LEARNING DYNAMICAL CHARACTERISTICS WITH NEURAL OPERATORS FOR DATA ASSIMILATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Data assimilation refers to a group of algorithms that combines numerical models with observations to obtain an optimal estimation of the system's states. In areas like earth science, numerical models are usually formulated by differential equations, also known as the prior dynamics. It is a great challenge for neural networks to properly exploit the dynamical characteristics for data assimilation, because first, it is difficult to represent complicated dynamical characteristics in neural networks, and second, the dynamics are likely to be biased. The state-of-the-art neural networks borrow from the traditional method to introduce dynamical characteristics by optimizing the 4D-Var objective function in which the dynamics are inherently quantified, but the iterative optimization process leads to high computational cost. In this paper, we develop a novel deep learning framework with neural operators for data assimilation. The key novelty of our proposed approach is that we design a so-called flow operator through self-supervised learning to explicitly learn dynamical characteristics for reconstructed states. Numerical experiments on the Lorenz-63 and Lorenz-96 systems, which are the standard benchmarks for data assimilation performance evaluation, show that the proposed method is at least three times faster than state-of-the-art neural networks, and reduces the dynamic loss by two orders of magnitude. It is also demonstrated that our method is well-adapted to biases in the prior dynamics.

1 INTRODUCTION

Data assimilation can be essentially defined as a statistical technique to combine the prior dynamics with a sequence of noisy and irregularly-sampled observations, which plays an important role in utilizing observational data, especially for numerical weather forecasting systems. For instance, the European Centre for Medium-Range Weather Forecasts (ECMWF) employs the variational assimilation algorithms in its operational systems to take full advantage of both in situ and satellite-derived data, as well as state-of-the-art numerical models (Rabier et al., 2000).

Classical Assimilation Methods Classical data assimilation methods range from early empirical analysis (Bergthorsson et al., 1955) and optimal interpolation (Gandin, 1963) to later variational-based assimilation algorithms (Sasaki, 1970) and the filtering methods based on statistical estimation theory, such as Kalman filter (Welch et al., 1995), ensemble Kalman filter (Evensen, 2003), and particle filter (Carpenter et al., 1999). Although classical methods have been essential tools for improving the capability of major numerical weather prediction centers worldwide (Kalnay, 2003), they generally do not account for the propagation of observation information along time and therefore are unable to utilize observations at different times effectively (Song et al., 2017).

4D-Var Assimilation Method Four-dimensional variational assimilation algorithm is a cutting-edge method for data assimilation. Unlike previous assimilation algorithms, it explicitly takes into consideration the revolution of observation information along time, in other words, the time-dependent dynamical characteristics (Song et al., 2017). Experimental results prove the advantage of 4D-Var assimilation methods in utilizing observational information (Lorenc & Rawlins, 2005). However, due to the complicated modeling and solving process of the 4DVar algorithm, it is considered computationally expensive, especially for high resolution cases. (Fisher, 1998).

Machine Learning for data assimilation Numerous studies developing machine learning for data assimilation have built their theories on classical assimilation methods to reduce computational

cost, especially those prior to 2020 (Hsieh & Tang, 1998; Vijaykumar et al., 2002; de Campos Velho et al., 2002; Härter & de Campos Velho, 2008; Cintra & de Campos Velho, 2018; Mack et al., 2020). As mentioned above, these works are subject to the limitations of the classical methods that do not consider time-dependent dynamical characteristics. Since 2020, the importance of 4D-Var assimilation methods is gradually recognized, and much more novel and bolder research has been done in the area of machine learning for 4D-Var assimilation. These works can be generally divided into three categories: learning observation operator, learning model biases, and learning the optimization algorithm.

Learning Observation Operator The aim of learning observation operator is to use neural network to construct a map from observations to reconstructed states. A representative work is Frerix et al. (2021), in which the observational data is exploited by the neural operator to provide better initial states for the optimization of 4D-Var objective function. This work has at least two flaws. The first is that the neural network is used only to exploit observational information, while the integration of observation and dynamical characteristics is still implemented by 4D-Var, leading to a limited improvement in efficiency. The second is that the strict constraints of time-dependent dynamical characteristics are the prerequisites for this work, which prevent it from generalizing to the cases in which the prior dynamics are biased.

Learning Model Biases As mentioned above, the prior dynamics can be biased, and another group of works uses neural network to learn the biases of prior dynamic models for data assimilation Brajard et al. (2020); Arcucci et al. (2021); Farchi et al. (2021); Bonavita & Laloyaux (2020). These works generally achieve a balance between accuracy and efficiency, but the improvement in computing efficiency is insignificant, because the assimilation of dynamical characteristics and observations is still done through iteratively optimizing 4D-Var objective function.

Learning Optimization Algorithm Since the main computational overhead of the 4D-Var algorithm lies in the process of iteratively optimizing 4D-Var objective function, some studies have focused on replacing traditional optimization algorithms with neural networks. Among them, the representative one is Fablet et al. (2021b;a). This work simultaneously learns both the dynamical characteristics of the model and the optimization algorithm of the 4D-Var objective function through neural networks. It not only takes into consideration the weak constraints of time-dependent dynamical characteristics, but also successfully reduces the number of iterations compared with traditional 4D-Var algorithms, subsequently improving computational efficiency. We consider this work to be the state-of-the-art method for comparison. Despite the achievements of the work, there is still room for improvement in terms of both accuracy and efficiency, and moreover, quantitative exploration on how the prior bias affects the performance of neural network frameworks remains to be done.

Our Contributions We develop a new approach of combing data-driven strategies with model-driven methods for time-dependent data assimilation. This framework consists of three operators implemented by neural networks, namely, observation operator, perturbator, and flow operator. Among them, the cooperation of the perturbator and flow operator decouples the two goals of learning observations and learning dynamical characteristics, and they succeed in continuously refining reconstructed states.

- To our knowledge, this is the first work that directly uses neural network to blend the dynamical prior with a sequence of observations without explicit formulations of 4D-Var objective function for assimilation.
- We test the proposed framework on the Lorenz-63 and Lorenz-96 systems. Experimental results support the effectiveness of our framework. The proposed architecture is at least three times faster than the state-of-the-art neural network. It reduces the dynamic loss by two orders of magnitude and is comparable to the state-of-the-art neural networks in terms of reconstruction error.
- We design experiments in which the prior dynamics are different from ground truth to simulate real-world scenarios. Experimental results prove that our method is well-adapted to deviations of the prior dynamic model.

