Advection Augmented Convolutional Neural Networks

Niloufar Zakariaei *
University of British Columbia
Vancouver, Canada
nilouzk@student.ubc.ca

Siddharth Rout *
University of British Columbia
Vancouver, Canada
siddharth.rout@ubc.ca

Eldad Haber
University of British Columbia
Vancouver, Canada
ehaber@eoas.ubc.ca

Moshe Eliasof
University of Cambridge
Cambridge, United Kingdom
me532@cam.ac.uk

Abstract

Many problems in physical sciences are characterized by the prediction of space-time sequences. Such problems range from weather prediction to the analysis of disease propagation and video prediction. Modern techniques for the solution of these problems typically combine Convolution Neural Networks (CNN) architecture with a time prediction mechanism. However, oftentimes, such approaches underperform in the long-range propagation of information and lack explainability. In this work, we introduce a physically inspired architecture for the solution of such problems. Namely, we propose to augment CNNs with advection by designing a novel semi-Lagrangian push operator. We show that the proposed operator allows for the non-local transformation of information compared with standard convolutional kernels. We then complement it with Reaction and Diffusion neural components to form a network that mimics the Reaction-Advection-Diffusion equation, in high dimensions. We demonstrate the effectiveness of our network on a number of spatio-temporal datasets that show their merit. Our code is available at https://github.com/Siddharth-Rout/deepADRnet.

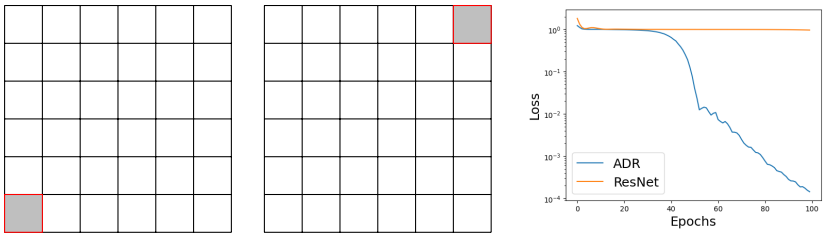
1 Introduction and Motivation

Convolution Neural Networks (CNNs) have long been established as one of the most fundamental and powerful family of algorithms for image and video processing tasks, in applications that range from image classification [27, 21], denoising [5] and reconstruction [26], to generative models [17]. More examples of the impact of CNNs on various fields and applications can be found in [40, 18, 32] and references within.

At the core of CNNs, stands the convolution operation – a simple linear operation that is local and spatially rotation and translation equivariant. The locality of the convolution, coupled with nonlinear activation functions and deep architectures have been the force driving CNN architectures to the forefront of machine learning and artificial intelligence research [50, 21]. One way to understand the success of CNNs and attempt to generate an explainable framework for them is to view CNNs from a Partial Differential Equation (PDE) point of view [43, 7]. In this framework, the convolution is viewed as a mix of discretized differential operators of varying order. The layers of the network are then associated with time. Hence, the deep network can be thought of as a discretization of a nonlinear time-dependent PDE. Such observations have motivated parabolic network design that

*Equal contribution.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



(A) Source image (B) Target image (C) Convergence

Figure 1: A simple task of moving information from one side of the image to the other. The source image in A is moved to the target image in B. The convergence of a simple ResNet and an ADRnet proposed in this work is in (C).

smooth and denoise images [44] as well as to networks that are based on hyperbolic equation [28] and semi-implicit architectures [20].

However, it is known from the literature [34], and is also demonstrated in our experiments, that CNN architectures tend to under-perform in tasks that require rapid transportation (also known as *advection*) of information from one side of an image to the other. In particular, in this paper, we focus on the prediction of the spatio-temporal behavior of image features, where significant transportation is present in the data. Examples of such data include the prediction of weather, traffic flow, and crowd movement.

Related work: In recent years, significant research was devoted to addressing spatio-temporal problems. Most of the works known to us are built on a combination of CNN to capture spatial dependencies and Recurrent Neural Networks (RNN) to capture temporal dependencies. A sample of papers that address this problem and the related problem of video prediction can be found in [4, 53, 46, 31, 22, 35, 13] and reference within. See also [65] and [39] for a recent comparison between different methods. Such methods typically behave as black boxes, in the sense that while they offer strong downstream performance, they often times lack a profound understanding of the learned underlying dynamics from the data. In addition, networks that try to predict the optical flow in videos [9, 64, 48] were proposed to estimate the flow in the original image domain. However, predictions on the image domain may be limited and not capture hidden dynamics. Another type of work that is designed for scientific datasets is [51], which uses Fourier-based methods to build the operators. See also [3] for a review on the topic.

Motivation: Notably, while a CNN is a versatile tool that allows learning spatial dependencies, it can have significant challenges in learning simple operations that require transportation. As an example, let us consider the problem of predicting the motion in the simple case that the input data is an image, where all pixels take the value of 0 except for a pixel on the bottom left (marked in gray), and the output is an image where the value is transported to a pixel on the top right. This example is illustrated in Figure 1. Clearly, no local operation, for example, a convolution of say, 3×3 or even 7×7 can be used to move the information from the bottom left of the image to the top right. Therefore, the architecture to achieve this task requires either many convolutions layers, or, downsampling the image via pooling, where the operations are local, performing convolutions on the downsampled image, and then upsampling the image via unpooling, followed by additional convolutions to "clean" coarsening and interpolation artifacts, as is typical in UNets [42, 8]. To demonstrate, we attempt to fit the data with a simple convolution residual network and with a residual network that has an advection block, as discussed in this paper. The convergence history for the two methods is plotted in Figure 1. We see that while a residual network is incapable of fitting the data, adding an advection block allows it to fit the data to machine precision.

This set of problems, as well as the relatively poor performance it offers on data that contains advection as in simple task in Figure 1 sets the motivation for our work. Our aim is to extend the set of tools that is available in CNNs beyond simple and local convolutions. For time-dependent PDEs, it is well known that it is possible to model most phenomena by a set of advection-diffusion-reaction equations (see, e.g., [12, 11] and references within). Motivated by the connection between the discretization of PDEs and deep network [43, 7], and our observations on the shortcomings of existing operations in CNNs, we propose reformulating CNNs into three different components.

Namely, (i) a pointwise term, also known as a *reaction* term, where channels interact locally. (ii) A *diffusion* term, where features are exchanged between neighboring pixels in a smooth manner. And, (iii) an *advection* term, where features are passed from pixels to other pixels, potentially not only among neighboring pixels, while preserving *feature mass or color loss*[2]. As we discuss in Section 3, the combination of diffusion and reaction is equivalent to a standard CNN. However, there is no CNN mechanism that is equivalent to the advection term. Introducing this new term equips the network with flexibility in cases where information is carried directly.

Contributions: The contributions of this paper are three-fold. First, we form the spatio-temporal dynamics in high dimensions as an advection-diffusion-reaction process, which is novel and has not been studied in CNNs prior to our work. Second, we propose the use of the semi-Lagrangian approach for its solution, introducing a new type of a learnable linear layer, that is sparse yet non-local. This is in contrast to standard convolutional layers, which act locally. In contrast to advection, other mechanisms for non-local interactions, require dense interactions, which are computationally expensive [57]. Specifically, our use of semi-Lagrangian methods offers a bridge between particle-based methods and convolutions [29]. Thus, we present a new operation in the context of CNNs, that we call the *push operator* to implement the advection term. This operator allows us to transport features anywhere on the image in a single step – an operation that cannot be modeled with small local convolution kernels. It is thus a simple yet efficient replacement to the standard techniques that are used to move information on an image. Third, we propose a methodology to learn these layers based on the splitting operator approach, and show that they can successfully model advective processes that appear in different datasets.

Limitations: The advection diffusion reaction model is optimal when applied to the prediction of images where the information for the prediction is somehow present in the given images. Such scenarios are often present in scientific applications. For example, for the prediction of the propagation of fluids or gasses, all we need to know is the state of the fluid now (and in some cases, in a few earlier time frames). A more complex scenario is the prediction of video. In this case, the next frame may have new features that were not present in previous frames. To this end, the prediction of video requires some generative power. While we show that our network can be used for video prediction and even obtain close to the state-of-the-art results, we observe that it performs best for scientific datasets.

2 Model Formulation

Notations and assumptions. We consider a spatio-temporal vector function of the form $\mathbf{q}(t, \mathbf{x}) = [\mathbf{q}_1(t, \mathbf{x}), \ldots, \mathbf{q}_m(t, \mathbf{x})] \in \mathcal{Q}$, where $\mathcal{Q}$ is the space vector function with m channels. The function $\mathbf{q}$ is defined over the domain $\mathbf{x} \in \Omega \subseteq \mathcal{R}^d$, and time interval $[0, t_j]$. Our goal is to predict the function at time t_k for some $t_k > t_j$, given the inputs up to time j. For the problem we consider here, the time is sampled on a uniform grid with equal spacing. Below, we define the advection-diffusion-reaction system that renders the blueprint of the method proposed in this paper to achieve our goal.

Reaction-Advection-Diffusion System. Given the input function $\mathbf{q}$, we first embed it in a higher dimensional space. We denote the embedding function by $\mathbf{I} :\in \mathcal{I}$, defined as

$$\mathbf{I}(t, \mathbf{x}) = M_{\text{In}}(\mathbf{q}(t, \mathbf{x}), \boldsymbol{\theta}_{\text{In}}) \tag{1}$$

where $M_{\text{In}} : \mathbb{R}^m \to \mathbb{R}^c$ is a multi-layer preceptron (MLP) that embeds the function $\mathbf{q}$ from m to $c > m$ channels with trainable parameters $\boldsymbol{\theta}_{\text{In}}$.

