OmniCast: A Masked Latent Diffusion Model for Weather Forecasting Across Time Scales

Tung Nguyen[1] **Tuan Pham**[2] **Troy Arcomano**[3,4] **Veerabhadra Kotamarthi**[3]
Ian Foster[3] **Sandeep Madireddy**[3] **Aditya Grover**[1]
[1]UCLA [2]UCI [3]Argonne National Laboratory [4]Allen Institute for AI

Abstract

Accurate weather forecasting across time scales is critical for anticipating and mitigating the impacts of climate change. Recent data-driven methods based on deep learning have achieved significant success in the medium range, but struggle at longer subseasonal-to-seasonal (S2S) horizons due to error accumulation in their autoregressive approach. In this work, we propose OmniCast, a scalable and skillful probabilistic model that unifies weather forecasting across timescales. OmniCast consists of two components, a VAE model that encodes raw weather data into a continuous, lower-dimensional latent space, and a diffusion-based transformer model that generates a sequence of future latent tokens given the initial conditioning tokens. During training, we mask random future tokens and train the transformer to estimate their distribution given conditioning and visible tokens using a per-token diffusion head. During inference, the transformer generates the full sequence of future tokens by iteratively unmasking random subsets of tokens. This joint sampling across space and time mitigates compounding errors from autoregressive approaches. The low-dimensional latent space enables modeling long sequences of future latent states, allowing the transformer to learn weather dynamics beyond initial conditions. OmniCast performs competitively with leading probabilistic methods at the medium-range timescale while being $10\times$ to $20\times$ faster, and achieves state-of-the-art performance at the subseasonal-to-seasonal scale across accuracy, physics-based, and probabilistic metrics. Furthermore, we demonstrate that OmniCast can generate stable rollouts up to 100 years ahead. Code and model checkpoints are available at https://github.com/tung-nd/omnicast.

1 Introduction

Accurate weather forecasting across time scales is essential for anticipating extreme events, managing resources, and mitigating the impacts of climate change. While medium-range forecasting, which encompasses predictions up to approximately two weeks, has seen remarkable progress with both numerical and data-driven approaches, extending prediction skill beyond this horizon remains a significant challenge. Subseasonal-to-seasonal (S2S) forecasting, which aims to predict atmospheric conditions from two to six weeks ahead, represents this next frontier. This timescale bridges the gap between short-term weather forecasts and longer-term climate projections, enabling more informed decision-making for extreme weather events such as droughts, floods, and heatwaves [51, 36, 52, 8]. S2S prediction is particularly challenging due to the interplay between atmospheric initial conditions, essential for short-term and medium-range forecasting, and boundary conditions dominating seasonal and climate predictions [27, 28]. Traditional numerical weather prediction (NWP) models, built upon solving differential equations of fluid dynamics and thermodynamics, have been instrumental in advancing S2S weather prediction [36, 47, 48]. However, numerical methods incur substantial computational costs due to the complexity of integrating large systems of differential equations,

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

particularly at fine spatial and temporal resolutions. This computational bottleneck also constrains the ensemble size of ensemble systems, which is crucial for achieving accurate S2S predictions.

To overcome the challenges of NWP systems, there has been a growing interest in data-driven approaches based on deep learning for weather forecasting [10, 43, 50]. These approaches involve training deep neural networks on historical datasets, such as ERA5 [14, 15, 39, 40], to learn the underlying weather patterns. Once trained, they can produce forecasts in seconds compared to the hours required by NWP models. Recent deep learning methods such as PanguWeather [2], Graphcast [22], and Stormer [33] have also shown superior accuracy in medium-range weather forecasting, surpassing operational IFS [49], the state-of-the-art NWP system. However, their application to the S2S timescale has been limited [31]. One possible explanation for this limitation is the rapid error compounding in their autoregressive designs, in which a model learns to forecast the future weather state at a small interval and iteratively feeds its prediction back as input to achieve longer-horizon forecasts. Even though previous works have proposed multi-step finetuning to mitigate this issue, back-propagation through a large number of forward passes required for S2S timescales is computationally prohibitive. Moreover, training a neural network to forecast at a small interval only allows the model to learn the initial conditions problem, ignoring boundary conditions that are critical for prediction at S2S timescales.

In this work, we propose OmniCast, a novel latent diffusion model for skillful probabilistic weather forecasting across time scales. OmniCast follows a two-stage training process. In the first stage, we train a VAE model [19] that compresses raw weather data into a continuous, lower-dimensional latent space. In the second stage, we train a transformer to model the distribution of a sequence of future latent tokens given the initial conditioning tokens using a masked generative framework [3, 55]. Specifically, during training, we randomly mask a subset of future tokens, and task the transformer to *unmask* these tokens based on the conditioning tokens and the visible tokens. Since the latent tokens lie in a continuous space, we use a small diffusion network on top of the transformer model to estimate the per-token distribution of unmasked tokens. In addition to the diffusion loss, we apply a mean-squared error (MSE) objective to enforce the model to accurately predict the first few latent frames deterministically. After training, OmniCast generates forecasts for the full sequence of future tokens through an iterative process. In each iteration, the model selects a subset of future tokens to unmask given the conditioning tokens and previously unmasked tokens, continuing this process until all future tokens are generated. The unmasking operation involves sampling from the diffusion model, with the number and positions of tokens selected to unmask in each iteration determined by a predefined schedule and unmasking order. This joint generation of future tokens across time and space significantly mitigates the compounding errors issue of an autoregressive approach. Furthermore, training on the full sequence of future frames enables OmniCast to address both initial condition problems and boundary condition challenges, which are critical for S2S prediction.

We evaluate OmniCast on ChaosBench [31], a recent benchmark for subseasonal-to-seasonal prediction. OmniCast achieves state-of-the-art performance on key atmospheric variables across various accuracy, physics-based, and probabilistic metrics. Additionally, we carefully study the impact of different design choices, including the auxiliary MSE loss, training sequence lengths, unmasking order, and diffusion sampling temperature, on the forecasting performance of OmniCast.

2 Related Work

Data-driven weather forecasting Deep learning has become a promising approach in the field of weather forecasting. Recent advancements with powerful architectures have achieved significant successes, providing faster inference and superior forecasting accuracy compared to IFS, the gold-standard numerical weather prediction system. Notable methods include FourCastNet [35], which utilizes an adaptive neural operator architecture; Keisler [17]'s, GraphCast [22], and AIFS [24], which leverage graph neural networks; and a series of transformer-based models such as PanguWeather [2], Stormer [33], and others [32, 6, 4, 7]. Beyond deterministic predictions, the field has increasingly focused on probabilistic forecasting to better account for forecast uncertainty. Common approaches involve integrating existing architectures with generative frameworks, including diffusion models [37, 30], normalizing flows [7], and latent variable models [34]. Others explore ensemble predictions through initial condition perturbations, exemplified by methods like AIFS-CRPS [24] and NeuralGCM [20].

Data-driven S2S prediction Recent benchmarks have emerged to evaluate data-driven methods at S2S timescales. While many focus on regional forecasts such as the US [16, 29], ChaosBench [31] offers a comprehensive framework for global S2S prediction, providing extensive numerical baselines and physics-based metrics. A key finding from ChaosBench shows that state-of-the-art deep learning methods struggle to extend to S2S timescales. These methods predominantly rely on autoregressive approaches that generate predictions iteratively at short time intervals, leading to error accumulation with increasing lead times. While multi-step finetuning helps mitigate this issue for medium-range forecasts, it becomes computationally prohibitive for S2S predictions due to the extensive number of required forward passes. Moreover, training models with short time intervals fails to capture boundary conditions essential for long-term weather patterns. While Fuxi-S2S [5] was proposed for S2S prediction, it focuses on forecasting daily averaged statistics, which fundamentally alters the underlying weather dynamics and makes it inapplicable to forecasting at instantaneous time steps.

3 Background and Preliminaries

3.1 Weather forecasting

The goal of weather forecasting is to forecast future weather conditions $X_T \in \mathbb{R}^{V \times H \times W}$ based on initial conditions $X_0 \in \mathbb{R}^{V \times H \times W}$, where T represents the target lead time, V denotes the number of input and output physical variables (e.g., temperature and geopotential), and $H \times W$ corresponds to the spatial resolution of the data, determined by the density of the global grid. In subseasonal-to-seasonal (S2S) forecasting, we focus on lead times ranging from 2 to 6 weeks. Autoregressive modeling is a dominant paradigm in data-driven weather forecasting, where a model iteratively produces forecasts $X_{\delta t}$ at a short interval δt to reach the target lead time T. In this work, we propose an alternative approach: training a generative model to estimate the distribution of the entire sequence of future weather states $X_{1:T}$ given initial conditions X_0. This approach mitigates error accumulation and enables the model to learn both initial and boundary condition dynamics by considering the complete sequence of weather states.

3.2 Masked generative modeling

Masked generative modeling is an efficient and powerful approach for image and video generation in computer vision [3, 55, 25]. In this framework, visual data $X_{1:T} \in \mathbb{R}^{T \times V \times H \times W}$ ($T = 1$ for images) is first embedded by a VAE encoder into a sequence of tokens $\mathbf{x} \in \mathbb{R}^{N \times D}$, where N represents the length of the flattened token sequence. During training, we apply a binary mask to randomly select a subset of tokens to be predicted, creating a corrupted sequence. We then train a transformer model to recover the original tokens at masked positions based on both the visible tokens and any additional conditioning information such as initial frames. For generation, the framework employs an iterative decoding process that starts with a fully masked sequence of future tokens. In each iteration, the model predicts a random subset of masked tokens in parallel, where the number and positions of the unmasked tokens follow a predefined schedule and order. This process continues until all tokens are unmasked, at which point the generated tokens are decoded back to the original domain through a VAE decoder. This framework offers key advantages for weather forecasting: it allows the model to capture long-range dependencies across the entire sequence while avoiding the error accumulation typical in autoregressive approaches, and the iterative refinement process enables the model to maintain consistency across both spatial and temporal dimensions.

3.3 Modeling continuous tokens with diffusion models

In the masked generative modeling framework, a common practice is to embed the raw visual data into a discrete latent space and train the transformer model using a cross-entropy objective. However, this approach relies on vector-quantized VAE models [45], which are sensitive to gradient approximation strategies [41, 38, 21] and typically achieve lower reconstruction quality than continuous-valued VAEs. Recent works [44, 26] have demonstrated that discretization can be eliminated by directly modeling the per-token probability distribution in a continuous latent space. In this work, we adopt diffusion models for continuous distribution modeling.

Given data $x \in \mathbb{R}^D$ and its conditioning information $z \in \mathbb{R}^D$, we model the conditional distribution $p(x \mid z)$ using a diffusion process that gradually transforms a Gaussian prior into the target distribution.

