Less Tokens, Better Forecasts: Sparse Residual Routing for Efficient Weather Prediction

Janet Wang[1], Yunbei Zhang[1], Lin Zhao[2], Xi Xiao[3], Jihun Hamm[1], and Xiao Wang[4]

[1] Tulane University
[2] Northeastern University
[3] University of Alabama at Birmingham
[4] Oak Ridge National Laboratory

Abstract. Existing ViT-based weather forecasting models apply uniform computation across all spatial tokens, even though nearby atmospheric grid points often contain similar values and large regions evolve smoothly over time. This makes much of the intermediate per-token computation redundant. Standard token-efficiency methods, such as pruning or merging, reduce cost by removing or fusing tokens. However, weather forecasting is a spatiotemporal dense prediction problem in which a history of atmospheric states must be mapped to future values on the original latitude-longitude grid. Thus, every grid cell must retain a physically meaningful representation, especially under autoregressive rollout. We introduce **Sparse-Reslim**, a parameter-free plug-in routing module that makes sparse token processing compatible with this fixed-grid requirement. Sparse-Reslim routes only 25% of spatial tokens through the expensive middle transformer blocks and treats those blocks as residual updates: it computes the change produced for the routed tokens and scatters only this delta back to the full sequence. Unselected tokens keep their pre-routing representations exactly, so no grid cell is dropped or replaced by a mask token, and no fusion layer or additional parameters are introduced. Across ERA5 resolutions up to the operational 0.25° standard and two model families, a deterministic Transformer and a diffusion model, Sparse-Reslim improves forecast accuracy on every evaluated variable while substantially reducing cost: training is about 2.5× faster in the main settings and reaches 3.18× speedup at 0.25°, with over 2.2× lower peak memory. A controlled decomposition shows that the accuracy gain comes primarily from sparse routing itself, while random token selection provides an additional regularization benefit without selector overhead. Code is available at https://github.com/janet-sw/Sparse-Reslim.

Keywords: Weather Forecast · Vision Transformer · Efficient Training

1 Introduction

Data-driven weather forecasting has reached a turning point [7, 26, 29]. Models trained on reanalysis data now match or exceed the accuracy of operational

numerical weather prediction (NWP) systems that took decades to develop, and they do so orders of magnitude faster at inference time [1, 2, 13]. These models operate on high-dimensional gridded atmospheric fields (ERA5 at 0.25° comprises over one million grid points per variable) and must produce spatially dense outputs where every grid cell carries a physically meaningful forecast. As the community pushes toward higher resolutions and larger model capacities, training cost becomes the primary bottleneck, because a finer grid produces more spatial tokens and self-attention scales quadratically with token count, while larger models add further compute and activation memory per token.

A striking pattern across the leading weather models (Pangu-Weather [1], ClimaX [17], Stormer [19], FuXi [5], Aurora [2], and their generative counterparts GenCast [21] and CorrDiff [16]) is that they all rely on the Vision Transformer (ViT) or its variants. Self-attention treats every spatial token identically, spending the same $O(L^2D)$ computation on a token over calm open ocean as on one at the core of an extratropical cyclone. Yet atmospheric fields are highly autocorrelated: large regions of the globe carry smooth, slowly varying conditions whose intermediate representations are largely redundant. This suggests that substantial computation can be saved by processing only a representative subset of tokens in the most expensive layers.

Token pruning [12, 14, 23, 30], token merging [3], and window-based attention [15] have been widely explored for ViT efficiency in the vision community. More recently, Sprint [20] proposed a dense-sparse-dense block schedule for diffusion transformers that routes only a random subset of tokens through the middle blocks, achieving large training speedups for image generation. Directly applying these methods to weather forecasting is nontrivial because weather prediction is a spatiotemporal dense prediction problem: the model conditions on past atmospheric states and must produce a physically meaningful future value at every grid point, often for autoregressive rollout. Token pruning and merging can remove or alter spatial positions, while Sprint replaces skipped tokens with learned mask embeddings and later fuses them through a linear projection. This is acceptable for static image generation, but in weather forecasting it creates a mismatch: late blocks operate on a mixture of true atmospheric representations and synthetic placeholder tokens at fixed grid positions, without explicitly preserving the temporal conditioning structure of the forecast problem.

In this work, we propose **Sparse-Reslim**, a parameter-free plug-in routing module for ViT-based weather forecasters that makes sparse token processing compatible with dense prediction. Sparse-Reslim partitions the transformer blocks into dense early, sparse middle, and dense late stages. At the entrance to the sparse middle stage, it randomly selects a fraction r of the L spatial tokens, yielding $K = \lfloor rL \rfloor$ routed tokens, and processes only this subset through the expensive middle blocks. Crucially, instead of replacing the full sequence with processed tokens and placeholders, Sparse-Reslim computes the *residual delta* produced by the sparse middle blocks and scatters only this delta back to the original token positions. Unselected tokens receive zero delta and therefore retain their pre-routing representations exactly, ensuring that all L spatial token

positions remain valid throughout the model. The module requires no mask embeddings, fusion layers, or additional learnable parameters, and can be enabled in an existing ViT block loop with a single configuration flag.

We validate Sparse-Reslim on ERA5 [10] at multiple resolutions and on two architecture families: (1) a deterministic Res-Slim-ViT adapted from the ORBIT-2 exascale system [27], and (2) a diffusion-based generative model following the EDM framework [11]. For the generative variant, we use an asymmetric attention design in which self-attention is sparsified while cross-attention retains the full conditioning sequence. Across all settings, a keep ratio of r=0.25 yields $\sim$2.5$\times$ training speedup. Notably, Sparse-Reslim also improves forecast accuracy over the dense baseline across all evaluated variables and metrics. Our ablations show that this gain is not explained by stochastic regularization alone: deterministic sparse routing already captures most of the improvement, while random token selection provides a smaller additional benefit without selector parameters or scoring overhead. Our main contributions are:

- We propose Sparse-Reslim, a sparse routing module for spatiotemporal weather forecasting that preserves all spatial token positions while respecting temporal conditioning. It adds no learnable components and integrates into ViT-based weather forecasters as a plug-in module.
- We provide an empirical study spanning two architecture families (deterministic and generative), three ERA5 grid resolutions, and four atmospheric variables. Sparse-Reslim consistently improves accuracy across evaluated settings, and its benefits become more pronounced at higher resolution.
- We show that sparse routing is orthogonal to other efficiency strategies, including mixed-precision training and resolution downsampling, and can be combined with them for further gains. To our knowledge, this is the first application of sparse token routing to efficient training of ViT-based weather models in both deterministic and generative settings.

