Monte Carlo Dropout for Weather Forecasting: Limitations and Lessons from Aurora

by Frederik Bode Thorbensen (20203483)

Abstract

This thesis investigates the use monte carlo dropout as a bayesian approximation for uncertainty estimations in machine learning-based weather prediction (MLWP) systems. While recent transformer-based foundation models like Aurora demonstrate strong deterministic performance, they offer limited insight into forecast uncertainty without using an ensemble of perturbed or fine-tuned models. This approach is computationally expensive, and therefore, this project explores MC-Dropout, a scalable approximation to Bayesian inference, that simulate a diverse ensemble forecaster from a single source models. Our experiments assess the calibration, reliability, and predictive skill of such stochastic ensembles compared to deterministic baselines, focusing on surface variable predictions such as 2-meter temperature (2t) and wind speed (10u, 10v) against established reanalysis benchmark datasets. Results highlight the importance of probabilistic modelling in medium long-range weather forecasting and provide insight into the trade-offs between ensemble diversity and computational cost.

Keywords: Bayesian Deep Learning, Earth-System Modelling, Monte-Carlo Dropout

1 Introduction

Accurate earth system forecasts are crucial for climate resilience, energy production, agriculture and other public interest. Traditional methods, though powerful, are computationally intensive and often limited in resolution or frequency. Recent breakthroughs in Machine Learning-based Weather Prediction (MLWP) systems like Aurora, GenCast, and Pangu-Weather, shows that machine-learning models can achieve skilful, fast and high-resolution forecasts, on par with numerical weather predictions systems (NWP).

Modern Earth-system modelling are positioned in two groups, where the traditional NWP systems like ENS from European Centre for Medium-Range Weather Forecasts (ECMWF, 2022) relies on physics based simulations of the atmosphere which is both slow and computationally expensive to produce. Development in MLWP, with models such as Aurora (Bodnar et al., 2024), GenCast (Price et al., 2025) and GraphCast (Lam et al., 2023) introduces a new perspective on earth system modelling, by learning complex patterns in how earth changes in atmospheric and surface variables affects global weather patterns. These modern MLWP models have shown to provide greater accuracy and computational efficiency than traditional NWP systems for both deterministic and non-probabilistic forecasts.

Despite their accuracy, single deterministic forecasts cannot capture the full distribution of possible weather outcomes as outlined by Bauer et al. (2015). Ensemble learning offers a principled way to quantify predictive uncertainty, which is essential for risk-aware decision making.

This work addresses a limitation in the current generation of machine learning-based weather prediction systems: their inability to quantify forecast uncertainty in a computationally inexpensive manner. While recent transformer-based MLWP systems have demonstrated strong deterministic performance, they offer limited insight into the range of plausible future states, an essential requirement for high-stakes applications such as disaster response, and climate adaptation. Ensemble forecasting is a well-established approach in both numerical weather prediction (NWP) and machine learning-based weather prediction (MLWP), typically achieved by perturbing initial conditions, varying model physics, or combining multiple fine-tuned models. However, applying these techniques to large ML foundation models is computationally prohibitive: training and maintaining an ensemble of such models can require multiple A100 GPUs, gradient checkpointing, reduced precision, and access to terabyte-scale datasets. In this work, We investigate Monte Carlo Dropout as a significantly cheaper alternative for uncertainty estimations. Unlike full model ensembles, MC-Dropout can be applied at inference time to a single trained foundation model, potentially offering calibrated forecast distributions at a fraction of the computational cost.

This study evaluates ensembles of foundation models over period in January 2022. We assess forecast skill, calibration, and uncertainty growth over lead time. To do so, we evaluate model performance over a time window of January 1st to January 5th, 2022, using 00:00 and 12:00 UTC initializations, and a 5-day forecast horizon. This limited window enables detailed analysis within computational resource constraints, while remaining somewhat

representative for assessing forecast skill, uncertainty calibration, and lead-time error growth. In addition we assess the ensemble via skill evaluation metrics such as Root-Mean Square Error, Anomaly Correlation Coefficient and uncertainty metrics such as Spread, Variance and Spread-to-Skill Ratio. Therefore, to systematically investigate the potential benefits of low-cost heterogenous ensembles, we formulate the following hypotheses:

H1: Forecast uncertainty in MC-Dropout ensembles increases with lead time due to error accumulation in autoregressive rollouts.

H2: Increasing ensemble size (m) improves both forecast skill and uncertainty calibration.

H3: Increasing dropout rate (p) increases ensemble spread and can improve calibration, but excessive dropout degrades spatial coherence and forecast skill.

2 Related Works

The surge of foundation models in MLWP has fundamentally reshaped the landscape of earth-system forecasting research. These developments align with broader trends seen in deep learning and NLP, where transformer-based ensembles have proven effective in improving accuracy and quantifying predictive uncertainty (Lakshminarayanan et al., 2017; Yu et al., 2024). However, integrating such models into a streamlined ensemble poses challenges due to the drastic increase in complexity inherent to these architectures.

2.1 Foundation Models in Weather Forecasting

MLWP models such as Aurora, GenCast, and GraphCast represent a new generation of data-driven approaches to Earth system modelling. Unlike traditional numerical weather prediction (NWP) systems that solve partial differential equations (PDEs), these models learn to forecast from both surface-level and atmospheric variables directly from historical reanalysis data. Despite architectural variations, most foundation models for MLWP follow a shared structural pattern composed of three stages: (1) an **encoder**, which maps physical inputs (e.g., temperature, geo-potential, wind fields) into a latent space representation; (2) a **processor**, which models the temporal evolution of this representation; and (3) a **decoder**, which projects the latent state back into physical variables aligned with the original input format. Forecasts are produced by feeding the model's predictions at a given time step as inputs for the next, thereby, facilitating autoregressive rollouts. Aurora employs a 3D-Swin Transformer processor that operates over spatio-temporal cubes to model Earth's dynamic systems (Bodnar et al., 2024). GraphCast, in contrast, uses a diffusion-based message-passing graph neural network, emphasizing spatial locality and structured inductive biases (Lam et al., 2023). GenCast introduces uncertainty-aware probabilistic forecasting via a diffusion-based generative framework (Price et al., 2025). These architectural differences reflect ongoing experimentation with how best to capture the underlying dynamics of the atmosphere using learned representations.

2.2 Ensembles of Deep Neural Networks

Ensemble learning is a cornerstone of robust machine learning. Traditionally, it has been used to improve generalization and reduce variance through techniques such as bagging, boosting, and stacking (Breiman, 1996, 2001). These methods combine multiple weak learners, such as decision trees or shallow neural networks, into a stronger predictive model by aggregating their outputs through averaging or majority voting. Although originally developed for simple models, ensembling or model fusing have gained substantial traction in deep learning. In particular, they provide a practical and effective means of estimating predictive uncertainty, which is critical in high-stakes domains such as medical diagnosis, autonomous driving, and weather forecasting. One influential technique in this space is **Monte Carlo Dropout**, introduced by Gal and Ghahramani (2016). This method enables approximate Bayesian inference by activating dropout layers at inference, thereby generating stochastic predictions from a single trained model. Each forward pass samples a different subnetwork, effectively producing a lightweight ensemble without retraining. This approach offers a scalable alternative to full Bayesian neural networks and is particularly attractive for large models where full posterior inference is infeasible. Another prominent method is **Deep Ensembles** (Lakshminarayanan et al., 2017), which trains multiple instances of the same model architecture using different random initializations or data folds. The resulting ensemble captures epistemic uncertainty through model diversity and has been shown to outperform many Bayesian approximations in practice, while remaining simple to implement. These techniques form the foundation of **Bayesian Deep Learning**, a field that aims to quantify uncertainty in deep neural networks by approximating posterior

distributions over model parameters or predictions. While exact Bayesian inference remains intractable for large neural networks, these practical approximations provide a principled framework for predictive uncertainty estimation. In the context of machine learning-based weather prediction (MLWP), such uncertainty-aware methods are especially relevant. Forecasting applications, ranging from disaster response, climate modelling to renewable energy production and not only for accurate predictions but also on reliable measures of uncertainty. This thesis builds on these insights by applying Monte Carlo Dropout to a large-scale weather foundation model to assess the feasibility and benefits of stochastic ensemble forecasting in Earth system science.

2.3 Ensembles in Numerical Weather Prediction

The foundations of ensemble forecasting in weather science can be traced back to the early insights of Poincaré, Thompson, and Lorenz. Poincaré first recognized that small perturbations in non-linear systems can lead to vastly different forecasts (Bauer et al., 2015). Later, Lorenz formalized these ideas in his seminal work on atmospheric predictability and chaos theory (Lorenz, 1963), showing that finite, state-dependent limits constrain long-term forecast skill. This led to the development of ensemble-based techniques to characterize the evolution of initial condition uncertainties, model errors, and their interactions with the atmospheric state. Traditional ensemble methods in numerical weather prediction (NWP) generate forecast distributions by perturbing initial conditions or model physics (Leutbecher and Palmer, 2008). These ensembles quantify forecast confidence and enable risk-aware decision-making. While effective, operational NWP ensembles like the ENS system (ECMWF, 2022), despite being state-of-the-art, is computationally intensive (Pathak et al., 2022; Rasp et al., 2024). Ensemble forecasting in NWP typically involves perturbations of initial conditions or stochastic physics to generate multiple forecast trajectories. These ensembles help quantify forecast uncertainty and improve robustness. In the MLWP context, current approaches draws on the same principles, i.e. train multiple variations of the same model. However, there is limited work on ensembling ML foundation models for earth-system forecasting via Bayesian Approximations such as MC-Dropout.

2.4 Ensembling Foundation Models

Outside of earth-system modelling, the field of natural language processing (NLP) has seen the emergence of sophisticated ensemble methods for combining foundation models with heterogeneous architectures, tokenizers, and generation styles. Techniques such as DEEPen (Huang et al., 2024), GaC (Yu et al., 2024), and UNiTE (Yao et al., 2025) address key challenges in ensemble diversity. While these approaches demonstrate promising results in NLP, their applicability to machine learning-based weather prediction (MLWP) is limited. In contrast to NLP models—which often share a common input format (token sequences) and generate low-dimensional outputs (token probabilities) earth system models operate on complex, high-dimensional spatio-temporal data. Each model may differ in grid resolution, timestep frequency, and required input variables (e.g., pressure levels, solar radiation, or soil moisture), making input alignment and output aggregation significantly more challenging. Furthermore, MLWP models are substantially more computationally demanding. Whereas many open-source large language models can be deployed on a single pc, weather models often require tens of gigabytes of GPU memory per forward pass due to the scale and richness of their output fields. This makes large-scale heterogeneous ensembling impractical under typical academic resource constraints. Finally, techniques such as deep ensembles, where models are trained on different data partitions, pose additional risks in autoregressive weather forecasting. Temporal discontinuities between training folds may degrade the model's ability to learn coherent spatio-temporal evolution, which is essential for accurate multistep forecasting. Given these challenges, Monte Carlo Dropout offers a practical and scalable alternative. It enables the generation of ensemble forecasts from a single deterministic model with minimal architectural or computational overhead. This approach avoids the complexities of model alignment and leverages a well-understood Bayesian approximation to deliver meaningful uncertainty estimates within a consistent model framework.

3 Background

Reliable weather forecasting depends fundamentally on our ability to model the Earth system: the coupled dynamics of the atmosphere, oceans, land surface, and cryosphere. Traditional methods approach this task through numerical weather prediction (NWP), which relies on solving the governing equations of atmospheric physics. While these physics-based simulations offer high accuracy, they are computationally expensive and require careful tuning of sub-grid processes. In recent years, the rise of deep learning has enabled an alternative, where data-driven models learn the statistical evolution of atmospheric states directly from historical reanalysis data. This section outlines the progression from traditional Earth-system models to learned simulators in

MLWP. We define the weather prediction task as a sequence modelling problem, describe how autoregressive forecasting is used to simulate multi-step trajectories, and explain the functional role of simulators within ML-based forecasting systems. Finally, we introduce ensemble forecasting as a principled method for quantifying predictive uncertainty.

3.1 Earth-System Modelling and Autoregressive Forecasting

The goal of Earth-system modelling is to forecast future global states by simulating how weather and climate variables evolve over time. From a computational perspective, this task can be framed as predicting the next system state x^{t+1} conditioned on one or more previous states. In traditional NWP, this is achieved using numerical solvers for partial differential equations. In MLWP, the same objective is pursued by training a *learned simulator* Φ, i.e. a deep learning based model, that learns the temporal dynamics of earth-system variables.

This predictive structure supports an *autoregressive rollout*, where forecasts are recursively generated from prior predictions:

$$\Phi(x^{t-1}, x^t) \to \hat{x}^{t+1}, \quad \Phi(x^t, \hat{x}^{t+1}) \to \hat{x}^{t+2}, \quad \ldots \tag{1}$$

Each rollout step conditions on a short input history, and the predicted state ($\hat{x}$) is fed back into the model as input for the next step. This chaining mechanism transforms a single-step predictor into a multi-step simulator. However, it also introduces compounding errors, as inaccuracies at early steps propagate through the rollout. Figure 1: illustrates this auto-regressive behaviour.