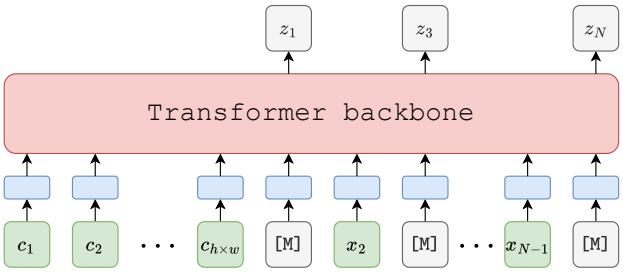


Figure 1: OmniCast processes the latent tokens through a transformer backbone that outputs a vector z_i for each position i in the sequence.

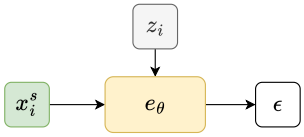


Figure 2: The denoising network e_θ predicts the noise ϵ from z_i and x_i^s.

Figure 3: The deterministic network predicts directly x_i from z_i.

The forward diffusion process progressively adds Gaussian noise to the data x following:

$$x_s = \sqrt{\alpha_s}x + \sqrt{1-\alpha_s}\epsilon, \tag{1}$$

where s indicates the diffusion step, α_s determines the noise schedule, and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ represents Gaussian noise. The reverse process employs a denoising network $\epsilon_\theta(x_s, s, z)$ parameterized by θ to predict the noise component from the noisy input x_s and condition z:

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{\epsilon,x}\left[\|\epsilon_\theta(x_s, s, z) - \epsilon\|^2\right]. \tag{2}$$

At inference time, conditional sampling begins with a random Gaussian noise $x_S \sim \mathcal{N}(0, \mathbf{I})$ and iteratively applies the reverse diffusion process:

$$x_{s-1} = \frac{1}{\sqrt{\alpha_s}}\left(x_s - \frac{1-\alpha_s}{\sqrt{1-\bar{\alpha}_s}}\epsilon_\theta(x_s, s, z)\right) + \tau\sigma_s\delta, \tag{3}$$

where $\bar{\alpha}_s = \prod_{k=1}^{s}\alpha_k$, $\delta \sim \mathcal{N}(0, \mathbf{I})$ and σ_s controls the magnitude of noise added at each step. This iterative process generates samples from the learned conditional distribution $p_\theta(x \mid z)$. Following [26], we additionally scale the noise $\sigma_s\delta$ by the temperature τ that controls the sample diversity from the diffusion model.

4 Methodology

We present OmniCast, a novel method for subseasonal-to-seasonal prediction. Similar to previous works in video generation, OmniCast consists of two components: a VAE model that compresses the raw weather data into a lower-dimensional space, and a masked generative transformer model in this latent space. We present the two components and their key design choices in this section.

4.1 VAE for weather data embedding

A VAE encoder embeds a weather state $X \in \mathbb{R}^{V\times H\times W}$ into a map of $h \times w$ latent tokens, where $h < H$ and $w < W$. In vector-quantized VAEs, each entry in the latent map is an integer index from a fixed-size vocabulary, representing a discrete latent space. While this discretization is widely adopted in computer vision due to its compatibility with cross-entropy training and straightforward sampling from softmax distributions, it presents significant challenges for weather data. Unlike RGB images with three channels, weather states can contain hundreds of physical variables, resulting in an extreme compression requirement. For instance, consider compressing weather data with 100 variables (32 bits per value) by a factor of 4 in each spatial dimension, using a vocabulary size of $2^{13} = 8192$ (13 bits per latent token). This results in a compression ratio of $(32\times100\times H\times W)/(13\times(H/4)\times(W/4)) \approx 3938$. Such aggressive compression leads to substantial reconstruction errors, ultimately degrading the performance of the second-stage generative modeling.

Therefore, we adopt a continuous VAE model for OmniCast, where each token in the $h\times w$ latent map is a continuous vector of D dimensions. With $D = 16$, for example, the compression ratio becomes $(32 \times 100 \times H \times W)/(32 \times 16 \times (H/4) \times (W/4)) = 100$, substantially lower than the discrete approach. While it is also possible to compress a sequence of weather states $X_{1:T} \in \mathbb{R}^{T\times V\times H\times W}$ in both temporal and spatial dimensions, our preliminary experiments showed no clear benefits from temporal compression, leading us to adopt per-frame embedding.

4.2 Masked generative modeling for S2S prediction

After training the VAE, we embed the initial condition into a sequence of tokens $\mathbf{c} = (c_1, c_2, \ldots, c_{h\times w})$. Similarly, each future weather state is embedded into a sequence of tokens, which are concatenated to form the complete sequence of future tokens $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, where $N = T \times h \times w$ represents the total number of future tokens. Each latent token is a continuous vector of dimension D. Our generative modeling objective is to estimate the conditional distribution $p(\mathbf{x} \mid \mathbf{c})$ from the training data.

We achieve this using a masked generative framework, as illustrated in Figure 1. During training, we sample a binary mask $\mathbf{m} = [m_i]_{i=1}^N \sim p_{\mathcal{U}}$ and replace tokens x_i with a learnable, continuous `[MASK]` token where $m_i = 1$, creating a corrupted sequence $\overline{\mathbf{x}} = \mathbf{m}(\mathbf{x})$. The generative objective is to estimate the distribution of masked tokens conditioned on the visible and conditioning tokens:

$$\mathcal{L}_{\text{gen}}(\theta) = \mathop{\mathbb{E}}_{\mathbf{m}\sim p_{\mathcal{U}}} \left[\sum_{i \text{ s.t. } m_i=1} -\log p_\theta(x_i \mid \mathbf{c}, \overline{\mathbf{x}}) \right]. \tag{4}$$

The model processes the input by concatenating the conditioning tokens $\mathbf{c}$ with the corrupted future tokens $\overline{\mathbf{x}}$, adding positional encodings to the embedded sequence, and passing it through a bi-directional transformer backbone to obtain vectors z_i for each masked position. Given these vectors, the per-token objective $\log p_\theta(x_i \mid \mathbf{c}, \overline{\mathbf{x}})$ in Equation 4 simplifies to $\log p_\theta(x_i \mid z_i)$. To model this continuous distribution, we employ a diffusion model where z_i serves as conditional information for a denoising network – implemented as a small MLP on top of the transformer (Figure 2). We train the denoising network and the transformer backbone jointly using the diffusion loss specified in Equation 2. Conceptually, this diffusion objective encourages the model to produce representations z_i that facilitate effective denoising.

Auxiliary deterministic objective To encourage accurate predictions of near-term future tokens, we incorporate an auxiliary mean-squared error loss in the latent space. We implement this through a separate MLP head that produces deterministic predictions $\hat{x}_i$ from z_i, training it jointly with the transformer backbone. Since weather dynamics become increasingly chaotic beyond day 10, making deterministic predictions progressively less meaningful, we apply this loss only to the first 10 future frames. Furthermore, we employ an exponentially decreasing weighting scheme to emphasize the importance of accurate predictions for earlier frames. The deterministic objective is thus:

$$\mathcal{L}_{\text{deter}}(\theta) = \mathop{\mathbb{E}}_{\mathbf{m}\sim p_{\mathcal{U}}} \left[\sum_{m_i=1} w(i) ||x_i - \hat{x}_i||_2^2 \right]. \tag{5}$$

Appendix A.2 presents the details of this objective. The complete training objective combines both losses: $\mathcal{L}(\theta) = \mathcal{L}_{\text{gen}}(\theta) + \mathcal{L}_{\text{deter}}(\theta)$.

Sampling from OmniCast At inference time, we generate samples from $p(\mathbf{x} \mid \mathbf{c})$ through an iterative decoding process, starting from a sequence of fully masked future tokens. Each iteration consists of three steps: first, the transformer backbone processes the conditioning tokens and corrupted future tokens to produce vectors z_i for each masked position; second, a subset of masked positions is randomly selected according to a predefined schedule for unmasking; third, for each selected position, the diffusion model generates token x_i by conditioning on z_i and performing a fixed number of diffusion steps. This process iterates until all future tokens are revealed, at which point the VAE decoder maps the generated tokens back to the weather domain. To generate an ensemble of forecasts, we simply replicate the initial tokens and perform independent sampling for each copy. Four hyperparameters affect the sampling procedure: the number of unmasking iterations, the unmasking order, the number of diffusion steps, and the diffusion temperature.

4.3 Implementation details

Architectural details For the transformer backbone, we adopt the encoder-decoder architecture from Masked Autoencoder (MAE) [13]. The model processes an input sequence in two stages: first, the encoder processes the conditioning and visible tokens; second, the encoded sequence is augmented with learnable `[MASK]` tokens at appropriate positions and passed through the decoder to produce z_i for each position i. Both the encoder and decoder are bidirectional, employing full attention. Before feeding to either the encoder or decoder, we add the input sequences with positional

embeddings that combine two components: temporal embeddings to distinguish different frames, and spatial embeddings to differentiate tokens within each frame. The encoder and decoder follow the Transformer [46] implementation in ViT [9], each having 16 layers with 16 attention heads, a hidden dimension of 1024, and a dropout rate of 0.1.

Mask sampling During training, we sample a masking ratio $\gamma \sim \mathcal{U}[0.5, 1.0]$ and generate a corresponding binary mask $\mathbf{m}$, where $\gamma = 0.75$ indicates that 75% of entries in $\mathbf{m}$ are 1. For inference, we start with full masking ($\gamma = 1.0$) and gradually reduce it to 0.0 following a cosine schedule [3]. We set the number of unmasking iterations to match the number of future weather states T by default. We employ random masking orders across both spatial and temporal dimensions for training and inference.

Diffusion loss details We use a linear noise schedule with 1000 steps at training time that are resampled to 100 steps at inference. The denoising network ϵ_θ is implemented as a small MLP following Li et al. [26]. Specifically, the network consists of six residual blocks, each comprising a LayerNorm (LN), a linear layer, a SiLU activation, and another linear layer, with a residual connection around the block. Each block maintains a width of 2048 channels. The network takes the vector z_i from the transformer as conditioning information, which is combined with the time embedding of the diffusion step s through adaptive layer normalization (AdaLN) in each block's LN layers.

5 Experiments

We compare OmniCast with state-of-the-art deep learning and numerical methods on both medium-range and S2S time scales, using WeatherBench2 [40] (WB2) and ChaosBench [31] as benchmarks, respectively, and conduct extensive ablation studies to assess the contribution of each component in OmniCast. We further test the stability of OmniCast up to 100 years ahead in Appendix B.5.

Across both tasks, we train and evaluate OmniCast on 69 variables from the ERA5 reanalysis dataset [15], including four surface-level variables – 2-meter temperature (T2m), 10-meter U and V wind components (U10, V10), and mean sea-level pressure (MSLP), as well as five atmospheric variables – geopotential (Z), temperature (T), U and V wind components, and specific humidity (Q), each at 13 pressure levels {50, 100, 150, 200, 250, 300, 400, 500, 600, 700, 850, 925, 1000} hPa. For medium-range forecasting, we use native 0.25° resolution (721×1440 grids) and follow WB2 to train on years 1979–2018, validate on 2019, and test on 2020 using initial conditions at 00UTC and 12UTC. For S2S prediction, we downsample the data to 1.40625° (128×256 grids) and follow ChaosBench to train on 1979–2020, validate on 2021, and test on 2022 using 00UTC initializations.

