AERIS: Argonne Earth Systems Model for Reliable and Skillful Predictions

Väinö Hatanpää[1,+], Eugene Ku[1,+], Jason Stock[1,+], Murali Emani[1], Sam Foreman[1], Chunyong Jung[1], Sandeep Madireddy[1], Tung Nguyen[4], Varuni Sastry[1], Ray A. O. Sinurat[2], Sam Wheeler[1], Huihuo Zheng[1], Troy Arcomano[1,3], Venkatram Vishwanath[1,*], and Rao Kotamarthi[1,*]

[1]Argonne National Laboratory, Lemont, Illinois, USA
[2]University of Chicago, Chicago, Illinois, USA
[3]Allen Institute for AI, Seattle, Washington, USA
[4]University of California, Los Angeles, California, USA
[+]*Joint First Authors* [*]{*venkat,vrkotamarthi*}*@anl.gov*

***Abstract*—Generative machine learning offers new opportunities to better understand complex Earth system dynamics. Recent diffusion-based methods address spectral biases and improve ensemble calibration in weather forecasting compared to deterministic methods, yet have so far proven difficult to scale stably at high resolutions. We introduce AERIS, a 1.3 to 80B parameter pixel-level Swin diffusion transformer to address this gap, and SWiPe, a generalizable technique that composes window parallelism with sequence and pipeline parallelism to shard window-based transformers without added communication cost or increased global batch size. On Aurora (10,080 nodes), AERIS sustains 10.21 ExaFLOPS (mixed precision) and a peak performance of 11.21 ExaFLOPS with 1×1 patch size on the 0.25° ERA5 dataset, achieving 95.5% weak scaling efficiency, and 81.6% strong scaling efficiency. AERIS outperforms the IFS ENS and remains stable on seasonal scales to 90 days, highlighting the potential of billion-parameter diffusion models for weather and climate prediction.**

***Index Terms*—high-performance computing, machine learning, generative diffusion, climate modeling, weather forecasting**

I. Justification

AERIS achieves a sustained mixed-precision performance of 10.21 ExaFLOPS and peak performance of 11.21 ExaFLOPS, scaling to 10,080 nodes (120,960 GPU-tiles) on the Aurora supercomputer—highest achieved to date in AI for Science. This accomplishment represents a significant breakthrough in generative modeling for science applications.

II. Performance Attributes

Performance Attribute	Our Submission
Category of achievement	Scalability; time-to-solution
Type of method used	Explicit; deep learning
Results reported with	Whole application + I/O
Precision reported	Mixed precision (`BF16`)
System scale	Measured on full system
Measurement mechanism	Timers; performance modeling

Authors are listed alphabetically by their last names.

III. Overview of the Problem

Weather and subseasonal-to-seasonal (S2S) forecasting is a fundamental problem for science and society. Accurate forecasts helps us prepare and recover from the effects of natural disasters and extreme weather events. Traditionally, domain scientists have relied on numerical weather prediction (NWP) techniques [1] to simulate and model complex atmospheric and climate dynamics, including both short- and long-term (on seasonal scale) weather forecasting. These models utilize systems of differential equations describing fluid flow and thermodynamics, which can be integrated over time to obtain future forecasts [1], [2]. When computed on large CPU-based HPC machines, these models typically produce global, 14-day forecasts four times a day [3].

Despite their widespread use, NWP models face several challenges. They suffer from parameterization uncertainties of important small-scale physical processes, including cloud physics and radiation, which can affect forecasting accuracy [4]. They also incur high computation costs due to the complexity of numerical integration. Furthermore, NWP forecast accuracy does not inherently improve with increasing amounts of data; instead, their effectiveness heavily depends on domain experts continuously refining equations, parameterizations, and numerical algorithms [5]. These challenges compound significantly with increased spatiotemporal resolutions and forecast dimensions, e.g., the number of ensembles and lead time of future forecasts.

To address these aforementioned challenges, there has been an increasing interest in data-driven approaches based on deep learning models for weather forecasting [6]–[8] (also see Section IV-A). The central idea involves parameterizing neural networks to predict future weather conditions using a vast amount of historical data, such as the ECMWF Reanalysis v5 (ERA5) dataset [9]–[12]. Once trained, these models generate forecasts within seconds, compared to the hours needed by typical NWP models.

Early attempts relied on coarse reanalysis data (∼400km)

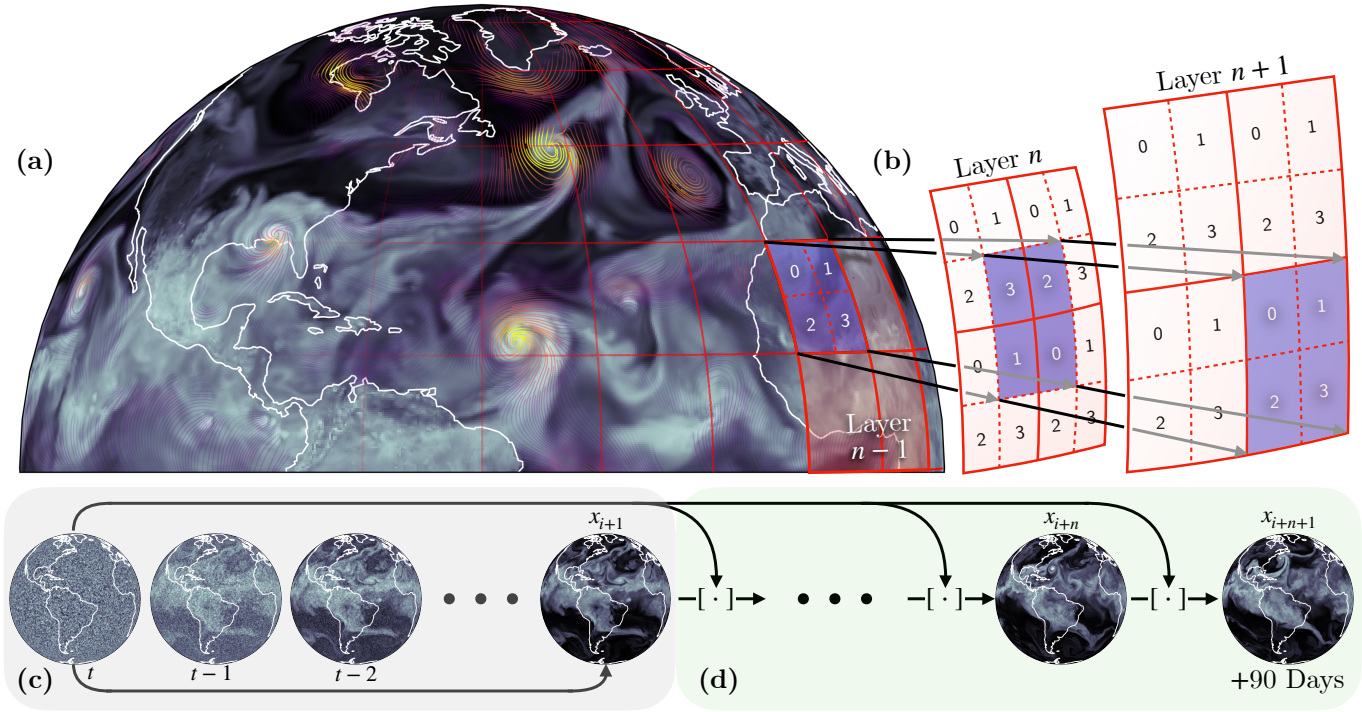


Fig. 1: High-level overview of AERIS. (a) Forecast over the Atlantic basin showing Hurricane Teddy, Tropical Storm Sally, and Post-Tropical Cyclone Paulette seen with 10m wind tangents and intensity superimposed on specific humidity at 700 hPa (Q700; 2020-09-16T18z) with 60×60-pixel (at $15°$) windows outlined in red; (b) sequence-window parallelism over network layers $n-1$ to $n+1$ optimally sliding partitions of an example window between nodes; (c) iterative diffusion steps to generate diverse global ensembles from any given initial condition; and (d) autoregressive steps to generate stable forecasts to 90 days.

and produced forecasts that fell short of current NWP [13]–[17]. In later works using the native $0.25°$ resolution of ERA5 ($\sim$30km), alongside advances in architectures (i.e., transformers and graph neural networks) and training strategies [18]–[25], performance became competitive with NWP. Similar challenges arose in natural language processing [26]–[29] and computer vision [30], [31], where breakthroughs inspired analogous efforts in weather and climate modeling [32]–[34]. These studies show that higher horizontal and vertical resolution, together with architectural scaling, improves forecast skill, particularly with transformer-based models [19], [20], [32]. However, these models remain expensive to train and are limited by existing parallelism techniques (e.g., sequence, pipeline, and sharded data parallelisms), limitations that compound with increasing model size and data resolution.

Among transformer-based architectures, the Swin Transformer [35], [36] is particularly well-suited for spatiotemporal modeling, combining the structured efficiency of window-based attention with the scaling advantages of standard transformers. Its design makes it effective for high-resolution data and compatible with existing parallelisms. At the same time, the independence of localized attention windows exposes an additional layer of parallelism that, despite its potential, has received little attention. In this work, we identify and exploit this windowed structure, significantly improving the scalability potential of window-based transformers.

To this end, we introduce AERIS, a pixel-level Swin Transformer for data-driven weather and S2S forecasting (Figure 1). The architecture builds on advances from recent large-scale models and leverages our generalizable window-based parallelism strategy to scale efficiently with both model size and data resolution. Trained as a generative diffusion model on global $0.25°$ ERA5 reanalysis data, our formulation produces 6- and 24-hourly forecast ensembles competitive on medium-range with impressive seasonal stability, using an architecture that can be finetuned or distilled to downstream tasks.