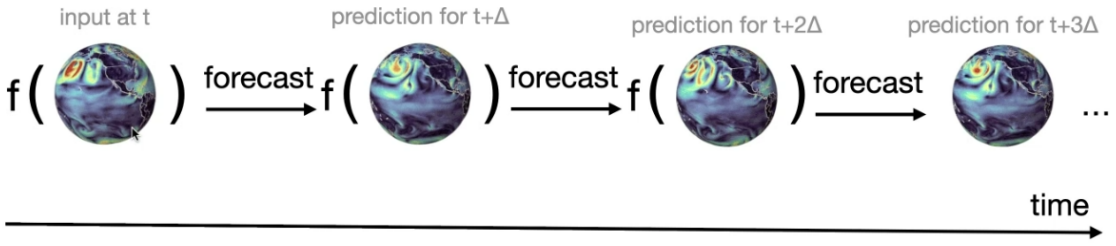

Figure 1: Illustration of autoregressive forecasting using a learned simulator Φ. At each time step, the model predicts the future state of the Earth system based on the most recent state(s), and the output is recursively fed back as input for future predictions. Illustration from: Machine Learning Reading Group @ CUED, ML for Medium-Range Weather Forecasting, University of Cambridge, Diaconu et al. (2025).

Modern MLWP simulators such as Aurora (Bodnar et al., 2024), GenCast (Price et al., 2025), and GraphCast (Lam et al., 2023) use deep transformer architectures to perform spatiotemporal modeling at global scales. Aurora, for instance, employs a 3D Swin Transformer with Perceiver-style encoders to process multivariable, multi-resolution Earth data and generate next-state predictions across key atmospheric fields.

Formally, the simulator learns a function $\Phi : \mathcal{X}^{t-n:t} \to \hat{x}^{t+1}$ that minimizes a loss function over training data derived from reanalysis datasets such as ERA5, and HRES. This loss is typically defined as mean squared error (MSE) or mean absolute error (MAE) between predicted and ground-truth fields.

3.2 Input Representation and State Encoding

Earth-system models require a structured representation of the Earth system state to generate forecasts. This representation is typically defined on a regular latitude–longitude grid, where variables are discretized at fixed spatial intervals in degrees, forming a rectilinear mesh that spans the entire globe. It consists of three main components: surface variables, atmospheric variables across pressure levels, and static geophysical inputs.

Surface variables include fields such as 2-meter temperature (*2t*), 10-meter zonal and meridional wind components (*10u*, *10v*), and mean sea-level pressure (*msl*). These are defined on the regular equirectangular grid and vary over time. **Atmospheric variables** are sampled at discrete vertical pressure levels, often including

100, 250, 500, and 850 hPa. Variables such as geopotential (z), temperature (t), humidity (q), and wind components (u, v) form a vertical profile at each horizontal location, enabling the model to capture three-dimensional atmospheric dynamics. Conceptually, one can imagine the global grid extending upward through the atmosphere, replicated across each pressure level to form a stacked tensor representation. **Static variables** provide time-invariant information such as potential (z), land-sea mask (lsm), and soil type (slt). Although constant in time, they condition the model's understanding of how weather systems behave locally. See appendix A, for a variable overview.

Each variable group is discretized on a shared spatial grid and batched over time steps. The atmospheric component introduces an additional dimension for pressure levels, effectively stacking multiple layers of the atmosphere above each grid cell. Together, these inputs form a high-dimensional tensor field encoding the Earth's current state. This input structure is then transformed into a latent representation by the model's encoder. In the case of Aurora, a Perceiver-style architecture processes the surface and atmospheric inputs using pressure-level encodings, spatial Fourier embeddings, and time encodings to produce a unified 3D latent field. This internal representation is then recursively evolved to simulate future Earth system states through autoregressive forecasting.

3.3 Dropout or DropPath Regularization

Dropout (Srivastava et al., 2014) is a widely used regularization technique in deep learning. During training, a fraction of activations is randomly set to zero, preventing co-adaptation and improving generalization. At inference, dropout is typically disabled and weights rescaled, yielding deterministic outputs. Gal and Ghahramani (2016) showed that keeping dropout active at test time and averaging multiple stochastic forward passes approximates sampling from a *Bayesian posterior distribution*. This **Monte Carlo Dropout** (MC-Dropout) enables uncertainty estimation without training multiple models.

In modern architectures such as Vision Transformers and 3D Swin Transformers, including *Aurora*, dropout on activations is often replaced by stochastic depth or **DropPath** (Huang et al., 2016), which randomly skips residual blocks during training. This provides more effective regularization for very deep networks while preserving spatial structure. Aurora is trained with $drop_rate = 0$ (no classical dropout) and $drop_path_rate = 0.2$, meaning that in principle, MC-DropPath would be the most consistent Bayesian approximation. However, as Aurora only exposes dropout hooks at inference time, our experiments use MC-Dropout to approximate epistemic uncertainty.

To evaluate its effect on Aurora outputs, we conducted a sensitivity analysis across dropout rates, as shown in Figure 2, increasing dropout introduces progressively stronger stochastic perturbations, especially in mid-latitude and coastal regions. At low rates, forecasts remain close to deterministic predictions, whereas high rates produce excessive speckling and reduced spatial coherence. This reflects the well-known trade-off identified by Gal and Ghahramani (2016): higher dropout enhances epistemic uncertainty representation but degrades predictive fidelity.

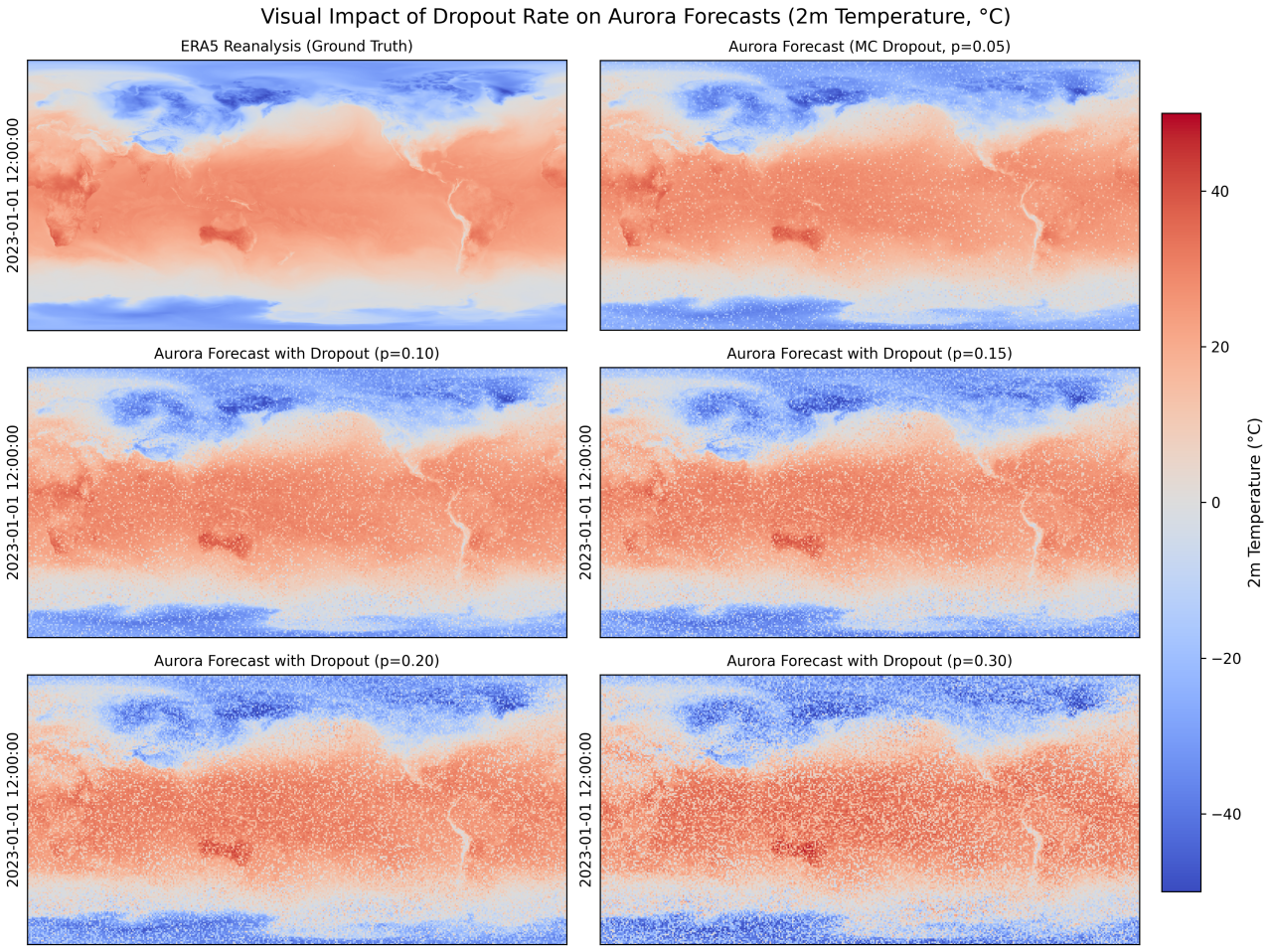

Figure 2: Visual impact of varying dropout rates on Aurora forecasts for 2-meter temperature (2t) on January 1st, 2022 at 12:00 UTC. The top-left panel shows the ERA5 reanalysis used as ground truth. The subsequent panels display Aurora forecasts generated with increasing drop rates.

4 Ensembles of MLWP Models

In real-world forecasting, it is not enough to produce a single deterministic trajectory. Atmospheric systems are chaotic and inherently uncertain small variations in inputs can lead to widely different outcomes. As mentioned earlier, this sensitivity was first highlighted by Poincaré and later formalized by Lorenz in his work on atmospheric predictability and chaos theory (Lorenz, 1963). To manage this uncertainty, ensemble forecasting has become a central tool in operational meteorology. Traditional ensemble methods in NWP quantify uncertainty by perturbing initial conditions or model parameters to produce a set of plausible future trajectories (Leutbecher and Palmer, 2008). However, in MLWP, a different opportunity arises: we can construct ensembles by aggregating the predictions of independently trained simulators. Each simulator Φ_i may differ in architecture, training data, loss functions, or initialization seeds capturing a broader spectrum of modeling assumptions. Let x^t denote the current state. A simple mean ensemble forecast at step $t+2$ can be defined as:

$$\Phi_{\text{ens}}(x^t, \hat{x}^{t+1}) = \frac{1}{k} \sum_{i=1}^{k} \Phi_i(x^t, \hat{x}^{t+1}) \to \hat{x}^{t+2} \tag{2}$$

This ensemble aggregates the next-step predictions from k simulators. Beyond the mean, we can also compute pointwise standard deviations across members to obtain spatial uncertainty estimates. This captures *epistemic uncertainty* arising from limited model knowledge rather than aleatoric uncertainty from measurement noise.

In practice, heterogeneous ensembles in MLWP can combine models trained on different reanalysis datasets (e.g., ERA5, MERRA-2), or architectures tuned for different trade-offs between resolution and speed. These ensembles are especially valuable in long-range forecasts, where autoregressive error accumulation makes reliable uncertainty estimates essential for downstream decision-making.

5 Methods

5.1 Model Selection

We evaluated three state-of-the-art global forecasting models Aurora (Microsoft), GraphCast (Google DeepMind), and Pangu-Weather (Huawei) against three criteria: (1) open-source (2) easy integration (3) accessible drop-rate parameter tuning.

We first considered **GraphCast** and **Pangu-Weather**, which we ultimately did not adopt. GraphCast, despite strong benchmarks, is not distributed as an installable package; it must be cloned from source (JAX/Haiku) and assembled with parameters stored in separate Google Cloud buckets. In practice, the lack of modular wrappers and orchestration made integration and autoregressive rollouts fragile and time-consuming, impeding rapid iteration. Pangu-Weather offers pretrained ONNX models and multiple temporal resolutions, an attractive flexibility for custom pipelines. However, core research plumbing, like rollout logic, data handling, and experiment harnessing remains largely do-it-yourself, which slowed the pace required for our uncertainty experiments.

By contrast, **Aurora** aligned cleanly with our methodological and practical needs. It is shipped as a production-grade Python package with multiple model sizes and spatial resolutions, uses only ERA5 initialization (without extra conditioning inputs), and produces hourly forecasts—supporting fine-grained analysis of uncertainty propagation in autoregressive rollouts. Crucially, Aurora is implemented in PyTorch and exposes inference-time dropout controls, enabling faster development of our MC-Dropout ensembles from a single deterministic architecture. Given these considerations, **Aurora** was selected as the primary candidate for our experiments, offering the optimal balance between model quality, practical usability, architectural transparency.

