CONTINUOUS ENSEMBLE WEATHER FORECASTING WITH DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Weather forecasting has seen a shift in methods from numerical simulations to data-driven systems. While initial research in the area focused on deterministic forecasting, recent works have used diffusion models to produce skillful ensemble forecasts. These models are trained on a single forecasting step and rolled out autoregressively. However, they are computationally expensive and accumulate errors for high temporal resolution due to the many rollout steps. We address these limitations with Continuous Ensemble Forecasting, a novel and flexible method for sampling ensemble forecasts in diffusion models. The method can generate temporally consistent ensemble trajectories completely in parallel, with no autoregressive steps. Continuous Ensemble Forecasting can also be combined with autoregressive rollouts to yield forecasts at an arbitrary fine temporal resolution without sacrificing accuracy. We demonstrate that the method achieves competitive results for global weather forecasting with good probabilistic properties.

1 INTRODUCTION

Forecasting of physical systems over both space and time is a crucial problem with plenty of real-world applications, including in the earth sciences, transportation, and energy systems. A prime example of this is weather forecasting, which billions of people depend on daily to plan their activities. Weather forecasting is also crucial for making informed decisions in areas such as agriculture, renewable energy production, and safeguarding communities against extreme weather events. Current Numerical weather prediction (NWP) systems predict the weather using complex physical models and large supercomputers (Bauer et al., 2015). Recently Machine learning weather prediction (MLWP) models have emerged, rivaling the performance of existing NWP systems. These models are not physics-based but data-driven and have been made possible thanks to advancements in deep learning. By analyzing patterns from vast amounts of meteorological data (Hersbach et al., 2020), MLWP models now predict the weather with the same accuracy as global operational NWP models in a fraction of the time (Kurth et al., 2023; Lam et al., 2023; Bi et al., 2023).

Following the success of deterministic MLWP models, the focus of the field has increasingly shifted towards probabilistic modeling. The probabilistic models generate samples of possible future weather trajectories. By drawing many such samples, referred to as ensemble members, it is possible to generate a set of possible forecasts, referred to as an ensemble forecast, for quantifying forecast uncertainty and detecting extreme events (Leutbecher & Palmer, 2008). Sampling forecasts from deep generative models also address the blurriness often observed in predictions from deterministic MLWP, yielding forecasts that better preserve the variability of the modeled entities. A popular class of deep generative models used for probabilistic MLWP are diffusion models (Ho et al., 2020; Price et al., 2024; Lang et al., 2024; Shi et al., 2024). While these models generate accurate and realistic looking forecasts, they are computationally expensive due to requiring multiple sequential forward passes through the neural network to generate a sample. Moreover, they are often applied iteratively to roll out longer forecasts (Price et al., 2024), which exacerbates the computational issue. Naively switching out this iterative rollout to directly forecasting each future time step does not result in trajectories that are consistent over time. The auto-regressive rollout additionally puts some limitations on the temporal resolution of the forecast. Taking too small timesteps results in large accumulation of error over time (Bi et al., 2023), which forces existing models to resort to a temporal resolution of 12 h Price et al. (2024); Lang et al. (2024). However, in many situations it is crucial to obtain probabilistic forecasts at a much higher temporal resolution. This is true not

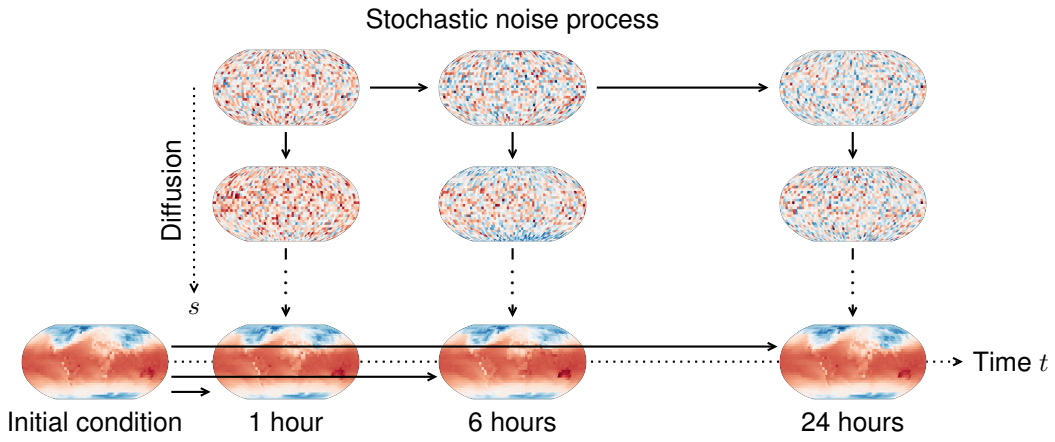

Figure 1: Our proposed framework, Continuous Ensemble Forecasting, generates ensemble weather forecasts using a conditional diffusion model. The model takes lead time as input and forecasts the future weather state in a single step, e.g. the forecast at 24 hours is generated directly from the initial condition, without seeing the intermediate predictions. To ensure temporal consistency, we correlate the driving noises for the different lead times. This can be done by fixing the noise, or by defining a stochastic noise process. Repeating the procedure for different starting noise gives an ensemble of forecasts. Using this framework we can generate 10 day forecasts with 1 hour resolution without sacrificing performance.

least when the forecasts are used as decision support in extreme weather situations and for capturing rapidly changing weather events.

We propose a continuous forecasting diffusion model that takes lead time as input and forecasts the future weather state in a single step, while maintaining a temporally consistent trajectory for each ensemble member.[1] This enables both autoregressive and direct forecasting within a single model, improves the accuracy compared to purely autoregressive models at high temporal resolutions, and enables forecasting at arbitrary (non-equidistant) lead times throughout the forecast trajectory. To generate ensemble forecasts, the model uses a deterministic ODE-solver to solve the lead-time-dependent probability flow ODE starting in different pure noise samples. To ensure temporal consistency, we correlate the driving noises for the different lead times, e.g., by using a single noise sample for all timesteps, as illustrated in fig. 1, which enables generating a continuous trajectory for each member. This enables parallel sampling of individual ensemble members, bypassing the need for multi-step loss functions, and thus accelerating ensemble forecasting.

Contributions. We propose a novel method for ensemble weather forecasting built on diffusion that: 1) can generate ensemble member trajectories without iteration, 2) can forecast arbitrary lead-times, 3) can be used together with iteration to improve performance on long rollouts, 4) achieves competitive performance in global weather forecasting.

2 Related Work

Ensembles from perturbations. In NWP, ensemble forecasts are created by perturbing initial conditions or model parameterizations. The same idea has been applied in MLWP using initial state perturbations (Bi et al., 2023; Kurth et al., 2023; Chen et al., 2023b; Li et al., 2024), model parameter perturbations (Hu et al., 2023; Weyn et al., 2021) and Monte Carlo dropout (Scher & Messori, 2021; Hu et al., 2023). In all these cases the underlying model is still deterministic, trained with a mean squared error loss function to predict the average future weather. These ensembles can thus be viewed as a mixture of means, inheriting the spatial oversmoothing characteristic of deterministic forecasts.

[1] Our code will be made openly available at publication.

Generative models. Another way of getting ensembles is by modeling the distribution directly through generative models. Price et al. (2024) train a graph-based diffusion model to produce 15-day global forecasts, at 12-hour steps and 0.25°resolution. Their GenCast model generates realistic forecasts but is slow to sample from, requiring sequential forward passes through the network both for sampling each time step and to roll out the forecast over time. Latent-variable-based MLWP models (Hu et al., 2023; Oskarsson et al., 2024; Zhong et al., 2024) can perform inference in a single forward pass, but have been shown to suffer from more blurriness compared to diffusion models and require delicate hyperparameter tuning (Oskarsson et al., 2024). Apart from forecasting, diffusion models have also been successfully applied to other weather-related tasks, including downscaling (Chen et al., 2023a; Mardani et al., 2024; Pathak et al., 2024), enlarging physics-based ensembles (Li et al., 2024) and generating realistic weather from climate scenarios (Bassetti et al., 2023). However, it is important to note that since the output from these generative models does not directly influence the forecasting process, there are no guarantees that the resulting dynamics will remain continuous over time. Hua et al. (2024) explore the possibility to incorporate prior information in MLWP diffusion forecasting, by guidance from existing NWP forecasts or climatology, but they do not consider ensemble forecasting.

Temporal resolution. Unlike in NWP, where stability conditions dictate the time step, MLWP models are free to predict at any temporal resolution. Still, the most common approach is to learn to forecast a single short time-step (6h) and iterate this process until the desired lead time (Lam et al., 2023; Chen et al., 2023b). Although intuitive, this process can lead to error accumulation and is impossible to parallelize due to its sequential nature (Bi et al., 2023). Multi-step losses have been shown to reduce the error accumulation for deterministic (Lam et al., 2023) and latent-variable based models (Oskarsson et al., 2024), but are not trivially implemented in diffusion models. Taking longer timesteps (24h) has been shown to give better results (Couairon et al., 2024; Bi et al., 2023), but comes at the loss of temporal resolution. Bi et al. (2023) resolves this by training multiple models to forecast different lead times, which are then combined in different ways to reach the lead times of interest. Nguyen et al. (2023a;b) use a similar setup, but train a single model taking the forecast lead time as an input. This continuous forecasting parallelizes the prediction of the fine temporal scales, but has so far only been applied to deterministic models. Other approaches have tried to learn fully time-continuous dynamics by using an ODE to generate forecasts (Verma et al., 2024; Saleem et al., 2024; Rühling Cachay et al., 2023; Kochkov et al., 2024).

Spatio-temporal forecasting with diffusion models. Outside of MLWP, diffusion models have also been applied to forecasting other spatio-temporal processes (Yang et al., 2024). Notable examples include turbulent flow simulation (Kohl et al., 2024; Rühling Cachay et al., 2023) and PDE solving (Lippe et al., 2023). Yang & Sommer (2023) apply diffusion models conditioned on the prediction lead time to a specific floating-smoke fluid field, but do not consider ensemble forecasting. Another alternative to autoregressive rollouts is the DYfussion framework (Rühling Cachay et al., 2023), where stochastic interpolation and deterministic forecasting is combined into a diffusion-like model. The method allows for forecasting at arbitrary temporal resolution, but still requires sequential computations for sampling the prediction. DYffusion has been successfully applied to climate modeling (Cachay et al., 2024), but not weather forecasting.

3 BACKGROUND

Problem statement. This paper targets the global weather forecasting problem, an initial-value problem with intrinsic uncertainty. Consider a weather state space $\mathcal{X}$ containing the grid of target variables. Given information about previous weather states $X(\Omega) \subset \mathcal{X}$ at times $\Omega \subset (-\infty, 0]$, the aim is to forecast a trajectory $X(\mathcal{T})$ of future weather states $X : (-\infty, \infty) \to \mathcal{X}$ at times $\mathcal{T} \subset (0, T]$ for some time horizon T. In particular, the task is to learn and sample from the conditional distribution $p(X(\mathcal{T})|X(\Omega))$.