In summary, our contributions are:

- Stable large-scale training of 1.3B–80B parameter vision models, sustaining 10.21 ExaFLOPS in mixed-precision performance on 10,080 Aurora nodes (120,960 GPU-tiles) and 0.54 ExaFLOPS on 1,008 LUMI nodes.
- The first billion-parameter diffusion model for weather and climate, operating at the pixel level (1×1 patch size) and further guided by physical priors.
- SWiPe, a generalizable parallelism strategy that shards window-based transformers across high-resolution inputs, enabling efficient small-batch training and scalable performance on large supercomputers.
- Medium-range forecast skill surpassing IFS ENS and competitive with GenCast, while uniquely stable on seasonal scales to 90 days, validated through domain-specific diagnostics and case studies of hurricanes and heatwaves.

IV. State of the Art

Section IV-A begins with a review of related machine learning approaches for weather prediction and their relevance to this work, followed by a discussion in Section IV-B comparing existing parallelism techniques and scaling strategies.

A. Deep Learning Methods

There are many data-driven, machine learning approaches to model the global evolution of the Earth system. Broadly, these can be divided into deterministic methods [19], [20], [37]–[39] and probabilistic or generative ones [40]–[43]. The former is widely adopted in practice, owing to their relative training simplicity, with models such as GraphCast [37] and FourCastNet [19] delivering competitive medium-range skill (5–10 days) at only a fraction of the cost of their numerical counterparts. However, their forecasts tend to underperform in longer term and ensemble settings, producing blurred, poorly calibrated distributions due to spectral biases [44], [45] and a lack of sensitivity to initial-condition perturbations [46]. These shortcomings persist across network architecture, although transformer-based approaches show evidence of improved forecast skill proportional to the scale of larger parameter counts and smaller patch sizes [20], [39].

Advances in generative modeling, particularly with diffusion methods [47]–[51], similarly show favorable scaling laws on computer vision tasks [52]–[54]. Unlike deterministic methods, these models learn conditional probabilities, naturally quantify uncertainty, and are robust under incomplete distributional assumptions, making them well-suited to model the stochastic dynamics of the atmosphere. Recent diffusion-based weather models, such as GenCast [40] (used as a baseline in this work), better captures small-scale variability and produce ensembles approaching those of numerical systems, thereby addressing several limitations of deterministic models.

Nonetheless, GenCast faces important challenges: its multi-step solver becomes unstable beyond two weeks at 0.25° resolution, and its graph neural network backbone is less amenable to large-scale training compared to transformers. This raises a fundamental question of whether diffusion transformers, when scaled to billions of parameters, can achieve the stability and expressivity required for global, high-resolution weather and climate prediction. Our work addresses this gap, with a non-hierarchical, pixel-level Swin (window-based) transformer [35], [36] akin to [20], but parameterized by diffusion with architectural and parallelism innovations to improve stability, scalability, and long-range forecast skill.

B. Scaling Strategies

State-of-the-art training methods for parallelizing large transformer models involve combining data parallelism, pipeline parallelism, tensor-parallelism, and sequence parallelism (i.e., 4D parallelism). Among them, **Data Parallelism (DP)** (e.g. DDP, FSDP, ZeRO [55], [56]) is the simplest approach and requires the least communication (except for ZeRo3), but is limited by its inability to parallelize beyond the batch size, and naively increasing the batch size can negatively affect training efficiency and convergence. On the other hand, **Pipeline Parallelism (PP)** (e.g. GPipe, 1F1B [57], [58]) can parallelize a model with respect to the number of layers instead of batch size, but its efficiency is hindered by the bubble size (idle time) which ironically also requires a large batch size to minimize. **Tensor Parallelism (TP)** (e.g., [59], [60]) can parallelize a model through sharding the head-dimension of multi-head attention and is effective at sharding memory incurring from model states (parameters, optimizers, gradients), but is ineffective at sharding activation memory. In addition, TP incurs high communication overhead, limiting it only as an intra-node parallelism in practice. On the contrary to TP, **Sequence Parallelism (SP)** (e.g. Ulysses, Tensor-Sequence Parallelism, Ring-Attention [61]–[63]) are effective at sharding the activation memory but do not shard any model state memory.

Another approach is **Domain Parallelism** (e.g., PyTorch DTensor and NVIDIA PhysicsNeMo's ShardTensor) that shards inputs over devices across spatiotemporal dimensions and automatically issues the necessary halo exchanges. This primarily targets activation-heavy regimes and composes with DP for model-state sharding, with efficiency governed by operator locality and interconnect bandwidth/latency. However, performance degrades for non-local operations (e.g., global attention or normalization), where large communication overhead becomes unavoidable.

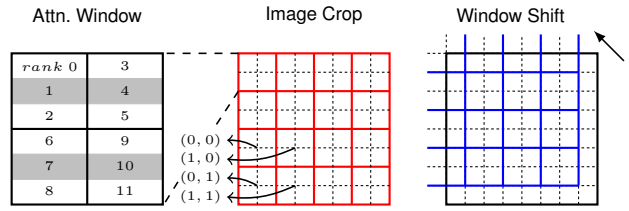
Previously, many of these parallelisms have been employed to train large transformer based weather forecasting models. Namely, ORBIT [33] uses a combination of TP and FSDP to scale their vision transformer model up to 113B, achieving a peak throughput of 1.6 EFLOPS. While such approaches alleviate memory pressure from model states, they remain ineffective at sharding activation memory, which is critical for high-resolution settings such as the global 0.25° data used in our work. Consequently, scaling transformer-based weather models in both parameter count and sequence length (driven by patch size) exposes the limitations of traditional 4D parallelism due to large communication overhead, limited parallelism degree, and other various inefficiencies (e.g., high number of synchronization points in Ring-attention). To address these challenges, we introduce a new dimension of parallelism—**Window Parallelism (WP)**—that is agnostic to domain and broadly applicable to window-based transformers. Furthermore, we propose a communication "merging" optimization that amortizes synchronization cost, significantly reducing the overhead of our method; see Section V-A.

V. Innovations Realized

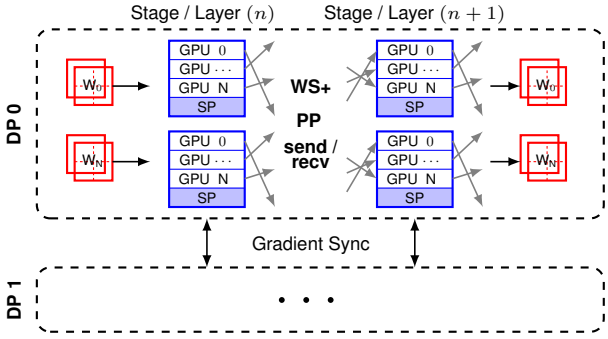
We highlight the key innovations in this work, starting with our novel parallelism in Section V-A, followed by the AERIS model architecture in Section V-B.

A. SWiPe: Sequence-Window Parallelism

Overview Our proposed Window Parallelism (WP) strategy exploits the inherent structure of Swin Transformers, which naturally partitions the computations across non-overlapping



(a) Window distribution. (left) window divided across SP ranks; (middle) attention grid with windows distributed in window parallel groups; and (right) shifted attention (in blue) for alternating layers.



(b) Parallelism stages and synchronization

Fig. 2: High-level parallelism architecture of SWiPe

spatial windows. In Swin, tokens are grouped into distinct attention windows, enabling localized attention within each window while maintaining a larger receptive field across layers by shifting these windows. This design preserves global context without requiring global attention in each layer. Unlike other localized attention mechanisms, such as sliding window attention or graph-based transformers, Swin's grid-based attention provides a clean and effective way to partition the input image across multiple ranks. Each rank handles a disjoint set of attention windows, enabling parallel computation without requiring halo exchange or duplication of overlapping data. This property makes Swin particularly amenable to WP, allowing AERIS to scale model parallelism efficiently by assigning separate windows to different GPUs while minimizing the communication overhead.

We adopt a hierarchical parallelism scheme for WP. The attention windows of a single image are first distributed across a 2D compute node grid of $A \times B$. The assigned window on each node is further partitioned across all the GPUs within that node using Ulysses's Sequence Parallelism (SP). The parallelism strategy is further enhanced by combining WP with other existing parallelism schemes such as Pipeline Parallelism (PP). Our resulting hybrid hierarchical parallelism scheme, termed **Sequence-Window Parallelism (SWiPe)**, significantly extends the scaling limit of large scale window-based model training. This is important for high-resolution global weather and climate modeling, where synoptic-scale systems and embedded mesoscale extremes arise from fine-grid interactions. SWiPe preserves pixel-level, shifted-window attention across the globe without the prohibitive cost of full global attention, while aligning compute with locality and minimizing cross-rank communication, and thereby enabling efficient, domain-guided scaling that sustains forecast fidelity.

Details Figure 2a illustrates an input image that is partitioned across the 12 GPU-tiles within an Aurora compute node (system details in Table I). The image is first divided into 4 spatial quadrants (a 2×2 grid) to accommodate the window shift. Accordingly, the tiles are logically arranged in a $2 \times 2 \times 3$ topology. The quadrants are one-to-one mapped to the 4 groups of tiles (each group contains 3 tiles). Each quadrant is sharded across 3 GPU tiles in that group. This method of data distribution primarily aims to balance computational load and minimize data movement during window shifting. In the context of SP compute and communication, however, the input tokens are flattened into a 1D sequence allowing efficiently gathering and redistribution across GPUs using `alltoall` communication both before and after the attention layer. This flattened representation simplifies the data exchange.