5.2 Data and Forecast Protocol

As briefly discussed earlier, we evaluate forecasts initialized at 00:00 and 12:00 UTC from 1–5 January 2022. Each forecast is verified through +120 h (5 days). Target variables are 2-meter temperature (`2t`) and 10-meter wind components (`10u`, `10v`). Verification uses ERA5 reanalysis at 0.25° spatial resolution. This choice ensures cross-model compatibility: Aurora, GenCast, and Pangu-Weather either natively predict ERA5 variables or provide pretrained checkpoints aligned with ERA5-format data.

While ECMWF HRES offers finer resolution (0.1°), it is not uniformly supported across models and would substantially increase data volumes for both truth and forecasts, making large-scale evaluation and ensemble construction impractical. ERA5 at 0.25° provides a balanced trade-off between resolution, reproducibility, and computational cost across all ensemble members.

Forecast generation follows an autoregressive protocol: for each initialization, the model's output at lead t is fed as input to lead $t+1$. This mirrors operational use and makes explicit the accumulation of uncertainty with lead time. Unless stated otherwise, metrics are reported for lead times up to +120 h.

5.3 Dropout Rate Selection for Stochastic Forecasting

Selecting an appropriate dropout rate is critical when applying Monte Carlo Dropout for uncertainty estimation. Too little dropout leads to under dispersed ensemble forecasts with limited epistemic variation, while excessive dropout can degrade spatial coherence and inflate uncertainty beyond useful bounds.

To determine a suitable dropout rate for the Aurora model, we conducted a preliminary sensitivity analysis across a range of dropout values $p \in \{0.05, 0.1, 0.15, 0.2\}$. For each value, we generated sample forecasts for a 24-hour horizon and visually inspected the spatial quality of key surface variables (e.g., 2-meter temperature, 10-meter wind components). Additionally, we monitored changes in spread, forecast smoothness, and the preservation of physically realistic gradients.

Based on this analysis, a dropout rate of $p = 0.1$ was selected for all subsequent MC Dropout experiments. This value provided a good trade-off between ensemble diversity and spatial fidelity, and aligns with findings in prior Bayesian deep learning literature (Gal and Ghahramani, 2016).

All MC Dropout-based ensemble forecasts in this study are generated using this fixed dropout rate, ensuring consistency across ensemble sizes and forecast windows.

5.4 Ensembling Strategy

To estimate predictive uncertainty from a single foundation model, we employ Monte Carlo (MC) Dropout during inference. This technique involves performing multiple stochastic forward passes through the model with dropout layers active, thereby producing an ensemble of plausible forecasts from the same model configuration.

At each timestep in the autoregressive rollout, we collect the output fields from multiple MC forward passes. These outputs are stacked along a newly introduced ensemble dimension for each surface and atmospheric variable. From this stacked set of predictions, we compute:

Mean ensemble forecast: At each timestep, we perform multiple stochastic forward passes through the model using MC Dropout, yielding an ensemble of predictions. The pointwise (per-grid-point) mean across these ensemble members serves as the best estimate of the predicted atmospheric or surface state. This mean forecast is then used as the unified input x^{t+1} for the next timestep across all ensemble members, ensuring consistent autoregressive progression.

Uncertainty estimation: Alongside the mean, we compute the pointwise standard deviation and variance across ensemble members, capturing spatial patterns of predictive uncertainty. By evaluating these statistics at each forecast step t, we can monitor how uncertainty evolves over time. This allows us to assess whether regions with higher ensemble spread are also associated with increased forecast error relative to the ground truth—an important indicator of calibration quality.

To streamline this process, we use a wrapper that extracts the relevant surface and atmospheric variables from each forecast, aligns them across dimensions, and aggregates them using standard tensor operations, see Appendix C. The result is a mean prediction batch accompanied by spread metrics that can be evaluated with metrics such as RMSE, spread-to-RMSE ratio, and prediction interval coverage

5.5 Implementation Details

We implement inference in **PyTorch** and distribute stochastic forward passes with **Ray** to parallelize MC-Dropout ensembles across GPUs. Evaluation uses **WeatherBench2** (standardized preprocessing, area weighting, and metrics). Forecast fields are written as chunked **Zarr** datasets and manipulated with **xarray** for labeled, multi-dimensional arrays. Artifacts are stored on **Google Cloud Storage** to support concurrent reads/writes during analysis. These choices are conventional for MLWP and do not alter model behavior beyond enabling efficient parallelism and I/O. Full software versions, cluster configuration, Zarr chunking, and storage layout appear in Appendix B.

6 Evaluation Methodology

This section outlines the methodology used to evaluate the both the deterministic weather forecasting via heterogeneous ensembles of machine learning-based foundation models. The goal is to assess whether ensembles of independently trained models improve forecast skill and uncertainty estimation relative to single-model baselines.

6.1 Research Design

We adopt an empirical evaluation strategy focused on assessing both deterministic and probabilistic performance of ensemble forecasts derived from pretrained foundation models. Our analysis emphasizes forecast accuracy and uncertainty calibration across spatial and temporal dimensions.

Experimental Protocol: All experiments follow a standardized protocol to ensure comparability across runs. Forecasts are initialized at 00:00 and 12:00 UTC during the evaluation period from January 1–5, 2022, with a lead time of 72 hours sampled at 1-hour intervals. Each forecast is evaluated against ERA5 reanalysis data at 0.25° spatial resolution, using the preprocessing and scoring routines defined by the WeatherBench2 framework.

Forecast outputs are stored in Zarr format at full temporal resolution, enabling efficient storage and analysis of high-dimensional atmospheric fields. Unless otherwise stated, the variables, we will evaluate agaist is 2-meter temperature (2t), 10-metre eastward wind component (10u) and 10-metre southward wind component (10v). Performance is assessed using Root Mean Square Error (RMSE), spread-to-RMSE ratio, and Prediction Interval Coverage (PIC).

6.2 Benchmarking Protocol

All forecasts are evaluated against the ERA5 reanalysis dataset from ECMWF, treated as ground truth on a 0.25° equirectangular grid. ERA5 offers globally consistent, high-resolution data and is the standard reference for recent ML-based weather forecasting studies (Bodnar et al., 2024). For rigorous and reproducible evaluation, we adopt the WeatherBench2 (WB2) framework (Rasp et al., 2024), which enforces standardized preprocessing, regridding, spatial weighting, and scoring metrics in line with WMO/ECMWF verification practices.

Evaluation framework: Forecast outputs are regridded to the WB2 target grid using bilinear interpolation, followed by cosine-latitude weighting for area-averaged scores. All verification metrics are computed using the official WB2 codebase and default configurations. Unless otherwise stated, all reported scores, deterministic and probabilistic, are directly generated through this pipeline.

Deterministic metrics: Root-mean-square error (RMSE) and anomaly correlation coefficient (ACC) are computed for all variables. Anomalies are defined relative to ERA5 climatology over 1990–2019, consistent with WB2.

Probabilistic metrics: For MC Dropout-based ensembles, we compute spread–skill ratio (SSR). Ensemble spread is defined as the spatially averaged standard deviation across N stochastic samples. SSR is the ratio of spread to the RMSE of the ensemble mean.

Aggregation and reporting: All scores are global, cosine-area-weighted, and computed at fixed forecast horizons unless otherwise stated. Forecast cycles are aligned to 00/12 UTC initialization times to match operational standards.

6.3 Evaluation Metrics

To evaluate the performance of MC Dropout ensembles of Aurora against a single deterministic model instance, we employ several standard evaluation metrics commonly used in Earth system modeling benchmarks. These include Root Mean Square Error (RMSE) and Anomaly Correlation Coefficient (ACC), both of which assess deterministic forecast skill. We have defined RMSE and ACC, in accordance with the defintions by (Bodnar et al., 2024). Additionally, we evaluate probabilistic performance using ensemble spread metrics such as standard deviation and variance, which provide insight into forecast uncertainty.

Root Mean Square Error (RMSE): is a widely used metric in both statistics and machine learning for evaluating prediction accuracy. It quantifies the average magnitude of the error between forecasted values and observed ground truth. In this context, RMSE is computed between the ensemble mean (or single model output) and ERA5 reanalysis data. To ensure spatial comparability across the globe, the WeatherBench2 evaluation framework applies latitude weighting to account for the non-uniform surface area represented by each grid cell. Specifically, errors at each latitude are weighted by the cosine of the latitude, reflecting the diminishing area near the poles. This adjustment ensures that the global RMSE is not biased toward high-latitude regions.

$$\text{RMSE} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W} w(i)\left(\hat{X}_{i,j}^{t} - X_{i,j}^{t}\right)^2} \tag{3}$$

Anomaly Correlation Coefficient (ACC): is a standard metric in meteorology for assessing the skill of a forecast relative to climatological expectations. Rather than measuring absolute error, ACC evaluates how well the forecast captures anomalies—deviations from the mean climatology—compared to the observed anomalies. This makes ACC particularly suitable for assessing the spatial and temporal consistency of forecasts. In this work, ACC is computed between the ensemble mean (or individual model output) and ERA5 reanalysis data, with all values anomaly-normalized using climatological means derived from the same dataset.

$$\text{ACC} = \frac{1}{T}\sum_{t=1}^{T}\frac{\sum_{i=1}^{H}\sum_{j=1}^{W} w(i)(\hat{X}_{i,j}^{t} - C_{i,j}^{t})(X_{i,j}^{t} - C_{i,j}^{t})}{\sqrt{\left[\sum_{i=1}^{H}\sum_{j=1}^{W} w(i)(\hat{X}_{i,j}^{t} - C_{i,j}^{t})^2\right]\left[\sum_{i=1}^{H}\sum_{j=1}^{W} w(i)(X_{i,j}^{t} - C_{i,j}^{t})^2\right]}} \tag{4}$$

Ensemble Spread and Spread–Skill Ratio (SSR): The spread of an ensemble quantifies the internal variability across ensemble members and serves as a proxy for forecast uncertainty. Ideally, a reliable ensemble

should exhibit a spread that reflects its actual forecast error: wide spread when forecasts are uncertain, and narrow spread when they are confident. To evaluate this calibration, the spread–skill ratio compares the ensemble spread with the RMSE of the ensemble mean forecast. A ratio close to one indicates well-calibrated uncertainty, while values below one imply under-dispersion (overconfident ensembles) and values above one indicate over-dispersion (underconfident ensembles) (Gneiting and Raftery, 2007; Leutbecher and Palmer, 2008).

For an M-member ensemble $A^{v,(m)}i, j, tm = 1^M$ with mean $\bar{A}^v_{i,j,t}$, the (sample) spread is:

$$\text{Spread}(v,t) \;=\; \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}\left(A^{v,(m)}i,j,t - \bar{A}^v i,j,t\right)^2}, \tag{5}$$

computed per grid point and then area-weighted. Calibration is assessed via:

$$\text{SSR}(v,t) \;=\; \frac{\text{Spread}(v,t)}{\text{RMSE}(v,t)}. \tag{6}$$

7 Experiments

While recent foundation models have been evaluated over extended periods—often spanning an entire year—such large-scale inference pipelines demand substantial computational resources, including multiple high-end GPUs (e.g., NVIDIA A100) and extensive storage capacity. In contrast, this thesis adopts a focused evaluation window from January 1–5, 2022. This reduced scope strikes a balance between scientific representativeness and computational feasibility, enabling detailed analysis of forecast skill, ensemble behaviour, and uncertainty estimations within realistic academic constraints. To ensure compatibility with established MLWP benchmarks, forecasts are initialized at standard synoptic times (00:00 and 12:00 UTC) and evaluated on key surface variables including 2-meter temperature (2t) and 10-meter wind components (10u, 10v). Forecasts are computed at hourly resolution over a 120-hour lead time and benchmarked against ERA5 reanalysis data at 0.25° spatial resolution using the WeatherBench2 framework. Output data is stored in compressed Zarr format for efficient access and evaluation. All model inference is executed on NVIDIA L40S GPUs, with one GPU dedicated to each model instance during forecasting. This hardware uniformity ensures a consistent runtime environment across deterministic and stochastic experiments, removing potential discrepancies related to performance variability.

Model: All experiments are conducted using the open-source Aurora foundation model developed by Microsoft (Bodnar et al., 2024).

Experiment 1: Baseline Evaluation of Deterministic Forecasts

This experiment establishes a fair reference point by evaluating deterministic forecasts under identical experimental conditions. While prior studies have reported benchmark results for foundation models such as Aurora, GraphCast, and Pangu-Weather, reusing those values would introduce inconsistencies due to differences in initialization times, spatial resolution, and inference hardware. This allows us to isolate the effects of Monte Carlo Dropout from confounding variables related to infrastructure or data alignment.