Autoregressive forecasting. To simplify the problem, $\mathcal{T}$ is often chosen as a set of discrete equally spaced times $\{k\delta\}_{k=1}^{N}$ for some timestep δ. By choosing $\Omega = \{-k\delta\}_{k=0}^{M}$ and assuming Mth order Markovian dynamics, the joint distribution of $X[k] := X(k\delta)$ can be factorized over successive

states,

$$p(X[1{:}N]|X[-M{:}0]) = \prod_{k=0}^{N-1} p(X[k{+}1]|X[k{-}M{:}k]), \tag{1}$$

and the forecasts can be sampled autoregressively. This way, the network only has to learn to sample a single step $p(X[k{+}1]|X[k{-}M{:}k])$.

Conditional Diffusion Models. Similarly to Price et al. (2024), we model $p(X[k{+}1]|X[k{-}M{:}k])$ using a conditional diffusion model (Ho et al., 2020; Song et al., 2021; Karras et al., 2022). Diffusion models generate samples by iteratively transforming noise into data. To forecast a future weather state $X[k{+}1]$ given $X[k{-}M{:}k]$ we sample a latent noise variable from p_{noise} and iteratively transform it until it resembles a sample from $p(X[k{+}1]|X[k{-}M{:}k])$. We consider the SDE formulation of diffusion models presented by Karras et al. (2022), but remark that our framework generalizes to any diffusion or flow matching framework based on stochastic differential equations. Sampling can then be done by solving the probability flow ODE

$$\mathrm{d}\boldsymbol{z}(s) = -\dot{\sigma}(s)\sigma(s)S_\theta(\boldsymbol{z}(s); X[k{-}M{:}k], \sigma(s))\mathrm{d}s, \quad s \in [0,1] \tag{2}$$

starting in pure noise $\boldsymbol{z}(1) \sim p_{\text{noise}}$ and ending in our forecast $\boldsymbol{z}(0) \sim p(X[k{+}1]|X[k{-}M{:}k])$. Here S_θ is the neural network trained to match the score function S through the denoising training objective as presented by Karras et al. (2022). Similar to (Karras et al., 2022), we choose $\sigma(s) = s$ and feed the noise level σ to S_θ in each layer as a Fourier embedding. Repeating this process for different noise samples $\boldsymbol{z}(1)$ gives an ensemble of forecasts. For a more detailed description of diffusion models see appendix B.

4 Continuous Ensemble Forecasting

Autoregressive forecasting models are simple to train but can suffer from error accumulation at long horizons. Consider again the general problem of sampling from $p(X(\mathcal{T})|X(\Omega))$. In continuous forecasting, a single-step prediction is given by conditioning on the lead time $t \in \mathcal{T}$ and training a conditional score network $S_\theta(\boldsymbol{z}; X(\Omega), t, \sigma)$ to simulate directly from the marginal distribution $p(X(t)|X(\Omega), t)$. This does not require setting a fixed δ, making the setup more flexible. The lead time t is added to the conditioning arguments for clarity and can be passed to the network in the same way as the noise level σ. While this allows sampling a distribution of states $X(t)$ at each time $t \in \mathcal{T}$, combining these naively does not result in a trajectory $X(\mathcal{T})$. This is because $X(t)$ are samples from the marginal distributions $p(X(t)|X(\Omega), t)$ and not the joint trajectory distribution $p(X(\mathcal{T})|X(\Omega))$. We propose *Continuous Ensemble Forecasting*, a novel method of combining samples from $p(X(t)|X(\Omega), t)$ into forecast trajectories $X(\mathcal{T})$ without resorting to autoregressive predictions.

The core idea in our method is to control the source of randomness. To sample from $p(X(t)|X(\Omega), t)$, we sample some noise $Z \sim p_{\text{noise}}$ and feed it to an ODE-solver that solves the probability flow ODE, giving us a forecast $X(t)$ for lead time t. Since the ODE-solver is deterministic, the randomness is limited to the noise initialization Z. If we freeze the noise, the ODE-solver becomes a deterministic map $f_\theta^Z : \mathcal{X}^{|\Omega|} \times \mathcal{T} \to \mathcal{X}$ parameterized by the neural network S_θ. Applying this map to previous states $X(\Omega)$ and a time $t \in \mathcal{T}$ gives a forecast $X(t)$. Repeating this for several $t_1, \ldots, t_N \subset \mathcal{T}$ gives a sequence of forecasts $X(\{t_i\}_{i=1}^N)$. Extending this to all $t \in \mathcal{T}$, we can construct a trajectory $X(\mathcal{T}) = f_\theta^Z(X(\Omega), \mathcal{T})$. We treat this as a sample from $p(X(\mathcal{T})|X(\Omega))$ and propose to use Algorithm 1 to sample such trajectories.

Algorithm 1 The Continuous Ensemble Forecasting algorithm

1: **input:** Initial conditions $x(\Omega)$, times $\{t_i\}_{i=1}^N$, ensemble size n_{ens}, network S_θ
2: **sample** $\{z^j\}_{j=1}^{n_{\text{ens}}} \sim \mathcal{N}(0, \mathbf{I})$
3: **for all** $i \in \{1, \ldots, N\}, j \in \{1, \ldots, n_{\text{ens}}\}$ **do** ▷ Can be done fully in parallel for all j and i
4: $\quad x_i^j \leftarrow$ Probability-Flow-Solver$(z^j, t_i; x(\Omega), S_\theta)$
return $\{x_i^j\}_{i=1:N}^{j=1:n_{\text{ens}}}$

4.1 MATHEMATICAL MOTIVATION

In a deterministic system, the dynamics can be described by a forecasting function $f : \mathcal{X}^{|\Omega|} \times \mathcal{T} \to \mathcal{X}$ that maps previous states $X(\Omega) \in \mathcal{X}^{|\Omega|}$, to future states $X(t)$. While weather is in principle governed by deterministic equations, its chaotic nature, lack of information, and our inability to resolve the dynamics at sufficiently high spatio-temporal resolution gives it an intrinsic uncertainty. This motivates the formulation of weather as a stochastic dynamical system. In such a system, no function can describe the entire dynamics. At each instance, there might be a range of functions $f^1, f^2, \ldots$ that all describe some possible evolution, some more likely than others. If we consider the space $\mathcal{F}$ of all such functions, we can formalize this by defining a probability density μ over this space, describing the likelihood of each function. To create an ensemble of possible functions, we sample several $f^1, \ldots, f^N$, from μ. Given previous states $X(\Omega)$, evaluating each function gives an ensemble of possible trajectories $X^i(\mathcal{T}) = f^i(X(\Omega), \mathcal{T})$.

The key insight in motivating our method is the identification of the latent noise space $(\mathcal{X}, p_{\text{noise}})$ with the solution space $(\mathcal{F}, \mu)$. In our method, the sampling algorithm can be represented by a parameterized function f_θ^Z, where we have frozen the noise $Z \sim p_{\text{noise}}$. Under the regularity conditions specifed in E, this function is uniquely defined by θ and the ODE-solver for any given Z. Thus, as illustrated in fig. 2, our setting mirrors the theoretical setting. Given sufficient data and model capacity, the neural network S_θ matches the score function S. Consequently, the distribution of the functions f_θ^Z should mirror that of the solutions f^i. Since f^i describes a possible evolution of weather, it has to be continuous as a function of time. To ensure that the generated trajectories are also continuous in t, regularity conditions need to be imposed on S_θ to ensure that it depends smoothly on the lead time. In appendix E, we provide sufficient conditions and a proof of this property. We also show empirically in sec. 5.1 that this is satisfied in practice.

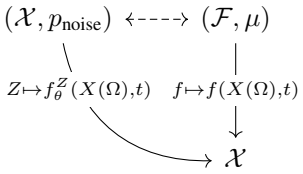

Figure 2: Diagram depicting the connection between the latent noise space and the solution space.

4.2 AUTOCORRELATED NOISE

As mentioned above, under regularity conditions, we expect the solution map $f_\theta^Z(X(\Omega), t)$ to be continuous in t. A shortcoming of this approach, however, is that it also constrains the trajectories to be conditionally deterministic, conditionally on the solution $f_\theta^Z(X(\Omega), t^\star)$ at any fixed time point $t^\star \in \mathcal{T}$. Specifically, consider the conditional distribution $p(X(t) \mid X(t^\star), X(\Omega), t)$ for $t, t^\star \in \mathcal{T}$ under the proposed model. Conditionally on $X(t^\star) = f_\theta^Z(X(\Omega), t^\star)$ we can, conceptually, invert the ODE which generated $X(t^\star)$ to recover the driving noise Z. Now, if $X(t) = f_\theta^Z(X(\Omega), t)$ is generated using the exact same noise variable, we find that $p(X(t) \mid X(t^\star), X(\Omega), t)$ is a Dirac point mass concentrated at $f_\theta^Z(X(\Omega), t)$. This violates the assumed stochasticity of the dynamical process that we are modelling.

To address this shortcoming, a simple extension of the proposed model is to replace the fixed Z with a stochastic process $Z(\mathcal{T})$ with continuous sample trajectories. The stochastic process is chosen to be stationary with marginal distribution $Z(t) \sim p_{\text{noise}}, \forall t \in \mathcal{T}$. This ensures that the generative model at any fixed t is probabilistically equivalent with the fixed noise setting. Specifically, no changes to the training algorithm are needed since the training is based solely on time marginals. Allowing temporal stochasticity in the driving noise process results in a non-deterministic relationship between the states at different lead times, while we can keep the temporal consistency by ensuring that the driving process is sufficiently autocorrelated. A simple choice is to select $Z(\mathcal{T})$ as an Ornstein-Uhlenbeck process (Gillespie, 1996), $dZ(t) = -\rho Z(t)dt + \sqrt{2\rho}dW(t)$, for some correlation parameter $\rho > 0$ and $W(t)$ being Brownian motion. This process can be easily simulated from to generate a noise sequence $\{Z(t_i)\}_{i=1}^N$ for lead-times $\{t_i\}_{i=1}^N$, as done in Algorithm 2. The initial noise sample $Z(0) \sim \mathcal{N}(0, \mathbf{I})$ is perturbed according to the Ornstein-Uhlenbeck process by sampling new noise $\nu \sim \mathcal{N}(0, \mathbf{I})$ in each time-step. The update step in line 5 comes from the choice of stochastic process and is designed to ensure a stationary marginal distribution $Z(t) \sim \mathcal{N}(0, \mathbf{I})$. This is what we use in Algorithm 2. Alternatively, we can use a Gaussian process with stronger smoothness properties, e.g., with a squared exponential kernel. This could prove useful for formally proving time continuity of the resulting trajectories, but we leave such an analysis for future work.