The windows are distributed across all the ranks in the WP group in a round-robin fashion in both X and Y directions as shown in the middle of Figure 2a. This distribution scheme allows batch processing of windows during window shifting leading to improved throughput; it also substantially simplifies the send/receive data movement pattern from one stage to another in PP while shifting windows. Each rank will send $1/SP$ of the window to the receiving rank in the next stage. No data redistribution is needed among the ranks in the next stage after receiving the window. If no WP is used, each rank will send an entire window and redistribution of data is needed among the ranks in the next stage to achieve the window shifting. A high-level illustration of how the parallelism strategies are integrated is shown in Figure 2b. To reduce communication overhead, SP groups are confined within individual nodes, enabling the frequent and bandwidth-intensive `alltoall` collectives to fully leverage the high-speed intra-node interconnect.

Communication overhead The primary communication overhead arises from intra-node `alltoall` communication related to SP or WP, internode `send/recv` communication due to PP, and the `allreduce` operation from data parallelism. The message size for both `alltoall` and `send/recv` communications is defined as $M = b \times s \times h/SP/WP$, where b is the batch size, s is sequence length, and h is the hidden dimension. Introducing WP reduces message size associated with `alltoall` and `send/recv` communications, while the overhead from gradient `allreduce` remains unchanged. Overall, enabling WP decreases the communication load per device. The `send/recv` communications can also overlap with computation, just like in regular PP, potentially hiding most inter-node communication.

Compared to input sharding with domain parallelism, which requires multiple re-sharding points for the Swin transformer, SWiPe avoids introducing additional communication or synchronization points. It is also fully compatible with efficient attention kernels, as tokens that attended to one another are implicitly gathered. In contrast, DTensor-based sharding demands explicit gathering of tokens for localized attention.

Activation memory When WP is enabled on top of both SP and PP, the activation memory is reduced by a factor of WP, thus reducing the need for activation checkpointing, which usually introduces additional recomputation of about $1/3$ of the total computation amount [64].

Data loading In WP, both the input and output are spatially partitioned so that each node loads only the data it processes. Only the first and last stages of the pipeline perform data loading and writing, respectively. With a WP group size of 16, each participating node handles just $1/16^{\text{th}}$ of the full image. This is particularly beneficial for high-resolution weather and climate datasets, where the size can be prohibitively large. Leveraging data formats that support efficient slicing (e.g., HDF5) allows each node to load only its required spatial windows, drastically reducing I/O per node and distributing the load evenly. Relative to configurations without WP, where full images are read redundantly or re-partitioned after loading, this strategy minimizes I/O overhead. In practice, the reduced and parallelized I/O is fully overlapped with the warm-up phase of the pipeline schedule, adding no training latency.

Mixed precision All compute-intensive operations, including matrix-matrix multiplications and the Flash Attention kernel, were performed using `BF16`, which is standard in large-scale deep learning workloads. To ensure numerical stability, components such as embeddings, primary gradients, model parameters, and gradient reductions were maintained in `FP32` (single-precision floating point). These operations account for only a small fraction of the total computational workload and were not measured separately. Empirically, we observed no significant throughput improvement when these components were cast to lower-precision formats, and thus retained them in `FP32` for consistency and stability.

We identify the following advantages of SWiPe, each of which are critical for large-scale climate model training:

- It **enhances** the degree of parallelism, enabling efficient training of models at larger scales.
- It **reduces** reliance on data parallelism for scaling, potentially improving model convergence behavior.
- It **decreases** communication overhead, potentially lead to better scaling efficiency.
- It **lowers** activation memory usage, potentially eliminating the need for activation checkpointing and reducing the overall memory footprint.

B. AERIS Model Architecture

The Swin Transformer [36], [65] is a prominent adaptation of the Vision Transformer [66] that has demonstrated strong performance across a variety of computer vision applications [20] despite lacking the global attention across all tokens. Instead, Swin shifts the receptive field of local attention windows every layer, effectively mimicking the global attention while obtaining the inductive bias of spatial-locality. Moreover, the lack of global attention affords greater flexibility of sequence length for attention, avoiding the costly quadratic compute complexity and enabling pixel-level patch-size.

The original Swin architecture uses hierarchical attention through shifted and downsampled windows for use in classification tasks. However, in this work, we map full resolution images in pixel-space, under a diffusion objective (see Section VI-B for details), while maintaining a non-hierarchical structure, known to be beneficial for spatiotemporal tasks [67]. Specifically, AERIS is designed to be used autoregressively, in both diffusion and data steps (Figure 1), taking as input a sample at time $i-1$ through T diffusion steps to estimate the residual of a sample at time i.

In addition to its non-hierarchical structure, AERIS introduces several modifications that deviate from the original Swin Transformer. While SwinV2 [36] aimed to enhance training stability and dynamics for larger-scale models, our implementation builds upon this goal by incorporating modern techniques such as pre RMSNorm [68] and SwiGLU [69], inspired by state-of-the-art large language models—particularly the Llama 3 series [26].

Figure 3 illustrates the conceptual flow of information and architecture, which is computationally optimized by our parallelism framework (Section V-A). We begin with adding a 2D sinusoidal positional encoding [70] to each channel of our input to serve as a proxy of locality given the spatial domain of our data. This input is embedded into an abstract representation through a learned linear layer before it passes through N Swin Layers. Each layer is composed of multiple transformer layers consisting of pre RMSNorm (in place of LayerNorm) and uses SwiGLU (in place of a singular linear layer) in the fully-connected layers. Prior to the multi-headed attention block, we "partition" the embedded images into 30×30 (for 6h model) or 60×60 (for 24h model) windows, that are "shifted" every other layer. These are then classically projected to queries, keys, and values before dot product attention, where the queries and keys are projected via axial frequency 2D rotary positional embeddings [71] (in place of relative positional biases). For the Attention we utilize the Ulysses sequence parallelism which does an `all-to-all` collective before and after the attention kernel. Following the last Swin Layer is a simple normalization and decoding block to project our embeddings back to pixel-space.

The time embedding for the diffusion timestep is projected through a shared linear layer, and then further broadcasted to

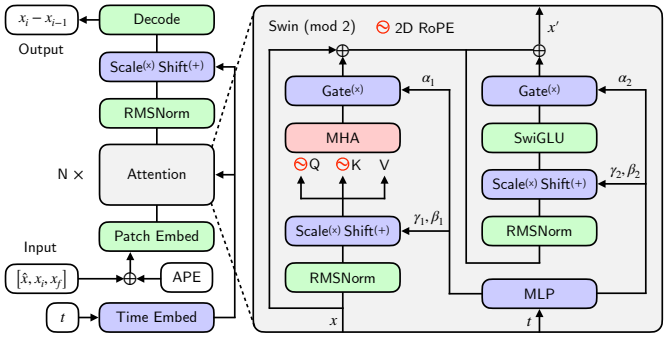


Fig. 3: Model architecture.

TABLE I: System configuration for performance evaluations.

	Aurora	**LUMI**
GPU	Intel Max 1550	AMD MI250X
GPUs (tiles) / node	6(12)	4(8)
GPU Memory (GB)	128	128
GPU Memory Technology	HBM2e	HBM2e
GPU Memory BW (TB/s)	2.0	3.2
Scale-Out Interconnect	HPE Slingshot 11	HPE Slingshot 11
NICs / node	8	4
Network BW / direction (GB/s)	200	100
Scale-up Interconnect	All-to-All X^e Links	Infinity Fabric
Scale-up BW / direction (GB/s)	28	50
Collective Communication Library	oneCCL	RCCL
Total nodes (GPU-tiles) scaled	10,080 (120,960)	1,008 (8,064)

all the layers, which contain another layer-specific linear layer. The output of this linear layer is used as the values α,β,γ for the adaptive layer norm [53], [72].

VI. How Performance was Measured

We describe the Aurora and LUMI systems used in the evaluation study, the application benchmark, the software stack and environment, and how we measure the performance.

A. System Details

Aurora is an exascale-class supercomputer hosted at the Argonne Leadership Computing Facility (ALCF). It is among the most powerful systems built [73], representing a major step forward in computational capabilities. We evaluate performance of training AERIS on Aurora. Table I describes the key architectural characteristics of this HPE Cray EX system with 10,624 nodes interconnected with HPE Slingshot 11 using a Dragonfly topology. Each node consists of two Intel Sapphire Rapids processors with a total of 1024 GB of system memory. Each node has six Intel Data Center Max 1550 GPUs, each with 128 GB memory. A GPU has two compute tiles. Each node has eight Slingshot-11 endpoints at 25 GB/s each for the interconnect network. The GPUs are configuration in a standard mode with 896 execution units. Each GPU is capable of achieving a peak of 45 TFLOPS in `FP32`, 229 TFLOPS in TF32, and 458 TFLOPS in `FP16` and `BF16`.

LUMI is a petascale supercomputer hosted by LUMI consortium, located at CSC datacenter in Kajaani, Finland. LUMI is a HPE Cray EX system, with the GPU-partition consisting of 2978 nodes. The system is interconnected with HPE Slingshot 11 using a Dragonfly topology. Each node has four Slingshot-11 endpoints at 25 GB/s each for the interconnect network. Each node has four AMD MI250X GPUs, each containing two graphic compute dies (GCD). Each MI250X GPU has 128GB of memory. Each AMD MI250X GPU is capable of achieving a peak of 95.7 TFLOPS in `FP32` and 383 TFLOPS in `FP16` and `BF16`.