Experiment 2: Monte Carlo Dropout Ensembles

In this experiment, we apply dropout at inference time (MC-Dropout) to the Aurora model to simulate diversity in the ensemble. By performing multiple stochastic forward passes per initialization, we generate ensemble forecasts that approximate the posterior predictive distribution over future atmospheric states, capturing epistemic uncertainty within a single architecture. We evaluate ensemble sizes of $m = 3$, $m = 5$, and $m = 8$ to investigate how increasing the number of ensemble members, affects forecast skill and calibration. Each ensemble member is generated using a different dropout mask with the dropout rate fixed at $p = 0.05$, $p = 0.10$, or $p = 0.15$, as determined through prior sensitivity analysis. This experiment directly tests, both hypothesis *H2* and *H3*, while also providing insights into the trade-off between ensemble size and computational cost. We will evaluate these different ensemble configurations against the aforementioned evaluation metrics, Section 6.3, using the benchmarking tool and and framework WeatherBench2, Section 6.2.

7.1 Main Results

This section presents the key findings, structured around the three hypotheses introduced in Section 1. Following the evaluation protocol in Section 6.3, we report RMSE, ACC, and SSR. Unless stated otherwise, all scores are computed by verifying forecasts against ERA5 at 0.25° resolution using the WeatherBench2 (Rasp et al., 2024).

Root Mean Squared Error

Using WeatherBench2, we computed lead-time–resolved RMSE for Aurora (deterministic baseline) and the MC-Dropout variants. Across variables we observe a sharp increase in error for the first few autoregressive steps, followed by a slower growth/plateau at longer leads.

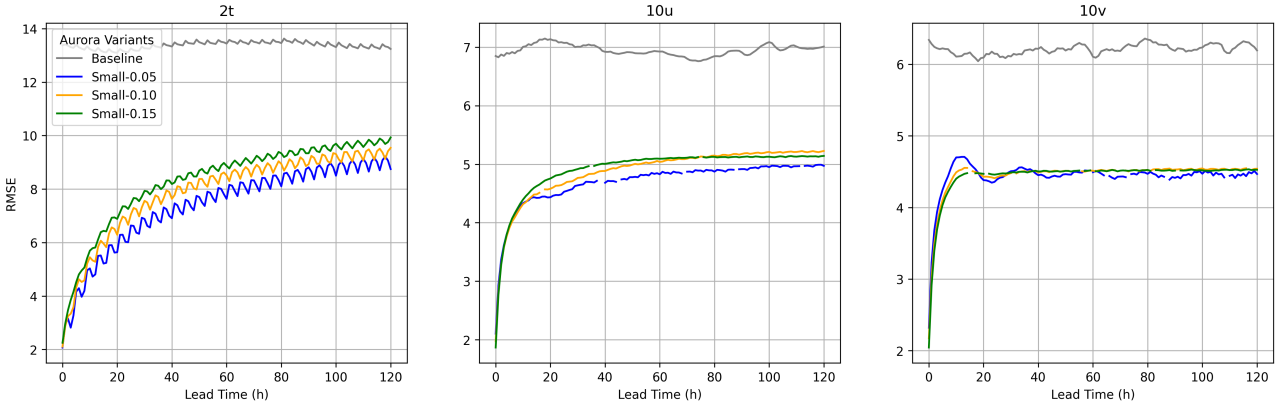

Figure 3: Root Mean Square Error (RMSE) over lead time for Aurora forecasts with stochastic dropout-based ensembling. Each panel shows RMSE for a distinct surface variable: 2-meter temperature (*2t*), 10-meter eastward wind (*10u*), and 10-meter southward wind (*10v*). The plotted line corresponds to Aurora Baseline with configuration.

	RMSE (2t)	RMSE (10u)	RMSE (10v)	Δ_{2t}
Aurora (deterministic)	13.37	6.94	6.21	5.98
Small ($p = 0.05$)	7.39	4.68	4.42	0
Small ($p = 0.10$)	7.90	4.87	4.44	0.49
Small ($p = 0.15$)	8.30	4.90	4.43	0.91
Medium ($p = 0.05$)	17.20	5.14	4.59	9.81
Medium ($p = 0.10$)	17.20	5.14	4.59	9.81
Medium ($p = 0.15$)	17.25	5.17	4.56	9.86
Large ($p = 0.05$)	17.20	5.14	4.59	9.81
Large ($p = 0.10$)	17.20	5.14	4.58	9.81
Large ($p = 0.15$)	17.20	5.14	4.59	9.81

Table 1: Forecast performance of deterministic and MC Dropout-based ensemble variants of the Aurora model, evaluated against ERA5. Metrics include RMSE for 2m temperature (*2t*), 10m eastward (*10u*) and southward (*10v*) wind components. Delta (Δ) denotes the mean *2t* RMSE relative to the best performing configuration

Anomaly Correlation Coefficient

ACC results are summarized in Fig. 4 (definition in 6.3). For m=3, ACC remains low across all leads and variables, indicating weak large-scale anomaly alignment for this configuration and evaluation window. We therefore do not observe a consistent ACC benefit from MC-Dropout at $p \in \{0.05, 0.10, 0.15\}$.

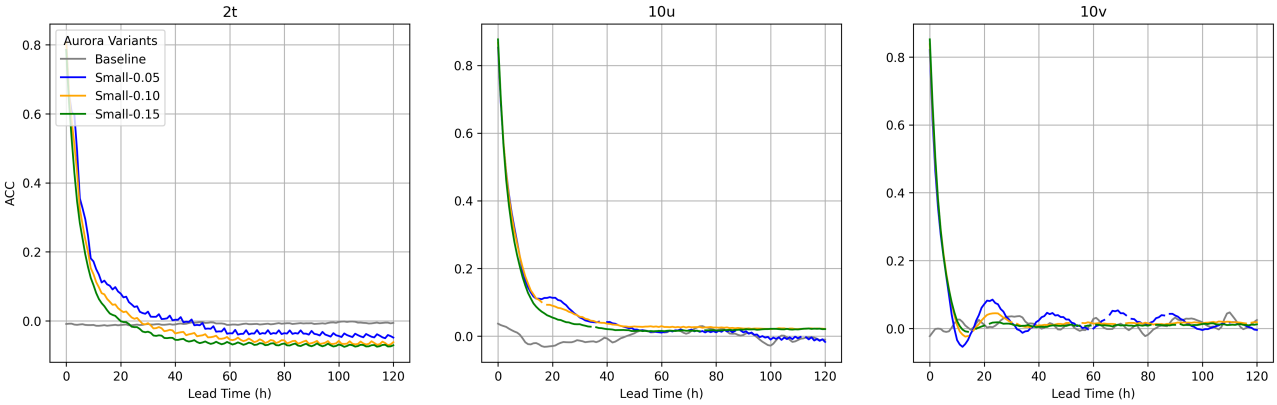

Figure 4: Anomaly Correlation Coefficient (ACC) over lead time for Aurora forecasts with stochastic dropout. Each subplot shows ACC for a different surface variable: 2-meter temperature (2t), 10-meter zonal wind (10u), and 10-meter meridional wind (10v). The negative or near-zero ACC values reflect poor correlation with climatological anomalies across almost all lead times, highlighting the degradation of large-scale patterns.

	ACC (2t)	ACC (10u)	ACC (10v)	Δ_{2t}
Aurora (deterministic)	-0.008	-0.001	0.0107	-0.11
Small ($p = 0.05$)	0.03	0.06	0.04	0
Small ($p = 0.10$)	0.002	0.07	0.04	0.01
Small ($p = 0.15$)	-0.02	0.06	0.04	-0.05
Medium ($p = 0.05$)	-0.10	0.03	0.02	-0.13
Medium ($p = 0.10$)	-0.10	0.03	0.02	-0.13
Medium ($p = 0.15$)	-0.10	0.03	0.02	-0.13
Large ($p = 0.05$)	-0.10	0.03	0.02	-0.13
Large ($p = 0.10$)	-0.10	0.03	0.02	-0.13
Large ($p = 0.15$)	-0.10	0.03	0.02	-0.13

Table 2: Forecast performance of Baseline and MC Dropout-based ensemble variants of the Aurora model, evaluated against ERA5. Metrics include ACC for 2m temperature (2t), 10m zonal (10u) and meridional (10v) wind components. Delta (Δ) denotes the mean ACC improvement relative to the best performing configuration

Spread-Skill Ratio (SSR)

In addition to point accuracy, we assess *calibration* using the spread–skill ratio (SSR). Values near 1 indicate well-calibrated uncertainty; values < 1 (>1) indicate under- (over-) dispersion. Across variables and dropout settings, SSR remains well below 1, meaning forecasts are over-confident, though higher dropout modestly increases spread and moves SSR upward. We therefore report SSR–vs–lead curves alongside a summary table of mean SSR (and $|\text{SSR} - 1|$) to quantify both calibration level and lead-time dependence.

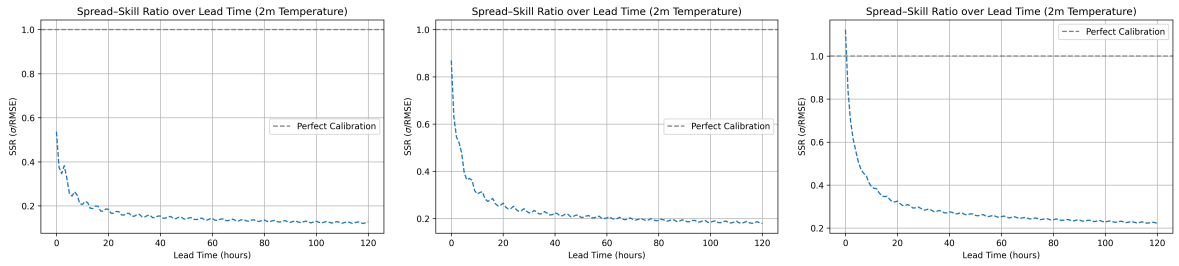

Figure 5: Spread-Skill Ratio (SSR) over lead time for Aurora forecasts with dropout-based ensembling. The plotted line indicates perfect calibration level. The steep initial slump reflects rapid error accumulation in early autoregressive steps, followed by a plateau in longer-range forecasts.

	SSR (2t)	SSR (10u)	SSR (10v)	Δ_{2t}
Small ($p = 0.05$)	0.233	0.142	0.138	0.766
Small ($p = 0.10$)	0.325	0.192	0.191	0.676
Small ($p = 0.15$)	0.392	0.230	0.229	0.614
Medium ($p = 0.05$)	0.039	0.116	0.136	0.961
Medium ($p = 0.10$)	0.058	0.150	0.175	0.945
Medium ($p = 0.15$)	0.058	0.150	0.175	0.945
Large ($p = 0.05$)	0.039	0.116	0.136	0.961
Large ($p = 0.10$)	0.040	0.119	0.139	0.960
Large ($p = 0.15$)	0.039	0.116	0.136	0.961

Table 3: Forecast performance of deterministic and MC Dropout-based ensemble variants of the Aurora model, evaluated against ERA5. Metrics include SSR for 2m temperature (2t), 10m zonal (10u) and meridional (10v) wind components. Delta (Δ) denotes the *2t* absolute error from ideal SSR-ratio (1.0)

8 Discussion

Main takeaways relative to the research questions. This thesis tested whether a wide-adopted ensemble method (MC-Dropout) can be retrofitted onto a modern MLWP simulator to yield useful probabilistic guidance without retraining. In short: *no*, not for deterministic accuracy. Small ensembles with modest dropout rates improved uncertainty coverage relative to a deterministic baseline, especially at longer lead times—but they also imposed a clear accuracy cost: an early RMSE jump at rollout start persisted, and ACC remained weak. Thus, while uncertainty can be obtained cheaply, accuracy gains are not automatic with this approach.

A practical contribution of this work is a clean, reproducible retrofit pipeline (settings, rollout logic, evaluation hooks) that can serve as a starting point for further exploration.

These findings align with prior experience using MC-Dropout as a scalable Bayesian approximation: it often provides useful calibration signals with minimal engineering overhead, but it rarely improves point forecasts compared with stronger (and costlier) methods such as Deep Ensembles (Lakshminarayanan et al., 2017).

Aurora-specific interpretation. The gap is unsurprising for two reasons. First, there is a *regularization mismatch*: inference-time activation dropout does not match the model's training regularization (stochastic depth), so the injected noise degrades large-scale coherence. In short, this off-policy, low-touch retrofit delivers uncertainty, but at a notable accuracy cost.

Limitations and caveats.

- *Regularization mismatch:* inference-time activation dropout vs. training-time stochastic depth; the Bayesian proxy is approximate and likely injects small-scale noise that harms predictive skill.
- *Short evaluation window:* MLWP models are typically benchmarked over months or a full year across multiple initialization times to capture seasonality and regime diversity; this thesis used a limited window, so results should be interpreted cautiously.
- *Regridding:* necessary interpolation between native output and the evaluation grid can nudge global scores.

