AUTONOMOUS SHAPING OF LATENT SPACES FROM REDUCED PDES FOR PHYSICAL NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Numerical simulations based on partial differential equations (PDEs) are a central tool for a wide variety of scientific and engineering applications. Due to their challenging nature, many numerical methods rely on a reduced representation of degrees of freedom and adopt an efficient solver that solves the PDEs in the reduced space. In general, however, it is extremely challenging to faithfully preserve the correct solutions over long timespans with reduced representations. This problem is particularly pronounced for solutions with large amounts of small scale features. To address this, data-driven methods can learn to restore the details as required for accurate solutions of the underlying PDE problem. This paper studies the training of deep neural network models that autonomously interact with a PDE solver to achieve the desired solutions. In contrast to previous work, we do not constrain the PDE solver but instead give the neural network complete freedom to shape the PDE solutions as degrees of freedom of a latent space. Surprisingly, this autonomy allows the neural network to discover new physical dynamics that allow for better performance in the given learning objectives. We showcase that this approach allows the trained encoder to transform accurate solutions into abstract yet physical reduced representations, which are significantly different from conventional down-sampling results. Moreover, we demonstrate that our decoder outperforms models trained with different methodologies in terms of restoration accuracy.

1 INTRODUCTION

Partial differential equations (PDEs) are a central tool to model a wide range of science and engineering problems, from blood flow simulations (Johnston et al., 2004), over aerodynamics (Cummins et al., 2018), to climate and weather (Randall et al., 2007). For those PDE problems, many numerical methods have been developed to achieve desired solutions as efficiently as possible. However, a central challenge of utilizing these numerical methods lies in the fact that the PDE problems of real-world scenarios are extremely costly to resolve. The desired solutions often require very fine degrees of freedom both temporally and spatially. Thus, solving the given PDE problem in a reduced space is often considered as an alternative, but at the price of approximation errors and a lack of fine details.

To address these difficulties, data-driven methods were proposed to learn to restore details and mitigate the errors introduced by the reduced representation (Morton et al., 2018; Wiewel et al., 2020; Kochkov et al., 2021). Despite the ongoing research, there remains the central, open question of how to best shape the states in the reduced space such that the targeted solutions can be restored as accurately as possible. Here the reduced space typically has much less degrees of freedom, denoted d_r, compared to the targeted, accurate solution space's, d_f (i.e., $d_r \ll d_f$).

In this paper, we present a novel training method that autonomously shapes the latent space of a neural network model into an effective reduced representation for physical states of spatio-temporal PDE problems. For a given PDE, our solving procedure is made of (a) an encoder model that reduces the degrees of freedom of a physical state, (b) a physics solver running on the reduced state, and (c) a decoder, turning the reduced state output by the physics solver back into the target high resolution space. Here, the autonomy given to the neural network to shape the latent space turns out to play a key role, as the reduced, physical states can deviate from states that closely correspond to the fine solutions.

We adopt a differentiable physics approach for training the models. Within the interaction of the encoder, differentiable solver, and decoder for the given target solutions, we let the encoder model learn the latent space representation without imposing any constraints. Therefore, this setup provides the training procedure with the complete autonomy of shaping the reduced representation. Our experiments demonstrate that this autonomy leads to a significantly better performance for solving spatio-temporal PDE problems. Moreover, interestingly, we observe that the learned reduced representation differs considerably from a typical down-sampled representation. The training process discovers new, physical evolution that encode the necessary information for the learning objectives. The proposed training setup also tailors the learned reduced representation to allow for a more accurate decoding procedure. We demonstrate with two complex, non-linear PDE problems that the resulting models clearly outperform conventional and more tightly constrained models in terms of restoration accuracy. Fig. 1 shows the temporal evolution of an example reduced space representation that was learned by our autonomous training approach, as well as the restored solution from the reduced state, for a buoyancy-driven flow ruled by the 2D Navier-Stokes equations.

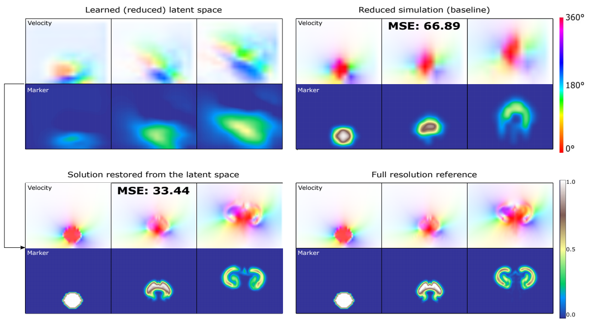


Figure 1: The learned latent space (top, left) strongly differs from reference (bottom, right) and from the reduced simulation (top, right). The restoration of details from it leads to a relative improvement of 50% w.r.t to the baseline.

Previous Work: The study of machine learning (ML) techniques for PDEs has a long history in science and engineering (Crutchfield & McNamara, 1987; Kevrekidis et al., 2003; Brunton et al., 2016). A popular direction in using ML for PDEs is to aim for replacement of entire PDE solvers by the neural network models that can efficiently approximate the solutions as accurately as possible (Lusch et al., 2018; Kim et al., 2019; Wang et al., 2020a; Bhattacharya et al., 2021). Instead of this pure ML-driven approach to solve for target PDEs, an alternative approach exists, which aims for hybrid methods that combine ML with traditional numerical methods. For example, a learned model can replace the most expensive part of an iterative PDE solver (Tompson et al., 2017) or supplement inexpensive yet under-resolved simulations (Um et al., 2018; Sirignano et al., 2020). These approaches have demonstrated the promising capabilities of ML to solve PDE problems for many different applications.

Recently, differentiable components for ML have received a significant amount of attention, particularly when training neural network models in recurrent setups for spatio-temporal problems (Amos & Kolter, 2017; de Avila Belbute-Peres et al., 2018; Toussaint et al., 2018; Chen et al., 2018; Schenck & Fox, 2018; Liang et al., 2019; Wang et al., 2020b; Um et al., 2020; Kochkov et al., 2021; Zhuang et al., 2021). Consequently, a variety of differentiable programming frameworks have been developed for different domains (Schoenholz & Cubuk, 2019; Hu et al., 2020; Innes et al., 2019; Holl et al., 2020). These differentiable frameworks allow neural networks to closely interact with PDE solvers, which provides the model with important feedback about the temporal evolution of the target problem from the recurrent evaluations. In order to provide gradients for shaping the latent spaces, we also make use of a differentiable framework in our training procedure.

Effectively utilizing latent spaces lies at the heart of many ML-based approaches for solving PDEs. A central role of the latent space is to embed important (often non-linear) information for the given training task into reduced degrees of freedom. With an autoencoder network architecture, for example, the latent space can be used for discovering interpretable, low-dimensional dynamical models and their associated coordinates from high-dimensional data (Champion et al., 2019). Moreover, thanks to their effectiveness in terms of embedding information and reducing the degrees of freedom, latent space solvers have been proposed for different problems such as advection-dominated systems (Maulik et al., 2021) and fluid flows (Wiewel et al., 2020; Fukami et al., 2021). While those studies typically focus on training equation-free evolution models, we focus on autonomously shaping the latent space that is integrated over time by a PDE solver. On the other hand, neural network models have been studied for the integration of a dynamical system with an ordinary differential equation (ODE) solver in the latent space (Chen et al., 2018). This targets general neural network approximations with a simple physical model in the form of an ODE, whereas we focus on learning tasks for complex nonlinear PDE systems and their latent spaces.