2 Problem Statement

We refer to Fablet et al. (2021b) to introduce the strict definition and assumptions under considered problem settings. Denote x for the time-dependent state in some space $\mathcal{X}$ satisfying $x(t) \in \mathcal{X}$, $\mathcal{M}_{gt}$

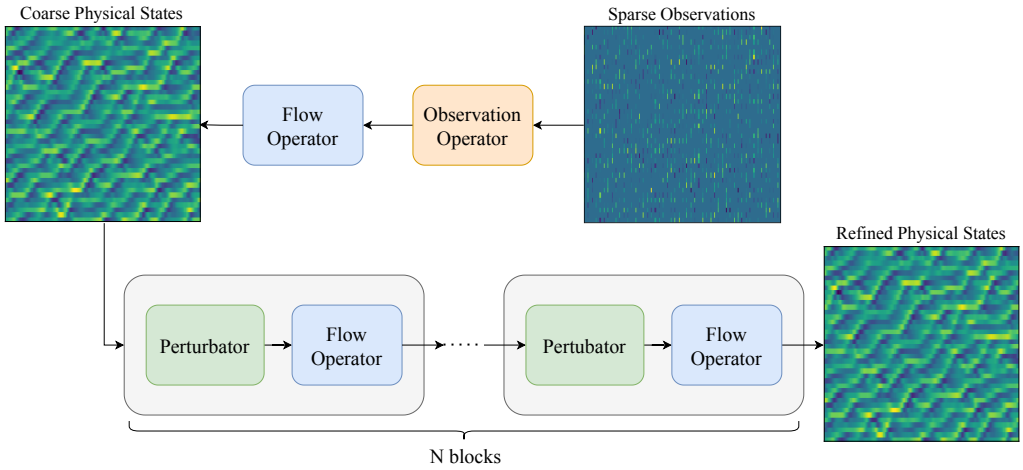

Figure 1: Proposed deep learning framework. The observation operator (orange box) and the flow operator (blue box) aim to convert sparse observations to complete physical states and bring them closer to the prior dynamics, respectively, while the second row demonstrates the process of refining reconstructed physical states with the help of perturbators (green box) and flow operators.

for the ground truth dynamic model, and then the dynamic system can be formulated as

$$\frac{\partial x(t)}{\partial t} = \mathcal{M}_{gt}\left(x(t)\right) \tag{1}$$

In real-world scenarios, the true dynamic model of earth systems is unknown and meteorologists develop physical models to estimate it, based on the laws of kinetics and thermodynamics. Denote $\mathcal{M}_{pr}$ for the prior model. In the ideal case, the bias of prior model should be bounded; that is, there exits a constant C, such that,

$$\|\mathcal{M}_{pr}(x) - \mathcal{M}_{gt}(x)\| \leq C \tag{2}$$

Observations $\{y(t_i)\}$ are assumed to be discrete and partial, with white Gaussian noise. Suppose that all the observations are obtained at time $t_i = t_0 + i\Delta t$, $i = 0, 1, ..., N$ and at time t_i, only components in domain Ω_{t_i} are observed. The formulation can be written as

$$y(t, p) = x(t, p) + \epsilon(t), \forall t \in \{t_0, t_0 + \Delta t, ..., t_0 + N\Delta t\}, \forall p \in \Omega_t \tag{3}$$

where $\epsilon(t)$ represents the white Gaussian noise.

The problem setting can be further explained within a deep learning paradigm. Our main goal is to construct an inverse map, implemented in the form of a neural network, from a time sequence of noisy and partial observations y to its corresponding reconstructed states x. The a priori knowledge is that the reconstructed states approximately satisfy certain dynamic equations with a bounded error. The major challenge is that it is difficult to incorporate the prior knowledge into neural network structures. Most of the previous work has relied on the minimization of the 4D-Var target function to tackle it. In our work, a pure neural network solution is given.

3 Learning Framework with Three Operators

To deal with the problem, we propose an end-to-end learning framework with three operators, including observation operator, flow operator and perturbator, as shown in Fig 1.

3.1 Observation Operator

The observation operator aims to learn lower-dimensional observations and convert them into physical states (Frerix et al., 2021). We design the structure of the observation operator based on the Residual U-Net structure (Ronneberger et al., 2015; Siddique et al., 2021). The hierarchical structure of U-Net allows it to capture both the large-scale and small-scale information of observation

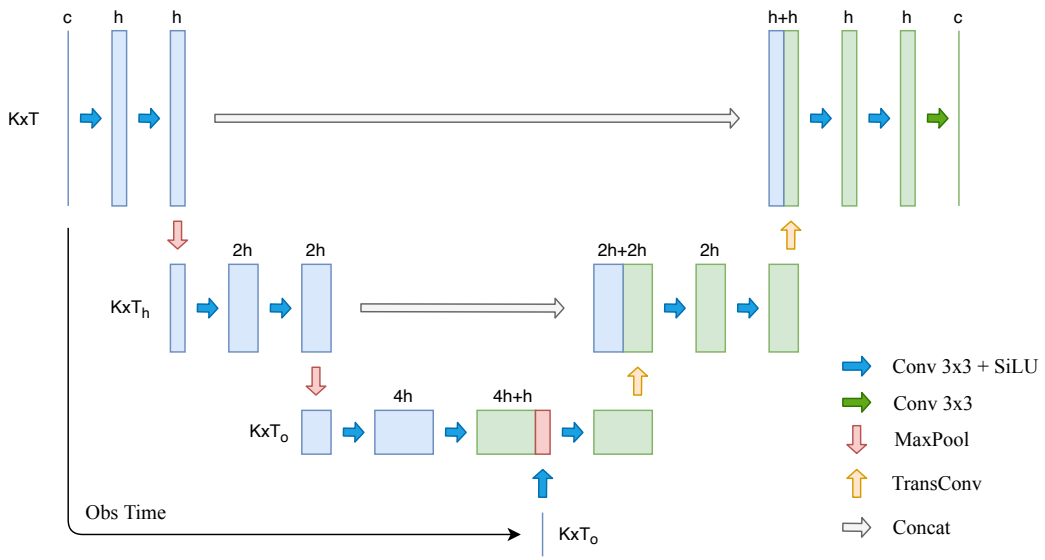

Figure 2: An example of the 3-layer U-Net architecture of the observation operator. We denote K for the spatial variables of physical states, T for the total sequence length, T_o for the sequence length with valid observations, T_h for the intermediate hidden sequence length. We require $T > T_h > T_o$.

states and is ubiquitously used in the reconstruction domain (Guan et al., 2019; Siddique et al., 2021; Punn & Agarwal, 2022; Feng et al., 2020). Moreover, the residual structure helps alleviate the vanishing gradient problem (He et al., 2016), allowing U-Net models with deeper layers to be designed.

