Earthformer: Exploring Space-Time Transformers for Earth System Forecasting

Zhihan Gao*
Hong Kong University of Science and Technology
zhihan.gao@connect.ust.hk

Xingjian Shi†
Amazon Web Services
xjshi@amazon.com

Hao Wang
Rutgers University
hw488@cs.rutgers.edu

Yi Zhu
Amazon Web Services
yzaws@amazon.com

Yuyang Wang
Amazon Web Services
yuyawang@amazon.com

Mu Li
Amazon Web Services
mli@amazon.com

Dit-Yan Yeung
Hong Kong University of Science and Technology
dyyeung@cse.ust.hk

Abstract

Conventionally, Earth system (e.g., weather and climate) forecasting relies on numerical simulation with complex physical models and hence is both expensive in computation and demanding on domain expertise. With the explosive growth of spatiotemporal Earth observation data in the past decade, data-driven models that apply Deep Learning (DL) are demonstrating impressive potential for various Earth system forecasting tasks. The Transformer as an emerging DL architecture, despite its broad success in other domains, has limited adoption in this area. In this paper, we propose *Earthformer*, a space-time Transformer for Earth system forecasting. Earthformer is based on a generic, flexible and efficient space-time attention block, named *Cuboid Attention*. The idea is to decompose the data into cuboids and apply cuboid-level self-attention in parallel. These cuboids are further connected with a collection of global vectors. We conduct experiments on the MovingMNIST dataset and a newly proposed chaotic N-body MNIST dataset to verify the effectiveness of cuboid attention and figure out the best design of Earthformer. Experiments on two real-world benchmarks about precipitation nowcasting and El Niño/Southern Oscillation (ENSO) forecasting show that Earthformer achieves state-of-the-art performance.

1 Introduction

The Earth is a complex system. Variabilities of the Earth system, ranging from regular events like temperature fluctuation to extreme events like drought, hail storm, and El Niño/Southern Oscillation (ENSO), impact our daily life. Among all the consequences, Earth system variabilities can influence crop yields, delay airlines, cause floods and forest fires. Precise and timely forecasting of these variabilities can help people take necessary precautions to avoid crisis, or better utilize natural resources such as wind and solar energy. Thus, improving forecasting models for Earth variabilities (e.g., weather and climate) has a huge socioeconomic impact. Despite its importance, the operational weather and climate forecasting systems have not fundamentally changed for almost 50 years [34]. These operational models, including the state-of-the-art High Resolution Ensemble Forecast (HREF)

*Work done while being an intern at Amazon Web Services. †Contact person.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

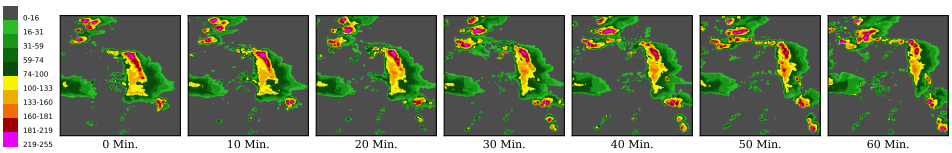


Figure 1: Example Vertically Integrated Liquid (VIL) observation sequence from the Storm EVent ImageRy (SEVIR) dataset. The observation intensity is mapped to pixel value of the range 0-255. The larger value indicates the higher precipitation intensity.

rainfall nowcasting model used in National Oceanic and Atmospheric Administration (NOAA) [32], rely on meticulous numerical simulation of physical models. Such simulation-based systems inevitably fall short in the ability to incorporate signals from newly emerging geophysical observation systems [12], or take advantage of the Petabytes-scale Earth observation data [43].

As an appealing alternative, deep learning (DL) is offering a new approach for Earth system forecasting [34]. Instead of explicitly incorporating physical rules, DL-based forecasting models are trained on the Earth observation data [36]. By learning from a large amount of observations, DL models are able to figure out the system's intrinsic physical rules and generate predictions that outperform simulation-based models [9]. Such technique has demonstrated success in several applications, including precipitation nowcasting [32, 6] and ENSO forecasting [15]. Because the Earth system is chaotic [21], high-dimensional, and spatiotemporal, designing appropriate DL architecture for modeling the system is particularly challenging. Previous works relied on the combination of Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) [36, 37, 43, 13, 45]. These two architectures impose temporal and spatial inductive biases that help capturing spatiotemporal patterns. However, as a chaotic system, variabilities of the Earth system, such as rainfall and ENSO, are highly sensitive to the system's initial conditions and can respond abruptly to internal changes. It is unclear whether the inductive biases in RNN and CNN models still hold for such complex systems.