The ability to learn underlying PDEs has allowed neural networks to improve reduced, approximate solutions. Residual correction models are trained to address numerical errors of PDE solvers (Um et al., 2020). Details at sub-grid scales are improved via learning discretizations of PDEs (Bar-Sinai et al., 2019) and learning solvers (Kochkov et al., 2021) from high-resolution solutions. Moreover, the super-resolution approach with downsampled skip-connection multi-scale models has been used for recovering under-resolved turbulent flows from low-resolution data (Fukami et al., 2019). These methods, however, typically employ a constrained solution manifold for the reduced representation; i.e., the reduced solutions are produced using coarse-grained simulations with standard numerical methods, while our method shows the advantages of freely determining the latent space content.

2 Learning with Autonomous Latent Space Exploration

Our goal is to explore how the training process for neural networks can leverage the environment of a PDE in order to achieve a given learning objective. Let $\mathbf{f} \in \mathbb{R}^{d_f}$ and $\mathbf{r} \in \mathbb{R}^{d_r}$ denote two discretized solutions of this PDE, where $\mathbf{f}$ and $\mathbf{r}$ denote a fine representation and its reduced version respectively, with $d_r \ll d_f$. Considering a reduction function $\mathcal{E}$, called *encoder*, we can evaluate the transformation of the accurate fine solution into a reduced representation as $\mathcal{E}(\mathbf{f}) = \hat{\mathbf{r}}$. We can also consider a restoration function $\mathcal{D}$, called *decoder*, that transforms the reduced representation into an accurate solution as $\mathcal{D}(\mathbf{r}) = \hat{\mathbf{f}}$.

We focus on the numerical integration of a target PDE problem and indicate the temporal evolution of each state as a subscript. A reference solution trajectory integrated from a given initial state $\mathbf{f}_t$ at time t for N steps can be defined as a finite set of states $\{\mathbf{f}_t, \mathbf{f}_{t+1}, \cdots, \mathbf{f}_{t+N}\}$. Without loss of generality, each reference state is integrated over time with a fixed time step size using a numerical solver, i.e., $\mathbf{f}_{t+1} = \mathcal{P}_f(\mathbf{f}_t)$. Similarly, we integrate a reduced state $\mathbf{r}_t$ over time using a corresponding numerical solver at the reduced space, which we will call *reduced solver* henceforth, i.e., $\mathbf{r}_{t+1} = \mathcal{P}(\hat{\mathbf{r}}_t)$. Although we give the model the freedom to reshape its latent states, in this work, we consider the same PDE for the two different representations.

For a given reference solution trajectory, we can denote its approximate solution trajectory analogously as a finite set of states $\{\hat{\mathbf{f}}_t, \hat{\mathbf{f}}_{t+1}, \cdots, \hat{\mathbf{f}}_{t+N}\}$, which are restored from the reduced trajectory $\{\mathbf{r}_t, \mathbf{r}_{t+1}, \cdots, \mathbf{r}_{t+N}\}$ using a decoder model $\mathcal{D}$. The goal of the model, then, naturally becomes to minimize the error between the approximate solutions and their corresponding reference solutions, which we evaluate with an l^2-norm, i.e., $\sum_{i=1}^{N} ||\hat{\mathbf{f}}_{t+i} - \mathbf{f}_{t+i}||_2$. Within this setup, we detail our autonomous learning methodology and its variants that use either differently constrained interactions between the reduced solver and neural network models, or no interaction. The interaction of differentiable physics within training shares the same spirit as in (Um et al., 2020; Kochkov et al., 2021); thus, our interacting models are along the line of their training methodology. However, our experiments study the autonomy in shaping the latent space, whereas (Um et al., 2020) and (Kochkov et al., 2021) focus on correcting the physical states of the reduced solver and the latent space, respectively, which represent their learning target.

Autonomous interaction (ATO): For a target PDE problem, we integrate the approximate solution trajectory using the interaction of an encoder, a reduced solver, and a decoder, i.e., $\hat{\mathbf{f}}_{t+k} = \mathcal{D}(\mathcal{P}(\mathcal{E}(\hat{\mathbf{f}}_{t+k-1})))$, with the initial state $\hat{\mathbf{f}}_t$ being initialized by the corresponding reference state $\mathbf{f}_t$. Within this setup for solving the target problem, we let both the encoder and decoder models be trainable, namely, $\mathcal{E}(\mathbf{f}|\theta_E)$ and $\mathcal{D}(\mathbf{r}|\theta_D)$, where θ_E and θ_D are the learned weights for the encoder and decoder, respectively.

Here, the encoder does not receive any explicit constraints and has the complete freedom to autonomously explore the reduced space to arrive at a suitable representation for interacting with the reduced solver and the decoder. Therefore, the training process retains complete autonomy in shaping the latent space representation, which encodes important signals from the fine representation of the accurate solution.

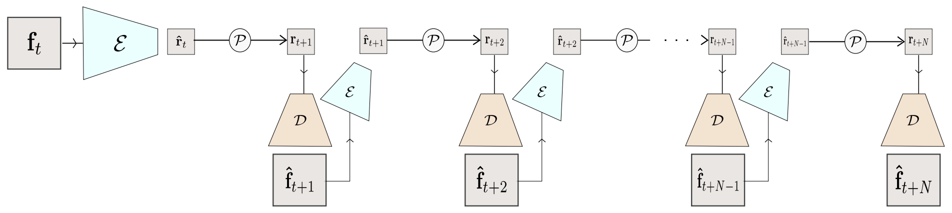


Figure 2: Architecture of our interacting models for n solver steps. The initial state $\mathbf{f}_t$ goes through the encoder model that gives a reduced state $\hat{\mathbf{r}}_t$. This reduced state is given to the reduced solver $\mathcal{P}$ that outputs the next reduced state $\mathbf{r}_{t+1}$ that is then decoded into the approximate solution $\hat{\mathbf{f}}_{t+1}$. This is repeated n times.

Constrained interaction: Contrary to the fully autonomous case, we consider two constrained variants of the training setup. Here, the constraints are inspired by problem-dependent prior knowledge and conventional algorithms for down-sampling solutions. These two variants correspond to algorithms from previous work, which typically rely on coarse physical simulations as approximations of the target phenomena (Bar-Sinai et al., 2019; Um et al., 2020; Kochkov et al., 2021).

In line with the ATO setup, the approximate solution is integrated with the interaction of a trainable encoder, a solver, and a trainable decoder. However, as a first variant, we consider a constraint of the latent space representation with an additional loss term that penalizes deviations from the physical state in the reduced representation. Such terms are problem dependent and can, e.g., provide gradients for divergence-freeness constraints. We denote such variants as constrained latent space (CLS) setups.

As a second variant, we can constrain the latent space encoding itself. We call this variant the constrained encoder (CEN) setup. This constraint is inspired by conventional filtering practices for coarse representations in numerical methods. Here the reduced representation is determined analytically via a down-sampling operator. We use linear interpolation in our experiments, i.e., $\hat{\mathbf{r}} = \mathcal{E}_{\text{lerp}}(\hat{\mathbf{f}})$. In this case, the encoder does not contain any parameters, and hence is not trainable.

No interaction: Lastly, we also consider the non-interacting case (NON). In this setup, no encoder exists, and each state of the approximate solution trajectory is integrated purely using the reduced solver in combination with a decoder, i.e., $\hat{\mathbf{f}}_{t+k} = \mathcal{D}(\mathbf{r}_{t+k}|\theta_D)$ where $\mathbf{r}_{t+k} = \mathcal{P}(\mathbf{r}_{t+k-1})$. Here, for the initial reduced state $\mathbf{r}_t$, we apply a simple down-sampling operator to the initial reference state $\mathbf{f}_t$, i.e., $\mathbf{r}_t = \mathcal{E}_{\text{lerp}}(\mathbf{f}_t)$.

