve prediction models. Earlier attempts for data-driven PDE discovery include sparse regression of potential differential terms [5, 52, 54] or neural networks approximating the solution and response function of PDEs [49, 48, 55]. Several approaches are dedicated to a specific PDE, e.g. advection-diffusion in [11]. Based on the connection between numerical schemes for solving PDEs (e.g. Euler, Runge-Kutta) and residual neural networks [71, 37, 8, 78], several specific architectures were designed for predicting and identifying dynamical systems [15, 35, 47]. PDE-Net [35, 34] discretizes a broad class of PDEs by approximating partial derivatives with convolutions. Although these works leverage physical knowledge, they either suppose physics behind data to be explicitly known or are limited to a fully visible state, which is rarely the case for general video forecasting.

Deep Kalman filters To handle unobserved phenomena, state space models, in particular the Kalman filter [25], have been recently integrated with deep learning, by modeling dynamics in learned latent space [28, 69, 20, 17, 3]. The Kalman variational autoencoder [17] separates state estimation in videos from dynamics with a linear gaussian state space model. The Recurrent Kalman Network [3] uses a factorized high dimensional latent space in which the linear Kalman updates are simplified and don’t require computationally-heavy covariance matrix inversions. These methods inspired by the data assimilation community [1, 4] have advantages in missing data or long-term forecasting contexts due to their mechanisms decoupling latent dynamics and input assimilation. However, they assume simple latent dynamics (linear) and don’t include any physical prior.
3. PhyDNet model for video forecasting

We introduce PhyDNet, a model dedicated to video prediction, which leverages physical knowledge on dynamics, and disentangles it from other unknown factors of variations necessary for accurate forecasting. To achieve this goal, we introduce a disentangling architecture (section 3.1), and a new physically-constrained recurrent cell (section 3.2).

**Problem statement** As discussed in introduction, physical laws do not apply at the pixel level for general video prediction tasks. However, we assume that there exists a conceptual latent space \( \mathcal{H} \) in which physical dynamics and residual factors are linearly disentangled. Formally, let us denote as \( u = u(t, x) \) the frame of a video sequence at time \( t \), for spatial coordinates \( x = (x, y) \). \( u(t, x) \in \mathcal{H} \) is the latent representation of the video up to time \( t \), which decomposes as \( h = h^p + h^r \), where \( h^p \) (resp. \( h^r \)) represents the physical (resp. residual) component of the disentanglement. The video evolution in the latent space \( \mathcal{H} \) is thus governed by the following partial differential equation (PDE):

\[
\frac{\partial h(t, x)}{\partial t} = \frac{\partial h^p}{\partial t} + \frac{\partial h^r}{\partial t} := \mathcal{M}_p(h^p, u) + \mathcal{M}_r(h^r, u) \tag{1}
\]

\( \mathcal{M}_p(h^p, u) \) and \( \mathcal{M}_r(h^r, u) \) represent physical and residual dynamics in the latent space \( \mathcal{H} \).

3.1. PhyDNet disentangling architecture

The main goal of PhyDNet is to learn the mapping from input sequences to a latent space which approximates the disentangling properties formalized in Eq (1).

To reach this objective, we introduce a recurrent bloc which is shown in Figure 2(a). A video frame \( u_t \) at time \( t \) is mapped by a deep convolutional encoder \( E \) into a latent space representing the targeted space \( \mathcal{H} \). \( E(u_t) \) is then used as input for two parallel recurrent neural networks, incorporating this spatial representation into a dynamical model.

The left branch in Figure 2(a) models the latent representation \( h^p \) fulfilling the physical part of the PDE in Eq (1), i.e. \( \frac{\partial h^p(t, x)}{\partial t} = \mathcal{M}_p(h^p, u) \). This PDE is modeled by our recurrent physical cell described in section 3.2, PhyCell, which leads to the computation of \( h^p_{t+1} \) from \( E(u_t) \) and \( h^p_t \).

From the machine learning perspective, PhyCell leverages physical constraints to limit the number of model parameters, regularizes training and improves generalization.

The right branch in Figure 2(a) models the latent representation \( h^r \) fulfilling the residual part of the PDE in Eq (1), i.e. \( \frac{\partial h^r(t, x)}{\partial t} = \mathcal{M}_r(h^r, u) \). Inspired by wavelet decomposition [39] and recent semi-supervised works [51], this part of the PDE corresponds to unknown phenomena, which do not correspond to any prior model, and is therefore entirely learned from data. We use a generic recurrent neural network for this task, e.g. ConvLSTM [73] for videos, which computes \( h^r_{t+1} \) from \( E(u_t) \) and \( h^r_t \).