Algorithm 2 The Extended Continuous Ensemble Forecasting algorithm

input: Initial conditions $x(\Omega)$, times $\{t_i\}_{i=1}^N$, ensemble size n_{ens}, network S_θ, correlation parameter ρ
sample $\{z_1^j\}_{j=1}^{n_{\text{ens}}} \sim \mathcal{N}(0, \mathbf{I})$
for $i = 2$ **to** N **do**
  sample $\{\nu_i^j\}_{j=1}^{n_{\text{ens}}} \sim \mathcal{N}(0, \mathbf{I})$
  $z_i^j \leftarrow \exp(-\rho(t_i - t_{i-1}))z_{i-1}^j + \sqrt{1 - \exp(-2\rho(t_i - t_{i-1}))}\nu_i^j$
for all $i \in \{1, \dots, N\}, j \in \{1, \dots, n_{\text{ens}}\}$ **do** ▷ Can be done fully in parallel for all j and i
  $x_i^j \leftarrow$ PROBABILITY-FLOW-SOLVER$(z_i^j, t_i; x(\Omega), S_\theta)$
return $\{x_i^j\}_{i=1:N}^{j=1:n_{\text{ens}}}$

4.3 AUTOREGRESSIVE ROLL-OUTS WITH CONTINUOUS INTERPOLATION

Continuous forecasting is effective for forecasting hours to days but can struggle to forecast longer lead times where the correlation is weaker. Autoregressive forecasting excels at long lead times when used with longer (24h) timesteps (Bi et al., 2023), but comes at the loss of temporal resolution. In our framework, it becomes possible to sample both autoregressive and continuous forecasts with the same model. We propose to leverage this by iterating on a longer timestep and forecasting the intermediate timesteps using Continuous Ensemble Forecasting, as outlined in Alg. 3. We refer to this combined method as Autoregressive Rollouts with Continuous Interpolation (ARCI). Our method limits the error accumulation without sacrificing temporal resolution. This allows for producing forecasts at an arbitrary fine temporal resolution, while retaining the accuracy of the best autoregressive methods throughout the whole forecast. By limiting the number of autoregressive steps, more of the forecast also becomes parallelizable, allowing for rapidly generating forecasts on large compute clusters. Furthermore, we can straightforwardly use different time resolutions during different parts of the forecast trajectories, by for instance forecasting with 1h steps for the first few days and then switching to longer time steps for long lead times. Note that ARCI can be used with either fixed (with Alg. 1) or stochastic (with Alg. 2) driving noise. However, we emphasize that these two algorithms are probabilistically equivalent for all time marginals, and only differ in the autocorrelation of forecast trajectories.

Algorithm 3 ARCI (Autoregressive roll-outs with continuous interpolation)

input: Initial conditions $x_{-L:0}$, interpolation times $\{t_i\}_{i=1}^N$, ensemble size n_{ens}, autoregressive steps M, network S_θ
for $m = 0$ **to** $M - 1$ **do**
  $\{x_{mN+i}^j\}_{i=1:N}^{j=1:n_{\text{ens}}} \leftarrow$ **Alg. 1**$(x_{mN-L:mN}, \{t_i\}_{i=1}^N, n_{\text{ens}}, S_\theta)$ ▷ Also possible to use **Alg. 2**
return $\{x_i^j\}_{i=1:MN}^{j=1:n_{\text{ens}}}$

5 EXPERIMENTS

Data. We evaluate our method on global weather forecasting up to 10 days at 1, 6, and 24 hour timesteps. We use the downsampled ERA5 reanalysis dataset (Hersbach et al., 2020) at 5.625°resolution and 1-hour increments provided by WeatherBench (Rasp et al., 2020). The models are trained to forecast 5 variables from the ERA5 dataset: geopotential at 500hPa (`z500`), temperature at 850hPa (`t850`), ground temperature (`t2m`) and the ground wind components (`u10, v10`). The atmospheric fields `z500` and `t850` offer a comprehensive view of atmospheric dynamics and thermodynamics, while the surface fields `t2m` and `u10, v10` are important for day-to-day activities. We also evaluate the forecast of ground wind speed `ws10`, computed from the model outputs as $\texttt{ws10} = \sqrt{\texttt{u10}^2 + \texttt{v10}^2}$. This is useful for evaluating how well the methods model cross-variable dependencies. All variables are standardized by subtracting their mean and dividing by their standard deviation. Together with the previous states we also feed the models with static fields. These include the land-sea mask and orography, both rescaled to $[0, 1]$. All models are trained on the period

1979–2015, validated on 2016–2017 and tested on 2018. We consider every hour of each year as forecast initialization times, except for the first 24 h and last 10 days in each subset. This guarantees that all times forecasted or conditioned on lie within the specific years.

Metrics. We evaluate the skill of the forecasting models by computing the Root Mean Squared Error (**RMSE**) of the ensemble mean. As a probabilistic metric we also consider Continuous Ranked Probability Score (**CRPS**) (Gneiting & Raftery, 2007), which measures how well the the predicted marginal distributions capture the ground truth. We also evaluate the Spread/Skill-Ratio (**SSR**), which is a common measure of calibration for ensemble forecasts. For a model with well calibrated uncertainty estimates the SSR should be close to 1 (Fortin et al., 2014). Detailed definitions of all metrics are given in appendix A.

Models. We propose to use the ARCI model described in algorithm 3 referred to as **ARCI-24/6h**. We train it to forecast $t \in \{6, 12, 18, 24\}$ (hours) and roll it out autoregressively with 24 h steps, hence the name. Training is done on a 40GB NVIDIA A100 GPU and takes roughly 2 days. We emphasize again that using fixed, correlated or uncorrelated noise results in probabilistically equivalent forecasts for all time marginals, and only differ in the autocorrelation of forecast trajectories. Hence the choice of algorithm inside ARCI does not matter, and for all metrics below that are computed for specific lead times we only report results for one version of the algorithm. We return to the difference between Alg. 1 and Alg. 2 when studying the temporal difference below.

To evaluate the effectiveness of our approach, we compare it to other MLWP baselines. **Deterministic** is a deterministic model trained using MSE-loss on a single 6 hour time step, and unrolled up to 10 days. **AR-6/24h** is a diffusion model trained only on forecasting a single fixed δ ahead, and then autoregressively unrolled up to 10 days. This is the exact forecasting setup of (Price et al., 2024) and the AR- models can thus be seen as a reimplementation of GenCast with a U-Net architecture. **CI-6h** is a diffusion model performing continuous forecasting conditioned on a specific lead time. It is trained on uniformly sampled lead times from $\{k\delta\}_{k=1}^{40}$, with $\delta = 6$ h. This is the method proposed in alg. 1. Sampling a 10-day forecast with 6h resolution for a single member from AR-6h takes 32 seconds, but by parallelizing the 6h timesteps in ARCI-24/6h this reduces to 8 seconds.

To compare against another family of ensemble forecasting models from the literature we retrain the **Graph-EFM** model (Oskarsson et al., 2024) on our exact data setup. Graph-EFM is a graph-based latent-variable model that produces forecasts by 6 h iterative rollout steps. For all models, unless otherwise specified, we condition on the two previous timesteps $\Omega = \{0, -\delta\}$ and sample 50 ensemble members at each initialization time. All models except Graph-EFM use the same architecture based on the U-net in Karras et al. (2022) as presented in appendix B.

5.1 Results

Quantitative results. Table 1 and figure 3 show metrics for a selection of lead times and variables. Scores for the remaining variables are listed in appendix C. All probabilistic models show a clear improvement over the deterministic model. CI-6h performs well on short-term forecasting but struggles at longer horizons. This is likely due to the challenge of learning any useful relationships between initial states and later lead times, which are weakly correlated. ARCI-24+6h outperforms all models at 6h resolution, including the external baseline Graph-EFM and the GenCast setup AR-6h, and matches the best overall model AR-24h in almost all scores. All diffusion-based models have SSR < 1, indicating some systematic underdispersion.

Qualitative results. Figure 4 shows an example forecast from ARCI-24/6h for temperature at 850 hPa (`t850`) at 10 days lead time.[2] The forecasts are rich in detail, resembling the true state more than the ensemble mean. Examples of other variables are given in appendix D.

Temporal Difference. Autoregressively sampled forecast trajectories are necessarily continuous. Since there is no standard way of measuring the continuity of a forecast, we propose using the mean temporal difference $\Delta X = |X(t) - X(t-1)|$ as a measure of forecast continuity. Figure 5 shows ΔX for a CI-1h continuous model trained up to 24 hours with 1-hour timesteps.

[2] Animations of forecasts from ARCI-24/6h for all variables are provided in the supplementary material.

Table 1: Selection of results for 5 and 10 day forecasting using models with 6h resolution for geopotential at 500 hPa (z500) and temperature at 850 hPa (t850). For RMSE and CRPS, lower values are better, and SSR should be close to 1. The best values are marked with **bold** and second best underlined. The AR-24h model is included for reference, but is not considered for best model since it operates at a coarser 24h resolution.

		Lead time 5 days			Lead time 10 days		
Variable	Model	RMSE	CRPS	SSR	RMSE	CRPS	SSR
z500	Deterministic	766.7	483.9	-	1042	661.5	-
	Graph-EFM	699.1	317.5	**1.13**	817.1	373.6	1.1
	AR-6h	602.3	287.8	0.75	811.8	391.9	0.88
	CI-6h	707.8	321.2	0.59	885.7	406.6	0.6
	ARCI-24/6h	**560.9**	**256.7**	0.86	**765.6**	**355.2**	**0.93**
	AR-24h	544.2	242.7	0.84	750.6	335.2	0.94
t850	Deterministic	3.48	2.36	-	4.54	3.17	-
	Graph-EFM	3.12	1.56	**1.11**	3.51	1.77	1.12
	AR-6h	2.72	1.34	0.82	3.39	1.69	0.92
	CI-6h	3.06	1.5	0.74	3.68	1.85	0.71
	ARCI-24/6h	**2.6**	**1.27**	0.9	**3.29**	**1.63**	**0.95**
	AR-24h	2.55	1.24	0.89	3.25	1.6	0.96

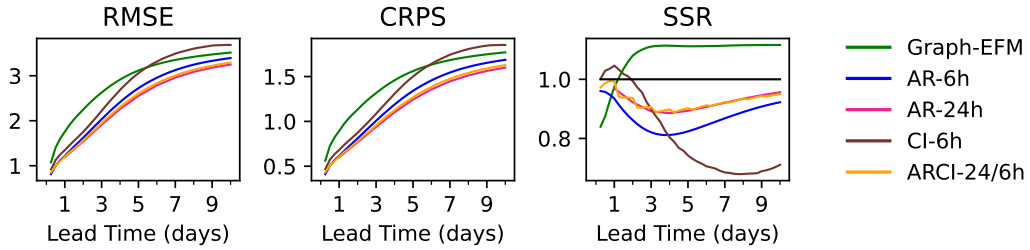

Figure 3: RMSE, CRPS and SSR for temperature at 850 hPa (t850) with a selection of models at 6h resolution.

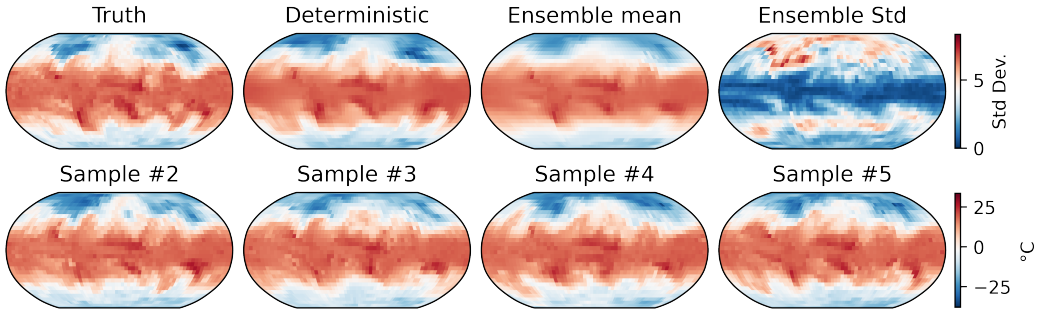

Figure 4: Example forecasts for temperature at 850hPa (t850) at lead time 10 days. The forecasts are generated using ARCI-24/6h except for Deterministic which is sampled using the deterministic model. The bottom row shows 4 ensemble members, randomly chosen out of the 50.

