RELIABILITY OF PROBABILISTIC EMULATION OF PHYSICAL SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Two dominant approaches have emerged for generating probabilistic forecasts of physical systems: generative models, such as diffusion or flow matching; and ensembles of deterministic models with stochasticity injected, trained using the continuous ranked probability score (CRPS) loss. While both approaches have demonstrated strong predictive accuracy, the reliability of their uncertainties has not been systematically assessed. We address this gap by developing a framework to evaluate both approaches across diverse 2D spatiotemporal physical systems, under matched model size and computational budget. We assess the reliability of probabilistic emulation by inspecting the empirical coverage of predictive intervals, while also considering accuracy and computational efficiency metrics. CRPS-trained ensembles typically achieve more reliable uncertainties on both single-step prediction and autoregressive rollouts, demonstrating better coverage than the standard alternative of training generative models in a latent space. Moreover, the CRPS approach offers significantly faster inference. When generative models are trained in ambient rather than a compressed latent space, which is often infeasible for high-dimensional problems, they exhibit comparable coverage to CRPS-trained ensembles, though with substantially larger inference latency. In contrast, when CRPS-trained ensembles are trained in latent space they do not show a marked degradation in coverage with respect to ambient space. Both generative models and CRPS-trained ensembles demonstrate good predictive accuracy. To facilitate future research and application, we release `AutoCast`, a modular framework implementing both generative models and CRPS-trained ensembles, alongside `AutoSim`, a flexible dataset generation package for rapid prototyping.

1 INTRODUCTION

The modelling of physical systems underpins discovery across diverse domains from climate to materials science. While computational simulations are often used for modelling these systems, their computational expense bottlenecks their practical deployment. Data-driven emulators offer a fast and accurate alternative. Current benchmarks predominantly focus on deterministic training and point-prediction accuracy (Ohana et al., 2024; Takamoto et al., 2022; Gupta & Brandstetter, 2023; Wyder et al., 2025). However, this overlooks a critical requirement for reliable uncertainty quantification (UQ) estimates in real-world deployment. Emulators must contend with both *epistemic uncertainty* (arising from the model's incomplete representation of the underlying physics and limited training data) and *aleatoric uncertainty* (arising from inherent stochasticity in the system or noisy observations). Consequently, reliable probabilistic forecasts can be more valuable than marginal improvements in mean prediction error, especially when they inform risk assessment and planning.

Two dominant probabilistic modelling approaches have emerged for physical system emulation: (i) generative models; and (ii) ensembles of deterministic models with stochasticity injected through noise, trained using continuous ranked probability score (CRPS) loss. Generative models, such as denoising diffusion (Ho et al., 2020) and flow matching (FM) (Lipman et al., 2023), have demonstrated strong performance, especially for long-term rollouts, but are computationally expensive at inference. To address this, recent work has shown that training in lower-resolution latent spaces can improve efficiency without degrading performance (Rozet et al., 2025a). Conversely, CRPS-trained ensembles offer a computationally efficient alternative, even when forecasting directly in ambient space, that has gained significant traction, particularly in weather forecasting applications (Lang et al., 2024;

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Alet et al., 2025; Bonev et al., 2025). Both approaches implement losses that target learning the full conditional forecast distribution, with generative models doing so implicitly by targeting a pushforward operator from noise to the target distribution, while CRPS-trained models directly optimise a proper scoring rule through the CRPS loss (Gneiting & Raftery, 2007).

While both approaches have demonstrated strong predictive accuracy, the reliability of their predictive intervals has not been systematically assessed or compared. In practice, miscalibrated uncertainties can result in catastrophic failure of downstream tasks, e.g. suboptimal decision making. This motivates a focus on empirical coverage, the empirical proportion $(1-\hat{\alpha})$ of cases in which a $(1-\alpha)$ prediction interval contains the underlying true value. Ideally, the empirical coverage $(1-\hat{\alpha})$ should closely approximate the nominal coverage $(1-\alpha)$. However, achieving well-calibrated uncertainty in probabilistic emulators remains challenging. Recent empirical evaluations indicate that traditional probabilistic methods, such as quantile regression, can exhibit miscalibration in practice (Gopakumar et al., 2026), motivating the need for rigorous assessment of the probabilistic reliability of generative and CRPS-trained models.

No work has systematically compared generative models and CRPS-trained ensembles using empirical coverage across diverse physical systems under controlled experimental conditions. Because both approaches target the full conditional forecast distribution, one might expect them to yield reliable predictive uncertainties. In practice, however, each approach relies on approximations, so empirical study is needed to determine how reliable their probabilistic forecasts are. We address this gap by assessing UQ reliability via empirical coverage across a number of 2D spatiotemporal physical systems. Our findings highlight promising future research directions to ensure reliable UQ for complex physical emulation, a necessary requirement for real-world deployment.

The remainder of this article is structured as follows. Section 2 introduces `AutoSim`, a lightweight dataset generation package for flexible benchmarking, and the studied 2D spatiotemporal systems. Section 3 presents our methodology, describing the two probabilistic frameworks (generative modelling and CRPS-trained ensembles) alongside `AutoCast`, a modular modelling and benchmarking framework. Numerical experiments and results are outlined in Section 4, comparing probabilistic reliability, accuracy, and computational cost of the two frameworks. Concluding remarks and future research directions motivated by this work are discussed in Section 5.

2 SIMULATIONS

2.1 AUTOSIM

We developed `AutoSim`[1], an open-source Python package for flexible dataset generation, to complement the existing benchmarking ecosystem for physical system emulation.

Existing datasets, such as The Well (Ohana et al., 2024), PDEBench (Takamoto et al., 2022), or BubbleML (Hassan et al., 2023), provide high-quality simulation data. However, their high-resolution nature is not always well suited to rapid experimentation or constrained compute budgets. `AutoSim` addresses this by providing multiple simulators through a consistent API, enabling users to generate datasets at the resolution, simulation horizon and trajectory count suitable to their compute.[2]

We focus on 2D spatiotemporal problems simulated on a regular 64×64 spatial grid with uniform time-step. For this paper, we selected a subset of the simulations available in `AutoSim`, chosen to demonstrate applicability across different dynamical regimes and levels of complexity. We give a short description of each physical system alongside a visualisation (see Figure 1).

Advection–Diffusion (AD) models the evolution of a single vorticity field under nonlinear advection and diffusion. We use it as a simple baseline: it provides a controlled single-channel transport problem that isolates the need to preserve coherent structures while tracking dissipative smoothing.

[1] `https://anonymous.4open.science/r/autosim-427F`

[2] The primary benefit of `AutoSim` is that researchers can generate data from multiple physical systems in a consistent format without having to become familiar with specialised simulation software, making ML benchmarking more accessible. That said, `AutoSim` is not a general-purpose simulation package. Frameworks such as COMSOL (COMSOL AB, 2026), FEniCS (Alnæs et al., 2015), PhiFlow (Holl & Thuerey, 2024), and Lily Pad (Weymouth, 2015) provide more flexible simulation capabilities.

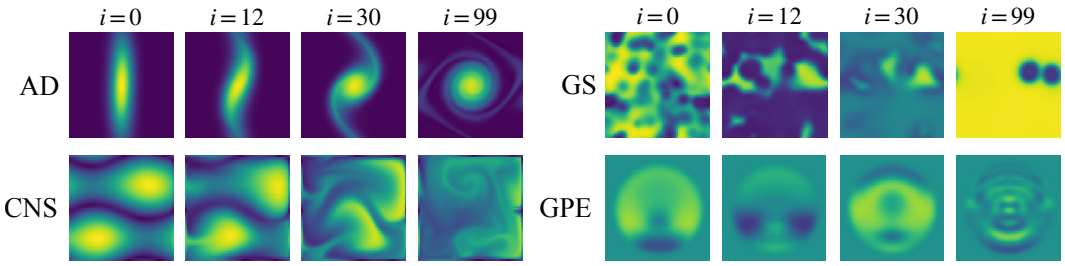


Figure 1: Snapshots at $i = \{0, 12, 30, 99\}$ of the Advection–Diffusion (AD) and Conditioned Navier–Stokes (CNS) datasets (left) and the Gray–Scott (GS) and Gross–Pitaevskii Equation (GPE) datasets (right); within each dataset, columns are ordered left-to-right by increasing index i.

Gray–Scott (GS) is a two-species reaction–diffusion system whose behaviour varies markedly with the feed and kill rates, producing qualitatively different regimes such as spots, worms, spirals, and maze-like patterns. It tests whether models can represent qualitatively different morphologies within a single partial differential equation (PDE) family.

Conditioned Navier–Stokes (CNS) couples passive-scalar transport to an incompressible buoyancy-driven flow. We include it to represent turbulence-like classical fluid behaviour with controllable forcing and boundary-condition variants.

Gross–Pitaevskii Equation (GPE) is a nonlinear quantum-fluid model of a Bose–Einstein condensate driven by a moving laser obstacle, spanning diverse superfluid phenomena including quantum turbulence. It extends the benchmark beyond classical fluids.

For each physical system, we generate multiple deterministic trajectories by varying initial conditions and simulation parameters to capture a diverse range of system dynamics. Specifications of each dataset, including governing equations and numerical methods, are provided in Appendix A.

3 Methodology

This section defines the probabilistic forecasting setup and the two modelling pipelines compared throughout the paper (see Figure 2). We first introduce the windowed single-step prediction and autoregressive rollout tasks, then describe latent space FM and ambient space CRPS-trained ensembles as alternative ways to represent forecast distributions. We then review scoring rules and coverage diagnostics used to assess probabilistic reliability before summarising the `AutoCast` implementation used in the experiments.

3.1 Spatiotemporal forecasting

The goal of spatiotemporal forecasting is to predict a future sequence of states $\boldsymbol{x}^{i+1:i+n}$ given a present sequence $\boldsymbol{x}^{i-m+1:i}$ and any additional conditioning on simulation parameters $\boldsymbol{c}$, which we treat here as constant over the sequence. We write the input window as $\boldsymbol{x}_{\text{in}} = \boldsymbol{x}^{i-m+1:i} \in \mathbb{R}^{m\times W\times H\times C}$ and the output window as $\boldsymbol{x}_{\text{out}} = \boldsymbol{x}^{i+1:i+n} \in \mathbb{R}^{n\times W\times H\times C}$, where m and n are, respectively, the input and output window sizes, W and H the spatial width and height, and C the number of channels. For probabilistic forecasting, we aim to learn the conditional distribution $p(\boldsymbol{x}_{\text{out}} \mid \boldsymbol{x}_{\text{in}}, \boldsymbol{c})$ and train models that represent the approximation $p_\theta(\boldsymbol{x}_{\text{out}}|\boldsymbol{x}_{\text{in}}, \boldsymbol{c})$, parameterised by θ, from which we can draw samples. The input window can either be the ground truth (i.e., a single-step prediction) or the model's previous prediction (autoregressive rollout) (Bonev et al., 2025; Gupta & Brandstetter, 2023). For latent space models, an encoder e_ψ and decoder d_φ, with parameters ψ and φ respectively, are trained to target $\hat{\boldsymbol{x}}^i = d_\varphi(e_\psi(\boldsymbol{x}^i))$, defining latent states $\boldsymbol{z}^i = e_\psi(\boldsymbol{x}^i)$ and latent windows $\boldsymbol{z}_{\text{in}} = \boldsymbol{z}^{i-m+1:i}$ and $\boldsymbol{z}_{\text{out}} = \boldsymbol{z}^{i+1:i+n}$. Autoregressive rollout recursively feeds predictions back as inputs with stride n; with $m = 1$, the first prediction conditions on the ground-truth initial state $\boldsymbol{x}^0$,

$$\hat{\boldsymbol{x}}^{1:n} \sim p_\theta(\cdot \mid \boldsymbol{x}^0, \boldsymbol{c}), \qquad \hat{\boldsymbol{x}}^{i+1:i+n} \sim p_\theta(\cdot \mid \hat{\boldsymbol{x}}^i, \boldsymbol{c}) \quad \text{for } i = n, 2n, \ldots$$

For latent models, the initial window is encoded once, $\hat{\boldsymbol{z}}^{0:m-1} = e_\psi(\boldsymbol{x}^{0:m-1})$, after which the recursion is performed entirely in latent space; predictions are decoded into ambient space with d_φ.

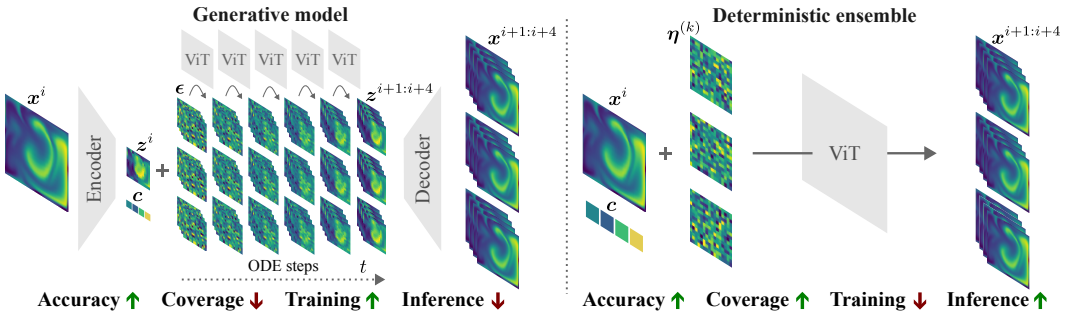


Figure 2: Illustration of the two inference pipelines: generative model and deterministic ensemble trained with CRPS. Both processors predict an output window of $n = 4$ future states given an input window of $m = 1$ state. For illustrative purposes we show an ensemble size of $M = 3$. For generative models, prediction happens in the latent rather than ambient space. Arrows indicate relative advantages and disadvantages. CRPS training is slower per epoch but it yields better coverage and faster inference than the generative model. The two approaches have comparable predictive accuracy.

3.2 Model uncertainty

The simulations are deterministic, the current state and parameters $\boldsymbol{c}$ uniquely determine the future trajectory, so the ideal conditional target is the point mass $p(\boldsymbol{x}_{\text{out}} \mid \boldsymbol{x}_{\text{in}}, \boldsymbol{c}) = \delta(\boldsymbol{x}_{\text{out}} - f_{\text{sim}}(\boldsymbol{x}_{\text{in}}, \boldsymbol{c}))$, where f_{sim} is the simulator map to the output window and δ denotes the Dirac delta. Nevertheless, epistemic uncertainty remains in the learned dynamics due to finite training data and limited model capacity.[3] The trained models output predictive intervals that represent the overall learned uncertainty without distinguishing between epistemic and aleatoric components.

3.3 Generative models

Generative models learn to sample from the conditional forecast distribution by transforming noise from a base distribution, conditioned on the input window and simulation parameters. Diffusion (Ho et al., 2020) and flow matching (FM) (Lipman et al., 2023) are common examples. Iterative generative sampling can be computationally costly for large spatial fields, motivating generation in a learned latent space (Rombach et al., 2022; Rozet et al., 2025a). Recent work has also found latent diffusion models to improve accuracy and rollout stability of physics emulators relative to deterministic alternatives (Rozet et al., 2025a). We focus on FM because it can offer faster sampling than diffusion (Lipman et al., 2023). We apply FM in latent space: the encoder e_ψ is applied to each frame of $\boldsymbol{x}_{\text{in}}$ and $\boldsymbol{x}_{\text{out}}$ to form latent windows $\boldsymbol{z}_{\text{in}}$ and $\boldsymbol{z}_{\text{out}}$, samples are generated from $p_\theta(\boldsymbol{z}_{\text{out}} \mid \boldsymbol{z}_{\text{in}}, \boldsymbol{c})$, and the decoder d_φ is applied to each predicted latent frame to obtain $\hat{\boldsymbol{x}}_{\text{out}}$.

FM defines a conditional target probability path from base noise $p_0 = \mathcal{N}(0, \boldsymbol{I})$ to the latent forecast distribution $p_1 = p(\boldsymbol{z}_{\text{out}} \mid \boldsymbol{z}_{\text{in}}, \boldsymbol{c})$. At sampling time, the learned flow map $\phi_{\theta,t}$ converts a noise sample $\boldsymbol{\epsilon} \sim p_0$ to $\boldsymbol{z}_t = \phi_{\theta,t}(\boldsymbol{\epsilon})$, with model pushforward $\hat{p}_t = [\phi_{\theta,t}]_{\#}\, p_0$, and is represented through a velocity field v_θ via the ordinary differential equation (ODE) form

$$\frac{d\boldsymbol{z}_t}{dt} = v_\theta(\boldsymbol{z}_t, t, \boldsymbol{z}_{\text{in}}, \boldsymbol{c}), \qquad \boldsymbol{z}_1 = \boldsymbol{\epsilon} + \int_0^1 v_\theta(\boldsymbol{z}_t, t, \boldsymbol{z}_{\text{in}}, \boldsymbol{c})\, dt.$$

Following Lipman et al. (2023), for the linear interpolation path, $\boldsymbol{z}_t = (1-t)\boldsymbol{\epsilon} + t\boldsymbol{z}_{\text{out}}$ and $u_t(\boldsymbol{z}_t \mid \boldsymbol{\epsilon}, \boldsymbol{z}_{\text{out}}) = d\boldsymbol{z}_t/dt = \boldsymbol{z}_{\text{out}} - \boldsymbol{\epsilon}$, the velocity field is trained with loss

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,(\boldsymbol{z}_{\text{in}},\boldsymbol{c},\boldsymbol{z}_{\text{out}}),\boldsymbol{\epsilon}} \left[\left\| v_\theta(\boldsymbol{z}_t, t, \boldsymbol{z}_{\text{in}}, \boldsymbol{c}) - u_t(\boldsymbol{z}_t \mid \boldsymbol{\epsilon}, \boldsymbol{z}_{\text{out}}) \right\|^2 \right].$$

[3] For the GPE dataset, absolute simulation time, and therefore the phase ωt of the sinusoidal laser motion (Eq. (7)), is not supplied to the emulator. To the extent that this phase cannot be inferred from the observed state, some residual aleatoric uncertainty might be present.

3.4 DETERMINISTIC ENSEMBLES WITH NOISE INJECTION

Efficient deterministic-backbone ensembles are widely used for probabilistic weather forecasting (Lang et al., 2024; Alet et al., 2025; Bonev et al., 2025). Deterministic ensembles produce probabilistic forecasts by drawing independent noise variables $\boldsymbol{\eta}^{(k)} \in \mathbb{R}^{d_\eta}$ with $\boldsymbol{\eta}^{(k)} \sim \mathcal{N}(0, \boldsymbol{I})$, where $d_\eta = n_{\text{noise_channels}}$, and evaluating $\hat{\boldsymbol{x}}^{(k)}_{\text{out}} = f_\rho(\boldsymbol{x}_{\text{in}}, \boldsymbol{c}, \boldsymbol{\eta}^{(k)})$ for processor model f_ρ with parameters ρ, where ensemble members are trained jointly using a CRPS-family loss. $\boldsymbol{\eta}$ is typically injected into the deterministic backbone using a conditional normalisation strategy (Lang et al., 2024; Alet et al., 2025; Diaconu et al., 2026a) (further implementation details are provided in Appendix B).

A CRPS-family ensemble objective compares the forecast distribution with the realised outcome. For deterministic forecasts, CRPS reduces to mean absolute error; in the ensemble form below, it becomes the error term minus a pairwise member-distance term, encouraging forecasts that are both accurate and appropriately dispersed. For an ensemble $\{\hat{\boldsymbol{x}}^{(k)}_{\text{out}}\}_{k=1}^{M}$ and target $\boldsymbol{x}_{\text{out}}$, with $\|\cdot\|_1$ denoting mean absolute difference over output dimensions, the loss is given by

$$\mathcal{L}_\lambda = \frac{1}{M}\sum_{j=1}^{M}\left\|\hat{\boldsymbol{x}}^{(j)}_{\text{out}} - \boldsymbol{x}_{\text{out}}\right\|_1 - \lambda \sum_{j=1}^{M}\sum_{k=1}^{M}\left\|\hat{\boldsymbol{x}}^{(j)}_{\text{out}} - \hat{\boldsymbol{x}}^{(k)}_{\text{out}}\right\|_1.$$

In practice, finite-ensemble variants such as fair CRPS (fCRPS) (Alet et al., 2025) or almost fair CRPS (αfCRPS) (Lang et al., 2024), with $\alpha = 1$ recovering fCRPS, are typically used. Ensemble models trained this way are a useful alternative to generative models because each ensemble member requires a single forward pass, making inference much faster. In $\mathcal{L}_\lambda$, the CRPS variants use

$$\lambda_{\text{CRPS}} = \frac{1}{2M^2}, \qquad \lambda_{\text{fCRPS}} = \frac{1}{2M(M-1)}, \qquad \lambda_{\text{afCRPS}_\alpha} = \frac{M-1+\alpha}{2M^2(M-1)}.$$

3.5 PROPER SCORING RULES

Scoring rules are used to evaluate the quality of probabilistic forecasts, assigning a numerical score $S(\hat{p}, \boldsymbol{x})$ to a sample $\boldsymbol{x}$ given predictive distribution $\hat{p}$ (Gneiting & Raftery, 2007). A scoring rule is *proper* if its expected value is minimised when the predictive distribution $\hat{p}$ matches the true underlying data distribution p, i.e.

$$\mathbb{E}_{\boldsymbol{x}\sim p}[S(p, \boldsymbol{x})] \leq \mathbb{E}_{\boldsymbol{x}\sim p}[S(\hat{p}, \boldsymbol{x})].$$

A scoring rule is *strictly proper* if equality holds only for $\hat{p} = p$, i.e., only when the predictive distribution matches the true underlying data distribution.