To represent the evolution of $\mathbf{q}$ we evolve $\mathbf{I}$ in the hidden dimension, c, and then project it back into the space $\mathcal{Q}$. One useful way to represent the evolution of a spatio-temporal process is by combining three different processes, as follows:

- *Reaction:* A pointwise process where channels interact pointwise (sometimes referred to as 1×1 convolutions)
- *Diffusion:* A process where features are being communicated and diffused locally.

[2]That is the sum of the features is constant.

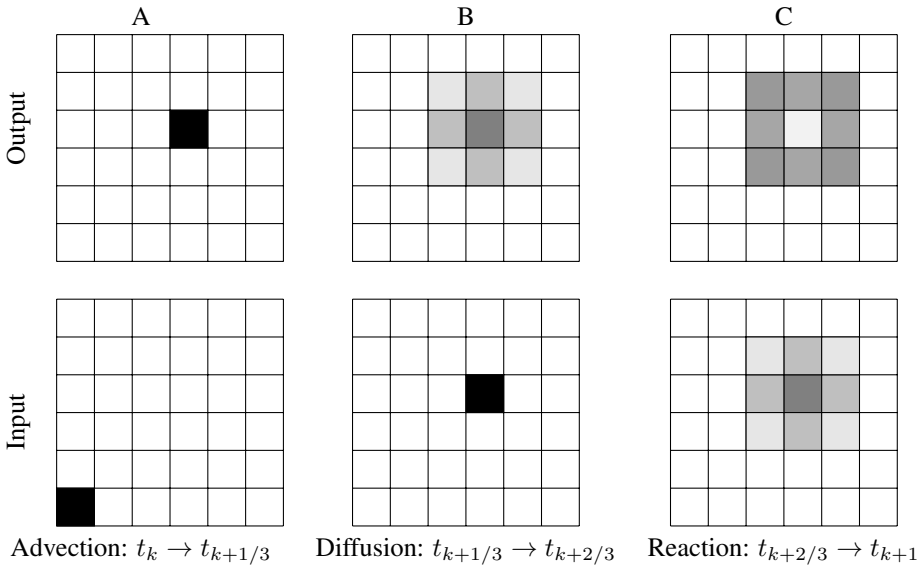


Figure 2: An illustration of the advection-diffusion reaction process. In the first step, Column A (advection), a pixel on the lower left of the image is transported into the middle of the mesh. In the second step, Column B (diffusion), the information is diffused to its neighbors, and finally, in the last step, Column C (reaction), each pixel interacts locally to change its value.

- *Advection:* A process where information transports along mediums.

These three processes are also illustrated in Figure 2, and their composition defines the advection-diffusion-reaction differential equation on the embedded vector $\mathbf{I}$.

The equation can be written as

$$\frac{\partial \mathbf{I}(t,\mathbf{x})}{\partial t} = \kappa \Delta \mathbf{I}(t,\mathbf{x}) + \nabla \cdot (\mathbf{U}\mathbf{I}(t,\mathbf{x})) + R(\mathbf{I}(t,\mathbf{x}), \boldsymbol{\theta}), \tag{2}$$

$$\mathbf{I}(t=0,\mathbf{x}) = M(\mathbf{q}(t=0,\mathbf{x})). \tag{3}$$

Here Δ is the Laplacian, and ∇ is the divergence operator, as classically defined in PDEs [12]. The equation is equipped with an initial condition and some boundary conditions. Here, for simplicity of implementation, we choose the Neumann boundary conditions, but other boundary conditions can also be chosen. The diffusivity coefficient κ, velocity field $\mathbf{U}$, and the parameters that control the reaction term R are trainable and are discussed in Section 3.

The equation is integrated on some interval $[0, T]$ and finally one obtains $\mathbf{q}(T, \mathbf{x})$ by applying a second MLP, that projects the hidden features in $\mathbf{I}(t = T, \mathbf{x})$ to the desired output dimension, which in our case is the same as the input dimension, i.e., m:

$$\mathbf{q}(T,\mathbf{x}) = M_{\text{Out}}(\mathbf{I}(T,\mathbf{x}), \boldsymbol{\theta}_{\text{Out}}), \tag{4}$$

where $\boldsymbol{\theta}_{\text{out}}$ are trainable parameters for the projection MLP.

Remark (Equation 2 Reformulation). The discretization of Equation 2 can be challenging due to conservation properties of the term $\nabla \cdot (\mathbf{U}\mathbf{I}(t,\mathbf{x}))$. An alternative equation, which may be easier to discretize in our context, can be obtained by noting that

$$\frac{\partial \mathbf{I}(t,\mathbf{x})}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{I}) = \frac{\partial \mathbf{I}(t,\mathbf{x})}{\partial t} + \mathbf{U} \cdot \nabla \mathbf{I} + \mathbf{I} \nabla \cdot \mathbf{U}. \tag{5}$$

The operator on the left-hand side in Equation 5 is the continuity equation [12], where the *mass* of $\mathbf{I}$ is conserved. The first two terms on the right hand side, namely, $\mathbf{I}_t + \mathbf{U} \cdot \nabla \mathbf{I}$ are sometimes refer to as the color equation [12] as they conserve the *intensity* of $\mathbf{I}$. For divergent free velocity fields, that is, when $\boldsymbol{\nabla} \cdot \mathbf{U} = 0$, these are equivalent, however, for non-divergent fields, the term $\mathbf{I}\nabla \cdot \mathbf{U}$ is a pointwise operator on $\mathbf{I}$, that is, it is a reaction term. When training a model, one can use either Equation 2 in its continuity form or replace the term with Equation 5 and learn the term $\mathbf{I}\nabla \cdot \mathbf{U}$ as a part of the reaction term, R. We discuss this in discretization of our model in Section 3.3.

3 From a Partial Differential Equation to a Neural Network

To formulate a neural network from the differential equation in Equation 2 needs to be discretized in time and space. In this work, we assume data that resides on a regular, structured mesh grid, such as 2D images, and the spatial operators to discretize Equation 2 are described below. To discretize Equation 2 in *time*, we turn to Operator Splitting methods [1] that are common for the discretization of equations with similar structures, and were shown to be effective in deep learning frameworks [11]. As we see next, such discretization leads to a neural a network that has three types of layers that are composed of each other, resulting in an effectively deeper neural network.

3.1 Operator Splitting

The idea behind operator splitting is to split the integration of the ODE into parts [30]. Specifically, consider a linear differential equation of the form

$$\frac{\partial \mathbf{I}(t,\mathbf{x})}{\partial t} = \mathbf{A}\mathbf{I}(t,\mathbf{x}) + \mathbf{D}\mathbf{I}(t,\mathbf{x}) + \mathbf{R}\mathbf{I}(t,\mathbf{x}), \tag{6}$$

where $\mathbf{A}, \mathbf{D}$ and $\mathbf{R}$ are matrices. The solution to this system at time t is well known [12] and reads

$$\mathbf{I}(t,\mathbf{x}) = \exp\left(t\mathbf{A} + t\mathbf{D} + t\mathbf{R})\right)\mathbf{I}(0,\mathbf{x}), \tag{7}$$

where exp denotes the matrix exponentiation operation. It is also possible to approximate the exact solution presented in Equation 7 as follows

$$\exp\left(t\mathbf{A} + t\mathbf{D} + t\mathbf{R})\right)\mathbf{I}(0,\mathbf{x}) \approx \exp(t\mathbf{A})\left((\exp(t\mathbf{D})(\exp(t\mathbf{R})\mathbf{I}(0,\mathbf{x}))\right) \tag{8}$$

The approximation is of order t, and it stems from the fact that the eigenvalues of the matrices $\mathbf{A}, \mathbf{D}$ and $\mathbf{R}$ do not commute (see [1] for a thorough discussion). Equation 8 can also be interpreted in the following way. The solution, for a short time integration time t, can be approximated by first solving the system $\frac{\partial \mathbf{I}(t,\mathbf{x})}{\partial t} = \mathbf{R}\mathbf{I}(t,\mathbf{x}), \mathbf{I}_0 = \mathbf{I}(0,\mathbf{x})$ obtaining a solution $\mathbf{I}_R(t,\mathbf{x})$, followed by the solution of the system $\frac{\partial \mathbf{I}(t,\mathbf{x})}{\partial t} = \mathbf{D}\mathbf{I}_R(t,\mathbf{x}), \mathbf{I}_0 = \mathbf{I}_R$ obtaining the solution $\mathbf{I}_{RD}(t,\mathbf{x})$ and finally solving the system $\frac{\partial \mathbf{I}(t,\mathbf{x})}{\partial t} = \mathbf{A}\mathbf{I}_{RD}(t,\mathbf{x}), \mathbf{I}_0 = \mathbf{I}_{RD}$. The advantage of this approach is that it allows the use of different techniques for the solution of different problems. The derivation of the approach employed in this work is presented in Appendix A.4, which also provides a detailed explanation of the invariance to the order of splitting.

Let $\mathcal{R}$ be the solution operator that advances $\mathbf{I}(t_j,\mathbf{x})$ to $\mathbf{I}_R(t_{j+1},\mathbf{x})$. Similarly, let $\mathcal{D}$ be the solution operator that advances $\mathbf{I}_R(t_{j+1},\mathbf{x})$ to $\mathbf{I}_{RD}(t_{j+1},\mathbf{x})$ and lastly, let $\mathcal{A}$ be the solution of the advection problem that advances $\mathbf{I}_{RD}(t_{j+1},\mathbf{x})$ to $\mathbf{I}(t_{j+1},\mathbf{x})$. Then, a layer in the system can be written as the composite of three-layer

$$\mathcal{L}\,\mathbf{I}(t_j,\mathbf{x}) = \mathcal{A} \circ \mathcal{D} \circ \mathcal{R}\,\mathbf{I}(t_j,\mathbf{x}). \tag{9}$$

That is, the resulting discretization in time yields a neural network architecture of a layer that is composed of three distinct parts. We now discuss each part separately.