5.1 OmniCast for S2S prediction

Training and inference details We train a VAE that embeds each weather state of shape $69 \times 128 \times 256$ into a latent map of shape $1024 \times 8 \times 16$, reducing spatial dimensions by a factor of 16. The architectural details and training process of the VAE are described in Appendix A.1. We train OmniCast to forecast a sequence of $T = 44$ future weather states at 24hr intervals, covering lead times from 1 to 44 days. Each training example consists of $45 \times 8 \times 16 = 5760$ latent tokens, including the initial condition. During inference, we generate the complete future sequence in 44 iterations (1 iteration per frame) using a diffusion temperature of $\tau = 1.3$. We produce an ensemble of 50 forecast sequences for each initial condition.

Baselines We compare OmniCast with PanguWeather (PW) [2] and GraphCast (GC) [22], two leading open-sourced deep learning methods, and ensemble systems of four numerical models from different national agencies: UKMO-ENS (UK) [53], NCEP-ENS (US) [42], CMA-ENS (China) [54], and ECMWF-ENS (Europe) [11]. We refer to ChaosBench for details about these baselines. Following ChaosBench, we report results on T850, Z500, and Q700 at lead times from 1 to 44 days. We additionally compare OmniCast with ClimaX [32] and Stormer [33] in Appendix B.2. We do not compare against Fuxi-S2S [5] as Fuxi-S2S forecasts daily average values from past daily averages, making it incomparable with OmniCast and the rest of the methods, which perform point-in-time weather forecasting based on an initial condition. We are also not able to run Gencast [37] and NeuralGCM [20] for S2S due to their significant computational demands.

Results Figure 4 compares different methods on three deterministic metrics: Root Mean-Squared Error (RMSE), Absolute Bias (ABS BIAS), and Multi-scale Structural Similarity (SSIM). At shorter

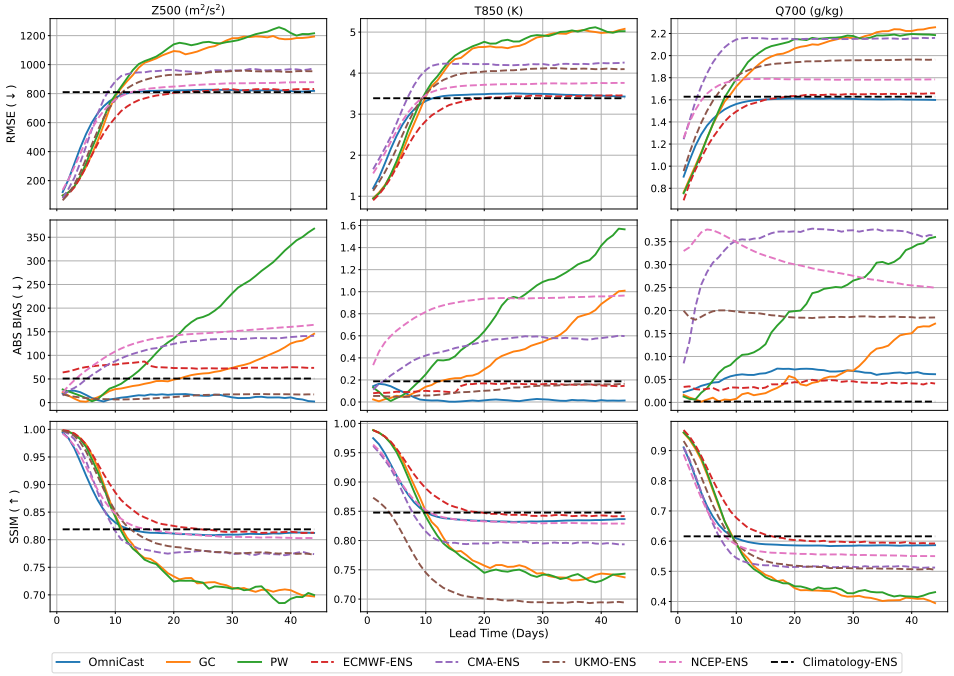


Figure 4: Deterministic performance of different methods at lead times from 1 to 44 days across three key variables. Solid curves are deep learning methods and dashed curves are numerical methods.

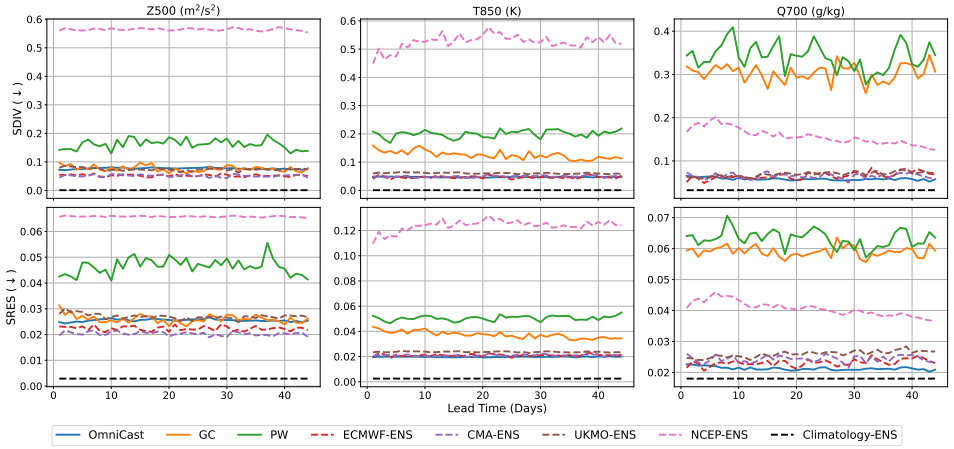


Figure 5: Physics-based metrics of different methods at lead times from 1 to 44 days across three key variables. Solid curves are deep learning methods and dashed curves are numerical methods.

lead times, OmniCast shows slightly worse performance on RMSE and SSIM than other baselines, which is expected since we train OmniCast to model a full sequence of future weather states rather than optimizing for short- and medium-range predictions. However, OmniCast's relative performance improves with increasing lead time, ultimately matching ECMWF-ENS as one of the top two performing methods beyond day 10. Notably, OmniCast demonstrates the lowest bias among all baselines, maintaining near-zero bias across all three target variables.

Physical consistency also plays a crucial role in S2S prediction, particularly for ensemble systems. We evaluate this aspect using two physics-based metrics: Spectral Divergence (SDIV) and Spectral Residual (SRES), which measure how closely the power spectra of predictions match those of ground-truths. As shown in Figure 5, OmniCast achieves substantially better physical consistency than other deep learning methods, and often outperforms all baselines on these metrics. These results demonstrate how OmniCast effectively preserves signals across the frequency spectrum.

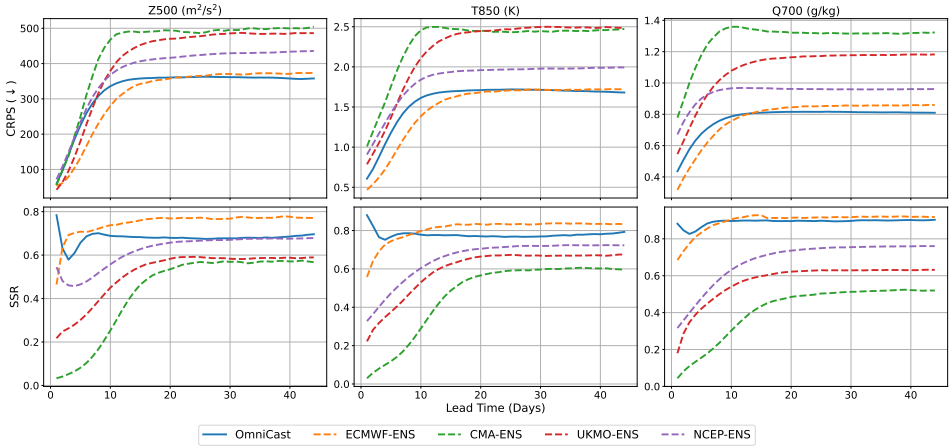


Figure 6: Probabilistic performance of different methods at lead times from 1 to 44 days across three key variables. Solid curves are deep learning methods and dashed curves are numerical methods.

Finally, we compare OmniCast with the four numerical ensemble systems on two probabilistic metrics: Continuous Ranked Probability Score (CRPS) and Spread/Skill Ratio (SSR) (closer to 1 is better). Figure 6 shows that OmniCast and ECMWF-ENS are the two leading methods across variables and lead times. Similar to deterministic results, OmniCast performs worse than ECMWF-ENS at shorter lead times but outperforms this baseline beyond day 15.

5.2 OmniCast for medium-range forecasting

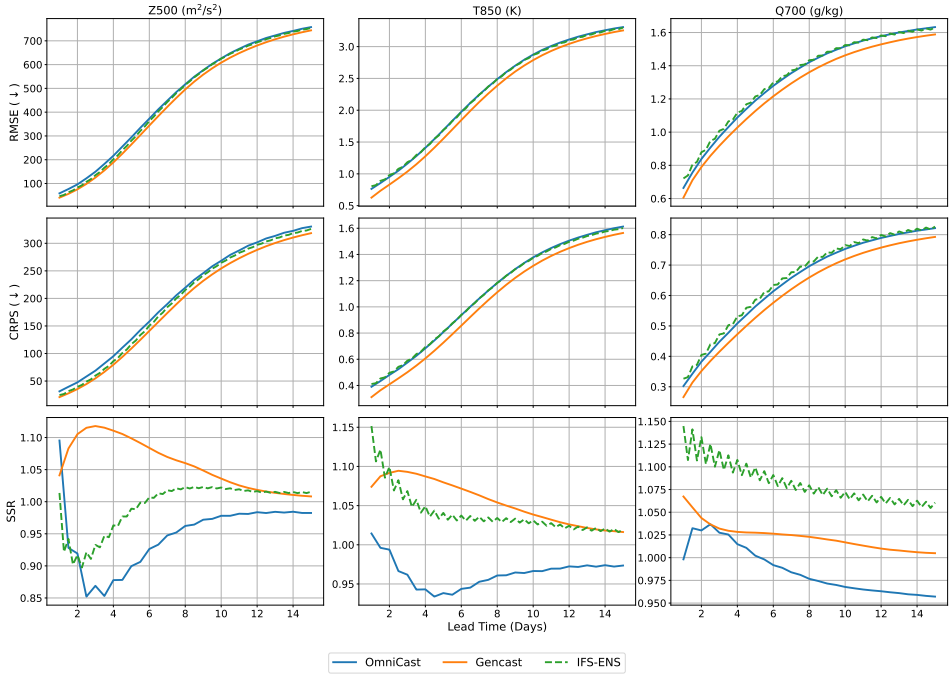


Figure 7: Probabilistic performance of different methods in medium-range forecasting. Solid curves are deep learning methods and dashed curves are numerical methods.