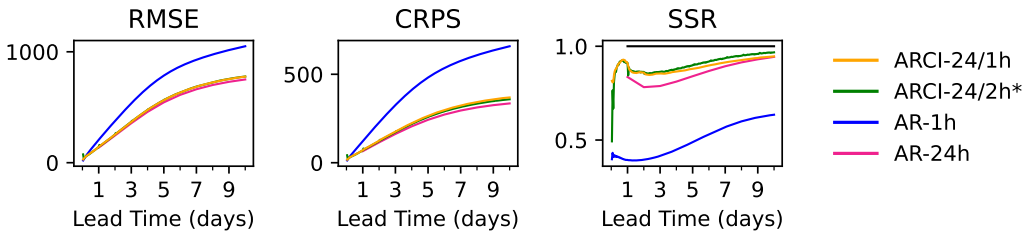

Figure 6: RMSE, CRPS, and SSR for geopotential at 500 hPa (`z500`) with a selection of models at 1h resolution.

Compared to using different noise at each step ($\rho \to \infty$), the temporal difference of our model ($\rho = 0$) stays close to the temporal difference of the data. This supports our claim that continuous ensemble forecasting produces continuous trajectories.

Figure 5 also shows the temporal difference of the continuous model above. When the noise is fixed ($\rho = 0$) the temporal difference decreases with lead time, corresponding to predictions with smaller temporal variations. Letting the noise vary with noise factor ($\rho = \ln 10$) as in alg. 2 stops this from happening. The bias between ΔX of the data and our model is likely due to the model producing slightly blurrier forecasts, making the differences smaller.

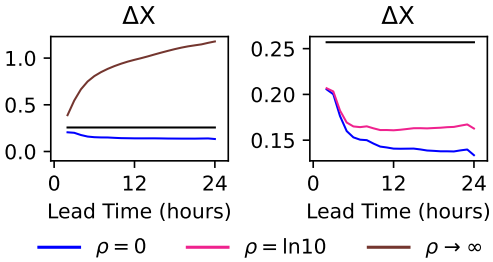

Figure 5: Temporal difference for temperature on 850 hPa (`t850`) for different values of ρ in algorithm 2. Choosing $\rho = 0$ fixes the noise, $\rho = \ln 10$ allows it to vary and $\rho \to \infty$ gives completely uncorrelated noise. The black line refers to the temporal difference of the data.

Continuous Time Forecasting. Our ARCI method allows for producing forecasts at arbitrary fine temporal resolution while retaining the accuracy of methods taking longer autoregressive steps. We here demonstrate this by producing hourly forecasts. Figure 6 shows the scores of a selection of models for `z500` for 10-day forecasts at 1h resolution. The AR-1h model has the same setup as AR-6h but on 1h resolution. ARCI-24/1h and ARCI-24/2h* are both trained with $\Omega = \{0, -24\}$ to have access to the same information at each timestep. The autoregressive AR-1h model performs much worse than on 6 or 24 hours. The continuous model, however, does not lose performance by increasing the temporal resolution, making the 1h timestep forecasts as skillful as the 24 hour ones. An alternative to directly producing forecasts at a fine temporal resolution would be to linearly interpolate the forecasts sampled using an autoregressive model. In fig. 9 in appendix C we show that linearly interpolated forecasts behave much worse than the ARCI model on both 1 and 6-hour resolution.

The ARCI-24/2h* model is trained only on lead times 2h apart (lead times in $\{2k\}_{k=1}^{12}$), but used to forecast each 1h timestep. This showcases the ability of the method to generalize to lead times not considered during training. It performs similarly to the ARCI-24/1h model trained on all timesteps, indicating that it can generalize beyond its training setup to even finer resolutions. For highly time-dependent fields such as `t2m`, the model performs worse at the first forecast in each iteration (1h, 25h,...), as seen in figure 11 in appendix C. For other lead-times t not considered during training, the model has trained on forecasting $t - 1$ and $t + 1$, thus only having to interpolate to t. However, since we do not train on forecasting 0h, the model instead has to extrapolate to $t = 1$h what was learned for 2h forecasts. This issue could possibly be fixed by letting the network also train on 0h forecasts.

6 Conclusion

We present Continuous Ensemble Forecasting, a novel framework for probabilistic MLWP that increases the efficiency, accuracy, and flexibility of weather forecasts at high temporal resolution.

When combined with autoregressive prediction, our ARCI method produces 10-day forecasts with a 1-hour resolution that matches the accuracy of a purely autoregressive model with 24-hour steps. With this work, we hope to show that the possibilities with generative modeling for spatio-temporal predictions are still largely unexplored and a fruitful area of research.

Limitations. While our proposed framework achieves good results on 5.625°Weatherbench (Rasp et al., 2020) data, we have yet to show that the method scales to problems with higher spatial resolution. Additionally, as the lead time increases, the correlation between initial and future states becomes weaker, limiting the application of continuous forecasting. While our method parallelizes more of the sampling than previous autoregressive models, solving the probability flow ODE in eq. 2 still requires many sequential forward passes through the network. Sampling is thus still slower than for latent variable models, but the predicted distribution more accurate. Finally, due to computational limitations, we were not able to retrain our models several times and as such do not report error bars in our results.

Future work. One interesting direction for future work is a further investigation of autocorrelated noise, in particular, how the choice of stochastic process can aid in producing continuous trajectories with a stationary temporal difference. This includes correlating the noise in the autoregressive steps with the continuous steps, which could help ease the transition between them. Another idea is to take the direction of DYffusion (Rühling Cachay et al., 2023) and directly adjust the diffusion objective to better suit temporal data. While we have demonstrated continuous ensemble forecasting for weather, the idea is generally applicable and it would also be of interest to apply it to other spatio-temporal forecasting problems.

REFERENCES

Michael S. Albergo and Eric Vanden-Eijnden. Building Normalizing Flows with Stochastic Interpolants, 2023. arXiv:2209.15571.

Seth Bassetti, Brian Hutchinson, Claudia Tebaldi, and Ben Kravitz. DiffESM: Conditional emulation of earth system models with diffusion models. In *ICLR 2023 Workshop on Tackling Climate Change with Machine Learning*, 2023.

Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3D neural networks. *Nature*, 619(7970):533–538, 2023.

Salva Rühling Cachay, Brian Henn, Oliver Watt-Meyer, Christopher S. Bretherton, and Rose Yu. Probabilistic Emulation of a Global Climate Model with Spherical DYffusion, 2024. arXiv:2406.14798.

Lei Chen, Fei Du, Yuan Hu, Zhibin Wang, and Fan Wang. SwinRDM: integrate SwinRNN with diffusion model towards high-resolution and high-quality weather forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 322–330, 2023a.

Lei Chen, Xiaohui Zhong, Feng Zhang, Yuan Cheng, Yinghui Xu, Yuan Qi, and Hao Li. FuXi: a cascade machine learning forecasting system for 15-day global weather forecast. *npj Climate and Atmospheric Science*, 6(1):1–11, 2023b.

Guillaume Couairon, Christian Lessig, Anastase Charantonis, and Claire Monteleoni. ArchesWeather: An efficient AI weather forecasting model at 1.5° resolution, 2024. arXiv:2405.14527.

Vincent Fortin, Mabrouk Abaza, Francois Anctil, and Raphael Turcotte. Why should ensemble spread match the RMSE of the ensemble mean? *Journal of Hydrometeorology*, 15(4):1708–1713, 2014.

Daniel T. Gillespie. Exact numerical simulation of the ornstein-uhlenbeck process and its integral. *Phys. Rev. E*, 1996.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.

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

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna De Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia De Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Noël Thépaut. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*, 33, 2020.

Yuan Hu, Lei Chen, Zhibin Wang, and Hao Li. SwinVRNN: A data-driven ensemble forecasting model via learned distribution perturbation. *Journal of Advances in Modeling Earth Systems*, 15 (2), 2023.

Zhanxiang Hua, Yutong He, Chengqian Ma, and Alexandra Anderson-Frey. Weather prediction with diffusion guided by realistic forecast processes, 2024. arXiv:2402.06666.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35, 2022.

Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, Milan Klöwer, James Lottes, Stephan Rasp, Peter Düben, Sam Hatfield, Peter Battaglia, Alvaro Sanchez-Gonzalez, Matthew Willson, Michael P. Brenner, and Stephan Hoyer. Neural general circulation models for weather and climate. *Nature*, 632(8027):1060–1066, August 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07744-y. URL https://www.nature.com/articles/s41586-024-07744-y. Publisher: Nature Publishing Group.

Georg Kohl, Liwei Chen, and Nils Thuerey. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. In *ICML 2024 AI for Science Workshop*, 2024.

Thorsten Kurth, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, and Anima Anandkumar. FourCastNet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, 2023.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

Simon Lang, Matthew Chantry, Rilwan Adewoyin, Mihai Alexe, Zied Ben Bouallègue, Mariana Clare, Jesper Dramsch, Christian Lessig, Linus Magnusson, Michael Maier-Gerber, Gert Mertes, Gabriel Moldovan, Ana Prieto Nemesio, Cathal O'Brien, Florian Pinault, Meghan Plumridge, Baudouin Raoult, Mario Santa Cruz, and Steffen Tietsche. Enter the ensembles, 2024. URL https://www.ecmwf.int/en/about/media-centre/aifs-blog/2024/enter-ensembles.

Martin Leutbecher and Tim N Palmer. Ensemble forecasting. *Journal of Computational Physics*, 227(7):3515–3539, 2008.

Lizao Li, Robert Carver, Ignacio Lopez-Gomez, Fei Sha, and John Anderson. Generative emulation of weather forecast ensembles with diffusion models. *Science Advances*, 10(13), 2024.

Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. PDE-Refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36, 2023.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in Adam, 2017a. arXiv:1711.05101.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017b.

Morteza Mardani, Noah Brenowitz, Yair Cohen, Jaideep Pathak, Chieh-Yu Chen, Cheng-Chin Liu, Arash Vahdat, Mohammad Amin Nabian, Tao Ge, Akshay Subramaniam, Karthik Kashinath, Jan Kautz, and Mike Pritchard. Residual corrective diffusion modeling for km-scale atmospheric downscaling, 2024. arXiv:2309.15214.

Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K. Gupta, and Aditya Grover. ClimaX: A foundation model for weather and climate. In *Proceedings of the 40th International Conference on Machine Learning*, 2023a.

Tung Nguyen, Rohan Shah, Hritik Bansal, Troy Arcomano, Sandeep Madireddy, Romit Maulik, Veerabhadra Kotamarthi, Ian Foster, and Aditya Grover. Scaling transformer neural networks for skillful and reliable medium-range weather forecasting, 2023b. arXiv:2312.03876.

Lecture Notes. Continuous dependence of solutions to differential equations on parameters, 2014. URL `https://users.math.msu.edu/users/hhu/848/lec_5.pdf`.

Joel Oskarsson, Tomas Landelius, Marc Peter Deisenroth, and Fredrik Lindsten. Probabilistic weather forecasting with hierarchical graph neural networks, 2024. arXiv:2406.04759.

Jaideep Pathak, Yair Cohen, Piyush Garg, Peter Harrington, Noah Brenowitz, Dale Durran, Morteza Mardani, Arash Vahdat, Shaoming Xu, Karthik Kashinath, and Michael Pritchard. Kilometer-scale convection allowing model emulation using generative diffusion modeling, 2024. arXiv:2408.10958.

Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. GenCast: Diffusion-based ensemble forecasting for medium-range weather, 2024. arXiv:2312.15796.