The CRPS-family losses are proper scoring rules for scalar predictive distributions. Averaged componentwise across time, space, and channels, they are proper for the marginals rather than the joint distribution. In contrast, FM implicitly optimises the predictive distribution toward the conditional data distribution by learning a flow map whose pushforward distribution $[\phi_{\theta,1}]_{\#}\, p_0$ is designed to match it. Since both CRPS-training and FM target the conditional data distribution (componentwise for CRPS, joint for FM), both approaches are theoretically expected to produce reliable learned distributions. In practice, there are approximations and sources of error introduced by each approach. For CRPS-trained models, the loss is computed from a fixed, typically small, number of ensemble members. For FM, discretisation error is introduced in the ODE solver, accumulating drift along the trajectory, and distributions are matched in the latent rather than the ambient space. Given both approaches are theoretically well-motivated, empirical studies, such as presented in this article, are necessary to assess how internal errors and approximations impact the overall reliability of probabilistic forecasts. These findings can, in turn, inform future research aimed at mitigating the limitations of current approaches.

Beyond CRPS, other proper scores can target different aspects of the predictive distribution. Multivariate scores, such as the energy score, can capture joint structure across output dimensions (Gneiting & Raftery, 2007; Pic et al., 2025). Proper scoring rules can also be constructed for cases where forecasts are summarised by intervals: the Winkler interval score,

$$W_\alpha(l, u; y) = (u - l) + \frac{2}{\alpha}(l - y)\mathbf{1}\{y < l\} + \frac{2}{\alpha}(y - u)\mathbf{1}\{y > u\},$$

targets both coverage and interval sharpness by penalising wide intervals and missed coverage, and so directly relates to the empirical-coverage diagnostics introduced next.

3.6 RELIABILITY OF LEARNED DISTRIBUTIONS: COVERAGE TESTING

Probabilistic emulator studies that report UQ quality metrics often use consistency metrics such as spread-skill ratio (SSR), which compares ensemble spread to prediction errors (Rozet et al., 2025a; Alet et al., 2025). Self-consistency as measured by SSR$= 1$ is important but not a sufficient condition for reliable forecasts (Diaconu et al., 2026a; Rozet et al., 2025a). SSR is limited by the fact that it does not provide a direct measure of whether predictive intervals contain the true value across quantiles of the distribution. We focus here instead on empirical coverage, which tests whether the true output window falls inside prediction intervals at the expected rate. For an ensemble $\{\hat{\boldsymbol{x}}_{\text{out}}^{(k)}\}_{k=1}^{M}$, let $l_{j,r}^{(\alpha)}$ and $u_{j,r}^{(\alpha)}$ be the $\alpha/2$ and $1-\alpha/2$ quantiles for output component r of test case j, and let $(\boldsymbol{x}_{\text{out}}^{(j)})_r$ be the corresponding truth; then empirical coverage is computed by

$$1-\hat{\alpha} = \frac{1}{NR}\sum_{j=1}^{N}\sum_{r=1}^{R}\mathbf{1}\Big\{(\boldsymbol{x}_{\text{out}}^{(j)})_r \in \big[l_{j,r}^{(\alpha)}, u_{j,r}^{(\alpha)}\big]\Big\}.$$

Here r indexes all output time, spatial, and channel dimensions of $\boldsymbol{x}_{\text{out}}$. For a well-calibrated model, $1-\hat{\alpha}$ should approximate $1-\alpha$ across coverage levels.

3.7 AUTOCAST

As part of this work, we release `AutoCast`[4], a flexible Python package for spatiotemporal forecasting that supports training in both ambient and latent space through an encoder–processor–decoder architecture. Its modular design makes it easy to swap out different processors or autoencoders within the same training pipeline. The package implements state-of-the-art (SotA) generative models (diffusion, FM) alongside popular deterministic architectures (vision transformer (ViT): Dosovitskiy et al. 2021, U-Net: Ronneberger et al. 2015, Fourier neural operators: Li et al. 2021). `AutoCast` also enables training ensembles with CRPS loss using a variety of noise injection strategies, e.g. adaptive layer normalisation (as presented here) or concatenating noise on the inputs.

`AutoCast` provides out-of-the-box support for `AutoSim` data, and we additionally target interoperability with `The Well` dataset (Ohana et al., 2024) given its scale, breadth, and popularity: `AutoCast` provides data loaders for both `AutoSim` and `The Well`, making it easy to switch between them in the same training pipeline and go from rapid prototyping to benchmarking models at scale. Beyond these defaults, `AutoCast` is extendable to alternative approaches and systems through common interfaces.

4 NUMERICAL EXPERIMENTS AND RESULTS

We focus on assessing and comparing the two training pipelines; namely, generative modelling and deterministic ensembles trained with CRPS loss. We consider the reliability of probabilistic forecasts, as assessed by coverage testing, forecast accuracy and computational cost. First we describe the experimental setup, followed by the main comparison, before presenting ablation studies.

4.1 EXPERIMENTAL SETUP

All training is carried out in `AutoCast`. All models are trained to predict an output window of $n = 4$ future steps from a ground-truth input window of $m = 1$, conditioned on any simulator parameters (i.e., single-step prediction). During validation and testing the models are additionally evaluated on autoregressive rollouts for 100 steps from an initial state and simulator parameters, which lets us measure how accuracy and uncertainty calibration evolve with lead time. For table summaries, we report coverage MAE as the mean absolute difference between empirical and nominal coverage over the evaluated coverage levels.

We consider the two pipelines outlined in Section 3.3 and Section 3.4 (Figure 2): (i) a generative pipeline in latent space, where an autoencoder is trained first and then frozen, with the FM processor trained in latent space; and (ii) an ambient space pipeline with an ensemble processor trained with

[4] `https://anonymous.4open.science/r/autocast-56B8`

Table 1: Single-step results across datasets: predictive accuracy (VRMSE), coverage error (MAE), probabilistic accuracy (CRPS), spread–skill calibration (SSR), inference latency, and epoch time.

Dataset	Model	VRMSE ↓	Coverage MAE ↓	CRPS ↓	SSR → 1	Latency (ms/sample) ↓	Epoch time (s) ↓
AD	CRPS	$\mathbf{1.8 \times 10^{-4}}$	0.04	$\mathbf{8.8 \times 10^{-5}}$	**1.07**	**15**	175
	FM	2.2×10^{-4}	**0.02**	1.3×10^{-4}	1.31	672	**24**
CNS	CRPS	$\mathbf{5.6 \times 10^{-3}}$	**0.12**	$\mathbf{2.7 \times 10^{-3}}$	**1.16**	**15**	178
	FM	8.7×10^{-3}	0.28	3.9×10^{-3}	0.65	715	**24**
GS	CRPS	$\mathbf{1.1 \times 10^{-3}}$	0.20	5.2×10^{-4}	**0.67**	**14**	213
	FM	1.3×10^{-3}	**0.18**	$\mathbf{5.1 \times 10^{-4}}$	0.57	687	**29**
GPE	CRPS	$\mathbf{2.8 \times 10^{-2}}$	**0.04**	$\mathbf{9.7 \times 10^{-3}}$	**0.92**	**15**	181
	FM	3.2×10^{-2}	0.24	1.1×10^{-2}	0.44	647	**24**

the αfCRPS loss. The CRPS model uses $M = 8$ ensemble members and FM sampling uses 50 Euler steps. To keep the comparison fair, we fix the processor budget: both processors use a ViT backbone, are constrained to roughly 80M parameters and are trained under the same maximum wall-clock budget of 4 GPUs for 23.5 hours (94 GPU-hours). All experiments were run on NVIDIA GH200 Grace Hopper GPUs on the Isambard-AI facility (McIntosh-Smith et al., 2024).

CRPS is only strictly proper for scalar marginals, and its componentwise and finite-ensemble approximations have known limitations on multivariate spatial forecasts that have motivated both auxiliary spectral losses (Bonev et al., 2025) and finite-ensemble corrections such as αfCRPS (Lang et al., 2024), which we adopt for CRPS training. We do not add further regularisation, but observed in some runs that interval coverage degraded while validation loss kept falling, consistent with drift toward an MAE-like regime. We therefore select CRPS checkpoints using the validation Winkler score averaged across several α levels, which rewards both coverage and sharpness (Section 3.5; ablation in Appendix F.3). For FM, we report the final checkpoint from the fixed-budget training run since early stopping was not advantageous.

See Appendix B for full details of the two model architectures, training pipelines and hyperparameters.

4.2 Comparison of generative and deterministic ensemble approaches

See Appendix C (Figure 5) for example rollout snapshots (mean and ensemble spread).

Overall performance. Table 1 shows that the CRPS-trained ensemble is generally stronger on single-step accuracy (variance scaled root mean squared error (VRMSE), CRPS) than the generative model pipeline, although the two methods perform comparably. Coverage MAE is comparable on AD and GS, and better for CRPS on CNS and GPE. The SSR values also consistently favour CRPS training, indicating that its predictive spread is more stable relative to its errors. The compute trade-off goes in opposite directions for training and inference. At inference time, CRPS training is substantially cheaper because each ensemble sample requires only a single forward pass (the factor difference is approximately the number of ODE steps used in FM). By contrast, the generative model FM trains faster per epoch because CRPS training must evaluate multiple ensemble members, so the observed gap is broadly consistent with the ensemble-size overhead.

See Appendix D for performance on the autoregressive rollout task, showing the two training approaches remain broadly comparable in predictive accuracy, consistent with the overall metrics.

UQ reliability on single-step prediction. We first assess calibration, i.e. probabilistic reliability, by coverage testing for the single-step prediction task. For each model and dataset, we sweep nominal coverage levels $(1 - \alpha)$ from 0.05 to 0.95 and plot empirical coverage $1 - \hat{\alpha}$ against the nominal coverage $1 - \alpha$. Perfectly calibrated forecasts would lie on the diagonal. Figure 3 shows strong single-step calibration on AD for both approaches, with curves close to the ideal diagonal. The difference between the methods is more visible on the harder datasets. On GPE, CRPS training remains reasonably well calibrated whereas the generative model clearly underestimates uncertainty.

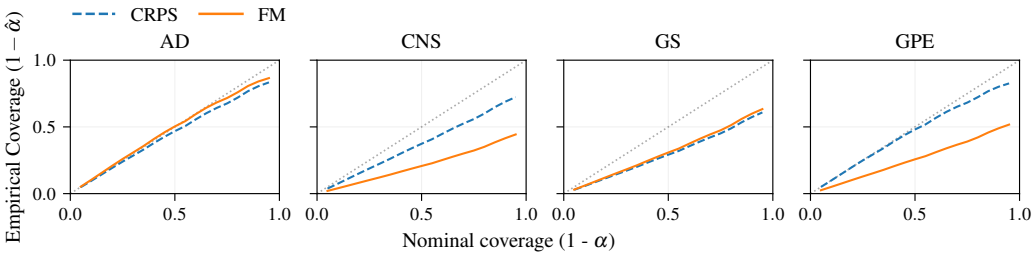


Figure 3: Coverage on the single-step prediction task by dataset (AD, CNS, GS, GPE) and model (CRPS, FM). Ideal coverage is denoted by a diagonal grey dotted line. The CRPS-trained ensembles perform better than, or on par with, the generative models.

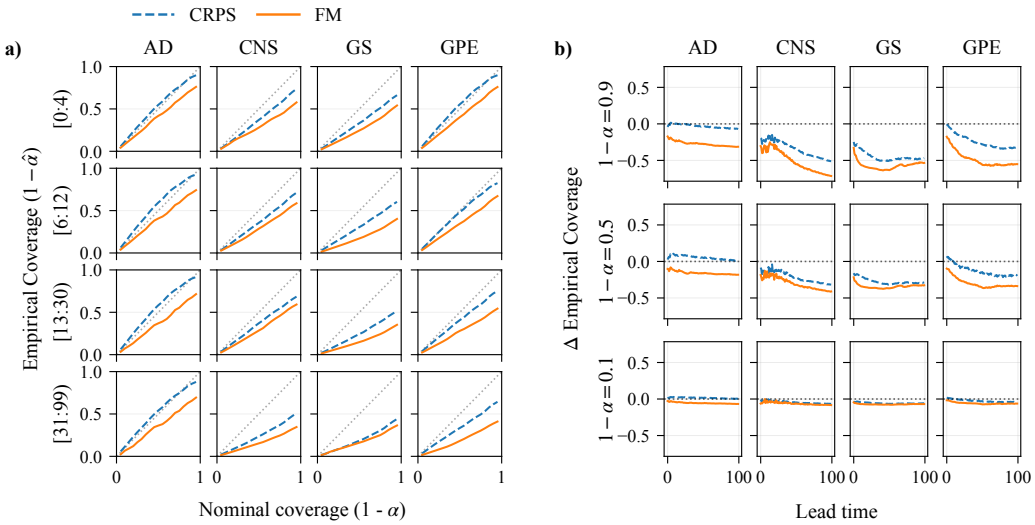


Figure 4: UQ reliability over lead time by dataset (AD, CNS, GS, GPE) and model (CRPS, FM). (a) Coverage for different lead time windows (averaging across rollout windows: $[0{:}4)$, $[6{:}12)$, $[13{:}30)$, and $[31{:}99)$). (b) Δ empirical coverage (observed minus nominal) by lead time for $1-\alpha = 0.9$ (top row), $1-\alpha = 0.5$ (middle row) and $1-\alpha = 0.1$ (bottom row). Ideal coverage, given by zero difference, is denoted by a horizontal grey dotted line. Overall, CRPS training is more reliable than the generative model approach across lead times.

On CNS, both methods under-cover, but CRPS stays noticeably closer to the target calibration line. GS is difficult for both model classes, with both similarly under-covering.

UQ reliability by lead time. We next turn to UQ reliability over 100-step autoregressive rollouts. This task repeatedly feeds predictions back into the model and measures how calibration evolves over longer horizons. We report coverage averaged within rollout windows $[0{:}4)$, $[6{:}12)$, $[13{:}30)$, and $[31{:}99)$ and also show Δ empirical coverage, the signed coverage error $(1-\hat{\alpha}) - (1-\alpha)$, as a function of lead time. Negative values indicate under-coverage and positive values indicate over-coverage. The Δ empirical coverage panels share a common vertical scale within each figure across nominal levels and datasets. Figure 4 shows that calibration typically degrades with lead time for both model classes. The main qualitative pattern from the single-step results persists, but the advantage of CRPS training over the generative model approach becomes more consistent across datasets when we consider long horizon forecasts. See Appendix E for standard errors computed over independent test trajectories.

4.3 Ablations and other experiments

Below we highlight the main findings from our ablations and additional experiments with further detail reported in Appendix F.

Ambient vs. latent space training: CRPS training in latent space does not worsen coverage, while generative training in ambient space exhibits comparable coverage to CRPS training (Appendix F.1). Increasing autoencoder compression degrades generative training performance (Appendix F.2).

CRPS: Using the Winkler score as a checkpoint selection criterion leads to comparable or better coverage than using the validation loss (Appendix F.3). Performance remained consistent when using fCRPS loss (Appendix F.4), varying ensemble size (Appendix F.5) or reducing the number of noise channels (Appendix F.6). Using backbone modulation instead of input channel concatenation to inject global simulation constants into the model worsens coverage over lead time (Appendix F.7).

Flow matching: Fewer ODE steps speed up inference with little change in VRMSE but can reduce coverage (Appendix F.8). Exponential moving average (EMA) weights do not consistently improve coverage or rollout accuracy (Appendix F.9).

Alternative processor architectures: Diffusion exhibits comparable coverage to FM while CRPS-trained ensembles with a U-Net backbone exhibit worse rollout coverage on CNS and GS relative to ViT (Appendix F.10). FNOs give overall less consistent rollout performance (Appendix F.11).

Training variability: Retraining with an independent random seed (Appendix F.12) or newly generated training data (Appendix F.13) shows some performance variability, most prominently for coverage on AD, but the relative performance trends and overall conclusions remain unchanged.

MC dropout: Using Monte Carlo (MC) dropout instead of conditional layer normalisation (CLN) within CRPS training yields comparable or worse performance (Appendix F.14). MC dropout trained with L_2-regularised MSE loss is less competitive than both main approaches (Appendix F.15).

Conformal prediction: Since both approaches still retained miscalibrated UQ, we show in Appendix F.16 that calibration can be improved *post-hoc* using conformal prediction.

5 Conclusions

CRPS-trained ensembles generally deliver more reliable uncertainty estimates than latent space generative models. While ambient space generative modelling is often too expensive to be practical for high-dimensional problems, we found it can recover coverage comparable to CRPS-trained ensembles without loss of accuracy. In contrast, CRPS-trained ensembles had reliable coverage when trained in both ambient or latent space.

Our findings suggest interesting directions for future research. Since generative models are commonly trained in latent space, a natural research direction is to develop methods to improve the latent representation itself so that it preserves calibration as well as reconstruction accuracy. In particular, the current autoencoder is trained to reconstruct individual spatial fields, whereas a distributional objective would treat the decoder as a pushforward of the full conditional forecast distribution, $[d_\varphi]_\# p(\boldsymbol{z}_{\text{out}} \mid \boldsymbol{z}_{\text{in}}, \boldsymbol{c}) = p(\boldsymbol{x}_{\text{out}} \mid \boldsymbol{x}_{\text{in}}, \boldsymbol{c})$, and incorporate that distributional target directly into autoencoder training. We intend to explore this in future work.

Our study is limited to modest resolution and to a small but diverse set of physical problems due to computational constraints (a 10,000 GH200 GPU-hour allocation). While we expect the main conclusions to persist at high resolution and across more diverse problems, additional experiments are required to probe this further.

To facilitate further experimentation, we release two software packages. `AutoSim` provides a lightweight data-generation workflow for controlled PDE experiments, making it easy to generate datasets at different resolutions and simulator parameters. `AutoCast` provides the corresponding modelling framework, with support for both generative and deterministic processors, ambient and latent space training, and ensemble training with CRPS-family objectives. `AutoCast` is also interoperable with `The Well` (Ohana et al., 2024), so models prototyped on smaller controlled simulations can be carried over to larger, diverse benchmark data with the same training pipeline.

AI USE STATEMENT

AI tools assisted with the implementation of the `AutoSim`, `AutoCast` and `AutoUQ` software packages. In this way, AI contributed to research execution and synthetic data generation. We also used AI to suggest potentially relevant work based on references we had already collected, to write plotting code, to format references, and to assist with drafting and editing the manuscript. AI-assisted code was tested with unit tests and continuous integration and results were cross-checked by multiple authors. We take responsibility of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All code used in this work is released: `AutoSim` generates the four datasets, `AutoCast` implements the models, training and evaluation, and `AutoUQ` (Appendix F.16) implements the conformal prediction for calibration. Appendix A specifies each dataset's governing equations, numerical methods, parameter ranges and train/validation/test splits. Section 4.1 and Appendix B give the architectures, hyperparameters, training budgets and checkpoint-selection criteria, and Sections 3.5 and 3.6 define the evaluation metrics. Appendices E, F.12 and F.13 quantify variability due to the finite test set, the training seed and the training data.

REFERENCES

Ferran Alet, Ilan Price, Andrew El-Kadi, Dominic Masters, Stratis Markou, Tom R. Andersson, Jacklynn Stott, Remi Lam, Matthew Willson, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Skillful joint probabilistic weather forecasting from marginals, 2025. URL https://arxiv.org/abs/2506.10772.

Martin S. Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. The FEniCS project version 1.5. *Archive of Numerical Software*, 3(100):9–23, 2015. doi: 10.11588/ans.2015.100.20553.

Anastasios N Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2021. URL https://arxiv.org/abs/2107.07511.

Anastasios N. Angelopoulos, Rina Foygel Barber, and Stephen Bates. Theoretical foundations of conformal prediction, 2026. URL https://arxiv.org/abs/2411.11824.

Weizhu Bao, Qiang Du, and Yanzhi Zhang. Dynamics of rotating Bose–Einstein condensates and its efficient and accurate numerical computation. *SIAM Journal on Applied Mathematics*, 66(3): 758–786, January 2006. ISSN 1095-712X. doi: 10.1137/050629392. URL http://dx.doi.org/10.1137/050629392.

Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical Fourier neural operators: Learning stable dynamics on the sphere. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2806–2823. PMLR, 2023. URL https://proceedings.mlr.press/v202/bonev23a.html.

Boris Bonev, Thorsten Kurth, Ankur Mahesh, Mauro Bisson, Jean Kossaifi, Karthik Kashinath, Anima Anandkumar, William D. Collins, Michael S. Pritchard, and Alexander Keller. FourCastNet 3: A geometric approach to probabilistic machine-learning weather forecasting at scale, 2025. URL https://arxiv.org/abs/2507.12144.

COMSOL AB. COMSOL Multiphysics, 2026. URL https://www.comsol.com/comsol-multiphysics. Accessed September 2026.

Cristiana Diaconu, Miles Cranmer, Richard E. Turner, Tanya Marwah, and Payel Mukhopadhyay. Probabilistic retrofitting of learned simulators. In *AI&PDE: ICLR 2026 Workshop on AI and Partial Differential Equations*, 2026a. URL https://openreview.net/forum?id=MUuOz8LtNb.

Cristiana Diaconu, Jonas Scholz, Aliaksandra Shysheya, Stratis Markou, Payel Mukhopadhyay, Miles Cranmer, and Richard E. Turner. Otter Weather: Skillful and computationally efficient medium-range weather forecasting, 2026b. URL https://arxiv.org/abs/2606.26421.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059. PMLR, 2016. URL https://proceedings.mlr.press/v48/gal16.html.

Sagar Garg, Stephan Rasp, and Nils Thuerey. WeatherBench Probability: A benchmark dataset for probabilistic medium-range weather forecasting along with deep learning baseline models, 2022. URL https://arxiv.org/abs/2205.00865.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

Vignesh Gopakumar, Ander Gray, Joel Oskarsson, Lorenzo Zanisi, Daniel Giles, Matt J Kusner, Stanislas Pamela, and Marc Peter Deisenroth. Uncertainty quantification of surrogate models using conformal prediction. *Machine Learning: Science and Technology*, 7(1):015025, February 2026. doi: 10.1088/2632-2153/ae2e7b. URL https://doi.org/10.1088/2632-2153/ae2e7b.

Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized PDE modeling. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=dPSTDbGtBY.

Sheikh Md Shakeel Hassan, Arthur Feeney, Akash Dhruv, Jihoon Kim, Youngjoon Suh, Jaiyoung Ryu, Yoonjin Won, and Aparna Chandramowlishwaran. BubbleML: A multi-physics dataset and benchmarks for machine learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=0Wmglu8zak.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Philipp Holl and Nils Thuerey. Φ_{Flow} (PhiFlow): Differentiable simulations for PyTorch, TensorFlow and Jax. In *International Conference on Machine Learning*. PMLR, 2024.

Woo Jin Kwon, Joon Hyun Kim, Sang Won Seo, and Y. Shin. Observation of von Kármán vortex street in an atomic superfluid gas. *Physical Review Letters*, 117(24), December 2016. ISSN 1079-7114. doi: 10.1103/physrevlett.117.245301. URL http://dx.doi.org/10.1103/PhysRevLett.117.245301.

Simon Lang, Mihai Alexe, Mariana C. A. Clare, Christopher Roberts, Rilwan Adewoyin, Zied Ben Bouallègue, Matthew Chantry, Jesper Dramsch, Peter D. Dueben, Sara Hahner, Pedro Maciel, Ana Prieto-Nemesio, Cathal O'Brien, Florian Pinault, Jan Polster, Baudouin Raoult, Steffen Tietsche, and Martin Leutbecher. AIFS-CRPS: Ensemble forecasting using a model trained with a loss function based on the continuous ranked probability score, 2024. URL https://arxiv.org/abs/2412.15832.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Ziqi Ma, Kamyar Azizzadenesheli, and Anima Anandkumar. Calibrated uncertainty quantification for operator learning via conformal prediction, 2024. URL https://arxiv.org/abs/2402.01960.

Simon McIntosh-Smith, Sadaf R Alam, and Christopher Woods. Isambard-AI: A leadership class supercomputer optimised specifically for artificial intelligence, 2024. URL https://arxiv.org/abs/2410.11199.

Ruben Ohana, Michael McCabe, Lucas Thibaut Meyer, Rudy Morel, Fruzsina Julia Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Stuart B. Dalziel, Drummond Buschman Fielding, Daniel Fortunato, Jared A. Goldberg, Keiya Hirashima, Yan-Fei Jiang, Rich Kerswell, Suryanarayana Maddu, Jonah M. Miller, Payel Mukhopadhyay, Stefan S. Nixon, Jeff Shen, Romain Watteaux, Bruno Régaldo-Saint Blancard, François Rozet, Liam Holden Parker, Miles Cranmer, and Shirley Ho. The Well: a large-scale collection of diverse physics simulations for machine learning. In *The Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=00Sx577BT3.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182, 2023. doi: 10.1109/ICCV51070.2023.00387.

R. Pic, C. Dombry, P. Naveau, and M. Taillardat. Proper scoring rules for multivariate probabilistic forecasts based on aggregation and transformation. *Advances in Statistical Climatology, Meteorology and Oceanography*, 11(1):23–58, 2025. doi: 10.5194/ascmo-11-23-2025. URL https://ascmo.copernicus.org/articles/11/23/2025/.

C. Raman, M. Köhl, R. Onofrio, D. S. Durfee, C. E. Kuklewicz, Z. Hadzibabic, and W. Ketterle. Evidence for a critical velocity in a Bose-Einstein condensed gas. *Physical Review Letters*, 83 (13):2502–2505, September 1999. ISSN 1079-7114. doi: 10.1103/physrevlett.83.2502. URL http://dx.doi.org/10.1103/PhysRevLett.83.2502.

M. T. Reeves, T. P. Billam, B. P. Anderson, and A. S. Bradley. Identifying a superfluid Reynolds number via dynamical similarity. *Physical Review Letters*, 114(15), April 2015. ISSN 1079-7114. doi: 10.1103/physrevlett.114.155302. URL http://dx.doi.org/10.1103/PhysRevLett.114.155302.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28.

François Rozet, Ruben Ohana, Michael McCabe, Gilles Louppe, Francois Lanusse, and Shirley Ho. Lost in latent space: An empirical study of latent diffusion models for physics emulation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=xoNrbfbekM.

François Rozet et al. Azula: Diffusion models in PyTorch. Software, 2025b. URL https://github.com/probabilists/azula. Version 0.7.0.

Salva Rühling Cachay, Duncan Watson-Parris, and Rose Yu. U-Cast: A surprisingly simple and efficient frontier probabilistic AI weather forecaster, 2026. URL https://arxiv.org/abs/2604.09041.

Kazuki Sasaki, Naoya Suzuki, and Hiroki Saito. Bénard–von Kármán vortex street in a Bose-Einstein condensate. *Physical Review Letters*, 104(15), April 2010. ISSN 1079-7114. doi: 10.1103/physrevlett.104.150404. URL http://dx.doi.org/10.1103/PhysRevLett.104.150404.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=dh_MkX0QfrK.

Gabriel D. Weymouth. Lily Pad: Towards real-time interactive computational fluid dynamics, 2015. URL https://arxiv.org/abs/1510.06886.

Philippe Martin Wyder, Judah A Goldfeder, Alexey Yermakov, Yue Zhao, Stefano Riva, Jan P. Williams, David Zoro, Amy Sara Rude, Matteo Tomasetto, Joe Germany, Joseph Bakarji, Georg Maierhofer, Miles Cranmer, and J. Nathan Kutz. Common task framework for a critical evaluation of scientific machine learning algorithms. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=pkFKjmUE3L.

A SIMULATION DETAILS

All datasets were generated using the `AutoSim` package. Each uses a 64×64 spatial resolution with 200 training trajectories per dataset, except Gray–Scott (GS) which has 240 training trajectories to capture all six pattern types equally, and 20 validation and 20 test trajectories (24 of each for GS). Every trajectory has length 321 (initial state plus 320 time-steps) and is generated by randomly sampling simulation parameters from pre-defined ranges.

A.1 ADVECTION–DIFFUSION

We consider an advection–diffusion problem describing a fluid motion in the shallow water limit given by

$$\frac{\partial \omega}{\partial t} + \mu \left(\frac{\partial \psi}{\partial x} \frac{\partial \omega}{\partial y} - \frac{\partial \psi}{\partial y} \frac{\partial \omega}{\partial x} \right) = d \nabla^2 \omega, \tag{1a}$$

$$\nabla^2 \psi = \omega, \tag{1b}$$

defined over a spatial domain $(x, y) \in [-L/2, L/2]^2$ and a time span $t \in [0, T]$, where $L = 10$ and $T = 80$. Here $\omega(x, y, t)$ and $\psi(x, y, t)$ represent the vorticity and stream function, respectively, and $\nabla^2 = \partial_x^2 + \partial_y^2$ is the two-dimensional Laplacian. The parameter d is the diffusion (viscosity-like) coefficient governing the dissipative smoothing of vorticity, while μ is a dimensionless coefficient that scales the nonlinear self-advection.

We assume periodic boundary conditions and a stretched Gaussian function as the initial condition of vorticity:

$$\omega(x, y, 0) = \exp\left(-x^2 - \frac{y^2}{20} \right), \qquad (x, y) \in [-L/2, L/2]^2 . \tag{2}$$

We are interested in approximating the time-dependent vorticity field ω as the parameters (d, μ) vary over the ranges in Table 2. The vorticity equation is integrated with a finite-difference discretisation of the Laplacian and advection terms, while the Poisson equation (1b) for the stream function is solved spectrally via fast Fourier transform (FFT); time stepping is performed with an adaptive RK45 solver and a snapshot stride $\Delta t = 0.25$.

Table 2: Parameter ranges used to sample Advection–Diffusion (AD) trajectories. Each parameter is drawn independently from a uniform distribution over the given interval.

Parameter	Symbol	Range
Diffusion coefficient	d	$[10^{-4}, 10^{-2}]$
Advection strength	μ	$[0.5, 2.0]$

A.2 CONDITIONED NAVIER–STOKES

We consider an incompressible, buoyancy-driven "smoke-flow" system in two dimensions, in which a passive scalar $s(\boldsymbol{r}, t)$ is advected by a velocity field $\boldsymbol{u}(\boldsymbol{r}, t) = (u, v)$ and exerts a vertical buoyancy force whose strength is controlled by a conditioning parameter β. The governing equations are

$$\frac{\partial s}{\partial t} + \boldsymbol{u} \cdot \nabla s = \kappa \, \nabla^2 s, \tag{3a}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \nu \, \nabla^2 \boldsymbol{u} + \big(0, \ \beta \, s\big), \tag{3b}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{3c}$$

where ν is the kinematic viscosity, κ the smoke diffusivity, and p the pressure. The buoyancy coefficient β (named `buoyancy_y` in the simulator) is the conditioning variable: each trajectory is generated at a fixed β drawn from the range below, mirroring the conditioned external-force setup of PDEArena (Gupta & Brandstetter, 2023), in which the velocity field is driven by a body force $(0, f)^\top$ with f varied across trajectories.

The initial smoke field is a smoothed Gaussian random field constructed in Fourier space following the `phiflow Noise` class used by PDEArena: independent complex Gaussian coefficients are scaled by $|\boldsymbol{k}|^{-2\sigma}$ with a low-frequency cut-off (where σ is the `smoothness` exponent), transformed back to real space, standardised, and finally rectified ($s_0 \mapsto |s_0|$) so that the initial scalar is non-negative. The velocity is initialised to zero and is spun up purely by the buoyancy forcing acting on the smoke field.

The simulation is performed on a uniform 64×64 grid over $[0, L]^2$ with $L = 32$ (so $\Delta x = 0.5$), using zero-gradient (Neumann) boundary conditions for the smoke and no-slip walls for the velocity, in a bounded-domain setup inspired by PDEArena (which uses 128×128 at the same L). Time integration uses an operator-splitting scheme: (i) semi-Lagrangian advection of s, u, and v via bilinear back-tracing along the local velocity, (ii) an explicit diffusion update for smoke and viscosity, with a vertical buoyancy term $\beta\, s$ added to the v-equation, and (iii) a spectral Hodge projection onto the divergence-free subspace, which simultaneously yields the pressure p; in the bounded mode the projected velocity has its mean drift removed and is then explicitly zeroed at the four boundary rows/columns to enforce no-slip. The step size is adaptively limited by an advective CFL condition with $\mathrm{CFL} = 0.35$ and the explicit-diffusion stability bound. We record roughly one snapshot per integration step over a total horizon $T \approx 85.87$, yielding $n_t = 321$ frames after discarding the first $\mathrm{skip}_{n_t} = 8$ so that buoyancy-driven structure has emerged before the rollout horizon. This corresponds to PDEArena's 84-time-unit window $[18, 102]$ but at $\sim 6\times$ the temporal resolution (PDEArena uses $\Delta t = 1.5$, 56 frames). The kinematic viscosity is fixed at $\nu = 0.01$ and the smoke diffusivity at $\kappa = 0$, isolating the influence of β and the random initial condition on the resulting flow. Simulator output channels are (s, u, v).

Each trajectory is parameterised by an independent uniform draw from the ranges in Table 3, where σ (`smoothness`) controls the spectral decay of the initial smoke field and η (`noise_scale`) sets its spectral scale.

Table 3: Parameter values and ranges used to sample Conditioned Navier–Stokes (CNS) trajectories. Ranges are sampled independently from a uniform distribution; fixed values are shown as scalars.

Parameter	Symbol	Value/range
Buoyancy coefficient	β	$[0.2, 0.8]$
Initial-field smoothness	σ	6.0
Initial-field noise scale	η	$[8.0, 18.0]$

A.3 GRAY–SCOTT

The GS equations are a set of coupled reaction–diffusion equations describing two chemical species, u and v, whose scalar concentrations vary in space and time:

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - uv^2 + F(1 - u), \tag{4a}$$

$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + uv^2 - (F + k)v. \tag{4b}$$

The parameter F controls the rate at which species u is added to the system and k controls the rate at which species v is removed (referred to as the "feed" and "kill" rates). Qualitatively different patterns emerge in the solutions depending on the two parameters F and k (see Table 4 for the F and k values used in our simulations for each of the six pattern types. These are the same as those used in `The Well`, Ohana et al., 2024).

The two constants D_u and D_v govern the rate of diffusion of each species. All simulations used diffusion coefficients $D_u = 2 \times 10^{-5}$ and $D_v = 1 \times 10^{-5}$.

The simulation domain is the square $[-L/2, L/2]^2$ with $L = 1$, discretised on a uniform 64×64 grid with periodic boundary conditions. The initial condition uses independent sums of randomly placed and weighted Gaussian bumps for the two species (with periodic image copies for consistency with

Table 4: Feed and kill parameters used to simulate each pattern type.

	F	k
Gliders	0.014	0.054
Bubbles	0.098	0.057
Maze	0.029	0.057
Worms	0.058	0.065
Spirals	0.018	0.051
Spots	0.030	0.062

the boundary), each normalised to $[0, 1]$, with $u_0 = 1 - \mathrm{bumps}_u$ and $v_0 = \mathrm{bumps}_v$. The system is integrated with a pseudospectral exponential time-differencing Runge–Kutta scheme of order four (ETDRK4), using FFTs for the linear diffusion operator and 2/3-rule dealiasing for the nonlinear terms. We used a simulation time step $\Delta t = 1.0$ over a horizon $T = 1280$ and recorded a snapshot every 4 steps, yielding 321 frames per trajectory. Trajectories are stratified equally across the six pattern types (40 train, 4 validation, and 4 test trajectories per pattern). Some patterns occasionally produce trajectories in which a channel becomes spatially uniform at one or more frames, which destabilises normalisation-based metrics during training; we therefore reject any trajectory whose per-timestep spatial standard deviation falls below 10^{-2} in either channel at any frame and resample with a fresh initial condition until this criterion is met.

A.4 Gross–Pitaevskii Equation

The Gross–Pitaevskii Equation (GPE) is the mean-field equation for a dilute-gas Bose–Einstein condensate (BEC), describing the dynamics of the macroscopic complex wavefunction $\psi(\boldsymbol{r}, t)$ of ultracold bosons in an external potential. We consider the two-dimensional GPE with a moving obstacle,

$$i\,\frac{\partial \psi}{\partial t} = \left[-\frac{1}{2}\nabla^2 + V_{\mathrm{box}}(\boldsymbol{r}) + V_{\mathrm{obs}}(\boldsymbol{r}, t) + g\,|\psi|^2\right]\psi, \tag{5}$$

in dimensionless harmonic-oscillator units, where g controls the strength of the contact (mean-field) interaction. The condensate is confined in a flat-bottomed Woods–Saxon box trap,

$$V_{\mathrm{box}}(\boldsymbol{r}) = \frac{V_0}{1 + \exp\big(-(|\boldsymbol{r}| - R_w)/a\big)}, \tag{6}$$

with wall height $V_0 = 200$, wall steepness $a = 0.15$, and effective radius $R_w = (\mathrm{box_param})^{-1/4} \approx 4.73$ (corresponding to $\mathrm{box_param} = 0.002$). A focused laser beam, modelled as a Gaussian repulsive obstacle, sweeps sinusoidally back and forth along the y-axis,

$$V_{\mathrm{obs}}(\boldsymbol{r}, t) = U \exp\left(-\frac{x^2 + (y - R\sin(\omega t))^2}{w^2}\right), \tag{7}$$

with peak speed $v_{\max} = R\,\omega$. Above a critical obstacle velocity v_c the superfluid response transitions from dissipationless laminar flow (Raman et al., 1999) to periodic vortex–antivortex pair shedding (a quantum von Kármán street; Sasaki et al., 2010; Kwon et al., 2016) and on to increasingly disordered wakes that follow a superfluid-Reynolds-number scaling (Reeves et al., 2015). The parameter ranges below are chosen so that $v_{\max}$ straddles v_c. Because the obstacle motion is sinusoidal, a single sweep can pass through slower and faster phases of the forcing.

The simulation is performed on a uniform 64×64 periodic Fourier grid over the spatial domain $[-L/2, L/2]^2$ with $L = 10$, with physical confinement provided by the Woods–Saxon box potential. We use a Strang split-step Fourier scheme (Bao et al., 2006) that alternates combined potential and nonlinear half-steps in real space with a kinetic full-step in Fourier space with 2/3-rule dealiasing. Each trajectory consists of two phases: (i) imaginary-time relaxation for 4000 steps at $\Delta t_{\mathrm{imag}} = 0.0025$ from a broad Gaussian seed centred at the origin, which produces a clean vortex-free ground state inside the box; followed by (ii) real-time evolution for $T = 17.55$ time units at $\Delta t = 0.005$, with snapshots recorded every $\Delta t_{\mathrm{snap}} = 0.05$ and the first 30 frames discarded so that the obstacle is well into its first sweep. Simulator output channels are the $\Re(\psi)$ and $\Im(\psi)$ fields, enabling both

density and phase to be captured by the emulator indirectly. $\Re(\psi)$ and $\Im(\psi)$ are jointly normalised so that phase is unaffected.

Each trajectory is parameterised by an independent uniform draw from the ranges in Table 5, chosen so that the resulting trajectories span a range of dynamical complexity, from largely laminar response with linear density excitations, through periodic vortex–antivortex pair shedding, to disordered wakes populated by many vortex cores.

Table 5: Parameter ranges used to sample GPE trajectories. Each parameter is drawn independently from a uniform distribution over the given interval.

Parameter	Symbol	Range
Interaction strength	g	$[80, 200]$
Obstacle strength	U	$[3, 8]$
Obstacle angular frequency	ω	$[0.5, 2.0]$
Obstacle Gaussian $1/e$ radius	w	$[0.8, 2.5]$
Obstacle sweep amplitude	R	$[2.0, 3.0]$
Initial Gaussian width	w_0	$[1.5, 2.0]$

B Model architectures and hyperparameters

In this section we provide further detail on training and model architectures.

Training

We keep the CRPS and FM training as aligned as possible. The nominal batch size is fixed at 1024. For CRPS training, the input batch is replicated across ensemble members, so the effective number of distinct training examples scales as $1024/M$ for ensemble size M; for example, with $M = 4$, the effective batch size is 256. We use a cosine learning-rate schedule decaying to zero over the precomputed number of epochs allowed by the time budget, with learning rate 2×10^{-4} for CRPS training and 10^{-4} for FM.