Modified U-Net Structure In addition, this paper also modifies the network according to the specificity of the data assimilation issue, as illustrated below. Unlike images, of which the three dimensions are length, height, and channels, the inputs of data assimilation are sequences of observations (states), whose dimensions are time, location, and channels. The length and height of images are symmetric, while time and location are generally not. To tackle with this difference, we carry out dimensionality reductions in U-Net for data assimilation in the temporal dimension. What is more, our proposed observation operator adds a skip layer from the input to the bottom hidden variables. It is implemented by extracting valid observations in the observed moments from the input and concatenating them directly with the bottom hidden variables. The advantage of this operation is that the information of observations can be more effectively transferred through the network. The number of layers for U-Net can be set to three (Figure 2), four, or even more, depending on the sparsity of observations.

Reconstruction Loss The reconstruction loss $\mathcal{L}_{rec}$ is applied for the training of the observation operator. If we denote the network by Ψ and the output by $\hat{x} = \Psi(y)$, the loss function is

$$\mathcal{L}_{rec}(\hat{x}) = \|\hat{x} - x\|^2 \tag{4}$$

With the help of the observation operator, we are already able to construct physical states, but with relatively low accuracy. Instead of seeking help from traditional 4D-Var algorithms to further improve accuracy Frerix et al. (2021), we construct the so-called perturbator and flow operator for such purposes.

3.2 Flow Operator

Dynamic Loss Before proceeding to the introduction of the flow operator, let us first illustrate an important concept, dynamic loss. The dynamic loss measures how well the given sequence of states fits the considered dynamics, which is formulated by

$$\mathcal{L}_{dyn}(\hat{x}) = \|\Phi(\hat{x}) - \hat{x}\|^2 \tag{5}$$

where Φ is defined by

$$\Phi(x)(t) = x(t - \Delta t) + \int_{t-\Delta t}^{t} \mathcal{M}(x(u))du \tag{6}$$

Φ is a flow-dependent operator, which describes the hypothetical result of first moving the sequence one step backward and then integrating one step forward. The smaller the dynamic loss, the better the sequence of states fits the dynamics. If the sequence of states x perfectly matches the dynamic system, then $\Phi(x) = x$ holds, and the dynamic loss is zero.

Function of Flow Operator The major function of the flow operator is to restore the reconstructed sequence of states with a relatively high dynamic loss to the corresponding sequence that better conforms to the prior dynamics. Suppose that $x(t) \in \mathcal{X}$ is a sequence of reconstructed system's states, we require that the dynamic loss after the operator is smaller than a given value $\mathcal{L}_0$, which is usually orders of magnitude smaller than the original dynamic loss. Denote the flow operator by $\mathcal{F}$, and the requirement can be formulated as

$$\mathcal{L}_{dyn}(\mathcal{F}(x)) \leq \mathcal{L}_0 \ll \mathcal{L}_{dyn}(x) \tag{7}$$

Network Structure Instead of using the neural network to directly compute the improved output, we use the U-Net structure to compute the increment compared to the input state; that is,

$$\mathcal{F}(x) = \text{UNet}(x) + x \tag{8}$$

This subtle modification is to better ensure that the amount of modification is small and make the training process converge faster.

Network Training and Stopping Criterion We train the flow operator with the dynamic loss, and the training process is self-supervised because no additional label is required. For the case in which the prior dynamics are unbiased, we simply fix the number of epochs in advance and do the training. In contrast, if the prior dynamic model is inaccurate, we stipulate a stopping criterion for the training procedure:

$$\mathcal{L}_{dyn-pr}(\hat{x}) \geq \mathbb{E}_{x_{gt}}\left[\mathcal{L}_{dyn-pr}(x_{gt})\right] \tag{9}$$

where $\mathbb{E}_{x_{gt}}\left[\mathcal{L}_{dyn-pr}(x_{gt})\right]$ is the expectation of the prior dynamic loss of ground truth states, which measures the difference between the prior dynamics and ground truth. Because the error of prior dynamics is bounded, $\mathbb{E}_{x_{gt}}\left[\mathcal{L}_{dyn-pr}(x_{gt})\right]$ is also a bounded value smaller than infinity. As long as inequality 9 is not satisfied, we end the current training phase and move on to the next one. Inequality 9 guarantees that the dynamic loss of the reconstructed states is no smaller than that of the true states. It is a *necessary but not sufficient* condition to avoid overfitting in terms of dynamics, and any reconstructed states with a smaller dynamic loss than true states indicate that more information than expected has been learned from the prior dynamics.

3.3 PERTURBATOR

Function of Perturbator The role of the flow operator is to guarantee that the reconstructed field conforms to the dynamics, but this is insufficient for data assimilation, because there are infinitely many sequences of states that conform to the dynamics, and we need to find the one that also fits the observations well. On the other hand, thanks to the flow operator's ability to guarantee that the output is consistent with the prior dynamics, we can add a new neural network, called the perturbator, in front, which has more free space to learn from observations and labels. It can use the learned knowledge to revise the reconstruction states without considering the problem of conforming to the prior dynamics.

In this way, we **decouple** the two goals of conforming to observations and conforming to prior dynamics, and both the flow operator and perturbator can perform their own duties. Specifically, The perturbator uses the observations and labels it has learned to perturb the reconstructed states to make it deviate from the original flow, and then the flow operator corrects the deviated flow to make it a new flow that conforms to the prior dynamics.

