Score-based Data Assimilation for a Two-Layer Quasi-Geostrophic Model

François Rozet
University of Liège
francois.rozet@uliege.be

Gilles Louppe
University of Liège
g.louppe@uliege.be

Abstract

Data assimilation addresses the problem of identifying plausible state trajectories of dynamical systems given noisy or incomplete observations. In geosciences, it presents challenges due to the high-dimensionality of geophysical dynamical systems, often exceeding millions of dimensions. This work assesses the scalability of score-based data assimilation (SDA), a novel data assimilation method, in the context of such systems. We propose modifications to the score network architecture aimed at significantly reducing memory consumption and execution time. We demonstrate promising results for a two-layer quasi-geostrophic model. The code for all experiments is made available at https://github.com/francois-rozet/sda.

1 Introduction

N-layer quasi-geostrophic (NLQG) models [1] are special cases of the Navier-Stokes equations that have been extensively used to describe the dynamics of oceans and atmospheres. NLQG models consider a stratified fluid of N superimposed layers of constant uniform densities; the layers being stacked according to increasing density. The state of the fluid is fully described by the potential vorticity field q and the velocity fields (u, v) of each layer. In practice, it is necessary to discretize temporally and spatially to solve/integrate the quasi-geostrophic equations, which leads to physical phenomena occurring at a smaller scale than the numerical resolution to be missed [1–4]. In the long run, neglecting such phenomena can lead to poor simulations. It is therefore common to simulate such models at high temporal and spatial resolutions to ensure the quality of the simulations, but this comes at a significant computational cost.

Although they make (shallow water) assumptions to neglect some terms of the Navier-Stokes equations, NLGQ models effectively capture the characteristic features of turbulent ocean systems, such as jet streams, mesoscale eddies, and ocean currents. As such, they are good candidates for designing and benchmarking data assimilation (DA) [5–11] algorithms. Formally, DA targets the problem of inferring the posterior distribution

$$p(x_{1:L} \mid y) = \frac{p(y \mid x_{1:L})}{p(y)} p(x_1) \prod_{i=1}^{L-1} p(x_{i+1} \mid x_i) \tag{1}$$

of discrete-time state trajectories $x_{1:L} = (x_1, x_2, \ldots, x_L)$ given an observation y resulting from an observation process $p(y \mid x_{1:L})$. In geosciences, the physical model underlying transition dynamics $p(x_{i+1} \mid x_i)$ is typically well known and the observation process is generally formulated as $y = \mathcal{A}(x_{1:L}) + \eta$, where $\mathcal{A}$ is the measurement function and η is a stochastic additive term that accounts for instrumental noise and systematic uncertainties.

In this setting, the simulation quality is very important as it strongly impacts the relevance of inference results. Unfortunately, high resolutions rapidly become a computational burden for classical DA

algorithms, such as variational methods [5–9] that require to repeatedly differentiate through the physical model, and Monte Carlo methods [12–15] that conduct large numbers of simulations.

Conversely, score-based data assimilation (SDA) [16], a novel DA method drawing its roots from score-based generative modeling, only relies on the physical model to generate training data. After training, inference can be carried out without relying on the model and at lower temporal and spatial resolutions. However, applying SDA to (very) high-dimensional systems remains an engineering challenge, which we attempt to address in this work.

2   Background

Score-based generative modeling has shown remarkable capabilities, powering many of the latest advances in image, video and audio generation [17–21]. In this section, we shortly review score-based generative models and outline how they can be used for solving data assimilation problems with SDA [16].

Continuous-time score-based generative models   Adopting the notations of Rozet et al. [16], samples x drawn from a distribution $p(x)$ are progressively perturbed through a continuous-time stochastic diffusion process. This process determines a series of marginal distributions $p(x(t))$ from $t = 0$ to $t = 1$ for which $p(x(0)) \approx p(x)$ (clean data) and $p(x(1)) \approx \mathcal{N}(0, I)$ (pure noise). By design, the diffusion process can be "reversed" in order to transform noise $x(1) \sim \mathcal{N}(0, I)$ into data samples $x(0) \sim p(x(0))$. However, this reverse process requires access to the quantity $\nabla_{x(t)} \log p(x(t))$ known as the score of $p(x(t))$.