Stephan Rasp, Peter D. Dueben, Sebastian Scher, Jonathan A. Weyn, Soukayna Mouatadid, and Nils Thuerey. WeatherBench: A benchmark dataset for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11), 2020.

Salva Rühling Cachay, Bo Zhao, Hailey Joren, and Rose Yu. DYffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *Advances in Neural Information Processing Systems*, 36, 2023.

Hira Saleem, Flora Salim, and Cormac Purcell. Stc-vit: Spatio temporal continuous vision transformer for weather forecasting, 2024. arXiv:2402.17966.

Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation, 2019. arXiv:1805.10965.

Sebastian Scher and Gabriele Messori. Ensemble Methods for Neural Network-Based Weather Forecasts. *Journal of Advances in Modeling Earth Systems*, 13(2), 2021.

Jimeng Shi, Bowen Jin, Jiawei Han, and Giri Narasimhan. CoDiCast: Conditional diffusion model for weather prediction with uncertainty quantification, 2024. arXiv:2409.05975.

George F. Simmons. *Differential Equations with Applications and Historical Notes*. Chapman and Hall/CRC, 3rd edition edition, 2016. ISBN 978-1-4987-0259-1.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation, 2019. arXiv:1905.07088.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Yogesh Verma, Markus Heinonen, and Vikas Garg. ClimODE: Climate and weather forecasting with physics-informed neural odes. In *International Conference on Learning Representations*, 2024.

Jonathan A. Weyn, Dale R. Durran, Rich Caruana, and Nathaniel Cresswell-Clay. Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models. *Journal of Advances in Modeling Earth Systems*, 13(7), 2021.

Gefan Yang and Stefan Sommer. A denoising diffusion model for fluid field prediction, 2023. arXiv:2301.11661.

Yiyuan Yang, Ming Jin, Haomin Wen, Chaoli Zhang, Yuxuan Liang, Lintao Ma, Yi Wang, Chenghao Liu, Bin Yang, Zenglin Xu, Jiang Bian, Shirui Pan, and Qingsong Wen. A survey on diffusion models for time series and spatio-temporal data, 2024. arXiv:2404.18886.

Xiaohui Zhong, Lei Chen, Hao Li, Jun Liu, Xu Fan, Jie Feng, Kan Dai, Jing-Jia Luo, Jie Wu, and Bo Lu. FuXi-ENS: A machine learning model for medium-range ensemble weather forecasting, 2024. arXiv:2405.05925.

A METRICS

We consider the following evaluation metrics used to assess the probabilistic forecasts produced by the diffusion model. Metrics from meteorology and general uncertainty quantification, such as RMSE, Spread/Skill ratio (SSR), and Continuous Ranked Probability Score (CRPS) are employed to measure the effectiveness and reliability of the model outputs. All of our metrics are weighted by latitude-dependent weights. For a particular variable and lead time,

- $x^k_{i,n}$ represents the value of the k-th ensemble member at initialization time indexed by $n = 1 \dots N$ for grid cell in the latitude and longitude grid indexed by $i \in I$.
- $y_{i,n}$ denotes the corresponding ground truth.
- $\bar{x}_{i,n}$ denotes the ensemble mean, defined by $\bar{x}_{i,n} = \frac{1}{n_{\text{ens}}} \sum_{k=1}^{n_{\text{ens}}} x^k_{i,n}$.
- a_i denotes the area of the latitude-longitude grid cell, which varies by latitude and is normalized to unit mean over the grid (Rasp et al., 2020).

RMSE or skill measures the accuracy of the forecast. Following Rasp et al. (2020) we define the RMSE as the mean square root of the ensemble mean:

$$\text{RMSE} := \frac{1}{N} \sum_{n=1}^{N} \sqrt{\frac{1}{|I|} \sum_{i \in I} a_i (y_{i,n} - \bar{x}_{i,n})^2}. \tag{3}$$

In the case of deterministic predictions, the ensemble mean is taken as the deterministic prediction.

Spread represents the variability within the ensemble and is calculated as the root mean square of the ensemble variance:

$$\text{Spread} := \frac{1}{N} \sum_{n=1}^{N} \sqrt{\frac{1}{|I|} \sum_{i \in I} \frac{1}{n_{\text{ens}} - 1} \sum_{k=1}^{n_{\text{ens}}} a_i (x^k_{i,n} - \bar{x}_{i,n})^2}. \tag{4}$$

Ideally, the forecast achieves a balance where skill and spread are proportional, leading to an optimal spread/skill ratio (SSR) close to 1, indicating effective uncertainty estimation:

$$\text{SSR} := \sqrt{\frac{n_{\text{ens}} + 1}{n_{\text{ens}}}} \frac{\text{Spread}}{\text{RMSE}}. \tag{5}$$

Continuous Ranked Probability Score (CRPS) (Gneiting & Raftery, 2007) measures the accuracy of probabilistic forecasts by comparing the cumulative distribution functions (CDFs) of the predicted and observed values. It integrates the squared differences between these CDFs, providing a single score that penalizes differences in location, spread, and shape of the distributions. An estimator of the CRPS is given by:

$$\text{CRPS} := \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|I|} \sum_{i \in I} a_i \left(\frac{1}{n_{\text{ens}}} \sum_{k=1}^{n_{\text{ens}}} |x^k_{i,n} - y_{i,n}| - \frac{1}{2n^2_{\text{ens}}} \sum_{k=1}^{n_{\text{ens}}} \sum_{k'=1}^{n_{\text{ens}}} |x^k_{i,n} - x^{k'}_{i,n}| \right).$$

Temporal Difference measures the mean absolute difference between states at consecutive times. It's used to measure the continuity of a forecast. For forecasts $x^k_{i,n}, \hat{x}^k_{i,n}$ at consecutive lead times, it is given by:

$$\Delta X := \frac{1}{N} \sum_{n=1}^{N} \frac{1}{n_{\text{ens}}} \sum_{k=1}^{n_{\text{ens}}} \frac{1}{|I|} \sum_{i \in I} a_i |x^k_{i,n} - \hat{x}^k_{i,n}|. \tag{6}$$

B MODEL

Preconditioning The sampling process is based on the denoising neural network D_θ that takes a noisy residual and tries to denoise it. To help in this, it is also given the noise level σ, the previous

Table 2: Scaling functions.

Skip scaling	$c_{\text{skip}}(\sigma)$	$\sigma_{\text{data}}^2/(\sigma^2+\sigma_{\text{data}}^2)$
Output scaling	$c_{\text{out}}(\sigma)$	$\sigma\cdot\sigma_{\text{data}}/\sqrt{\sigma^2+\sigma_{\text{data}}^2}$
Input scaling	$c_{\text{in}}(\sigma)$	$1/\sqrt{\sigma^2+\sigma_{\text{data}}^2}$
Noise scaling	$c_{\text{noise}}(\sigma)$	$\frac{1}{4}\ln(\sigma)$

state $X(\Omega)$ and the lead time t. To simplify learning, D_θ is parameterized by a different network F_θ defined by

$$D_\theta(X_t^\sigma;\sigma,X(\Omega),t)=c_{\text{skip}}(\sigma)\cdot X_t^\sigma+c_{\text{out}}(\sigma)\cdot F_\theta\left(c_{\text{in}}(\sigma)\cdot X_t^\sigma;c_{\text{noise}}(\sigma),X(\Omega),t\right),$$

where X_t^σ denotes a noisy version of the target time $X(t)$ at noise level σ, and c_{skip}, c_{out}, c_{in} and c_{noise} are scaling functions taken from (Karras et al., 2022) defined in Tab. 2. These scaling functions c_{skip}, c_{out}, c_{in} and c_{noise}, are specifically chosen to handle the influence of the noise level within the network, allowing D_θ to adapt dynamically to different noise intensities without the need for adjusting the scale of σ externally. Consequently, for consistency with the normalization of the data where σ_{data} is set to 1, the lead time t is also scaled to fit within the range $[0,1]$. This normalization ensures that the network inputs are uniformly scaled, enhancing the efficiency and effectiveness of the denoising process.

Conditioning To condition on the initial conditions $X(\Omega)$ and static fields, these are concatenated along the channel dimension with the input to the denoiser, increasing the dimension of the input. To condition on the noise level σ and lead time t, we use Fourier embedding as specified in (Karras et al., 2022). Fourier embedding captures periodic patterns in noise and time, enhancing the model's ability to handle complex time-series dependencies effectively. They work by transforming the time/noise into a vector of sine/cosine features at 32 frequencies with period 16. These vectors are added and then passed through two fully connected layers with SiLU activation to obtain a 128-dimensional encoding.

Architecture The backbone of the diffusion model is a U-Net architecture. Our model is based on the one used in (Karras et al., 2022), reconfigured for our purposes with 32 filters as the base multiplier. It is built up by blocks configured as in fig. 7. The blocks consist of two convolutional layers and are constructed as in fig. 8. If the block is a down-/up-sample or if the number of input filters is different from the number of output filters, there is an additional skip layer from the input to the output. The time/noise embedding is fed directly into each block and not passed through the network. Unlike the network in Karras et al. (2022), our convolutions uses zero padding at the poles and periodic padding at the left/right edges. This periodic padding ensures periodicity over longitudes. The model has 3.5M parameters.

Sampling To generate forecasts using our diffusion model, we solve the probability flow ODE as defined in (Karras et al., 2022)

$$\mathrm{d}\boldsymbol{z}=-\dot{\sigma}(s)\sigma(s)\nabla_{\boldsymbol{z}}\log p_s\left(\boldsymbol{z}\right)\mathrm{d}s. \tag{7}$$

We employ the second-order Heun's method, a deterministic ODE solver, as outlined in Algorithm 4. For the noise parameters, we define the noise level function as $\sigma(s)=s$. Additionally, we set a noise level schedule to lower the noise during sampling from $\sigma_{\max}$ to $\sigma_{\min}$ over N steps:

$$s_i=\left(\sigma_{\max}^{\frac{1}{\rho}}+\frac{i}{N-1}\left(\sigma_{\min}^{\frac{1}{\rho}}-\sigma_{\max}^{\frac{1}{\rho}}\right)\right)^\rho,\quad i\in\{0,\ldots,N-1\}.$$

The relevant parameters for training and sampling are given in tab. 3.

Training The dataset is partitioned into three subsets: training, validation, and testing. The training subset is used for model training, the validation subset for evaluating generalization, and the testing subset to determine final accuracy. The diffusion model is trained using the following training objective

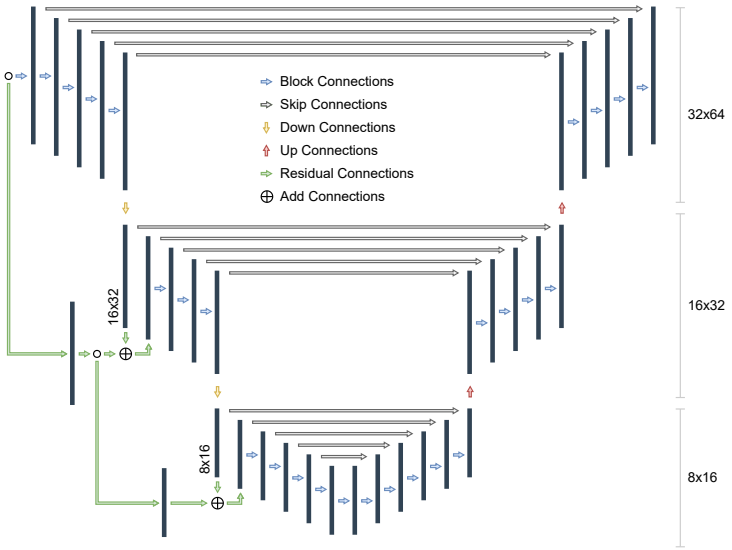

Figure 7: Overview of the U-Net Architecture, detailing layer configurations and the flow of information through convolutional blocks and skip connections.