\( h_{t+1} = h^p_{t+1} + h^r_{t+1} \) is the combined representation processed by a deep decoder \( D \) to forecast the image \( \hat{u}_{t+1} \).

Figure 2(b) shows the "unfolded" PhyDNet. An input video \( u_{1:T} = (u_1, ..., u_T) \in \mathbb{R}^{T \times n \times m \times c} \) with spatial size.
n × m and c channels is projected into H by the encoder E and processed by the recurrent block unfolded in time. This forms a Sequence To Sequence architecture [57] suited for multi-step prediction, outputting ∆ future frame predictions ΦT+1:T+Δ. Encoder, decoder and recurrent block parameters are all trained end-to-end, meaning that PhyDNet learns itself without supervision the latent space H in which physics and residual factors are disentangled.

3.2. PhyCell: a deep recurrent physical model

PhyCell is a new physical cell, whose dynamics is governed by the PDE response function \( \mathcal{M}_p(h^p, u) \):

\[
\mathcal{M}_p(h, u) := \Phi(h) + C(h, u) \tag{2}
\]

where \( \Phi(h) \) is a physical predictor modeling only the latent dynamics and \( C(h, u) \) is a correction term modeling the interactions between latent state and input data.

**Physical predictor:** \( \Phi(h) \) in Eq (2) is modeled as follows:

\[
\Phi(h(t, x)) = \sum_{i,j: j+2 \leq q} c_{i,j} \frac{\partial^{i+j}h}{\partial x^i \partial y^j}(t, x) \tag{3}
\]

\( \Phi(h(t, x)) \) in Eq (3) combines the spatial derivatives with coefficients \( c_{i,j} \) up to a certain differential order \( q \). This generic class of linear PDEs subsumes a wide range of classical physical models, e.g. the heat equation, the wave equations, the advection-diffusion equations.

**Correction:** \( C(h, u) \) in Eq (2) takes the following form:

\[
C(h, u) := K(t, x) \odot [E(u(t, x)) - (h(t, x) + \Phi(h(t, x)))] \tag{4}
\]

Eq (4) computes the difference between the latent state after physical motion \( h(t, x) + \Phi(h(t, x)) \) and the embedded new observed input \( E(u(t, x)) \). \( K(t, x) \) is a gating factor, where \( \odot \) is the Hadamard product.

3.2.1 Discrete PhyCell

We discretize the continuous time PDE in Eq (2) with the standard forward Euler numerical scheme [37], leading to the discrete time PhyCell (derivation in supplementary 1.1):

\[
h_{t+1} = (1 - K_t) \odot (h_t + \Phi(h_t)) + K_t \odot E(u_t) \tag{5}
\]

Depicted in Figure 3, PhyCell is an atomic recurrent cell for building physically-constrained RNNs. In our experiments, we use one layer of PhyCell but one can also easily stack several PhyCell layers to build more complex models, as done for stacked RNNs [66, 64, 67]. To gain insight into PhyCell in Eq (5), we write the equivalent two-steps form:

\[
\begin{align*}
\hat{h}_{t+1} &= h_t + \Phi(h_t) & \text{Prediction} \tag{6} \\
h_{t+1} &= \hat{h}_{t+1} + K_t \odot (E(u_t) - \hat{h}_{t+1}) & \text{Correction} \tag{7}
\end{align*}
\]

In the sequel, we drop the index \( p \) in \( h^p \) for the sake of simplicity.

The prediction step in Eq (6) is a physically-constrained motion in the latent space, computing the intermediate representation \( \hat{h}_{t+1} \). Eq (7) is a correction step incorporating input data. This prediction-correction formulation is reminiscent of the way to combine numerical models with observed data in the data assimilation community [1, 4], e.g. with the Kalman filter [25]. We show in section 3.3 that this decoupling between prediction and correction can be leveraged to robustly train our model in long-term forecasting and missing data contexts. \( K_t \) can be interpreted as the Kalman gain controlling the trade-off between both steps.

3.2.2 PhyCell implementation

We now specify how the physical predictor \( \Phi \) in Eq (6) and the correction Kalman gain \( K_t \) in Eq (7) are implemented.