Below we will evaluate different encoder and decoder models trained with these different learning setups. For the interacting setups (i.e., ATO, CLS, and CEN), the models are trained using a differentiable physics framework such that the evaluation of trained models influences the approximated solution trajectory, where the number of recurrent interacting steps varies. The architecture of such interaction is detailed in Fig.2.

3 Experiments

We investigate two different PDE scenarios to evaluate our approach: Burgers equation for non-linear advection-diffusion and Navier-Stokes equations for buoyancy-driven flows.

3.1 Non-linear Advection-Diffusion

We first consider a two-dimensional non-linear advection-diffusion problem, where constant randomized forces are periodically injected into the system such that it leads to a complex evolution of physical state. This problem is modeled as the Burgers' equation as follows:

$$\partial \mathbf{v}/\partial t = -(\mathbf{v} \cdot \nabla)\mathbf{v} - \nu \nabla^2 \mathbf{v} + \mathbf{g} \tag{1}$$

where $\mathbf{v}$ is the velocity, ν is the kinematic viscosity coefficient, and $\mathbf{g}$ denotes the external forces. Similarly to previous work (Bar-Sinai et al., 2019; Um et al., 2020), the external forces, i.e., $\mathbf{g} = [g_x\ g_y]^T$, are evaluated using 20 overlapping sine functions with a random direction, amplitude, and phase shift as follows:

$$g_x(t) = \sum_{i=1}^{20} a_i \cos(\alpha_i) \sin(w_i t - kx + \phi_i) \quad \text{and} \quad g_y(t) = \sum_{i=1}^{20} a_i \sin(\alpha_i) \sin(w_i t - kx + \phi_i). \tag{2}$$

The reference solution trajectories are generated for N = 200 steps from ten different initial conditions with a fixed time step size. Each simulation uses a domain discretized with 64^2 cells and a staggered layout with periodic boundary conditions. A randomized force sequence is generated for 50 steps. This sequence is applied as $\mathbf{g}$ for all simulations periodically over the whole 200 steps. This periodic forcing prevents overly chaotic temporal evolutions, and ensures that the simulations stay within a controllable regime.

We consider a four times coarser discretization for the reduced representation, and the velocity field as our restoration target. Since the external forces are an important influence in this scenario, we condition the encoder on $\mathbf{g}$, which is provided as an additional input to this model. This allows the networks to take the forcing into account when inferring latent space representations. However, the reduced solver takes the analytically down-sampled force field as input. The loss for n interacting steps is evaluated with the velocity as:

$$\mathcal{L}_{\text{AD}} = \sum_{i=1}^{n} \|\hat{\mathbf{v}}_{t+i} - \mathbf{v}_{t+i}\|_2 \tag{3}$$

where $\hat{\mathbf{v}}$ denotes the velocity field restored using the trained model.

3.2 Buoyancy-driven Flows

In the second scenario we consider a complex constrained PDE problem, namely the Navier-Stokes equations, in the two-dimensional space as follows:

$$\partial \mathbf{v}/\partial t = -(\mathbf{v}\cdot\nabla)\mathbf{v} - \nabla p/\rho + \nu\nabla^2\mathbf{v} + \mathbf{g} \quad \text{subject to} \quad \nabla\cdot\mathbf{v} = 0 \tag{4}$$

$$\partial d/\partial t = -(\mathbf{v}\cdot\nabla)d \tag{5}$$

where ρ is the density, p is the pressure, and d is a marker field whose values lie in $[0, 1]$ and which is advected with the flow. We use the Boussinesq approximation, which means that buoyancy forces are determined by the marker field as $\mathbf{g} = [0\ \eta d]^T$, where η denotes the buoyancy factor. This coupled system results in a chaotic and complex evolution of the solution trajectories and thus constitutes a challenging learning problem.

For the reference solutions, we use a numerical fluid solver that adopts the operator splitting scheme, Chorin projection for implicit pressure solve (Chorin, 1967), and the less dissipative advection algorithm (Selle et al., 2008), enabling the chaotic evolution of marker volumes. Moreover, we do not solve for the viscosity effect relying on the numerical viscosity inherent in discretization. The domain is discretized with 64 cells in each dimension, where the velocity field follows the staggered layout and is set with the closed boundary condition at the domain wall. Randomizing the position and radius of an initial round marker volume, we generate 30 reference solution trajectories, each of which consists of N = 400 steps integrated with a fixed time step size.

We consider a four times coarser discretization for the reduced representation and adopt the more dissipative advection algorithm (Stam, 1999) for the reduced solver that is interacting with the trained models. Since the solver mainly aims to resolve the evolution of velocity, we focus on restoring the velocity field; thus, the decoder outputs only the velocity field. The marker field, on the other hand, is passively advected using the restored velocity field. The encoder transforms both the marker and velocity fields, and then their reduced representations are used for the reduced solver.

For n interacting steps, we can define the loss for this example as follows:

$$\mathcal{L}_{\text{BF}} = \sum_{i=1}^{n} \|\hat{\mathbf{v}}_{t+i} - \mathbf{v}_{t+i}\|_2 + \|\hat{d}_{t+i} - d_{t+i}\|_2 \tag{6}$$

where $\hat{d}$ is the advected marker field using $\hat{\mathbf{v}}$.

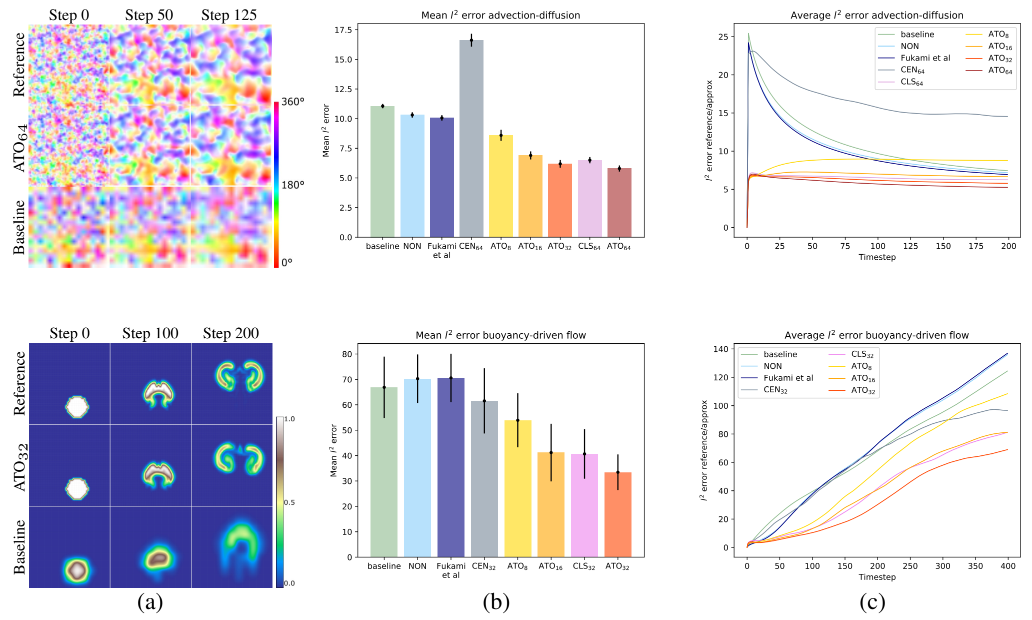


Figure 3: Evaluations of non-linear advection-diffusion (top) and buoyancy-driven flows (bottom). (a) Visual overviews of different solution trajectories. (b) Velocity errors of restored solution trajectories with respect to the reference solution, averaged over all test simulations of N time steps, for differently trained models. The error bars indicate the variance over the test runs. (c) Evolution of the errors over time, averaged over test simulations.