2 Related Work

2.1 AI Weather Forecasting Models

Data-driven weather forecasting has emerged as a compelling alternative to traditional NWP, reaching comparable or better accuracy at a fraction of the computational cost. The task is to forecast future atmospheric states $\mathbf{X}_T \in \mathbb{R}^{V\times H\times W}$ from initial conditions $\mathbf{X}_0$, where V denotes physical variables, $H \times W$ the spatial grid, and T the target lead time. Among deterministic approaches, Pangu-Weather [1] introduced a 3D Earth-Specific Transformer operating on pressure-level volumes, while GraphCast [13] adopted a mesh-based graph neural network for message passing on the sphere. ClimaX [17] proposed a variable-agnostic ViT trained across heterogeneous climate datasets, and Stormer [19] combined shifted window attention with weather-specific inductive biases. FengWu [4] uses

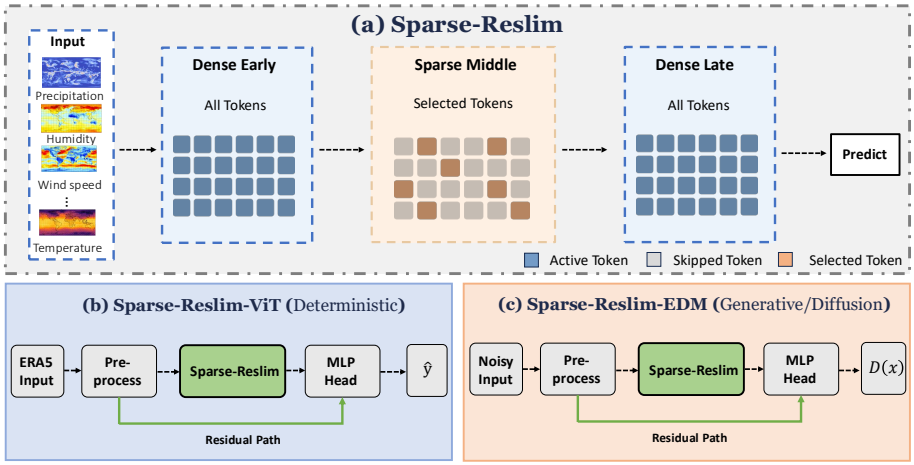

Fig. 1: Overview of Sparse-Reslim. (a) Three-stage token routing: dense early (n_1 blocks), sparse middle (n_2 blocks on $K{=}\lfloor rL \rfloor$ tokens), and dense late (n_3 blocks), with unselected tokens following an identity path. (b, c) Integration into the deterministic Res-Slim-ViT and generative Res-Slim-EDM backbones.

a multi-modal, multi-task cross-modal fusion transformer to push skillful forecasts beyond 10 lead days, and FuXi [5] and Aurora [2] extended the frontier with cascade and foundation model designs, respectively.

On the probabilistic side, GenCast [21] applied diffusion models to produce calibrated ensemble forecasts, while OmniCast [18] and CorrDiff [16] explored generative approaches for modeling the full distribution of future weather states, addressing the blurring and error accumulation that affect deterministic autoregressive rollout. Despite their architectural diversity, all these models share one trait: they apply uniform computation across all spatial tokens, regardless of local informational complexity. This becomes increasingly wasteful at higher resolutions, where quadratic attention cost dominates training time yet much of the grid is smooth and redundant. Res-Slim-ViT, introduced as part of the ORBIT-2 exascale climate downscaling system [27], showed strong scalability through a dual-path design combining a ViT backbone with a convolutional residual path, making it a natural testbed for efficient token routing.

2.2 Efficient Vision Transformers

The quadratic cost of self-attention has motivated extensive work on reducing the computational burden of ViTs, and decreasing the number of tokens processed is among the most studied strategies. Attention-based token pruning removes tokens with low attention scores [8, 12, 14, 30]. Several methods go further by training scoring modules that learn task-dependent pruning criteria [22, 23]. Token merging offers a complementary direction: ToMe [3] condenses similar tokens

via bipartite matching, reducing sequence length while preserving representational content. Other approaches operate at the attention level itself, including window-based attention [15] and learned sparse attention patterns [6, 28].

A more recent line of work targets generative model training specifically. Sprint [20] introduced a dense-sparse-dense block schedule for diffusion transformers: only a fraction of randomly selected tokens pass through the middle blocks, while dropped tokens are replaced by learnable mask embeddings and later fused with shallow dense features via a linear projection. While effective for image generation, Sprint is not directly suited to weather forecasting. Weather models condition on past atmospheric states and must return a complete future state on the original grid. In autoregressive use, this output becomes the next input. Replacing unprocessed positions with mask embeddings therefore does more than add a small projection layer: it breaks the continuity of the atmospheric state representation at fixed grid locations. Our residual delta formulation addresses this gap by keeping unselected tokens as their actual pre-routing representations and scattering back only the updates computed for routed tokens. In the generative setting, we further preserve temporal conditioning by sparsifying self-attention over target tokens while retaining full cross-attention to the conditioning sequence.

3 Methods

This section describes Sparse-Reslim, a drop-in sparse token routing module for ViT-based weather models. Figure 1 provides a visual overview of Sparse-Reslim. We first cover the shared tokenization and base architectures (Sec. 3.1), then the routing module (Sec. 3.2), and its integration into deterministic (Sec. 3.3) and generative (Sec. 3.4) forecasting pipelines.

3.1 Preliminaries

Tokenization. Both architectures share a per-variable tokenization scheme. Each variable v is independently patchified by a dedicated convolutional embedding $\texttt{PatchEmbed}_v$, producing tokens of dimension D. A learnable variable embedding $\mathbf{e}_v$ is added, and variables are aggregated via cross-attention with a shared query $\mathbf{q} \in \mathbb{R}^D$:

$$\mathbf{z}_i = \mathrm{CrossAttn}\big(\mathbf{q},\ \{\mathbf{p}_{i,v} + \mathbf{e}_v\}_{v \in V}\big) \in \mathbb{R}^D, \tag{1}$$

where $\mathbf{p}_{i,v}$ is the patch token at spatial position i for variable v. This yields a sequence $\mathbf{Z} \in \mathbb{R}^{L \times D}$ with $L = (h/p)(w/p)$ tokens, augmented with sinusoidal positional and spatial resolution embeddings.

Deterministic forecasting. The deterministic model f_θ directly regresses the forecast target. Given input history $\mathbf{X}$, the history dimension is flattened into the channel dimension and the model predicts:

$$\hat{Y} = f_\theta(\mathbf{X}) = \mathrm{CNN}_{\mathrm{skip}}(\mathbf{X}) + \mathrm{Unpatchify}\big(\mathrm{MLP}_{\mathrm{head}}(\mathrm{ViT}(\mathbf{Z}))\big), \tag{2}$$

where $\mathrm{ViT}(\cdot)$ denotes N transformer blocks and $\mathrm{CNN}_{\mathrm{skip}}$ is a convolutional residual path (Conv $\rightarrow$ PixelShuffle $\rightarrow$ Conv) that bypasses the transformer. The dual-path design lets the lightweight CNN capture dense local patterns, freeing the transformer to focus on long-range dependencies. Training minimizes a latitude-weighted mean squared error:

$$\mathcal{L}_{\mathrm{det}} = \frac{1}{|V_{\mathrm{out}}|} \sum_{v} \omega_v \cdot \frac{\sum_i \cos(\phi_i) \big(\hat{Y}_{v,i} - Y_{v,i}\big)^2}{\sum_i \cos(\phi_i)}, \tag{3}$$

where ω_v are per-variable weights and ϕ_i is the latitude at grid point i.