B. Application Benchmark

Dataset We model the global evolution of the atmosphere, capturing medium-range and seasonal scales, by learning the state evolution $p(x_{i+1}|x_i)$ given four decades of ERA5 reanalysis data from the European Center for Medium-Range Weather Forecasting (ECMWF) [10], as provided by WeatherBench2 (WB2) [74], [75]. Data are on the native 0.25° (720×1440 pixel-grid with poles removed) spatial resolution with 6-hourly samples. We predict five surface-level variables: 2-meter temperature (T2m), 10-meter u- and v-components of wind (U10 and V10), mean sea-level pressure (MSLP), and sea surface temperature (SST); and five atmospheric variables: geopotential (Z), temperature (T), u- and v-components of wind (U and V), and specific humidity (Q), each at 13 pressure levels $\{50,\ 100,\ 150,\ 200,\ 250,\ 300,\ 400,\ 500,\ 600,\ 700,\ 850,\ 925,\ 1000\}$ hPa. To stabilize phase shift and simplify orographic representations, we also force the model with top-of-atmosphere solar radiation, surface geopotential, and land-sea mask as input. Data from 1979–2018 are used for training, 2019 for validation, and 2020 for testing to be consistent with WB2 evaluations. The dataset totals 16 TiB in HDF5 format.

Training To model the stochastic dynamics of the atmosphere, we use a conditional diffusion model parameterized by TrigFlow [50], which unifies EDM [51] and flow matching [76]–[79] under a simpler v-prediction estimate. Given clean samples $\boldsymbol{x}_0 \sim p_d$ from our data distribution, we construct noisy input samples by spherical interpolation with Gaussian noise, $\boldsymbol{x}_t = \cos(t)\boldsymbol{x}_0 + \sin(t)\boldsymbol{z}$, where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \sigma_d^2\boldsymbol{I})$ and $\sigma_d = 1$ denotes the data standard deviation. The interpolation parameter (or diffusion time step) is defined as $t = \arctan(e^{\tau}/\sigma_d) \in [0, \pi/2]$ with values drawn from a prior log-uniform distribution $\tau = (1-u)\ \log(\sigma_{\min}) + u\ \log(\sigma_{\max})$, $u \sim \mathcal{U}(0,1)$ with empirically chosen bounds of $\sigma_{\min} = 0.2$ and $\sigma_{\max} = 500$. This noise distribution is seen to better cover the heavy tailed distribution of target samples.

We parameterize our diffusion model as $\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) = \boldsymbol{F}_\theta(\boldsymbol{x}_t/\sigma_d, t)$ where $\boldsymbol{F}_\theta$ is our network with distributed parameters θ. Under TrigFlow, we estimate the target velocity $\boldsymbol{v}_t = \cos(t)\boldsymbol{z} - \sin(t)\boldsymbol{x}_0$ with the following objective

$$\ell^{\text{Diff}}(\theta) = \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{z}, t}\left[\left\|\sigma_d \boldsymbol{F}_\theta\left(\tfrac{\hat{\boldsymbol{x}}_t}{\sigma_d}, t\right) - \boldsymbol{v}_t\right\|_2^2\right], \tag{1}$$

where $\hat{\boldsymbol{x}}_t$ is a sample conditioned input. With a slight abuse of notation, let the sample at forecast time i be $\boldsymbol{x}_i$. Our model estimates the target residual $\boldsymbol{x}_0 = \boldsymbol{x}_i - \boldsymbol{x}_{i-1}$ such that $\hat{\boldsymbol{x}}_t = [\boldsymbol{x}_t, \boldsymbol{x}_{i-1}, \boldsymbol{x}_f]$ has initial condition $\boldsymbol{x}_{i-1}$ and forcings $\boldsymbol{x}_f$ at $i-1$ concatenated channel-wise. Data are z-score standardized with per-variable training statistics and predictions are unstandardized and added to $\boldsymbol{x}_{i-1}$ to recover the full-field state $\boldsymbol{x}_i$.

Considering the large number of prognostic variables, we modify the above objective to a more meaningful, physically weighted loss by leveraging a latitude- and pressure-weighting, as in prior works [23], [32], to account for the non-uniformity of the re-gridded sphere and to emphasize near-surface variables that are most important for weather forecasting. The functions $\alpha(s)$ and $\kappa(v)$ represent the latitude and variable weights for each variable $v \in \mathcal{V}$ in our objective

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\sum_{v\in\mathcal{V}} \kappa(v)\alpha(s)\ell_{v,s}^{\text{Diff}}(\theta), \tag{2}$$

where $\mathcal{S}$ is the set of spatial indices over all batches.

In our distributed setting with shared input windows loaded independently across ranks, we need to ensure noise levels are consistent per-sample. We achieve this by sharing a random seed for the state t in interpolant generation for all ranks in model parallel (i.e., SP, PP, and WP), but not data parallel. The Gaussian noise $\boldsymbol{z}$ is spatially uncorrelated, sharing no seed, and is truly random across ranks.

We train all models with AdamW ($\beta = [0.85, 0.9]$, $\epsilon = 1e-8$, and weight decay of $\lambda = 0.01$). The learning rate peaks at 5e-4 following a linear warmup over 50k images, then remains constant until decaying linearly to zero over the final 100k of 3m total images. We maintain an exponential moving average (EMA) of model parameters with a 100k image halflife, using only these weights during inference.

Inference The learned dynamics are governed by the corresponding probability flow ordinary differential equation (PF-ODE) $\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \sigma_d \boldsymbol{F}_\theta(\boldsymbol{x}_t/\sigma_d, t)$, which describes the evolution of a sample under the trained model. To generate a single forecast step, we numerically integrate this with 10 steps of a second-order DPMSolver++ 2S solver [80], modified using a log-uniform schedule for t to match the training prior. Under TrigFlow, we further introduce a trigonometric Langevin-like churn that temporarily injects noise to improve sample quality and ensemble spread. New ensemble members are generated by resampling noise $\boldsymbol{z}$ at $t = \pi/2$, with each output serving as the initial condition for the next autoregressive step.

Evaluation We evaluate AERIS using domain preferred diagnostics under medium-range and seasonal scales. In the former we compare forecasts of a subset of variables to baseline models used in WB2. See Section VII-B for details.

C. Software Environment

AERIS is implemented using PyTorch which natively supports Intel GPUs where several key kernels are optimized using Intel's extension for PyTorch (IPEX) [81] for the GPUs. A key enabler of distributed AI at scale on Aurora is the *oneAPI Collective Communications Library* (oneCCL) [82]. Built atop X^e-Links for intra-node communication and Slingshot for inter-node connectivity, oneCCL provides a highly optimized and flexible set of collective primitives, including `broadcast`, `reduce`, `allreduce`, `allgather`, and others—tailored for Intel's XPU architecture. oneCCL is integrated with all major deep learning frameworks such as PyTorch, TensorFlow, DeepSpeed, among others.

On LUMI we leverage the AMD software stack, which leverages ROCm backend for PyTorch, and the RCCL communication library for efficient collectives.

SWiPe leverages DeepSpeed Ulysses [61] from DeepSpeed as a building block. Pipeline parallelism, data parallelism, and a Zero1-like distributed optimizer were designed using custom-built modules we developed with PyTorch. This path enabled us to tightly integrate and optimize the various parallelisms involved in AERIS, allowing us to minimize the many data transfer overheads faced with existing implementations.

On Aurora, the SWiPe communication is overlapped with computation by offloading the communication to the CPU and then communicated with MPI-based libraries. This does not introduce additional latency, as the NICs on Aurora are connected to the CPU, requiring that all messages traverse it in any case. Unfortunately, we were not yet successful in implementing this overlap on other systems as we faced hangs and poor performance with MPI-based communication, and deadlocks with *CCL based communication. The deadlocks can happen due to the asynchronous p2p communication that is executed on a stream. This implies a modified pipeline schedule with coupled send and receive calls is required to take advantage of the stream-based deep learning-focused communication libraries.

We primarily used the `BF16` floating point format for most of our computational kernels. The embeddings, gradients, parameters, and gradient reduction were done in single precision floating point format (`FP32`) for numerical stability. As a note, `FP32` computations were just a very small fraction of the overall compute as most compute work is in the forward and backward pass with `BF16` activations and model weights. We used a PyTorch dataloader using h5py to load the datasets. Our implementation of AERIS is portable and can run on any system or supercomputer that supports PyTorch as demonstrated by our evaluations on two different accelerators from Intel and AMD.

D. Measurement Methodology

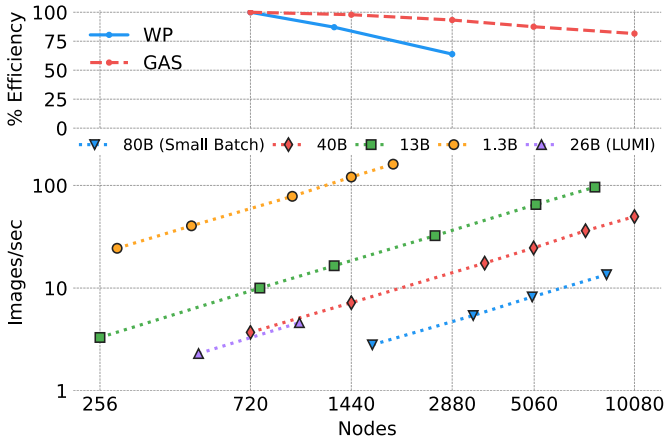
We develop an analytical model to estimate floating point operations, which takes into account various AERIS model parameters. This model builds on our prior work in accurately modeling transformers in Megatron-DeepSpeed at large scale [83]. The sustained FLOPS were determined by measuring the end-to-end time for the entire training loop. This included the I/O time needed to load and pre-process the data, communication time, including time spent in model-parallelism, gradient synchronization, among others. The peak FLOPS were determined by measuring only the time taken for the compute-intensive part of the training loop - executing the full pipeline schedule and accounts for all the communication time needed for SWiPe; it does not account for the time spent on gradient synchronization.