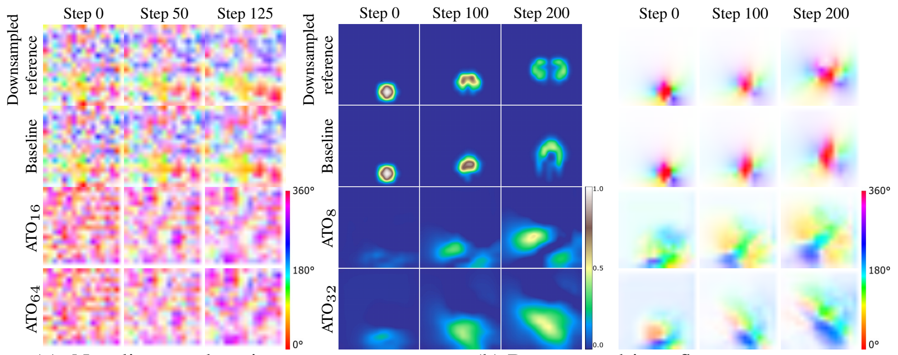
3.3 Network Architecture and Training Procedure

The encoder $\mathcal{E}(\cdot|\theta_E)$ and decoder $\mathcal{D}(\cdot|\theta_D)$ are modeled as convolutional neural networks. The encoder model contains two max-pooling layers that reduce the input dimension into a four times lower dimension for the output. Correspondingly, the decoder uses two up-sampling layers that restore the reduced dimension to the original input dimension. For the convolutional layers of the models, we adopt circular padding for the periodic boundary condition problem and zero padding for the closed boundary condition problem. We employ a U-net structure such that the information of the intermediate features from the encoder can be propagated through the decoder (Ronneberger et al., 2015). Our networks contain 11 layers including five Leaky ReLU activations for the encoder, and 13 layers including five Leaky ReLU activations and two concatenations for the decoder. Details of the architecture are given in the appendix.

When training the interacting setups (i.e., ATO, CLS and CEN) for a given number of n integrated steps, at each training iteration, the encoder, differentiable physics solver, and decoder are recurrently evaluated n times in the forward pass. Correspondingly, the weights of the neural network models are updated by back-propagating through the n steps to accumulate the gradient of the weights. In the following, we will denote the number of recurrently integrated steps as a subscript, e.g., ATO$_n$. For these setups, different numbers of integrated steps are investigated. At each training iteration, for a given batch size, we randomly sample the initial states from the reference solution trajectories and integrate the approximate solution trajectories for the n steps. All our training runs use an Adam optimizer (Kingma & Ba, 2014) and an adaptive learning rate starting in $\{10^{-4}, 10^{-3}\}$.

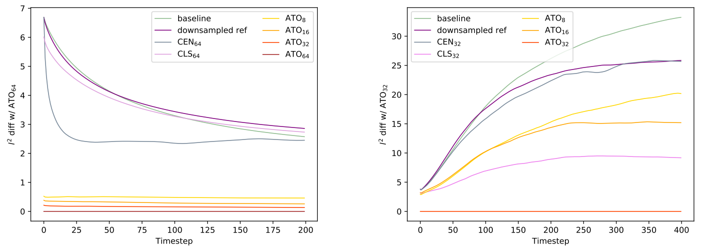
4 Results and Discussion

We evaluate trained models based on relative improvements over a reduced *baseline* simulation. The baseline solutions are the trajectories integrated purely by the reduced solver, without interactions with an encoder nor decoder model. Errors are computed with respect to the reference solutions, and hence an improvement of 100% would mean that the restored solutions are identical to the reference.



(a) Non-linear advection-diffusion (b) Buoyancy-driven flow

Figure 4: Visual comparisons of reduced states. The bottom two rows for the ATO models represent physical dynamics that noticeably deviate from the original time evolution.



(a) Non-linear advection-diffusion (b) Buoyancy-driven flow

Figure 5: l^2 difference of the reduced states with respect to the states of the best performing ATO model. Larger distances consistently coincide with reduced model performance.

We present our evaluation with an l^2 error metric; additional metrics such as SSIM (Wang et al., 2004) and LSiM (Kohl et al., 2020) are also provided in the appendix.

Non-Linear Advection-Diffusion: This example considers a scenario where an external factor, i.e., randomized forcing, influences the evolution of solution trajectories. The reference solution of this problem typically evolves from an initial state with high-frequency details towards a smoother state over time due to its diffusive nature. Consequently, the evolution of the reduced solution generally shows larger error at the beginning and becomes more similar to the reference in later steps. For the constrained latent space (CLS) setup, we adapt the training loss by adding the difference between the encoded state and down-sampled reference state as follows:

$$\mathcal{L}_{\text{AD}} = \sum_{i=1}^{n} \|\hat{\mathbf{v}}_{t+i} - \mathbf{v}_{t+i}\|_2 + \|\mathcal{E}(\hat{\mathbf{v}}_{t+i-1}) - \mathcal{E}_{\text{lerp}}(\mathbf{v}_{t+i-1})\|_2. \tag{7}$$

This encourages the encoded states to stay close to the reference states produced by a conventional method for reduction. We evaluate the models with five trajectories from test simulations consisting of 200 steps. In this test setup, we observe that the error of the reduced solution quickly drops to half of its initial error after approximately 50 steps.

This example demonstrates that the complete autonomy significantly improves the training quality. As shown at the top row of Fig. 3, the ATO$_{64}$ model clearly outperforms the others. This particularly stands out in the early steps, where most of high-frequency details would be lost due to the smaller degrees of freedom of the reduced representation. Comparing the ATO models with different numbers of integrated steps, we can observe greater improvement when more steps are integrated. The ATO$_{64}$ model successfully restores the high-frequency details and shows 48% of improvement on average for the five trajectories of the test solutions. Contrarily, despite the same number of integrated steps, the model trained with a constrained encoding, CEN$_{64}$, significantly deteriorates the

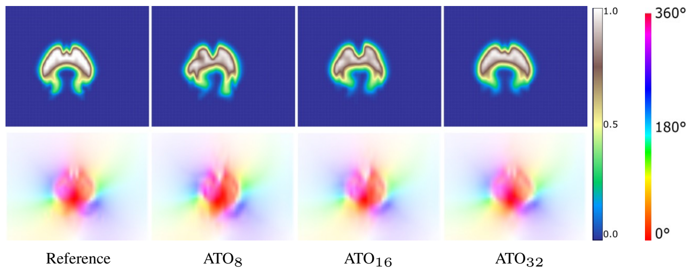


Figure 6: Comparison among the ATO models for a buoyancy-driven flow example. Increasing numbers of steps of the recurrent training yields better and better reconstructions of the reference on the left.

solutions. On the other hand, the NON model fails to yield any improvements, presenting a similar error as the baseline. Moreover, we compare with a super-resolution architecture from previous work (Fukami et al., 2019) as a *state-of-the-art* variant of the NON model. Despite the refined architecture, this no-interaction setup consistently gives a poor performance. Details of the comparison are given in the appendix. For the CLS$_{64}$ case, the constraint noticeably diminishes the improvement of our ATO model, with 41% of improvement on average.

The images of Fig. 4-(a) show visual examples of four reduced representations for a selected test trajectory for the non-linear advection-diffusion case. When the encoder is trained with complete autonomy, we observe that, surprisingly, the reduced representations encoded by the ATO models differ considerably from the conventionally down-sampled reference states (Fig. 4-(a) first row) and the states integrated purely by the reduced solver, i.e., baseline (Fig. 4-(a) second row). Moreover, the graphs from Fig. 5 show the quantified differences between the reduced states produced by the different trained models and the ones from the best performing model, i.e., ATO$_{64}$. We observe over the course of repeated runs that the different ATO models produce reduced states that are consistently closer to the best performing model's when they are trained with more integrated steps. This indicates that the complete freedom to shape the physical states in the given reduced degrees of freedom allows the encoder to arrive at an “optimal” representation. For a given target PDE problem and training setup, the different ATO versions reliably converge towards this representation, which seems to best encode the information of the reference solutions.