Autoencoder

For latent space experiments, we follow the deep compression autoencoder design used in Rozet et al. (2025a). Each dataset has its own 49M parameter autoencoder trained for 512 epochs with learning rate 10^{-5}, a cosine schedule, and a variance-scaled mean-squared reconstruction loss. The encoder has three resolution levels with three residual blocks per level, 3×3 convolutions, layer normalisation (LN), and dropout 0.05. Hidden widths follow the standard convention of doubling at each downsample, giving $[128, 256, 512]$ channels at successive levels, which keeps per-level compute balanced and provides more channel capacity at coarser spatial scales. Stride-2 pixel-shuffle downsampling is applied between consecutive levels, so the two downsamples (between levels 1–2 and 2–3) reduce the spatial resolution by a factor of 4 and take the input from 64×64 at level 1 to 16×16 at level 3. The decoder mirrors this with widths $[512, 256, 128]$ and pixel-shuffle upsampling; the latent bottleneck is softly clipped with scale 5 to avoid unbounded latent amplitudes. Overall the encoder compresses states from $64 \times 64 \times C_{\text{in}}$ to $16 \times 16 \times C_{\text{latent}}$ with $C_{\text{latent}} = 8$. This corresponds to a relatively mild compression rate compared with some recent latent-emulation settings, but is appropriate for our 64×64 resolution regime and allows us to isolate the effect of latent space forecasting without introducing an aggressively lossy bottleneck.

Processors

Our main comparison is between two model classes with matched ViT backbones. The first is a stochastic generative model trained in latent space with a FM objective (Lipman et al., 2023). The second is a deterministic backbone trained probabilistically using noise injection and αfCRPS loss (Lang et al., 2024) in ambient space. By default, the CRPS model uses $M = 8$ ensemble members, with additional ensemble sizes considered in the ablations. For the CRPS ensemble, stochasticity is introduced through conditional layer normalisation (CLN) with adaLN-Zero-style modulation. Because the injected noise channels add parameters, we reduce the hidden width of the

Table 6: Processor architectures used in the main comparison and most of the ablations. The first two columns are the main comparison; the remaining four columns are architecture ablations. The right-most column is the only CRPS variant that uses a U-Net backbone instead of a ViT. Dashes mark settings that do not apply. Parameter counts are for the processor and exclude the autoencoder; reported values are for CNS ($C = 3$) and vary by less than 0.3M across the $C \in \{1, 2, 3\}$ datasets, since state-channel count only enters the patch embedding and output head. Latent processors operate on the autoencoder latent ($C_{\text{latent}} = 8$) and are identical across datasets.

	Main		Ablations			
	FM (latent)	CRPS (ambient)	FM (ambient)	CRPS (latent)	Diffusion (latent)	CRPS (ambient)
Backbone	ViT	ViT	ViT	ViT	ViT	U-Net
Width / channels	704	568	704	568	704	62, 124, 248, 496
Blocks	12	12	12	12	12	3, 3, 3, 3
Heads	8	8	8	8	8	—
Patch size	1	4	4	1	1	—
MLP expansion	4	4	4	4	4	1
Noise channels	—	1024	—	1024	—	1024
Ensemble M	—	8	—	8	—	8
Sampler / steps	Euler / 50	—	Euler / 50	—	Euler / 50	—
Loss	FM	αfCRPS	FM	αfCRPS	Karras DM	αfCRPS
Parameters	80.0M	80.9M	80.2M	81.8M	80.0M	81.3M

CRPS processor slightly so that its size remains comparable to the flow-matching processor. FM sampling uses Euler integration; the step count is ablated in Appendix F.8.

Table 6 summarises the processor architectures used in the main comparison and most of the ablations, including with U-Net and diffusion models. The FNO and MC-dropout variants are specified in their dedicated appendices (Appendices F.11–F.15).

The ViT processors share the same backbone hyperparameters (12 transformer blocks, 8 attention heads) and operate on a 16×16 token grid in both spaces, using patch size 4 on 64×64 ambient states and patch size 1 on 16×16 autoencoder latents. The autoencoder compression ablation (Appendix F.2) is the one exception, where an $8 \times 8 \times 8$ latent yields an 8×8 token grid; all other rows of Table 6 use the 16×16 grid. In both the FM/diffusion and CRPS pipelines, the single input frame and four output frames are folded into the channel dimension at the processor boundary, so the backbone always runs with a single time token; temporal mixing is therefore mediated entirely through the channel axis. The FM and diffusion ViTs use width 704; the CRPS-trained ViT uses width 568 to keep the parameter count near 80M after adding the 1024-dimensional noise modulation path. Reported counts are processor-only and exclude the autoencoder.

Simulation constants and conditioning

For each trajectory, the simulator parameters $\boldsymbol{c} \in \mathbb{R}^{C_c}$ are sampled once and held fixed for the entire rollout, including under autoregressive prediction. The state is normalised, and (for latent processors) passed through the autoencoder; $\boldsymbol{c}$ is not encoded and is used only for conditioning. We inject $\boldsymbol{c}$ into the processor in one of two ways.

Channel concatenation (used in the main ambient CRPS baseline). Each scalar in $\boldsymbol{c}$ is broadcast across the $W \times H$ grid and concatenated to the current state $\boldsymbol{x}^i$ along the channel axis, giving the processor input

$$\tilde{\boldsymbol{x}} \;=\; \boldsymbol{x}^i \,\|\, \boldsymbol{c} \;\in\; \mathbb{R}^{W \times H \times (C + C_c)}, \tag{8}$$

where $\|$ denotes concatenation along the channel axis (the spatial broadcast of $\boldsymbol{c}$ is implicit in the shape). Backbone modulation by $\boldsymbol{c}$ is then disabled so that $\boldsymbol{c}$ is not injected twice.

Backbone modulation (used in latent FM, latent CRPS, and diffusion). The vector $\boldsymbol{c}$ is embedded by a small MLP and added to the timestep/noise modulation vector that drives the per-block modulation in the backbone (see equations below). The state itself is not channel-augmented with $\boldsymbol{c}$.

The global-conditioning ablation in Appendix F.7 is the only run where the ambient CRPS model is switched from channel concatenation to backbone modulation, with all other settings of Table 6 unchanged.

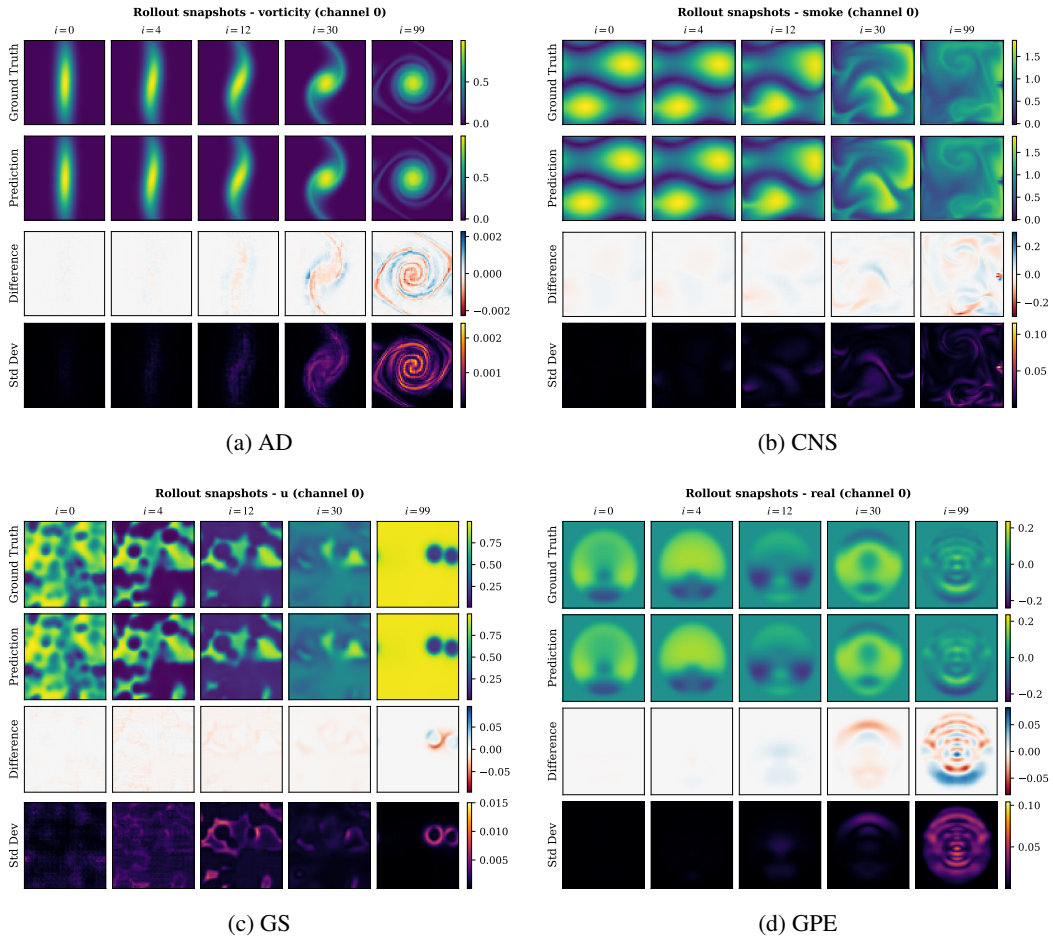
Backbone modulation

We use the ViT and U-Net backbones implemented in `Azula` (Rozet et al., 2025b). They both use a single shared adaptive LN per block. Letting F denote the block branch (self-attention followed by an MLP in the ViT; optional spatial attention followed by a convolutional feed-forward block in the U-Net), the per-block computation mapping input $\boldsymbol{h}$ to output $\boldsymbol{h}^+$ is

$$\begin{aligned}\tilde{\boldsymbol{h}} &= (1 + a(\boldsymbol{\eta})) \odot \mathrm{LN}(\boldsymbol{h}) + b(\boldsymbol{\eta}),\\ \boldsymbol{h}^+ &= \big(\boldsymbol{h} + g(\boldsymbol{\eta}) \odot F(\tilde{\boldsymbol{h}})\big)\big(1 + g(\boldsymbol{\eta})^2\big)^{-1/2},\\ [a, b, g] &= A(\boldsymbol{\eta}),\end{aligned}$$

with LN a non-affine layer normalisation, A a two-layer MLP (Linear–SiLU–Linear) whose final-layer weights are multiplied by 10^{-2} at initialisation, and (a, b, g) shared across the attention and MLP sub-layers within a block (rather than the two-modulation variant of the canonical adaLN-Zero (Peebles & Xie, 2023)). The variance-preserving factor $(1 + g(\boldsymbol{\eta})^2)^{-1/2}$ keeps the residual norm bounded as g grows during training.

C EXAMPLE ROLLOUTS



(a) AD

(b) CNS

(c) GS

(d) GPE

Figure 5: Example rollout snapshots (channel 0) for one test case per dataset. Top to bottom row: the ground-truth trajectory, ensemble mean, difference between ground truth and ensemble mean, and ensemble standard deviation.

D OTHER ROLLOUT METRICS

Figure 6 shows a number of accuracy and calibration metrics measuring performance over 100-step autoregressive rollouts. We report the same metrics as those used for single-step performance evaluation in the main text (SSR, CRPS and VRMSE) as well as the energy score (a multivariate generalisation of the CRPS) and power spectrum RMSE (PSRMSE) at different spatial frequencies (following Rozet et al., 2025a).

The results indicate that the two training approaches remain broadly comparable in predictive accuracy across a range of metrics. As expected, error increases with lead time for both methods. AD is the clearest separation in favour of CRPS, where both models remain accurate but the CRPS rollout error is one to two orders of magnitude lower. On CNS and GS, CRPS also retains a modest accuracy advantage at larger lead times. The CRPS training approach also tends to have better SSR over long lead times, except on the AD dataset.

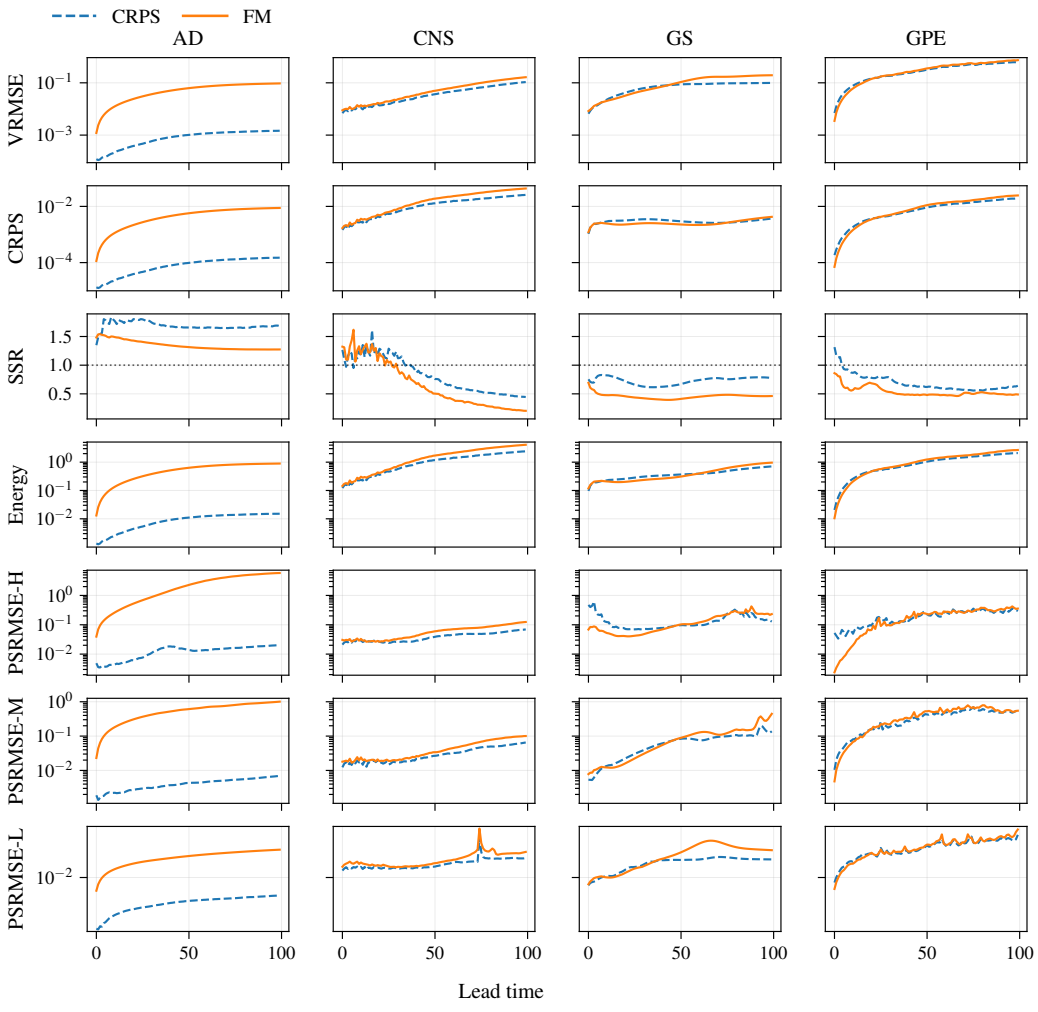


Figure 6: Top to bottom: VRMSE, CRPS, SSR, Energy score, power spectrum RMSE (high), power spectrum RMSE (medium), power spectrum RMSE (low) over lead time by dataset (AD, CNS, GS, GPE) and model (CRPS, FM). The two approaches have comparable accuracy across metrics while the CRPS training approach tends to have better SSR except on the AD dataset.

E FINITE-TEST-TRAJECTORY STANDARD ERRORS

We regenerated predictions while retaining trajectory identifiers and quantify sampling uncertainty from the finite test set conditional on each trained model. Since windows, output times, spatial locations, and channels from the same rollout are dependent, the independently generated test trajectory is the sampling unit. For a scalar metric, let U_i denote the statistic obtained after first aggregating all relevant predictions within test trajectory i. For n trajectories, we report

$$\overline{U} = \frac{1}{n}\sum_{i=1}^{n} U_i, \qquad \widehat{\mathrm{SE}}(\overline{U}) = \sqrt{\frac{1}{n(n-1)}\sum_{i=1}^{n}(U_i - \overline{U})^2}. \tag{9}$$

This calculation is used for VRMSE, CRPS, SSR, energy score, and the spectral error metrics. For GS, which has six equally represented pattern strata, we preserve the sampling design: if stratum h has weight w_h, size n_h, mean $\overline{U}_h$, and sample variance s_h^2, then

$$\overline{U}_{\mathrm{strat}} = \sum_{h=1}^{6} w_h \overline{U}_h, \qquad \widehat{\mathrm{SE}}(\overline{U}_{\mathrm{strat}}) = \left(\sum_{h=1}^{6} \frac{w_h^2 s_h^2}{n_h}\right)^{1/2}. \tag{10}$$

In our GS test set the strata are equally weighted, so $w_h = 1/6$, and each contains $n_h = 4$ trajectories.

Coverage MAE requires an additional covariance calculation because coverage at different nominal levels is evaluated on the same trajectories. Let c_{il} be the empirical coverage for trajectory i at nominal level q_l, where $l \in \{1, \ldots, L\}$, so that $\boldsymbol{c}_i = (c_{i1}, \ldots, c_{iL}) \in \mathbb{R}^L$ and $\overline{c}_l = n^{-1}\sum_i c_{il}$. Define coverage MAE as the scalar function

$$g(\overline{\boldsymbol{c}}) = \frac{1}{L}\sum_{l=1}^{L} |\overline{c}_l - q_l|, \qquad \partial_l g(\overline{\boldsymbol{c}}) = \frac{1}{L}\,\mathrm{sign}(\overline{c}_l - q_l). \tag{11}$$

For the unstratified datasets, the estimated covariance of the mean coverages is

$$\widehat{\mathrm{Cov}}(\overline{c}_l, \overline{c}_m) = \frac{1}{n(n-1)}\sum_{i=1}^{n}(c_{il} - \overline{c}_l)(c_{im} - \overline{c}_m). \tag{12}$$

The stratified analogue is used for GS. The delta method then gives

$$\widehat{\mathrm{SE}}[g(\overline{\boldsymbol{c}})] = \left[\partial_l g(\overline{\boldsymbol{c}})\, \widehat{\mathrm{Cov}}(\overline{c}_l, \overline{c}_m)\, \partial_m g(\overline{\boldsymbol{c}})\right]^{1/2}, \tag{13}$$

where repeated level indices l and m are summed; i remains the trajectory index and is summed explicitly in the covariance estimator. This retains the covariance across nominal coverage levels.

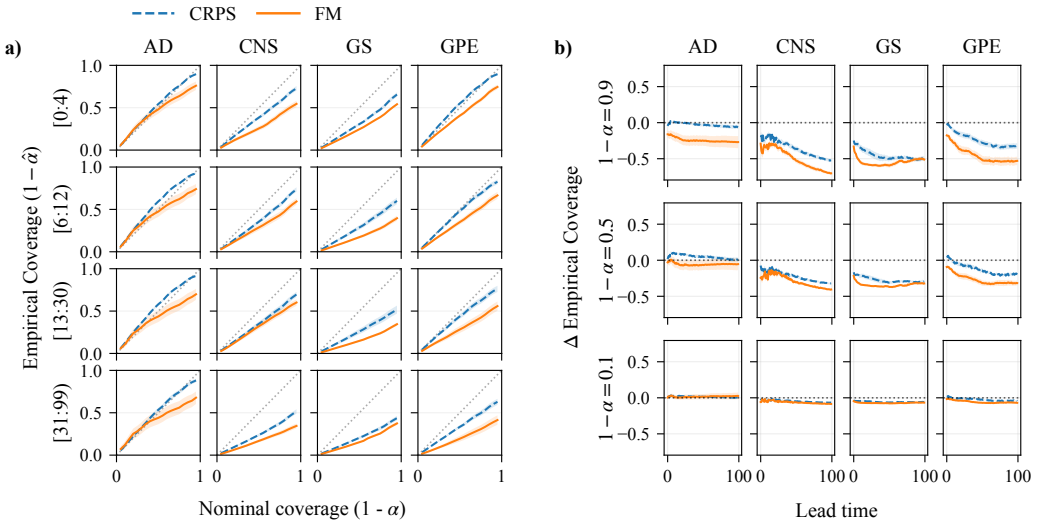


Figure 7: Coverage with finite-test-trajectory standard errors for the main comparison across datasets (AD, CNS, GS, GPE). Lines show trajectory-aggregated estimates and bands show plus or minus one standard error. Panels report (a) empirical vs. nominal coverage over rollout windows and (b) Δ empirical coverage (observed minus nominal) at three nominal levels over rollout lead time. The GS bands account for its six pattern strata.

Table 7: Single-step main-comparison results reported as mean (finite-test-trajectory standard error). Bold denotes the better result for each dataset.