Cooperation with Flow Operator The perturbator and flow operator always work in pairs. Given a sequence of states with relatively low dynamic loss $\mathcal{L}_{dyn}$ and relatively high reconstruction loss $\mathcal{L}_{rec}$, assuming that we have trained a flow operator $\mathcal{F}$, the function of the perturbator $\mathcal{P}$ can be

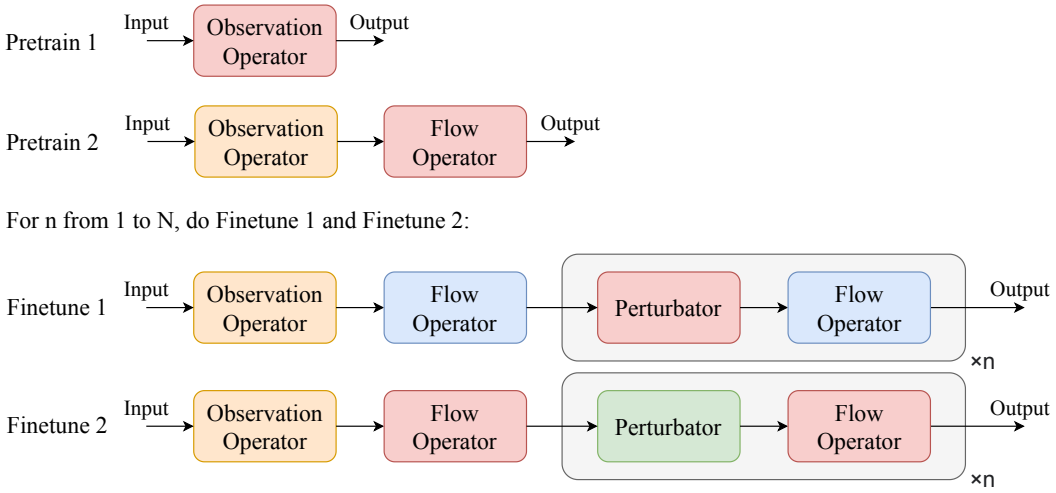

Figure 3: Pretraining and finetuning procedure for the framework. In each step, only the parameters of the network modules that are represented by red boxes are trained, while the parameters of those represented by other colors are fixed. The color markings match those in Figure 1. Pretraining is performed separately for observation operator (Pretrain 1) and flow operator (Pretrain 2), and finetuning is performed alternatively for the perturbators (Finetune 1) and flow operators (Finetune 2). We adopt this seemingly complicated training approach because these operators must be individually trained with parameters of other operators fixed to perform as we expect.

formulated as

$$\begin{aligned} \mathcal{L}_{rec}(\mathcal{F}(\mathcal{P}(x))) &\leq \mathcal{L}_{rec}(x) \\ \mathcal{L}_{dyn}(\mathcal{F}(\mathcal{P}(x))) &\sim \mathcal{L}_{dyn}(x) \end{aligned} \tag{10}$$

The first formulation requires that the reconstruction loss is reduced, and the second requires that the dynamic loss should remain in the same order of magnitude. Ideally, as more pairs of perturbators and flow operators are applied to the input, the reconstruction loss will continuously decrease while maintaining the dynamic loss.

We emphasize that the output of the perturbator $\mathcal{P}(x)$ is likely to have both high reconstruction loss and high dynamic loss, but it does not matter because we only require that the losses after the flow operator $\mathcal{F}(\mathcal{P}(x))$ are small.

Network Structure A typical U-Net structure is adopted to build the perturbator, and in this case we use a hybrid loss to learn from both the dynamics and labels.

$$\mathcal{L}_{dis}(\hat{x}) = \mathcal{L}_{rec}(\hat{x}) + \lambda \mathcal{L}_{dyn}(\hat{x}) \tag{11}$$

3.4 Network Training

The training process of the proposed framework can be divided into two parts: pretraining and finetuning. During pretraining, network parameters of the three operators in the framework are roughly determined, while the purpose of finetuning is to further adjust and modify the parameters to fit the recurrent structure. We let readers refer to Figure 3 for details.

4 Numerical Experiments

In this section, numerical experiments are reported for Lorenz-63 and Lorenz-96 systems. Owing to their chaotic properties and similarities with meteorological systems, Lorenz systems are widely used for evaluation of various data assimilation algorithms, especially in data-driven and machine learning case-studies (Lguensat et al., 2017; Raissi, 2018; Fablet et al., 2021b; Frerix et al., 2021).

Dynamic System	Model	# of Blocks	R-score	LossDyn	Time-to-solution (s)
Lorenz-63	NN with Three-Op	2	1.28	1.10e-3	3.1
	NN with Three-Op	4	0.923	3.82e-4	4.5
	NN with Three-Op	6	0.799	2.08e-4	5.8
	NN with Three-Op	8	**0.751**	**1.52e-4**	6.4
	GENN aG-Conv	20	1.34	>1.00e-2	27.1
Lorenz-96	NN with Three-Op	2	0.571	1.40e-3	7.8
	NN with Three-Op	4	0.450	5.09e-4	9.8
	NN with Three-Op	6	0.397	3.14e-4	11.5
	NN with Three-Op	8	0.373	2.66e-4	13.5
	NN with Three-Op	10	**0.367**	**2.36e-4**	15.8
	GENN aG-LSTM	20	0.38	>1.00e-2	47.3

Table 1: Results of comparative experiments. R-score is the synonym for reconstruction loss, and LossDyn is the synonym for dynamic loss. We report the reconstruction and dynamical performance of the proposed framework for the Lorenz-63 and Lorenz-96 systems. The results of the state-of-the-art neural networks draw directly from the original paper (Fablet et al., 2021b). Bold values refer to the best score. All of the models are tested on an independent test set with 2000 samples for Lorenz-63 and 3000 samples for Lorenz-96. The batch size is set to eight, and all the evaluations are performed on one TITAN RTX GPU. Test time may vary across experiments, and we average the results over 10 independent tests.

4.1 Design of Experiments

The Lorenz-63 and Lorenz-96 dynamic systems are governed by the ordinary differential equations (ODE) expressed as Equation 12 and Equation 13, respectively (Lorenz, 1963; 1996). Both systems are one-dimensional. We further design two experiments to test the performance of our framework in different scenarios.