Buoyancy-Driven Flow: This example targets a challenging learning task where complex swirling motions are produced over time due to the coupled system of the velocity and marker fields. In this scenario, the reduced PDE solver is based on a more dissipative advection algorithm. Thus, unlike the advection-diffusion scenario, which discretized the same PDE formulation on the reduced and fine spaces, the buoyancy-driven flow scenario contains additional challenges in the learning task. For the constrained latent space (CLS) model, we include loss terms for the velocity divergence as follows:

$$\mathcal{L}_{\mathrm{BF}} = \sum_{i=1}^{n} \|\hat{\mathbf{v}}_{t+i} - \mathbf{v}_{t+i}\|_2 + \|\hat{d}_{t+i} - d_{t+i}\|_2 + \|\nabla \cdot \mathcal{E}(\hat{\mathbf{v}}_{t+i-1})\|_2. \tag{8}$$

In this way, the encoded velocity field is guided to satisfy the divergence-free condition, which is important for the given flow problem. The trained models are evaluated with seven test trajectories consisting of 400 steps.

In this buoyancy-driven flow case, the trained models likewise benefit from larger numbers of time steps at training time as shown in Fig. 6. The model trained with the autonomous interaction setup significantly outperforms the other variants: the average l^2 error over the seven test trajectories for the ATO$_{32}$ model is 33.44, which is equivalent to a 50% improvement. The CEN$_{32}$ model does not achieve a comparable performance giving a higher error of 49.73. With the CLS$_{32}$ model, we report that the additional latent space loss once again reduces the improvement of our ATO model; see ATO$_{32}$ and CLS$_{32}$ of Fig. 3. The model trained with the non-interacting setup (NON) and a regular architecture leads to large errors of 70.28, while the NON super-resolution architecture leads to similarly large errors of 70.59. These error measurements can be confirmed visually, as shown in the bottom row of Fig. 3. The ATO$_{32}$ model is able to produce finely resolved details that stay close to the reference solution.

In this scenario, the reduced PDE is driven by the marker field in addition to the velocity field. A set of reduced representations for both fields is shown in Fig. 4-(b). Again, we observe that both reduced fields significantly differ from the down-sampled reference fields (Fig. 4-(b) first row) and the reduced solution fields (Fig. 4-(b) second row). Additionally, the graphs in Fig. 5 present a comparison among the reduced velocity fields of different setups. As in the non-linear advection-diffusion case, the reduced states produced with the ATO models get closer to the states of the best performing model as they approach a better performance for the learning objective.

For the buoyancy-driven flow, the latent spaces learned by the ATO models allow for an intuitive interpretation by human observers. The learned dynamics are best observed in the supplemental video, but the still images in Fig. 4-(b) already give an impression: it becomes apparent that the neural network rearranges space and translates the upwards motion caused by buoyancy in physical space into a diagonal motion in latent space. In addition, the localized markers in physical space occupy larger regions in the latent space. Most likely, this encodes more information about the configurations in the target space with its higher resolution.

Runtime Performance: While gains in terms of the runtime of the learned solver in reduced space represent the central goal of our approach, our current version focuses on a proof of concept implementation. Despite this, the buoyancy flow solver yields improvements in runtime, where the reduced simulation takes 0.02s on average per simulation step, and the neural networks cause an additional 0.02s of evaluation time. Instead, the reference solver requires 0.06s per step. For the advection-diffusion scenario, there is no improvement in runtime performance, with 0.01s for the reduced simulation, 0.02s for the neural networks, and 0.02s on average for the reference. This is directly linked to the explicit nature of the numerical solver step, which has an inherently low runtime.

However, these numbers all employ efficient CPU-based PDE solvers and include GPU-transfer overhead for the neural network evaluations. Hence, due to the super-linear scaling of typical PDE solvers, we see the potential for large performance gains from our approach.

5 Conclusions

The work presented in this paper has demonstrated that neural networks can learn new physical evolutions in the spaces of coupled PDE solvers, resulting in a shaping of latent spaces that yields benefits for learning objectives. Specifically, giving complete freedom to the encoder and decoder models to influence the content of a PDE-based latent space has shown to be highly effective for restoring complex reference solutions. The models trained with this autonomy have shown their superiority on the tasks we experimented with: reducing a physical state, integrating this reduced state forward in time, and eventually transforming the integrated state into an accurate and fine solution. We have demonstrated that the reduced state learned with this methodology forms a physical solution that is significantly different from conventional reduced states.

Our results indicate that the proposed training methodology and correspondingly trained models have the potential to lead to novel ways of solving complex PDE problems. To the best of our knowledge, we have presented the first study about shaping the latent spaces of neural networks for interactions with PDE solvers. We believe that our study sheds light on how the unconstrained, learning-based reshaping of physical states can be used to improve PDE solvers.

6 Future Work

Our work poses a variety of interesting avenues for future work. Despite the great reduction in terms of the solver's degrees of freedom as shown in the buoyancy-driven flow example, the general question of which reduced PDE solver to employ in the latent space in order to achieve the best performance for a given problem remains open. Moreover, we plan to investigate the evolution of the reduced space representations of our models in more detail. For example, analyzing how the dynamics of the reduced states relate to known PDE formulations will be highly valuable as future work.

Reproducibility Statement

The source code of our experiments and simulations for generating all data sets will be published upon acceptance.

References

Brandon Amos and J Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, 2017.

Simon Baker, Stefan Roth, Daniel Scharstein, Michael J. Black, J.P. Lewis, and Richard Szeliski. A database and evaluation methodology for optical flow. In *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007. doi: 10.1109/ICCV.2007.4408903.

Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model reduction and neural networks for parametric PDEs. *The SMAI journal of computational mathematics*, 7:121–157, 2021. doi: 10.5802/smai-jcm.74.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, November 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1906995116.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Alexandre Joel Chorin. The numerical solution of the navier-stokes equations for an incompressible fluid. *Bulletin of the American Mathematical Society*, 73(6):928–931, 1967.

James P Crutchfield and Bruce S McNamara. Equations of motion from a data series. *Complex systems*, 1(417-452):121, 1987.

Cathal Cummins, Madeleine Seale, Alice Macente, Daniele Certini, Enrico Mastropaolo, Ignazio Maria Viola, and Naomi Nakayama. A separated vortex ring underlies the flight of the dandelion. *Nature*, 562(7727):414, 2018.

Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. In *Advances in neural information processing systems*, 2018.

Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

Kai Fukami, Takaaki Murata, Kai Zhang, and Koji Fukagata. Sparse identification of nonlinear dynamics with low-dimensionalized flow representations. *Journal of Fluid Mechanics*, 926, November 2021. ISSN 0022-1120, 1469-7645. doi: 10.1017/jfm.2021.697.

Philipp Holl, Vladlen Koltun, and Nils Thuerey. Learning to control pdes with differentiable physics. *International Conference on Learning Representations (ICLR)*, 2020.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *International Conference on Learning Representations (ICLR)*, 2020.

Mike Innes, Alan Edelman, Keno Fischer, Chris Rackauckas, Elliot Saba, Viral B Shah, and Will Tebbutt. A differentiable programming system to bridge machine learning and scientific computing. *arXiv 1907.07587*, 2019.

Barbara M Johnston, Peter R Johnston, Stuart Corney, and David Kilpatrick. Non-newtonian blood flow in human right coronary arteries: steady state simulations. *Journal of biomechanics*, 37(5): 709–720, 2004.