On the other hand, recent years have witnessed major breakthroughs in DL brought by the wide adoption of Transformer. The model was originally proposed for natural language processing [42, 7], and later has been extended to computer vision [8, 22], multimodal text-image generation [31], graph learning [52], etc. Transformer relies on the attention mechanism to capture data correlations and is powerful at modeling complex and long-range dependencies, both of which appear in Earth systems (See Fig. 1 for an example of Earth observation data). Despite being suitable for the problem, Transformer sees limited adoption for Earth system forecasting. Naively applying the Transformer architecture is infeasible because the $O(N^2)$ attention mechanism is too computationally expensive for the high-dimensional Earth observation data. How to design a space-time Transformer that is good at predicting the future of the Earth systems is largely an open problem to the community.

In this paper, we propose *Earthformer*, a space-time Transformer for Earth system forecasting. To better explore the design of space-time attention, we propose *Cuboid Attention*, which is a generic building block for efficient space-time attention. The idea is to decompose the input tensor to non-overlapping cuboids and apply cuboid-level self-attention in parallel. Since we limit the $O(N^2)$ self-attention inside the local cuboids, the overall complexity is greatly reduced. Different types of correlations can be captured via different cuboid decompositions. By stacking multiple cuboid attention layers with different hyperparameters, we are able to subsume several previously proposed video Transformers [19, 23, 4] as special cases, and also come up with new attention patterns that were not studied before. A limitation of this design is the lack of a mechanism for the local cuboids to communicate with each other. Thus, we introduce a collection of global vectors that attend to all the local cuboids, thereby gathering the overall status of the system. By attending to the global vectors, the local cuboids can grasp the general dynamics of the system and share information with each other.

To verify the effectiveness of cuboid attention and figure out the best design under the Earth system forecasting scenario, we conducted extensive experiments on two synthetic datasets: the MovingMNIST [36] dataset and a newly proposed N-body MNIST dataset. Digits in the N-body MNIST follow the chaotic 3-body motion pattern [25], which makes the dataset not only more challenging than MovingMNIST but also more relevant to Earth system forecasting. The synthetic experiments reveal the following findings: 1) stacking cuboid attention layers with the Axial attention pattern is both efficient and effective, achieving the best overall performance, 2) adding global vectors provides consistent performance gain without increasing the computational cost, 3) adding hierarchy in the encoder-decoder architecture can improve performance. Based on these findings, we figured out the optimal design for Earthformer and made comparisons with other baselines on the SEVIR [43]

benchmark for precipitation nowcasting and the ICAR-ENSO dataset [15] for ENSO forecasting. Experiments show that Earthformer achieves state-of-the-art (SOTA) performance on both tasks.

2 Related Work

Deep learning architectures for Earth system forecasting. Conventional DL models for Earth system forecasting are based on CNN and RNN. U-Net with either 2D CNN or 3D CNN have been used for precipitation nowcasting [43], Seasonal Arctic Sea ice prediction [1], and ENSO forecasting [15]. Shi et al. [36] proposed the ConvLSTM network that combines CNN and LSTM for precipitation nowcasting. Wang et al. [45] proposed PredRNN which adds the spatiotemporal memory flow structure to ConvLSTM. To better learn long-term high-level relations, Wang et al. [44] proposed E3D-LSTM that integrates 3D CNN to LSTM. To disentangle PDE dynamics from unknown complementary information, PhyDNet [13] incorporates a new recurrent physical cell to perform PDE-constrained prediction in latent space. Espeholt et al. [9] proposed MetNet-2 that outperforms HREF for forecasting precipitation. The architecture is based on ConvLSTM and dilated CNN. Very recently, there are works that tried to apply Transformer for solving Earth system forecasting problems. Pathak et al. [28] proposed the FourCastNet for global weather forecasting, which is based on Adaptive Fourier Neural Operators (AFNO) [14]. Bai et al. [3] proposed Rainformer for precipitation nowcasting, which is based on an architecture that combines CNN and Swin-Transformer [22]. In our experiments, we can see that Earthformer outperforms Rainformer.