We refer to the previous work for the selection of parameterization (Fablet et al., 2021b), where for Lorenz-63, $\sigma = 10$, $\rho = 28$, $\beta = 8/3$ and for Lorenz-96, $N = 40$, $F = 8$. We simulate time series with a 0.01 time step for Lorenz-63 and a 0.05 time step for Lorenz-96, using a RK45 integration scheme (Dormand & Prince, 1980). The assimilation window length is 200 time steps. For the Lorenz-63 system, the observation data is sampled every eight steps for only the first component, and for the Lorenz-96 system, it is sampled every four steps for random half of the components. In both cases, a Gaussian noise with a variance of 2.0 is involved.

$$\begin{cases} \frac{dX_1}{dt} &= \sigma\left(X_2 - X_1\right) \\ \frac{dX_2}{dt} &= \rho X_1 - X_2 - X_1 X_3 \\ \frac{dX_3}{dt} &= X_1 X_2 - \beta X_3 \end{cases} \tag{12}$$

$$\frac{dX_i}{dt} = \left(x_{i+1} - x_{i-2}\right) x_{i-1} - x_i + F,\ i = 1, 2, ..., N \tag{13}$$

For the Lorenz-63 system, We adopt a 4-layer U-Net architecture for the construction of the observation operator, perturbator, and flow operator, where $c = 3$, $K = 1$, $T = 200$, and $h = 32$. For the Lorenz-96 system, we adopt a 3-layer architecture, where $c = 1$, $K = 40$, $T = 200$, and $h = 32$.

4.2 Unbiased Prior Dynamics

First, we consider the case in which the prior dynamics are identical to the ground truth dynamic, that is $\mathcal{M}_{gt} = \mathcal{M}_{pr}$. We use a total of eight and ten blocks to train the deep learning framework for the Lorenz-63 and Lorenz-96 systems, respectively, including an observation operator and a flow operator in the front, followed by pairs of perturbators and flow operators, as shown in Figure 1 with $N = 3$ and $N = 4$.

In Table 1, we compare our finetuned neural network structure with the state-of-the-art networks (Fablet et al., 2021b) in terms of R-score, LossDyn, and time-to-solution on test set. Our proposed

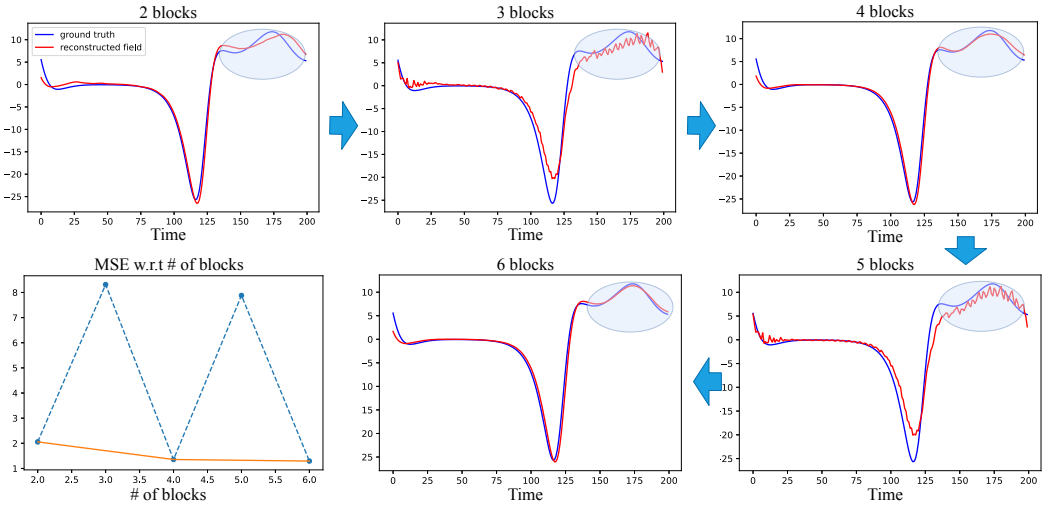

Figure 4: An intuitive presentation of how the proposed neural network structure works. A sample from the test set is randomly selected to evaluate the proposed framework with six blocks. We truncate the model into shorter lengths, from two blocks with only one observation operator and one flow operator, to six blocks, and evaluate their performances. The five upper right panels demonstrate the change in reconstructed states when more blocks are added, while the bottom-left panel shows the relationship between the R-score on the test set and the number of blocks.

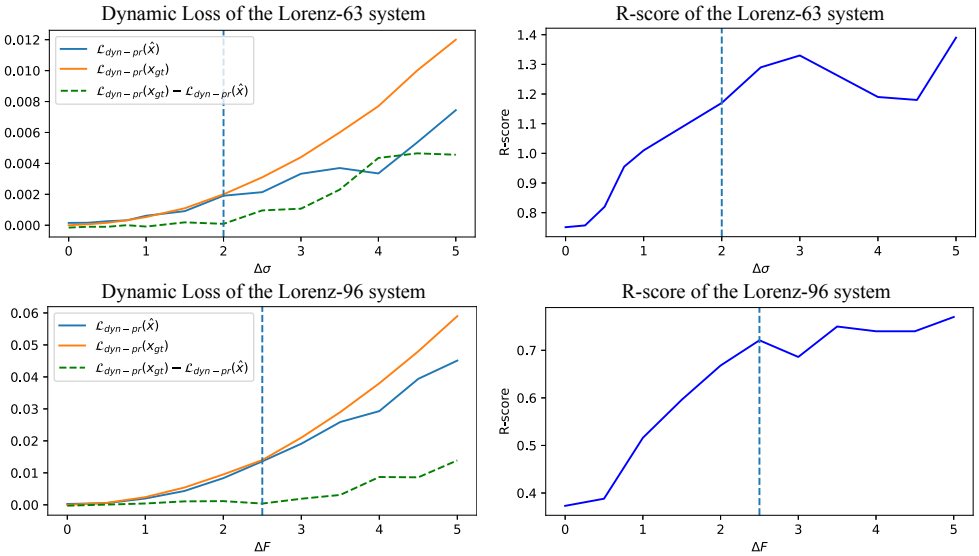

Figure 5: The left panels shows the dynamic loss of reconstructed states (blue line), the biases of prior dynamics (orange line), and their differences (green dotted line) against $\Delta\sigma$ or ΔF. The right panels shows the reconstruction loss (R-score) in the test set against $\Delta\sigma$ or ΔF. The upper and lower panels represent the results of the Lorenz-63 and Lorenz-96 systems, respectively.

method achieves better reconstruction performance than the state-of-the-art method on the metric of R-score. We reduce the reconstruction error by 44% and 3% for Lorenz-63 and Lorenz-96 systems, respectively, and reduce the dynamic loss by two orders of magnitude, which are significant achievements. Moreover, the dynamic loss is at least one order of magnitude smaller than the state-of-the-art, regardless of the number of blocks. In terms of time-to-solution, our proposed framework is four times and three times faster than the state-of-the-art method for the two dynamic systems.