Generative forecasting. The diffusion model follows the EDM framework [11], parameterizing a denoiser D_θ via preconditioning scalars derived from the noise level σ:

$$D_\theta(\mathbf{y}; \sigma, \mathbf{c}) = c_{\mathrm{skip}}(\sigma) \cdot \mathbf{y} + c_{\mathrm{out}}(\sigma) \cdot F_\theta\big(c_{\mathrm{in}}(\sigma) \cdot \mathbf{y};\ \sigma,\ \mathbf{c}\big), \tag{4}$$

where $\mathbf{y} = Y + \sigma\boldsymbol{\epsilon}$ is the noisy target with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{c}$ represents the conditioning fields (history frames), and $c_{\mathrm{skip}}, c_{\mathrm{out}}, c_{\mathrm{in}}$ are analytic functions of σ and the data standard deviation σ_{data}. The raw network F_θ is a ViT whose blocks contain noise-conditioned FiLM modulation, noise-gated cross-attention, and a CNN skip path:

$$F_\theta(\cdot) = \mathrm{Unpatchify}\big(\mathrm{MLP}_{\mathrm{head}}(\mathrm{ViT}_{\mathrm{edm}}(\mathbf{Z}_x, \mathbf{Z}_c, \mathbf{e}_\sigma))\big) + \mathrm{CNN}_{\mathrm{skip}}(c_{\mathrm{in}} \cdot \mathbf{y}). \tag{5}$$

Here $\mathbf{Z}_x$ and $\mathbf{Z}_c$ are token sequences from the noisy target and conditions respectively, and $\mathbf{e}_\sigma = \mathrm{MLP}(\ln \sigma/4)$ is the noise embedding. Each EDM block consists of FiLM modulation (*scale*, *shift* from $\mathbf{e}_\sigma$), self-attention, σ-gated cross-attention (gate $= \mathrm{sigmoid}(\mathbf{W}\mathbf{e}_\sigma)$), and MLP. Training minimizes the denoising loss:

$$\mathcal{L}_{\mathrm{edm}} = \mathbb{E}_{\sigma \sim p(\sigma)} \Big[\lambda(\sigma) \big\| D_\theta(\mathbf{y}; \sigma, \mathbf{c}) - Y \big\|^2 \Big], \tag{6}$$

where $\ln \sigma \sim \mathcal{N}(P_{\mathrm{mean}}, P_{\mathrm{std}}^2)$ and $\lambda(\sigma) = (\sigma^2 + \sigma_{\mathrm{data}}^2)/(\sigma \cdot \sigma_{\mathrm{data}})^2$ is the weighting.

3.2 Sparse-Reslim: Sparse Token Routing

Motivation. Self-attention costs $O(L^2 D)$ per block. For ERA5 at 1.0° with patch size 2, L=16,200 tokens, and the N=12 transformer blocks dominate training time. Atmospheric fields, however, exhibit strong spatial autocorrelation: nearby grid points carry highly redundant information, especially in the smooth interior of large-scale circulation features. Every token must appear in the final output (forecasts are spatially dense), but the *intermediate representations* in the middle layers can be approximated by processing only a subset of tokens and broadcasting their updates back via a residual delta.

Three-stage block partition. Given N transformer blocks, Sparse-Reslim partitions them as:

$$\underbrace{\mathrm{Block}_1, \ldots, \mathrm{Block}_{n_1}}_{\text{Dense Early}}, \quad \underbrace{\mathrm{Block}_{n_1+1}, \ldots, \mathrm{Block}_{n_1+n_2}}_{\text{Sparse Middle}}, \quad \underbrace{\mathrm{Block}_{n_1+n_2+1}, \ldots, \mathrm{Block}_{N}}_{\text{Dense Late}}, \tag{7}$$

where $n_1+n_2+n_3 = N$. The dense early stage (n_1 blocks) builds rich initial representations through full token interaction. The sparse middle stage (n_2 blocks) processes only a subset of tokens, yielding the bulk of the speedup. The dense late stage (n_3 blocks) refines all tokens before the prediction head, restoring spatial coherence.

Random token selection. At the boundary between the dense early and sparse middle stages, we randomly select $K = \lfloor L \cdot r \rfloor$ tokens per sample, where $r \in (0, 1]$ is the *keep ratio*. Let $\mathcal{I} = \{i_1, \ldots, i_K\} \subset \{1, \ldots, L\}$ be the index set, drawn uniformly without replacement and sorted to maintain spatial ordering. Random selection, rather than attention-score-based selection, has three advantages: (1) it requires no scoring overhead, (2) it acts as a stochastic regularizer during training (analogous to dropout on the token dimension), and (3) it avoids systematic bias that could cause certain geographic regions to be consistently underrepresented.

Residual delta formulation. Let $\mathbf{x} \in \mathbb{R}^{B \times L \times D}$ denote the full token sequence after the dense early stage. We save a copy $\mathbf{x}_0 = \mathbf{x}$, then gather the selected tokens: $\mathbf{h}_0 = \text{Gather}(\mathbf{x}, \mathcal{I}) \in \mathbb{R}^{B \times K \times D}$. These K tokens pass through the n_2 sparse middle blocks:

$$\mathbf{h}_{n_2} = \text{Block}_{n_1+n_2} \circ \cdots \circ \text{Block}_{n_1+1}(\mathbf{h}_0). \tag{8}$$

The residual delta is then computed and scattered back:

$$\boldsymbol{\Delta} = \mathbf{h}_{n_2} - \mathbf{h}_0, \qquad \mathbf{x}' = \mathbf{x}_0 + \text{Scatter}(\boldsymbol{\Delta}, \mathcal{I}), \tag{9}$$

where Scatter places the K-dimensional deltas into an L-dimensional zero tensor at the selected indices. For the $L-K$ unselected tokens, $\boldsymbol{\Delta}_i = 0$, so they retain their pre-routing representation $\mathbf{x}_{0,i}$ exactly, forming an implicit identity skip connection. This is a key departure from Sprint [20], which replaces dropped tokens with learnable mask embeddings and requires an extra linear projection to fuse them back. The mask embeddings never pass through the middle transformer blocks, so they carry a fundamentally different representation distribution than the processed tokens; the fusion layer must learn to compensate for this mismatch. Our formulation sidesteps the issue entirely: unselected tokens retain their actual dense-early representations, the delta scatter is additive and parameter-free, and no fusion layer is needed. We verify this advantage empirically in Table 5, where even within our framework, deterministic selection strategies underperform random selection, confirming that the stochastic regularization from our zero-parameter design is a feature, not a compromise.

Computational savings. In the sparse middle stage, self-attention operates on K tokens instead of L, reducing cost from $O(n_2 L^2 D)$ to $O(n_2 r^2 L^2 D)$. With the default r=0.25 and n_1=n_3=2, the theoretical FLOPs reduction for the transformer blocks is:

$$\text{Speedup} \approx \frac{N}{n_1 + n_2 r^2 + n_3} = \frac{12}{2 + 8 \cdot 0.0625 + 2} = \frac{12}{4.5} \approx 2.67\times. \tag{10}$$

In practice, we observe ∼2.46× speedup on AMD MI250X GPUs, with the gap from theory explained by the overhead of `gather`/`scatter` operations and non-transformer components (patch embedding, prediction head, CNN skip path).