Dataset	Model	VRMSE ↓	Coverage MAE ↓	CRPS ↓	SSR → 1
AD	CRPS	**1.8e-04 (2.2e-05)**	0.041 (0.009)	**8.8e-05 (1.3e-05)**	**1.07 (0.03)**
	FM	2.2e-04 (3.2e-05)	**0.019 (0.003)**	1.3e-04 (2.0e-05)	1.30 (0.04)
CNS	CRPS	**5.6e-03 (1.2e-04)**	**0.124 (0.007)**	**2.7e-03 (8.6e-05)**	**1.16 (0.04)**
	FM	8.7e-03 (3.5e-04)	0.284 (0.006)	3.9e-03 (1.5e-04)	0.66 (0.03)
GS	CRPS	**1.1e-03 (8.4e-05)**	0.198 (0.008)	5.2e-04 (7.2e-05)	**0.67 (0.01)**
	FM	1.3e-03 (5.3e-05)	**0.183 (0.006)**	**5.1e-04 (2.3e-05)**	0.57 (0.01)
GPE	CRPS	**2.8e-02 (2.9e-03)**	**0.034 (0.010)**	**9.6e-03 (1.2e-03)**	**0.92 (0.05)**
	FM	3.2e-02 (3.4e-03)	0.24 (0.01)	1.1e-02 (1.6e-03)	0.43 (0.03)

The compact coverage comparison in Figure 7 and the single-step results in Table 7 leave the qualitative comparison unchanged: CRPS retains substantially better coverage on CNS and GPE, while the smaller AD and GS differences remain dataset dependent.

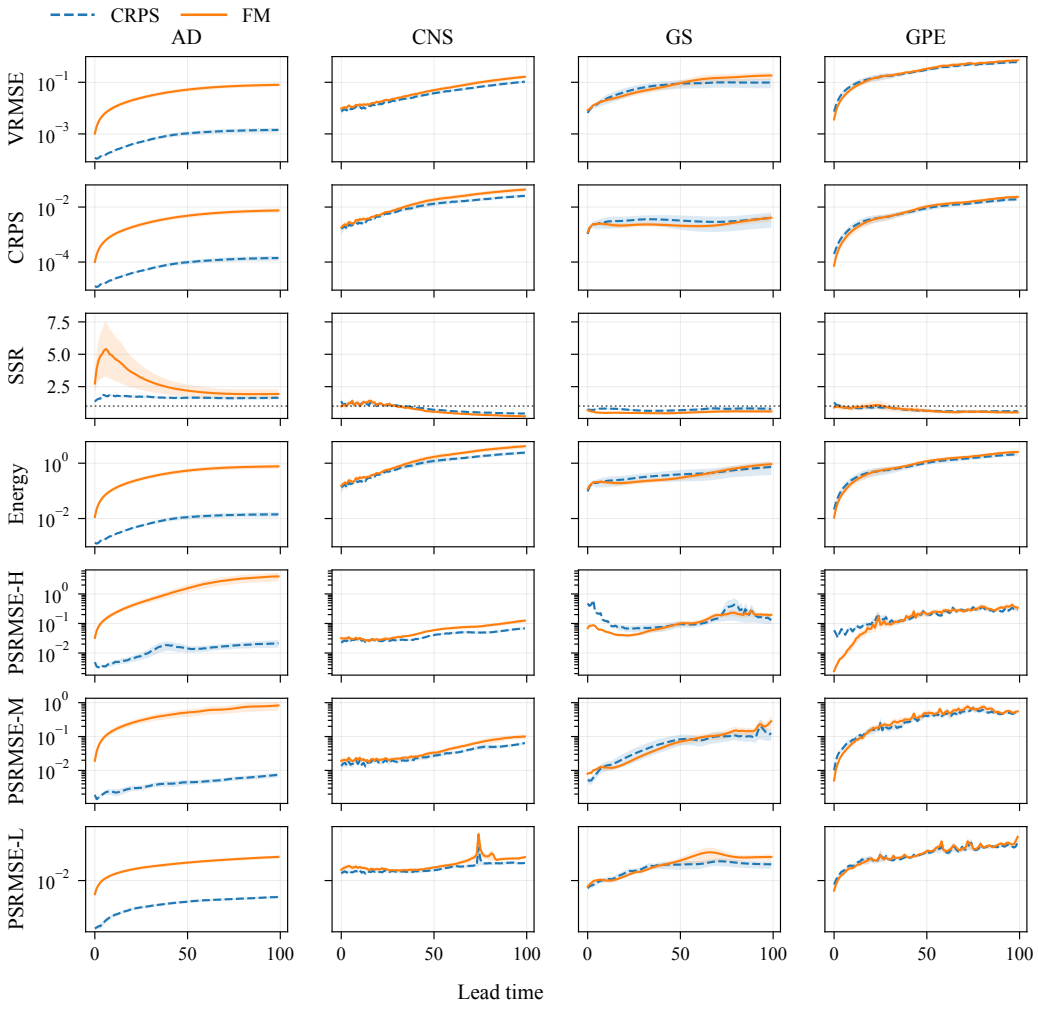


Figure 8: Finite-test-trajectory uncertainty for the main CRPS and FM comparison across datasets (AD, CNS, GS, GPE). Lines show the trajectory-aggregated metric and shaded bands show plus or minus one standard error over test trajectories. Rows report VRMSE, CRPS, SSR, energy score, and high-, mid-, and low-frequency power-spectrum RMSE over rollout lead time. The GS bands account for its six pattern strata.

Figure 8 provides the corresponding standard-error bands for accuracy, probabilistic scores, and spectral errors. These standard errors quantify finite-test-trajectory sampling only and exclude training-seed and training-data variability (Appendices F.12 and F.13).

F Ablations and other experiments

Unless noted otherwise, each ablation below varies a single setting (named by its subsection title) from the corresponding column of Table 6; all other architectural choices, training schedule, and evaluation protocol follow the main runs (Section 4.1).

F.1 Latent vs. ambient space training

We consider FM training in ambient space and CRPS training in latent space, providing counterparts to the results in the main comparison. Applying FM directly in ambient space follows by substituting $\boldsymbol{x}_{\text{in}}$, $\boldsymbol{x}_{\text{out}}$, and $\boldsymbol{x}_t$ for $\boldsymbol{z}_{\text{in}}$, $\boldsymbol{z}_{\text{out}}$, and $\boldsymbol{z}_t$ as outlined in Section 3.3, without the need for encoding and

decoding. Applying CRPS in latent space uses the same latent representation as FM, with the CRPS loss calculated directly on the latent space predictions $p_\theta(\boldsymbol{z}_{\text{out}}|\boldsymbol{z}_{\text{in}}, \boldsymbol{c})$.

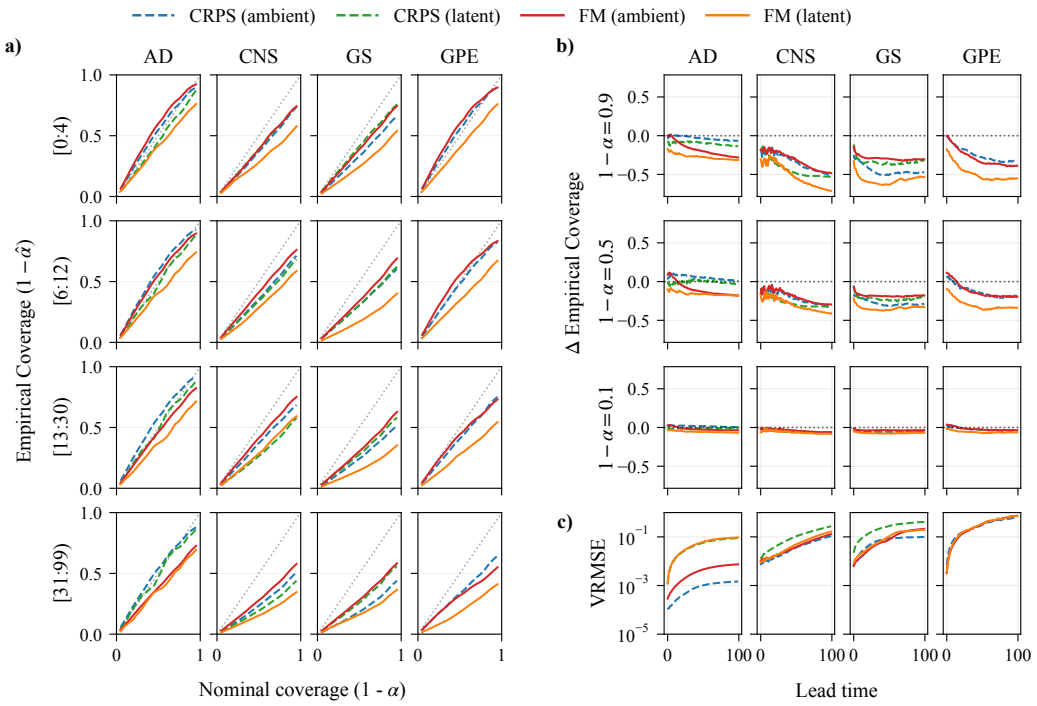


Figure 9: Latent vs. ambient training: FM (ambient vs. latent) and CRPS training (ambient vs. latent), $M = 8$, across datasets (AD, CNS, GS, GPE). (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time. The latent FM model has consistently worse coverage while all the other approaches perform comparably. No result is reported for the CRPS model trained in latent space on the GPE dataset because the ensemble collapsed.

Figure 9 shows that moving FM from latent space back to ambient space substantially improves its coverage and makes it broadly comparable to CRPS training. The corresponding shift for CRPS is much smaller: training CRPS in latent space causes only a limited degradation in coverage, although it does increase VRMSE for CNS and GS. For GPE, the latent space CRPS run collapsed entirely, so in that case latent training is not a viable substitute for the ambient space model. All variants perform well on AD, although the CRPS-trained models are still more stable at large lead times.

F.2 INCREASED AUTOENCODER COMPRESSION

The compression rate of the autoencoder reported in the main results is between 2x and 6x due to the original resolution being 64×64 (the rate varies due to the variation in the number of channels per dataset). While this compression rate is fairly small, it already had a detrimental effect on performance (coverage for FM and accuracy for CRPS trained models). In this ablation, we trained an autoencoder for a latent space at lower resolution of $8 \times 8 \times 8$ for CNS, increasing compression to 24x, while keeping the number of parameters roughly constant at 50M. Figure 10 shows the effect on FM where coverage and accuracy degrade further at 24x compression, confirming that the latent space size is a limiting factor for performance in latent space models.

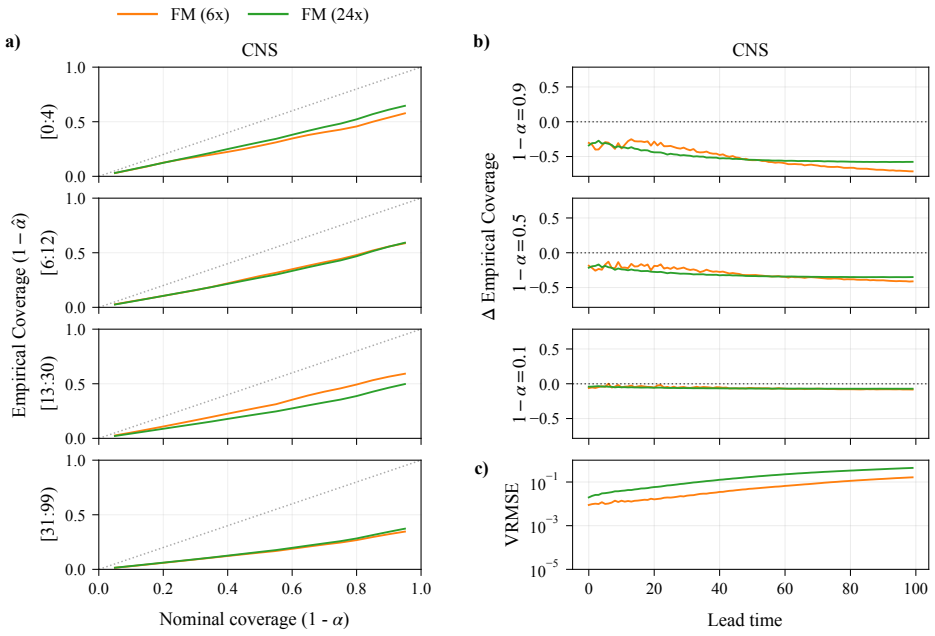


Figure 10: FM performance on CNS at increased autoencoder compression ($8 \times 8 \times 8$ latent, 24x compression) vs. the 6x compression used in the main results. (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time. Increasing the autoencoder compression rate further degrades FM coverage and accuracy performance.

F.3 WINKLER-BASED MONITORING

As discussed in Section 4.1, during CRPS training we used the validation Winkler score to select the best model checkpoint. Figure 11 compares this against choosing a checkpoint based on the best validation loss. In most cases, the two checkpoint selection strategies have comparable coverage but the Winkler score checkpoint performs better on the GPE dataset.

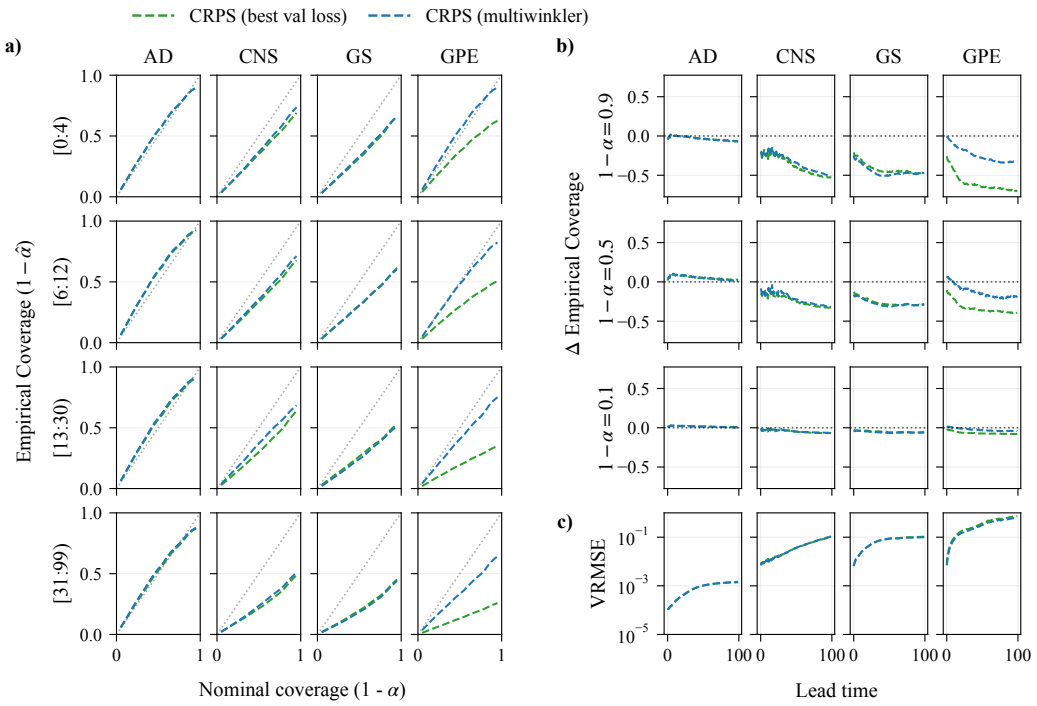


Figure 11: Checkpoint selection driven by Winkler score (default) compared with best validation loss ($M = 8$) across datasets (AD, CNS, GS, GPE). (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time. The two checkpoint selection strategies have comparable coverage in most cases, but the Winkler score-driven strategy performs better on the GPE dataset.

F.4 CRPS LOSS VARIANT

Across all four datasets, Figure 12 compares fCRPS with the αfCRPS baseline from the main results. Rollout VRMSE remains similar. Coverage improves most clearly on CNS, changes little on AD and GPE, and worsens at long lead times on GS.

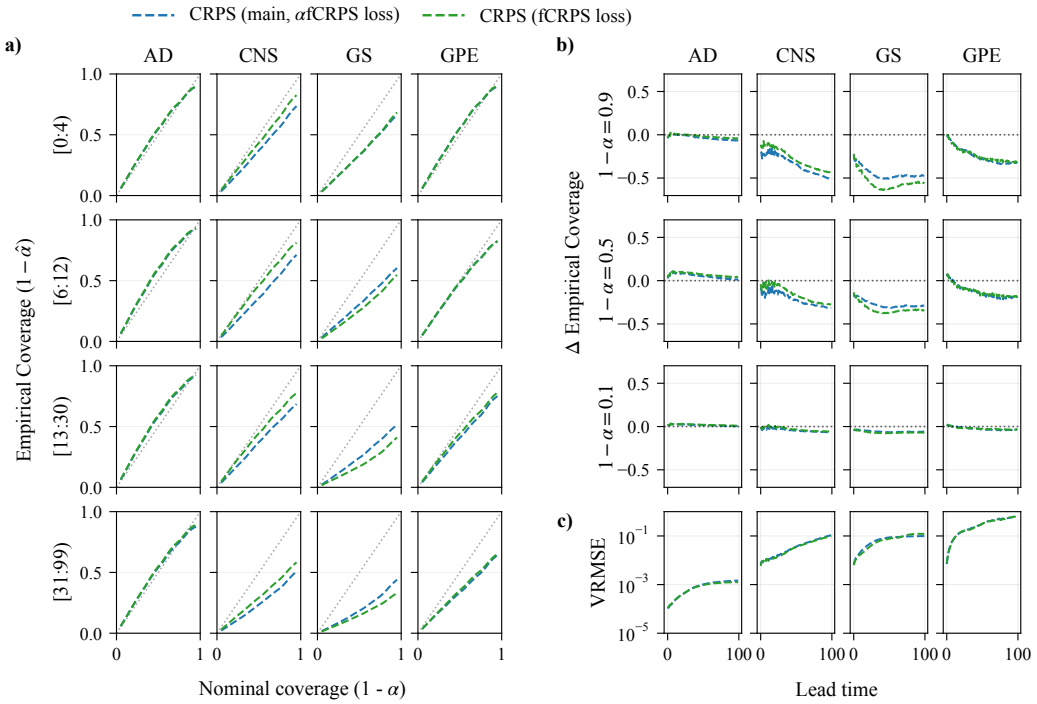


Figure 12: fCRPS vs. αfCRPS training losses for the noise-injected ensemble ($M = 8$) across datasets (AD, CNS, GS, GPE). The main CRPS model uses αfCRPS loss. (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time.

F.5 ENSEMBLE SIZE

We compared the effect of ensemble size on coverage by comparing $M \in \{4, 8, 16\}$ on all datasets. As shown in Figure 13, while there is some slight variability, we did not observe a consistent effect of ensemble size on coverage across datasets.

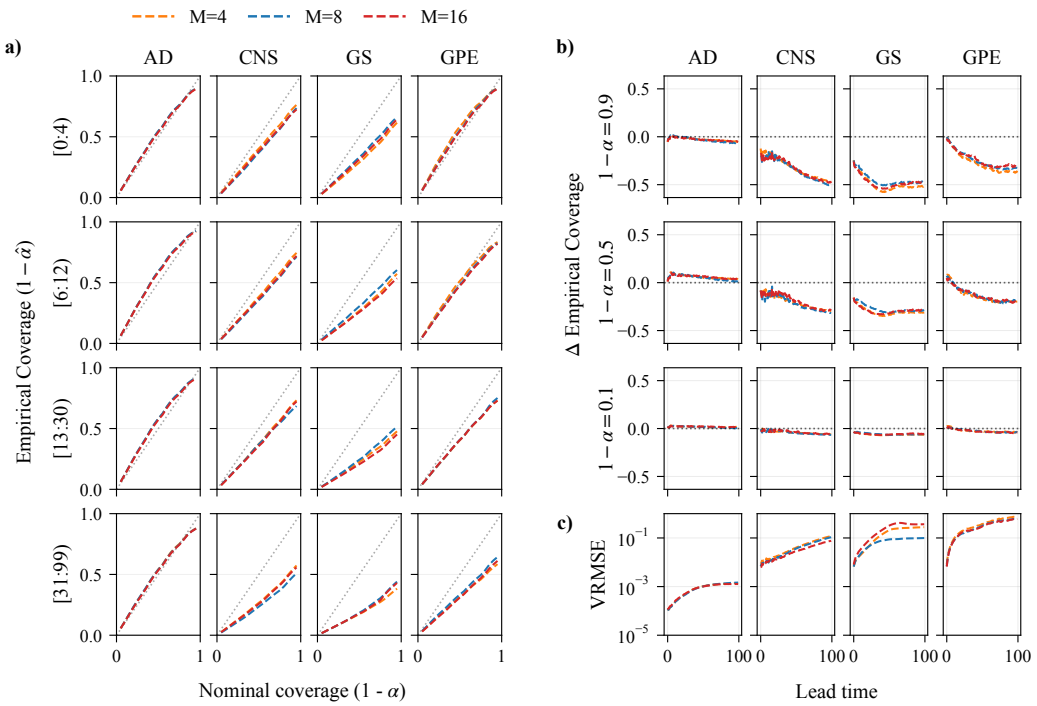


Figure 13: Effect of ensemble size on CRPS training for $M \in \{4, 8, 16\}$ across datasets (AD, CNS, GS, GPE). (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) vs. lead time over 100-step rollouts. (c) VRMSE vs. lead time. Ensemble size does not have a consistent effect on coverage across datasets.