A score network is a neural network $s_\phi(x(t), t)$ trained – usually by denoising score matching [22, 23] – to approximate the score $\nabla_{x(t)} \log p(x(t))$. After training, a score network can be plugged into the reverse process as a substitute for the true score in order to generate data samples from $p(x(0))$. A handy property of score-based models is that, under some assumptions on the likelihood $p(y \mid x)$, it is possible to approximate the posterior score

$$\nabla_{x(t)} \log p(x(t) \mid y) = \nabla_{x(t)} \log p(x(t)) + \nabla_{x(t)} \log p(y \mid x(t)) \tag{2}$$

without retraining the score network $s_\phi(x(t), t)$, and hence to generate data samples from $p(x(0) \mid y)$.

Score-based data assimilation   In score-based data assimilation, the data we try to model are trajectories $x_{1:L}$ and we therefore approximate the trajectory score $\nabla_{x_{1:L}(t)} \log p(x_{1:L}(t))$. Rozet et al. [16] show that an element $\nabla_{x_i(t)} \log p(x_{1:L}(t))$ of the trajectory score can be approximated locally by $\nabla_{x_i(t)} \log p(x_{i-k:i+k}(t))$ for some $k \geq 1$. They therefore propose to train a local score network $s_\phi(x_{i-k:i+k}(t), t)$ to approximate the score over short segments, and compose this local score at inference to generate arbitrarily long trajectories. In addition, the authors introduce a novel approximation for the likelihood score $\nabla_{x_{1:L}(t)} \log p(y \mid x_{1:L}(t))$ in the assumption of a Gaussian observation process $p(y \mid x_{1:L}) = \mathcal{N}(y \mid \mathcal{A}(x_{1:L}), \Sigma_y)$, which covers many observation scenarios. Finally, to prevent approximation errors from accumulating along the simulation of the reverse process, Rozet et al. [16] perform a few Langevin Monte Carlo corrections between each step of the discretized reverse process.

3   Task

Although the Kolmogorov system presented in the experiments conducted by Rozet et al. [16] is high-dimensional (tens of thousands of dimensions) compared to what is approachable with classical posterior inference methods, it remains small in comparison to the millions of dimensions of some operational DA systems.

In this study, we attempt to perform data assimilation for a two-layer quasi-geostrophic (2LQG) model. We choose a torus-like $1000\,\text{km}$ periodic domain, discretized into 256×256 spatial bins. The state is described by the potential vorticity field q^l and the zonal and meridional velocity fields u^l and v^l for each layer $l \in \{1, 2\}$. A state snapshot $x = (q^1, u^1, v^1, q^2, u^2, v^2)$ is therefore a 256×256 grid with 6 channels, which exceeds the dimensionality of Rozet et al. [16]'s Kolmogorov system by a factor 48.

We use the `pyqg` [24] Python library to solve the quasi-geostrophic equations using pseudo-spectral methods. To ensure the quality of the simulation, the domain is discretized into 512×512 spatial bins and the integration timestep is set to $15 \, \text{min}$. We coarsen the simulation to the target resolution (256×256) afterwards and separate two snapshots x_i and x_{i+1} by a day ($24 \, \text{h}$). To ensure that the system has reached statistical stationarity, that is $p(x_i) = p(x_{i+1})$, it is simulated for 5 years starting from random initial conditions before saving the first state x_1 [2–4].