Require: Input $\mathbf{X} \in \mathbb{R}^{B\times(H\cdot|V_{\text{in}}|)\times h\times w}$, blocks $\{\text{Block}_i\}_{i=1}^{N}$, keep ratio r, block split (n_1, n_2, n_3)
Ensure: Forecast $\hat{Y} \in \mathbb{R}^{B\times|V_{\text{out}}|\times h\times w}$
$\mathbf{s} \leftarrow \text{CNN}_{\text{skip}}(\mathbf{X})$ ▷ Parallel CNN residual path
$\mathbf{Z} \leftarrow \text{VarAgg}(\text{PatchEmbed}(\mathbf{X})) + \mathbf{p}_{\text{pos}} + \mathbf{p}_{\text{res}}$ ▷ $\mathbf{Z} \in \mathbb{R}^{B\times L\times D}$
% Stage 1: Dense early (all L tokens)
for $i = 1$ to n_1 **do**
  $\mathbf{Z} \leftarrow \text{Block}_i(\mathbf{Z})$
end for
% Stage 2: Sparse middle (K random tokens)
$K \leftarrow \lfloor L \cdot r \rfloor$; $\mathcal{I} \leftarrow \texttt{sort}(\texttt{randperm}(L)[:K])$
$\mathbf{x}_0 \leftarrow \mathbf{Z}.\texttt{clone}()$; $\mathbf{h}_0 \leftarrow \texttt{gather}(\mathbf{Z}, \mathcal{I})$; $\mathbf{h} \leftarrow \mathbf{h}_0.\texttt{clone}()$
for $i = n_1{+}1$ to $n_1{+}n_2$ **do**
  $\mathbf{h} \leftarrow \text{Block}_i(\mathbf{h})$ ▷ Self-Attn cost: $O(K^2D)$
end for
$\boldsymbol{\Delta} \leftarrow \mathbf{h} - \mathbf{h}_0$; $\mathbf{Z} \leftarrow \mathbf{x}_0 + \texttt{scatter}(\mathbf{0}, \mathcal{I}, \boldsymbol{\Delta})$
% Stage 3: Dense late (all L tokens)
for $i = n_1{+}n_2{+}1$ to N **do**
  $\mathbf{Z} \leftarrow \text{Block}_i(\mathbf{Z})$
end for
$\hat{Y} \leftarrow \text{Unpatchify}(\text{MLP}_{\text{head}}(\text{LN}(\mathbf{Z}))) + \mathbf{s}$

Algorithm 1: Sparse-Reslim for Deterministic Forecasting (Res-Slim-ViT)

3.3 Integration with Deterministic Forecasting

For Res-Slim-ViT, integration is straightforward: each block contains only self-attention and MLP with pre-norm residual connections. The sparse middle blocks process only K tokens through the standard block:

$$\mathbf{h} \leftarrow \mathbf{h} + \text{Self-Attn}(\text{LN}(\mathbf{h})), \quad \mathbf{h} \leftarrow \mathbf{h} + \text{MLP}(\text{LN}(\mathbf{h})), \tag{11}$$

with attention cost $O(K^2D)$ per block. All other components (patch embedding, variable aggregation, positional embedding, MLP head, CNN skip path, and unpatchify) remain unchanged and operate on the full L tokens. The complete procedure is given in Algorithm 1.

3.4 Integration with Generative Forecasting

The EDM architecture presents a more involved integration because each block contains cross-attention with the conditioning sequence $\mathbf{Z}_c \in \mathbb{R}^{B\times L\times D}$ and noise-conditioned FiLM modulation. During the sparse middle stage, we adopt an *asymmetric attention design*:

Self-attention operates on only the K selected tokens ($\text{Q}, \text{K}, \text{V} \in \mathbb{R}^{K\times D}$), reducing cost from $O(L^2D)$ to $O(K^2D)$.

Cross-attention uses the selected tokens as queries but retains the *full* conditioning sequence as keys and values ($\text{Q} \in \mathbb{R}^{K\times D}$, $\text{K}, \text{V} \in \mathbb{R}^{L\times D}$), reducing cost from $O(L^2D)$ to $O(KL \cdot D)$.

This asymmetry is essential. The conditioning fields $\mathbf{Z}_c$ encode the full atmospheric context (all history frames, all input variables), and restricting keys/values to only K positions would lose meteorological information needed for denoising. The noise embedding $\mathbf{e}_\sigma$ is a global vector broadcast to all tokens and needs no

Require: Noisy target $\mathbf{y} \in \mathbb{R}^{B\times|V_{\text{out}}|\times h\times w}$, conditions $\mathbf{c} \in \mathbb{R}^{B\times H\times|V_{\text{in}}|\times h\times w}$, noise level σ, blocks $\{\text{Block}_i^{\text{edm}}\}_{i=1}^{N}$, keep ratio r, split (n_1, n_2, n_3)
Ensure: Denoised output $D_\theta(\mathbf{y}; \sigma, \mathbf{c})$
$c_{\text{in}}, c_{\text{out}}, c_{\text{skip}} \leftarrow \text{EDM\_precond}(\sigma)$
$\mathbf{s} \leftarrow \text{CNN}_{\text{skip}}(c_{\text{in}} \cdot \mathbf{y})$ ▷ Parallel CNN residual path
$\mathbf{Z}_x \leftarrow \text{VarAgg}(\text{PatchEmbed}(c_{\text{in}} \cdot \mathbf{y})) + \mathbf{p}_{\text{pos}} + \mathbf{p}_{\text{res}} + \mathbf{p}_{\text{time}}$
$\mathbf{Z}_c \leftarrow \text{TemporalProj}(\text{VarAgg}(\text{PatchEmbed}(\mathbf{c}))) + \mathbf{p}_{\text{pos}} + \mathbf{p}_{\text{res}}$ ▷ $\mathbf{Z}_c \in \mathbb{R}^{B\times L\times D}$
$\mathbf{e}_\sigma \leftarrow \text{MLP}(\text{SiLU}(\text{MLP}(\ln \sigma/4)))$ ▷ Noise embedding
% Stage 1: Dense early (all L tokens)
for $i = 1$ to n_1 **do**
  $\mathbf{Z}_x \leftarrow \text{Block}_i^{\text{edm}}(\mathbf{Z}_x,\ \mathbf{Z}_c,\ \mathbf{e}_\sigma)$
end for
% Stage 2: Sparse middle (K random tokens)
$K \leftarrow \lfloor L \cdot r \rfloor$; $\mathcal{I} \leftarrow \texttt{sort}(\texttt{randperm}(L)[:K])$
$\mathbf{x}_0 \leftarrow \mathbf{Z}_x\texttt{.clone()}$; $\mathbf{h}_0 \leftarrow \texttt{gather}(\mathbf{Z}_x, \mathcal{I})$; $\mathbf{h} \leftarrow \mathbf{h}_0\texttt{.clone()}$
for $i = n_1{+}1$ to $n_1{+}n_2$ **do**
  $\mathbf{h} \leftarrow \text{Block}_i^{\text{edm}}(\mathbf{h},\ \mathbf{Z}_c,\ \mathbf{e}_\sigma)$ ▷ Self-Attn: $O(K^2)$; Cross-Attn Q=sparse, KV=full
end for
$\boldsymbol{\Delta} \leftarrow \mathbf{h} - \mathbf{h}_0$; $\mathbf{Z}_x \leftarrow \mathbf{x}_0 + \texttt{scatter}(\mathbf{0}, \mathcal{I}, \boldsymbol{\Delta})$
% Stage 3: Dense late (all L tokens)
for $i = n_1{+}n_2{+}1$ to N **do**
  $\mathbf{Z}_x \leftarrow \text{Block}_i^{\text{edm}}(\mathbf{Z}_x,\ \mathbf{Z}_c,\ \mathbf{e}_\sigma)$
end for
$F_\theta \leftarrow \text{Unpatchify}(\text{MLP}_{\text{head}}(\text{LN}(\mathbf{Z}_x))) + \mathbf{s}$
$D_\theta \leftarrow c_{\text{skip}} \cdot \mathbf{y} + c_{\text{out}} \cdot F_\theta$