In addition to its strong performance on the S2S task, we demonstrate that OmniCast also performs competitively at the medium-range timescale. We train a VAE model with a spatial downsampling ratio of 16, compressing each weather state of shape $69 \times 721 \times 1440$ into a latent representation of size $256 \times 45 \times 90$. We then train OmniCast to predict two steps ahead at 12-hour intervals, following

the setup of Gencast [37]. During inference, we use autoregressive sampling, recursively feeding the most recent predicted frame as the new initial condition until the target lead time is reached. We generate forecasts using a single sampling iteration per frame with a diffusion temperature $\tau = 1.0$, and produce an ensemble of 50 members.

We compare OmniCast against Gencast [37], a leading deep learning method for probabilistic forecasting, and IFS-ENS [23], the gold-standard numerical ensemble system. Following WeatherBench2, we use ensemble RMSE, CRPS, and spread-skill ratio (SSR) as evaluation metrics. As shown in Figure 7, OmniCast performs comparably with IFS-ENS across all variables and metrics, and is only slightly behind Gencast. These results indicate that OmniCast achieves strong performance across both medium-range and S2S timescales.

5.3 Efficiency of OmniCast

Beyond its strong empirical performance, OmniCast offers substantial efficiency gains over existing methods. We trained OmniCast for 4 days using 32 NVIDIA A100 GPUs. In comparison, Gencast requires 5 days of training on 32 TPUv5e devices – hardware significantly more powerful than A100s, and NeuralGCM [20] requires 10 days on 128 TPUv5e devices. Additionally, Gencast employs a two-stage training pipeline, first pretraining on $1.0°$ resolution and then finetuning on $0.25°$, while we trained OmniCast in a single stage.

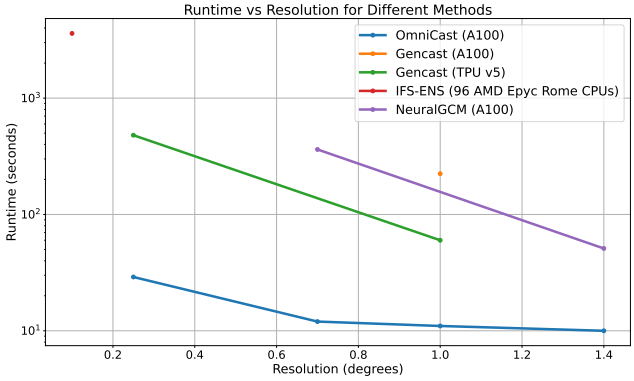


Figure 8: Runtime vs resolution to produce a 15-day forecast.

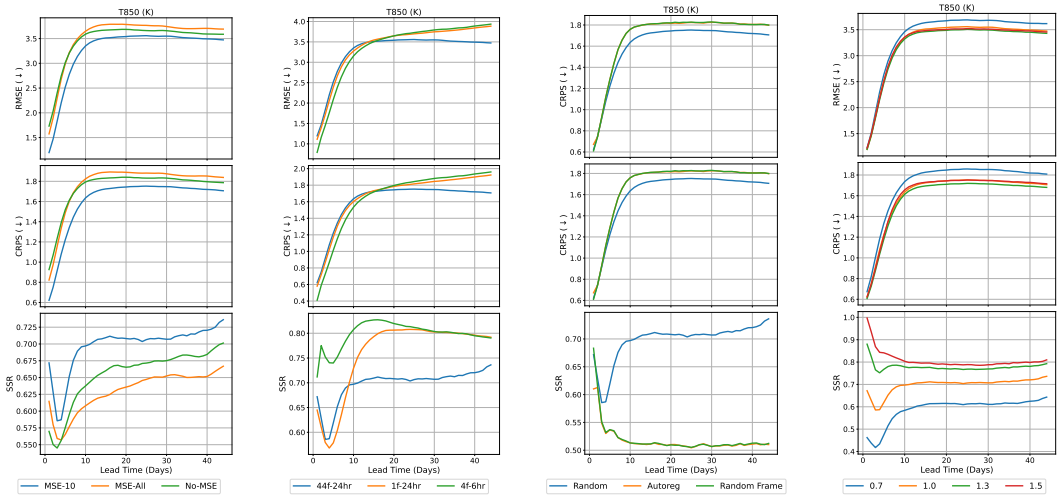
At inference time, OmniCast is orders of magnitude faster than Gencast, NeuralGCM, and IFS-ENS. Figure 8 compares the runtime (in seconds) required to generate a 15-day forecast across different resolutions. At $0.25°$ resolution, Gencast requires 480 seconds on TPUv5, whereas OmniCast achieves the same forecast in just 29 seconds on an A100. At $1.0°$, OmniCast completes inference in only 11 seconds, compared to 224 seconds for Gencast on the same hardware. These results highlight the scalability and practicality of OmniCast for operational forecasting.

The efficiency of OmniCast stems from two key architectural innovations. First, OmniCast operates in a much lower-dimensional latent space (45×90 latent grid vs 721×1440 original grid), significantly reducing the computational cost of training and inference. Second, OmniCast employs a highly efficient sampling mechanism. Unlike Gencast, which performs 50 full forward passes through the entire network for 50 diffusion steps, OmniCast requires only a single forward pass through the transformer backbone. The subsequent diffusion steps involve only lightweight forward passes through a compact MLP diffusion head, resulting in orders-of-magnitude lower inference time. Together, these design choices enable OmniCast to deliver fast and scalable forecasts.

5.4 Ablation studies

We analyze four key factors that influence OmniCast's performance: the auxiliary deterministic objective, training sequence length T, unmasking order during sampling, and diffusion sampling temperature τ. We present results for T850 on RMSE, CRPS, and SSR. We additionally study the impact of IC perturbations in Appendix B.3.

Impact of the deterministic objective Figure 9a demonstrates the important role of the deterministic loss in OmniCast's performance. Removing the MSE objective (No-MSE) degrades both RMSE and CRPS scores, with particularly noticeable impact at short lead times. However, naively applying MSE to all future frames (MSE-All-Frames) also proves counterproductive, as it forces deterministic predictions even for S2S timescales where weather systems become inherently chaotic. Our approach of applying MSE only to the first 10 frames achieves the best RMSE and CRPS scores across medium-range and S2S timescales.



(a) Impact of the deterministic objective.

(b) Impact of the training sequence length.

(c) Comparison of different unmasking strategies.

(d) Impact of diffusion sampling temperature τ.

Figure 9: Ablation studies showing the impact of different components in OmniCast.

Impact of training sequence length In the main S2S experiment, we train OmniCast to generate 44 future weather states at 24 hour intervals. One could alternatively train the model on shorter sequences and/or smaller intervals, then apply multiple roll-outs during inference to reach longer horizons. Figure 9b shows that models trained on shorter sequences or smaller intervals excel at short- and medium-range forecasting but underperform at S2S timescales. This trade-off emerges because shorter sequences allow models to specialize in near-term predictions, leading to better performance at shorter lead times. However, these models suffer from error accumulation at longer horizons, ultimately performing worse than the model trained on full sequences.

Impact of unmasking orders While our approach randomly masks tokens across both space and time during training, one may try more structured masking strategies at inference. We evaluate two such alternatives: an autoregressive strategy that unmasks entire frames sequentially, and a random framewise approach that unmasks complete frames in random order. Figure 9c shows that our fully randomized strategy achieves the best SSR scores, while both alternatives produce under-dispersive ensembles. The superior performance of the fully randomized approach stems from its introduction of additional randomness through the unmasking order, generating more diverse ensemble forecasts. This greater diversity consequently leads to better performance across other metrics.

Impact of diffusion sampling temperature Higher values of the temperature τ produce more diverse forecasts. Figure 9d demonstrates this empirically. Setting $\tau < 1$ produces under-dispersive ensembles, degrading performance across other metrics. Increasing τ boosts sample diversity, improving SSR scores and overall better performance. However, pushing τ too high (e.g., $\tau = 1.5$) causes samples to deviate from the mean prediction, compromising RMSE and CRPS performance. We identify $\tau = 1.3$ as the optimal value, providing the best balance between ensemble diversity and forecast quality, which we adopt for our main experiments.

6 Conclusion

We present OmniCast, a novel latent diffusion model for S2S prediction. By combining the masked generative framework with a diffusion objective, our approach enables direct modeling of long sequences of future weather states while avoiding error accumulation inherent in autoregressive methods. OmniCast achieves state-of-the-art performance in deterministic and probabilistic metrics while maintaining exceptional physical consistency. In medium-range forecasting, OmniCast performs competitively with existing methods while being significantly more efficient. Future work could study the fundamental trade-off between VAE reconstruction quality and transformer modeling capacity, and explore more sophisticated generative frameworks to enhance the diffusion objective.

References

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

[2] 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.

[3] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.

[4] Kang Chen, Tao Han, Junchao Gong, Lei Bai, Fenghua Ling, Jing-Jia Luo, Xi Chen, Leiming Ma, Tianning Zhang, Rui Su, et al. Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. *arXiv preprint arXiv:2304.02948*, 2023.

[5] Lei Chen, Xiaohui Zhong, Jie Wu, Deliang Chen, Shangping Xie, Qingchen Chao, Chensen Lin, Zixin Hu, Bo Lu, Hao Li, et al. Fuxi-s2s: An accurate machine learning model for global subseasonal forecasts. *arXiv preprint arXiv:2312.09926*, 2023.

[6] 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. *arXiv preprint arXiv:2306.12873*, 2023.

[7] Guillaume Couairon, Christian Lessig, Anastase Charantonis, and Claire Monteleoni. Archesweather: An efficient ai weather forecasting model at 1.5 {\deg} resolution. *arXiv preprint arXiv:2405.14527*, 2024.

[8] Daniela IV Domeisen, Christopher J White, Hilla Afargan-Gerstman, Ángel G Muñoz, Matthew A Janiga, Frédéric Vitart, C Ole Wulff, Salomé Antoine, Constantin Ardilouze, Lauriane Batté, et al. Advances in the subseasonal prediction of extreme events: Relevant case studies across the globe. *Bulletin of the American Meteorological Society*, 103(6):E1473–E1501, 2022.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[10] P. D. Dueben and P. Bauer. Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, 11(10):3999–4009, 2018. doi: 10.5194/gmd-11-3999-2018. URL `https://gmd.copernicus.org/articles/11/3999/2018/`.

[11] ECMWF. *IFS Documentation CY41R1 - Part V: The Ensemble Prediction System*. Number 5. ECMWF, 2015 2015. doi: 10.21957/eow1lonc. URL `https://www.ecmwf.int/node/9212`. Operational implementation 12 May 2015 .

[12] Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[14] Hans Hersbach, Bill Bell, Paul Berrisford, Gionata Biavati, András Horányi, Joaquín Muñoz Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Iryna Rozum, Dinand Schepers, Adrian Simmons, Cornel Soci, Dick Dee, and Jean-Noël Thépaut. ERA5 hourly data on single levels from 1979 to present. *Copernicus Climate Change Service (C3S) Climate Data Dtore (CDS)*, 10(10.24381), 2018.