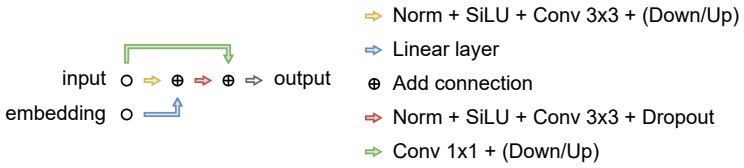

Figure 8: Construction of a Diffusion Model Block, showing the sequence of operations and the integration of embeddings with add connections.

Algorithm 4 Deterministic sampling using Heun's 2nd order method.

procedure HEUNSAMPLER($D_\theta(\boldsymbol{z}; \sigma, X(\Omega), t)$, $s_{i\in\{0,\ldots,N\}}$, Z)
$\boldsymbol{z}_0 \leftarrow \sigma^2(s_0) \cdot Z$ ▷ Generate initial sample at s_0
for $i = 0$ **to** $N-1$ **do** ▷ Solve ODE over N time steps
  $\boldsymbol{d}_i \leftarrow \frac{\dot{\sigma}(s_i)}{\sigma(s_i)}(\boldsymbol{z}_i - D_\theta(\boldsymbol{z}_i; \sigma(s_i), X(\Omega), t))$ ▷ Evaluate $d\boldsymbol{z}/ds$ at s_i
  $\boldsymbol{z}_{i+1} \leftarrow \boldsymbol{z}_i + (s_{i+1} - s_i)\boldsymbol{d}_i$ ▷ Take Euler step from s_i to s_{i+1}
  if $s_{i+1} \neq 0$ **then** ▷ Apply 2nd order correction unless σ goes to zero
    $\boldsymbol{d}'_i \leftarrow \frac{\dot{\sigma}(s_{i+1})}{\sigma(s_{i+1})}(\boldsymbol{z}_{i+1} - D_\theta(\boldsymbol{z}_{i+1}; \sigma(s_{i+1}), X(\Omega), t))$ ▷ Evaluate $d\boldsymbol{z}/ds$ at s_{i+1}
    $\boldsymbol{z}_{i+1} \leftarrow \boldsymbol{z}_i + \frac{1}{2}(s_{i+1} - s_i)\,(\boldsymbol{d}_i + \boldsymbol{d}'_i)$ ▷ Explicit trapezoidal rule at s_{i+1}
return $\boldsymbol{z}_N$ ▷ Return noise-free sample at s_N

Table 3: Parameters used for sampling and training.

Name	Notation	Value, sampling	Value, training
Maximum noise level	σ_{max}	80	88
Minimum noise level	σ_{min}	0.03	0.02
Shape of noise distribution	ρ	7	7
Number of noise levels	N	20	

Table 4: Optimizer Hyperparameters.

Optimizer hyperparameters	
Optimiser	AdamW (Loshchilov & Hutter, 2017a)
Initialization	Xavier Uniform (Glorot & Bengio, 2010)
LR decay schedule	Cosine (Loshchilov & Hutter, 2017b)
Peak LR	5e-4
Weight decay	0.1
Warmup steps	1e3
Epochs	300
Batch size	256
Dropout probability	0.1

Table 5: Training schedule for Graph-EFM, using the notation from Oskarsson et al. (2024).

Epochs	Learning Rate	Unrolling steps	λ_{KL}	λ_{CRPS}
20	10^{-3}	1	0	0
75	10^{-3}	1	0.1	0
20	10^{-4}	4	0.1	0
8	10^{-4}	8	0.1	10^5

$$\mathbb{E}_{t\sim p^t}\mathbb{E}_{\sigma\sim p_\sigma}\mathbb{E}_{(X(\Omega),X(t))\sim p_{\text{data}}}\mathbb{E}_{\epsilon|\sigma\sim\mathcal{N}(0,\sigma^2\mathbf{I})}\left[\frac{1}{\sigma^2}\sum_i\sum_j\frac{a_i}{s_j(t)}\frac{1}{|I||J|}\left(\hat{X}(t)_{i,j}-X(t)_{i,j}\right)^2\right].$$

with $\hat{X}(t) = D_\theta(X(t)+\epsilon;\sigma,X(\Omega),t)$ and J being the set of variables. Here, p^t represents a uniform distribution over the lead times. We have also included a scaling term $s_j(t)^{-1}$ which scales the loss by the precomputed standard deviation $s_j(t)$ based on lead time t for each variable $j \in J$. This normalization process is designed to weigh short and longer times equally. The noise level distribution p_σ is chosen to be consistent with the sampling noise level described above. Specifically, its inverse CDF is:

$$F^{-1}(u) = \left(\sigma_{\max}^{\frac{1}{\rho}} + u\left(\sigma_{\min}^{\frac{1}{\rho}} - \sigma_{\max}^{\frac{1}{\rho}}\right)\right)^\rho,$$

and we sample from it by drawing $u \sim U[0,1]$. The training process is executed in Pytorch, with setup and parameters detailed in Tab. 4.

Graph-EFM Baseline For the Graph-EFM baseline we use the same data setup as for the other models. Since we are working on a coarser resolution than Oskarsson et al. (2024) some adaptation was necessary to the exact graph structure used in the model. We construct the graph by splitting a global icosahedron 2 times, resulting in 3 hierarchical graph levels. The training follows the same schedule as in Oskarsson et al. (2024), with pre-training on single step 6 h prediction and fine-tuning on rollouts. Details of the training schedule are given in tab. 5.

C Additional Results

Interpolation An alternative to directly producing forecasts at a fine temporal resolution would be to simply interpolate the forecasts sampled using an autoregressive model. We argue that our ARCI method produces more accurate and realistic predictions than simple linear interpolation. Figure 9 shows the RMSE and Spread/Skill for a linear interpolation of AR-24h, compared to ARCI-24/6h. The interpolated forecasts behave much worse than the ARCI model on the same resolution, which strengthens the role of our model as an advanced interpolator.

Table 6 and figure 10 show the results on 10 day forecasting for all variables. Figure 11 shows the results on forecasting with 1h resolution. Figure 12 shows the temporal difference for different values of ρ.

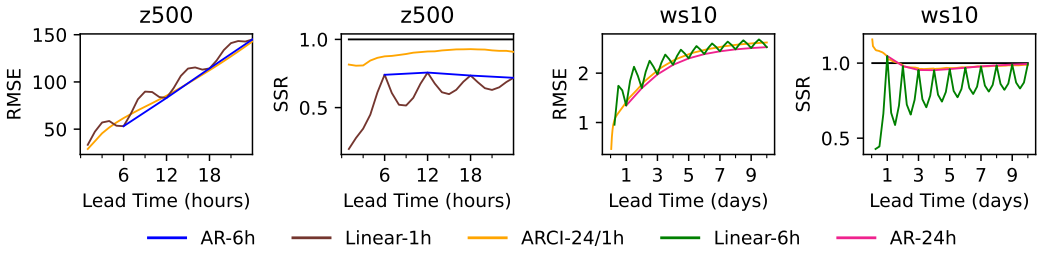

Figure 9: RMSE and SSR for linear interpolations of geopotential at 500 hPa (`z500`) and ground wind speed (`ws10`). Linear-1h/Linear-6h are linear interpolations of AR-6h/AR-24h.

Table 6: Results on 5 and 10 day forecasting for all variables. For RMSE and CRPS, lower values are better, and SSR should be close to 1.

		Lead time 5 days			Lead time 10 days		
Variable	Model	RMSE	CRPS	SSR	RMSE	CRPS	SSR
`z500`	Deterministic	766.7	483.9	-	1042	661.5	-
	Graph-EFM	699.1	317.5	1.13	817.1	373.6	1.1
	AR-6h	602.3	287.8	0.75	811.8	391.9	0.88
	AR-24h	544.2	242.7	0.84	750.6	335.2	0.94
	CI-6h	707.8	321.2	0.59	885.7	406.6	0.6
	ARCI-24/6h	560.9	256.7	0.86	765.6	355.2	0.93
`t850`	Deterministic	3.48	2.36	-	4.54	3.17	-
	Graph-EFM	3.12	1.56	1.11	3.51	1.77	1.12
	AR-6h	2.72	1.34	0.82	3.39	1.69	0.92
	AR-24h	2.55	1.24	0.89	3.25	1.6	0.96
	CI-6h	3.06	1.5	0.74	3.68	1.85	0.71
	ARCI-24/6h	2.6	1.27	0.9	3.29	1.63	0.95
`t2m`	Deterministic	2.71	1.72	-	3.56	2.28	-
	Graph-EFM	2.51	1.1	1.09	2.88	1.32	1.14
	AR-6h	2.13	0.96	0.83	2.62	1.18	0.89
	AR-24h	1.98	0.87	0.9	2.48	1.09	0.95
	CI-6h	2.29	1.0	0.79	2.75	1.21	0.75
	ARCI-24/6h	2.02	0.9	0.9	2.51	1.11	0.94
`u10`	Deterministic	4.37	2.93	-	5.14	3.53	-
	Graph-EFM	3.81	1.93	0.97	4.08	2.07	0.97
	AR-6h	3.47	1.71	0.86	3.95	1.97	0.94
	AR-24h	3.32	1.62	0.92	3.85	1.9	0.97
	CI-6h	3.87	1.91	0.77	4.27	2.14	0.78
	ARCI-24/6h	3.35	1.64	0.93	3.87	1.92	0.97
`v10`	Deterministic	4.48	3.0	-	5.23	3.58	-
	Graph-EFM	3.88	1.96	0.94	4.11	2.08	0.93
	AR-6h	3.55	1.76	0.86	4.02	2.0	0.96
	AR-24h	3.42	1.68	0.93	3.96	1.96	0.99
	CI-6h	4.0	1.99	0.77	4.39	2.2	0.8
	ARCI-24/6h	3.45	1.69	0.94	3.97	1.97	0.99
`ws10`	Deterministic	3.09	2.2	-	3.44	2.49	-
	Graph-EFM	2.56	1.35	1.01	2.65	1.4	1.01
	AR-6h	2.38	1.23	0.9	2.57	1.34	0.96
	AR-24h	2.3	1.18	0.96	2.53	1.31	0.99
	CI-6h	2.54	1.32	0.88	2.68	1.4	0.89
	ARCI-24/6h	2.31	1.19	0.96	2.54	1.32	0.99