**Physical predictor:** we implement \( \Phi \) using a convolutional neural network (left gray box in Figure 3), based on the connection between convolutions and differentiations [13, 35]. This offers the possibility to learn a class of filters approximating each partial derivative in Eq (3), which are constrained by a kernel moment loss, as detailed in section 3.3. As noted by [35], the flexibility added by this constrained learning strategy gives better results for solving PDEs than handcrafted derivative filters. Finally, we use 1 × 1 convolutions to linearly combine these derivatives with coefficients \( c_{i,j} \) in Eq (3).

**Kalman gain:** We approximate \( K_t \) in Eq (7) by a gate with learned convolution kernels \( W_h, W_u \) and bias \( b_k \):

\[
K_t = \tanh \left( W_h * \hat{h}_{t+1} + W_u * E(u_t) + b_k \right) \tag{8}
\]
Note that if $K_t = 0$, the input is not accounted for and the dynamics follows the physical predictor; if $K_t = 1$, the latent dynamics is resetted and only driven by the input. This is similar to gating mechanisms in LSTMs or GRUs.

**Discussion:** With specific $\Phi$ predictor, $K_t$ gain and encoder $E$, PhyCell recovers recent models from the literature:

<table>
<thead>
<tr>
<th>model</th>
<th>$\Phi$</th>
<th>$K_t$</th>
<th>$E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDE-Net [34]</td>
<td>Eq (6)</td>
<td>0</td>
<td>Id</td>
</tr>
<tr>
<td>RKF [3]</td>
<td>locally linear, no approx. deep encoder</td>
<td>0</td>
<td>Id</td>
</tr>
<tr>
<td>PhyDNet (ours)</td>
<td>Eq (6)</td>
<td>Eq (8)</td>
<td>deep encoder</td>
</tr>
</tbody>
</table>

PDE-Net [35] directly works on raw pixel data (identity encoder $E$) and assumes Markovian dynamics (no correction, $K_t = 0$): the model solves the autonomous PDE $\frac{\partial u}{\partial t} = \Phi(u)$ given in Eq (6) but in pixel space. This prevents from modeling time-varying PDEs such as those tackled in this work, e.g. varying advection terms. The flow model in [11] uses the closed-form solution of the advection-diffusion equation as predictor; it is however limited only to this PDE, whereas PhyDNet models a much broader class of PDEs. The Recurent Kalman Filter (RKF) [3] also proposes a prediction-correction scheme in a deep latent space, but their approach does not include any prior physical information, and the prediction step is locally linear, whereas we use deep models. An approximated form of the covariance matrix is used for estimating $K_t$ in [3], which we find experimentally inferior to our gating mechanism in Eq (8).

### 3.3. Training

Given a training set of $N$ videos $\mathcal{D} = \{u^{(i)}\}_{i=1:N}$ and PhyDNet parameters $w = (w_p, w_r, w_e)$, where $w_p$ (resp. $w_r$) are parameters of the PhyCell (resp. residual) branch, and $w_e$ are encoder and decoder shared parameters, we minimize the following objective function:

$$\mathcal{L}(\mathcal{D}, w) = \mathcal{L}_{\text{image}}(\mathcal{D}, w) + \lambda \mathcal{L}_{\text{moment}}(w_p)$$

We use the $L^2$ loss for the image reconstruction loss $\mathcal{L}_{\text{image}}$, as commonly done in the literature [66, 64, 44, 65, 67].

Let $\mathcal{L}_{\text{moment}}(w_p)$ imposes physical constraints on the $k^2$ learned filters $\{w^k_{p,i,j}\}_{i,j,k}$ such that each $w^k_{p,i,j}$ of size $k \times k$ approximates $\frac{\partial^k}{\partial x^k}$ (9). This is achieved by using a loss based on the moment matrix $M(w^k_{p,i,j})$ [34], representing the order of the filter differentiation [13]. $M(w^k_{p,i,j})$ is compared to a target moment matrix $\Delta^k_{i,j}$ (see $M$ and $\Delta$ computations in supplementary 1.2), leading to:

$$\mathcal{L}_{\text{moment}} = \sum_{i,k} \sum_{j,k \leq k} ||M(w^k_{p,i,j}) - \Delta^k_{i,j}||_F$$

**Prediction mode** An appealing feature of PhyCell is that we can use and train the model in a "prediction-only" mode by setting $K_t = 0$ in Eq (7), i.e. by only relying on the physical predictor $\Phi$ in Eq (6). It is worth mentioning that the "prediction-only" mode is not applicable to standard Seq2Seq RNNs: although the decomposition in Eq (2) still holds, i.e. $\mathcal{M}(\hat{h}, u) = \Phi(h) + \mathcal{C}(\hat{h}, u)$, the resulting predictor is naive and useless for multi-step prediction $\hat{h}_{t+1} = 0$, see supplementary 1.3).