Algorithm 2: Sparse-Reslim for Generative Forecasting (EDM)

modification; FiLM modulation and the σ-gate operate element-wise and naturally accommodate the reduced sequence length.

The per-block cost in the sparse middle stage becomes:

$$\underbrace{O(K^2D)}_{\text{sparse self-attn}} + \underbrace{O(KLD)}_{\text{sparse cross-attn}} + \underbrace{O(KD^2)}_{\text{MLP + FiLM}}, \tag{12}$$

compared to $O(L^2D) + O(L^2D) + O(LD^2)$ in the dense baseline. The complete procedure is given in Algorithm 2.

Training and inference. A fresh random permutation is drawn at every forward pass, providing stochastic regularization analogous to dropout on the token dimension. The same random selection is used at inference; averaging multiple runs yields implicit ensembling. For the EDM model, routing is applied identically across all denoising steps.

Architecture-agnostic design. Sparse-Reslim requires no additional learnable parameters. It introduces only three hyperparameters: the keep ratio r and the block split (n_1, n_3) (with $n_2 = N - n_1 - n_3$). The module wraps the existing block loop and is activated by a single configuration flag. When disabled (r=1), the model reverts exactly to the dense baseline with zero overhead.

4 Experiments

4.1 Experimental Setup

Dataset. We train and evaluate on ERA5 reanalysis [9, 10, 24, 25], the most widely used benchmark for data-driven weather forecasting. Following the pro-

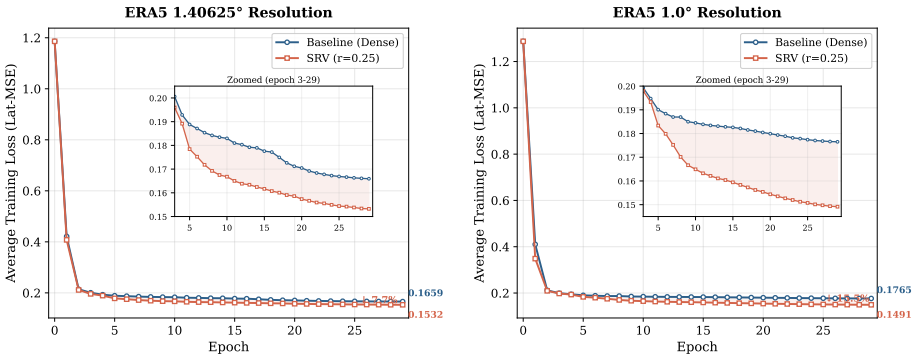

Fig. 2: Training convergence for Res-Slim-ViT at 1.40625° and 1.0°. Sparse-Reslim (solid) converges faster than the dense baseline (dashed) in both wall-clock time and training loss, reaching lower final values at both resolutions.

tocol of ClimaX [17], we use 1979–2017 for training, 2018–2019 for validation, and 2020 for testing. The input consists of 40 atmospheric variables: six variables (geopotential, temperature, specific humidity, u-wind, v-wind, relative humidity) at seven pressure levels (50, 250, 500, 600, 700, 850, 925 hPa), plus four surface and constant fields (2m temperature, 10m u-wind, 10m v-wind, land-sea mask). Evaluation targets four variables standard in NWP verification: geopotential at 500 hPa (Z500), temperature at 850 hPa (T850), 2-meter temperature (T2m), and 10-meter zonal wind (U10). Z500 and T850 are upper-air variables used for synoptic-scale assessment; T2m and U10 are surface variables directly relevant to human activities. Unless noted otherwise, experiments use a 5-day (120-hour) forecast lead time. We run experiments at three spatial resolutions: 1.40625° (128×256 grid, L=8,192 tokens with patch size 2), 1.0° (181×360 grid, L=16,200 tokens with patch size 2), and 0.25° (720×1440 grid, L=16,200 tokens with patch size 8). The 0.25° setting uses the same 1979–2017 training period as the coarser-resolution experiments.

Model configurations. We evaluate Sparse-Reslim on two backbone architectures: *Deterministic model (Res-Slim-ViT).* Adapted from ORBIT-2 [27], this model uses a dual-path architecture with a ViT backbone and a convolutional residual skip path. At 1.40625°, the model has embed_dim=256, depth=12, four attention heads, totaling 12.18M parameters. At 1.0°, the same architecture on the larger grid yields 14.23M parameters. At 0.25°, we use patch size 8 so that the routed token count matches the 1.0° setting. All deterministic settings use block split (n_1, n_2, n_3)=$(2, 8, 2)$ with keep ratio r=0.25. *Generative model (EDM).* Following the EDM framework [11], this model uses a ViT backbone with FiLM-conditioned noise modulation, σ-gated cross-attention, and a CNN skip path. The architecture has patch_size=6, embed_dim=1024, depth=8, eight attention heads, totaling ∼180M parameters, with block split (n_1, n_2, n_3)=$(2, 4, 2)$ and r=0.25.

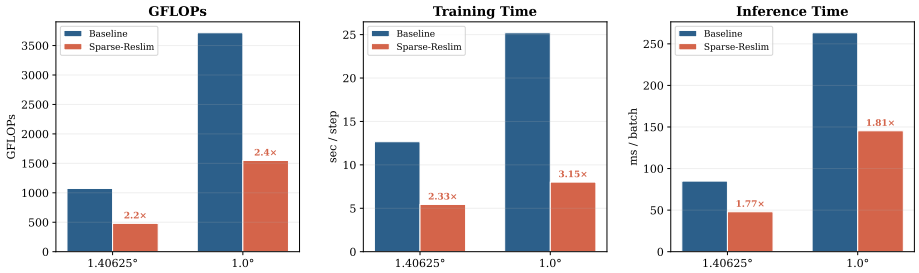

Fig. 3: Training efficiency across resolutions for the deterministic model. Sparse-Reslim achieves consistent wall-clock speedup at both resolutions, with larger gains at 1.0° due to the higher token count (L=16,200 vs. 8,192).