4 Architecture

The high-dimensionality of the task at hand (and operational DA systems) introduces many engineering challenges. Notably, naively scaling the U-Net [25] inspired score network architecture proposed by Rozet et al. [16] to our task results in memory consumption exceeding the total memory of a A5000 GPU ($24 \, \text{GB}$) when training on segments, which gets worse at inference on full trajectories. We propose several modifications to the architecture to address these problems, avoiding low-level tricks such as custom CUDA kernels.

First, at inference, except for the extremities of the trajectory, only the $k + 1$-th element of the local score network's output is actually used. The other elements are therefore wasting both memory and compute resources. Instead, we make the score network fully-convolutional in the time axis, such that it can be applied to trajectories of any length. However, we limit its temporal receptive field such that it can still be trained on short segments. To do so, we make all layers 2-D spatial convolutions (no mixing along the time axis) and strategically add a few ($2k$) 1-D temporal convolutions. Assuming the same number of channels per state, this modification roughly reduces the amount of memory consumed and computation performed at inference by a factor $2k + 1$.

Still, training on segments of 256×256 grid states consumes a lot of memory. To solve this, we combine two ideas: gradient checkpointing and inverted bottleneck blocks. Gradient checkpointing [26, 27] is an automatic differentiation trick that consists in only keeping in memory some intermediate values of the computation graph, instead of all. The missing values are recomputed during the backward pass, effectively trading memory consumption for execution time. Inverted bottleneck blocks [28, 29] consist in limiting the number of channels of a convolutional network except within residual blocks where the number of channels is expanded to perform non-linear computations and contracted back afterwards. By checkpointing only the intermediate values with a limited number of channels, we are able to reduce memory consumption by $72 \, \%$, while increasing execution time by only $27 \, \%$.

A diagram of the architecture incorporating the proposed modifications is provided in Figure 2.

5 Results

Following Rozet et al. [16], we choose a variance preserving diffusion process for our experiments and simulate the reverse process in 256 evenly spaced discretization steps and one Langevin Monte Carlo correction per step. The score networks are trained once and then evaluated under various observation scenarios.

As described in Section 4, the local score network presents a U-Net [25] inspired architecture with efficiency-oriented modifications. The temporal receptive field of the score network is such that $k = 4$ and, therefore, it can be trained on segments of length $2k + 1 = 9$ instead of entire trajectories. Using `pyqg` [24], we generate 1024 trajectories of 32 state snapshots, which are split into training ($80 \, \%$), validation ($10 \, \%$) and evaluation ($10 \, \%$) sets. For practical reasons, the vorticity and velocity fields (channels) are standardized to have unit variance. We train the score network for 1024 epochs ($48 \, \text{h}$ on a single A5000 GPU) with the AdamW [30] optimizer and a linearly decreasing learning rate. Further architecture and training details are provided in Appendix A.

We consider a scenario where weather stations placed regularly over the domain measure the local velocities (u^1, v^1, u^2, v^2) every day for a month ($L = 32$). We generate an observation y for a trajectory of the evaluation set. Given the observation, we sample three trajectories with SDA and find that they closely resembles the original trajectory, as illustrated in Figure 1. A similar experiment where weather stations are placed randomly upon the domain and with more measurement noise is presented in Figure 3. In both cases, however, we observe that the generated trajectories exhibit