We select an example from the test set in Figure 4 to further illustrate how our architecture works. Focus on the blue oval box area in the upper right corner, where we find how the cooperation of perturbator and flow operator succeeds in correcting the initial states. In the beginning, the states generated by two blocks are clearly inconsistent with the ground truth in the upper right region. After a perturbator is added (block 3), disturbances with high frequencies are involved in the misaligned area to distinguish them from well-constructed areas. If a flow operator is further added behind the perturbator (block 4), the high-frequency noise disappears, and the inconsistency is corrected to some extent. Such a perturbation-refinement step is repeated (block 5 & block 6) and the misalignment becomes smaller with repetition. The lower-left panel also shows a continuous decrease in the reconstruction error, from two to four to six blocks (orange line).

4.3 Biased Prior Dynamics

In the following experiment, biases are introduced to the prior dynamics by perturbing the ODE parameters. In the case of Lorenz-63 systems, σ in Equation 12 is selected to be the perturbed parameter, and in the case of Lorenz-96 system, F in Equation 13 is selected.

Specifically, we select $\Delta\sigma$ from the set $\{0.25, 0.50, 0.75, 1.00, 1.50, 2.00, 2.50, 3.00, 3.50, 4.00, 4.50, 5.00\}$, select ΔF from the set $\{0.50, 1.00, 1.50, 2.00, 2.50, 3.00, 3.50, 4.00, 4.50, 5.00\}$, and conduct numerical experiments accordingly. The neural network structures and training losses are almost identical to those in the experiment with no bias, except that we use the prior dynamics rather than ground truth dynamics to construct the dynamic loss.

We report our results in Figure 5. Take the Lorenz-63 system as an example. As illustrated in the upper-right figure, when $\Delta\sigma$ is smaller than 1.0, the increase in R-score is smaller than 20%, which shows that our framework is robust to small biases. When $\Delta\sigma$ is smaller than 2.00, the R-score increases as $\Delta\sigma$ increases, whereas if $\Delta\sigma$ is larger than 2.00, the R-score fluctuates without monotonic changes. Such a phenomenon can be explained by looking into how our network structure works: When $\Delta\sigma$ is small, the difference between the prior dynamics and ground truth is small, and the information of the prior dynamics can be successfully exploited by the flow operator. Therefore, as $\Delta\sigma$ gets larger, useful information provided by the prior dynamics decreases and the performance of our framework worsens. However, if $\Delta\sigma$ is too large, it is no longer necessary for the flow operator to learn from the bad prior dynamics; it is cheaper to learn directly from ground truth labels. Hence, in this case, the value of $\Delta\sigma$ does not affect the performance of the neural network. The results shown in the upper-left figure can further confirm our explanation. When $\Delta\sigma < 2$, the difference between $\mathcal{L}_{dyn-pr}(\hat{x})$ and $\mathcal{L}_{dyn-pr}(x_{gt})$ is close to zero, which is consistent with our stopping criterion (Inequality 9), so the network works as expected. However, if $\Delta\sigma > 2$, the bias of prior dynamics is too large for the flow operator to learn, and solely learning from the ground truth is sufficient for the neural network to generate states whose dynamic loss is smaller than $\mathcal{L}_{dyn-pr}(x_{gt})$. The findings above also holds for the Lorenz-96 system, as shown in the lower figures. The difference is that the threshold changes from 2.0 to around 2.5. They provide further evidence that our network learns features from prior dynamics rather than simply remembering ground-truth labels, and that our framework works properly despite biases in prior dynamics.

5 Conclusions

We have introduced a deep learning framework for data assimilation. The key novelty of our research is that we develop the flow operator to learn from the prior dynamics and continuously refine the reconstructed states. The results of numerical experiments on Lorenz-63 and Lorenz-96 systems support the relevance of our framework. In terms of efficiency, our framework has been found to be four times and three times faster than state-of-the-art networks on Lorenz-63 and Lorenz-96 systems, respectively. In terms of assimilation accuracy, our framework has reduced the dynamic loss by two orders of magnitude, and at least maintained the reconstruction accuracy, compared with the state-of-the-art method. It has also been found to be well-adapted to biases in prior dynamics.

The results indicate that the proposed method can serve as a new method for integrating ODE information into neural networks. They suggest that ODE information can be applied to modify coarse reconstruction states instead of being explicitly embedded into neural network structures.

REFERENCES

Rossella Arcucci, Jiangcheng Zhu, Shuang Hu, and Yi-Ke Guo. Deep data assimilation: integrating deep learning with data assimilation. *Applied Sciences*, 11(3):1114, 2021.

P Bergthorsson, BR Döös, S Fryklund, O Haug, and R Lindquist. Routine forecasting with the barotropic model. *Tellus*, 7(2):272–274, 1955.

Massimo Bonavita and Patrick Laloyaux. Machine learning for model error inference and correction. *Journal of Advances in Modeling Earth Systems*, 12(12):e2020MS002232, 2020.

Julien Brajard, Alberto Carrassi, Marc Bocquet, and Laurent Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the lorenz 96 model. *Journal of Computational Science*, 44:101171, 2020.

James Carpenter, Peter Clifford, and Paul Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings-Radar, Sonar and Navigation*, 146(1):2–7, 1999.

Rosangela Saher Cintra and Haroldo F de Campos Velho. Data assimilation by artificial neural networks for an atmospheric general circulation model. *Advanced applications for artificial neural networks*, 265, 2018.

HF de Campos Velho, S Stephany, AJ Preto, NL Vijaykumar, and AG Nowosad. A neural network implementation for data assimilation using mpi. *WIT Transactions on Information and Communication Technologies*, 27, 2002.

John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Geir Evensen. The ensemble kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

Ronan Fablet, Mohamed Mahmoud Amar, Quentin Febvre, Maxime Beauchamp, and Bertrand Chapron. End-to-end physics-informed representation learning for satellite ocean remote sensing data: Applications to satellite altimetry and sea surface currents. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, (3), 2021a.