Implementation details. All models are trained with Fully Sharded Data Parallel (FSDP) and activation checkpointing. The deterministic model runs on 2 nodes with 16 AMD MI250X GPUs, using AdamW with learning rate 5×10^{-4} (scaled by num_gpus$^{0.8}$), cosine schedule with 5 epochs of linear warmup, and batch size 32 (1.40625°) or 16 (1.0° and 0.25°). The generative model is trained on 8 nodes with 64 AMD MI250X GPUs. Both use bfloat16 mixed precision. The latitude-weighted MSE (Eq. 3) and EDM denoising loss (Eq. 6) are used for the deterministic and generative models, respectively.

Baselines. Our primary comparison is the dense baseline, i.e., the identical architecture with sparse routing disabled (r=1). This isolates the effect of sparse routing from other design choices. We refer to the dense baseline as **Reslim** and the sparse variant as **Sparse-Reslim**, trained for the same number of epochs.

Evaluation metrics. Forecast quality is measured with three complementary metrics: (1) **RMSE** (root mean squared error), the standard pointwise accuracy metric; (2) **ACC** (anomaly correlation coefficient), which measures correlation between predicted and observed anomalies relative to climatology and is the primary skill score in operational NWP; and (3) **SSIM** (structural similarity index), which captures perceptual similarity and penalizes spatial blurring. For efficiency, we report wall-clock training time, throughput (samples per second per GPU), and peak VRAM.

4.2 Results on Deterministic Forecasting

Table 1 compares forecast quality between the dense Reslim baseline and Sparse-Reslim across all four target variables and all three resolutions. Sparse-Reslim outperforms the dense baseline on every metric and every variable, often by a substantial margin. At 1.0°, Z500 RMSE drops from 868.7 to 768.3 and ACC rises from 0.551 to 0.651. T2m shows a similar pattern: RMSE improves from 3.19 to 2.68 and ACC from 0.797 to 0.859. The gains are consistent at 1.40625° as well, though slightly smaller in magnitude, which is expected given the lower token count. The 0.25° experiment further confirms the high-resolution behavior

Table 1: Forecast quality on the ERA5 test set, reported as single-step, non-autoregressive 120-hour forecasts for the deterministic Res-Slim-ViT. Sparse-Reslim uses $r = 0.25$ with block split $(2, 8, 2)$. The 0.25° setting uses the full 1979–2017 training period with patch size 8. **Bold** = better within each resolution.

		1.40625°			**1.0°**			**0.25°**		
Variable	**Method**	RMSE↓	ACC↑	SSIM↑	RMSE↓	ACC↑	SSIM↑	RMSE↓	ACC↑	SSIM↑
T2m	Dense	3.18	0.798	0.858	3.19	0.797	0.857	3.31	0.775	0.865
	Sparse-Reslim	**2.84**	**0.835**	**0.878**	**2.68**	**0.859**	**0.894**	**2.82**	**0.846**	**0.899**
U10	Dense	3.93	0.296	0.318	4.09	0.227	0.296	4.16	0.243	0.304
	Sparse-Reslim	**3.87**	**0.341**	**0.334**	**3.83**	**0.366**	**0.362**	**3.70**	**0.392**	**0.374**
Z500	Dense	872.9	0.546	0.761	868.7	0.551	0.802	1042	0.548	0.815
	Sparse-Reslim	**809.6**	**0.616**	**0.794**	**768.3**	**0.651**	**0.852**	**892**	**0.670**	**0.865**
T850	Dense	3.92	0.629	0.737	3.86	0.629	0.771	4.03	0.651	0.780
	Sparse-Reslim	**3.60**	**0.690**	**0.763**	**3.37**	**0.726**	**0.812**	**3.45**	**0.753**	**0.824**

Table 2: Peak VRAM per AMD MI250X GCD for the deterministic model. Sparse-Reslim uses r=0.25 and split $(2, 8, 2)$.

Tokens L	**Setting**	**Dense (GB)**	**Sparse (GB)**	**Reduction**
8,192	1.40625°, p=2	26.7	12.1	**2.20**×
16,200	1.0°, p=2	49.1	20.4	**2.41**×
16,200	0.25°, p=8	49.3	20.6	**2.40**×

under the full 39-year training split: Sparse-Reslim improves all four variables while using the same routed token count as the 1.0° patch-size-2 setting.

Efficiency methods typically aim to *preserve* accuracy, not improve it. Our ablations indicate a dual mechanism: token-level sparsity itself reduces redundant intermediate computation and can improve generalization, while random selection adds a further dropout-like effect without selector parameters or scoring overhead. Figure 2 confirms that Sparse-Reslim converges faster and reaches lower final loss than the dense baseline, ruling out under-training as an explanation. Figure 3 shows that speedup is consistent across resolutions and grows at higher resolutions where the larger token count amplifies quadratic savings.

Table 2 reports empirical peak VRAM at fixed token counts on 16 AMD MI250X GCDs. Sparse-Reslim reduces peak activation memory by more than 2.2× across all tested settings. The measured reduction is close to the theoretical middle-block estimate $12/(4 + 8r^2) = 2.67\times$ for split $(2, 8, 2)$ at r=0.25; the remaining gap comes from full-sequence components such as patch embedding, dense early/late blocks, the prediction head, and the CNN skip path.

Table 3 addresses the stability of stochastic routing under autoregressive deployment. Sparse-Reslim remains above the dense baseline throughout the 10-day rollout, and the skill decays smoothly rather than showing compounding instability. The result is consistent with the residual identity path: an unselected token is not replaced by noise or a mask embedding, but simply receives zero middle-block delta before being refined by the dense late blocks.

Table 3: Autoregressive rollout stability on the 2020 test year. Both models are trained for 24-hour prediction at 1.40625° and rolled out for 1–10 days. Values are T2m ACC.

Method	1	2	3	4	5	6	7	8	9	10
Dense	0.85	0.81	0.79	0.76	0.71	0.68	0.65	0.60	0.58	0.53
Sparse-Reslim	**0.92**	**0.89**	**0.86**	**0.83**	**0.80**	**0.77**	**0.75**	**0.73**	**0.71**	**0.69**

4.3 Results on Generative Forecasting

We apply Sparse-Reslim to the EDM-based generative model at 1.0° with the asymmetric attention design described in Sec. 3.4. The EDM backbone has ∼180M parameters, depth 8 with split $(2, 4, 2)$, and uses patch size 6 with embed dim 1024. All metrics are computed on the ensemble mean of 4 samples.

Table 4 shows that Sparse-Reslim transfers to the generative setting: Z500 RMSE drops from 951.3 to 889.7 and ACC rises from 0.517 to 0.571. T2m and T850 follow the same pattern with RMSE reductions of 6.6% and 6.4%, respectively. The improvement margin is smaller than in the deterministic case, which is expected: the EDM backbone uses split $(2, 4, 2)$ with only 4 sparse middle blocks instead of 8, reducing the computational savings and the strength of stochastic regularization. Sparse-Reslim EDM trains at 1.82× the wall-clock speed of the dense EDM baseline (20.8 vs. 11.4 samples/s/GPU), consistent with the smaller n_2 and the fact that cross-attention retains full-length conditioning tokens.