Space-time Transformers for video modeling. Inspired by the success of ViT [8] for image classification, space-time Transformer is adopted for improved video understanding. In order to bypass the huge memory consumption brought by joint spatiotemporal attention, several pioneering work proposed efficient alternatives, such as divided attention [4], axial attention [19, 4], factorized encoder [27, 2] and separable attention [54]. Beyond minimal adaptation from ViT, some recent work introduced more prior to the design of space-time transformers, including trajectory [29], multi-scale [23, 11] and multi-view [49]. However, no prior work focuses on exploring the design of space-time Transformers for Earth system forecasting.

Global and local attention in vision Transformers. To make self-attention more efficient in terms of both memory consumption and speed, recent works have adapted the essence of CNN to perform local attention in transformers [16, 52]. HaloNets [41] develops a new self-attention model family consisting of simple local self-attention and convolutional hybrids, which outperform both CNN and vanilla ViT on a range of downstream vision tasks. GLiT [5] introduces a locality module and uses neural architecture search to find an efficient backbone. Focal transformer [51] proposes focal self-attention that can incorporate both fine-grained local and coarse-grained global interactions. However, these architectures are not directly applicable to spatiotemporal forecasting. Besides, they are also different from our design because we keep K global vectors to summarize the statistics of the dynamic system and connect the local cuboids; experiments show that such a global vector design is crucial to successful spatiotemporal forecasting.

3 Model

Similar to previous works [36, 43, 3], we formulate Earth system forecasting as a spatiotemporal sequence forecasting problem. The Earth observation data, such as radar echo maps from NEXRAD [17] and climate data from CIMP6 [10], are represented as a spatiotemporal sequence $[\mathcal{X}_i]_{i=1}^{T}$, $\mathcal{X}_i \in \mathbb{R}^{H \times W \times C_{\text{in}}}$. Based on these observations, the model predicts the K-step-ahead future $[\mathcal{Y}_{T+i}]_{i=1}^{K}$, $\mathcal{Y}_{T+i} \in \mathbb{R}^{H \times W \times C_{\text{out}}}$. Here, H, W denote the spatial resolution, and $C_{\text{in}}, C_{\text{out}}$ denote the number of measurements available at each space-time coordinate from the input and target sequences, respectively. As illustrated in Fig. 2, our proposed *Earthformer* is a hierarchical Transformer encoder-decoder based on *Cuboid Attention*. The input observations are encoded as a hierarchy of hidden states and then decoded to the prediction target. In what follows, we will present the detailed design of cuboid attention and the hierarchical encoder-decoder architecture adopted in Earthformer.

3.1 Cuboid Attention

Compared with images and text, spatiotemporal data in Earth systems usually have higher dimensionality. As a consequence, applying Transformers to this task is challenging. For example, for a 3D

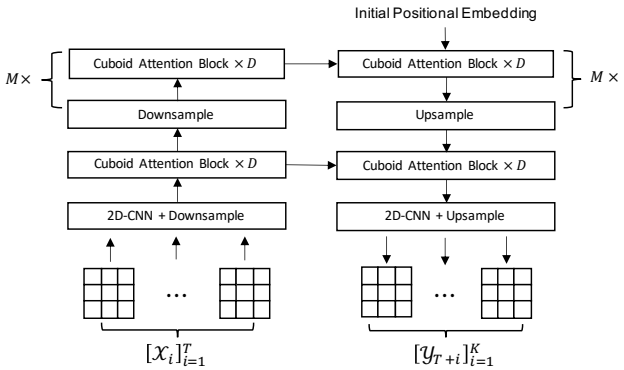


Figure 2: Illustration of the Earthformer architecture. It is a hierarchical Transformer encoder-decoder based on cuboid attention. The input sequence has length T and the target sequence has length K. "$\times D$" means to stack D cuboid attention blocks with residual connection. "$M\times$" means to have M layers of hierarchies.