3.2 Advection

The innovative part of our network is advection. The advection approximately solves the equation

$$\frac{\partial \mathbf{I}}{\partial t} = \nabla \cdot \left(\mathbf{U}(\mathbf{I},\mathbf{x},t)\mathbf{I}\right), \tag{10}$$

for a general velocity field $\mathbf{U}$. For the solution of this equation, we now introduce a linear operation that we use to enhance the performance of our network. Our goal is to allow for information to pass over large distances. To this end, consider a displacement field $\mathbf{U} = (\mathbf{U}_1, \mathbf{U}_2)$ and consider the push operation, $\mathbf{A}(\mathbf{U})\mathbf{I}$ as the operation that takes every pixel in $\mathbf{I}$ and displaces it from point $\mathbf{x}$ to $\mathbf{x}_u = \mathbf{x} + \mathbf{U}$. Since the point $\mathbf{x}_u$ does not necessarily reside on a grid point, the information from $\mathbf{x}_u$ is spread over four grid points neighbors, in weights that are proportional to the distance from these points. A sketch of this process is plotted in Figure 3 (a). The operator discussed above conserves that *mass* of the features. A different implementation, as discussed in Remark 1, is to discretize the color equation. This is done by looking backward and using the interpolated value as shown in

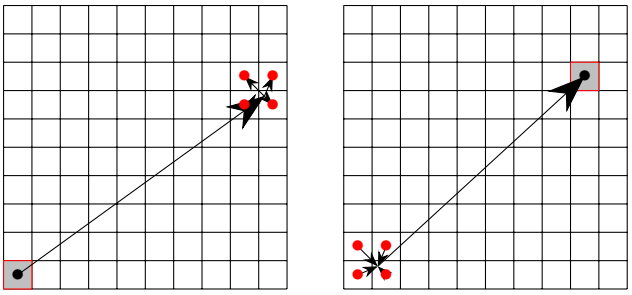


Figure 3: Discretization of the push operator. (a) Left: Semi-Lagrangian mass preserving transport, discretizing the continuity. (b) Right: Semi-Lagrangian color-preserving transport.

Figure 3(b). It is possible to show [14] that these linear operators are transposed of each other. Here, for each implementation, we chose to use the color equation. We show in ablation studies that the results when using either formulation are equivalent.

The process allows for a different displacement vector $\mathbf{u}$ for every grid point. The displacement field $\mathbf{U}$ in has $2c$ channels and can vary in space and time. To model the displacement field, we propose to use the data at times,

$$\mathbf{Q}_k = [\mathbf{q}(t_{k-j}, \mathbf{x}), \mathbf{q}(t_{k-j+1}, \mathbf{x}), \ldots, \mathbf{q}(t_k, \mathbf{x})], \tag{11}$$

where j is the length of history used to learn the displacements.

Using $\mathbf{Q}_k$, the displacement field is computed by a simple residual convolution network, which we formally write as

$$\mathbf{U}_k = RN(\mathbf{Q}_k, \boldsymbol{\eta}), \tag{12}$$

where RN is the residual network parameterized by $\boldsymbol{\eta}$.

3.3 Reaction

The reaction term is a nonlinear 1×1 convolution. This yields a residual network of the form

$$\mathbf{I}_{j+1} = \mathbf{I}_j + hM(\mathbf{I}_j, \boldsymbol{\theta}_j) = \mathcal{R}_j(\boldsymbol{\theta}_j)\,\mathbf{I}_j, \tag{13}$$

where M is a standard, double-layer MLP with parameters $\boldsymbol{\theta}_j$ and h is a step size that is a hyper-parameter. We may choose to have more than a single reaction step per iteration.

3.4 Diffusion

For the diffusion step, we need to discretize the Laplacian on the image. We use the standard 5-point Laplacian [16] that can also be expressed as 2D group convolution [37]. Let Δ_h be the discrete Laplacian. The diffusion equation reads

$$\mathbf{I}_{j+1} - \mathbf{I}_j = h\kappa\Delta_h\mathbf{I}_k.$$

If we choose $k = j$ we obtain an explicit scheme

$$\mathbf{I}_{j+1} = \mathbf{I}_j + h\kappa\Delta_h\mathbf{I}_j. \tag{14}$$

Note that the diffusion layer can be thought of as a group convolution where each channel is convolved with the same convolution and then scaled with a different κ. The forward Euler method for the diffusion requires $h\kappa$ to be small if we want to retain stability. By choosing $k = j + 1$ we obtain the backward Euler method, which is unconditionally stable

$$\mathbf{I}_{j+1} = (\mathbf{I} - h\kappa\Delta_h)^{-1}\mathbf{I}_j = \mathcal{D}(\kappa)\mathbf{I}_j. \tag{15}$$

To invert the matrix we use the cosine transform [25] which yields an $n \log n$ complexity for this step.

Table 1: Datasets statistics. Training and testing splits, image sequences, and resolutions

Dataset	N_{train}	N_{test}	(C, H, W)	History	Prediction
PDEBench-SWE	900	100	(1, 128, 128)	10	1
CloudCast	5241	1741	(1, 128, 128)	4	4, 8, 12, 16
Moving MNIST	10000	10000	(1, 64, 64)	10	10
KITTI	2042	1983	(3, 154, 512)	2	1, 3

Combining Diffusion and Reaction to a Single Layer. In the above network the diffusion is handled by an implicit method (that is a matrix inversion) and the reaction is handled by an explicit method. For datasets where the diffusion is significant, this may be important; however, in many datasets where the diffusion is very small, it is possible to use an explicit method for the diffusion. Furthermore, since both the diffusion and reaction are computed by convolutions, it is possible to combine them into a 3×3 convolution (see [43] and [20] for additional discussions). This yields a structure that is very similar to a classical Convolutional Residual Network that replaces the diffusion and reaction steps. For the datasets used in this paper, we noted that this modest architecture was sufficient to obtain results that were close to state-of-the-art.

3.5 Implementing the ADR Network

Implementing the diffusion and reaction terms, either jointly or combined, we use a standard Convolutional Residual Network. The advection term is implemented by using the `sampleGrid` command in PyTorch [41], which uses an efficient implementation to interpolate the images.
While the network can be used as described above, we found that better results can be obtained by denoising the output of the network. To this end, we have used a standard UNet and applied it to the output. As we show in our numerical experiments, this allows us to further improve downstream performance. The complete network is summarized in Algorithm 1.

Algorithm 1 The ADR network

Set $\mathbf{I}_0 \leftarrow M(\mathbf{q}_k, \boldsymbol{\theta}_o)$, $\mathbf{Q}_k$ as in equation 11.
for $j = 0, 1, ...m-1$ **do**
  Diffusion-Reaction $\mathbf{I}_{DR} \leftarrow \mathcal{D}_{\boldsymbol{\kappa}_j} \mathcal{R}_{\boldsymbol{\theta}_j} \mathbf{I}_j$
  Compute displacement $\mathbf{U}_j = RN(\mathbf{I}_{DR}, \boldsymbol{\eta}_j)$ as in equation 12
  Push the image $\mathbf{I}_{j+1} = \mathcal{A}(\mathbf{U}_j)\mathbf{I}_{DR}$
end for
Set $\mathbf{q}_{k+\ell} = M(\mathbf{I}_m, \boldsymbol{\theta}_T)$
(Optional) Denoise $\mathbf{q}_{k+\ell} = \text{UNet}(\mathbf{q}_{k+\ell})$

4 Experiments

Our goal is to develop architectures that perform well for scientific-related datasets that require advection. In our experiments, we use two such datasets, CloudCast [70], and the Shallow Water Equation in PDEbench [51]. However, our ADRNet can also be used for the solution of video prediction. While such problems behave differently than scientific datasets, we show that our ADRNet can perform reasonably well for those applications as well. Below, we elaborate on the utilized datasets. We run our codes using a single NVIDIA RTX-A6000 GPU with 48GB of memory. Besides the experimental results reported in this Section, we provide additional results, from ablations to visualizations and measured runtimes in Appendix A.

4.1 Datasets

We now describe the datasets considered in our experiments, which are categorized below.

Scientific Datasets: We consider the following datasets which arise from scientific problems and communities: (1) **SWE.** The shallow-water equations are derived from the compressible Navier-

Stokes equations. The data is comprised of 900 sets of 101 images, each of which is a time step. (2) **CloudCast.** The CloudCast dataset comprises 70,080 satellite images captured every 15 minutes and has a resolution of 3712×3712 pixels, covering the entire disk of Earth.

Video Prediction Datasets: These datasets are mainly from the Computer Vision community, where the goal is to predict future frames in videos. The datasets are as follows: (1) **Moving MNIST.** The Moving MNIST dataset is a synthetic video dataset designed to test sequence prediction models. It features 20-frame sequences where two MNIST digits move with random trajectories. (2) **KITTI.** The KITTI is a widely recognized dataset extensively used in mobile robotics and autonomous driving, and it also serves as a benchmark for computer vision algorithms.

The statistics of the datasets are summarized in Table 1, and in Appendix A, we provide results on additional datasets, namely TaxiBJ [70] and KTH [45].

4.2 Evaluation

Ranking of Methods. Throughout all experiments where other methods are considered, we rank the top 3 methods using the color scheme of **First**, **Second**, and **Third**.