Fig. 4: Forecast quality of the EDM generative model on the ERA5 test set (year 2020, 120-hour lead time) at 1.0°. Sparse-Reslim uses r=0.25 with block split $(2, 4, 2)$. **Bold** = better.

Variable	Method	RMSE↓	ACC↑
T2m	Dense EDM	3.510	0.771
	Sparse-Reslim EDM	3.280	0.802
U10	Dense EDM	4.420	0.204
	Sparse-Reslim EDM	4.170	0.251
Z500	Dense EDM	951.300	0.517
	Sparse-Reslim EDM	889.700	0.571
T850	Dense EDM	4.220	0.596
	Sparse-Reslim EDM	3.950	0.648

4.4 Ablation Studies

Ablations use the deterministic Res-Slim-ViT at 1.0° unless stated otherwise.

Effect of keep ratio r. We vary $r \in \{0.10, 0.25, 0.50, 0.75, 1.0\}$ with a fixed split $(2, 8, 2)$. Figure 5 plots Z500 RMSE and wall-clock speedup as a function of r, where r=1.0 corresponds to the dense baseline. The accuracy-speedup tradeoff exhibits a clear U-shape: r=0.25 achieves the lowest RMSE (768.3) and the highest ACC (0.651), outperforming both the dense baseline (r=1.0, RMSE 868.7) and the more aggressive r=0.10 (RMSE 801.5). At r=0.10, the model loses too many tokens and intermediate representations become impoverished; at $r \geq 0.50$, the stochastic regularization effect diminishes and accuracy degrades toward the

Table 4: Effect of block split (n_1, n_2, n_3) on Z500 at 1.0°. r=0.25, N=12 total blocks.

Split (n_1, n_2, n_3)	Z500 RMSE↓	Z500 ACC↑	Speedup↑
(1, 10, 1)	812.400	0.609	3.120
(**2**, **8**, **2**)	768.300	0.651	2.460
(3, 6, 3)	783.600	0.638	1.780
(4, 4, 4)	831.200	0.590	1.440

dense baseline. The speedup ranges from 1.27× at r=0.75 to 2.91× at r=0.10, with r=0.25 yielding 2.46×.

Effect of block split (n_1, n_2, n_3)**.** Table 4 compares four block partitions for a 12-block model at r=0.25. The split $(2, 8, 2)$ produces the best forecast accuracy. Reducing either end to a single dense block, as in $(1, 10, 1)$, degrades RMSE by 5.7% despite yielding the highest speedup (3.12×): one dense early block is insufficient for building rich initial token representations, and one dense late block cannot adequately restore spatial coherence. Conversely, allocating more blocks to the dense stages, as in $(4, 4, 4)$, leaves too few sparse middle blocks for meaningful regularization and limits speedup.

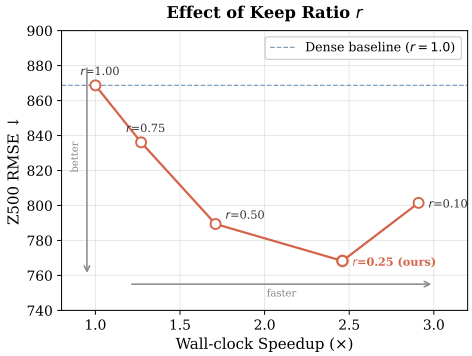

Fig. 5: Keep ratio r vs. Z500 RMSE and speedup. r=0.25 is the sweet spot.

Random vs. learned token selection. A natural question is whether informed token selection can outperform random sampling. Table 5 compares three strategies: (1) random selection (our default), (2) top-K by self-attention score from the last dense early block, and (3) a learned difficulty head (a lightweight MLP that predicts per-token importance). Random selection achieves the best Z500 RMSE while adding zero parameters and zero computational overhead. Attention-score-based selection introduces a systematic spatial bias, consistently prioritizing dynamically active regions (jet streams and frontal boundaries) and underrepresenting quiescent tropical areas, which hurts generalization. The learned head partially mitigates this bias but remains deterministic, forgoing the regularization benefit that random selection provides.

Scaling behavior with model size. We train three model sizes (Small: embed dim 128, depth 8, 3.1M params; Base: embed dim 256, depth 12, 12.2M; Large: embed dim 512, depth 16, 48.6M) with split $(2, N-4, 2)$ and r=0.25. Figure 6 shows that both speedup and accuracy improvement grow with model size. The Large model achieves 2.85× speedup (vs. 1.72× for Small) because deeper networks have a larger fraction of their compute in the sparse middle blocks.

Table 5: Token selection strategy comparison on Z500 at 1.0°. r=0.25, split $(2, 8, 2)$.

Selection	Z500 RMSE↓	Z500 ACC↑	Extra params	Overhead (ms/step)
Random (ours)	768.300	0.651	0	0
Top-K attn score	784.100	0.636	0	+14
Learned head	779.800	0.641	66K	+9

The RMSE improvement over the dense baseline also grows: −4.0% for Small, −11.6% for Base, and −12.6% for Large, indicating that larger models benefit more from the stochastic regularization provided by random token dropping.

5 Conclusion

We introduced Sparse-Reslim, a parameter-free sparse token routing module that partitions ViT blocks into dense early, sparse middle, and dense late stages, processing only a random 25% of tokens in the dominant middle layers while a residual delta formulation maintains spatially dense outputs. Experiments on ERA5 at three resolutions show ∼2.5× wall-clock training speedup and more than 2.2× peak VRAM reduction alongside improved forecast accuracy: Z500 RMSE drops by 11.6% at 1.0° relative to the dense baseline, with similar gains for T2m, T850, and U10. We attribute this to a combination of sparse routing and stochastic regularization: deterministic sparse routing already captures much of the gain, while random token selection adds a smaller but consistent benefit without selector parameters or scoring overhead. The method generalizes across two architecturally distinct backbones, a deterministic Res-Slim-ViT and a diffusion-based generative model, remains stable in 10-day autoregressive rollout, and adds no learnable parameters. A natural extension is adaptive token selection that allocates computation to dynamically active regions such as storm fronts and jet streams.

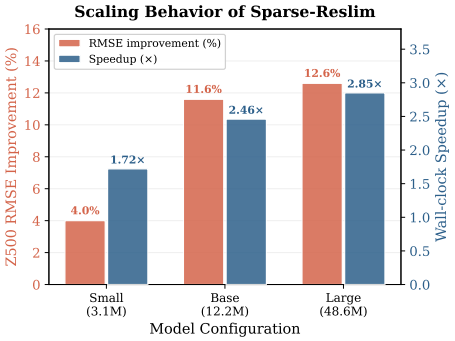

Fig. 6: Scaling behavior: both speedup and RMSE improvement grow with model size.

Acknowledgements. This manuscript has been authored by UTBattelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). This research was supported by the ORNL's AI Initiative sponsored by the Director's Research and Development Program at ORNL. This work's computing time was supported through the Oak Ridge Leadership Computing Facility.