Ioannis G Kevrekidis, C William Gear, James M Hyman, Panagiotis G Kevrekidid, Olof Runborg, Constantinos Theodoropoulos, et al. Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis. *Communications in Mathematical Sciences*, 1(4):715–762, 2003.

Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum*, 2019. ISSN 1467-8659. doi: 10.1111/cgf.13619.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]*, December 2014.

Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), May 2021. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2101784118.

Georg Kohl, Kiwon Um, and Nils Thuerey. Learning similarity metrics for numerical simulations. *Proceedings of the International Conference on Machine Learning*, 1, 2020.

Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. In *Advances in Neural Information Processing Systems*, pp. 771–780, 2019.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, November 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-07210-0.

Romit Maulik, Bethany Lusch, and Prasanna Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3):037106, March 2021. ISSN 1070-6631. doi: 10.1063/5.0039986.

Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. In *Advances in Neural Information Processing Systems*, 2018.

David A Randall, Richard A Wood, Sandrine Bony, Robert Colman, Thierry Fichefet, John Fyfe, Vladimir Kattsov, Andrew Pitman, Jagadish Shukla, Jayaraman Srinivasan, et al. Climate models and their evaluation. In *Report of the IPCC (FAR)*, pp. 589–662. Cambridge University Press, 2007.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241. Springer, 2015.

Connor Schenck and Dieter Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*, pp. 317–335, 2018.

Samuel S Schoenholz and Ekin D Cubuk. Jax, md: End-to-end differentiable, hardware accelerated, molecular dynamics in pure python. *arXiv:1912.04232*, 2019.

Andrew Selle, Ronald Fedkiw, ByungMoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable maccormack method. *Journal of Scientific Computing*, 35(2-3):350–371, June 2008. ISSN 0885-7474, 1573-7691. doi: 10.1007/s10915-007-9166-4.

Justin Sirignano, Jonathan F. MacArt, and Jonathan B. Freund. DPM: A deep learning PDE augmentation method with application to large-eddy simulation. *Journal of Computational Physics*, 423:109811, December 2020. ISSN 0021-9991. doi: 10.1016/j.jcp.2020.109811.

Jos Stam. Stable fluids. In *SIGGRAPH '99*, pp. 121–128. ACM, 1999. ISBN 0-201-48560-5. doi: 10.1145/311535.311548.

Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating eulerian fluid simulation with convolutional networks. In *Proceedings of Machine Learning Research*, pp. 3424–3433, 2017.

Marc Toussaint, Kelsey Allen, Kevin Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Robotics: Science and Systems*, 2018.

Kiwon Um, Xiangyu Hu, and Nils Thuerey. Liquid splash modeling with neural networks. *Computer Graphics Forum*, 37(8):171–182, December 2018. ISSN 1467-8659. doi: 10.1111/cgf.13522.

Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems*, 33, 2020.

Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pp. 1457–1466, New York, NY, USA, August 2020a. Association for Computing Machinery. ISBN 978-1-4503-7998-4. doi: 10.1145/3394486.3403198.

Wujie Wang, Simon Axelrod, and Rafael Gómez-Bombarelli. Differentiable molecular simulations for control and learning. *arXiv:2003.00868*, 2020b.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

S. Wiewel, B. Kim, V. C. Azevedo, B. Solenthaler, and N. Thuerey. Latent space subdivision: Stable and controllable time predictions for fluid flow. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '20, pp. 1–11, Goslar, DEU, October 2020. Eurographics Association. doi: 10.1111/cgf.14097.

Jiawei Zhuang, Dmitrii Kochkov, Yohai Bar-Sinai, Michael P. Brenner, and Stephan Hoyer. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Physical Review Fluids*, 6(6):064605, June 2021. doi: 10.1103/PhysRevFluids.6.064605.

A Network Architecture and Training Procedure

We provide details of the U-net architecture used for our ATO models, and more specifically the architecture used for our buoyancy-driven flow scenario in Fig. 7. In this scenario, the input layer contains three channels: one for the scalar marker field and two for the velocity vector field. The encoder reduces their degrees of freedom and outputs three channels, which the physics solver integrates forward in time. Then, the updated state goes into the decoder, being concatenated with the previous state in line with the skip connection of the U-net structure. Note that, since we target the velocity field, the decoder takes the velocity fields as its input and outputs two channels for the restored velocity field. Fig. 8 shows the selected examples of our training datasets.

In the non-linear advection-diffusion scenario, on the other hand, the encoder takes four channels of the velocity vector field and external force field while generating two channels of the reduced velocity field. Since the forcing in this scenario is an external parameter of the physics, the solver takes analytically downsampled force fields as external forcing in the reduced space. The decoder, then, again generates the velocity field with two channels.

As in (Um et al., 2020), we find that training our ATO models is quickly stabilized when adopting a "warm" start, which initializes the model from another model that is trained with less number of integrated steps. Our models are trained with this warm start, e.g., when training ATO$_{32}$, the weights of the model are initialized using the ATO$_{16}$ model's.

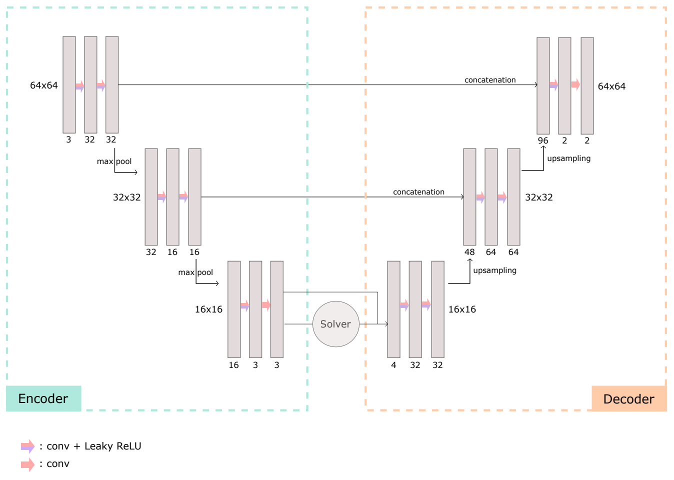


Figure 7: A detailed architecture of the encoder and decoder models interacting with the differentiable physics solver for the buoyancy-driven flow scenario.

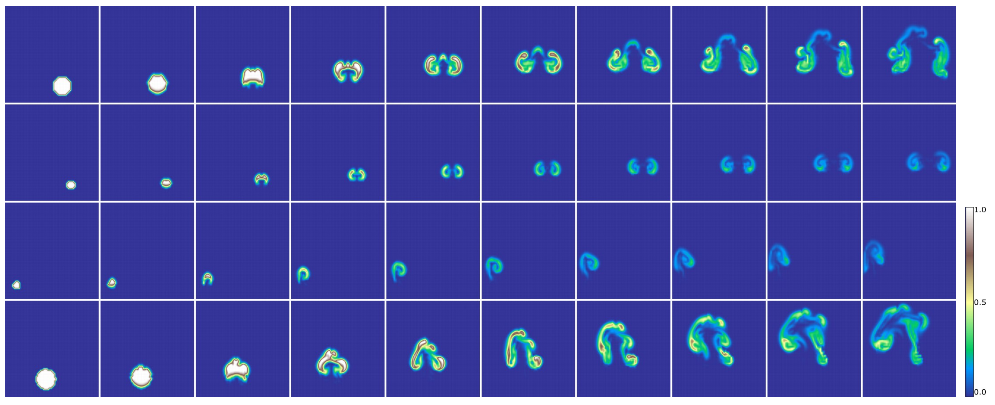


Figure 8: Training dataset examples of the buoyancy-driven flow scenario. The selected marker fields of four reference trajectories are shown at each row.