less small-scale details than a simulation, which indicates that the score network does not model the data perfectly. There are several avenues to explore in order to address this issue, such as training for longer with more data or increasing the capacity (more channels or layers) of the score network. Fourier neural operators (FNOs) could be another way to make the network more expressive and have proved to be effective for emulating partial differential equations [4, 31, 32].
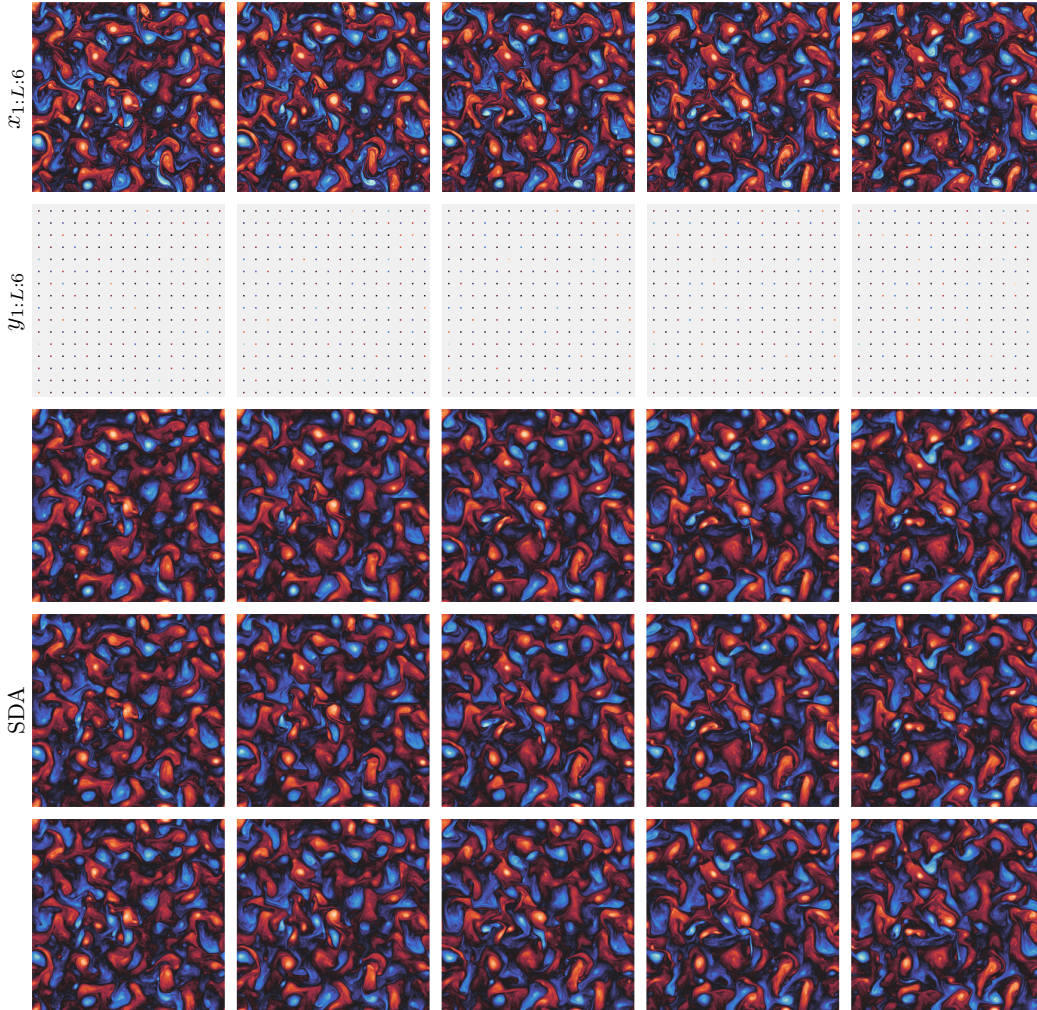

Figure 1: Example of sampled trajectories for a spatially sparse observation. States are visualized by their potential vorticity field q. Positive values (red) indicate clockwise rotation and negative values (blue) indicate counter-clockwise rotation. The observation y corresponds to a spatial subsampling of factor 16 of the velocity fields (u^1, v^1, u^2, v^2) with small Gaussian noise ($\Sigma_y = 0.01^2 I$). SDA generates trajectories similar to the original one, despite the limited amount of information available in the observation. The three trajectories present slight physically plausible variations, as expected from sampling from a narrow posterior. We observe that the trajectories exhibit less small-scale details than the original one.