Next steps. Having established that MC-Dropout does not behave as simply here as in MLPs or other simpler architectures, the immediate follow-up is to *match the stochastic mechanism to training*: implement a DropPath-based Monte Carlo variant and re-evaluate on the same metrics. In parallel, strengthen uncertainty assessment with basic reliability diagnostics (e.g., rank histograms, CRPS) and extend verification to a longer, mixed-season period.

9 Conclusion

This thesis set out to test a simple question: can a time-tested framework for homogeneous model ensembling such as MC-Dropout, cleanly be retrofitted onto a modern encoder-processor-decoder MLWP model? The answer is pragmatic rather than spectacular. We showed that it is straightforward to make a deterministic foundation model uncertainty, aware at low operational cost and in a reproducible way. While this did not translate into improved forecast accuracy in our setup, the work delivers two enduring outcomes: (1) a reusable pipeline for probabilistic forecasting, and (2) evidence about where this approach helps and where it falls short.

Within the scope of a single-investigator project, that is a useful contribution, since it turns a widely adopted idea into a concrete baseline for MLWP uncertainty estimations. The path forward is also clear, align the stochastic mechanism with training and relax rollout coupling, so the next iteration can focus on unlocking accuracy gains.

Acknowledgements

Firstly, I would like to express my gratitude to my supervisor, Andres Masegosa, for his support and guidance throughout the course of this project. His expertise, critical feedback, and encouragement have been instrumental to the development and completion of this thesis.

Use of Artificial Intelligence: This thesis has benefited from the use of generative artificial intelligence tools, including Grammarly, NotebookLM, and ChatGPT, which were employed to refine grammar, enhance clarity, and suggest improvements to language expression. GitHub Copilot was also used to assist with code generation and software development tasks. All usage was conducted in accordance with ethical and academic guidelines.

References

P. Bauer, A. Thorpe, and G. Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567): 47–55, Sept. 2015. ISSN 1476-4687. doi: 10.1038/nature14956. URL https://www.nature.com/articles/nature14956. Publisher: Nature Publishing Group.

C. Bodnar, W. P. Bruinsma, A. Lucic, M. Stanley, A. Vaughan, J. Brandstetter, P. Garvan, M. Riechert, J. A. Weyn, H. Dong, J. K. Gupta, K. Thambiratnam, A. T. Archibald, C.-C. Wu, E. Heider, M. Welling, R. E. Turner, and P. Perdikaris. A Foundation Model for the Earth System, Nov. 2024. URL http://arxiv.org/abs/2405.13063. arXiv:2405.13063 [physics].

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug. 1996. ISSN 1573-0565. doi: 10.1007/BF00058655. URL https://doi.org/10.1007/BF00058655.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct. 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

C. Diaconu, A. Shysheya, and A. McDonald. talks.cam : ML for Medium-Range Weather Forecasting, Dec. 2025. URL https://talks.cam.ac.uk/talk/index/229309.

ECMWF. Medium-range forecasts, May 2022. URL https://www.ecmwf.int/en/forecasts/documentation-and-support/medium-range-forecasts.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, Oct. 2016. URL http://arxiv.org/abs/1506.02142. arXiv:1506.02142 [stat].

A. Github. Form of a Batch — Aurora: A Foundation Model of the Atmosphere. URL https://microsoft.github.io/aurora/batch.html.

T. Gneiting and A. E. Raftery. Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477):359–378, Mar. 2007. ISSN 0162-1459, 1537-274X. doi: 10.1198/016214506000001437. URL http://www.tandfonline.com/doi/abs/10.1198/016214506000001437.

G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger. Deep Networks with Stochastic Depth, July 2016. URL http://arxiv.org/abs/1603.09382. arXiv:1603.09382 [cs].

Y. Huang, X. Feng, B. Li, Y. Xiang, H. Wang, B. Qin, and T. Liu. Ensemble Learning for Heterogeneous Large Language Models with Deep Parallel Collaboration, May 2024. URL http://arxiv.org/abs/2404.12715. arXiv:2404.12715 [cs].

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, Nov. 2017. URL http://arxiv.org/abs/1612.01474. arXiv:1612.01474 [stat].

R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, and P. Battaglia. GraphCast: Learning skillful medium-range global weather forecasting, Aug. 2023. URL http://arxiv.org/abs/2212.12794. arXiv:2212.12794 [cs].

M. Leutbecher and T. N. Palmer. Ensemble forecasting. *Journal of Computational Physics*, 227(7):3515–3539, Mar. 2008. ISSN 0021-9991. doi: 10.1016/j.jcp.2007.02.014. URL https://www.sciencedirect.com/science/article/pii/S0021999107000812.

E. N. Lorenz. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, Mar. 1963. ISSN 0022-4928, 1520-0469. doi: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2. URL https://journals.ametsoc.org/view/journals/atsc/20/2/1520-0469_1963_020_0130_dnf_2_0_co_2.xml. Publisher: American Meteorological Society Section: Journal of the Atmospheric Sciences.

J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, P. Hassanzadeh, K. Kashinath, and A. Anandkumar. FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators, Feb. 2022. URL http://arxiv.org/abs/2202.11214. arXiv:2202.11214 [physics].

I. Price, A. Sanchez-Gonzalez, F. Alet, T. R. Andersson, A. El-Kadi, D. Masters, T. Ewalds, J. Stott, S. Mohamed, P. Battaglia, R. Lam, and M. Willson. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, Jan. 2025. ISSN 1476-4687. doi: 10.1038/s41586-024-08252-9. URL https://www.nature.com/articles/s41586-024-08252-9. Publisher: Nature Publishing Group.

S. Rasp, S. Hoyer, A. Merose, I. Langmore, P. Battaglia, T. Russel, A. Sanchez-Gonzalez, V. Yang, R. Carver, S. Agrawal, M. Chantry, Z. B. Bouallegue, P. Dueben, C. Bromberg, J. Sisk, L. Barrington, A. Bell, and F. Sha. WeatherBench 2: A benchmark for the next generation of data-driven global weather models, Jan. 2024. URL http://arxiv.org/abs/2308.15560. arXiv:2308.15560 [physics].

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Y. Yao, H. Wu, M. Liu, S. Luo, X. Han, J. Liu, Z. Guo, and L. Song. Determine-Then-Ensemble: Necessity of Top-k Union for Large Language Model Ensembling, Feb. 2025. URL http://arxiv.org/abs/2410.03777. arXiv:2410.03777 [cs].

Y.-C. Yu, C.-C. Kuo, Z. Ye, Y.-C. Chang, and Y.-S. Li. Breaking the Ceiling of the LLM Community by Treating Token Generation as a Classification for Ensembling, Sept. 2024. URL http://arxiv.org/abs/2406.12585. arXiv:2406.12585 [cs].

A Aurora Batch Structure

Category	Name	Description	Units	Tensor shape
Surface-level variables		(history length t=2: index 0 = previous, index 1 = current)		
Surface	`2t`	Two-meter temperature	K	(b, t, h, w)
Surface	`10u`	Ten-meter eastward wind speed	m s^{-1}	(b, t, h, w)
Surface	`10v`	Ten-meter southward wind speed	m s^{-1}	(b, t, h, w)
Surface	`msl`	Mean sea-level pressure	Pa	(b, t, h, w)
Static variables				
Static	`lsm`	Land–sea mask	—	(h, w)
Static	`slt`	Soil type	—	(h, w)
Static	`z`	Surface geopotential	m^2 s^{-2}	(h, w)
Atmospheric variables		(all share the same pressure levels and order)		
Atmos	`t`	Temperature	K	(b, t, c, h, w)
Atmos	`u`	Eastward wind speed	m s^{-1}	(b, t, c, h, w)
Atmos	`v`	Southward wind speed	m s^{-1}	(b, t, c, h, w)
Atmos	`q`	Specific humidity	kg kg^{-1}	(b, t, c, h, w)
Atmos	`z`	Geopotential	m^2 s^{-2}	(b, t, c, h, w)

Table 4: Variables accepted by `aurora.Batch`. Shapes use b: batch size, t: history length (= 2), c: number of pressure levels, h: latitudes, w: longitudes. All inputs are unnormalised; the model applies internal normalisation.

Field	Type / Shape	Constraints / Notes
`lat`	vector (h) or matrix (h,w)	Strictly decreasing in latitude; for curvilinear grids (matrix), each *column* must be decreasing.
`lon`	vector (w) or matrix (h,w)	In $[0, 360)$ and strictly increasing; for curvilinear grids (matrix), each *row* must be increasing.
`atmos_levels`	tuple of length c (hPa)	Order must exactly match the third dimension of `atmos_vars`; must be a *tuple*, not a list.
`time`	tuple of datetimes of length b	Time of the *current* step; e.g., (datetime$(2024, 1, 1, 12{:}00),$) for b=1.

Table 5: Required fields of `aurora.Metadata` used by `aurora.Batch`. Extracted from: Github

B Software & Infrastructure

Runtime stack

Component	Version / Notes
PyTorch	`2.6.0`
Ray	`2.44.1`
WeatherBench2	`d47c644b1284a37cb668fb7ff90c963c74f29164`
xarray / zarr	`2025.4.0` / `2.18.3`
Python	3.10 (`singularity` image `containers/python-torch-3.10.sif`)
GCS client	`gcsfs 2025.3.0` with service-account auth

Distributed inference (Ray)

`num_gpus=>8`,

`cpus-per-task=12`;

`memory=[48/96/256gb]`;

SLURM scripts are in `stormstack/scripts`.

Storage

Google Cloud Storage bucket `gs://stormstack/aurora`.

Approx artifact size per experiment: `36.3GB`.

Resource profile

Per forward pass peak memory: `26 GB` on `L40s`;

Typical wall-clock per 1h forecast: `02:30`

C Reproducibility & Availability

Code & Config

Repository: `bodeby/stormstack` at commit `d163aac`.

Jobs run on AAU AI Cloud via SLURM using a Singularity image built by `make-container-3.10.sh` (produces `containers/python-torch-3.10.sif`, Python 3.10).

Analysis runs on a Strato VM (Ubuntu 22.04, JupyterLab over SSH).

Environment

CUDA (version as provided by NVIDIA L40S cluster runtime),

NVIDIA driver (cluster default, R535 series),

PyTorch 2.6, Python 3.10.

Data

We evaluate on WeatherBench2 using ERA5.

Obersvation Dataset: gs://weatherbench2/datasets/era5/1959-2023_01_10-wb13-6h-1440x721.zarr

Climatology Dataset: gs://weatherbench2/datasets/era5-hourly-climatology/1990-2019_6h_1440x721.zarr

Forecast Dataset: gs://stormstack/aurora

Evaluation Dataset: gs://stormstack/aurora_benchmarks

Inference

Hardware: `L40S/A100/A40` on AAU AI Cloud.

MC-Dropout: rate $p \in= \{0.05, 0.10, 0.15\}$;

stochastic members $m \in \{3, 5, 8\}$

Analysis (VM/Jupyter)

Start JupyterLab on the Strato VM and connect via SSH tunnel to port 8888.

Provided notebooks regenerate all figures/tables from saved predictions and metrics.

Aggregator Class

```python
@dataclass
class Aggregator:
    """Utility class for aggregating predictions from multiple Aurora model forward passes."""

    @staticmethod
    def aggregate_batches(batches: List[SharedBatch], agg_fn: Callable) -> Tuple[dict, dict]:
        surf_keys = batches[0].surf_vars.keys()
        atmos_keys = batches[0].atmos_vars.keys()

        if len(batches) == 1:
            surf = {k: batches[0].surf_vars[k] for k in surf_keys}
            atmos = {k: batches[0].atmos_vars[k] for k in atmos_keys}
        else:
            stacked_surf = {
                k: torch.stack([b.surf_vars[k].squeeze(0) for b in batches], dim=0)
                for k in surf_keys
            }
            stacked_atmos = {
                k: torch.stack([b.atmos_vars[k].squeeze(0) for b in batches], dim=0)
                for k in atmos_keys
            }

            surf = {k: agg_fn(v, dim=0).unsqueeze(0) for k, v in stacked_surf.items()}
            atmos = {k: agg_fn(v, dim=0).unsqueeze(0) for k, v in stacked_atmos.items()}

        return surf, atmos

    @staticmethod
    def aggregate(predictions: List[SharedBatch]) -> Tuple[SharedBatch, dict, dict]:
        """
        Aggregates a list of predictions by computing mean, std, var, and selected quantiles.
        """

        mean_surf, mean_atmos = Aggregator.aggregate_batches(predictions, torch.mean)
        std_surf, _ = Aggregator.aggregate_batches(predictions, torch.std)
        var_surf, _ = Aggregator.aggregate_batches(
            predictions, lambda x, dim=0: torch.var(x, dim=dim, unbiased=False)
        )

        mean_batch = replace(predictions[0], surf_vars=mean_surf, atmos_vars=mean_atmos)
        return mean_batch, std_surf, var_surf
```