Therefore, standard RNNs are not equipped to deal with unreliable input data $u_t$. We show in section 4.4 that the gain of PhyDNet over those models increases in two important contexts with unreliable inputs: multi-step prediction and dealing with missing data.

### 4. Experiments

#### 4.1. Experimental setup

**Datasets** We evaluate PhyDNet on four datasets from various origins. **Moving MNIST** [56] is a standard synthetic benchmark in video prediction with two random digits bouncing on the walls. **Traffic BJ** [77] represents complex real-world traffic flows, which requires modeling transport phenomena and traffic diffusion for prediction. **SST** (Sea Surface Temperature) [11] consists in meteorological data, whose evolution is governed by the physical laws of fluid dynamics. Finally, **Human 3.6** [22] represents general human actions with complex 3D articulated motions. We give details about all datasets in supplementary 2.1.

<table>
<thead>
<tr>
<th>Method</th>
<th>Moving MNIST</th>
<th>Traffic BJ</th>
<th>Sea Surface Temperature</th>
<th>Human 3.6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>MAE</td>
<td>SSIM</td>
<td>MSE</td>
</tr>
<tr>
<td>ConvLSTM [73]</td>
<td>103.3</td>
<td>182.9</td>
<td>0.707</td>
<td>48.5</td>
</tr>
<tr>
<td>PredRNN [66]</td>
<td>56.8</td>
<td>126.1</td>
<td>0.867</td>
<td>46.4</td>
</tr>
<tr>
<td>Causal LSTM [64]</td>
<td>46.5</td>
<td>106.8</td>
<td>0.898</td>
<td>44.8</td>
</tr>
<tr>
<td>MIM [67]</td>
<td>44.2</td>
<td>101.1</td>
<td>0.910</td>
<td>42.9</td>
</tr>
<tr>
<td>EID-LSTM [65]</td>
<td>41.3</td>
<td>86.4</td>
<td>0.920</td>
<td>43.2</td>
</tr>
<tr>
<td>Advection-diffusion flow [11]</td>
<td>38.9</td>
<td>90.7</td>
<td>0.922*</td>
<td>-</td>
</tr>
<tr>
<td>DDPAE [21]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Quantitative forecasting results of PhyDNet compared to baselines using various datasets. Numbers are copied from original or citing papers. * corresponds to results obtained by running online code from the authors. The first five baseline are general deep models applicable to all datasets, whereas DDPAE [21] (resp. advection-diffusion flow [11]) are specific state-of-the-art models for Moving MNIST (resp. SST). Metrics are scaled to be in a similar range across datasets to ease comparison.
Network architectures and training PhyDNet shares a common backbone architecture for all datasets where the physical branch contains 49 PhyCells ($7 \times 7$ kernel filters) and the residual branch is composed of a 3-layers ConvLSTM with 128 filters in each layer. We set up the trade-off parameter between $L_{\text{image}}$ and $L_{\text{moment}}$ to $\lambda = 1$. Detailed architectures and $\lambda$ impact are given in supplementary 2.2. Our code is available at https://github.com/vincent-leguen/PhyDNet.

Evaluation metrics We follow evaluation metrics commonly used in state-of-the-art video prediction methods: the Mean Squared Error (MSE), Mean Absolute Error (MAE) and the Structural Similarity (SSIM) [68] that computes the perceived image quality with respect to a reference. Metrics are averaged for each frame of the output sequence. Lower MSE, MAE and higher SSIM indicate better performances.

4.2. State of the art comparison

We evaluate PhyDNet against strong recent baselines, including very competitive data-driven RNN architectures: ConvLSTM [73], PredRNN [66], Causal LSTM [64], Memory in Memory (MIM) [67]. We also compare to methods dedicated to specific datasets: DDPAE [21], a disentangling method specialized and state-of-the-art on Moving MNIST; and the physically-constrained advection-diffusion flow model [11] that is state-of-the-art for the SST dataset.