[15] 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.

[16] Jessica Hwang, Paulo Orenstein, Judah Cohen, Karl Pfeiffer, and Lester Mackey. Improving subseasonal forecasting in the western us with machine learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2325–2335, 2019.

[17] Ryan Keisler. Forecasting global weather with graph neural networks. *arXiv preprint arXiv:2202.07575*, 2022.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL `https://arxiv.org/abs/1312.6114`.

[20] Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, Milan Klöwer, James Lottes, Stephan Rasp, Peter Düben, et al. Neural general circulation models for weather and climate. *Nature*, 632(8027):1060–1066, 2024.

[21] Alexander Kolesnikov, André Susano Pinto, Lucas Beyer, Xiaohua Zhai, Jeremiah Harmsen, and Neil Houlsby. Uvim: A unified modeling approach for vision with learned guiding codes. *Advances in Neural Information Processing Systems*, 35:26295–26308, 2022.

[22] 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*, 0(0):eadi2336, 2023. doi: 10.1126/science.adi2336. URL `https://www.science.org/doi/abs/10.1126/science.adi2336`.

[23] Simon Lang, Mark Rodwell, and Dinand Schepers. Ifs upgrade brings many improvements and unifies medium-range resolutions. *ECMWF Newsletter*, 176:21–28, 2023.

[24] Simon Lang, Mihai Alexe, Matthew Chantry, Jesper Dramsch, Florian Pinault, Baudouin Raoult, Mariana CA Clare, Christian Lessig, Michael Maier-Gerber, Linus Magnusson, et al. Aifs-ecmwf's data-driven forecasting system. *arXiv preprint arXiv:2406.01465*, 2024.

[25] Tianhong Li, Huiwen Chang, Shlok Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. Mage: Masked generative encoder to unify representation learning and image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2142–2152, 2023.

[26] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *arXiv preprint arXiv:2406.11838*, 2024.

[27] Edward N Lorenz. Forced and free variations of weather and climate. *Journal of Atmospheric Sciences*, 36(8):1367–1376, 1979.

[28] Annarita Mariotti, Paolo M Ruti, and Michel Rixen. Progress in subseasonal to seasonal prediction through a joint weather and climate community effort. *Npj Climate and Atmospheric Science*, 1(1):4, 2018.

[29] Soukayna Mouatadid, Paulo Orenstein, Genevieve Flaspohler, Miruna Oprescu, Judah Cohen, Franklyn Wang, Sean Knight, Maria Geogdzhayeva, Sam Levang, Ernest Fraenkel, et al. Subseasonalclimateusa: a dataset for subseasonal forecasting and benchmarking. *Advances in Neural Information Processing Systems*, 36, 2024.

[30] Congyi Nai, Xi Chen, Shangshang Yang, Yuan Liang, Ziniu Xiao, and Baoxiang Pan. Boosting weather forecast via generative superensemble. *arXiv preprint arXiv:2412.08377*, 2024.

[31] Juan Nathaniel, Yongquan Qu, Tung Nguyen, Sungduk Yu, Julius Busecke, Aditya Grover, and Pierre Gentine. Chaosbench: A multi-channel, physics-based benchmark for subseasonal-to-seasonal climate prediction. *arXiv preprint arXiv:2402.00712*, 2024.

[32] Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. ClimaX: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*, 2023.

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

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

[35] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[36] Kathy Pegion, Ben P Kirtman, Emily Becker, Dan C Collins, Emerson LaJoie, Robert Burgman, Ray Bell, Timothy DelSole, Dughong Min, Yuejian Zhu, et al. The subseasonal experiment (subx): A multimodel subseasonal prediction experiment. *Bulletin of the American Meteorological Society*, 100(10):2043–2060, 2019.

[37] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Timo Ewalds, Andrew El-Kadi, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. Gencast: Diffusion-based ensemble forecasting for medium-range weather. *arXiv preprint arXiv:2312.15796*, 2023.

[38] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[39] Stephan Rasp, Peter D Dueben, Sebastian Scher, Jonathan A Weyn, Soukayna Mouatadid, and Nils Thuerey. WeatherBench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002203, 2020.

[40] Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, Matthew Chantry, Zied Ben Bouallegue, Peter Dueben, Carla Bromberg, Jared Sisk, Luke Barrington, Aaron Bell, and Fei Sha. WeatherBench 2: A benchmark for the next generation of data-driven global weather models. *arXiv preprint arXiv:2308.15560*, 2023.

[41] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[42] Suranjana Saha, Shrinivas Moorthi, Xingren Wu, Jiande Wang, Sudhir Nadiga, Patrick Tripp, David Behringer, Yu-Tai Hou, Hui-ya Chuang, Mark Iredell, et al. The ncep climate forecast system version 2. *Journal of climate*, 27(6):2185–2208, 2014.

[43] Sebastian Scher. Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. *Geophysical Research Letters*, 45(22):12–616, 2018.

[44] Michael Tschannen, Cian Eastwood, and Fabian Mentzer. Givt: Generative infinite-vocabulary transformers. In *European Conference on Computer Vision*, pages 292–309. Springer, 2025.

[45] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[47] Frédéric Vitart. Evolution of ecmwf sub-seasonal forecast skill scores. *Quarterly Journal of the Royal Meteorological Society*, 140(683):1889–1899, 2014.

[48] Frederic Vitart, Constantin Ardilouze, Axel Bonet, Anca Brookshaw, M Chen, C Codorean, M Déqué, L Ferranti, E Fucile, M Fuentes, et al. The subseasonal to seasonal (s2s) prediction project database. *Bulletin of the American Meteorological Society*, 98(1):163–173, 2017.

[49] NP Wedi, P Bauer, W Denoninck, M Diamantakis, M Hamrud, C Kuhnlein, S Malardel, K Mogensen, G Mozdzynski, and PK Smolarkiewicz. *The modelling infrastructure of the Integrated Forecasting System: Recent advances and future challenges*. European Centre for Medium-Range Weather Forecasts, 2015.

[50] Jonathan A Weyn, Dale R Durran, and Rich Caruana. Can machines learn to predict weather? Using deep learning to predict gridded 500-hPa geopotential height from historical weather data. *Journal of Advances in Modeling Earth Systems*, 11(8):2680–2693, 2019.

[51] Christopher J White, Henrik Carlsen, Andrew W Robertson, Richard JT Klein, Jeffrey K Lazo, Arun Kumar, Frederic Vitart, Erin Coughlan de Perez, Andrea J Ray, Virginia Murray, et al. Potential applications of subseasonal-to-seasonal (s2s) predictions. *Meteorological applications*, 24(3):315–325, 2017.

[52] Christopher J White, Daniela IV Domeisen, Nachiketa Acharya, Elijah A Adefisan, Michael L Anderson, Stella Aura, Ahmed A Balogun, Douglas Bertram, Sonia Bluhm, David J Brayshaw, et al. Advances in the application and utility of subseasonal-to-seasonal predictions. *Bulletin of the American Meteorological Society*, 103(6):E1448–E1472, 2022.

[53] KD Williams, CM Harris, A Bodas-Salcedo, J Camp, RE Comer, D Copsey, D Fereday, T Graham, R Hill, T Hinton, et al. The met office global coupled model 2.0 (gc2) configuration. *Geoscientific Model Development*, 88(55):1509–1524, 2015.

[54] Tongwen Wu, Yixiong Lu, Yongjie Fang, Xiaoge Xin, Laurent Li, Weiping Li, Weihua Jie, Jie Zhang, Yiming Liu, Li Zhang, et al. The beijing climate center climate system model (bcc-csm): The main progress from cmip5 to cmip6. *Geoscientific Model Development*, 12(4):1573–1600, 2019.

[55] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, et al. Magvit: Masked generative video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10459–10469, 2023.

A Implementation details

A.1 VAE details

Our VAE model follows the UNet implementation from PDEArena [12]. We use the following hyperparameters for UNet in our experiments.

Table 1: Default hyperparameters of UNet

Hyperparameter	Meaning	Value
Padding size	Padding size of each convolution layer	1
Kernel size	Kernel size of each convolution layer	3
Stride	Stride of each convolution layer	1
Input channels	The number of channels of the input	69
Input channels	The number of channels of the output	69
Base channels	The base hidden dimension of the UNet	256
Channel multiplications	Determine the number of output channels for Down and Up blocks	$[1, 2, 4, 4, 8]$
Dimension of z	The dimension of the latent space	1024
Blocks	Number of blocks	2
Use attention	If use attention in Down and Up blocks	False
Dropout	Dropout rate	0.0

The VAE encoder embeds each weather state of shape $69 \times 128 \times 256$ to a latent map of shape $1024 \times 8 \times 16$, reducing the spatial dimensions by 16. We use a KL weight of $5e-5$ and optimize the VAE model with Adam [18] for 200 epochs with a batch size of 32, a base learning rate of $2e-4$, parameters $(\beta_1 = 0.9, \beta_2 = 0.95)$, and weight decay of $1e-5$. The learning rate follows a linear warmup for the first 20 epochs, followed by a cosine decay schedule for the remaining 180 epochs.

A.2 Weighted deterministic objective

In OmniCast, we employ a weighted MSE objective to encourage accurate deterministic predictions for near-term frames. The objective is formulated as:

$$\mathcal{L}_{\text{deter}}(\theta) = \mathop{\mathbb{E}}_{\mathbf{m}\sim p_{\mathcal{U}}} \left[\sum_{m_i=1} w(i) ||x_i - \hat{x}_i||_2^2 \right], \tag{6}$$

where $w(i)$ is an exponentially decreasing weighting function. We compute this weight in three steps. First, for each token i, we determine its corresponding frame index $k = \lfloor \frac{i}{h \times w} \rfloor$, where $h \times w$ represents the spatial dimensions of each frame's latent map. Second, we assign weights to tokens based on their frame index: $w(i) = e^{-k} = e^{-\lfloor \frac{i}{h \times w} \rfloor}$, ensuring all tokens from the same frame receive equal weight. Third, we set $w(i) = 0$ for tokens beyond frame 10 and normalize the remaining weights to sum to one.

A.3 Optimization details

We optimize OmniCast with AdamW [18] for 100 epochs with a batch size of 32, a base learning rate of $2e-4$, parameters $(\beta_1 = 0.9, \beta_2 = 0.95)$, and weight decay of $1e-5$. The learning rate follows a linear warmup for the first 10 epochs, followed by a cosine decay schedule for the remaining 90 epochs.