F.6 NOISE CHANNELS

To create an ensemble for CRPS training, we have to specify the number of noise channels used in the noise injection mechanism. As this increases the parameters in the model, we reduced the hidden dimension of the ViT backbone to maintain the same model size as used in FM. In this ablation we evaluate the trade-offs between noise channel size and hidden dimension size by reducing the noise channels to 256 instead of the original 1024 and increasing the hidden dimension (from 568 to 704) to maintain the same model size.

Figure 14 shows that the model accuracy did not suffer from the reduced hidden dimension in the backbone, which is not surprising given the difference is fairly mild. Additionally, there might be a slight benefit from having larger noise channel size in terms of coverage.

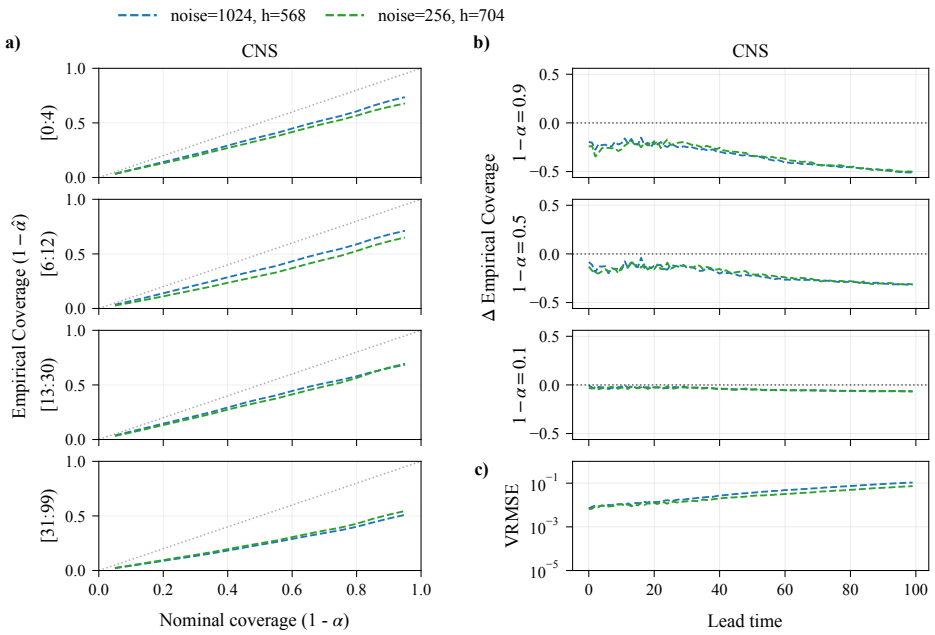


Figure 14: Effect of noise injection width (256 vs. 1024 noise channels) with adjusted hidden width to keep parameter count comparable on the CNS dataset. (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time. The models perform comparably, although there may be a slight benefit from having larger noise channel size in terms of coverage.

F.7 ALTERNATIVE GLOBAL CONDITIONING STRATEGIES

For the CRPS-trained ViT ensemble on CNS ($M = 8$), we ablate how global simulator parameters are injected into the model. We compare (i) *channel concatenation*, where a broadcast embedding of the conditioning variables is concatenated to the input channels, against (ii) *backbone modulation*, where conditioning drives feature-wise modulation within the network (analogous to the conditioning mechanism used in our FM backbone).

Figure 15 shows that our strategy of channel concatenation outperforms backbone modulation in terms of coverage performance.

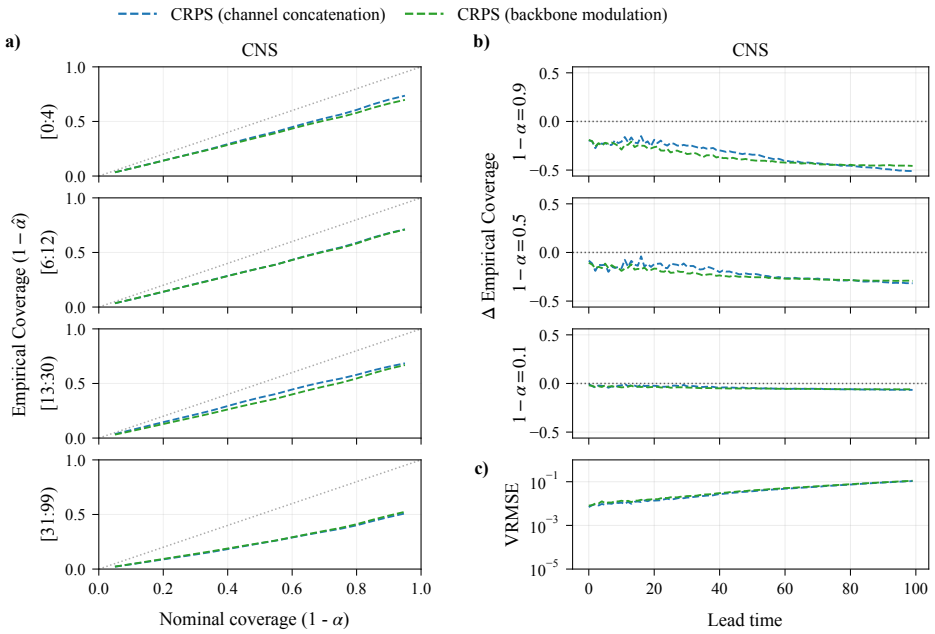


Figure 15: Global conditioning ablation for the CRPS-trained ViT ensemble ($M = 8$) on the CNS dataset. We compare conditioning via channel concatenation against backbone modulation. (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time. Channel concatenation outperforms backbone modulation in terms of coverage performance.

F.8 FLOW MATCHING ODE STEPS

FM sampling integrates a learned velocity field along a chosen number of ODE steps; more steps trade compute for sample quality. We swept the number of integration steps to explore performance sensitivity to this choice. Figure 16 shows that coverage typically improves with increased ODE steps, while VRMSE remains similar.

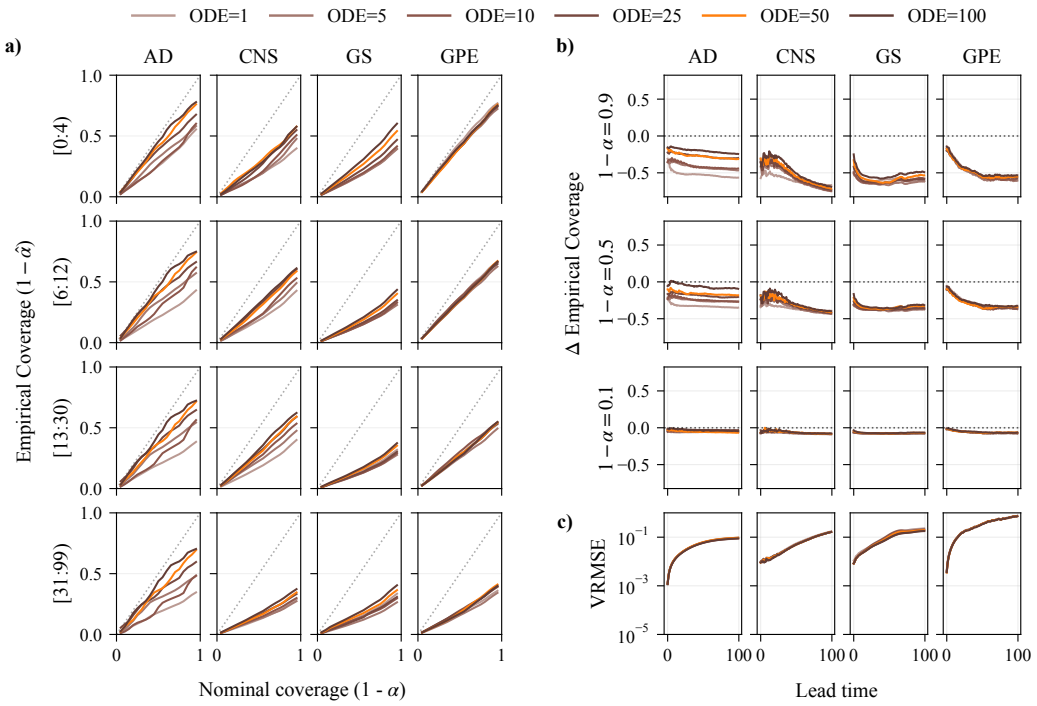


Figure 16: FM performance as a function of the number of ODE integration steps used at sampling time across datasets (AD, CNS, GS, GPE). (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time. Coverage typically improves with increased ODE steps, while VRMSE remains similar.

F.9 Flow matching EMA weights

For the latent FM models, we compare the original weights used in the main results with their exponential moving average (EMA, decay 0.999) from the same selected checkpoint. Both evaluations use the same autoencoder, 10 ensemble members and 50 Euler steps. Figure 17 shows no consistent benefit from EMA: coverage worsens most clearly on AD, with smaller differences on the other datasets. Rollout VRMSE remains broadly similar, with some increase on GS.

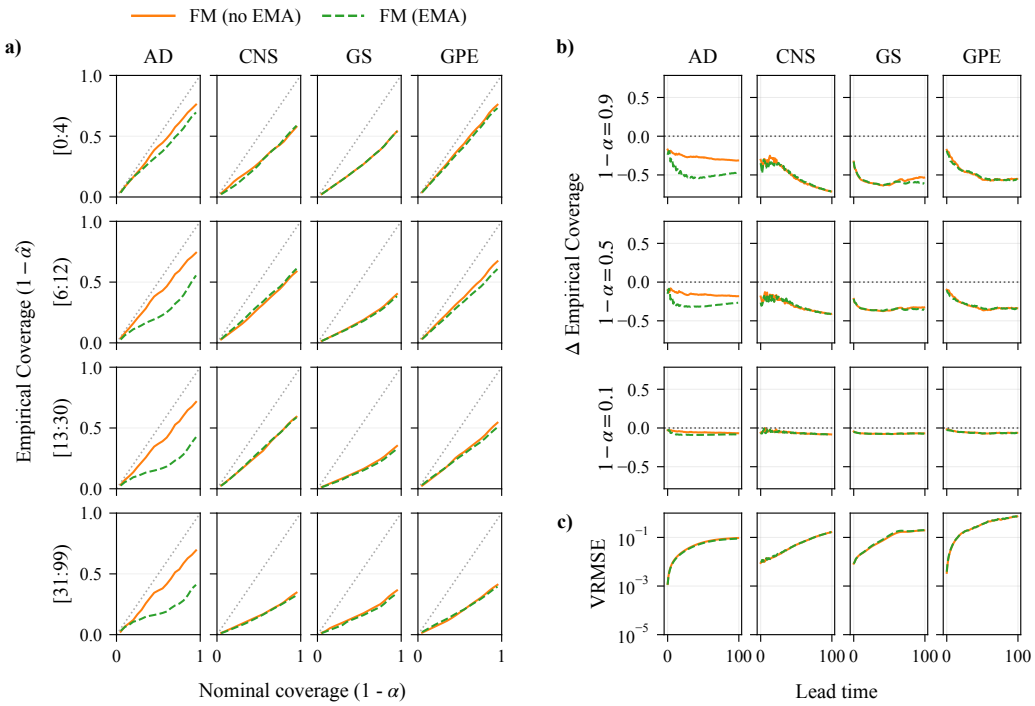


Figure 17: Original vs. EMA weights from the same selected latent FM checkpoints across datasets (AD, CNS, GS, GPE). Orange solid lines show the original weights used in the main results; green dashed lines show EMA weights. (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time.

F.10 ALTERNATIVE PROCESSOR ARCHITECTURES

We tested whether the results generalise across different architectures. Specifically, we compared the FM model against denoising diffusion (Ho et al., 2020) using the same ViT backbone. We also compared a CRPS-trained ensemble of U-Net models against the ViT ensemble. We used a U-Net implemented in `Azula` (Rozet et al., 2025b). This is an adaptation of the classic architecture from Ronneberger et al. (2015) to work as a backbone in generative models. Using this variant allowed us to use a similar noise injection strategy as that used in the ViT ensemble (see Appendix B). All models were matched on size and evaluated using the metrics described in Section 4.1.

Figures 18 and 19 compare these models across all four datasets (AD, CNS, GS, GPE). Diffusion and FM have broadly similar rollout VRMSE on AD, CNS and GPE, while diffusion has higher rollout error on GS; coverage differences vary with dataset and lead time. The U-Net ensemble has lower VRMSE on AD, but higher error on CNS, GS and GPE and worse rollout coverage on CNS and GS than the ViT ensemble.

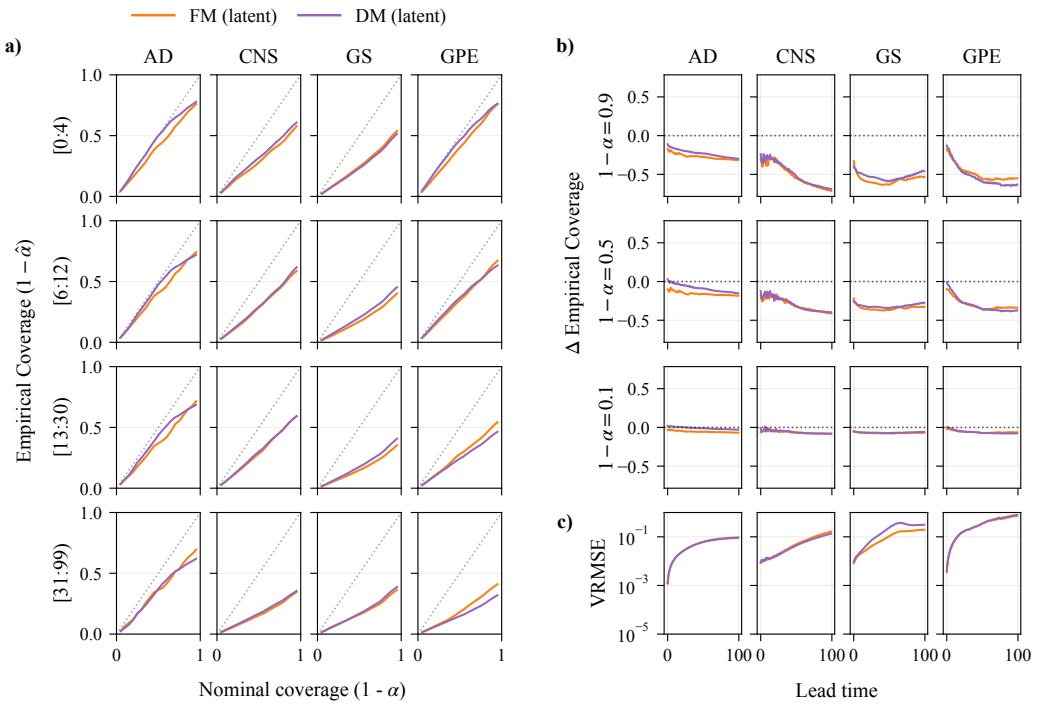


Figure 18: FM vs. denoising diffusion in *latent* space (matched backbones) across datasets (AD, CNS, GS, GPE). (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time.

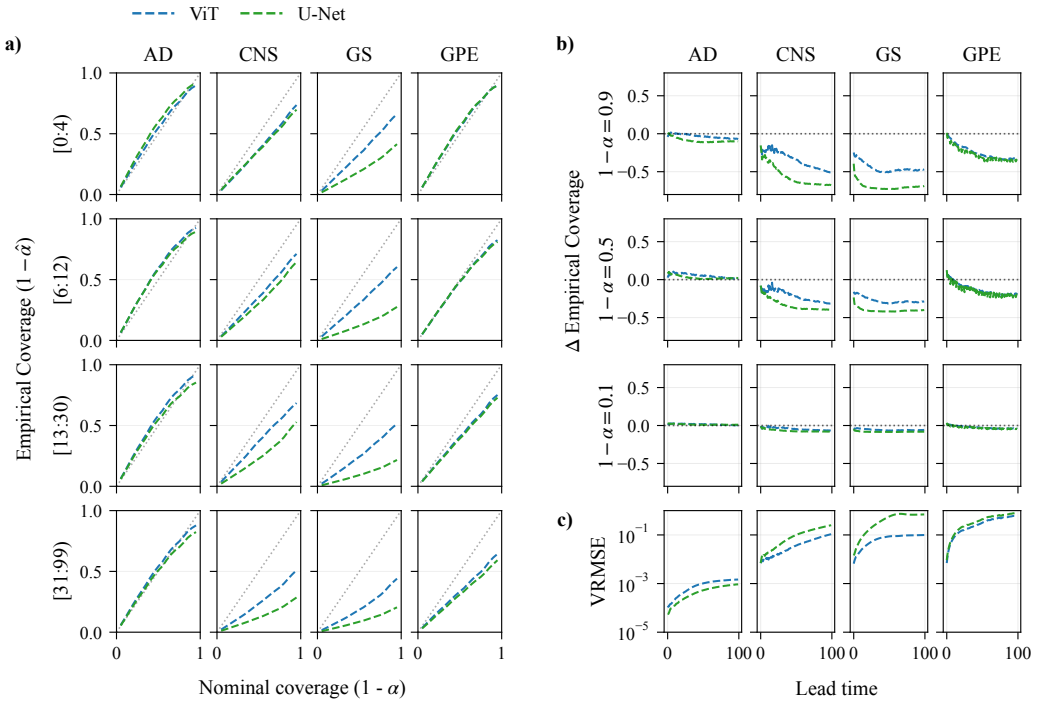


Figure 19: ViT vs. U-Net CRPS-trained ensemble ($M = 8$) performance across datasets (AD, CNS, GS, GPE). (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) vs. lead time over 100-step rollouts. (c) VRMSE vs. lead time.

F.11 FOURIER NEURAL OPERATOR BACKBONE

We evaluate how the CRPS-training framework performs with a Fourier neural operator (FNO) backbone (Li et al., 2021). An FNO layer combines a pointwise linear map with a learned spectral convolution,

$$\boldsymbol{h}_{j+1}(\boldsymbol{s}) = \sigma\big(W_j \boldsymbol{h}_j(\boldsymbol{s}) + \mathcal{F}^{-1}[R_j \odot \mathcal{F}[\boldsymbol{h}_j]]\,(\boldsymbol{s})\big)\,, \tag{14}$$

where $\boldsymbol{s}$ is spatial position, $\boldsymbol{h}_j(\boldsymbol{s}) \in \mathbb{R}^d$ is the activation entering layer j, σ is an elementwise nonlinear activation, and $j \in \{0, \ldots, 3\}$ indexes the four FNO layers; here $d = 264$. The matrix W_j mixes channels pointwise, $\mathcal{F}$ is the spatial Fourier transform, and R_j contains learned multipliers for the retained modes. Both $\mathcal{F}[\boldsymbol{h}_j]$ and R_j are complex valued, so the full complex coefficients, including their magnitude and phase, contribute to the prediction. The direct spectral path truncates modes outside the retained band, although the pointwise path and nonlinear layers can still generate higher-frequency structure (Li et al., 2021).

Our FNO uses four layers, 264 hidden channels, and 16 modes along each spatial axis. Counting each complex coefficient as two real scalars gives approximately 81.1M parameters, closely matching the 80.8M ViT baseline. Because the FNO does not use the ViT conditional-normalisation path, each of the eight ensemble members instead receives an independently sampled spatial white-noise channel concatenated to its input. Both backbones use ambient-space αfCRPS training, the same optimiser and training budget, and the same checkpoint-selection and evaluation protocol. FourCastNet 3 uses a related spherical neural operator for probabilistic weather forecasting, but combines local spherical convolutions with global spectral convolutions defined through spherical harmonic transforms to respect the geometry of the globe, rather than using the planar Fourier transform employed here (Bonev et al., 2023; 2025).