D Uncertainty Evolution Across Variables and Dropout Settings

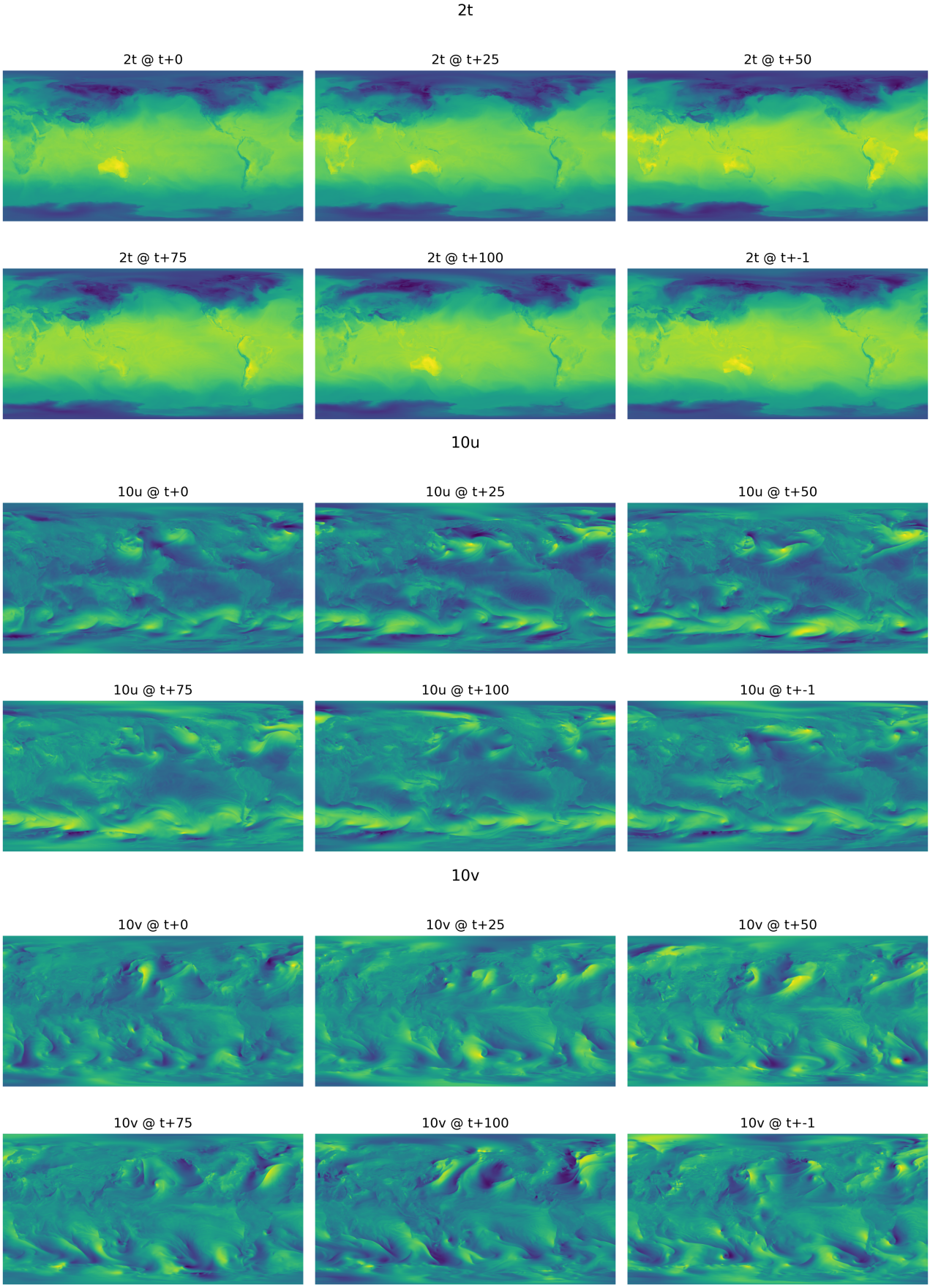

Figure 6: Baseline:

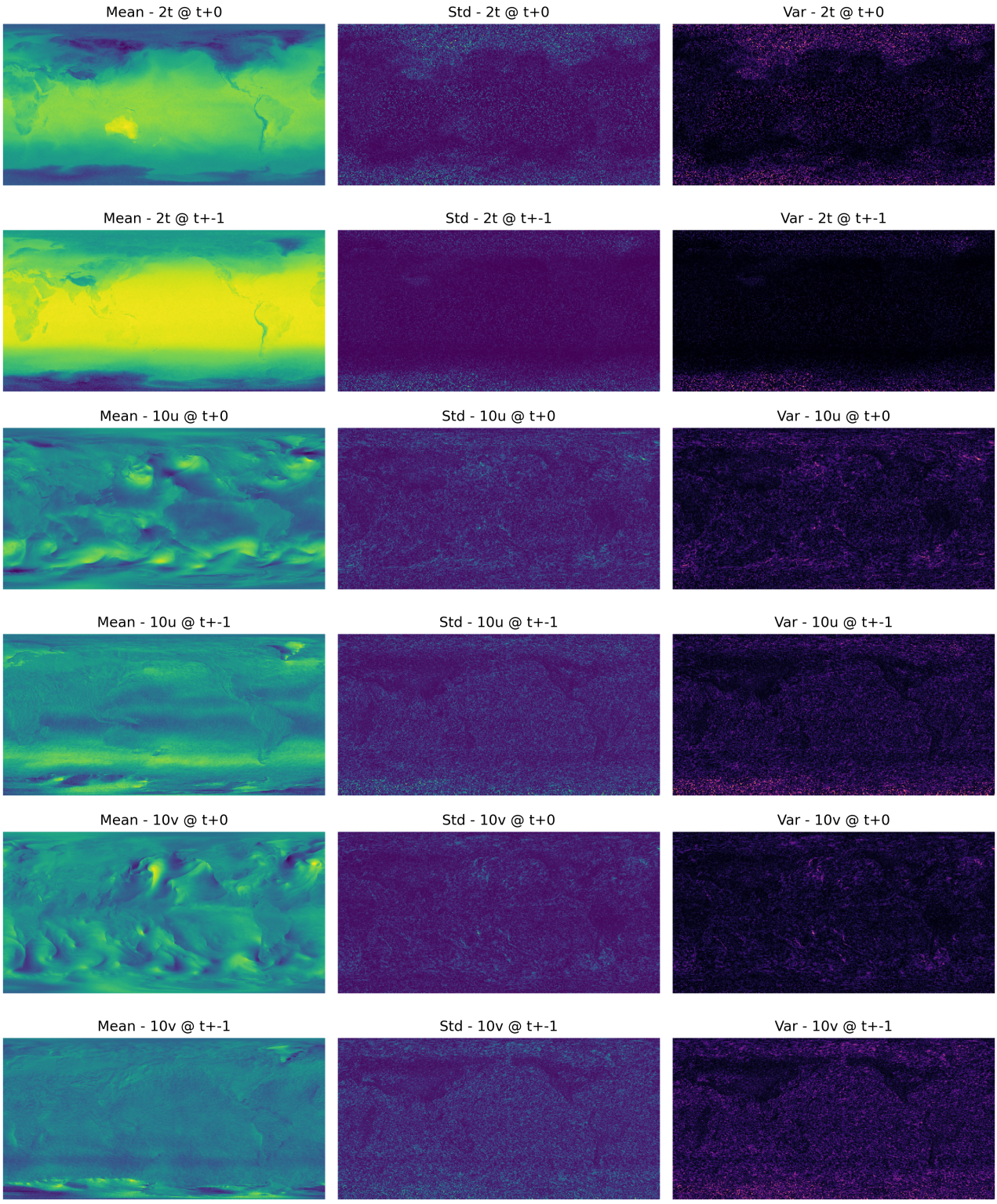

Figure 7: SMALL ($m = 3, p = 0.05$):

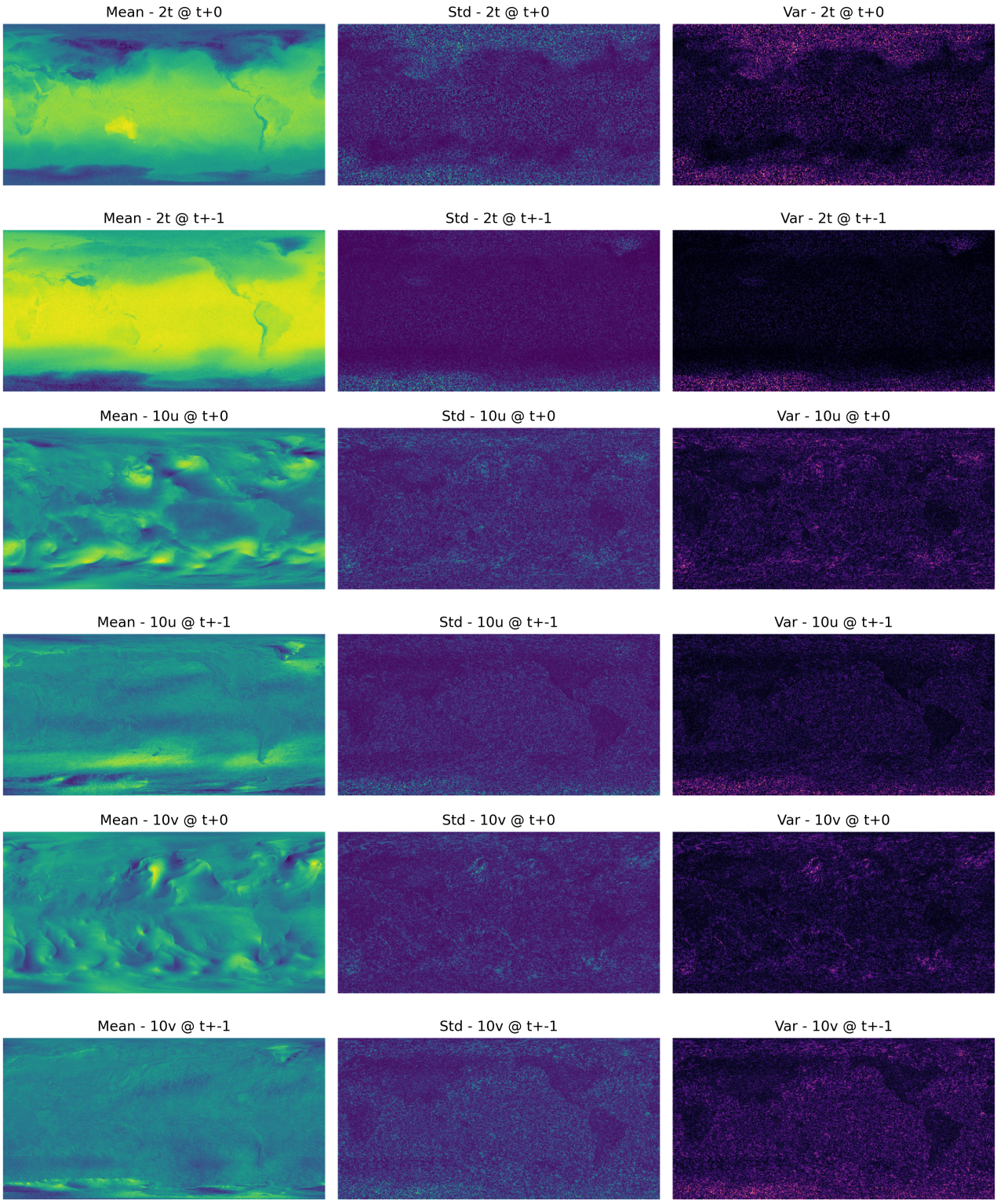

Figure 8: SMALL ($m = 3, p = 0.10$):