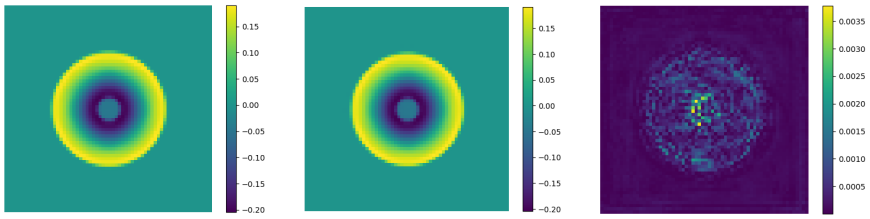
Performance on Scientific Datasets. We start our comparisons with the SWE and CloudCast datasets. These datasets fit the description of our ADRNet as future images depend on the history alone (that is, the history should be sufficient to recover the future). Indeed, Table 2 and Table 3 show that our ADRNet performs much better than other networks for these goals. Additional experiments on the Navier-Stokes dataset are provided in Appendix A.3

Table 2: Results on PDEBench SWE Dataset.

Method	NRMSE ↓
UNet [52]	8.3e-2
PINN [52]	1.7e-2
MPP-AVIT-TI [36]	6.6e-3
ORCA-SWIN-B [47]	6.0e-3
FNO [52]	4.4e-3
MPP-AVIT-B [36]	**2.4e-3**
MPP-AVIT-L [36]	**2.2e-3**
ADRNet	**1.3e-4**

Table 3: Results on CloudCast dataset.

Method	SSIM (↑)	PSNR (↑)
AE-ConvLSTM [70]	**0.66**	**8.06**
MD-GAN [67]	**0.60**	**7.83**
TVL1 [56]	0.58	7.50
Persistent [70]	0.55	7.41
ADRNet	**0.83**	**38.17**



Ground-Truth Prediction Error

Figure 4: Prediction and error for the SWE problem using our ADRNet.

Examples of the predictions of the SWE dataset and the CloudCast datasets are plotted in Figure 4 and Figure 6. For the SWE dataset, the errors are very small and close to machine precision. For CloudCast, the data is noisy, and it is not clear how well it should fit. Predicting a single-time step, while useful, has limited applicability. Our goal is to push the prediction for longer, hence providing an alternative to expensive numerical integration. The results with SWE for long-time prediction (i.e. using the same 10 timesteps from history to predict 5, 10, 20, 50 timesteps in the future) are presented in Table 4, together with a comparison of the celebrated state of the art FNO method [33] where we see that our model performs well even for long-time prediction. As summarized in Table 4, ADRNet outperforms other models on long-range predictions for the SWE dataset. This is further illustrated in Figure 5, which shows the prediction accuracy of ADRNet at future time steps of 10, 20, and 50, demonstrating its ability to capture extended dependencies effectively.

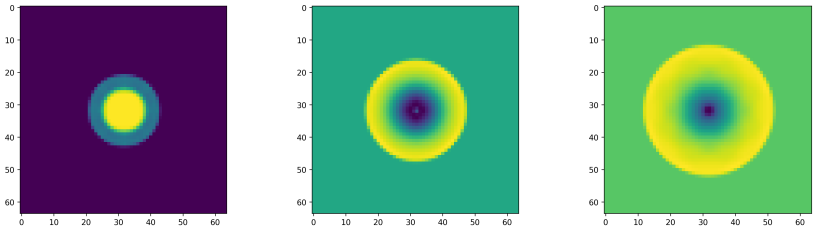


Figure 5: Predictions on the SWE dataset at future time steps 10,20,50 (left to right). Our results demonstrate the large receptive field learned by ADRNet.

Table 4: Comparison of ADRNet and FNO on long range-predictions. We consider the prediction of different numbers of steps given different numbers of input steps. For example, the setting of 10 input steps and 5 prediction steps is denoted by $10 \rightarrow 5$.

Metric	$10 \rightarrow 5$		$10 \rightarrow 10$		$10 \rightarrow 20$		$10 \rightarrow 50$	
	ADRNet	FNO	ADRNet	FNO	ADRNet	FNO	ADRNet	FNO
MSE ↓	9.2e-08	4.0e-07	1.5e-07	5.8e-07	2.1e-07	6.7e-07	8.5e-07	1.4e-06
nMSE ↓	8.5e-08	3.7e-07	1.4e-07	5.4e-07	1.9e-07	6.2e-07	7.8e-07	1.3e-06
RMSE ↓	3.0e-04	6.3e-04	3.9e-04	7.6e-04	4.5e-04	8.1e-04	9.2e-04	1.2e-03
nRMSE ↓	2.9e-04	6.1e-04	3.7e-04	7.3e-04	4.4e-04	7.8e-04	8.8e-04	1.1e-03
MAE ↓	2.0e-04	2.8e-04	1.7e-04	3.7e-04	1.9e-04	3.6e-04	4.1e-04	5.7e-04

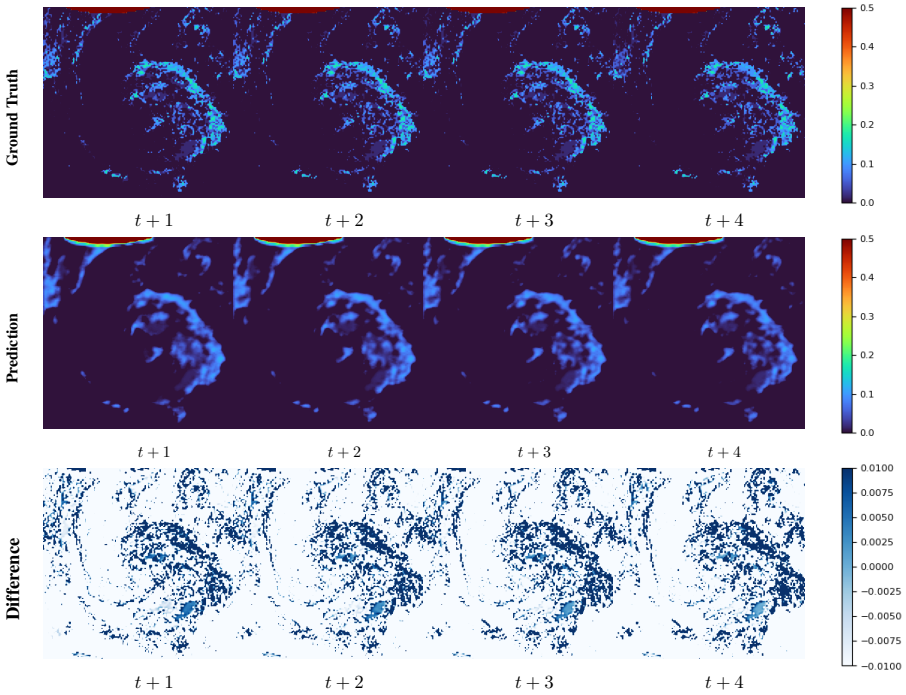


Figure 6: Example of the forecast by ADRNet compared to the ground truth over four time steps. 't' denotes the forecast time in 15-minute intervals. We use four input images to predict the subsequent four images. While changes in the CloudCast dataset in the two subsequent frames are slow, ADRNet achieved superior results in terms of PSNR and SSIM. A quantitative comparison is shown in Table 3.

To evaluate the generalization capability of ADRNet, we conducted additional experiments using pre-trained ADRNet models on different datasets. These experiments, detailed in Appendix A.6, demonstrate the effectiveness of ADRNet in transfer learning tasks, such as adapting from the Navier-Stokes dataset to the SWE dataset.

Table 5: Moving MNIST.

Method	MSE ↓	MAE ↓
MSPred [58]	34.4	-
MAU [6]	27.6	-
PhyDNet [19]	24.4	70.3
SimVP [53]	23.8	68.9
CrevNet [69]	22.3	-
TAU [54]	19.8	**60.3**
SwinLSTM [55]	**17.7**	-
IAM4VP [46]	**15.3**	**49.2**
ADRNet	**16.1**	**50.3**

Table 6: Results on KITTI.

Method	MS-SSIM ($\times 10^{-2}$) ↑		LPIPS ($\times 10^{-2}$) ↓	
	$t+1$	$t+3$	$t+1$	$t+3$
SADM [2]	83.06	72.44	14.41	24.58
MCNET [59]	75.35	63.52	24.04	37.71
CorrWise [15]	82.00	N/A	17.20	N/A
OPT [66]	82.71	69.50	12.34	20.29
DMVFN (w/o R) [23]	**88.06**	**76.53**	**10.70**	**19.28**
DMVFN [23]	**88.53**	**78.01**	**10.74**	**19.27**
ADRNet	**85.86**	**83.62**	**7.54**	**9.26**

Video Prediction Performance. We have used a number of video datasets to test our ADRNet. The results of two of them (Moving MNIST and KITTI) are reported in Table 5 and Table 6. We perform additional experiments for the KTH Action and TaxiBJ datasets in the appendix A. The moving MNIST dataset adheres to the assumptions of our ADRNet. Indeed, for this dataset, we obtain results that are very close to state-of-the-art methods.

The KTH Action dataset is more complex as not all frames can be predicted from the previous frames without generation power. Nonetheless, even for this dataset our ADRNet performs close to the state of the art. This limiting aspect of video synthesis is studied through experiments in appendix A.1.

5 Conclusion

In this paper, we have presented a new network for tasks that reside on a regular mesh that can be viewed as a multi-channel image. The method combines standard convolutions with a linear operator that transports information from one part of the image to another. The transportation vector field is learned from previous images (that is, history), allowing for information to pass from different parts of the image to others without loss. We combine this information within a diffusion-reaction process that can be coded by itself or by using a standard ResNet.

Acknowledgments

ME is funded by the Blavatnik-Cambridge fellowship, the Cambridge Accelerate Programme for Scientific Discovery, and the Maths4DL EPSRC Programme.