B Ablation study

We conduct an ablation study to show the importance of the U-Net architecture adopted in our ATO model. Fig. 9 shows the l^2 error analysis of our ATO models as well as a version of the ATO_{64} model without skip connections for the advection-diffusion case. As shown in the graphs, skip connections yield significant improvements with our ATO models. Here, the mean error for the ATO model with 64 integrated steps is 16.2 without skip-connections, while it is 5.8 with skip connections (compared to 11.1 for the baseline). Hence, the ATO model performs significantly better when it can make use of the skip connections while the latent space remains autonomous. We note that the reduced solver interacts only with the latent-space state.

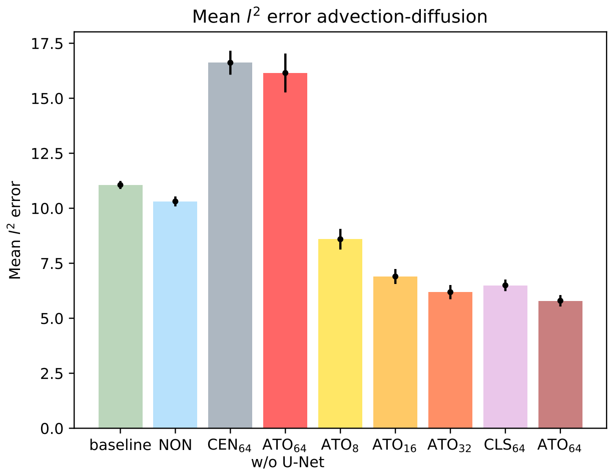
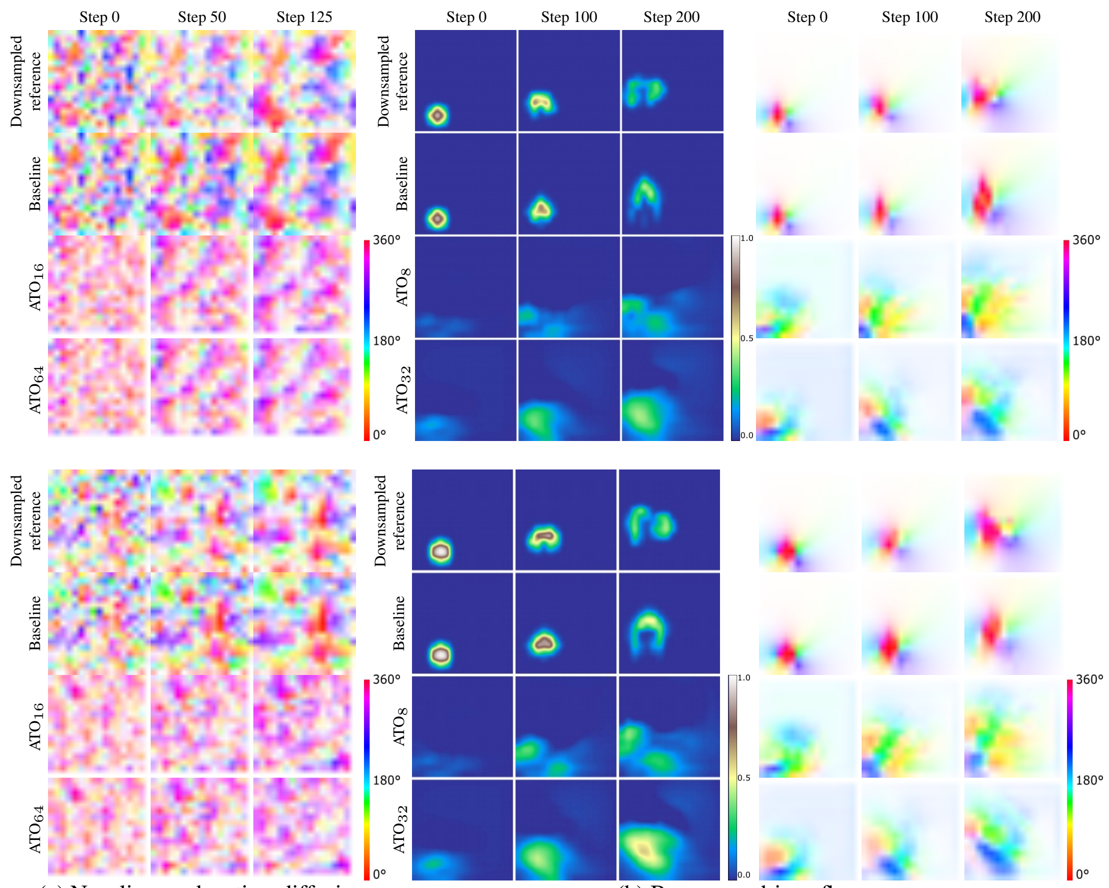


Figure 9: Ablation study of the influence of skip connections in the ATO model.

C Additional results

In this section, we provide additional information about our comparison with the work of (Fukami et al., 2019). We implemented the "DSC/MS" model for super-resolution that is described in their work. In comparison to our regular NON model, it contains two advanced components: skip-connections and multi-scale layers with convolutional filters of different sizes. For the advection-diffusion example, we experimented with two different models: a large model of 120k weights, which is close to their original model, and a smaller one with 17K weights, which contains a similar number of weights as ours. Both were then trained with the data sets and tasks outlined in our work. When evaluated with our test sets, these new models do not show any significant improvement compared to the regular NON model; the l^2 errors of the large and small models are 10.6 and 10.1, respectively, while the NON model shows the l^2 error of 10.3 and the ATO model shows 5.80. Additionally, we tested the large model of 120k weights with the buoyancy-flow example; its l^2 error is 70.59, whereas the NON and ATO models show the error of 70.28 and 33.44, respectively. Thus, our ATO model outperforms the others in every case. We believe that, when using the super-resolution NON model (Fukami et al., 2019), the complete lack of temporal integration in the training process leads to the observed, poor performance despite its advanced architecture.

We also provide additional results of our experiments. We note that the scalar marker fields are visualized using the "terrain" color scheme and the velocity fields are visualized as in (Baker et al., 2007). The color bars for these two visualizations can be seen next to each figure. For the velocity fields, the color bar only shows the angle visualization, while the magnitude of the vectors is represented using the intensity. Fig. 10 shows two additional visual comparisons of four reduced representations. We, again, observe that the reduced representations encoded by our ATO models differ considerably from the conventionally downsampled reference states and the baseline. Moreover, as shown in Fig. 11, our best performing models in both scenarios successfully restore the simulation results close to the references.



(a) Non-linear advection-diffusion (b) Buoyancy-driven flow

Figure 10: Two additional examples of reduced space visualizations.

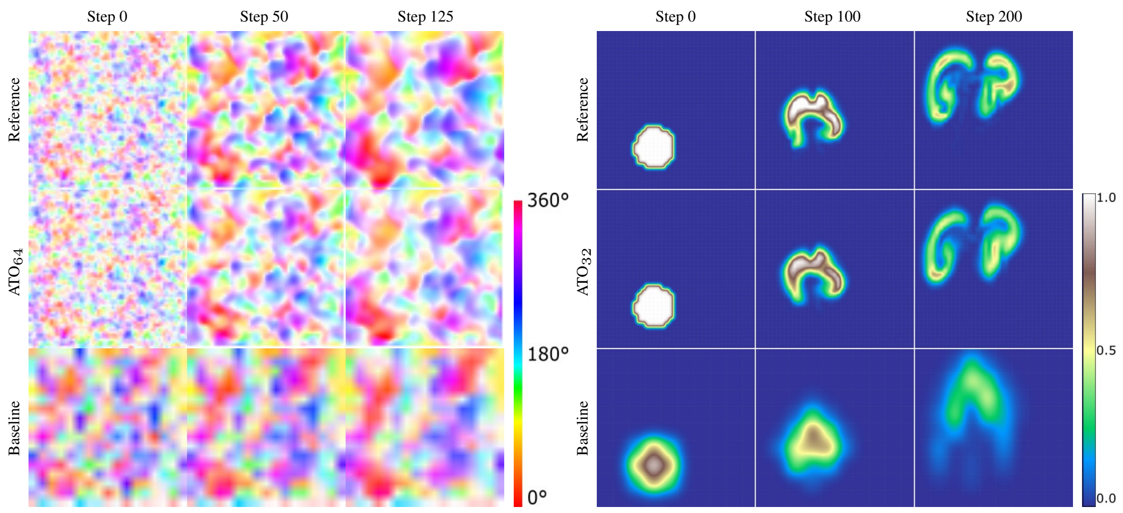


Figure 11: Comparisons of temporal evolutions in a non-linear advection-diffusion (left) and a buoyancy-driven flow example (right).