Importantly, the three trajectories in Figure 1 were generated concurrently on a single A5000 GPU in 18 minutes and using $14\,\mathrm{GB}$ of memory. In comparison, naively scaling the original architecture of Rozet et al. [16] would have required roughly 10 times more compute and 20 times more memory. With classical data assimilation algorithms, such as variational [5–9] or Monte Carlo [12–15] methods, we estimate that inference would have required the compute equivalent of thousands of simulations, each taking roughly 3 minutes to complete using pyqg [24] on 4 CPU cores. In conclusion, we believe that SDA is ready to be tested in full operational conditions.

Acknowledgments and Disclosure of Funding

References

[1] Colin Cotter et al. "Data Assimilation for a Quasi-Geostrophic Model with Circulation-Preserving Stochastic Transport Noise". In *Journal of Statistical Physics* 179.5-6 (2020), pp. 1186–1221. URL: https://arxiv.org/abs/1910.03574.

[2] Thomas Bolton and Laure Zanna. "Applications of deep learning to ocean data inference and subgrid parameterization". In *Journal of Advances in Modeling Earth Systems* 11.1 (2019), pp. 376–399.

[3] Andrew Ross et al. "Benchmarking of Machine Learning Ocean Subgrid Parameterizations in an Idealized Model". In *Journal of Advances in Modeling Earth Systems* 15.1 (2023).

[4] Victor Mangeleer and Gilles Louppe. "Ocean subgrid parameterization in an idealized model using machine learning". MA thesis. University of Liège, 2023. URL: https://hdl.handle.net/2268.2/18251.

[5] Andrew C. Lorenc. "Analysis methods for numerical weather prediction". In *Quarterly Journal of the Royal Meteorological Society* 112.474 (1986), pp. 1177–1194.

[6] François-Xavier Le Dimet and Olivier Talagrand. "Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects". In *Tellus A: Dynamic Meteorology and Oceanography* 38.2 (1986), pp. 97–110.

[7] Yannick Trémolet. "Accounting for an imperfect model in 4D-Var." In *ECMWF Technical Memoranda* 477 (2005), p. 20. URL: https://www.ecmwf.int/node/12845.

[8] Yannick Trémolet. "Model error estimation in 4D-Var". In *ECMWF Technical Memoranda* 520 (2007), p. 19. URL: https://www.ecmwf.int/node/12819.

[9] Mike Fisher et al. "Weak-Constraint and Long-Window 4D-Var". In *ECMWF Technical Memoranda* 655 (2011), p. 47. URL: https://www.ecmwf.int/node/9414.

[10] Alberto Carrassi et al. "Data assimilation in the geosciences: An overview of methods, issues, and perspectives". In *Wiley Interdisciplinary Reviews: Climate Change* 9.5 (2018).

[11] ECMWF. "Part II: Data Assimilation". In *IFS Documentation CY47R1* 2 (2020). URL: https://www.ecmwf.int/node/19746.

[12] Geir Evensen. "Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics". In *Journal of Geophysical Research: Oceans* 99.C5 (1994), pp. 10143–10162.

[13] Jun S. Liu and Rong Chen. "Sequential Monte Carlo methods for dynamic systems". In *Journal of the American statistical association* 93.443 (1998), pp. 1032–1044.

[14] Thomas M. Hamill. "Ensemble-based Atmospheric Data Assimilation". In *Predictability of weather and climate* 124 (2006), p. 156.

[15] Arnaud Doucet et al. "A Tutorial on Particle Filtering and Smoothing". In *Handbook of nonlinear filtering* 12 (2011), pp. 656–705. URL: https://wrap.warwick.ac.uk/37961.

[16] François Rozet and Gilles Louppe. "Score-based Data Assimilation". In *Advances in Neural Information Processing Systems* (2023). URL: https://arxiv.org/abs/2306.10574.

[17] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022. URL: https://arxiv.org/abs/2112.10752.

[18] Aditya Ramesh et al. "Hierarchical text-conditional image generation with clip latents". In (2022). URL: https://arxiv.org/abs/2204.06125.