References

[1] U.M. Ascher. *Numerical methods for Evolutionary Differential Equations*. SIAM, Philadelphia, 2010.

[2] Xinzhu Bei, Yanchao Yang, and Stefano Soatto. Learning semantic-aware dynamics for video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 902–912, 2021.

[3] Jan Blechschmidt and Oliver G Ernst. Three ways to solve partial differential equations with neural networks—a review. *GAMM-Mitteilungen*, 44(2):e202100006, 2021.

[4] Thomas Bohnstingl, Stanisław Woźniak, Angeliki Pantazi, and Evangelos Eleftheriou. Online spatio-temporal learning in deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[5] Harold C Burger, Christian J Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *2012 IEEE conference on computer vision and pattern recognition*, pages 2392–2399. IEEE, 2012.

[6] Zheng Chang, Xinfeng Zhang, Shanshe Wang, Siwei Ma, Yan Ye, Xiang Xinguang, and Wen Gao. Mau: A motion-aware unit for video prediction and beyond. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 26950–26962. Curran Associates, Inc., 2021.

[7] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

[8] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 424–432, Cham, 2016. Springer International Publishing.

[9] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[10] Moshe Eliasof, Eldad Haber, and Eran Treister. PDE-GCN: Novel architectures for graph neural networks motivated by partial differential equations. *Advances in Neural Information Processing Systems*, 34:3836–3849, 2021.

[11] Moshe Eliasof, Eldad Haber, and Eran Treister. Adr-gnn: Advection-diffusion-reaction graph neural networks. *arXiv preprint arXiv:2307.16092*, 2023.

[12] L. C. Evans. *Partial Differential Equations*. American Mathematical Society, San Francisco, 1998.

[13] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29, 2016.

[14] J. Fohring, E. Haber, and L. Ruthutto. Geophysical imaging of fluid flow in porous media. *SIAM J. on Sci. Comp*, to appar, 2014.

[15] Daniel Geng, Max Hamilton, and Andrew Owens. Comparing correspondences: Video prediction with correspondence-wise losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3365–3376, 2022.

[16] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1983.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[18] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[19] Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11471–11481, 2020.

[20] Eldad Haber, Keegan Lensink, Eran Treister, and Lars Ruthotto. Imexnet a forward stable deep neural network. In *International Conference on Machine Learning*, pages 2525–2534. PMLR, 2019.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[22] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. *Advances in neural information processing systems*, 31, 2018.

[23] Xiaotao Hu, Zhewei Huang, Ailin Huang, Jun Xu, and Shuchang Zhou. A dynamic multi-scale voxel flow network for video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6121–6131, 2023.

[24] Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1771–1779. PMLR, 2017.

[25] C.T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, 1995.

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[28] Keegan Lensink, Bas Peters, and Eldad Haber. Fully hyperbolic convolutional neural networks. *Research in the Mathematical Sciences*, 9(4):60, 2022.

[29] M. Lentine, J. T. Gretarsson, and R. Fedkiw. An Unconditionally Stable Fully Conservative Semi-Lagrangian Method. *Journal of Computational Physics*, 230:2857–2879, 2011.

[30] R.J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhauser, 1990.

[31] Siyuan Li, Zedong Wang, Zicheng Liu, Cheng Tan, Haitao Lin, Di Wu, Zhiyuan Chen, Jiangbin Zheng, and Stan Z Li. Moganet: Multi-order gated aggregation network. In *The Twelfth International Conference on Learning Representations*, 2023.

[32] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

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

[34] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

[35] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[36] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for physical surrogate models. In *NeurIPS 2023 AI for Science Workshop*, 2023.

[37] J. Nagy and P.C. Hansen. *Deblurring Images*. SIAM, Philadelphia, 2006.

[38] Marc Oliu, Javier Selva, and Sergio Escalera. Folded recurrent neural networks for future video prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 716–731, 2018.

[39] Sergiu Oprea, Pablo Martinez-Gonzalez, Alberto Garcia-Garcia, John Alejandro Castro-Vargas, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Antonis Argyros. A review on deep learning techniques for video prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):2806–2826, 2020.

[40] Keiron O'shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems*, 2017.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[43] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *arXiv preprint arXiv:1804.04272*, 2018.

[44] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pages 1–13, 2019.

[45] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36. IEEE, 2004.

[46] Minseok Seo, Hakjin Lee, Doyi Kim, and Junghoon Seo. Implicit stacked autoregressive model for video prediction. *arXiv preprint arXiv:2303.07849*, 2023.

[47] Junhong Shen, Liam Li, Lucio M. Dery, Corey Staten, Mikhail Khodak, Graham Neubig, and Ameet Talwalkar. Cross-modal fine-tuning: Align then refine. ICML, 2023.

[48] Xiaoyu Shi, Zhaoyang Huang, Weikang Bian, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Videoflow: Exploiting temporal cues for multi-frame optical flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12469–12480, 2023.

[49] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional lstm network: a machine learning approach for precipitation nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 802–810, Cambridge, MA, USA, 2015. MIT Press.

[50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[51] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning, 2023.

[52] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[53] Cheng Tan, Zhangyang Gao, Siyuan Li, and Stan Z Li. Simvp: Towards simple yet powerful spatiotemporal predictive learning. *arXiv preprint arXiv:2211.12509*, 2022.

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

[55] Song Tang, Chuang Li, Pu Zhang, and RongNian Tang. Swinlstm: Improving spatiotemporal prediction accuracy using swin transformer and lstm. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13470–13479, 2023.

[56] Isabel Urbich, Jörg Bendix, and Richard Müller. A novel approach for the short-term forecast of the effective cloud albedo. *Remote Sensing*, 10(6):955, 2018.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[58] Angel Villar-Corrales, Ani J. Karapetyan, Andreas Boltres, and Sven Behnke. Mspred: Video prediction at multiple spatio-temporal scales with hierarchical recurrent networks. In *British Machine Vision Conference*, 2022.

[59] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *International Conference on Learning Representations*, 2017.

[60] Yunbo Wang, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and Philip S Yu. PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5123–5132. PMLR, 10–15 Jul 2018.

[61] Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. Eidetic 3d LSTM: A model for video prediction and beyond. In *International Conference on Learning Representations*, 2019.

[62] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[63] Yunbo Wang, Jianjin Zhang, Hongyu Zhu, Mingsheng Long, Jianmin Wang, and Philip S Yu. 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*, pages 9154–9162, 2019.

[64] Henglai Wei, Xiaochuan Yin, and Penghong Lin. Novel video prediction for large-scale scene using optical flow. *arXiv preprint arXiv:1805.12243*, 2018.

[65] Christopher K Wikle. Comparison of deep neural networks and deep hierarchical models for spatio-temporal data. *Journal of Agricultural, Biological and Environmental Statistics*, 24(2):175–203, 2019.

[66] Yue Wu, Qiang Wen, and Qifeng Chen. Optimizing video prediction via video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17814–17823, 2022.

[67] Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2364–2373, 2018.

[68] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

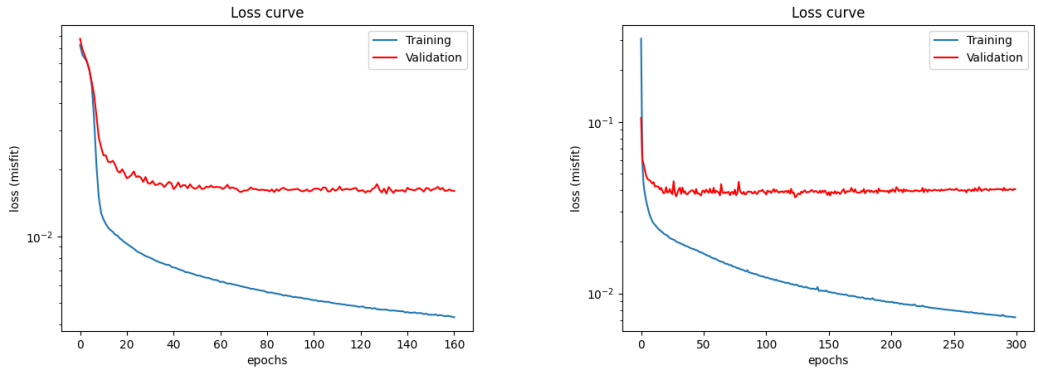
[69] Wei Yu, Yichao Lu, Steve Easterbrook, and Sanja Fidler. Efficient and information-preserving future frame prediction and beyond. In *International Conference on Learning Representations*, 2020.

[70] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

A Ablation Studies and Additional Experiments

A.1 Limited Generative Synthesis

Two real-life video datasets are taken to predict future time frames. Their statistics can be found in Table 7. The specific challenge posed by these datasets is due to the dissimilarity in the train and test sets. This is evident from the notable difference in the training and validation/test losses, which can be seen in Figure 7. The validation loss starts increasing with more epochs. For example, the KTH Action uses the movement behavior of 16 people for training while the models are tested on the movement behavior of 9 other people in a slightly altered scenario. So, we can say that the problem is to learn the general logic to predict unseen scenarios. Thus generative capability of a model could be crucial for better prediction.



KTH Action TaxiBJ

Figure 7: Bias in training and testing samples in KTH Action and TaxiBJ datasets

KTH Action The KTH dataset features 25 individuals executing six types of actions: walking, jogging, running, boxing, hand waving, and hand clapping. Following methodologies established in references [59, 61], we utilize individuals 1-16 for training and individuals 17-25 for testing. The models are trained to predict the subsequent 20 frames based on the preceding 10 observations.

TaxiBJ TaxiBJ is a collection of real-world GPS spatiotemporal data of taxis recorded as frames of 32x32x2 heat maps every half an hour, quantifying traffic flow in Beijing. We split the whole dataset into a training set and a test set as described in [70]. We train the networks to predict 4 future time frames from 4 observations.