B Additional experiments

B.1 VAE reconstruction quality

The VAE model is a critical component in OmniCast, since it imposes an upper bound on the forecasting performance. We dedicated substantial efforts to designing the VAE model that balances

reconstruction quality and compression. Our primary goal was to achieve a high compression ratio along the spatial dimensions, as this directly reduces the number of training tokens required for the subsequent transformer model. We employed $16\times$ spatial reduction for both the medium-range setting with $0.25°$ data and the S2S setting with the $1.40625°$ data. We then incrementally increased the latent dimension until we obtained an acceptable reconstruction error. Table 2 shows that increasing the latent dimension consistently improves the reconstruction errors across different variables. We did not increase the latent dimension beyond 1024 since it would create difficulties for training with the diffusion objective. It is also noticeable that the VAE model trained on $0.25°$ data performs much better than the one trained on $1.40625°$ with the same spatial compression ratio and latent dimension. This is expected since higher-resolution data has more spatial redundancy, leading to easier compression for the VAE model. Figures 10 and 11 visualize the VAE reconstructions for 6 surface and pressure-level variables. The VAE was able to retain important details and structures of the input, albeit slightly smoothing out the data.

Table 2: **Reconstruction error of VAE models for different physical variables and latent dimensions (D).** Results are shown for datasets at two spatial resolutions: $1.40625°$ (left) and $0.25°$ (right). Lower values indicate better reconstruction.

	$1.40625°$ resolution					$0.25°$ resolution				
	T2m	U10	V10	Z500	T850	T2m	U10	V10	Z500	T850
$D = 256$	0.96	0.65	0.62	48.72	0.77	**0.55**	**0.25**	**0.23**	**18.77**	**0.37**
$D = 512$	0.80	0.51	0.48	35.42	0.64	—	—	—	—	—
$D = 1024$	**0.71**	**0.43**	**0.40**	**27.34**	**0.57**	—	—	—	—	—

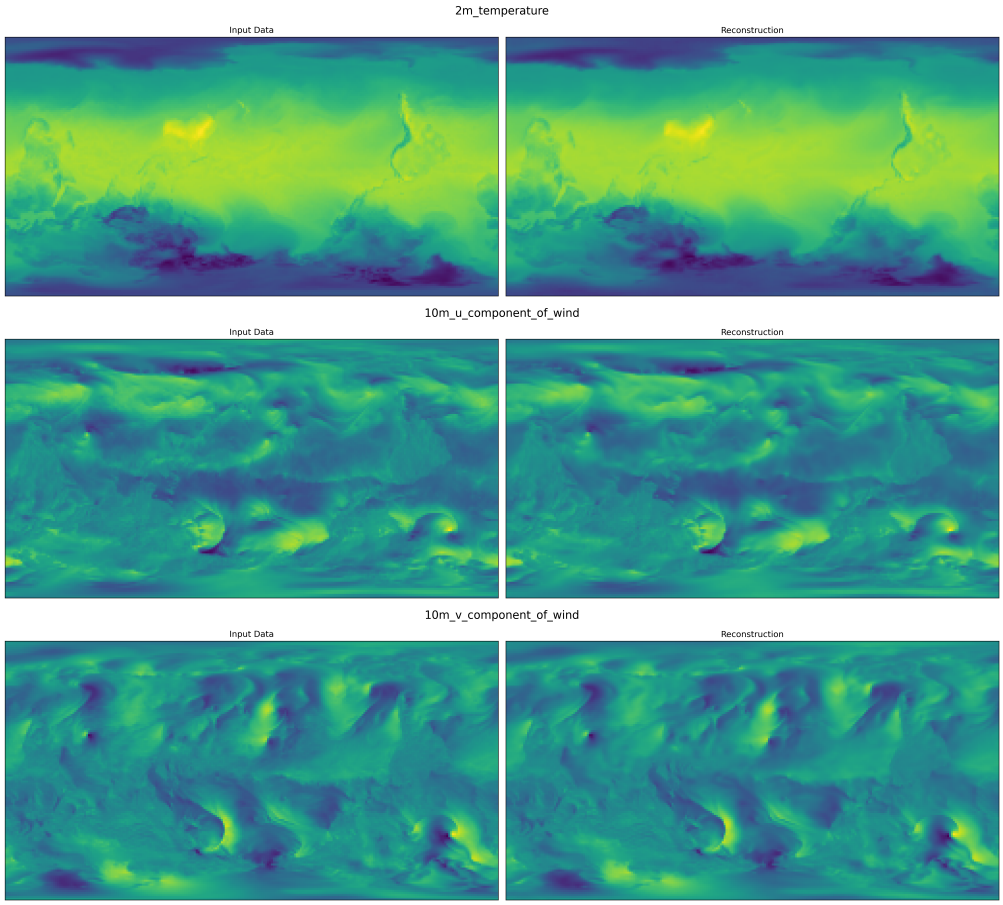


Figure 10: Reconstructions of the VAE model for T2m, U10, and V10.

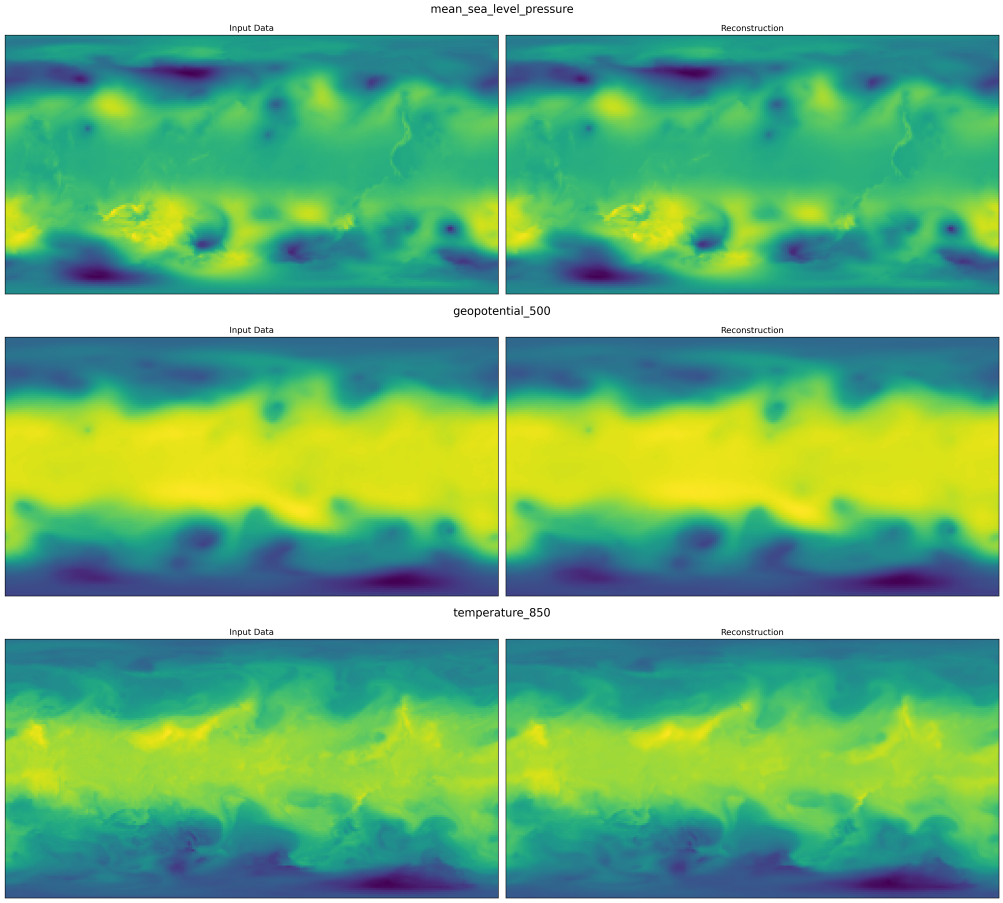


Figure 11: Reconstructions of the VAE model for MSLP, Z500, and T850.

Before selecting the specific VAE model presented in our paper, we also tried two alternative VAE architectures: VQ-VAE [45], which compresses data into a discrete latent space, and Video VAE [1], which compresses data across both spatial and temporal dimensions. Our early experiments with VQ-VAE did not achieve satisfactory reconstruction qualities, as the errors were consistently 2 to 3 times higher than those obtained with a continuous VAE using the same spatial downsampling factor. We also found that for an equivalent effective compression ratio, a per-frame VAE consistently outperformed a video VAE. These results led us to opt for the per-frame continuous VAE model.

B.2 Comparison with more deep learning baselines

In addition to PanguWeather and GraphCast, we compare OmniCast with two advanced transformer-based methods: ClimaX [32] and Stormer [33]. Figure 12 shows that Stormer achieves superior accuracy in short-to-medium timescales, consistent with its reported results. However, as an autoregressive method, its performance degrades more rapidly than OmniCast, eventually falling below Climatology, albeit at a slower rate than PanguWeather and GraphCast. ClimaX takes a different approach as a direct forecasting method, where a model trained on large-scale climate data is finetuned specifically for individual lead times. This approach avoids error accumulation and achieves comparable performance with OmniCast at S2S scales. However, ClimaX requires fine-tuning separate models for each target lead time, while a single OmniCast model can simultaneously generate the complete sequence of future weather states.

B.3 Impact of IC perturbations

Initial condition (IC) perturbations—adding random noise to initial conditions X_0 – are a standard technique in numerical methods for generating ensemble forecasts. This approach complements our

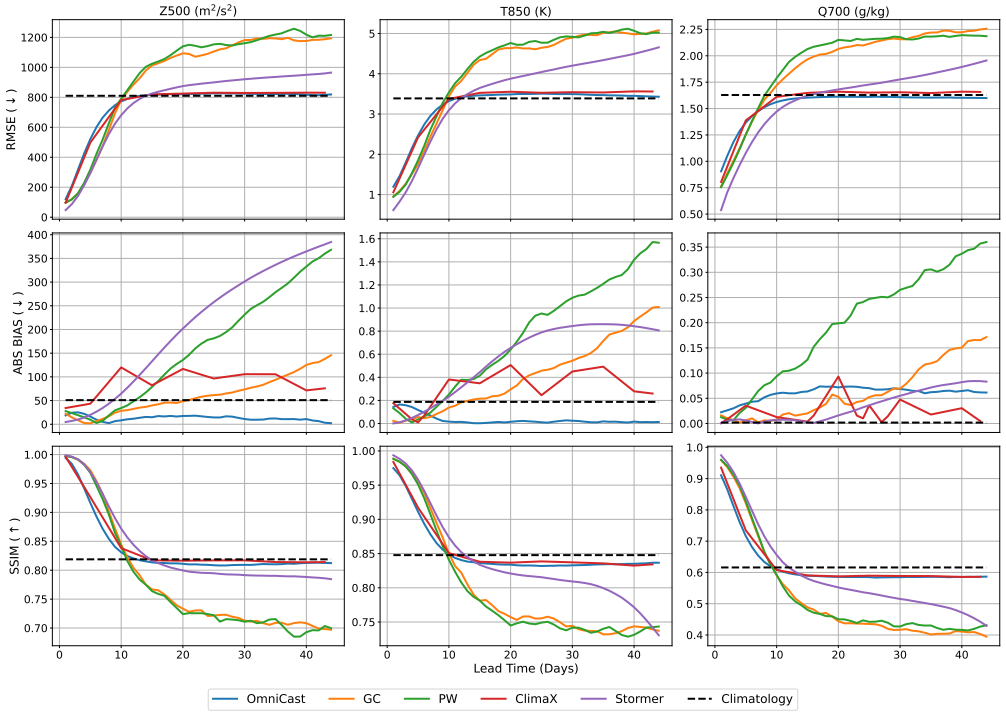


Figure 12: Comparison of deterministic performance of OmniCast with more deep learning methods.