[19] Chitwan Saharia et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In *Advances in Neural Information Processing Systems*. 2022. URL: https://openreview.net/forum?id=08Yk-n5l2Al.

[20] Jonathan Ho et al. "Video Diffusion Models". In *ICLR Workshop on Deep Generative Models for Highly Structured Data*. 2022. URL: https://openreview.net/forum?id=BBelR2NdDZ5.

[21] Karan Goel et al. "It's raw! audio generation with state-space models". In *International Conference on Machine Learning*. PMLR. 2022. URL: https://proceedings.mlr.press/v162/goel22a.

[22] Aapo Hyvärinen. "Estimation of Non-Normalized Statistical Models by Score Matching". In *Journal of Machine Learning Research* 6.24 (2005), pp. 695–709. URL: http://jmlr.org/papers/v6/hyvarinen05a.

[23] Pascal Vincent. "A connection between score matching and denoising autoencoders". In *Neural computation* 23.7 (2011), pp. 1661–1674.

[24] Ryan Abernathey et al. "pyqg: Python Quasigeostrophic Model". 2015. DOI: 110.5281/zenodo.592513.

[25] Olaf Ronneberger et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In *Medical Image Computing and Computer-Assisted Intervention*. 2015, pp. 234–241. URL: https://arxiv.org/abs/1505.04597.

[26] Tianqi Chen et al. "Training Deep Nets with Sublinear Memory Cost". In (2016). URL: https://arxiv.org/abs/1604.06174.

[27] Audrunas Gruslys et al. "Memory-Efficient Backpropagation Through Time". In *Advances in Neural Information Processing Systems* (2016). URL: https://arxiv.org/abs/1606.03401.

[28] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018. URL: https://arxiv.org/abs/1801.04381.

[29] Zhuang Liu et al. "A ConvNet for the 2020s". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11976–11986. URL: https://arxiv.org/abs/2201.03545.

[30] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[31] Zongyi Li et al. "Fourier neural operator for parametric partial differential equations". In *arXiv preprint arXiv:2010.08895* (2020).

[32] Alasdair Tran et al. "Factorized Fourier Neural Operators". In *International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=tmIiMPl4IPa.

[33] Kaiming He et al. "Deep Residual Learning for Image Recognition". In *IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778. URL: https://arxiv.org/abs/1512.03385.

[34] Stefan Elfwing et al. "Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning". In *Neural Networks* 107 (2018), pp. 3–11. URL: https://arxiv.org/abs/1702.03118.

[35] Jimmy Lei Ba et al. "Layer normalization". 2016. URL: https://arxiv.org/abs/1607.06450.

A Experiment details

Resources Experiments were conducted with the help of a cluster of GPUs. In particular, score networks were trained and evaluated concurrently, each on a single GPU with 24 GB of memory.

Score network The local score network is a U-Net [25] with inverted bottleneck residual blocks [28, 29, 33], SiLU [34] activation functions and layer normalization [35]. Residual blocks either operate on the temporal axis or the spatial axes (see Section 4). Gradient checkpointing [26, 27] is applied to the residual blocks to reduce memory consumption. Other hyperparameters are provided in Table 1 and a schematic representation of the architecture is provided in Figure 2.

Table 1: Score network hyperparameters for the 2LQG experiment.