Results Our model is easily able to predict and outperform the state-of-the-art models in real-life video examples as well. The results for KTH Action and TaxiBJ can be seen in 8 and 9.

Table 7: Additional Dataset Statistics: Details on Training and Testing, Image Sequences, and Resolutions

Dataset	N_{train}	N_{test}	(C, H, W)	$History$	$Prediction$
KTH Action	5200	3167	(1, 128, 128)	10	20
TaxiBJ	19627	1334	(2, 32, 32)	4	4

A.2 CloudCast

The CloudCast dataset is used for multiple long-range predictions like 4, 8, 12, and 16 timesteps. It can be noticed that even if the MSE or the quality degrades, the degradation is noticeably minimal. It can be seen in Table 10, the figures for predicting 16 steps in future is still better than the state of art for 4 steps in future.

Table 8: Comparison of Our Method for KTH Action Dataset.

Method	SSIM ↑	PSNR (dB) ↑
ConvLSTM [49]	0.712	23.58
PredRNN [62]	0.839	27.55
CausalLSTM [60]	0.865	28.47
MSPred [58]	0.930	28.93
E3D-LSTM [61]	0.879	29.31
SimVP [53]	0.905	33.72
TAU [54]	0.911	34.13
SwinLSTM [55]	0.903	34.34
ADRNet	0.808	31.58

Table 9: Comparison of Our Method for TaxiBJ Dataset.

Method	MSE ↓	MAE ↓	SSIM ↑
ST-ResNet [70]	0.616	-	-
VPN [24]	0.585	-	-
ConvLSTM[49]	0.485	17.7	0.978
FRNN [38]	0.482	-	-
PredRNN [62]	0.464	17.1	0.971
CausalLSTM [60]	0.448	16.9	0.977
MIM [63]	0.429	16.6	0.971
E3D-LSTM [61]	0.432	16.9	0.979
PhyDNet [19]	0.419	16.2	0.982
SimVP [53]	0.414	16.2	0.982
SwinLSTM [55]	0.390	-	0.980
IAM4VP [46]	0.372	16.4	0.983
TAU [54]	0.344	15.6	0.983
ADRNet	0.445	16.6	0.975

Table 10: Results for CloudCast dataset. Comparison of our model (ADRNet) with state of art models

ADRNet Predictive performance				
Metric	t + 4	t + 8	t + 12	t + 16
MSE (↓)	0.015	0.016	0.018	0.019
SSIM (↑)	0.83	0.79	0.76	0.74
PSNR (↑)	38.17	37.89	37.35	37.23

A.3 Navier-Stokes Dataset

To further demonstrate the effectiveness of our ADRNet on scientific data, we conduct experiments on an additional dataset from PDEbench, specifically the Navier-Stokes equations. This large dataset consists of 21,000 images, each with a resolution of 512x512. Our ADRNet ranks second among various methods, positioning it in line with state-of-the-art approaches, as shown in Table 11 and Figure 8. Additional metrics on this dataset obtained with UNet, FNO, and our ADRNet are reported in Table 12.

Table 11: Comparison of methods based on normalized Mean Squared Error (nMSE). Our ADRNet achieves competitive performance, ranking second among state-of-the-art methods.

Method	UNet	FNO	MPP-AViT-TI	MPP-AViT-S	MPP-AViT-B	MPP-AViT-L	ADRNet (Ours)
nMSE (↓)	1.67	0.243	0.0312	0.0213	0.0172	0.0142	0.0168

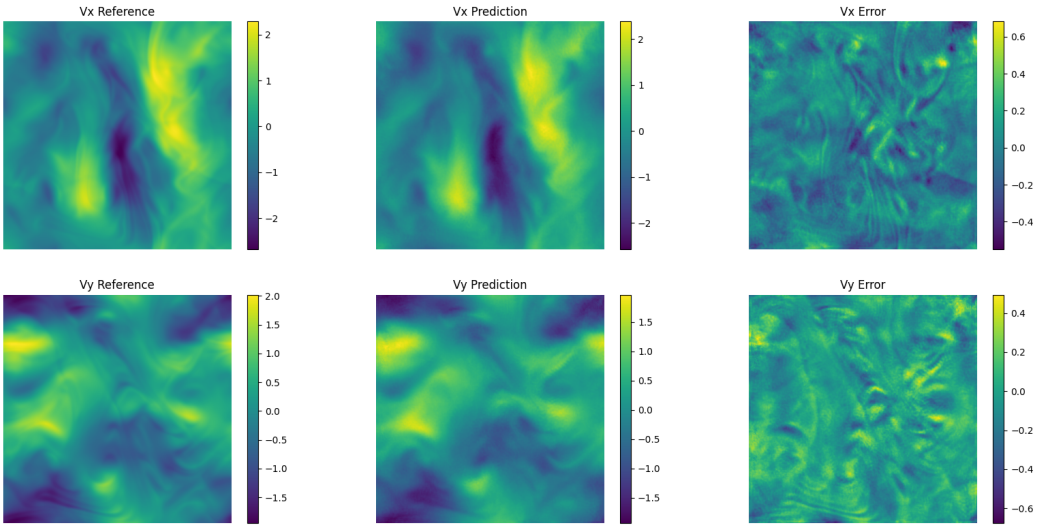


Figure 8: Visualization of ADRNet on Navier-Stokes dataset from PDEBench. Left column: ground-truth velocity maps. Middle column: ADRNet prediction velocity maps prediction. Right Column: Error between ground-truth and prediction.

Table 12: Comparison of different metrics on UNet, FNO, and our ADRNet, on the PDEBench Navier-Stokes Inviscid Compressible at $M = 0.1$ with Turbulent Initial Condition.

Metric	UNet	FNO	ADRNet (Ours)
RMSE	3.3×10^{-1}	2.8×10^{-1}	1.2676×10^{-1}
nRMSE	1.9×10^{-1}	1.6×10^{-1}	1.53×10^{-2}
Max Error	2.2×10^{0}	1.8×10^{0}	6.8378×10^{-1}
cRMSE	1.5×10^{-2}	1.2×10^{-2}	4.0903×10^{-3}
bRMSE	3.6×10^{-1}	2.8×10^{-1}	8.3314×10^{-2}
fRMSE (low)	6.5×10^{-2}	5.0×10^{-2}	1.4105×10^{-2}
fRMSE (mid)	3.2×10^{-2}	3.1×10^{-2}	2.6598×10^{-2}
fRMSE (high)	8.5×10^{-3}	6.5×10^{-3}	1.7822×10^{-3}

A.4 Order of Operator Splitting

We now show that numerically, the network computations are agnostic to the order of operator splitting. Following that, we perform an experiment to verify our theoretical derivation.

Numerical Behavior. The following derivation demonstrates the validity of the operator splitting approach used, as detailed below. In numerical PDEs, for an initial value problem (IVP) $dx/dt = Ax$, the solution is $x(t) = \exp(tA)x(0)$. For $dx/dt = Ax + Bx$, the solution is $x(t) = \exp(t(A + B))x(0)$. If matrix exponentials were treated as scalars, the solution would be $x(t) = \exp(tA)\exp(tB)x(0)$, implying that the order of operations (reaction-diffusion and advection) is invariant.

To analyze this, we review matrix exponentials. The matrix exponential $\exp(A)$ is defined by:

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k$$

For the sum of matrices A and B, the expansion is:

$$\exp(t(A + B)) = I + t(A + B) + 0.5t^2(A^2 + B^2 + AB + BA) + O(t^3)$$

For the product of two matrix exponentials:

$$\exp(tA) \cdot \exp(tB) = (I + tA + 0.5t^2A^2 + O(t^3)) \cdot (I + tB + 0.5t^2B^2 + O(t^3))$$

Generally, for matrices A and B, $AB \neq BA$ unless they share eigenvectors.

Expanding Equation (2) and collecting terms, we get:

$$\exp(tA) \cdot \exp(tB) = I + tA + tB + 0.5t^2(A^2 + B^2 + 2AB) + O(t^3)$$

Comparing this with:

$$\exp(t(A + B)) = I + tA + tB + 0.5t^2(A^2 + B^2 + AB + BA) + O(t^3)$$

we find the approximation error is $O(t^2)$:

$$\exp(tA) \cdot \exp(tB) - \exp(t(A + B)) = 0.5t^2(AB - BA) + O(t^3)$$

This error depends on how AB differs from BA.

Regarding the sequence of operations, changing the order of A and B yields:

$$\exp(t(A + B)) = \exp(tA) \cdot \exp(tB) + O(t^2) = \exp(tB) \cdot \exp(tA) + O(t^2)$$

Thus, the order of operations (advection vs. reaction-diffusion) does not fundamentally affect numerical accuracy.

Advection-Diffusion-Reaction (ADR) vs. Diffusion-Reaction-Advection (DRA). To verify our understanding of the numerical behavior of the order of splitting, we conduct an experiment that compares two possible orders of operations: advection followed by reaction-diffusion, and vice versa, on the Moving MNIST dataset. Figure 9 shows that the convergence plots and obtained predictions for the two possible orderings are similar. In addition, we report the obtained test set performance on the Moving MNIST dataset obtained with the two considered variants (ADRNet and DRANet) in Table 13.