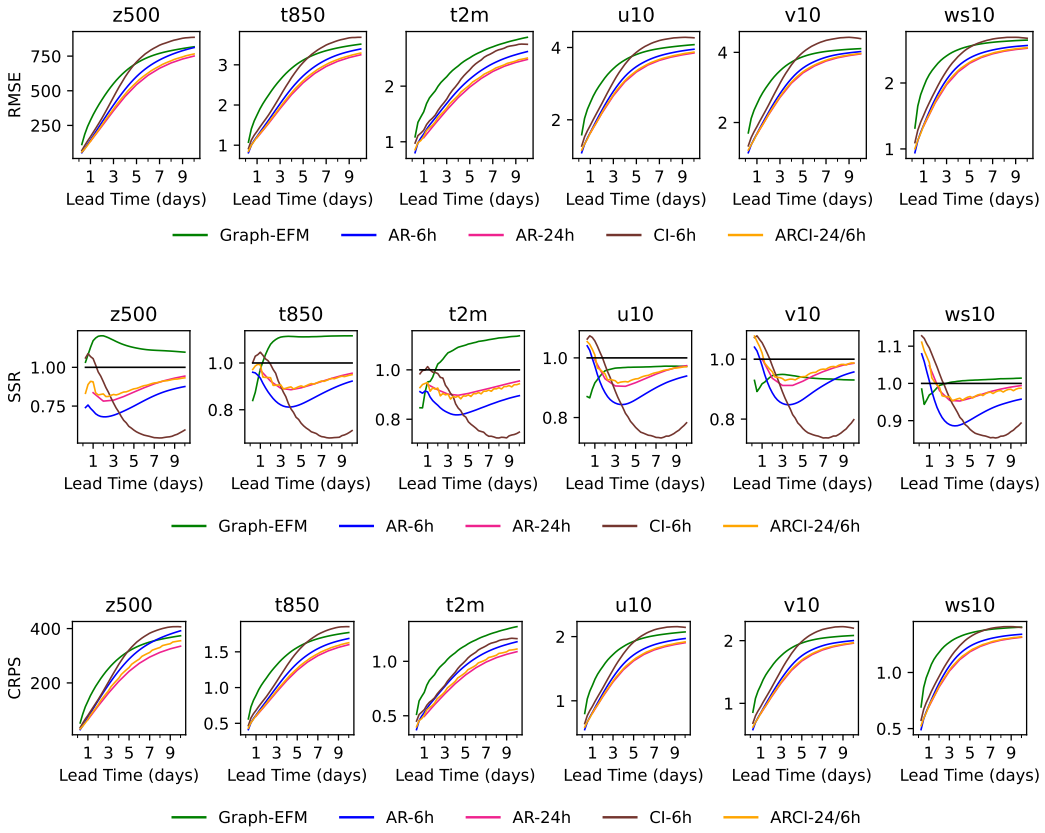

Figure 10: RMSE, SSR, and CRPS for a selection of models at 6h resolution.

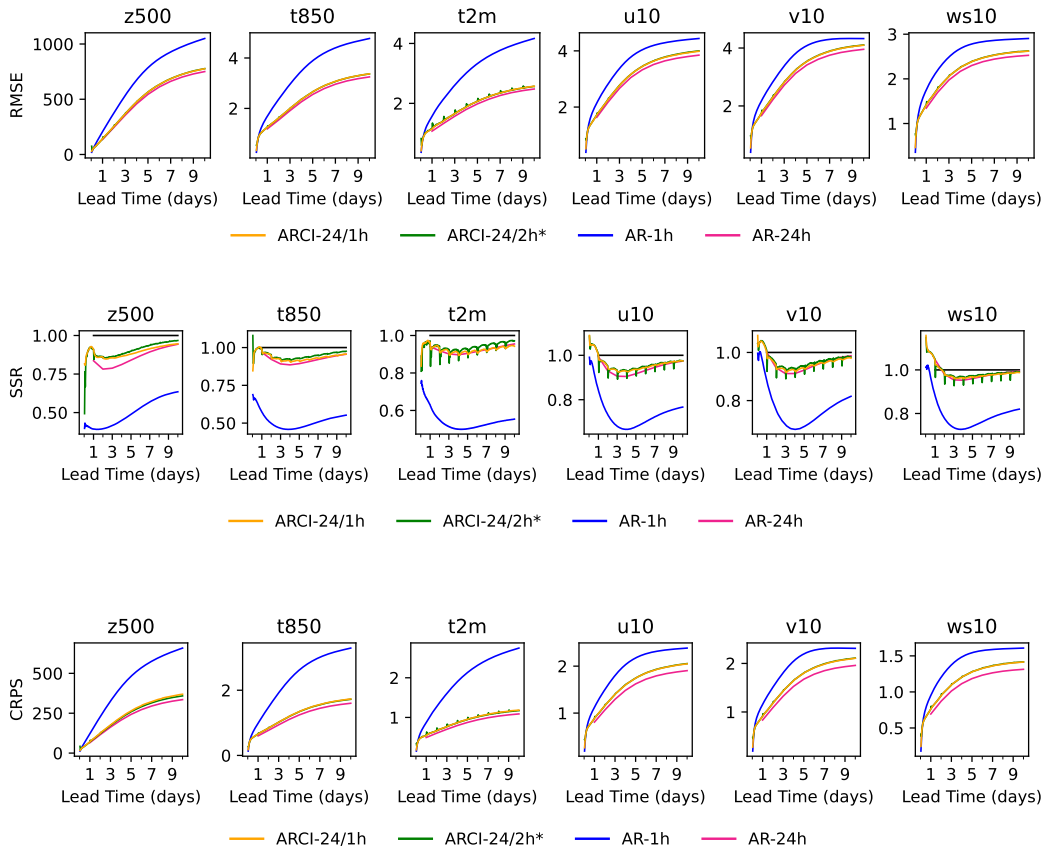

Figure 11: RMSE, SSR and CRPS for a selection of models at 1h resolution.

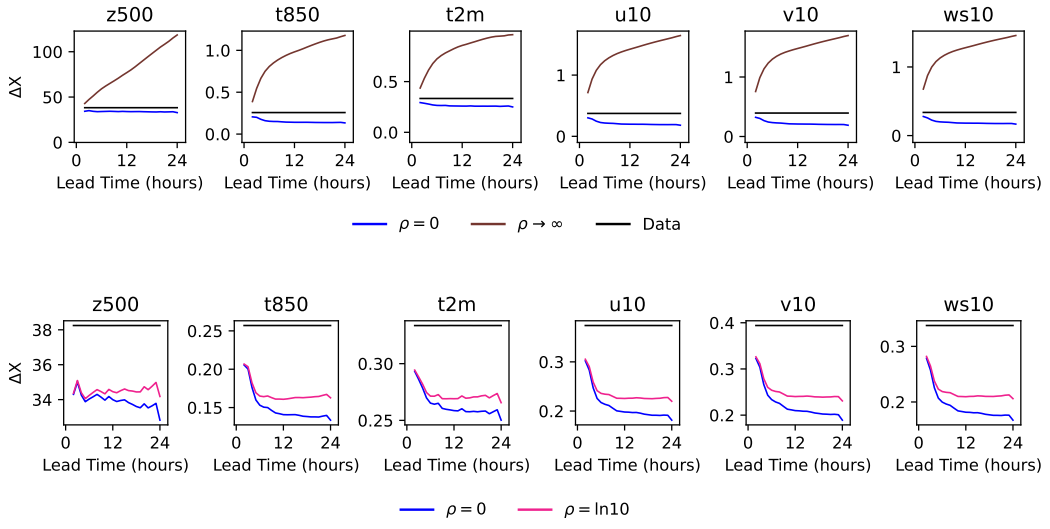

Figure 12: Temporal difference for different values of ϵ in algorithm 2. Choosing $\rho = 0$ fixes the noise, $\rho = \ln(10)$ allows it to vary slightly and $\rho \to \infty$ gives completely uncorrelated noise. The black line refers to the temporal difference of the data.

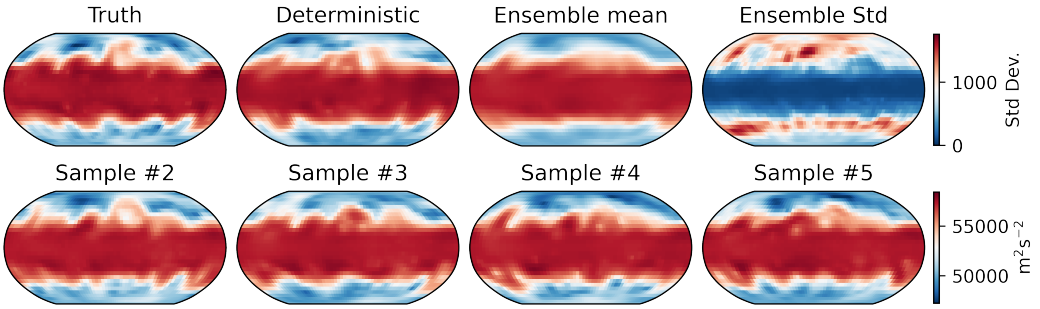

Figure 13: Example forecasts for geopotential at 500 hPa (`z500`) at lead time 10 days. The forecasts are generated using ARCI-24/6h except for Deterministic which is sampled using the deterministic model. The bottom row shows 4 ensemble members, randomly chosen out of the 50.

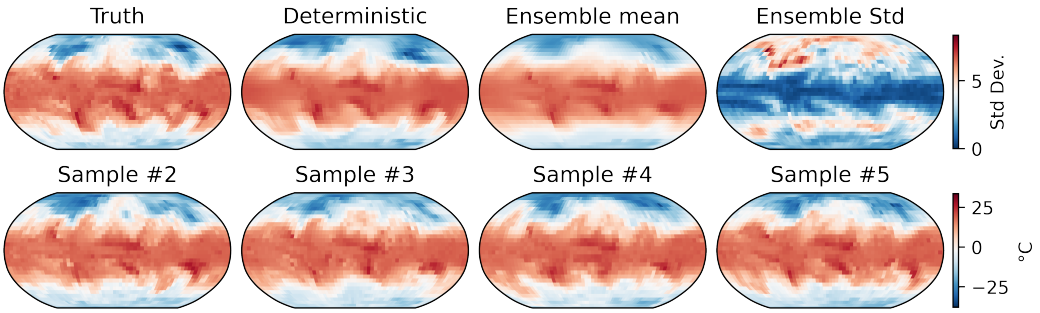

Figure 14: Example forecasts for temperature at 850hPa (`t850`) at lead time 10 days. The forecasts are generated using ARCI-24/6h except for Deterministic which is sampled using the deterministic model. The bottom row shows 4 ensemble members, randomly chosen out of the 50.

D EXAMPLE FORECASTS

Figures 13–18 show example forecasts for the remaining variables.

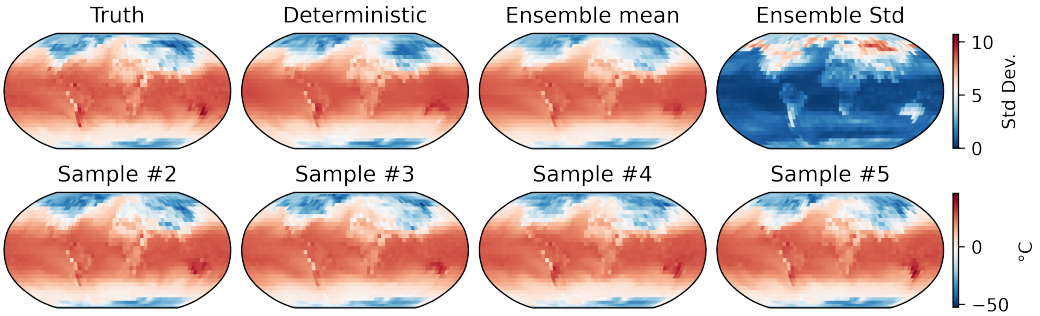

Figure 15: Example forecasts for ground temperature (`t2m`) at lead time 10 days. The forecasts are generated using ARCI-24/6h except for Deterministic which is sampled using the deterministic model. The bottom row shows 4 ensemble members, randomly chosen out of the 50.