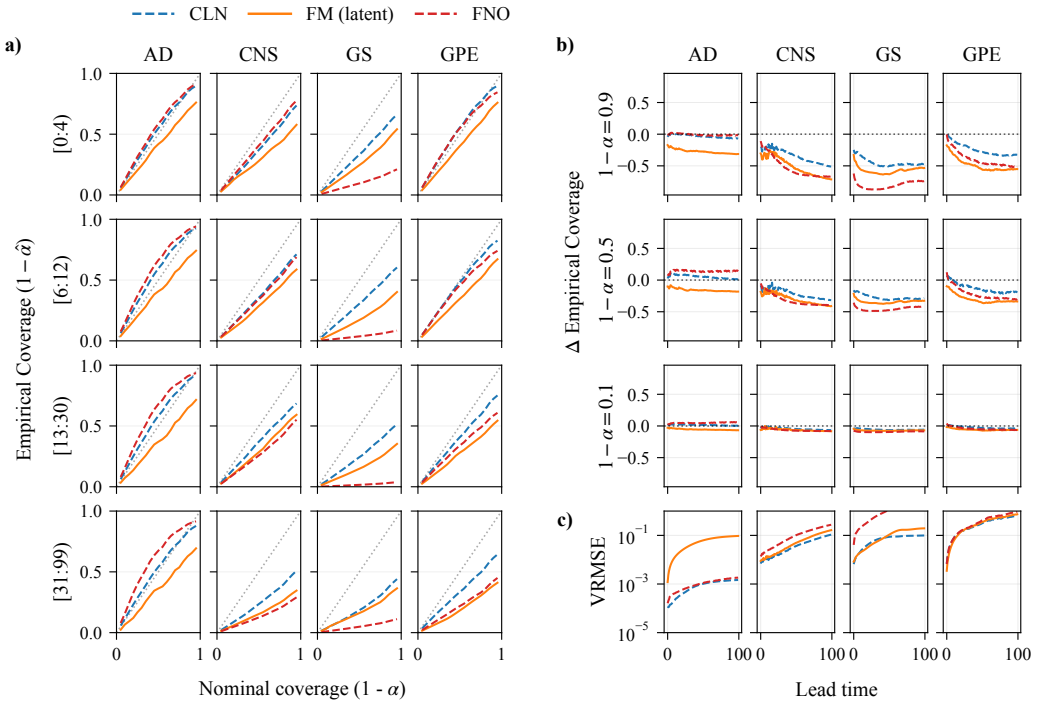


Figure 20: FNO and ViT CRPS-trained ensembles (the ViT ensemble is labelled CLN), with latent FM included for context, across datasets (AD, CNS, GS, GPE). (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time. The FNO is competitive on some early rollout windows but degrades more consistently over long rollouts.

Table 8: Single-step comparison of the ViT and FNO backbones under CRPS training. Bold denotes the better result for each dataset.

Dataset	Backbone	VRMSE ↓	Coverage MAE ↓	CRPS ↓	SSR → 1
AD	ViT	1.8e-04	**0.04**	**8.8e-05**	**1.07**
	FNO	**1.7e-04**	0.06	8.9e-05	1.82
CNS	ViT	**5.6e-03**	**0.12**	**2.7e-03**	**1.16**
	FNO	1.4e-02	0.18	6.4e-03	0.71
GS	ViT	**1.1e-03**	0.20	**5.2e-04**	**0.67**
	FNO	3.7e-03	**0.18**	2.3e-03	0.56
GPE	ViT	**2.8e-02**	**0.04**	**9.7e-03**	**0.92**
	FNO	6.3e-02	0.09	2.4e-02	0.56

The FNO is competitive for single-step AD accuracy and GS coverage, but the ViT has lower coverage MAE on the other systems and better SSR on all four (Table 8). Figure 20 shows a clearer rollout separation with the ViT backbone providing more consistent rollout accuracy and coverage, with lower coverage MAE in the $[31{:}99)$ window on all four datasets. The largest long-window VRMSE gap occurs on GS (1.3 for FNO versus 0.095 for ViT), apparently reflecting spatial displacement of otherwise coherent localised structures.

F.12 SENSITIVITY TO TRAINING SEED

We retrained every model in the main comparison with an independent random seed while keeping the training, validation, and test trajectories and all other training settings fixed. The comparison measures sensitivity to optimisation and other seeded training operations.

Table 9: Single-step results for the original and repeated training runs, reported as seed 1 / seed 2. Both seeds use the same training, validation, and test trajectories.

Dataset	Model	VRMSE ↓	Coverage MAE ↓	CRPS ↓	SSR → 1
AD	CRPS	1.8e-04 / 1.9e-04	0.04 / 0.04	8.8e-05 / 9.2e-05	1.07 / 1.06
	FM	2.2e-04 / 2.1e-04	0.02 / 0.02	1.3e-04 / 1.2e-04	1.31 / 1.27
CNS	CRPS	5.6e-03 / 5.8e-03	0.12 / 0.10	2.7e-03 / 2.8e-03	1.16 / 1.18
	FM	8.7e-03 / 8.7e-03	0.28 / 0.29	3.9e-03 / 3.9e-03	0.65 / 0.63
GS	CRPS	1.1e-03 / 1.1e-03	0.20 / 0.21	5.2e-04 / 5.2e-04	0.67 / 0.65
	FM	1.3e-03 / 1.3e-03	0.18 / 0.18	5.1e-04 / 5.0e-04	0.57 / 0.59
GPE	CRPS	2.8e-02 / 3.0e-02	0.04 / 0.04	9.7e-03 / 1.1e-02	0.92 / 0.96
	FM	3.2e-02 / 2.8e-02	0.24 / 0.23	1.1e-02 / 9.6e-03	0.44 / 0.48

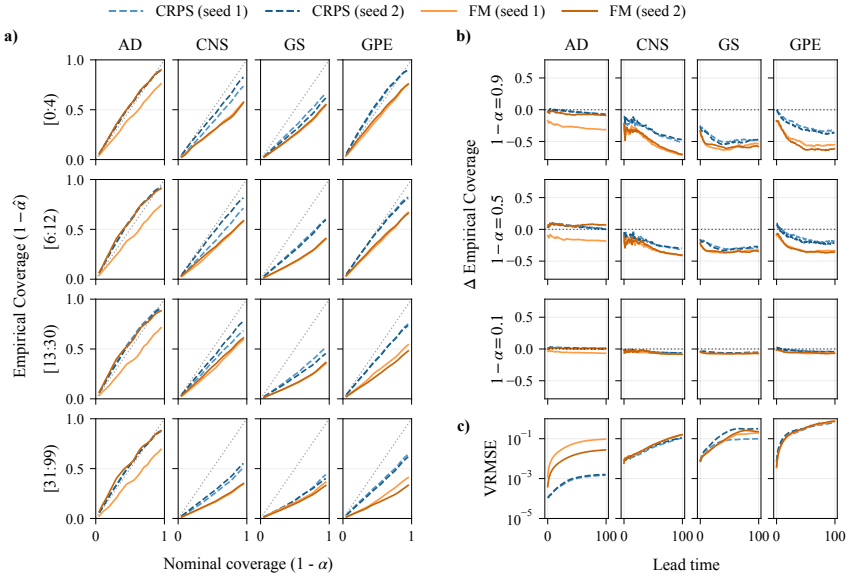


Figure 21: Performance of the main CRPS and FM models across datasets (AD, CNS, GS, GPE) reported for two independent training runs with different random seeds, denoted as seed 1 / seed 2. (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time. The two seeds give similar overall coverage trends but FM coverage and rollout VRMSE differ notably for AD. Seed differences are also visible in CRPS coverage on CNS and rollout VRMSE on GS.

Table 9 shows close agreement in the single-step metrics across both seeds. Coverage MAE and SSR retain the same qualitative pattern on all four systems. For seed 2 on GPE, FM has lower VRMSE and CRPS, but the CRPS-trained ensemble still has substantially lower coverage MAE and an SSR closer to one. Figure 21 shows that the full rollout trajectories are also broadly consistent, although seed sensitivity is more visible for FM accuracy on AD and CRPS accuracy on GS at long lead times.

F.13 SENSITIVITY TO TRAINING DATA

The seed experiment conditions on a fixed benchmark. We separately test sensitivity to the finite training sample on all four datasets by generating independent training trajectories from the same simulator parameter distributions and retraining both pipelines. The original test trajectories, training seed, and all other training settings remain fixed, so each main model and training-data rerun is evaluated on the same test cases. This isolates the effect of replacing the sampled training trajectories, following the fixed-test design used in related PDE-emulator datasets (Takamoto et al., 2022; Ohana et al., 2024).

Table 10: Single-step results for the main training data and the independently generated training set, reported as main / new training data. All pairs use the same test trajectories; the corresponding rollout estimates appear in Figure 22.

Dataset	Model	VRMSE ↓	Coverage MAE ↓	CRPS ↓	SSR → 1
AD	CRPS	1.8e-04 / 1.7e-04	0.041 / 0.040	8.8e-05 / 8.3e-05	1.07 / 1.08
	FM	2.2e-04 / 2.1e-04	0.019 / 0.019	1.3e-04 / 1.2e-04	1.30 / 1.30
CNS	CRPS	5.6e-03 / 6.0e-03	0.124 / 0.111	2.7e-03 / 2.9e-03	1.16 / 1.11
	FM	8.7e-03 / 8.9e-03	0.284 / 0.288	3.9e-03 / 4.1e-03	0.66 / 0.64
GS	CRPS	1.1e-03 / 1.2e-03	0.198 / 0.20	5.2e-04 / 6.1e-04	0.67 / 0.66
	FM	1.3e-03 / 1.3e-03	0.183 / 0.186	5.1e-04 / 5.2e-04	0.57 / 0.58
GPE	CRPS	2.8e-02 / 3.0e-02	0.034 / 0.026	9.6e-03 / 1.1e-02	0.92 / 0.99
	FM	3.2e-02 / 3.0e-02	0.24 / 0.24	1.1e-02 / 1.0e-02	0.43 / 0.45

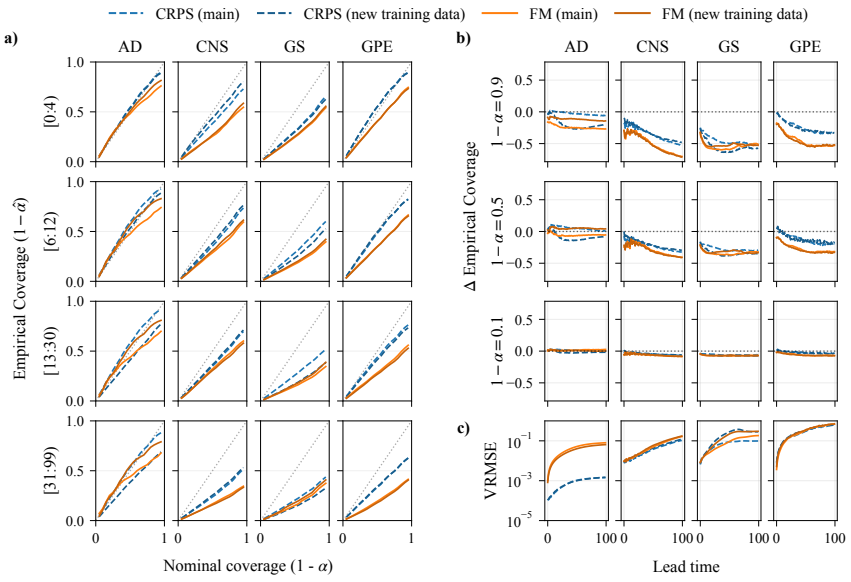


Figure 22: Training-data sensitivity of the main CRPS and FM models across datasets (AD, CNS, GS, GPE). Each rerun uses a new, independently generated training set but the same test trajectories as the main models. (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time.

The paired single-step results in Table 10 are close across all four datasets, and the independently trained models track their main-run counterparts over much of the rollout in Figure 22. For CRPS on CNS, coverage MAE decreases from 0.124 to 0.111, while VRMSE and CRPS increase slightly; the FM single-step metrics also change only modestly across the four systems. Differences become more visible in some long-horizon curves, particularly GS VRMSE, so individual rollout trajectories are

not invariant to the sampled training set. Nevertheless, the broad calibration and accuracy comparison between CRPS and FM remains consistent across the four controlled reruns.

F.14 MC DROPOUT TRAINED WITH CRPS LOSS

The main CRPS-trained ensemble samples Gaussian noise and injects it through CLN. In this ablation, we replace that stochastic modulation with dropout in the feed-forward blocks of the ViT backbone. Specifically, an elementwise Bernoulli mask is applied after the activation and before the second linear map in the feed-forward branch of every transformer block. Each feed-forward network expands the token width from $d = 704$ to $4d = 2816$ before the activation. For a processor batch of size B and $L = 16 \times 16 = 256$ spatial patch tokens, the corresponding mask in block j is therefore $\boldsymbol{\xi}_j \in \{0, 1\}^{B \times L \times 4d}$; $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{12})$ denotes the collection for one processor evaluation. Dropout is applied with probability $p = 0.1$, remains active at evaluation time, and $\boldsymbol{\xi}$ is resampled independently on every forward call and at every autoregressive rollout step. Removing the 1024-dimensional modulation path reduces the processor size, so we increase the ViT width from 568 to 704 while retaining 12 blocks and 8 heads, giving 79.8M parameters compared with 80.8M for the main ViT.

Training otherwise follows the ambient-space CRPS protocol used throughout: eight predictions with independently sampled mask collections are evaluated for each of 32 cases per GPU update and optimised with αfCRPS. Evaluation uses ten stochastic forward passes. This treats MC dropout as a shared-parameter ensemble sampler (Gal & Ghahramani, 2016); closely related weather-forecasting work has likewise trained dropout samples using a CRPS-family objective (Rühling Cachay et al., 2026; Diaconu et al., 2026b).

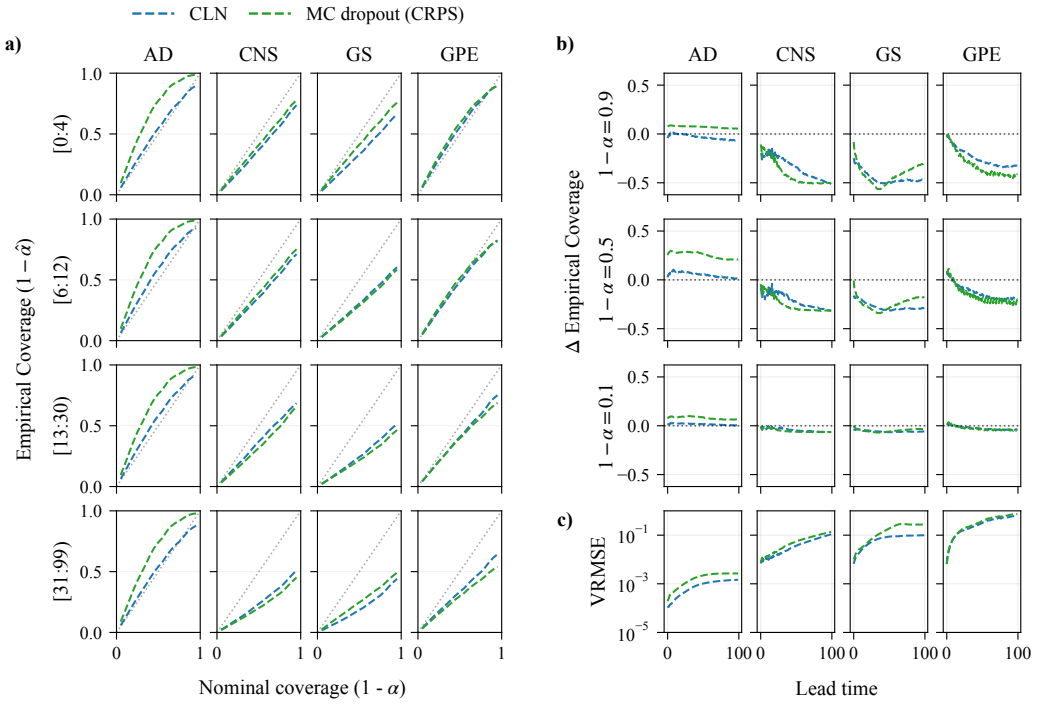


Figure 23: CLN vs. MC dropout as the stochastic mechanism for CRPS-trained ensembles across datasets (AD, CNS, GS, GPE). (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time. MC dropout changes coverage in a dataset-dependent manner.

Table 11: Single-step comparison of CLN and MC dropout as ensemble mechanisms under CRPS training. Bold denotes the better result for each dataset.

Dataset	Mechanism	VRMSE ↓	Coverage MAE ↓	CRPS ↓	SSR → 1
AD	CLN	**1.8e-04**	**0.04**	**8.8e-05**	**1.07**
	MC dropout (CRPS)	2.8e-04	0.15	1.8e-04	2.16
CNS	CLN	**5.6e-03**	0.12	**2.7e-03**	1.16
	MC dropout (CRPS)	6.2e-03	**0.09**	3.0e-03	**1.10**
GS	CLN	**1.1e-03**	0.20	**5.2e-04**	**0.67**
	MC dropout (CRPS)	1.4e-03	**0.04**	6.3e-04	1.55
GPE	CLN	**2.8e-02**	0.04	**9.7e-03**	**0.92**
	MC dropout (CRPS)	3.2e-02	**0.03**	1.2e-02	0.84

Table 11 and Figure 23 show that neither stochastic mechanism dominates coverage across datasets and lead times, whereas CLN has lower single-step VRMSE and CRPS on all four datasets and generally retains lower rollout VRMSE. For single-step prediction, MC dropout improves coverage MAE on GS and worsens it on AD, with smaller improvements on CNS and GPE. A broader hyperparameter study would be needed to determine whether either mechanism is generally preferable.

F.15 MC DROPOUT TRAINED WITH MSE LOSS

We evaluate conventional MC dropout trained with L_2-regularised MSE loss using the same parameter-matched ViT as in Appendix F.14. Dropout remains restricted to the feed-forward branches with $p = 0.1$, using the mask collection $\boldsymbol{\xi}$ defined above, but training uses one independently sampled $\boldsymbol{\xi}$ per case. The per-GPU batch size increases from 32 to 256, keeping 256 processor evaluations per update. Using the processor f_ρ and its parameter collection ρ defined in Section 3.4, let $\rho_{\geq 2} \subset \rho$ contain the trainable processor parameters with at least two dimensions. The training objective is

$$\mathcal{L}_{\mathrm{MSE+L2}}(\rho; \boldsymbol{\xi}) = \mathrm{MSE}(f_\rho(\boldsymbol{x}_{\mathrm{in}}, \boldsymbol{c}; \boldsymbol{\xi}), \boldsymbol{x}_{\mathrm{out}}) + \lambda \sum_{P \in \rho_{\geq 2}} \|P\|_F^2, \qquad \lambda = 10^{-5}. \tag{15}$$

For this conventional MC-dropout baseline, we adopt dropout probability $p = 0.1$ and regularisation coefficient $\lambda = 10^{-5}$, matching the released WeatherBench Probability configuration (Garg et al., 2022). Biases and normalisation vectors are excluded from the penalty. AdamW weight decay is zero, and the L_2 penalty is applied only during training; validation and test loss use plain MSE. At evaluation, ten stochastic forward passes form the predictive ensemble.

Table 12: Single-step comparison of the main CRPS and latent FM models with conventional MC dropout trained using MSE and explicit L_2 regularisation. Bold denotes the best result for each dataset.

Dataset	Model	VRMSE ↓	Coverage MAE ↓	CRPS ↓	SSR → 1
AD	CRPS	**1.8e-04**	0.04	**8.8e-05**	1.07
	FM (latent)	2.2e-04	**0.02**	1.3e-04	1.31
	MC dropout (MSE)	4.8e-03	0.07	2.6e-03	**1.05**
CNS	CRPS	**5.6e-03**	**0.12**	**2.7e-03**	**1.16**
	FM (latent)	8.7e-03	0.28	3.9e-03	0.65
	MC dropout (MSE)	1.9e-02	0.19	9.2e-03	0.68
GS	CRPS	**1.1e-03**	0.20	5.2e-04	0.67
	FM (latent)	1.3e-03	0.18	**5.1e-04**	0.57
	MC dropout (MSE)	1.1e-02	**0.06**	3.9e-03	**0.93**
GPE	CRPS	**2.8e-02**	**0.04**	**9.7e-03**	0.92
	FM (latent)	3.2e-02	0.24	1.1e-02	0.44
	MC dropout (MSE)	3.6e-02	**0.04**	1.6e-02	**0.98**

Conventional MC dropout has the highest single-step VRMSE and CRPS on every dataset (Table 12). Its competitive coverage MAE on GS and GPE does not imply accurate or sharp predictions.

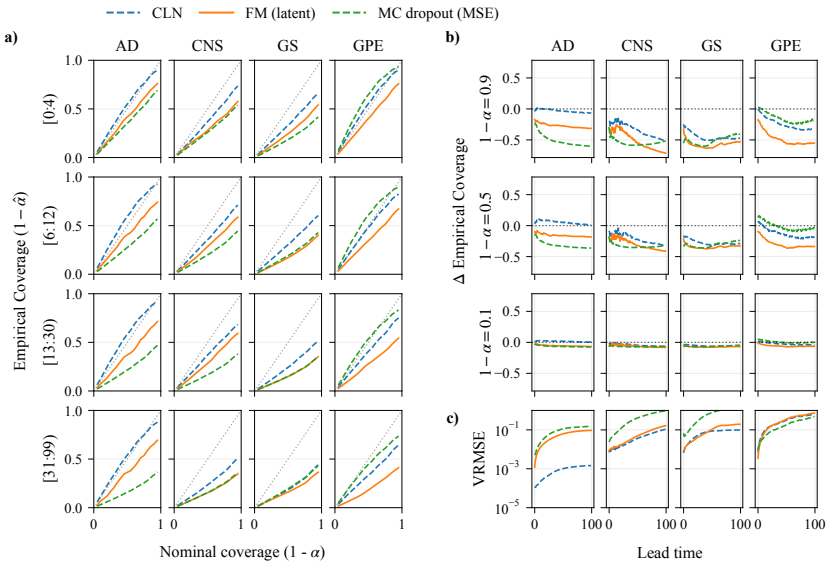


Figure 24: Conventional MC dropout trained with MSE and explicit L_2 regularisation compared with the main CRPS and latent FM models across datasets (AD, CNS, GS, GPE). (a) Empirical vs. nominal coverage averaged over rollout windows. (b) Δ empirical coverage (observed minus nominal) by lead time over 100-step rollouts. (c) VRMSE by lead time. MC dropout with MSE is less accurate and reliable overall, despite competitive coverage for GPE.