Ronan Fablet, Bertrand Chapron, Lucas Drumetz, Etienne Mémin, Olivier Pannekoucke, and François Rousseau. Learning variational data assimilation models and solvers. *Journal of Advances in Modeling Earth Systems*, 13(10):e2021MS002572, 2021b.

Alban Farchi, Patrick Laloyaux, Massimo Bonavita, and Marc Bocquet. Using machine learning to correct model error in data assimilation and forecast applications. *Quarterly Journal of the Royal Meteorological Society*, 147(739):3067–3084, 2021.

Jinchao Feng, Jianguang Deng, Zhe Li, Zhonghua Sun, Huijing Dou, and Kebin Jia. End-to-end res-unet based reconstruction algorithm for photoacoustic imaging. *Biomedical optics express*, 11(9):5321–5340, 2020.

M Fisher. Minimization algorithms for variational data assimilation. In *Proc. of ECMWF Seminar on Recent Development in Numerical Methods for Atmospheric Modelling (7-11 September 1998, Reading, UK)*. ECMWF, 1998.

Thomas Frerix, Dmitrii Kochkov, Jamie Smith, Daniel Cremers, Michael Brenner, and Stephan Hoyer. Variational data assimilation with a learned inverse observation operator. In *International Conference on Machine Learning*, pp. 3449–3458. PMLR, 2021.

Lev Semenovich Gandin. Objective analysis of meteorological field. *Gidrometeorologicheskoe Izdate'stvo*, 286, 1963.

Steven Guan, Amir A Khan, Siddhartha Sikdar, and Parag V Chitnis. Fully dense unet for 2-d sparse photoacoustic tomography artifact removal. *IEEE journal of biomedical and health informatics*, 24(2):568–576, 2019.

Fabrício P Härter and Haroldo Fraga de Campos Velho. New approach to applying neural network in nonlinear dynamic model. *Applied Mathematical Modelling*, 32(12):2621–2633, 2008.

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*, pp. 770–778, 2016.

William W Hsieh and Benyang Tang. Applying neural network models to prediction and data analysis in meteorology and oceanography. *Bulletin of the American Meteorological Society*, 79(9): 1855–1870, 1998.

Eugenia Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

Redouane Lguensat, Pierre Tandeo, Pierre Ailliot, Manuel Pulido, and Ronan Fablet. The analog data assimilation. *Monthly Weather Review*, 145(10):4093–4107, 2017.

Andrew C Lorenc and F Rawlins. Why does 4d-var beat 3d-var? *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(613):3247–3257, 2005.

Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996.

Julian Mack, Rossella Arcucci, Miguel Molina-Solana, and Yi-Ke Guo. Attention-based convolutional autoencoders for 3d-variational data assimilation. *Computer Methods in Applied Mechanics and Engineering*, 372:113291, 2020.

Narinder Singh Punn and Sonali Agarwal. Modality specific u-net variants for biomedical image segmentation: a survey. *Artificial Intelligence Review*, pp. 1–45, 2022.

Florence Rabier, Heikki Järvinen, E Klinker, J-F Mahfouf, and A Simmons. The ecmwf operational implementation of four-dimensional variational assimilation. i: Experimental results with simplified physics. *Quarterly Journal of the Royal Meteorological Society*, 126(564):1143–1170, 2000.

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

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.

Yoshikazu Sasaki. Some basic formalisms in numerical variational analysis. *Monthly Weather Review*, 98(12):875–883, 1970.

Nahian Siddique, Sidike Paheding, Colin P Elkin, and Vijay Devabhaktuni. U-net and its variants for medical image segmentation: A review of theory and applications. *Ieee Access*, 9:82031–82057, 2021.

Hyo-Jong Song, Seoleun Shin, Ji-Hyun Ha, and Sujeong Lim. The advantages of hybrid 4denvar in the context of the forecast sensitivity to initial conditions. *Journal of Geophysical Research: Atmospheres*, 122(22):12–226, 2017.

Nandamudi Lankalapalli Vijaykumar, Stephan Stephany, HF Campos Velho, AJ Preto, and AG Nowosad. Optimized neural network code for data assimilation. In *Brazilian Congress on Meteorology*, volume 12, pp. 38413849, 2002.

Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

A APPENDIX

A.1 DATASET CONSTRUCTION

We do a long integration over time and divide the time sequence into three parts. The first part is for building the training set, the second is for the validation set, and the last is for the test set, as shown in Figure 6. For each part, we randomly sample time series of the same length as the assimilation window multiple times, and then use the sampled series as the dataset.

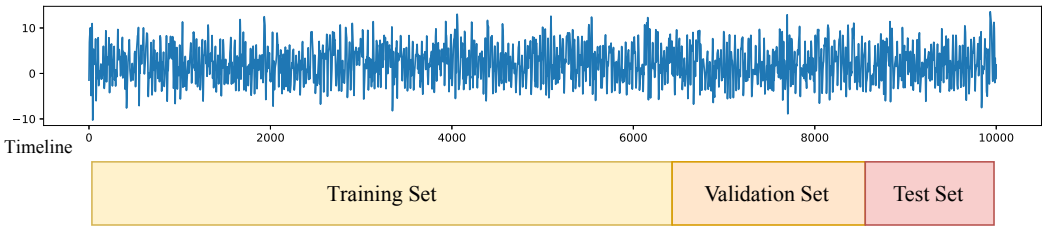

Figure 6: Demonstration of the construction and splitting of the dataset. We let readers refer to Section A.1 for detailed explanation.

Specifically, for the Lorenz-63 system, the training data comprises 10000 sequences sampled from time series with 12000 steps; the validation data comprises 2000 sequences from time series with 5000 steps and the test data comprises with 3000 sequences from 10000 time steps. For the Lorenz-96 system, the training data comprises 2000 sequences sampled from time series with 7500 steps; the validation data comprises 300 sequences from time series with 2500 steps and the test data comprises with 200 sequences from 10000 time steps.