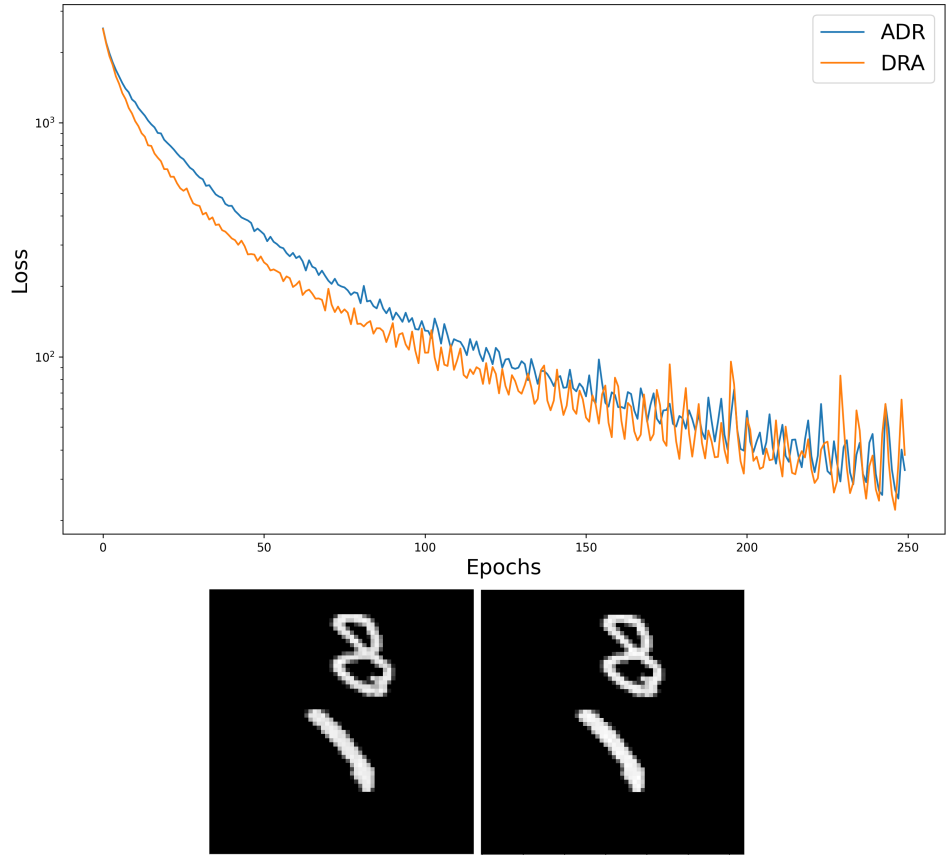


Figure 9: Convergence plot comparing the order of operations: ADR vs. DRA, along with examples of predictions made by the models (from left to right, respectively). The different order of layers in the ADR model yields similar results, consistent with our theoretical analysis.

A.5 Advection vs. Dilated Convolutions

Dilated convolutions are known to be a mechanism that allows a wide field of view [68], which is also obtained by our Advection operator. To demonstrate the benefit of our advection operator, we consider two possible uses of dilated convolutions: (i) Using dilation instead of the diffusion mechanism, combined with our advection operator, and (ii) using dilation instead of advection. We compare their performance Moving MNIST dataset, which requires the ability to transport information across distant pixels. Our results are presented in Table 14, and visualized in Figure 10. As can be seen, the performance obtained when utilizing the advection operator is significantly improved, both in terms of training convergence and the obtained downstream performance, highlighting the importance of advection in tasks that require long-range transportation of features.

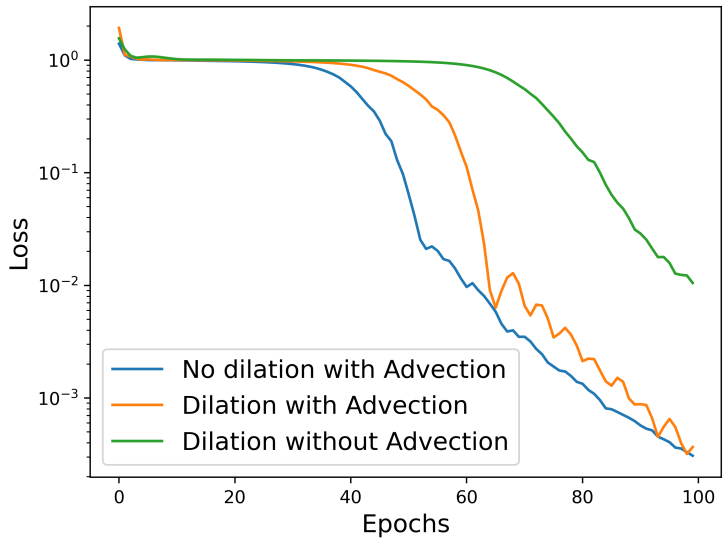


Figure 10: Comparison of ADRNet vs. using dilated convolutions. Blue: ADRNet. Orange: Using dilation to implement diffusion, coupled with the advection operator. Green: Using dilation in place of advection, with standard convolutions for reaction-diffusion. ADRNet demonstrates superior convergence.

Method	MSE (↓)	MAE (↓)
ADRNet	16.1	50.3
DRANet	16.2	50.3

Table 13: Moving MNIST Test set performance with ADR vs. DRA operator splitting ordering. Both cases yield similar results.

Method	MSE (↓)	MAE (↓)
Dilation with Advection	16.6	51.1
Dilation without Advection	25.7	72.8
ADRNet	16.1	50.3

Table 14: A comparison of two possible uses of Dilated Convolutions vs. our ADRNet, on the Moving MNIST test set.

A.6 Generalization and Transferability

Our ADRNet can be classified as a physics-inspired neural network [43, 10]. Thus, it is by construction equipped with an implicit bias and mathematically-grounded behavior. It is therefore, interesting to study whether training ADRNet on one dataset and task can be useful for a different dataset and task. To this end, in Figure 11 illustrates the performance of ADRNet on the SWE dataset, leveraging a pre-trained model initially trained on the Navier-Stokes dataset from PDEBench. Our results indicate that using a pre-trained ADRNet, even with minimal fine-tuning (a single linear adaptation layer), significantly improves predictive accuracy compared to a randomly initialized model. This outcome highlights the adaptability of ADRNet and generalization capability across scientific datasets, showcasing its potential for tasks requiring domain transfer.

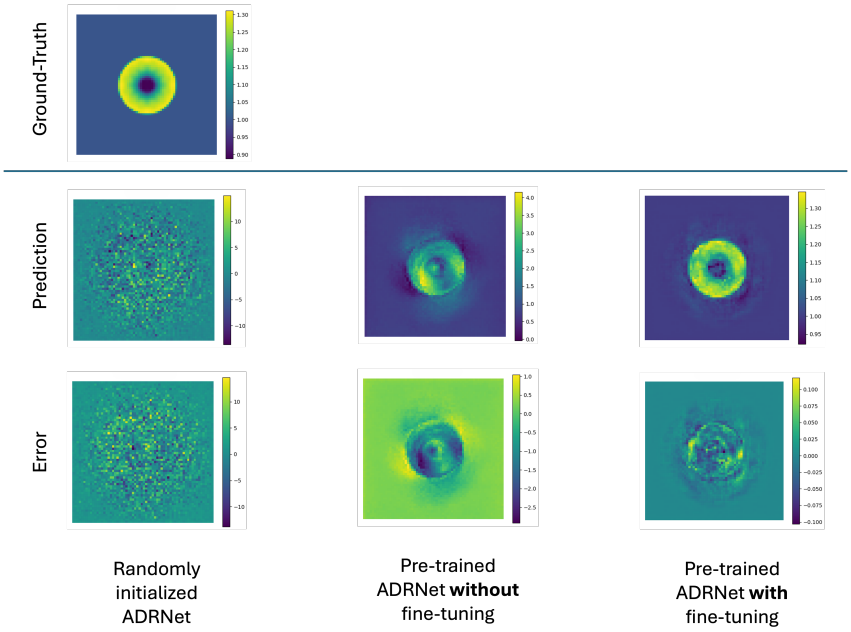


Figure 11: Results on SWE dataset using ADRNet pre-trained on the Navier-Stokes dataset. Using a pre-trained ADRNet is beneficial both with and without fine-tuning a single linear adaptation layer compared with a randomly initialized ADRNet.

A.7 Illustration of the Learned Velocity Fields

In Figure 12, we illustrate the learned advection field and attention maps. The obtained velocity fields and their application confirm the concept described in Figure 1 and Figure 2 in the paper. We note that the advection operator works on all channels in the embedded space. For the example at hand (Moving MNIST), we have 64 channels and 5 convolution layers, that blend 10 previous time steps given as the input. To generate the figure, we inspect one of the channels across all layers, and plot a quiver plot of the advection field. This quiver plot shows the direction in which the advection is guided to solve the task in the moving MNIST dataset. In addition, we plot the absolute value of the advection field. The absolute value of the advection field is equivalent to an attention map, because it shows the areas in the original image that move the most to generate the final image. Including this figure allows us to visualize which areas in the input need to be moved to obtain the desired target. As can be seen from Figure 5, the pixels that correspond to the digits in the input images are the ones that obtain larger values of displacement in the learned advection field – this result is in accordance with the concept of learning advection fields, as done in our ADRNet.