VII. Performance Results

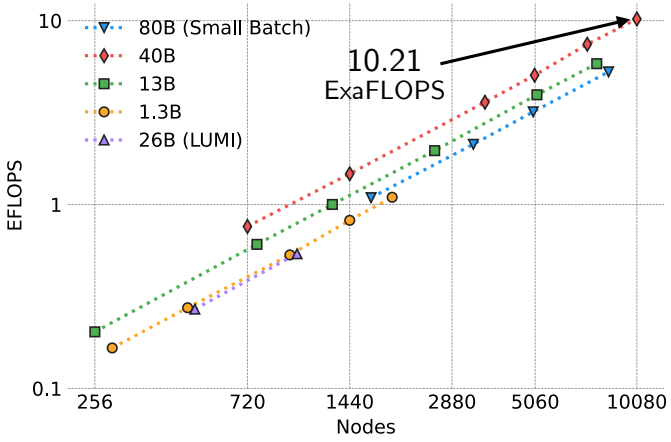
We assess the scalability of the AERIS model training on Aurora, followed by a comparative evaluation of its performance across a variety of forecasting tasks against state-of-the-art models.

A. Computational Performance

We evaluate the scaling performance of four AERIS model configurations as depicted in Table II. This represents models with increasing number of parameters ranging from 1.3 Billion to 80 Billion. The table includes the window parallelism (WP), pipeline parallelism (PP), and sequence parallelism (SP) used, as well as the model parameters - hidden dimensions,



(a) Strong scaling efficiency (top), images/sec weak scaling (bottom).



(b) Weak scaling performance—sustained FLOPS.

Fig. 4: Strong and weak scaling of training on Aurora (and LUMI) for multiple configurations with fixed (WP $\times$ PP $\times$ SP) in log-log scale. Strong scaling in the top for the 40B configuration is driven by changing either gradient accumulation steps (GAS) or window parallelism degree (WP); and the weak scaling is driven by increasing data parallelism, demonstrating consistent exascale performance and high utilization across large node counts.

TABLE II: AERIS model configurations.

Params	WP	PP	GAS	Dim	Heads	FFN	Nodes
1.3B	$4(2 \times 2)$	12	60	1536	12	9216	48
13B	$16(4 \times 4)$	16	48	4608	36	25600	256
40B	$16(4 \times 4)$	20	140	6144	48	40960	720
80B	$36(6 \times 6)$	26	52	7680	60	46080	1664
26B(L)	$36(6 \times 6)$	14	70	6144	48	32768	504

number of heads and feed-forward network size - for each. As mentioned previously in Section V, we target sequence parallelism within a node to use all 12 GPU tiles on an Aurora node and 8 GPU tiles on LUMI node to optimize the performance by restricting the communication within the node. The number of nodes needed to run a single model instance is given by $WP \times PP$. The LUMI 26B configuration differs from the Aurora 40B configurations mainly due to the reduced sequence parallelism, which is balanced by increased window parallelism. We also had to reduce the number of pipeline stages to fit the configuration to the standard queue of 1024 nodes. We weak scale these configurations using data parallelism to the node counts, parallelism and global batch size depicted in Table III.

Figure 4 depicts the weak scaling performance achieved in terms of throughput in images/sec as we increase the data parallelism to scale on Aurora. From the figure, we observe nearly linear scaling as we weak scale for all the model configurations on Aurora. In general, at similar node count, we achieve higher throughput with larger models in comparison to smaller models due to the increase computational requirements. The exception is the 79B configuration, as that one has significantly smaller batch size.

At the scale of 1440 nodes, we observe a 18$\times$ improvement in throughput for the 1.3B model over the 40B model, which is 31.5$\times$ larger. This can be attributed to the underlying Model FLOPS Utilization (MFU) in Table III due to the lower compute to communication ratio with respect to the 40B model at this scale.

Figure 4 also demonstrates the large-scale performance of training AERIS on Aurora achieving **sustained multi-exaflop training** in mixed-precision at scale. Notably, for the 40B AERIS configuration with a WP=36 and PP=20, we attain a **sustained performance of 10.21 ExaFLOPS**, and a **peak performance of 11.21 ExaFLOPS**, marking the highest throughput observed across all configurations. These results are enabled by a large degree of parallelism, combining domain-decomposition via window parallelism and sequence parallelism, and pipeline model parallelism, yielding an overall parallel degree of $36 \times 20 \times 12$. This parallel strategy allows AERIS training at a large scale without a large data parallelism degree, thus, leading to more stable training. At full scale of 10,080 nodes, the 40B model achieved a throughput of 50 samples per second. At this pace, it would take approximately 15 hours to complete training for 3M samples.

We also demonstrate an extreme case of scaling the model size and parallelism by presenting the 80B parameter configuration with WP=64 and smaller gradient accumulation, achieving **5.27 ExaFLOPS with a global batch size of just 260 samples at 8320 nodes**, resulting in unprecedented model and input parallelism requiring just one sample per 384 GPU tiles. In practice, the model parameters and the individual samples are sharded further, but more samples pass through a single GPU.

Our approach enables AERIS to scale efficiently across the full Aurora system. The compute throughput scaling shown in the bottom side of Figure 4 highlights the weak scaling efficiency as we increase the number of nodes along the data parallel dimension, and thus the batch size, under fixed model-parallel settings. In particular, the 40B (WP=36, PP=20)

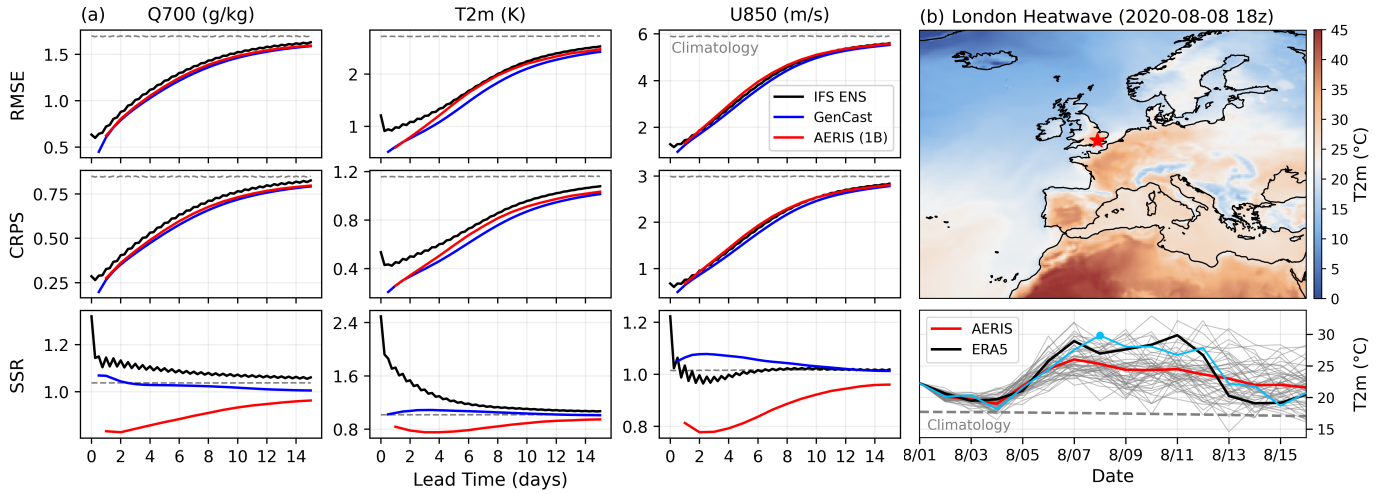


Fig. 5: Medium-range forecast skill. (a) Latitude-weighted root-mean-squared error (RMSE), continuous ranked probability score (CRPS), and spread/skill ratio (SSR) across 3 key variables for 50 ensembles of 155 initial conditions; and (b) accurate heat wave forecast over London, England (red star), initialized 2020-08-01T18z, showing 50 ensembles and the closest member.

configuration demonstrates good weak scaling efficiency of 95%, maintaining high throughput across 10,080 nodes of the Aurora production system. On both systems, we achieve a mean Model FLOPS Utilization (MFU) of over 30%, indicating a good overall utilization. The utilization is observed to be limited by typical reasons such as computational kernel efficiency, hard-to-hide communication overhead, and the idle time caused by the pipeline bubble. The performance achieved showcases the ability of our hybrid parallel architecture to support training at extreme scales, making AERIS a compelling foundation for future scientific AI workloads.

The top side of Figure 4 shows the strong scaling properties when adjusting the gradient accumulation steps (GAS) or window parallelism (WP) degree to get the same batch size (1960 for GAS scaling, 140 for WP scaling), and thus an equivalent training step across the range. The GAS scaling achieves strong scaling of 81.6%. The losses are mainly from the increasing pipeline bubble. The WP scaling goes through WP of 36, 64, and 144, with scaling efficiencies 100%, 87%, and 64%. The loss of efficiency is because the reduced GPU saturation due to less data per GPU, and relatively larger portion of time spent on gradient reduction, overall reducing GPU utilization. WP=144 is $4\times$ larger than WP=36, but only achieves $2.4\times$ speedup, resulting in strong scaling of 64% in the extreme case of 140 samples on 2880 nodes without data parallelism.