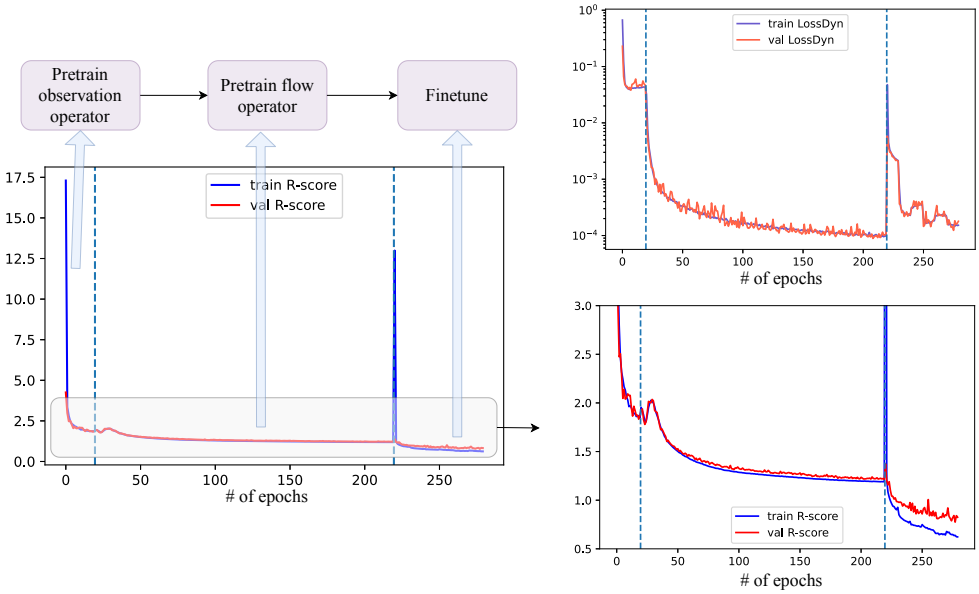

Figure 7: The variation of R-score and dynamic loss for training and validation with the number of epochs for the Lorenz-63 system. Two vertical dashed lines divide the training procedure into three periods: pretraining observation operator, pretraining flow operator, and finetuning. The left panel demonstrates the changes in R-score during training, the details of which are enlarged and shown in the bottom right panel. The red and blue dashed lines in the bottom right panel show how continuing to train the observation operator without adding new operators would result in a change in the reconstructed loss function. The top right panel demonstrates the variation of dynamic loss during training.

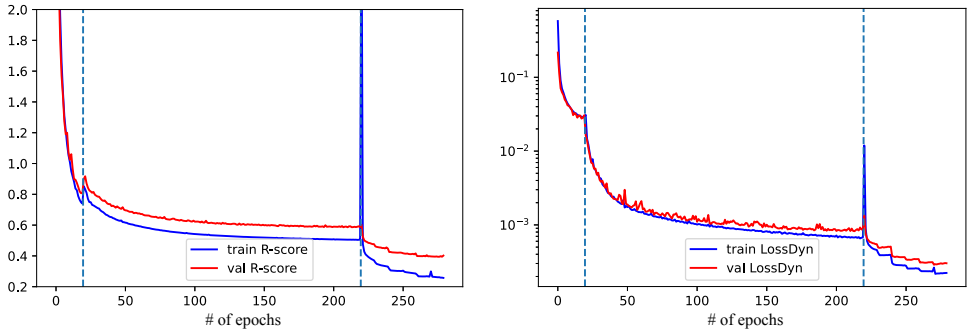

Figure 8: The variation of R-score and dynamic loss with the number of epochs for the Lorenz-96 system. The left panel demonstrates the changes in the R-score during training, while the right panel demonstrates the variation of the dynamic loss during training.

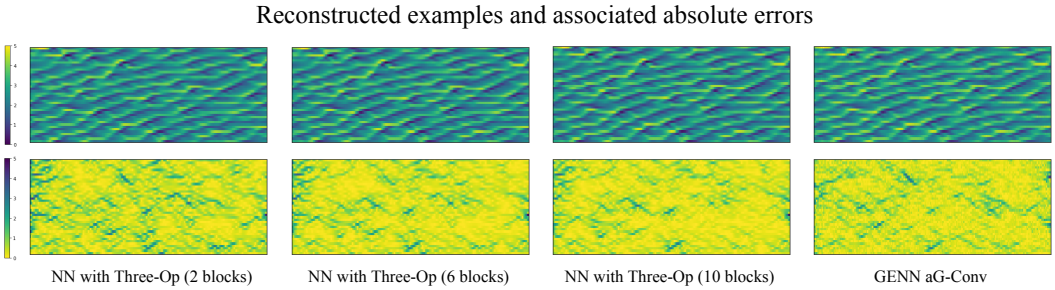

Figure 9: The sequences of reconstructed states and their corresponding absolute error maps of the Lorenz-96 system, with the horizontal axis being the time axis and the vertical axis being the spatial axis. All the displayed figures in the same row share the same colormap.

A.2 Loss Change During Training

The variations of reconstruction and dynamic loss against the number of epochs for Lorenz-63 and Lorenz-96 systems are reported in Figure 7 and Figure 8, respectively. During pretraining, the observation operator learns a map from the observational space to the physical states at first. Since a shift between two different spaces takes place in this period, we see a steep decrease of loss within the first 20 epochs. After the pretraining of observation operator is finished, we start the pretraining of flow operator to correct the flow dynamics of reconstructed states. The pretraining of the flow operator is relatively slow because a self-supervised loss is employed in this procedure. We claim that the flow operator is indispensable in further decreasing the reconstruction loss and dynamic loss, for the reason that without the flow operator, the observation operator would quickly lead to overfitting, in other words, the difference between the training loss and validation loss will diverge, as shown in the red and blue dashed lines in the figure. During the finetuning procedure, the reconstruction loss drops steadily while the dynamic loss fluctuates. This is not surprising considering that the finetuning procedure alternates between the perturbator and flow operator. If the training object is the perturbator, the dynamic loss is relatively high because the optimized target is the hybrid loss; if the training object is flow operator, the dynamic loss is relatively low because the optimized target is just the dynamic loss. Such alternating finetuning process contributes to the continuous decrease of the R-score.

A.3 Sample Study of the Lorenz-96 System

Figure 9 demonstrates the physical states reconstructed by different neural networks and their associated absolute error maps. Visually, the dark regions in the error map becomes smaller and lighter

as the number of block grows, which confirms the effects of the collaboration of the perturbator and flow operator in this example.