This pattern generally persists over rollouts, with GPE the main exception (Figure 24). Overall, the baseline is not consistently competitive with either probabilistic training approach.

F.16 POST-HOC UNCERTAINTY CALIBRATION

Conformal prediction (CP) turns a heuristic measure of predictive uncertainty into prediction intervals with a finite-sample coverage guarantee (Angelopoulos & Bates, 2021; Angelopoulos et al., 2026). In its split form, it needs only a trained model and a calibration set that the model has not seen: the model's errors on the calibration set, measured relative to its own uncertainty estimate, determine how much its intervals must be widened or narrowed. Conformal prediction has previously been applied to neural surrogates of PDEs (Gopakumar et al., 2026; Ma et al., 2024).

We apply split conformal prediction to the 100-step rollouts of the main CRPS and FM models without retraining them. For each dataset, we simulated 150 additional trajectories with the same `AutoSim` settings as the datasets of the main results but different random seeds. Rollouts for these trajectories and for the test set of the main results were generated with the checkpoints and evaluation settings of the main results, using ten ensemble members. Of the 150 new trajectories, $n = 100$ ($n = 102$ for GS, 17 of each of the six pattern types) form the calibration set, and the remaining 50 (48 for GS) are held out for the calibration-set-size check below. All other results are evaluated on the test set of the main results (20 trajectories, 24 for GS).

The procedure is applied separately to every output component r, i.e. every lead time, grid point and channel, as in Section 3.6. For calibration trajectory $i = 1, \dots, n$, let $x_{i,r}$ be the true value of component r, and $\hat{\mu}_{i,r}$ and $\hat{\sigma}_{i,r}$ the mean and standard deviation of the ensemble members. Following Angelopoulos & Bates (2021), the score is the error of the ensemble mean in units of the ensemble spread,

$$s_{i,r} = \frac{|x_{i,r} - \hat{\mu}_{i,r}|}{\hat{\sigma}_{i,r}}.$$

For nominal coverage $1 - \alpha$, the threshold $\hat{q}_r$ is the $\lceil (n+1)(1-\alpha) \rceil$-th smallest of $s_{1,r}, \dots, s_{n,r}$. For a test trajectory with ensemble mean $\hat{\mu}_r$ and standard deviation $\hat{\sigma}_r$, the conformal interval is

$$\left[\hat{\mu}_r - \hat{q}_r \hat{\sigma}_r,\ \hat{\mu}_r + \hat{q}_r \hat{\sigma}_r\right].$$

The interval is centred on the ensemble mean and proportional to the ensemble spread, and $\hat{q}_r$ sets its width so that a fraction of about $1 - \alpha$ of the calibration errors for component r fall inside it. If the calibration and test trajectories are exchangeable, i.e. their joint distribution does not change when they are reordered, which holds if both are independent draws from the same simulator settings, the interval contains the truth with probability at least $1 - \alpha$, and at most $1 - \alpha + 1/(n+1)$ when the scores have no ties (Angelopoulos & Bates, 2021). The guarantee holds for each component marginally, not jointly over the field. It holds for any underlying model, but the intervals are only as informative as the uncertainty estimate they start from: a poor estimate gives wide intervals. We compute the intervals at each nominal level used in the main results, from 0.05 to 0.95, and use them in place of the ensemble quantiles when computing empirical coverage (Section 3.6) and the Winkler interval score (Section 3.5). The calibration is implemented in `AutoUQ`[5], an open-source Python package for evaluating and calibrating uncertainty estimates.

To check that 100 calibration trajectories are enough, we repeat the calibration on random subsets of $K \in \{9, 12, 15, 20, 30, 50, 75, 100\}$ trajectories drawn from the calibration set, with 20 draws for each K, and evaluate on the remaining 50 new trajectories (48 for GS). Figure 25 shows 90% coverage and the Winkler interval score relative to its value at $K = 100$, each averaged over the held-out trajectories, grid points, channels and all 100 lead times, as the mean and standard deviation over the 20 subsets. Coverage stays between 0.87 and 0.96 for every K, so the number of calibration trajectories mainly determines how wide the intervals are rather than whether they cover. For $9 \le K \le 18$, $\lceil 0.9(K+1) \rceil = K$, so the threshold is the largest calibration score ($K = 9$ is the smallest calibration set for which a 90% threshold exists) and the interval score reaches up to twice its $K = 100$ value on AD, CNS and GPE, and 8–24 times on GS, where it also varies strongly between subsets, with a standard deviation comparable to or larger than the mean. By $K = 50$, the interval score on AD, CNS and GPE is within 4% of its value at $K = 100$. On GS it is still 13–19% higher at $K = 75$ and has not levelled off, so additional calibration trajectories would narrow the GS intervals further, whereas 100 trajectories are sufficient for the other three datasets. These sizes are specific to these four datasets and are not a general recommendation for how many calibration trajectories conformal prediction requires.

[5] `https://anonymous.4open.science/r/autouq-3CFC`

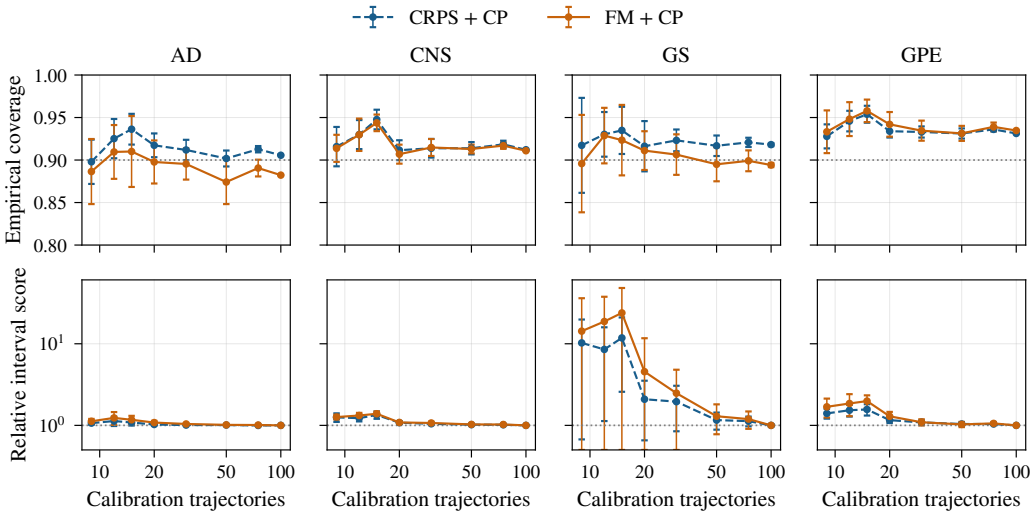


Figure 25: Coverage at $1 - \alpha = 0.9$ (top) and relative Winkler interval score (bottom) against the number of calibration trajectories K, on the held-out new trajectories, averaged over the 100-step rollout. Mean and standard deviation over 20 random calibration subsets; bars below the bottom axes are cut. Dashed blue: CRPS; solid orange: FM.

Figure 26 shows the effect of conformal calibration on the test set. It compares the empirical coverage of the raw ensembles with that of the conformal intervals across nominal levels in each rollout window (a), and shows how far each lies from nominal coverage at every lead time (b). Averaged over test trajectories, grid points, channels and all 100 lead times, 90% coverage of the raw ensembles ranges from 0.43 (GS) to 0.87 (AD) for CRPS and from 0.34 (GS) to 0.65 (AD) for FM. After conformal calibration it lies between 0.90 and 0.92 on AD, CNS and GPE for both models, and the Winkler interval score decreases, apart from CRPS on AD, where it is unchanged. The shortfall of the raw ensembles grows with lead time: at the 90% level it widens from 0.17 below nominal at the first lead time to 0.53 below at the last for CRPS on CNS, and from 0.30 to 0.71 below for FM, with similar growth on GS and GPE. At the 10% level, raw coverage stays within 0.09 of nominal for every model. At the 90% level, the conformal intervals stay within 0.06 of nominal at every lead time on AD, CNS and GPE. On GS, the conformal intervals over-cover the test set (0.99 for CRPS and 0.98 for FM), above the upper bound of 0.91 for $n = 102$, whereas the same calibration gives 0.92 and 0.89 on the held-out new GS trajectories (Figure 25, $K = 100$). Per trajectory, most held-out new GS trajectories are also covered at 0.96 or more, but a few (4 of 48 for CRPS and 5 of 48 for FM, all with maze, spiral or spot patterns) have coverage below 0.8 and bring the average down to about 0.9. The 24 test trajectories include none of these, which at the rate seen on the held-out set happens by chance roughly one time in ten, so the over-coverage is consistent with the small size of the test set (the bound applies to the expected coverage, not to that of one particular set of test trajectories) rather than with a difference between the two sets. The guarantee holds on average over the six parameter settings of GS, not for each one: for CRPS, coverage on the held-out trajectories ranges from 0.69 for maze to 1.00 for gliders.

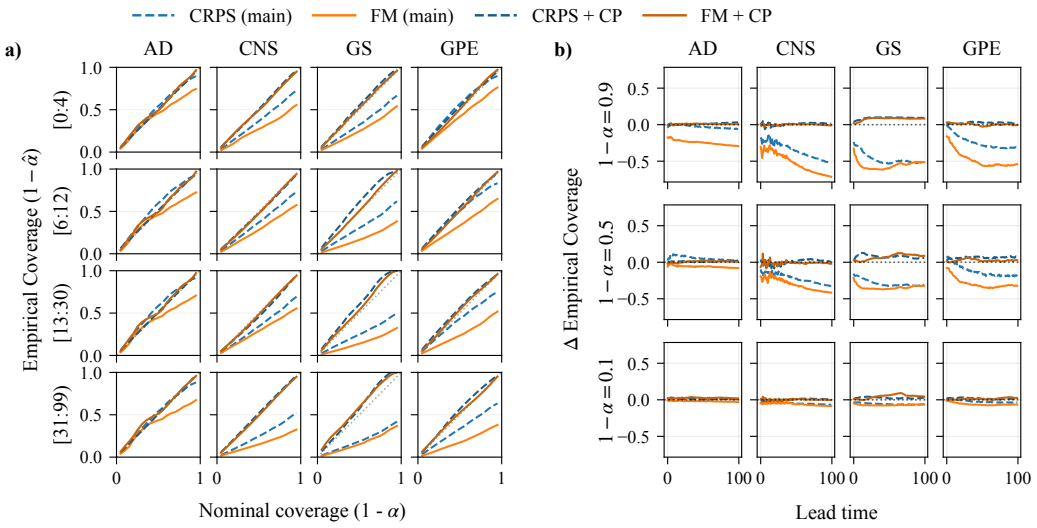


Figure 26: Coverage of the main CRPS (dashed blue) and latent FM (solid orange) models before and after conformal prediction (CP; darker shades), fitted on the new calibration set. Each pair uses the same re-evaluated paper test forecasts. (a) Empirical against nominal coverage in each rollout window, as in Figure 4. (b) Empirical minus nominal coverage at each lead time for nominal coverage 0.9, 0.5 and 0.1; zero (dotted line) is ideal.

Figures 27 and 28 show where on the grid the intervals are too narrow or too wide, which coverage averaged over the whole field can hide. Each panel shows 90% coverage at every grid point on the test set, averaged over test trajectories, channels and the lead times in one rollout window, for the raw ensemble (left four columns) and the conformal intervals (right four columns). With 20 test trajectories (24 for GS), differences of a few hundredths between grid points are expected from sampling alone. The raw ensembles under-cover almost everywhere on CNS, GS and GPE, increasingly so in later windows, and the FM ensemble also under-covers on AD. After conformal calibration, most grid points are close to nominal coverage on AD, CNS and GPE, with some remaining spatial structure, such as a band of mild over-coverage (about 0.95) across the centre of the GPE domain before the last window. On GS, the conformal intervals over-cover across the whole domain, consistent with the field-averaged results above.

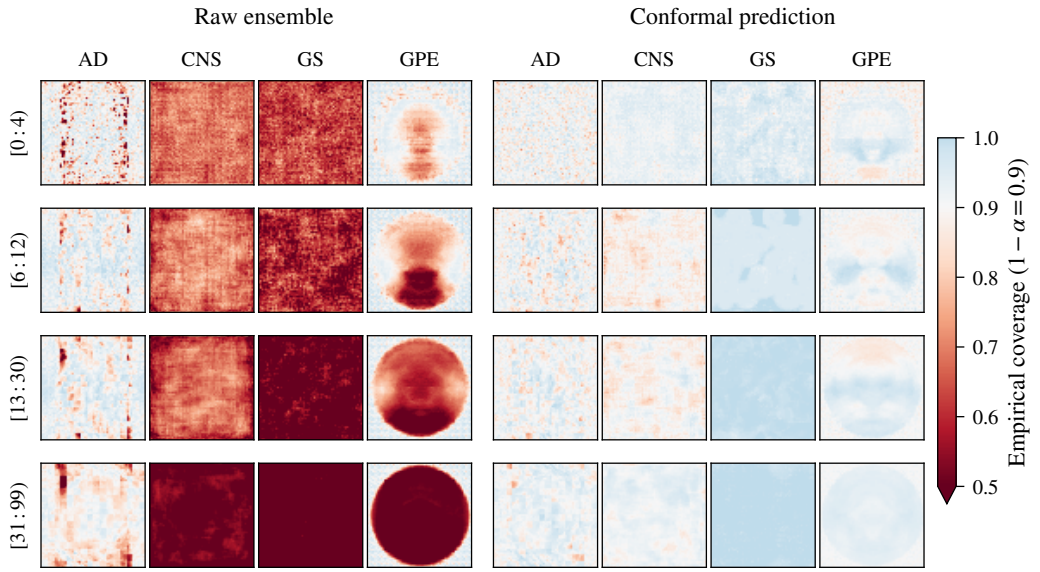


Figure 27: Coverage at $1 - \alpha = 0.9$ at each grid point for the main CRPS model on the test set, averaged over test trajectories, channels and each rollout window: raw ensemble (left) and conformal prediction (right). The colour scale is linear and white at the nominal 0.9, with red for under-coverage and blue for over-coverage; values below 0.5 share the darkest red.

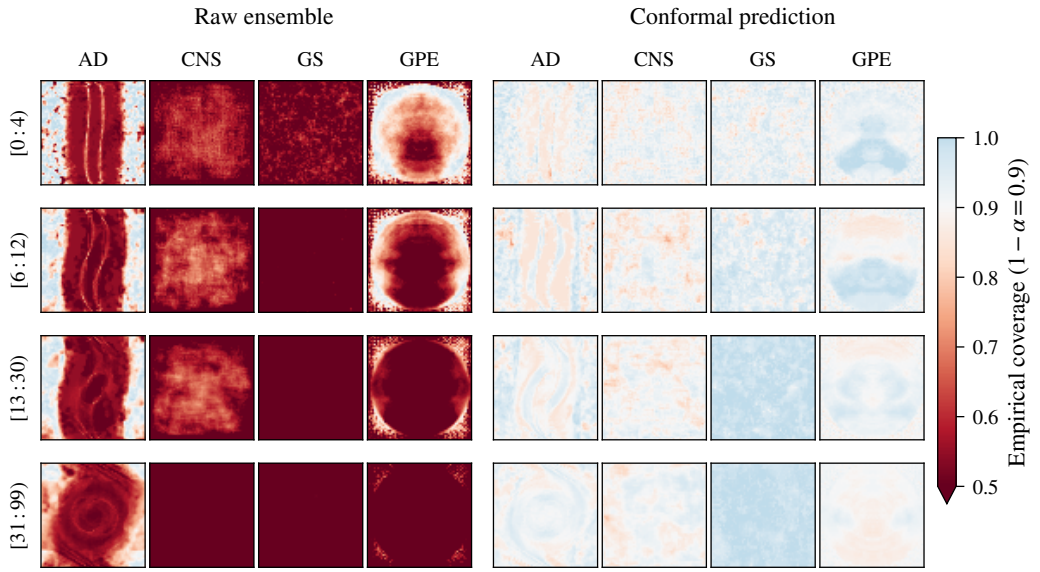


Figure 28: Coverage at $1 - \alpha = 0.9$ at each grid point for the main latent FM model on the test set, averaged over test trajectories, channels and each rollout window: raw ensemble (left) and conformal prediction (right). The colour scale is linear and white at the nominal 0.9, with red for under-coverage and blue for over-coverage; values below 0.5 share the darkest red.

Figure 29 shows the width of the 90% intervals at every grid point at lead time 60 for one representative test trajectory per dataset: the conformal intervals keep the spatial structure of the raw ensemble intervals but are wider. Over the whole test set, taking the ratio of conformal to raw width at every grid point, channel, lead time and trajectory, the median ratio is 1.1 for CRPS and 1.7 for FM on

AD, 2.9 and 6.7 on CNS, 3.4 and 5.9 on GPE, and 18 and 8.4 on GS, and the ratio grows with lead time. On GS and GPE, the conformal interval is wider than the range of the true field at 2–8% of grid points.

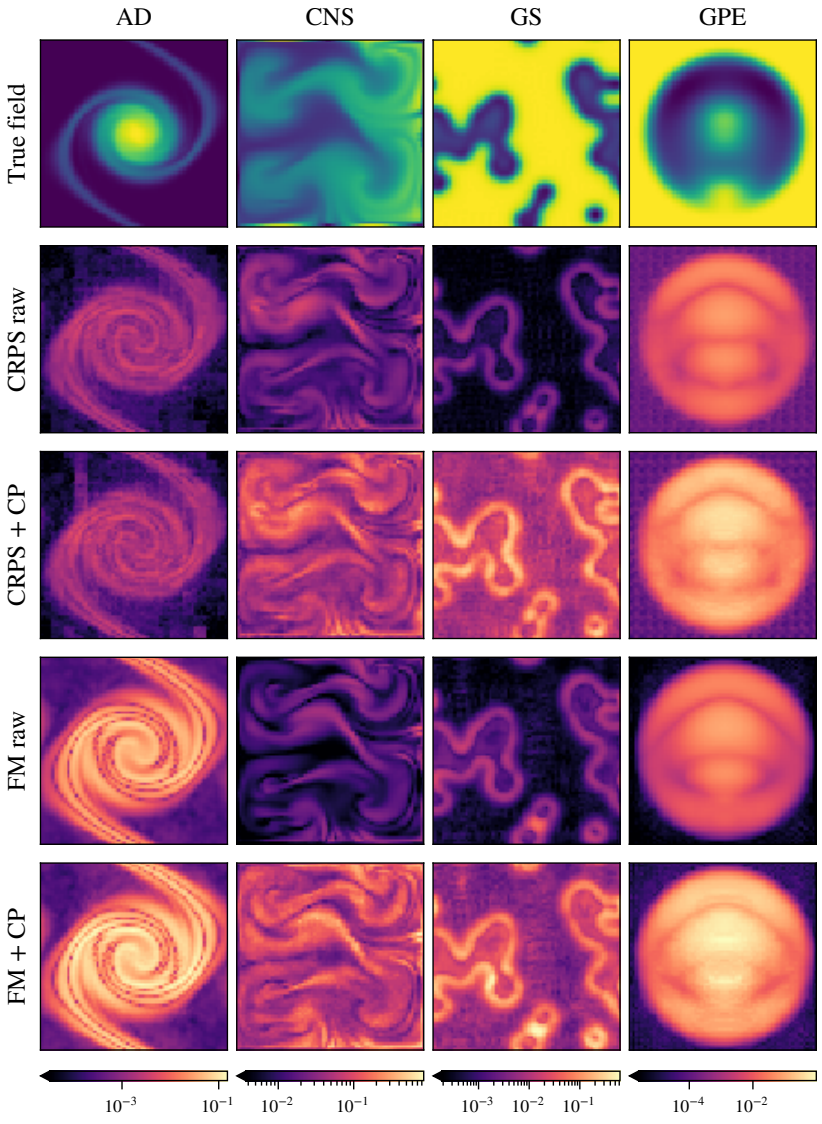


Figure 29: Width of the 90% intervals at each grid point at lead time 60 (first channel) for one test trajectory per dataset: the one whose median ratio of conformal to raw width is closest to the median over test trajectories, for both models. Top row: true field. Colour scales are logarithmic and shared within each column.