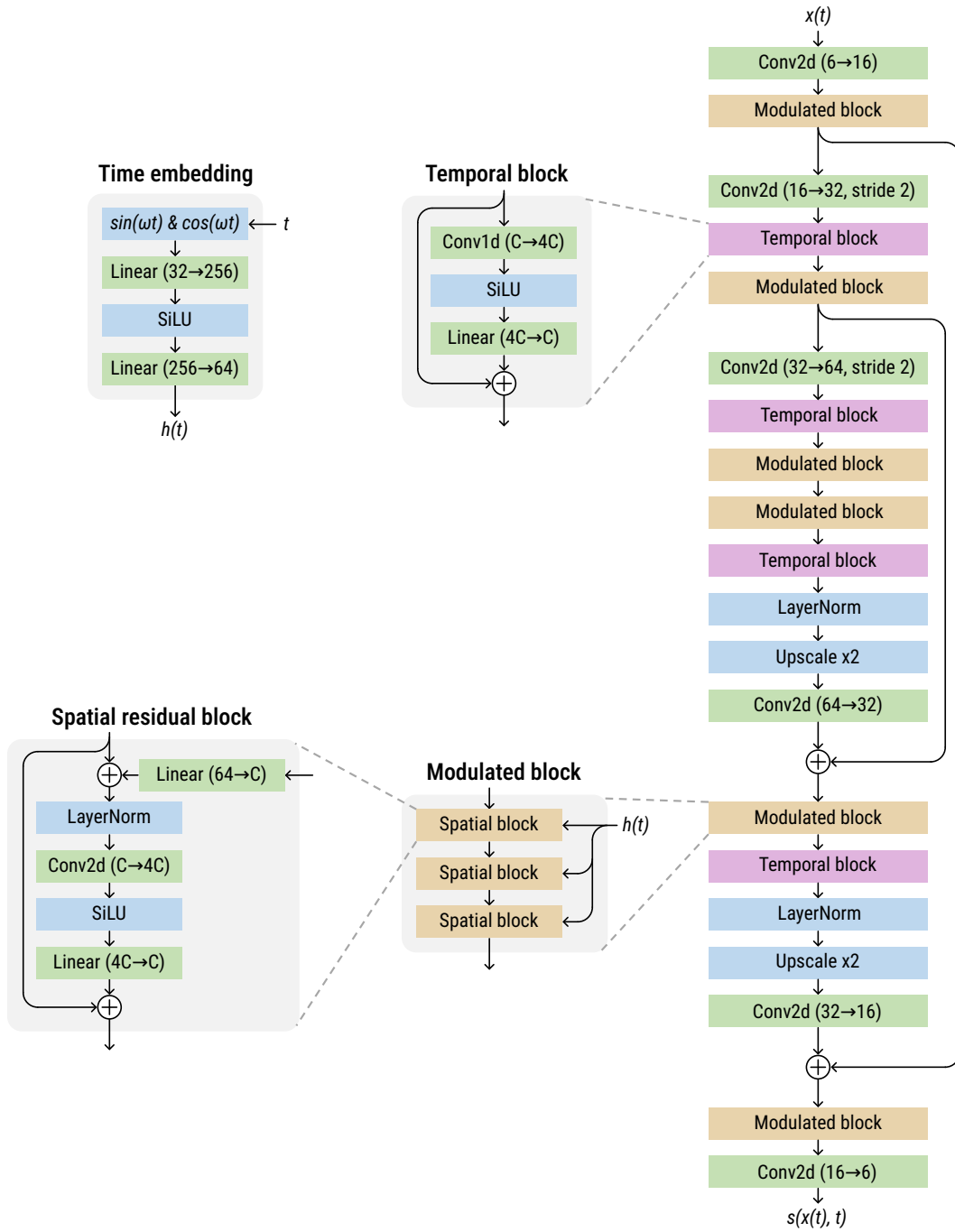|	
Spatial blocks per level	$(3, 3, 3)$
Channels per level	$(16, 32, 64)$
Inverted bottleneck factor	4
Kernel size	5
Padding	circular
Activation	SiLU
Normalization	LayerNorm
Optimizer	AdamW
Weight decay	10^{-3}
Learning rate	2×10^{-4}
Scheduler	linear
Epochs	1024
Batch size	4