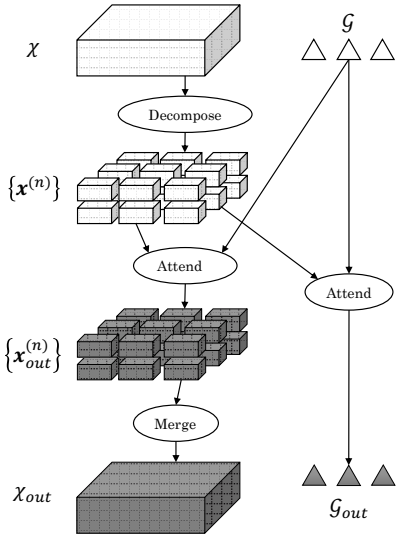
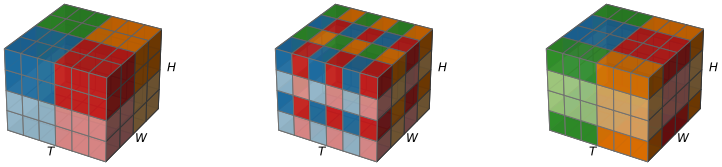


Figure 3: Illustration of the cuboid attention layer with global vectors.



(a) strategy="`local`" shift=$(0,0,0)$ (b) strategy="`dilated`" shift=$(0,0,0)$ (c) strategy="`local`" shift $=(0,1,1)$

Figure 4: Illustration of cuboid decomposition strategies when the input shape is $(T,H,W) = (6,4,4)$, and cuboid size $(b_T,b_H,b_W) = (3,2,2)$. Cells that have the same color belong to the same cuboid and will attend to each other. shift $=(0,1,1)$ shifts the cuboid decomposition by 1 pixel along height and width dimensions. strategy = "`local`" means to aggregate contiguous (b_T,b_H,b_W) pixels as a cuboid. strategy = "`dilated`" means to aggregate pixels every $\lceil\frac{T}{b_T}\rceil$ ($\lceil\frac{H}{b_H}\rceil$, $\lceil\frac{W}{b_W}\rceil$) steps along time (height, width) dimension. (Best viewed in color).

tensor with shape (T,H,W), the complexity of the vanilla self-attention is $O(T^2H^2W^2)$ and can be computationally infeasible. Previous literature proposed various structure-aware space-time attention mechanisms to reduce the complexity [19, 23, 4].

These space-time attention mechanisms share the common design of stacking multiple elementary attention layers that focus on different types of data correlations (e.g., temporal correlation and spatial correlation). Steming from this observation, we propose the generic cuboid attention layer that involves three steps: "decompose", "attend", and "merge".

Decompose. We first decompose the input spatiotemporal tensor $\mathcal{X}\in\mathbb{R}^{T\times H\times W\times C}$ into a sequence of cuboids $\{\boldsymbol{x}^{(n)}\}$.

$$\{\boldsymbol{x}^{(n)}\} = \texttt{Decompose}(\mathcal{X}, \text{cuboid_size}, \text{strategy}, \text{shift}), \tag{1}$$

where cuboid_size $=(b_T,b_H,b_W)$ is the size of the local cuboid, strategy $\in\{$"`local`", "`dilated`"$\}$ controls whether to adopt the local decomposition strategy or the dilated decomposition strategy [4], shift $=(s_T,s_H,s_W)$ is the window shift offset [22]. Fig. 4 provides three examples showing how an input tensor will be decomposed following different hyperparameters of `Decompose`$(\cdot)$. There are a total number of $\lceil\frac{T}{b_T}\rceil\lceil\frac{H}{b_H}\rceil\lceil\frac{W}{b_W}\rceil$ cuboids in $\{\boldsymbol{x}^{(n)}\}$. To simplify the notation, we assume that T,H,W are divisible by b_T,b_H,b_W. In the implementation, we pad the input tensor if it is not divisible.

Assume $\boldsymbol{x}^{(n)}$ is the (n_T, n_H, n_W)-th cuboid in $\{\boldsymbol{x}^{(n)}\}$. The (i, j, k)-th element of $\boldsymbol{x}^{(n)}$ can be mapped to the (i', j', k')-th element of $\mathcal{X}$ via Eqn. 2 if the strategy is "local" or Eqn. 3 if the strategy is "dilated".

$$\begin{aligned} i' &\leftrightarrow s_T + b_T(n_T - 1) + i \mod T \\ j' &\leftrightarrow s_H + b_H(n_H - 1) + j \mod H \\ k' &\leftrightarrow s_W + b_W(n_W - 1) + k \mod W \end{aligned} \tag{2}$$

$$\begin{aligned} i' &\leftrightarrow s_T + b_T(i - 1) + n_T \mod T \\ j' &\leftrightarrow s_H + b_H(j - 1) + n_H \mod H \\ k' &\leftrightarrow s_W + b_W(k - 1) + n_W \mod W \end{aligned} \tag{3}$$

Since the mapping is bijective, one can then map the elements from $\mathcal{X}$ to $\{\boldsymbol{x}^{(n)}\}$ via the inverse operation.