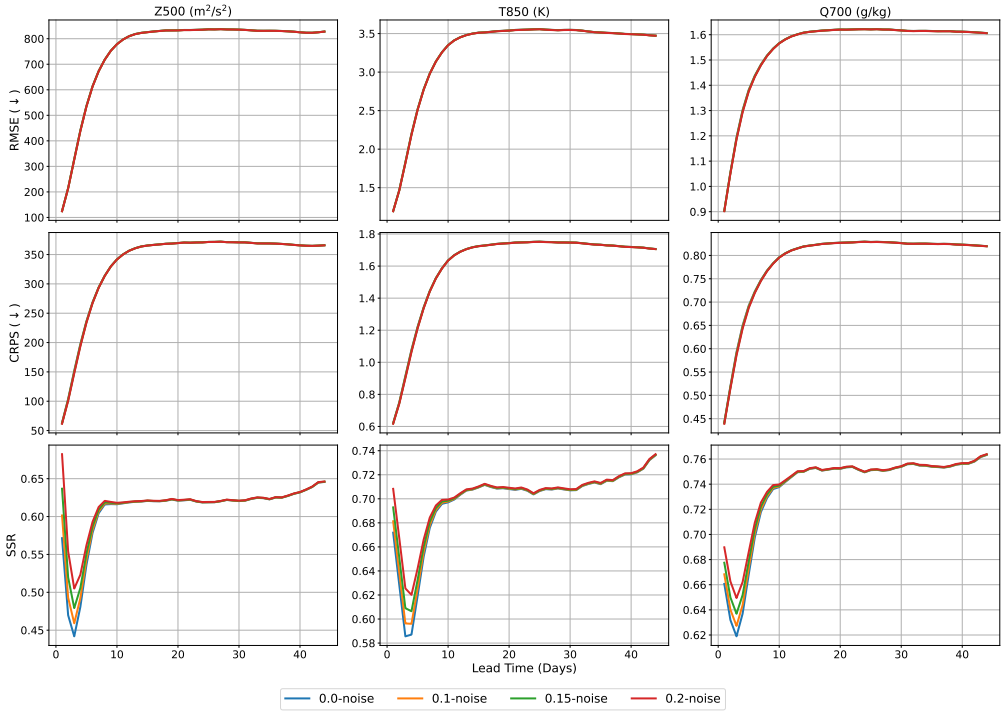


Figure 13: Performance of OmniCast with different levels of IC noise.

generative framework. Figure 13 evaluates OmniCast's performance across different noise levels, varying the standard deviation of the Gaussian distribution used for generating perturbations. The results demonstrate OmniCast's robustness to input noise, maintaining consistent RMSE and CRPS

scores across noise levels from 0.0 to 0.2, with only minor variations in SSR scores at short lead times.

B.4 Scaling inference compute

Finally, we examine how increasing inference compute affects OmniCast's performance through two hyperparameters: the number of ensemble forecasts and the average number of unmasking iterations per frame, i.e., 1-iter means a total of 44 iterations for 44 frames. Figure 14 shows that generating more ensemble forecasts improves both system diversity (higher SSR) and mean prediction accuracy (lower RMSE). Interestingly, while increasing the number of unmasking iterations shows minimal impact on RMSE, it yields slight improvements in SSR. This improvement likely stems from the increased randomness in unmasking order with more iterations, leading to greater ensemble diversity.

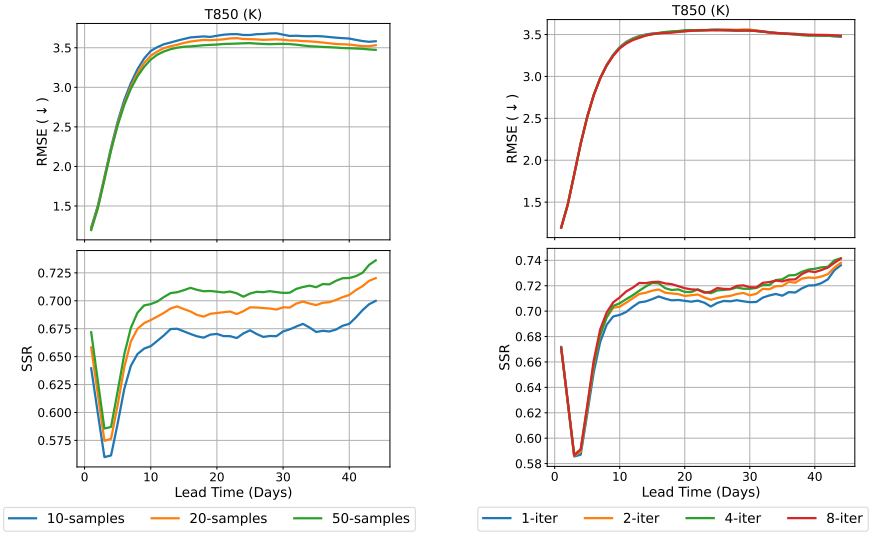


Figure 14: Performance of OmniCast as we vary the number of ensemble forecasts (left) and the number of unmasking iterations.

B.5 Testing OmniCast stability

We tested the stability of OmniCast by rolling out the model to 100 years into the future. We found that OmniCast consistently produces stable and physically feasible forecasts, even at 100 years ahead. Please see below for visualizations of OmniCast's rollouts for various weather variables with 4 samples each.

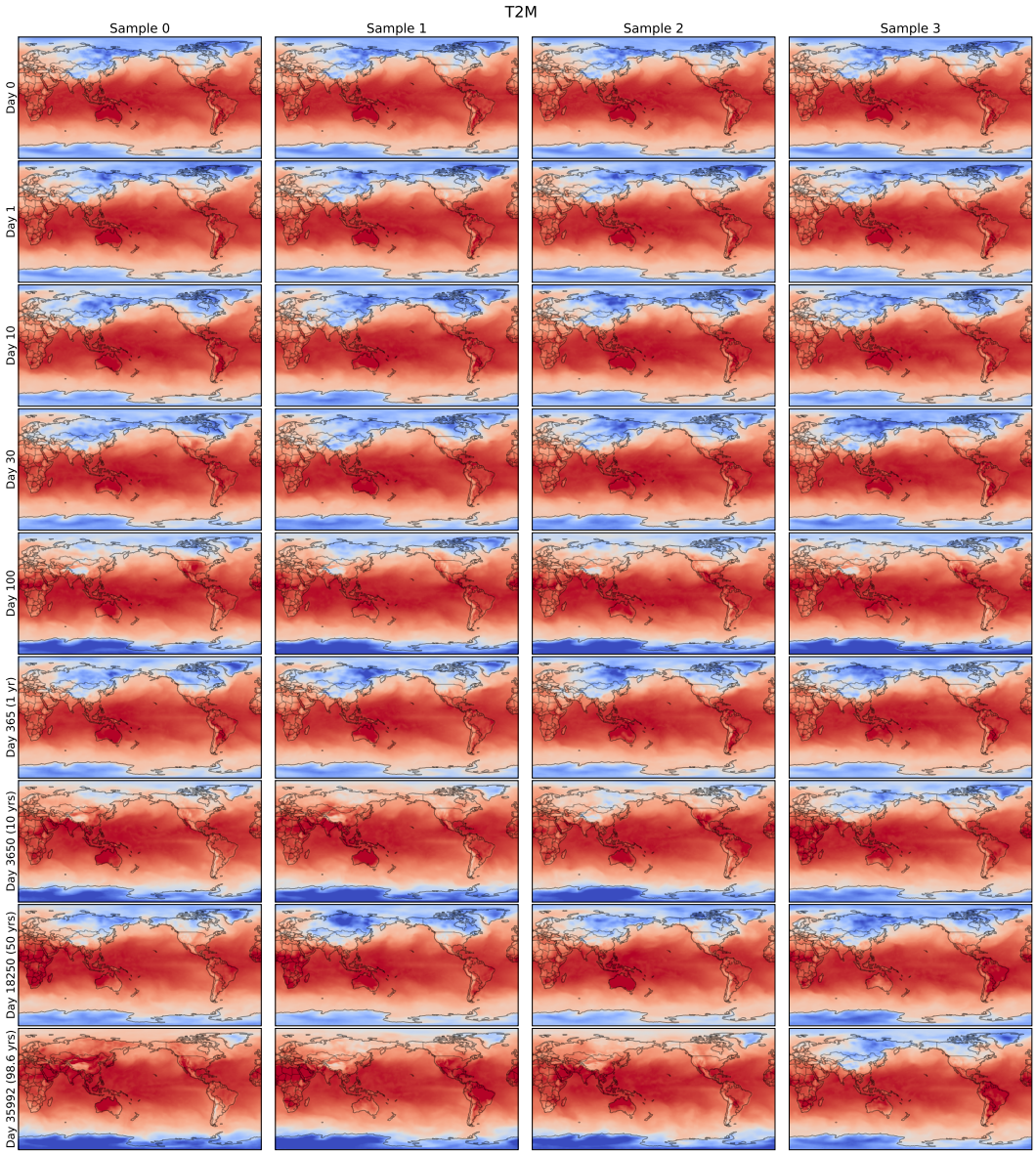


Figure 15: Rollouts for 2-meter temperature up to 100 years ahead.