Figure 2: Schematic representation of the score network architecture. All spatial and temporal blocks are gradient checkpointed to reduce memory consumption.
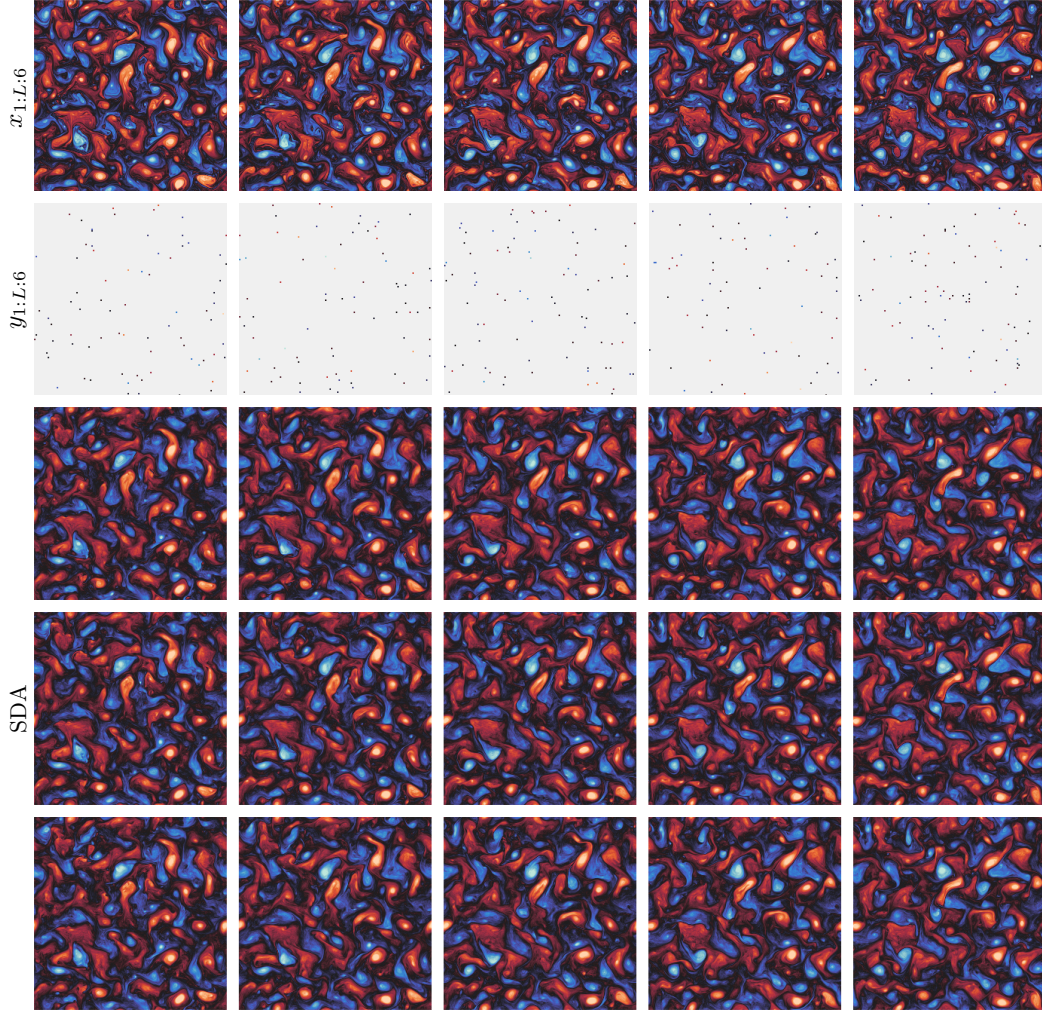
B Assimilation examples

Figure 3: Example of sampled trajectories for a spatially sparse observation. The observation y corresponds to a random (uniform) sampling of ± 80 bins of the velocity fields (u^1, v^1, u^2, v^2) with medium Gaussian noise ($\Sigma_y = 0.1^2 I$). SDA generates trajectories similar to the original one, despite the limited amount of information available in the observation. The three trajectories present slight physically plausible variations, as expected from sampling from a narrow posterior. We observe that the trajectories exhibit less small-scale details than the original one.