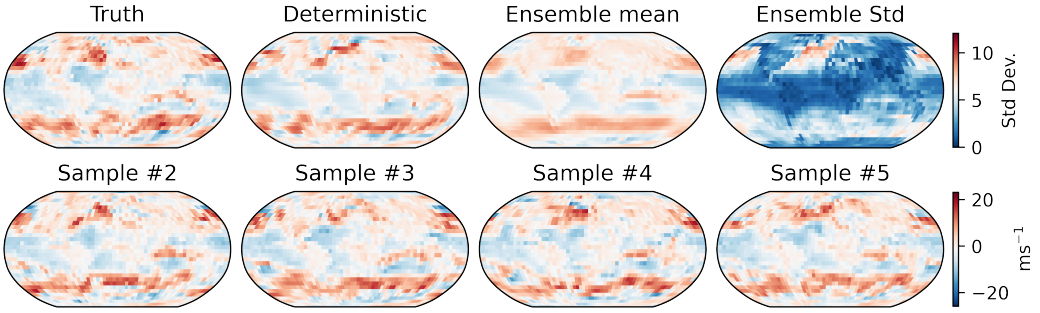

Figure 16: Example forecasts for u-component of wind at 10m (`u10`) at lead time 10 days. The forecasts are generated using ARCI-24/6h except for Deterministic which is sampled using the deterministic model. The bottom row shows 4 ensemble members, randomly chosen out of the 50.

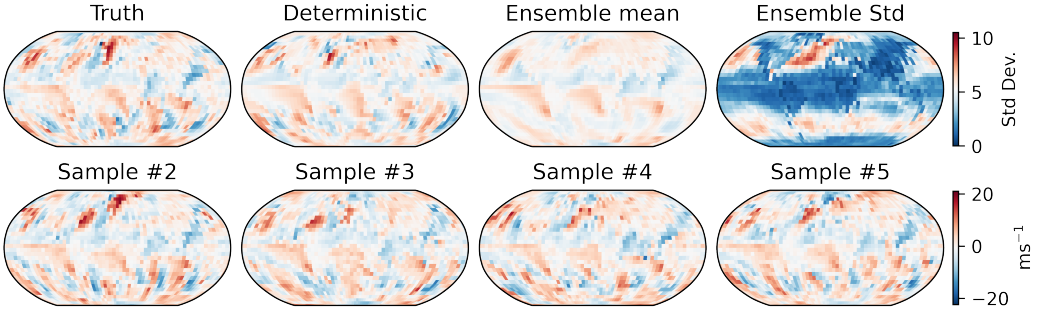

Figure 17: Example forecasts for v-component of wind at 10m (`v10`) at lead time 10 days. The forecasts are generated using ARCI-24/6h except for Deterministic which is sampled using the deterministic model. The bottom row shows 4 ensemble members, randomly chosen out of the 50.

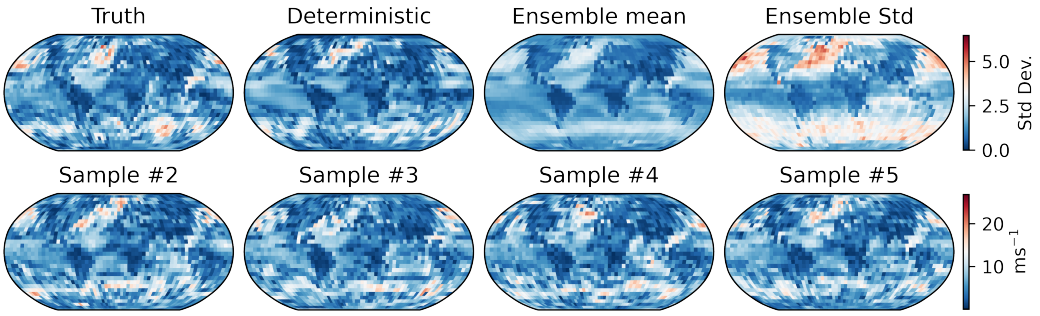

Figure 18: Example forecasts for wind speed at 10m (`ws10`) at lead time 10 days. The forecasts are generated using ARCI-24/6h except for Deterministic which is sampled using the deterministic model. The bottom row shows 4 ensemble members, randomly chosen out of the 50.

E CONTINUITY OF SOLUTIONS TO PROBABILITY FLOW ODE

In section 4.1, we conjecture that the trajectories generated by the Continuous Ensemble Forecasting method are continuous as functions of lead time. This property is verified in practice, as seen in figure 5, but no formal proof of this is presented. In this appendix we show that the solutions to the Probability Flow ODE 2 are continuous functions of t. Since we are primarily interested in the continuity of solutions, we assume access to a unique solution to 2. The existence and uniqueness of such a solution can be proven under the same conditions as theorem 1 below. For a proof of this, we refer the reader to theorem B in Section 70, Chapter 13 of Simmons (2016).

The continuous dependence of solutions to differential equations on parameters is a well-studied problem and there exists several necessary and sufficient conditions for this (2014). We provide a proof for the linear noise level $\sigma(s) = s$ case used in the paper. We want to show that the solution $\boldsymbol{z}(s)$ to the probability flow ODE

$$\mathrm{d}\boldsymbol{z}(s) = -sS_\theta(\boldsymbol{z}(s); x_0, s, t)\mathrm{d}s, \quad \boldsymbol{z}(1) = \boldsymbol{z}_0, \quad s \in [\epsilon, 1] \tag{8}$$

is continuous for all $s \in [\epsilon, 1]$ with respect to the lead time t. We consider the case where the noise $\boldsymbol{z}_0$ is fixed for all t. Here we have chosen to stop at $\boldsymbol{z}(\epsilon)$ for some $\epsilon \ll 1$ instead of $\boldsymbol{z}(0)$ to avoid the singularity of the score function $\nabla \log p_s$ in $s = 0$. This is common for all diffusion models and done in practice for numerical stability.

Let $f(s, \boldsymbol{z}, t) = -sS_\theta(\boldsymbol{z}(s); x_{-1:0}, s, t)$, where we have suppressed the dependence on the previous state $x_{-1:0}$ since all forecasts are conditioned on a fixed $x_{-1:0}$. The following theorem covers the sufficient conditions for continuity of solutions to 8.

Theorem 1 (Continuity of trajectories on lead time t)**.** *Suppose $f(s, \boldsymbol{z}, t)$ is continuous and locally Lipschitz in $\boldsymbol{z}$ in an open set $D \times (0, T) \subseteq \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}$. If $\phi(s, t)$ is a solution of the IVP*

$$\dot{\boldsymbol{z}}(s) = f(s, \boldsymbol{z}, t), \quad \boldsymbol{z}(1) = \boldsymbol{z}_0,$$

which is defined on the closed interval $[\epsilon, 1]$ and $(s, \phi(s, t), t) \in D$ for $s \in [\epsilon, 1]$, then the function $\phi(s, t)$ is continuous on $[\epsilon, 1] \times (0, T)$.

The proof of theorem 1 is based on the proof of theorem 5.5 in (2014) with variables $x = \boldsymbol{z}, \lambda = t, t = s$ which we provide here for reference.

Theorem 2 (Theorem 5.5 in (2014))**.** *Suppose $f(t, x, \lambda)$ is continuous and locally Lipschitz in x in an open set $D \subseteq \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^k$. If $\phi(t, a, x_0, \lambda_0)$ is a solution of the IVP*

$$\dot{x} = f(t, x, \lambda_0), \quad x(a) = x_0,$$

which is defined on the closed interval $[a, b]$ and $(t, \phi(t, a, x_0, \lambda_0), \lambda_0) \in D$ for $t \in [a, b]$, then there is a neighborhood V of (a, x_0, λ_0) in $\mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^k$ such that, for $(u, y, \lambda) \in V$, the IVP

$$\dot{x} = f(t, x, \lambda), \quad x(u) = y,$$

also has a solution defined on the interval $[u, b]$. Moreover the function $\phi(t, u, y, \lambda)$ is continuous on $[a, b] \times V$.

The full proof of theorem 2 is given in (2014). By closely examining this proof, we provide a proof-sketch of theorem 1.

Proof sketch of Theorem 1. Suppose $f(s, \boldsymbol{z}, t)$ is continuous and locally Lipschitz in $\boldsymbol{z}$ in an open set $D \times (0, T) \subseteq \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}$. Let $\phi(s, t)$ be a solution of the IVP

$$\dot{\boldsymbol{z}}(s) = f(s, \boldsymbol{z}, t_0), \quad \boldsymbol{z}(1) = \boldsymbol{z}_0,$$

defined on the closed interval $[\epsilon, 1]$ such that $(s, \phi(s, t), t) \in D$ for $s \in [\epsilon, 1]$. Theorem 2 implies the existence of a neighborhood V of $(1, \boldsymbol{z}_0, t_0) \in \mathbb{R} \times \mathbb{R}^n \times (0, T)$ for the IVP

$$\dot{\boldsymbol{z}}(s) = f(s, \boldsymbol{z}, t), \quad \boldsymbol{z}(u) = y,$$

for which the solution $\phi(s, u, y, t)$ is continuous on $[u, b] \times V$. We note however that this only guarantees that the solutions are continuous on a subset of the the entire interval $(0, T)$. To prove theorem 1 we need to show that the t-part of V can be chosen as $(0, T)$

To see this, we begin by noting that the proof of theorem 5.5 never actually restricts the t-part of V. This can be seen by following the proof steps and keeping track of the restrictions on V. V is first defined in the proof of theorem 5.2 (p.2 (2014)) where it is explicitly constructed as

$$V = I_\alpha(1) \times B_{2\beta}(\boldsymbol{z}_0) \times C_\gamma(t_0),$$

where $I_\alpha(1), B_{2\beta}(\boldsymbol{z}_0)$ is some neighborhood of $(1, \boldsymbol{z}_0)$. These are constructed using some unknown Lipschitz constant and can thus not be explicitly written down.

We note however, that the only requirement on the neighborhood $C_\gamma(t_0)$ of t_0 is that it should lie in the domain $C_\gamma(t_0) \subset (0, T)$. Thus we can simply choose γ, t_0 such that $C_\gamma(t_0) = (0, T)$. Finally, we note that the remaining steps in the proof of theorem 2 never restricts or redefines $C_\gamma(t_0)$. Thus, for this choice of V, the solutions to the IVP will be continuous for all $t \in (0, T)$. This completes the proof. □

Theorem 1 shows that the solutions to the probability flow ODE 8 are continuous as functions of lead-time if $sS_\theta(\boldsymbol{z}(s); x_{-1:0}, s, t)$ is continuous and locally Lipschitz in $\boldsymbol{z}$ in an open set $D \subseteq \mathbb{R} \times \mathbb{R}^n \times (0, T)$. Since the neural network S_θ is a composition of continuous functions, it is necessarily a continuous function. The remaining assumption on S_θ being Lipschitz in $\boldsymbol{z}$ is not a particularly strong one, and commonly assumed when proving results about neural networks (Karras et al., 2022; Song et al., 2019; Albergo & Vanden-Eijnden, 2023). To calculate the Lipschitz constant one could for example use the method proposed in Scaman & Virmaux (2019).