For achieving the performance, we also had to optimize our data loading and processing pipelines. To realize this, we isolate the input/output embedding layers and data I/O into separate stages resulting in a reduction of the pipeline bubble. Combining I/O and embeddings within the main pipeline stages would introduce additional latency that propagates as pipeline bubbles across all stages. By separating them out, we localize the impact to the first and last stages, resulting in slightly reduced GPU utilization at the edges but significantly better efficiency across the full pipeline. With this design, the number of pipeline stages is $PP = L + 2$ where L is the number of Swin layers.

TABLE III: Sustained and peak training throughput for AERIS on Aurora (and LUMI for 26B) across different model sizes: DP – Data Parallel degree, GBS – Global Batch Size, MFU – Model FLOPS Utilization, TF/T – TFLOPS per tile, EF(S) – sustained ExaFLOPS, and EF(P) – peak ExaFLOPS. The gap between peak and sustained ExaFLOPS is primarily due to the time spent on the optimizer step and gradient reduction. These components occur outside the pipelined forward-backward pass and thus contribute to the reduction in sustained throughput relative to the peak.

Config	Nodes	DP	GBS	TF/T	MFU(%)	EF(S)	EF(P)
1.3B	1920	40	2400	47.6	21.6	1.1	1.2
13B	7680	30	1440	63.3	28.8	5.8	6.4
40B	**10080**	14	1960	**84.4**	**38.4**	**10.21**	**11.21**
80B	8320	5	**260**	52.8	24	5.27	6.1
26B(L)	1008	2	140	66.5	34.8	0.54	0.62

B. Domain Results

A core innovation of AERIS is in being an end-to-end generative model for short-term weather forecasting to seasonal scales. We demonstrate this by evaluating skill on medium-range weather forecasting (lead-times of 1–14 days), performance on notable features that are predictable on the subseasonal-to-seasonal (S2S) time scales, e.g., sea surface temperature (SST) and Madden–Julian oscillation (MJO), and lastly explore performance on extreme events including record breaking heatwaves and tropical cyclones.

Medium-range forecasting For medium-range weather forecasting, we assess the ability of our model to produce well calibrated and skillful probabilistic forecasts. We compare to two well-established models: (1) GenCast [40], a diffusion-

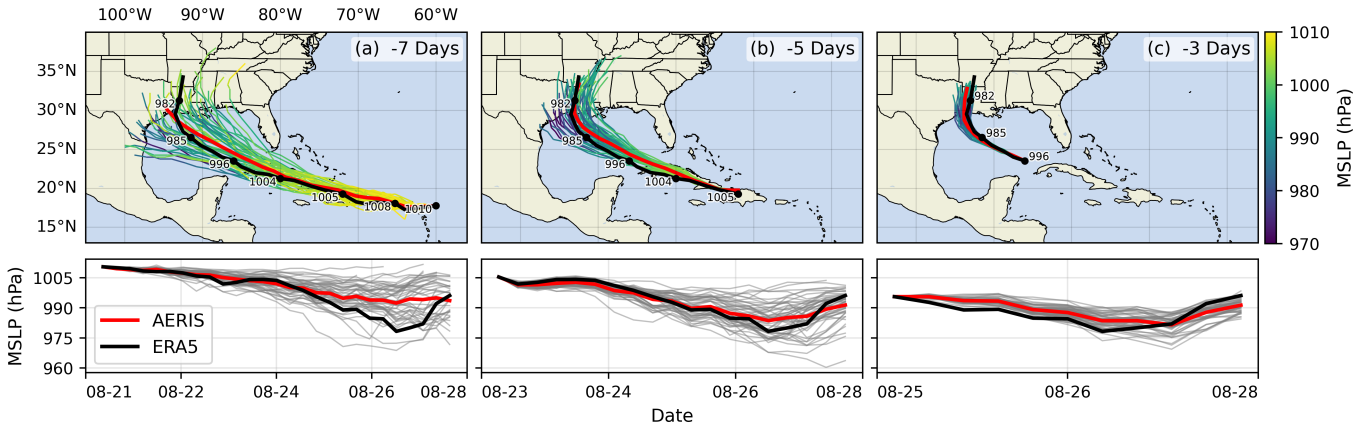


Fig. 6: Hurricane Laura tracks (top) and intensity (bottom). Initialized 7(a), 5(b), and 3(c) days prior to 2020-08-28T00z.

based ensemble system from Google DeepMind; and (2) the IFS ensemble (IFS ENS) a state-of-the-art, numerical-based ensemble system from the European Center for Medium Range Weather Forecasting (ECMWF) [84]. Evaluations are conducted on held-out, test data for 2020, where we showcase ensemble mean latitude-weighted root-mean-squared error (RMSE), a probabilistic metric called Continuous Ranked Probability Score (CRPS), and the spread/skill ratio (SSR) for a subset of variables over 14-day rollouts (on 24h intervals).

Figure 5a illustrates our results for each of these variables. We find AERIS is on-par or outperforms the IFS ENS for ensemble mean RMSE and CPRS, while performing competitively with GenCast especially during the 1–3 day and 10+ day time frames. In fact, for specific humidity at 700 hPa, AERIS is nearly identical to GenCast in terms of forecast performance. This is despite AERIS being an under-dispersive ensemble system (SSR < 1), suggesting the potential for improvement in model performance by increasing the diversity and spread of individual ensemble members (see Section VII-C). This result of using only diffusion for ensemble members leading to under-dispersion is not unique as this was also found to be true in GenCast.

Subseasonal-to-seasonal (S2S) To assess trends that extend beyond that of our medium-range weather forecasts (beyond 14-days) and evaluate the stability of our model, we made 3,000 forecasts (60 initial conditions each with 50 ensembles) out to 90 days. AERIS was found to be stable during these 90-day forecasts with realistic atmospheric states (Figure 7b), and correct power-spectra even at the smallest scales (not shown). We demonstrate for the first time, the ability of a generative, high-resolution (native ERA5) diffusion model to produce skillful forecasts on the S2S timescales with realistic evolutions of the Earth system (atmosphere + ocean).

We showcase this by evaluating the model's ability to predict the El Niño Southern Oscillation (ENSO) state by demonstrating skillful predictions of daily Niño 3.4 indices out to at least 90 days during the winter and spring months in 2020 (Figure 7a). We find the ensemble mean mirrors the true evolution of the equatorial Pacific SSTs for 90 days with realistic spread along the spring barrier. We also find on a global scale, realistic error growth and forecast skill on-par with numerical-based coupled systems (not shown). Lastly, we look at a brief study of how convectively coupled equatorial waves propagate through longitude and time. More specifically, in Figure 7c, we qualitatively compare Hovmöller diagrams [85] of a single ensemble, seeing skill to at least 3 weeks and shows realistic variability out to 90 days. We emphasize that to our knowledge, this is the first diffusion-based model for atmosphere and ocean prediction that demonstrates skill on the S2S timescales at $0.25°$.

Extreme Events The correct predictions of extreme events (e.g., tropical cyclones and heatwaves), are vital for weather forecasting due to their high socioeconomic impacts and their potential to cause large loss of life. With its probabilistic formulation, we find that AERIS is able to predict extreme events with exceptional skill. To demonstrate this, we examine two case studies: (1) correctly predicting Hurricane Laura's track and intensity up to 7 days before landfall; and (2) the prediction of the record-breaking heatwave in Europe during early August of 2020 (Figure 5c and 6).

Hurricane Laura made landfall near Cameron, Louisiana, on August 27, 2020, causing $19 billion in damage and 47 direct deaths [86]. This record-breaking hurricane is well forecasted by AERIS with minimal track errors even with a lead-time of 7 days. By day 5, the ensemble mean correctly captures the eventual northward track and has landfall in Louisiana, nearly identical to the ERA5 track. Another aspect of Hurricane Laura is the rapid intensification in the Gulf of Mexico. Our model accurately predicts this intensification period with a lead time of 5 days, a phenomena that is not typically well captured by global numerical models.

During August 2020, an intense heat wave impacted Europe, with the United Kingdom being the most affected. We showAERIS successfully identifying this heatwave with a lead time of more than a week (Figure 5b). Specifically, all ensemble members capture the sharp rise in temperatures, followed by the return to climatology, with the ensemble mean closely following ERA5. These results signify the ability of AERIS in positioning itself as a reliable and skillful model, even at the tail of distributions, where extreme events occur.

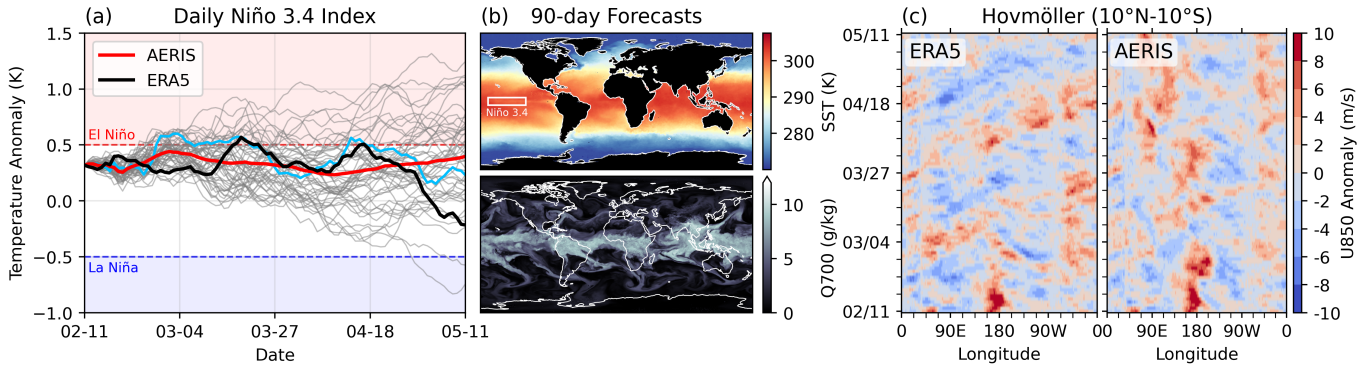


Fig. 7: Seasonal forecast stability. (a) Spring barrier under El Niño with realistic ensemble spread in the ocean; (b) qualitatively sharp fields of SST and Q700 predicted 90 days in the future from the closest ensemble member to ERA5 in (a); and (c) stable Hovmöller diagrams of U850 anomalies (climatology removed; m/s), averaged between 10°N and 10°S for a 90-day rollout.