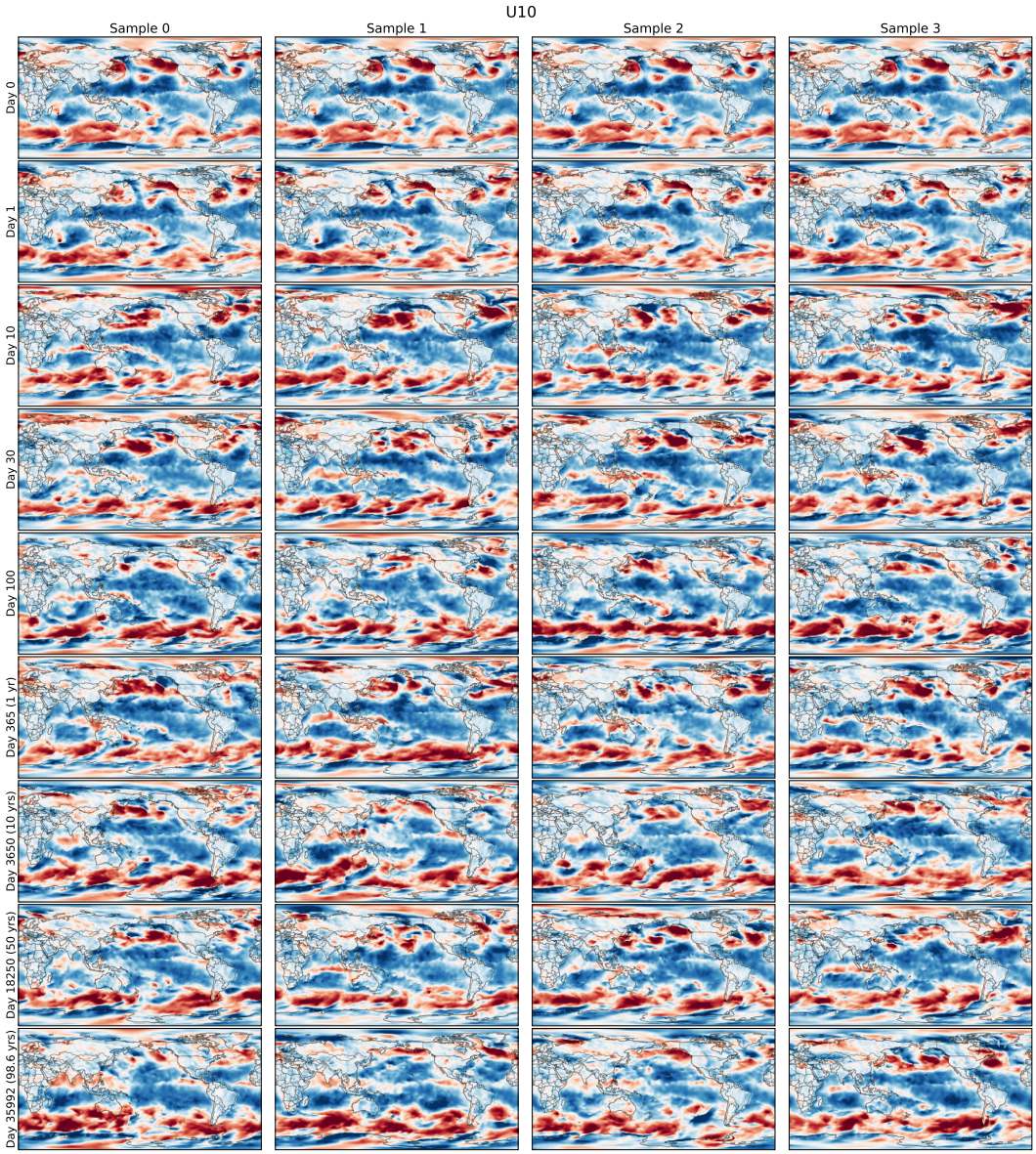


Figure 16: Rollouts for 10-meter u component of wind up to 100 years ahead.

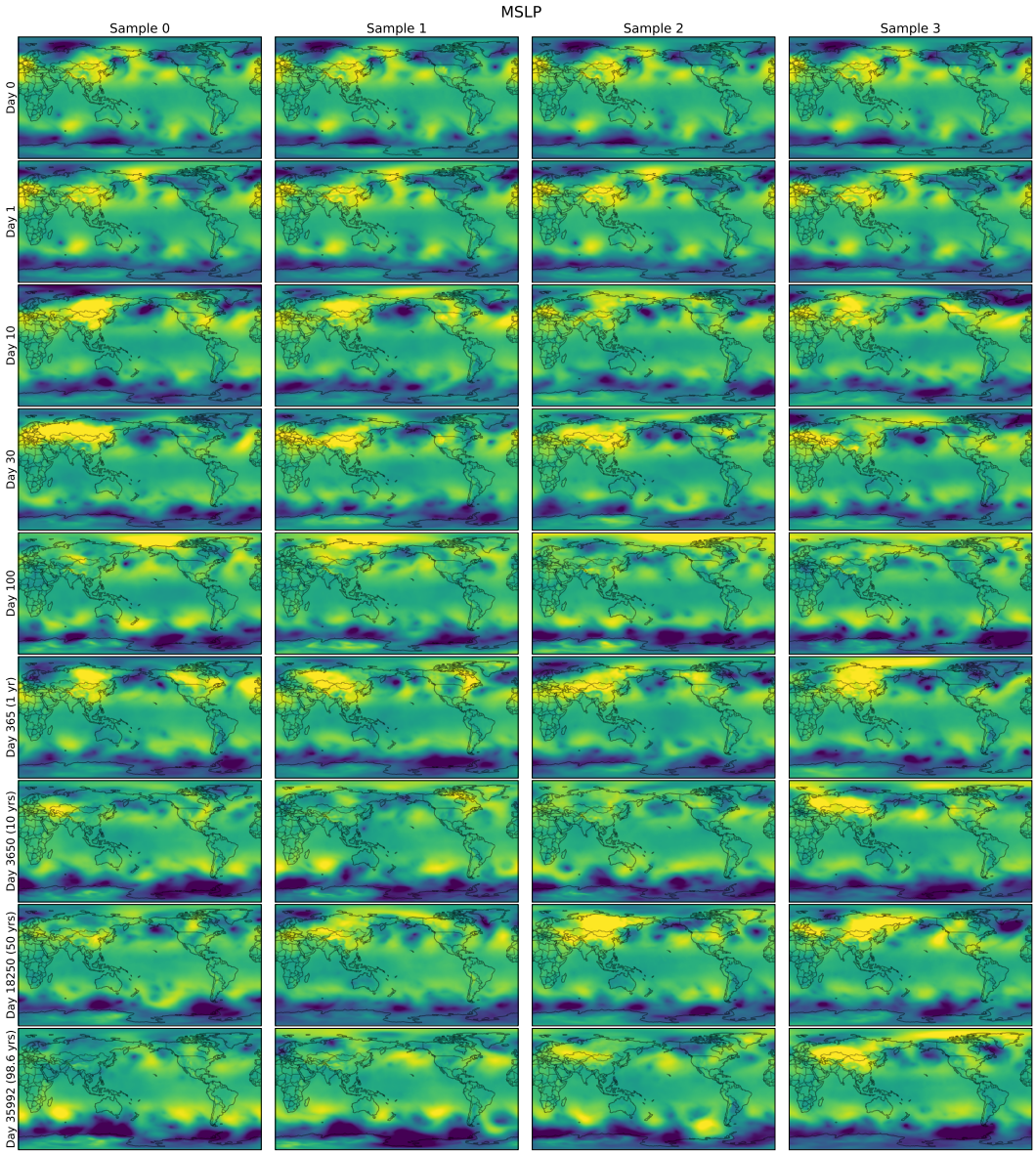


Figure 17: Rollouts for mean sea level pressure up to 100 years ahead.

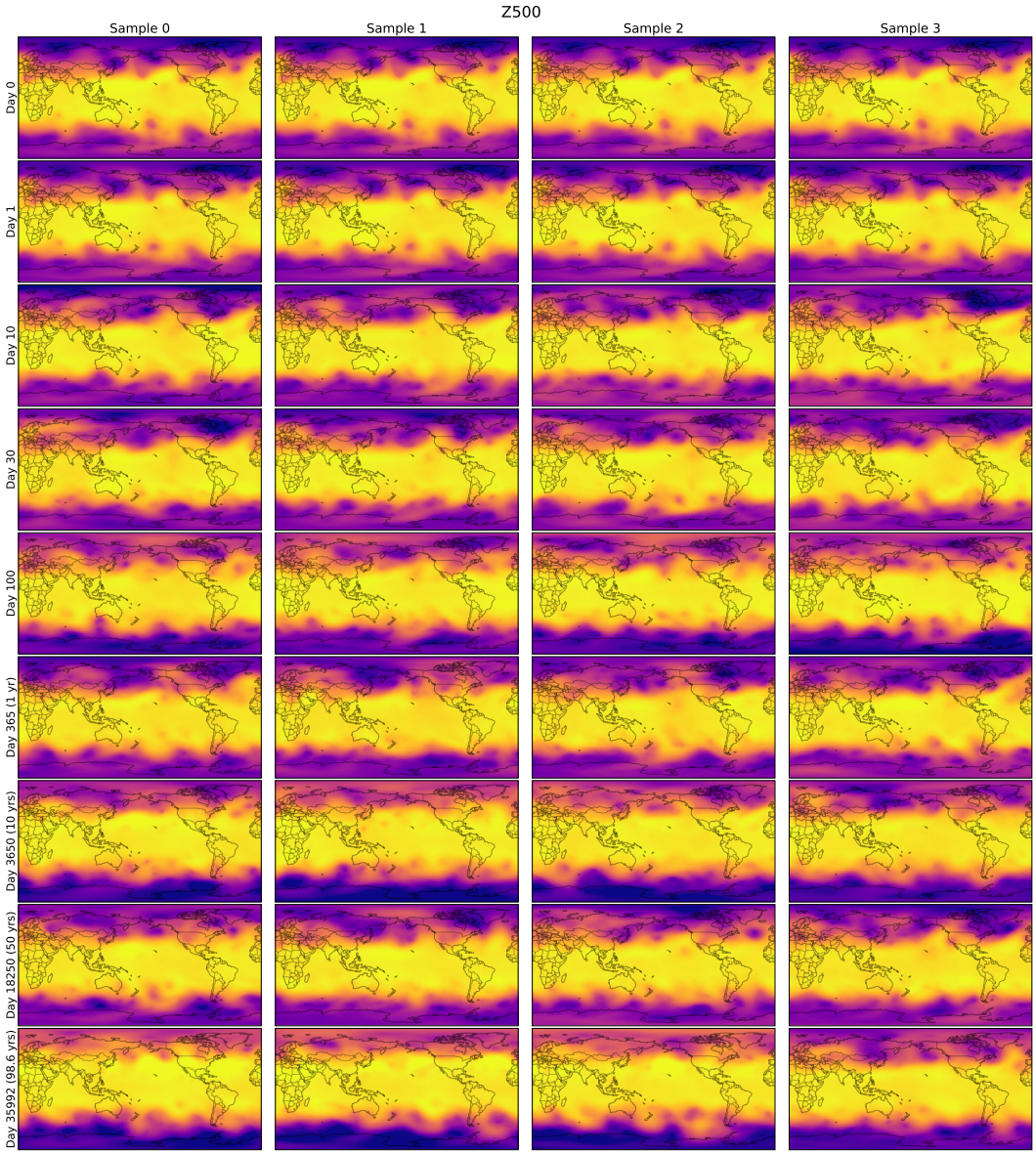


Figure 18: Rollouts for 500hPa geopotential up to 100 years ahead.

Figure 19: Rollouts for 700hPa specific humidity up to 100 years ahead.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction reflect the paper's contributions – the paper proposes a novel method for probabilistic S2S prediction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We discussed limitations in the Conclusion section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not present theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provided the implementation details in the Experiments section and Appendix. We will also open-source the code and pretrained weights.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will publish the code and model checkpoints upon paper acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide these details in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: It's too expensive to do so and it's not a standard in the field of weather forecasting.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide this information in the Experiments section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper is foundational research and there is no direct negative societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This does not apply to our work.

    Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We used open-sourced models and code and properly cited them.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The paper does not introduce new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.