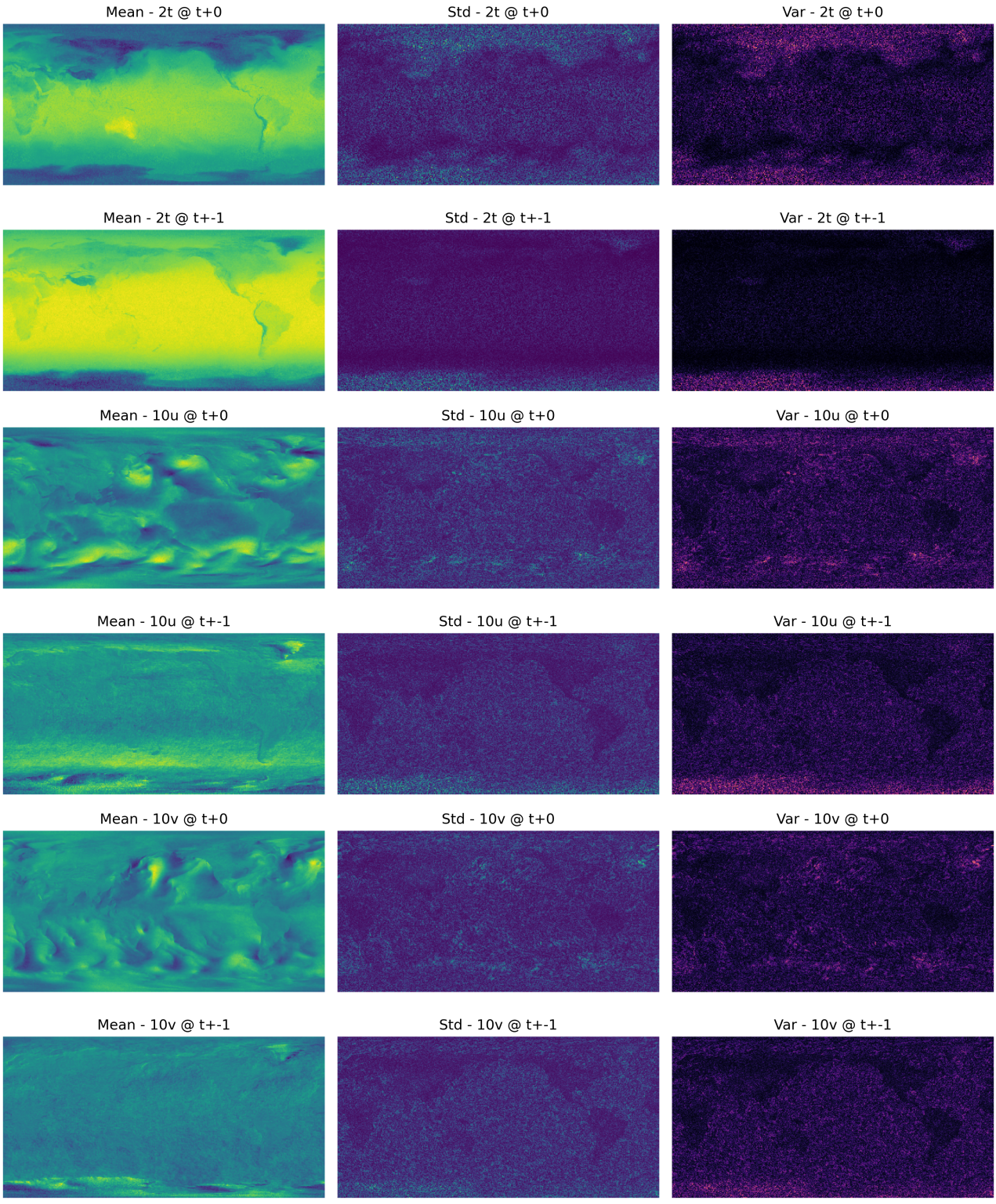

Figure 9: SMALL ($m = 3, p = 0.15$):

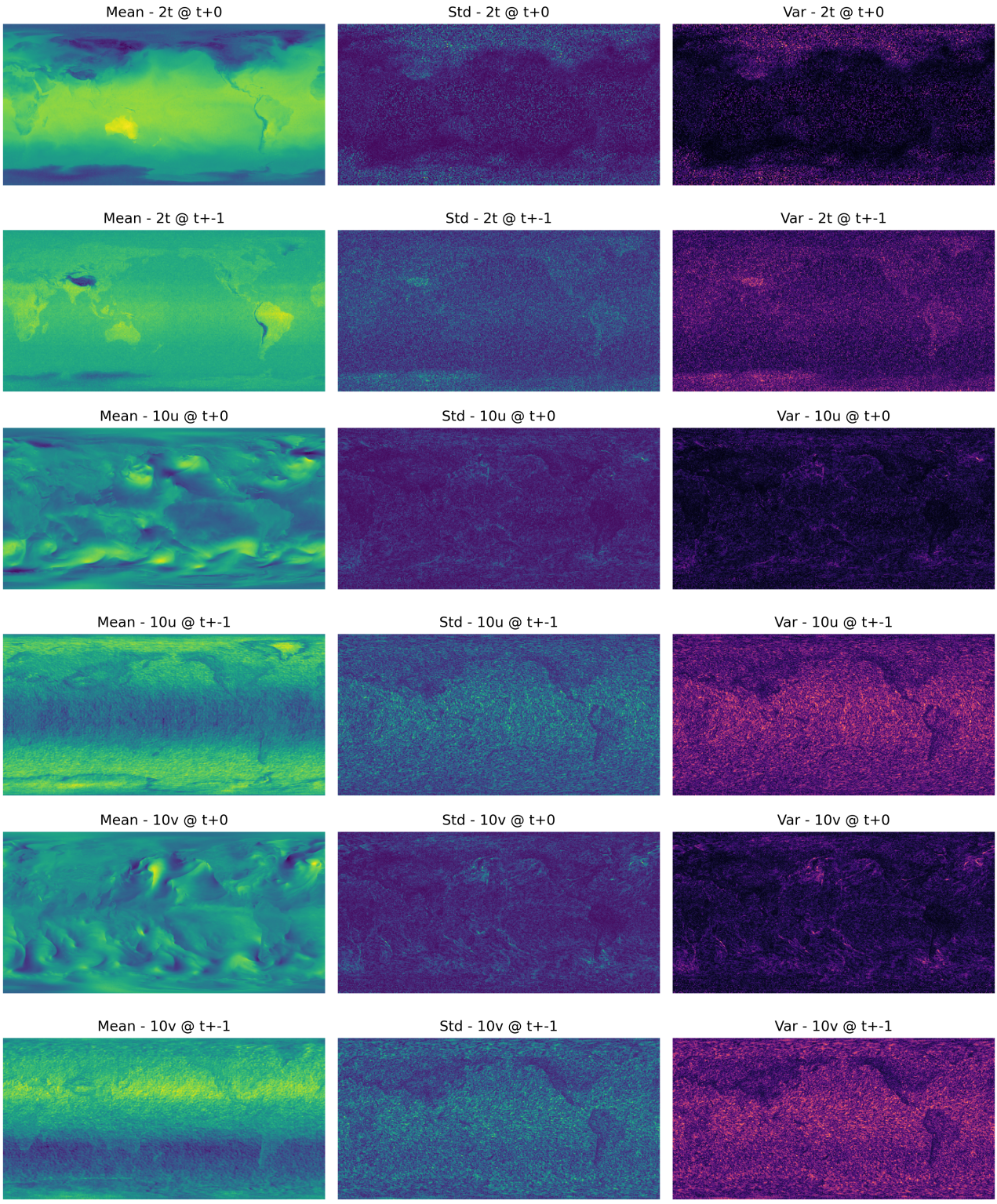

Figure 10: MEDIUM ($m = 5, p = 0.05$):

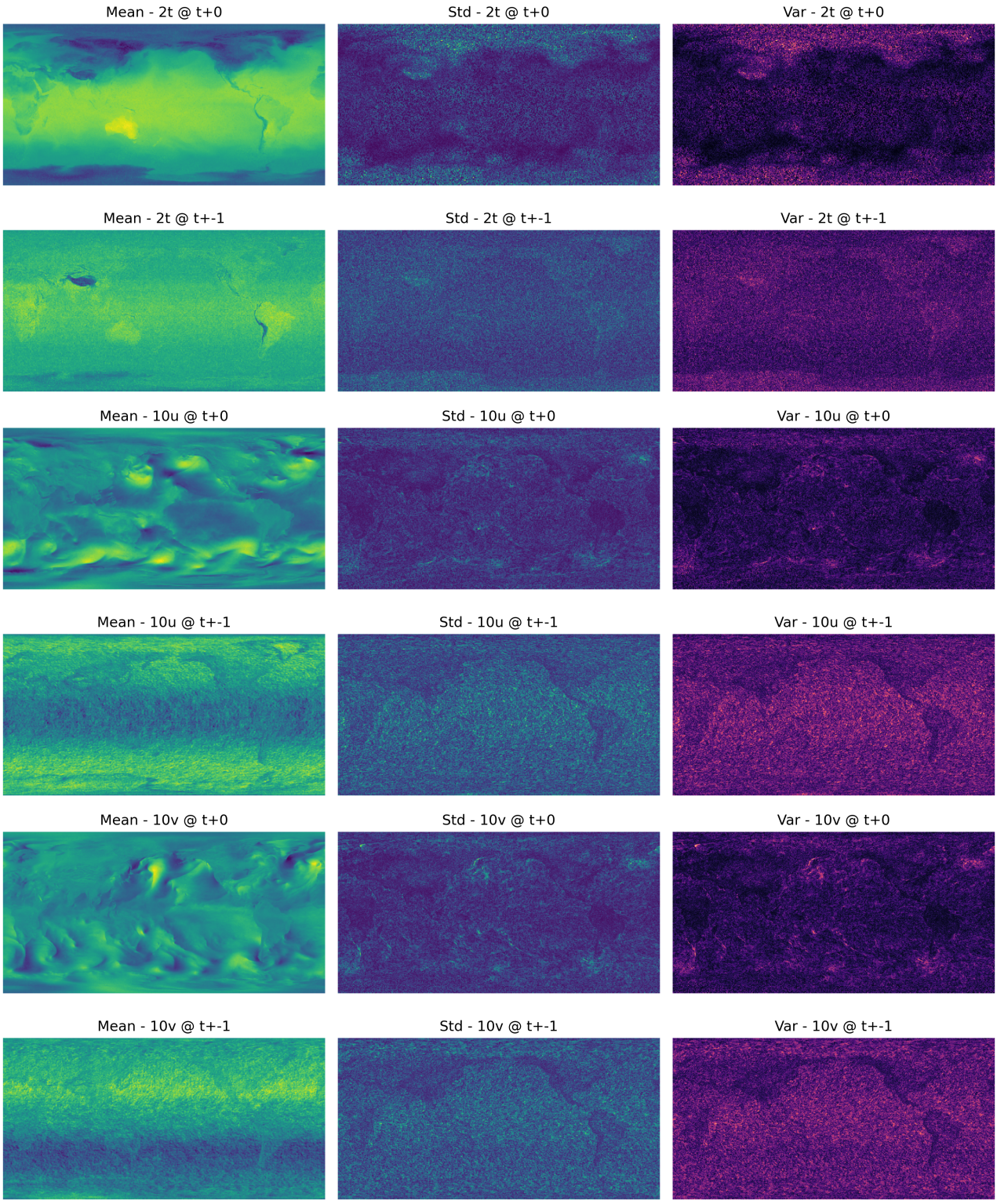

Figure 11: MEDIUM ($m = 5, p = 0.10$):

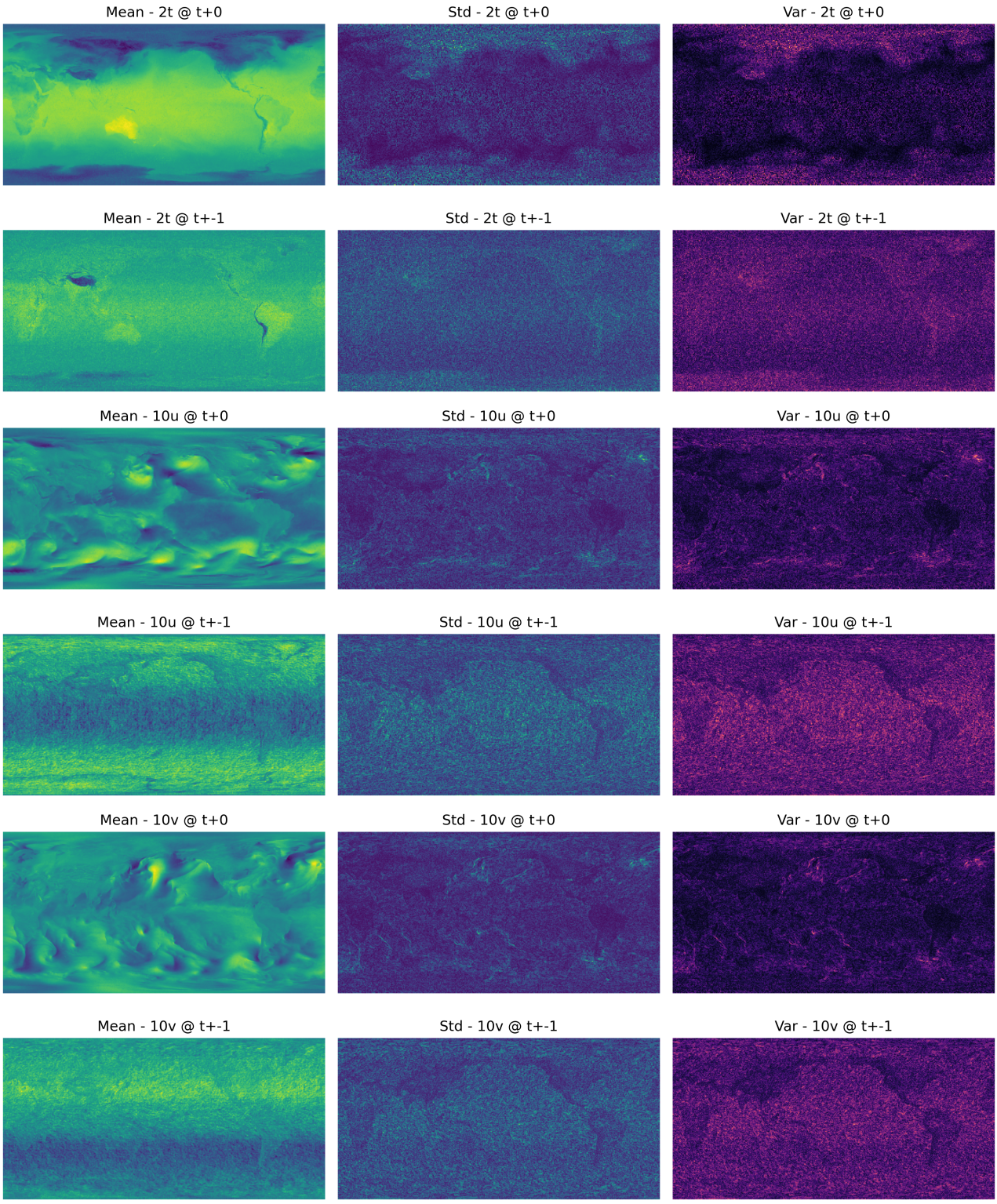

Figure 12: MEDIUM ($m = 5, p = 0.15$):