C. Limitations and Future Works

Despite strong results, AERIS faces several limitations that we aim to address. Our science findings are based on our 1.3B parameter model, and while we are currently pursuing highly ambitious trainings (e.g., 13B with 1×1 patch-size), these experiments require extensive compute resources to converge—on the order of one week of run-time on exascale machines ($\sim$1.5M node hours)—which poses practical challenges. Beyond compute constraints, current medium-range skill (Figure 5) remains overconfident. Improving the spread/skill ratio through initial condition perturbations and tuning our stochastic churning schedule under TrigFlow may improve ensemble spread without hurting skill. Our diffusion parameterization also allows for consistency distillation [50], which allows us to compress the model size and reduce inference to a single step, thereby lowering computational cost by orders of magnitude for generating new forecasts. As a consistency model, AERIS could benefit from multi-step finetuning [87], which may yield measurable improvements to forecast skill [88].

AERIS is data-driven and trained on historical reanalysis, not governed by numerical equations; thus, non-physical artifacts or unrealistic dynamics can arise. Additionally, to make our model operational under current conditions, finetuning on IFS HRES 0th frame data would be required. At the same time, the methods presented herein are generally adaptable and can scale to higher-resolution inputs and generalize to alternative datasets. Ongoing work explores such datasets and architectural changes to extend forecast skill to even longer time horizons. Finally, SWiPe itself can be improved by reducing the bubble size of pipeline parallelism, as GPUs currently idle when waiting for data from another pipeline stage under 1F1B; adopting zero-bubble pipeline parallelism [89] offers a promising solution.

VIII. Implications

Weather forecasting and climate modeling have been a grand challenge in science and computing for the past 50 years. Advances in computing have helped extend weather forecast skill from 3 to nearly 8 days and improved climate model resolution from 500 km in the 1980s to about 25 km today [1]. While this progress is impressive, this has come at a cost of expensive model simulations. Even with the latest computing capabilities, it remains a challenge to produce both large ensembles and achieve high-spatial resolution for both weather and seasonal scales.

Today, the state-of-the-art approaches for modeling weather are inspired by the tremendous progress in artificial intelligence (AI) approaches and building observational data-based models. These models have shown tremendous progress over the past few years, and with forecast capability on short- and medium-range weather forecasting either matching or beating those produced by conventional numerical weather forecast models operated by national and international facilities.

We demonstrate a significant advancement in AI weather and climate modeling with AERIS by efficient scaling of window-based transformer models. We have performed global medium-range forecasts with performance competitive with GenCast and surpassing the IFS ENS model, with longer, 90-day rollouts showing our ability to learn atmospheric dynamics on seasonal scales without collapsing, becoming the first diffusion-based model that can work across forecast scales from 6 hours all the way to 3 months with remarkably accurate out of distribution predictions of extreme events.

On the system architecture side, AERIS exploits an optimized hybrid parallelism strategy—integrating our window-based transformer parallelism with pipeline, sequence, and data parallelisms—to achieve exceptional computational throughput and scaling performance. Our proposed parallelisms in SWiPe accelerates training of our AERIS models and enables scaling to large node counts at lower global batch size. We are able to scale to 10,080 nodes (120,960 GPU tiles) and achieve a maximum sustained performance of 10.21 ExaFLOPS in mixed precision, setting a high bar for model training performance. These innovations significantly advance the feasibility of training foundation climate models on the highest resolution data on exascale systems.

ACKNOWLEDGMENTS

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357. We acknowledge CSC – IT Center for Science, Finland, for computational resources on LUMI and thank Pekka Manninen at CSC for timely support. We thank Servesh Muralidharan from Argonne Leadership Computing Facility (ALCF) for help reducing job startup time at launch and help with debugging computational performance, Nithin Chalapathi from UC Berkeley for running experimental GenCast experiments for baselines, and Joseph Insley from Argonne National Laboratory for supporting preliminary experimental visualization efforts.

REFERENCES

[1] P. Bauer, A. Thorpe, and G. Brunet, "The quiet revolution of numerical weather prediction," *Nature*, vol. 525, no. 7567, pp. 47–55, 2015.

[2] P. Lynch, "The origins of computer weather prediction and climate modeling," *Journal of computational physics*, vol. 227, no. 7, pp. 3431–3444, 2008.

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

[4] D. J. Stensrud, *Parameterization Schemes: Keys to Understanding Numerical Weather Prediction Models*. Cambridge University Press, 2009.

[5] L. Magnusson and E. Källén, "Factors influencing skill improvements in the ecmwf forecasting system," *Monthly Weather Review*, vol. 141, no. 9, pp. 3142–3153, 2013.

[6] P. D. Dueben and P. Bauer, "Challenges and design choices for global weather and climate models based on machine learning," *Geoscientific Model Development*, vol. 11, no. 10, pp. 3999–4009, 2018. [Online]. Available: https://gmd.copernicus.org/articles/11/3999/2018/

[7] S. Scher, "Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning," *Geophysical Research Letters*, vol. 45, no. 22, pp. 12–616, 2018.

[8] J. A. Weyn, D. R. Durran, and R. 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*, vol. 11, no. 8, pp. 2680–2693, 2019.

[9] H. Hersbach, B. Bell, P. Berrisford, G. Biavati, A. Horányi, J. Muñoz Sabater, J. Nicolas, C. Peubey, R. Radu, I. Rozum *et al.*, "Era5 hourly data on single levels from 1979 to present," *Copernicus Climate Change Service (C3S) Climate Data Store (CDS)*, vol. 10, 2018.

[10] H. Hersbach, B. Bell, P. Berrisford, S. Hirahara, A. Horányi, J. Muñoz-Sabater, J. Nicolas, C. Peubey, R. Radu, D. Schepers *et al.*, "The era5 global reanalysis," *Quarterly Journal of the Royal Meteorological Society*, vol. 146, no. 730, pp. 1999–2049, 2020.

[11] S. Rasp, P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, and N. Thuerey, "Weatherbench: a benchmark data set for data-driven weather forecasting," *Journal of Advances in Modeling Earth Systems*, vol. 12, no. 11, p. e2020MS002203, 2020.

[12] 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," *arXiv preprint arXiv:2308.15560*, 2023.

[13] S. Rasp and N. Thuerey, "Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for weatherbench," *Journal of Advances in Modeling Earth Systems*, vol. 13, no. 2, p. e2020MS002405, 2021.

[14] T. Arcomano, I. Szunyogh, J. Pathak, A. Wikner, B. R. Hunt, and E. Ott, "A machine learning-based global atmospheric forecast model," *Geophysical Research Letters*, vol. 47, p. e2020GL087776, 2020.

[15] J. A. Weyn, D. R. Durran, and R. Caruana, "Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere," *Journal of Advances in Modeling Earth Systems*, vol. 12, no. 9, p. e2020MS002109, 2020.

[16] J. A. Weyn, D. R. Durran, R. Caruana, and N. Cresswell-Clay, "Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models," *Journal of Advances in Modeling Earth Systems*, vol. 13, no. 7, p. e2021MS002502, 2021.

[17] M. C. Clare, O. Jamil, and C. J. Morcrette, "Combining distribution-based neural networks to predict weather forecast probabilities," *Quarterly Journal of the Royal Meteorological Society*, vol. 147, no. 741, pp. 4337–4357, 2021.

[18] R. Keisler, "Forecasting global weather with graph neural networks," *arXiv preprint arXiv:2202.07575*, 2022.

[19] J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli *et al.*, "Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators," *arXiv preprint arXiv:2202.11214*, 2022.

[20] J. D. Willard, P. Harrington, S. Subramanian, A. Mahesh, T. A. O'Brien, and W. D. Collins, "Analyzing and exploring training recipes for large-scale transformer-based weather prediction," 2024. [Online]. Available: https://arxiv.org/abs/2404.19630

[21] T. Nguyen, J. Brandstetter, A. Kapoor, J. K. Gupta, and A. Grover, "ClimaX: A foundation model for weather and climate," *arXiv preprint arXiv:2301.10343*, 2023.

[22] K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, and Q. Tian, "Accurate medium-range global weather forecasting with 3D neural networks," *Nature*, vol. 619, no. 7970, pp. 533–538, 2023.

[23] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, A. Pritzel, S. Ravuri, T. Ewalds, F. Alet, Z. Eaton-Rosen *et al.*, "Graphcast: Learning skillful medium-range global weather forecasting," *arXiv preprint arXiv:2212.12794*, 2022.

[24] K. Chen, T. Han, J. Gong, L. Bai, F. Ling, J.-J. Luo, X. Chen, L. Ma, T. Zhang, R. Su, Y. Ci, B. Li, X. Yang, and W. Ouyang, "FengWu: Pushing the skillful global medium-range weather forecast beyond 10 days lead," *arXiv preprint arXiv:2304.02948*, 2023.

[25] L. Chen, X. Zhong, F. Zhang, Y. Cheng, Y. Xu, Y. Qi, and H. Li, "FuXi: A cascade machine learning forecasting system for 15-day global weather forecast," *arXiv preprint arXiv:2306.12873*, 2023.