Attend. After decomposing the input tensor into a sequence of non-overlapping cuboids $\{\boldsymbol{x}^{(n)}\}$, we apply self-attention within each cuboid in parallel.

$$\boldsymbol{x}_{\text{out}}^{(n)} = \texttt{Attention}_{\Theta}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n)}), 1 \leq n \leq N. \tag{4}$$

The query, key, and value matrices $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ of $\texttt{Attention}_{\theta}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \texttt{Softmax}\left((\boldsymbol{W_Q Q})(\boldsymbol{W_K K})^T/\sqrt{C}\right)(\boldsymbol{W_V V})$ are all flattened versions of $\boldsymbol{x}^{(n)}$, and we unravel the resulting matrix back to a 3D tensor. $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$ and $\boldsymbol{W_V}$ are linear projection weights and are abbreviated together as Θ. The self-attention parameter Θ are shared across all cuboids. The computational complexity of the "attend" step is $O\left(\lceil \frac{T}{b_T} \rceil \lceil \frac{H}{b_H} \rceil \lceil \frac{W}{b_W} \rceil (b_T b_H b_W)^2\right) \approx O(THW \cdot b_T b_H b_W)$, which scales linearly with the cuboid size. Since the cuboid size can be much smaller than the size of the input tensor, the layer is more efficient than full attention.

Merge. $\texttt{Merge}(\cdot)$ is the inverse operation of $\texttt{Decompose}(\cdot)$. The sequence of cuboids obtained after the attention step $\{\boldsymbol{x}_{\text{out}}^{(n)}\}$ are merged back to the original input shape to produce the final output of cuboid attention, as shown in Eqn. 5. The mapping follows the same bijections in Eqn. 2 and Eqn. 3.

$$\mathcal{X}_{\text{out}} = \texttt{Merge}(\{\boldsymbol{x}_{\text{out}}^{(n)}\}_n, \text{cuboid_size}, \text{strategy}, \text{shift}). \tag{5}$$

We combine the "decompose", "attend" and "merge" steps described in Eqn. 1,4,5 to construct the generic cuboid attention as in Eqn. 6.

$$\mathcal{X}_{\text{out}} = \texttt{CubAttn}_{\Theta}(\mathcal{X}, \text{cuboid_size}, \text{strategy}, \text{shift}). \tag{6}$$

Explore cuboid attention patterns. By stacking multiple cuboid attention layers with different choices of "cuboid_size", "strategy" and "shift", we are able to efficiently explore existing and potentially more effective space-time attention. In this paper, we explore the cuboid attention patterns as listed in Table 1. From the table, we can see that cuboid attention subsumes previously proposed space-time attention methods like axial attention, video swin-Transformer, and divided space-time attention. Also, we manually picked the patterns that are reasonable and not computationally expensive as our search space. The flexibility of cuboid attention allows us to conduct Neural Architecture Search (NAS) to automatically search for a pattern but we will leave it as future work.

3.2 Global Vectors

One limitation of the previous formulation is that the cuboids do not communicate with each other. This is undesirable because each cuboid is not capable of understanding the global dynamics of the system. Thus, inspired by the `[CLS]` token adopted in BERT [7, 53], we propose to introduce a collection of P global vectors $\mathcal{G} \in \mathbb{R}^{P \times C}$ to help cuboids scatter and gather crucial global information. When each cuboid is performing the self-attention, the elements will not only attend to the other elements within the same cuboid but also attend to the global vectors $\mathcal{G}$. We revise Eqn. 4 to Eqn. 7 to enable local-global information exchange. We also use Eqn. 8 to update the global vectors $\mathcal{G}$ by aggregating the information from all elements of the input tensor $\mathcal{X}$.