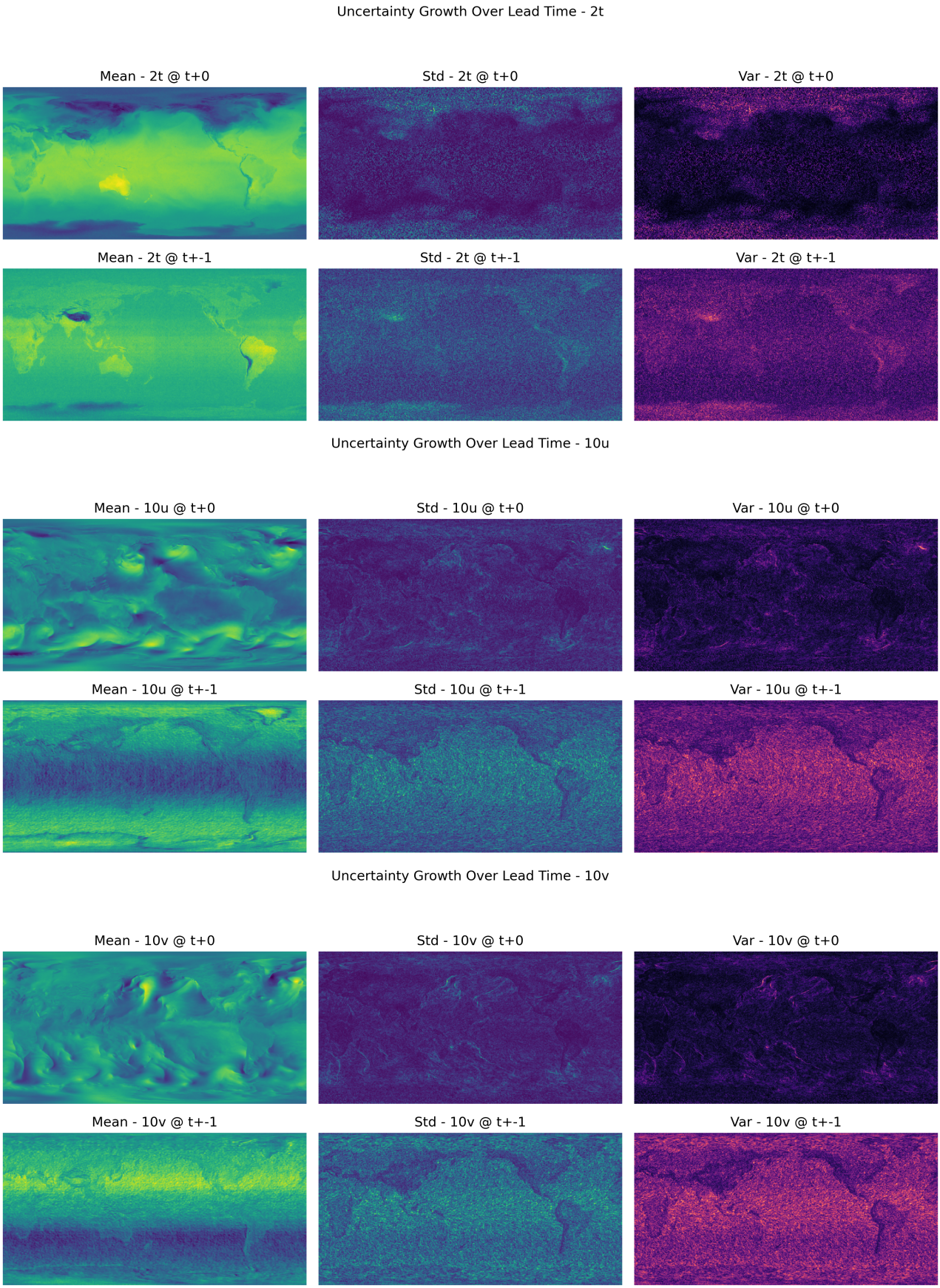

Figure 13: LARGE ($m = 8, p = 0.05$)

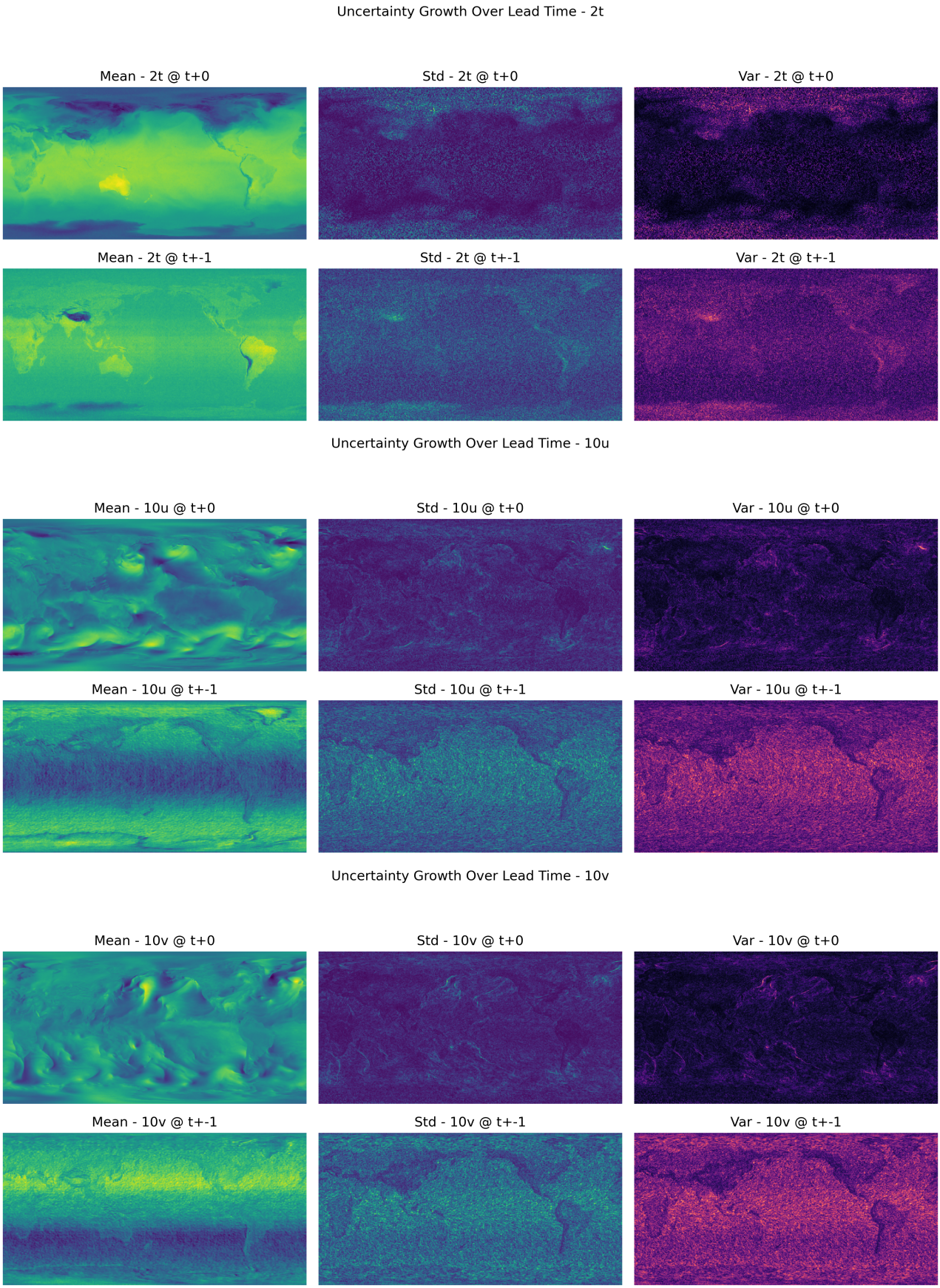

Figure 14: LARGE ($m = 8, p = 0.10$)

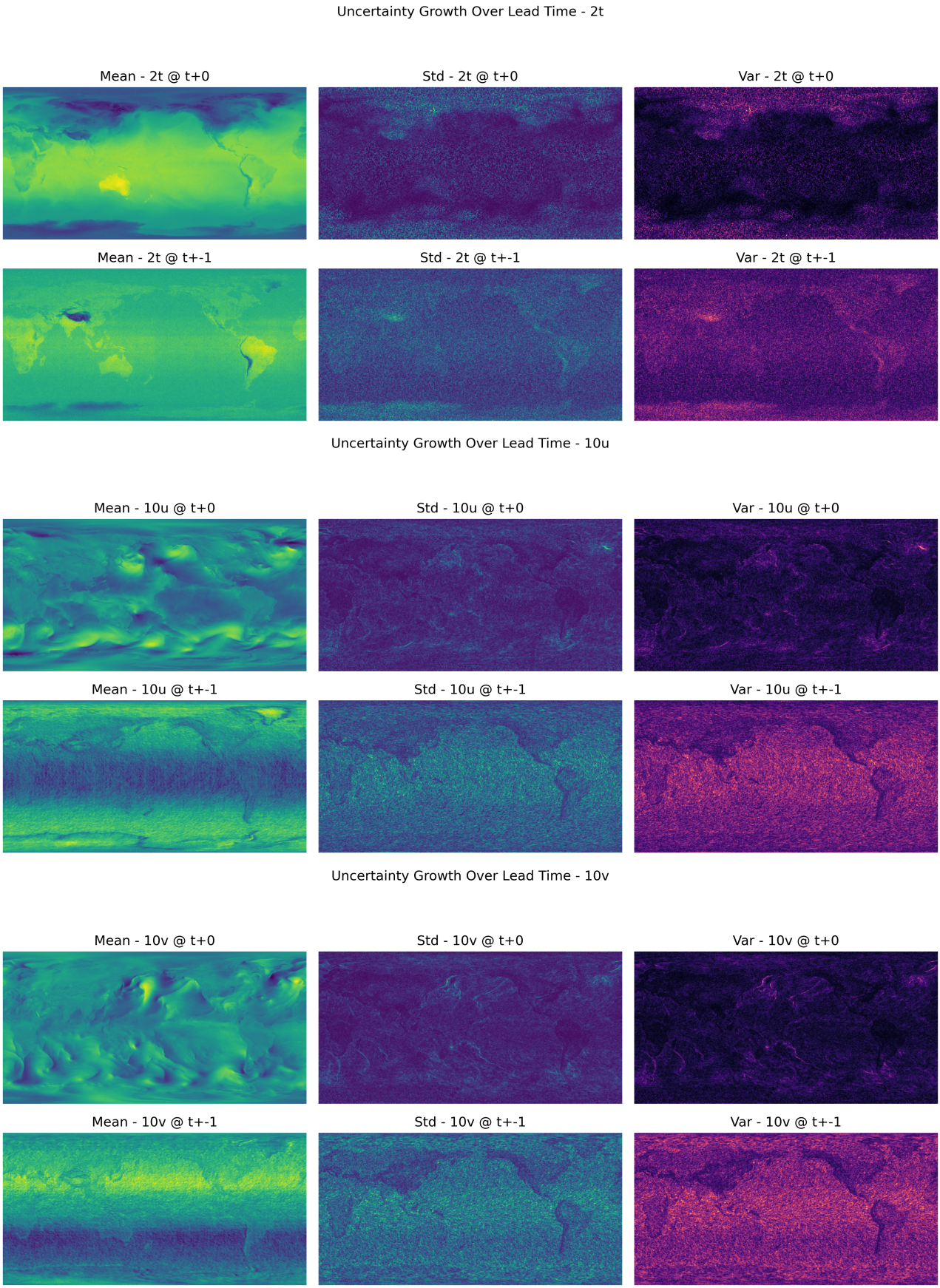

Figure 15: LARGE ($m = 8, p = 0.15$)