[26] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[27] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang *et al.*, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.

[28] X. Bi, D. Chen, G. Chen, S. Chen, D. Dai, C. Deng, H. Ding, K. Dong, Q. Du, Z. Fu *et al.*, "Deepseek llm: Scaling open-source language models with longtermism," *arXiv preprint arXiv:2401.02954*, 2024.

[29] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[30] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 104–12 113.

[31] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. P. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin *et al.*, "Scaling vision transformers to 22 billion parameters," in *International Conference on Machine Learning*. PMLR, 2023, pp. 7480–7512.

[32] T. Nguyen, R. Shah, H. Bansal, T. Arcomano, R. Maulik, R. Kotamarthi, I. Foster, S. Madireddy, and A. Grover, "Scaling transformer neural networks for skillful and reliable medium-range weather forecasting," *Advances in Neural Information Processing Systems*, vol. 37, pp. 68 740–68 771, 2024.

[33] X. Wang, S. Liu, A. Tsaris, J.-Y. Choi, A. Aji, M. Fan, W. Zhang, J. Yin, M. Ashfaq, D. Lu, and P. Balaprakash, "Orbit: Oak ridge base foundation model for earth system predictability," 2024. [Online]. Available: https://arxiv.org/abs/2404.14712

[34] T. Han, S. Guo, F. Ling, K. Chen, J. Gong, J. Luo, J. Gu, K. Dai, W. Ouyang, and L. Bai, "Fengwu-ghr: Learning the kilometer-scale medium-range global weather forecasting," *arXiv preprint arXiv:2402.00059*, 2024.

[35] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[36] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong *et al.*, "Swin transformer v2: Scaling up capacity and

resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 12 009–12 019.

[37] 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," 2023. [Online]. Available: https://arxiv.org/abs/2212.12794

[38] K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, and Q. Tian, "Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast," *arXiv preprint arXiv:2211.02556*, 2022.

[39] T. Nguyen, R. Shah, H. Bansal, T. Arcomano, R. Maulik, V. Kotamarthi, I. Foster, S. Madireddy, and A. Grover, "Scaling transformer neural networks for skillful and reliable medium-range weather forecasting," 2024. [Online]. Available: https://arxiv.org/abs/2312.03876

[40] 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, "Gencast: Diffusion-based ensemble forecasting for medium-range weather," 2024. [Online]. Available: https://arxiv.org/abs/2312.15796

[41] G. Couairon, R. Singh, A. Charantonis, C. Lessig, and C. Monteleoni, "Archesweather & archesweathergen: a deterministic and generative model for efficient ml weather forecasting," *arXiv preprint arXiv:2412.12971*, 2024.

[42] J. Stock, J. Pathak, Y. Cohen, M. Pritchard, P. Garg, D. Durran, M. Mardani, and N. Brenowitz, "Diffobs: Generative diffusion for global forecasting of satellite observations," *arXiv preprint arXiv:2404.06517*, 2024.

[43] M. Mardani, N. Brenowitz, Y. Cohen, J. Pathak, C.-Y. Chen, C.-C. Liu, A. Vahdat, M. A. Nabian, T. Ge, A. Subramaniam *et al.*, "Residual corrective diffusion modeling for km-scale atmospheric downscaling," *Communications Earth & Environment*, vol. 6, no. 1, p. 124, 2025.

[44] I. Ebert-Uphoff, L. Ver Hoef, J. S. Schreck, J. Stock, M. J. Molina, A. McGovern, M. Yu, B. Petzke, K. Hilburn, D. M. Hall *et al.*, "Measuring sharpness of ai-generated meteorological imagery," *Artificial Intelligence for the Earth Systems*, 2025.

[45] A. Chattopadhyay, Y. Q. Sun, and P. Hassanzadeh, "Challenges of learning multi-scale dynamics with ai weather models: Implications for stability and one solution," 2024. [Online]. Available: https://arxiv.org/abs/2304.07029

[46] T. Selz and G. C. Craig, "Can artificial intelligence-based weather prediction models simulate the butterfly effect?" *Geophysical Research Letters*, vol. 50, no. 20, p. e2023GL105747, 2023, e2023GL105747 2023GL105747. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2023GL105747

[47] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. pmlr, 2015, pp. 2256–2265.

[48] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[49] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[50] C. Lu and Y. Song, "Simplifying, stabilizing and scaling continuous-time consistency models," *arXiv preprint arXiv:2410.11081*, 2024.

[51] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," *Advances in neural information processing systems*, vol. 35, pp. 26 565–26 577, 2022.

[52] H. Li, Y. Zou, Y. Wang, O. Majumder, Y. Xie, R. Manmatha, A. Swaminathan, Z. Tu, S. Ermon, and S. Soatto, "On the scalability of diffusion-based text-to-image generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 9400–9409.

[53] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4195–4205.

[54] Z. Liang, H. He, C. Yang, and B. Dai, "Scaling laws for diffusion transformers," *arXiv preprint arXiv:2410.08184*, 2024.

[55] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," 2020. [Online]. Available: https://arxiv.org/abs/1910.02054

[56] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, A. Desmaison, C. Balioglu, P. Damania, B. Nguyen, G. Chauhan, Y. Hao, A. Mathews, and S. Li, "Pytorch fsdp: Experiences on scaling fully sharded data parallel," 2023. [Online]. Available: https://arxiv.org/abs/2304.11277

[57] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. X. Chen, D. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," 2019. [Online]. Available: https://arxiv.org/abs/1811.06965

[58] J. Reed, P. Belevich, K. Wen, H. Huang, and W. Constable, "Pippy: Pipeline parallelism for pytorch," https://github.com/pytorch/PiPPy, 2022.

[59] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[60] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 3505–3506.

[61] S. A. Jacobs, M. Tanaka, C. Zhang, M. Zhang, S. L. Song, S. Rajbhandari, and Y. He, "Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models," 2023. [Online]. Available: https://arxiv.org/abs/2309.14509

[62] V. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro, "Reducing activation recomputation in large transformer models," 2022. [Online]. Available: https://arxiv.org/abs/2205.05198

[63] H. Liu, M. Zaharia, and P. Abbeel, "Ring attention with blockwise transformers for near-infinite context," 2023. [Online]. Available: https://arxiv.org/abs/2310.01889

[64] V. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro, "Reducing activation recomputation in large transformer models," 2022. [Online]. Available: https://arxiv.org/abs/2205.05198

[65] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021. [Online]. Available: https://arxiv.org/abs/2103.14030

[66] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[67] Z. Gao, X. Shi, H. Wang, Y. Zhu, Y. Wang, M. Li, and D.-Y. Yeung, "Earthformer: Exploring space-time transformers for earth system forecasting," 2023. [Online]. Available: https://arxiv.org/abs/2207.05833

[68] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[69] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[70] Z. Wang and J.-C. Liu, "Translating math formula images to latex sequences using deep neural networks with sequence-level training," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 24, no. 1, pp. 63–75, 2021.

[71] B. Heo, S. Park, D. Han, and S. Yun, "Rotary position embedding for vision transformer," in *European Conference on Computer Vision*. Springer, 2024, pp. 289–305.

[72] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[73] TOP500 list. [Online]. Available: https://www.top500.org/lists/top500/

[74] 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," 2024. [Online]. Available: https://arxiv.org/abs/2308.15560

[75] Weatherbench2 repository. [Online]. Available: https://weatherbench2.readthedocs.io/en/latest/

[76] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[77] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," *arXiv preprint arXiv:2209.03003*, 2022.
[78] M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden, "Stochastic interpolants: A unifying framework for flows and diffusions," *arXiv preprint arXiv:2303.08797*, 2023.
[79] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio, "Improving and generalizing flow-based generative models with minibatch optimal transport," *arXiv preprint arXiv:2302.00482*, 2023.
[80] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models," *Machine Intelligence Research*, pp. 1–22, 2025.
[81] Ipex code repository. [Online]. Available: https://github.com/intel/intel-extension-for-pytorch
[82] oneccl code repository. [Online]. Available: https://github.com/uxlfoundation/oneCCL
[83] G. Dharuman, K. Hippe, A. Brace, S. Foreman, V. Hatanpää, V. K. Sastry, H. Zheng, L. Ward, S. Muralidharan, A. Vasan *et al.*, "Mprot-dpo: Breaking the exaflops barrier for multimodal protein design workflows with direct preference optimization," in *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2024, pp. 1–13.
[84] S. Lang, M. Rodwell, and D. Schepers, "Ifs upgrade brings many improvements and unifies medium-range resolutions," *ECMWF Newsletter*, vol. 176, pp. 21–28, 2023.
[85] E. Hovmöller, "The trough-and-ridge diagram," *Tellus*, vol. 1, no. 2, pp. 62–66, 1949.
[86] R. Pasch, R. Berg, D. Roberts, and P. Papin, "Hurricane laura (al 132020)," *National Hurricane center tropical cyclone report*, 2021.
[87] J. Stock, T. Arcomano, and R. Kotamarthi, "Swift: An autoregressive consistency model for efficient weather forecasting," in *NeurIPS 2025 Workshop on Tackling Climate Change with Machine Learning*, 2025.
[88] S. A. Siddiqui, J. Kossaifi, B. Bonev, C. Choy, J. Kautz, D. Krueger, and K. Azizzadenesheli, "Exploring the design space of deep-learning-based weather forecasting systems," *arXiv preprint arXiv:2410.07472*, 2024.
[89] P. Qi, X. Wan, G. Huang, and M. Lin, "Zero bubble (almost) pipeline parallelism," in *The Twelfth International Conference on Learning Representations*, 2024.