Fig. 12 shows additional test examples of the buoyancy-driven flow scenario, where the initial states are selected at the middle of our test dataset's trajectories. As shown, our model successfully restores the results close to the reference. Fig. 13 shows the l^2 error analysis of these additional examples, and our ATO$_{32}$ model presents the lowest error in the graphs.

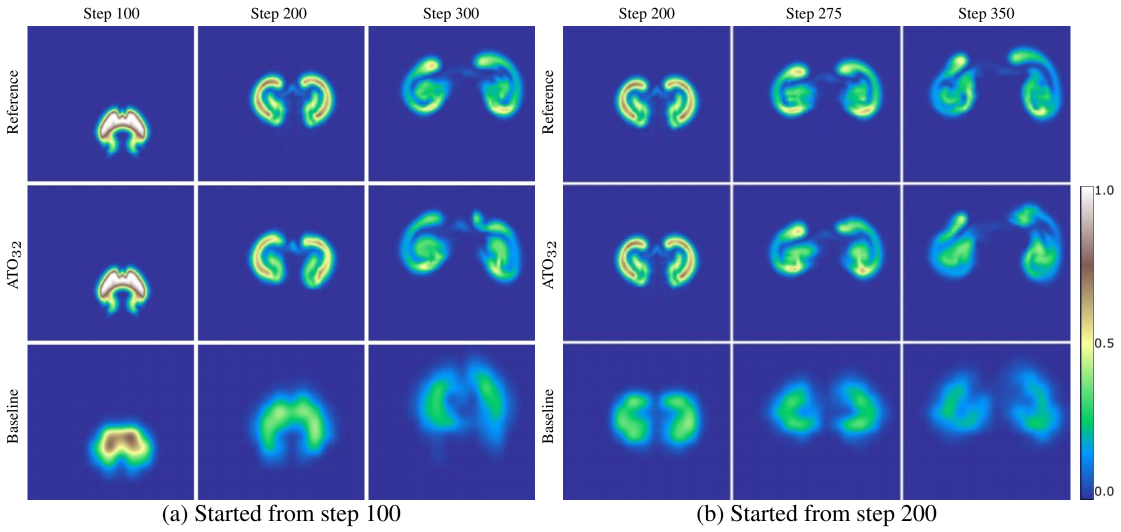


Figure 12: Selected simulation steps of two additional test examples started from different initial states.

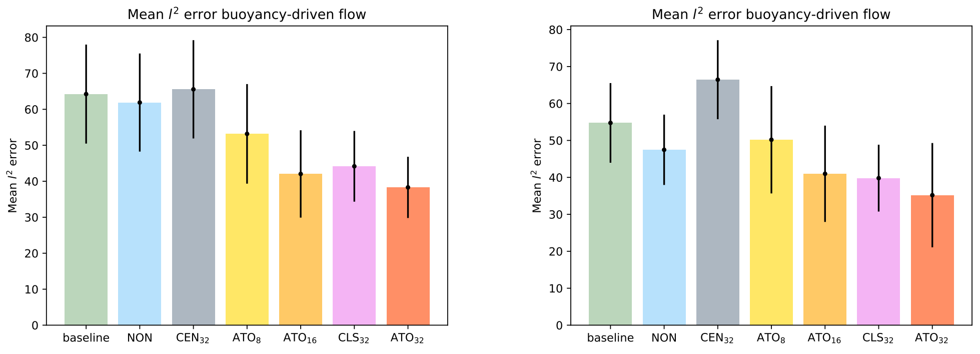
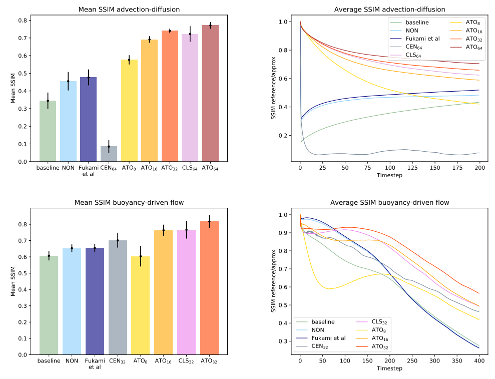


Figure 13: Mean l^2 errors for the buoyancy-driven flow case, starting from step 100 (left) and step 200 (right).

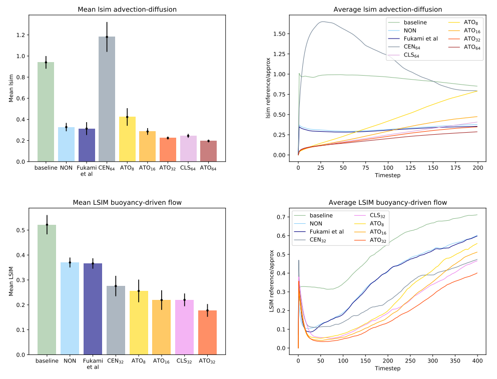
As an additional study, we extrapolate our results beyond the time-range that is observed during training, for the buoyancy-driven flow case. We use the 200th time step of our usual test simulations as an initial state to apply our models to. We compare the results of this experiment for 400 consecutive steps for ATO$_{32}$, the baseline and the NON model. The baseline gives an average error of 63.1, the NON model an error of 60.2 and our ATO model of 53.7. Therefore, the fully autonomous model still performs better than the baseline and NON models outside of the observed time range.

D Additional metrics

In addition to the l^2 error shown in Fig. 3, Fig. 14 shows the comparisons of different models in two other metrics: structural similarity index (SSIM) (Wang et al., 2004) and learned simulation metric (LSiM) (Kohl et al., 2020). SSIM is a perceptual metric considering structural information while LSiM is a learning-based metric that intends for comparing numerical simulations. In these different metrics, our experiments consistently show that training with the autonomy leads to the best performing model while the constrained training deteriorates its performance. We also note that a lack of interaction in terms of integrated simulation steps yields a failure at learning the task as shown for ATO_2 and ATO_4 in Fig. 15.



(a) SSIM



(b) LSiM

Figure 14: Evaluations of non-linear advection-diffusion (top) and buoyancy-driven flow (bottom) in two additional metrics: SSIM and LSiM.

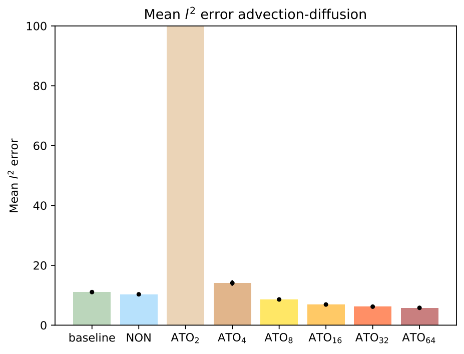


Figure 15: Mean l^2 errors for the non-linear advection-diffusion scenario with different numbers of integrated steps.