References

1. Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., Tian, Q.: Accurate medium-range global weather forecasting with 3d neural networks. Nature **619**(7970), 533–538 (2023) 2, 3
2. Bodnar, C., Bruinsma, W.P., Lucic, A., Stanley, M., Allen, A., Brandstetter, J., Garvan, P., Riechert, M., Weyn, J.A., Dong, H., et al.: A foundation model for the earth system. Nature **641**(8065), 1180–1187 (2025) 2, 4
3. Bolya, D., Fu, C.Y., Dai, X., Zhang, P., Feichtenhofer, C., Hoffman, J.: Token merging: Your vit but faster. arXiv preprint arXiv:2210.09461 (2022) 2, 4
4. Chen, K., Han, T., Gong, J., Bai, L., Ling, F., Luo, J.J., Chen, X., Ma, L., Zhang, T., Su, R., et al.: Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. arXiv preprint arXiv:2304.02948 (2023) 3
5. Chen, L., Zhong, X., Zhang, F., Cheng, Y., Xu, Y., Qi, Y., Li, H.: Fuxi: a cascade machine learning forecasting system for 15-day global weather forecast. npj climate and atmospheric science **6**(1), 190 (2023) 2, 4
6. Chen, T., Cheng, Y., Gan, Z., Yuan, L., Zhang, L., Wang, Z.: Chasing sparsity in vision transformers: An end-to-end exploration. Advances in neural information processing systems **34**, 19974–19988 (2021) 5
7. Dueben, P.D., Bauer, P.: Challenges and design choices for global weather and climate models based on machine learning. Geoscientific Model Development **11**(10), 3999–4009 (2018) 1
8. Fayyaz, M., Koohpayegani, S.A., Jafari, F.R., Sengupta, S., Joze, H.R.V., Sommerlade, E., Pirsiavash, H., Gall, J.: Adaptive token sampling for efficient vision transformers. In: European conference on computer vision. pp. 396–414. Springer (2022) 4
9. Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Muñoz Sabater, J., Nicolas, J., Peubey, C., Radu, R., Rozum, I., et al.: Era5 hourly data on single levels from 1940 to present. Copernicus climate change service (c3s) climate data store (cds) **10**(1.24381), 24381 (2023) 9
10. Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., et al.: The era5 global reanalysis. Quarterly journal of the royal meteorological society **146**(730), 1999–2049 (2020) 3, 9
11. Karras, T., Aittala, M., Aila, T., Laine, S.: Elucidating the design space of diffusion-based generative models. Advances in neural information processing systems **35**, 26565–26577 (2022) 3, 6, 10
12. Kong, Z., Dong, P., Ma, X., Meng, X., Niu, W., Sun, M., Shen, X., Yuan, G., Ren, B., Tang, H., et al.: Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In: European conference on computer vision. pp. 620–640. Springer (2022) 2, 4
13. Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., et al.: Learning skillful medium-range global weather forecasting. Science **382**(6677), 1416–1421 (2023) 2, 3
14. Liang, Y., Ge, C., Tong, Z., Song, Y., Wang, J., Xie, P.: Not all patches are what you need: Expediting vision transformers via token reorganizations. arXiv preprint arXiv:2202.07800 (2022) 2, 4
15. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings

of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021) 2, 5
16. Mardani, M., Brenowitz, N., Cohen, Y., Pathak, J., Chen, C.Y., Liu, C.C., Vahdat, A., Nabian, M.A., Ge, T., Subramaniam, A., et al.: Residual corrective diffusion modeling for km-scale atmospheric downscaling. Communications Earth & Environment **6**(1), 124 (2025) 2, 4
17. Nguyen, T., Brandstetter, J., Kapoor, A., Gupta, J.K., Grover, A.: Climax: A foundation model for weather and climate. arXiv preprint arXiv:2301.10343 (2023) 2, 3, 10
18. Nguyen, T., Pham, T., Arcomano, T., Kotamarthi, V., Foster, I., Madireddy, S., Grover, A.: Omnicast: A masked latent diffusion model for weather forecasting across time scales. arXiv preprint arXiv:2510.18707 (2025) 4
19. Nguyen, T., Shah, R., Bansal, H., Arcomano, T., Maulik, R., Kotamarthi, R., Foster, I., Madireddy, S., Grover, A.: Scaling transformer neural networks for skillful and reliable medium-range weather forecasting. Advances in Neural Information Processing Systems **37**, 68740–68771 (2024) 2, 3
20. Park, D., Haji-Ali, M., Li, Y., Menapace, W., Tulyakov, S., Kim, H.J., Siarohin, A., Kag, A.: Sprint: Sparse-dense residual fusion for efficient diffusion transformers. arXiv preprint arXiv:2510.21986 (2025) 2, 5, 7
21. Price, I., Sanchez-Gonzalez, A., Alet, F., Andersson, T.R., El-Kadi, A., Masters, D., Ewalds, T., Stott, J., Mohamed, S., Battaglia, P., et al.: Gencast: Diffusion-based ensemble forecasting for medium-range weather. arXiv preprint arXiv:2312.15796 (2023) 2, 4
22. Rao, Y., Liu, Z., Zhao, W., Zhou, J., Lu, J.: Dynamic spatial sparsification for efficient vision transformers and convolutional neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(9), 10883–10897 (2023) 4
23. Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., Hsieh, C.J.: Dynamicvit: Efficient vision transformers with dynamic token sparsification. Advances in neural information processing systems **34**, 13937–13949 (2021) 2, 4
24. Rasp, S., Dueben, P.D., Scher, S., Weyn, J.A., Mouatadid, S., Thuerey, N.: Weatherbench: a benchmark data set for data-driven weather forecasting. Journal of Advances in Modeling Earth Systems **12**(11), e2020MS002203 (2020) 9
25. Rasp, S., Hoyer, S., Merose, A., Langmore, I., Battaglia, P., Russell, T., Sanchez-Gonzalez, A., Yang, V., Carver, R., Agrawal, S., et al.: Weatherbench 2: A benchmark for the next generation of data-driven global weather models. Journal of Advances in Modeling Earth Systems **16**(6), e2023MS004019 (2024) 9
26. Scher, S.: Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. Geophysical Research Letters **45**(22), 12–616 (2018) 1
27. Wang, X., Choi, J.Y., Kurihaya, T., Lyngaas, I., Yoon, H.J., Xiao, X., Pugmire, D., Fan, M., Nafi, N.M., Tsaris, A., et al.: Orbit-2: Scaling exascale vision foundation models for weather and climate downscaling. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 86–98 (2025) 3, 4, 10
28. Wei, C., Duke, B., Jiang, R., Aarabi, P., Taylor, G.W., Shkurti, F.: Sparsifiner: Learning sparse instance-dependent attention for efficient vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22680–22689 (2023) 5
29. Weyn, J.A., Durran, D.R., Caruana, R.: 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) 1

30. Xu, Y., Zhang, Z., Zhang, M., Sheng, K., Li, K., Dong, W., Zhang, L., Xu, C., Sun, X.: Evo-vit: Slow-fast token evolution for dynamic vision transformer. In: Proceedings of the AAAI conference on artificial intelligence. vol. 36, pp. 2964–2972 (2022) 2, 4