Overall results presented in Table 1 reveal that PhyDNet outperforms significantly all baselines on all four datasets. The performance gain is large with respect to state-of-the-art general RNN models, with a gain of 17 MSE points for Moving MNIST, 6 MSE points for Human 3.6, 3 MSE points for SST and 1 MSE point for Traffic BJ. In addition, PhyDNet also outperforms specialized models: it gains 14 MSE points compared to the disentangling DDPAE model [21] specialized for Moving MNIST, and 2 MSE points compared to the advection-diffusion model [11] dedicated to SST data. PhyDNet also presents large and consistent gains in SSIM, indicating that image quality is greatly improved by the physical regularization. Note that for Human 3.6, a few approaches use specific strategies dedicated to human motion with additional supervision, e.g. hu-
man pose in [61]. We perform similarly to [61] using only unsupervised training, as shown in supplementary 2.3. This is, to the best of our knowledge, the first time that physically-constrained deep models reach state-of-the-art performances on generalist video prediction datasets.

In Figure 4, we provide qualitative prediction results for all datasets, showing that PhyDNet properly forecasts future images for the considered horizons: digits are sharply and accurately predicted for Moving MNIST in (a), the absolute traffic flow error is low and approximately spatially independent in (b), the evolving physical SST phenomena are well anticipated in (c) and the future positions of the person is accurately predicted in (d). We add in Figure 4(a) a qualitative comparison to DDPAE [21], which fails to predict the future frames properly. Since the two digits overlap in the input sequence, DDPAE is unable to disentangle them. In contrast, PhyDNet successfully learns the physical dynamics of the two digits in a disentangled latent space, leading a correct prediction. In supplementary 2.4, we detail this comparison to DDPAE, and provide additional visualizations for all datasets.

4.3. Ablation Study

We perform here an ablation study to analyse the respective contributions of physical modeling and disentanglement. Results are presented in Table 2 for all datasets. We see that a 1-layer PhyCell model (only the left branch of PhyDNet in Figure 2(b)) outperforms a 3-layers ConvLSTM architecture (PhyDNet) by 4 points (270,000 vs 3 million parameters). This confirms that PhyCell is a very effective recurrent cell that successfully incorporates physical prior in deep models. When we further add our disentangling strategy with the two-branch architecture (PhyDNet), we have another performance gap on all datasets (25 MSE points for Moving MNIST, 7 points for Traffic and SST, and 5 points for Human 3.6), which proves that physical modeling is not sufficient by itself to perform general video prediction and that learning unknown factors is necessary.

We qualitatively analyze in Figure 5 partial predictions of PhyDNet for the physical branch \( \hat{u}^p_{t+1} = D(h^p_{t+1}) \) and residual branch \( \hat{u}^r_{t+1} = D(h^r_{t+1}) \). As noted in Figure 1 for Moving MNIST, \( h^p \) captures coarse localisations of objects, while \( h^r \) captures fine-grained details that are not useful for the physical model. Additional visualizations for the other datasets and a discussion on the number of parameters are provided in supplementary 2.5.

**Table 2.** An ablation study shows the consistent performance gain on all datasets of our physically-constrained PhyCell vs the general purpose ConvLSTM, and the additional gain brought up by the disentangling architecture PhyDNet. * corresponds to results obtained by running online code from the authors.

<table>
<thead>
<tr>
<th>Method</th>
<th>Moving MNIST</th>
<th>Traffic BJ</th>
<th>Sea Surface Temperature</th>
<th>Human 3.6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>MAE</td>
<td>SSIM</td>
<td>MSE \times 10</td>
</tr>
<tr>
<td>ConvLSTM</td>
<td>103.3</td>
<td>182.9</td>
<td>0.707</td>
<td>48.5*</td>
</tr>
<tr>
<td>PhyCell</td>
<td>50.8</td>
<td>129.3</td>
<td>0.870</td>
<td>48.9</td>
</tr>
<tr>
<td>PhyDNet</td>
<td>24.4</td>
<td>70.3</td>
<td>0.947</td>
<td>41.9</td>
</tr>
</tbody>
</table>

**Figure 5.** Qualitative ablation results on Moving MNIST: partial predictions show that PhyCell captures coarse localisation of digits, whereas the ConvLSTM branch models the fine shape details of digits. Every two frames are displayed.