A.8 Runtimes

In Table 15, we provide a runtimes comparison of ResNet and our ADRNet. The runtimes were measured using an NVIDIA RTX-A6000 GPU, with an ADRNet and a ResNet, both with 32 hidden

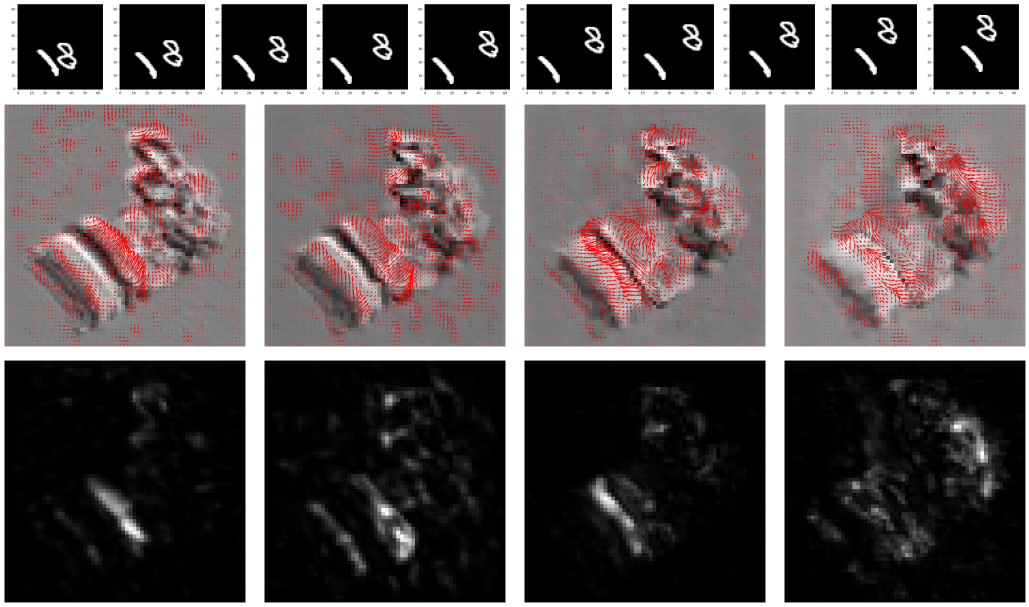


Figure 12: Top Row: 10 input time-steps to ADRNet. Middle Row: Quiver maps of the learned advection fields. Bottom Row: Attention maps are defined by the magnitude of the advection field at each pixel. The advection fields and attention maps are in layers $0, \ldots, 3$, from left to right.

dimensions, 10 input features, 1 output feature, batch size of 32, and 2 layers, on a varying image size input. The results show that while our ADRNet requires more runtimes than standard CNNs, it maintains a reasonable computational cost.

Image Size	32×32	64×64	128×128	256×256
ResNet	4.2 / 0.6	20.3 / 10.4	79.2 / 45.8	192.8 / 58.2
ADRNet (Ours)	12.8 / 3.7	53.3 / 19.1	175.8 / 72.8	484.3 / 179.1

Table 15: Runtimes (training/inference) comparison of ResNet and our ADRNet, in milliseconds.

B Evaluation Metrics

Moving MNIST, KTH Action, TaxiBJ, CloudCast These specific video prediction datasets have been using MAE (Mean Absolute Error), MSE (Mean Squared Error), SSIM (Structural Similarity) and PSNR (Peak Signal-to-Noise Ratio). The evaluated SSIM and PSNR are averaged over each image. The MSE and MAE have a specific way to calculate, where the pixel-wise evaluation values are summed up for all the pixels in the image.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}\sum_{c=1}^{C}(y-\hat{y})^2 \tag{16}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}\sum_{c=1}^{C}|y-\hat{y}| \tag{17}$$

$$\text{PSNR} = \frac{1}{N}\sum_{i=1}^{N} 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right) \tag{18}$$

$$\text{SSIM(x,y)} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{19}$$

$$\overline{\text{SSIM}} = \frac{1}{N}\sum_{i=1}^{N}\text{SSIM}(x,y) \tag{20}$$

where:

N is the number of images in the dataset,
H is the height of the images,
W is the width of the images,
C is the number of channels (e.g., 3 for RGB images),
y is the true pixel value at position (i,h,w,c), and
$\hat{y}$ is the predicted pixel value at position (i,h,w,c).
MAX is the maximum possible pixel value of the image (e.g., 255 for an 8-bit image),
MSE is the Mean Squared Error between the original and compressed image.
μ_x is the average of x,
μ_y is the average of y,
σ_x^2 is the variance of x,
σ_y^2 is the variance of y,
σ_{xy} is the covariance of x and y,
$C_1 = (K_1 L)^2$ and $C_2 = (K_2 L)^2$ are two variables to stabilize the division with weak denominator,
L is the dynamic range of the pixel values (typically, this is 255 for 8-bit images),
K_1 and K_2 are small constants (typically, $K_1 = 0.01$ and $K_2 = 0.03$).

PDEBench-SWE PDEBench uses the concept of pixel-wise mean squared error (MSE) and normalized mean squared error (nMSE) to validate scaled variables in simulated PDEs. Along with these, we also use root mean squared error (RMSE) and normalized root mean squared error (nRMSE).

$$\text{MSE} = \frac{1}{N \cdot H \cdot W \cdot C}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}\sum_{c=1}^{C}(x-\hat{x})^2 \tag{21}$$

$$\text{nMSE} = \frac{1}{N \cdot H \cdot W \cdot C} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{C} \frac{(x - \hat{x})^2}{x^2} \tag{22}$$

$$\text{RMSE} = \frac{1}{N} \sqrt{\frac{1}{H \cdot W \cdot C} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{C} (x - \hat{x})^2} \tag{23}$$

$$\text{nRMSE} = \frac{1}{N} \sqrt{\frac{1}{H \cdot W \cdot C} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{C} \frac{(x - \hat{x})^2}{x^2}} \tag{24}$$

where:

N is the number of images in the dataset,
H is the height of the images,
W is the width of the images,
C is the number of channels (e.g., 3 for RGB images),
x is the true pixel value at position (n, h, w, c), and
$\hat{x}$ is the predicted pixel value at position (n, h, w, c).

KITTI

$$\text{MS-SSIM}(x, y) = [l_M(x, y)]^{\alpha_M} \prod_{j=1}^{M} [c_j(x, y)]^{\beta_j} [s_j(x, y)]^{\gamma_j} \tag{25}$$

$$\overline{\text{MS-SSIM}} = \frac{1}{N} \sum_{i=1}^{N} \text{MS-SSIM}(x, y) \tag{26}$$

where:

N is the number of images in the dataset,
$l_M(x, y)$ is the luminance comparison at the coarsest scale M,
$c_j(x, y)$ is the contrast comparison at scale j,
$s_j(x, y)$ is the structure comparison at scale j,
$\alpha_M, \beta_j, \gamma_j$ are the weights applied to the luminance, contrast, and structure terms at each scale respectively,
M is the number of scales used in the comparison.

The luminance, contrast, and structure comparisons are given by:

$$l(x, y) = \frac{2\mu_x \mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$
$$c(x, y) = \frac{2\sigma_x \sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$
$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3}$$

where:

μ_x, μ_y are the local means of x and y,
σ_x, σ_y are the local standard deviations of x and y,
σ_{xy} is the local covariance of x and y,
C_1, C_2, C_3 are constants to stabilize the division.

$$\text{LPIPS}(x, \hat{x}) = \sum_{l} \frac{1}{H_l W_l} \sum_{h=1}^{H_l} \sum_{w=1}^{W_l} \| w_l \odot (\phi_l(x)_{hw} - \phi_l(\hat{x})_{hw}) \|_2^2 \tag{27}$$

$$\overline{\text{LPIPS}} = \frac{1}{N} \sum_{i=1}^{N} \text{LPIPS}(x, \hat{x}) \tag{28}$$

where:

N is the number of images in the dataset,
$\phi_l(x)$ is the activation of the l-th layer of a deep network for the image x,
$\phi_l(\hat{x})$ is the activation of the l-th layer of a deep network for the image $\hat{x}$,
w_l is a learned weight vector for the l-th layer,
H_l and W_l are the height and width of the l-th layer activations,
$\odot$ denotes element-wise multiplication.

C Hyperparameter Settings and Computational Resources

C.1 ADRNet Training on PDEBench-SWE

Hyperparameter	Symbol	Value
Learning Rate	η	$1e-04$
Batch Size	B	64
Number of Epochs	N	200
Optimizer	-	Adam
Number of Layers	-	1
Hidden Channels	-	128
Activation Function	-	SiLU

Table 16: Neural Network Hyperparameters

C.2 ADRNet Training on Other Datasets

Hyperparameter	Symbol	Value
Learning Rate	η	$2e-06$
Batch Size	B	16
Number of Epochs	N	1000
Optimizer	-	Adam
Number of Layers	-	8
Hidden Channels	-	192
Activation Function	-	SiLU
Learning Rate Scheduler	-	ExponentialLR

Table 17: Neural Network Hyperparameters

C.3 Computational Resources

All our experiments are conducted using an NVIDIA RTX-A6000 GPU with 48GB of memory.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The contribution paragraph in the introduction (Section 1) explicitly describes the main contributions of the paper. The claims are supported by theory (Section 3) and experiments (Section 4). The results show our ADRNet exceptionally beating state-of-the-art methods (Section 4.2) in spatiotemporal prediction with a wide variety of datasets.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The proposed method is very much generalizable for spatiotemporal predictions. Yet, the evaluation section (Section 4) and appendix (Appendix A.1) include separate paragraphs discussing the limiting generative synthesis with experiments.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Section 3 presents our main theoretical motivation, and it contains the full set of assumptions in devising our ADRNet. However, no theoretical result, theorem or lemma is proposed.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 3 fully describes our proposed method, and Appendix A contains experimental details to reproduce our results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We offer comprehensive details regarding the implementation and evaluation of our ADRNet (Appendix A), and our code is available at https://github.com/Siddharth-Rout/deepADRnet.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix C.1 contains all the details, including the hyperparameters and other experimental choices.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We follow the same report practice as in preceding papers in the field, which also do not report the standard deviation of their results. To ensure fairness, we still ran our model on 5 different seeds and reported the average result.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C.3 describes the type of GPUs used in the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conforms with the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The work is fundamental and not tied to any particular application. Hence, there is no associated societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have credited the owners of the assets and the licenses in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not introduce new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.