**Influence of physical regularization** We conduct in Table 3 a finer ablation on Moving MNIST to study the impact of the physical regularization \( \mathcal{L}_{\text{moment}} \) on the performance of PhyCell and PhyDNet. When we disable \( \mathcal{L}_{\text{moment}} \) for training PhyCell, performances improve by 7 points in MSE. This underlines that physical laws alone are too restrictive for learning dynamics in a general context, and that complementary factors should be accounted for. On the other side, when we disable \( \mathcal{L}_{\text{moment}} \) for training our disentangled architecture PhyDNet, performances decrease by 5 MSE points (29 vs 24.4) compared to the physically-constrained version. This proves that physical constraints are relevant, but should be incorporated carefully in order to make both branches cooperate. This enables to leverage physical prior, while keeping remaining information necessary for pixel-level prediction. Same conclusions can be drawn for the other datasets, see supplementary 2.6.
Table 3. Influence of physical regularization for Moving MNIST.

<table>
<thead>
<tr>
<th>Method</th>
<th>MSE</th>
<th>MAE</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhyCell</td>
<td>50.8</td>
<td>129.3</td>
<td>0.870</td>
</tr>
<tr>
<td>PhyCell without $L_{\text{moment}}$</td>
<td>43.4</td>
<td>112.8</td>
<td>0.895</td>
</tr>
<tr>
<td>PhyDNet</td>
<td>24.4</td>
<td>70.3</td>
<td><strong>0.947</strong></td>
</tr>
<tr>
<td>PhyDNet without $L_{\text{moment}}$</td>
<td>29.0</td>
<td>81.2</td>
<td>0.934</td>
</tr>
</tbody>
</table>

4.4. PhyCell analysis

4.4.1 Physical filter analysis

With the same general backbone architecture, PhyDNet can express different PDE dynamics associated to the underlying phenomena by learning specific $c_{i,j}$ coefficients combining the partial derivatives in Eq (3). In Figure 6, we display the mean amplitude of the learned coefficients $c_{i,j}$ with respect to the order of differentiation. For Moving MNIST, the 0th and 1st orders are largely dominant, meaning a purely advective behaviour coherent with the piecewise-constant translation dynamics of the dataset. For Traffic BJ and SST, there is also a global decrease in amplitude with respect to order, we nonetheless notice a few higher order terms appearing to be useful for prediction. For Human 3.6, where the nature of the prior motion is less obvious, these coefficients are more spread across order derivatives.

4.4.2 Dealing with unreliable inputs

We explore here the robustness of PhyDNet when dealing with unreliable inputs, that can arise in two contexts: long-term forecasting and missing data. As explained in section 3.3, PhyDNet can be used in a prediction mode in this context, limiting the use of unreliable inputs, whereas general RNNs cannot. To validate the relevance of the prediction mode, we compare PhyDNet to DDPAE [21], based on a standard RNN (LSTM) as predictor module. Figure 7 presents the results in MSE obtained by PhyDNet and DDPAE on Moving MNIST (see supplementary 2.7 for similar results in SSIM).

For long-term forecasting, we evaluate the performances of both methods far beyond the prediction range seen during training (up to 80 frames), as shown in Figure 7(a). We can see that the performance drop (MSE increase rate) is approximately linear for PhyNet, whereas it is much more pronounced for DDPAE. For example, PhyDNet for 80-steps prediction reaches similar performances in MSE than DDPAE for 20-steps prediction. This confirms that PhyDNet can limit error accumulation during forecasting by using a powerful dynamical model.

Finally, we evaluate the robustness of PhyDNet on DDPAE on missing data, by varying the ratio of missing data (from 10 to 50%) in input sequences during training and testing. A missing input image is replaced with a default value (0) image. In this case, PhyCell relies only on its latent dynamics by setting $K_t = 0$, whereas DDPAE takes the null image as input. Figure 7(b) shows that the performance gap between PhyDNet and DDPAE increases with the percentage of missing data.

5. Conclusion

We propose PhyDNet, a new model for disentangling prior dynamical knowledge from other factors of variation required for video prediction. PhyDNet enables to apply PDE-constrained prediction beyond fully observed physical phenomena in pixel space, and to outperform state-of-the-art performances on four generalist datasets. Our introduced recurrent physical cell for modeling PDE dynamics generalizes recent models and offers the appealing property to decouple prediction from correction. Future work include using more complex numerical schemes, e.g. Runge-Kutta [15], and extension to probabilistic forecasts with uncertainty estimation [18, 9], e.g. with stochastic differential equations [23].
References