$$\boldsymbol{x}_{\text{out}}^{(n)} = \texttt{Attention}_{\Theta}\left(\boldsymbol{x}^{(n)}, \texttt{Cat}(\boldsymbol{x}^{(n)}, \mathcal{G}), \texttt{Cat}(\boldsymbol{x}^{(n)}, \mathcal{G})\right), 1 \leq n \leq N. \tag{7}$$

$$\mathcal{G}_{\text{out}} = \texttt{Attention}_{\Phi}\left(\mathcal{G}, \texttt{Cat}(\mathcal{G}, \mathcal{X}), \texttt{Cat}(\mathcal{G}, \mathcal{X})\right). \tag{8}$$

Table 1: Configurations of the cuboid attention patterns explored in the paper. The input tensor has shape (T, H, W). If "shift" or "strategy" is not given, we use shift $= (0, 0, 0)$ and strategy $=$ "`local`" by default. When stacking multiple cuboid attention layers, each layer will be coupled with layer normalization layers and feed-forward network as in the Pre-LN Transformer [48]. The first row shows the configuration of the generic cuboid attention.

Name	Configurations	Values
Generic Cuboid Attention	cuboid_size shift strategy	$(\mathtt{T}_1, \mathtt{H}_1, \mathtt{W}_1) \rightarrow (\mathtt{T}_2, \mathtt{H}_2, \mathtt{W}_2) \rightarrow \cdots \rightarrow (\mathtt{T}_\mathtt{L}, \mathtt{H}_\mathtt{L}, \mathtt{W}_\mathtt{L})$ $(\mathtt{P}_1, \mathtt{M}_1, \mathtt{M}_1) \rightarrow (\mathtt{P}_2, \mathtt{M}_2, \mathtt{M}_2) \rightarrow \cdots \rightarrow (\mathtt{P}_\mathtt{L}, \mathtt{M}_\mathtt{L}, \mathtt{M}_\mathtt{L})$ "`loc./dil.`" $\rightarrow$ "`loc./dil.`" $\rightarrow \cdots \rightarrow$ "`loc./dil.`"
Axial	cuboid_size	$(\mathtt{T}, 1, 1) \rightarrow (1, \mathtt{H}, 1) \rightarrow (1, 1, \mathtt{W})$
Divided Space-Time	cuboid_size	$(\mathtt{T}, 1, 1) \rightarrow (1, \mathtt{H}, \mathtt{W})$
Video-Swin $P \times M$	cuboid_size shift	$(\mathtt{P}, \mathtt{M}, \mathtt{M}) \rightarrow (\ \mathtt{P}\ ,\ \mathtt{M}\ ,\ \mathtt{M}\ )$ $(0, 0, 0) \rightarrow (\mathtt{P}/2, \mathtt{M}/2, \mathtt{M}/2)$
Spatial Local-Dilate-M	cuboid_size strategy	$(\mathtt{T}, 1, 1) \rightarrow (1, \mathtt{M}, \mathtt{M}) \rightarrow (1, \mathtt{M}, \mathtt{M})$ "`local`" $\rightarrow$ "`local`" $\rightarrow$ "`dilated`"
Axial Space Dilate-M	cuboid_size strategy	$(\mathtt{T}, 1, 1) \rightarrow (1, \mathtt{H}/\mathtt{M}, 1) \rightarrow (1, \mathtt{H}/\mathtt{M}, 1) \rightarrow (1, 1, \mathtt{W}/\mathtt{M}) \rightarrow (1, 1, \mathtt{W}/\mathtt{M})$ "`local`" $\rightarrow$ "`dilated`" $\rightarrow$ "`local`" $\rightarrow$ "`dilated`" $\rightarrow$ "`local`"

Here, $\mathtt{Cat}(\cdot)$ flattens and concatenates its input tensors. By combining Eqn. 1,7,8,5, we abbreviate the overall computation of the cuboid attention layer with global vectors as in Eqn. 9.

$$\begin{aligned} \mathcal{X}_{\text{out}} &= \mathtt{CubAttn}_\Theta(\mathcal{X}, \mathcal{G}, \text{cuboid_size}, \text{strategy}, \text{shift}), \\ \mathcal{G}_{\text{out}} &= \mathtt{Attn}_\Phi^{\text{global}}(\mathcal{G}, \mathcal{X}). \end{aligned} \tag{9}$$

The additional complexity caused by the global vectors is approximately $O\left(THW \cdot P + P^2\right)$. Given that P is usually small (in our experiments, P is at most 8), the computational overhead induced by the global structure is negligible. The architecture of the cuboid attention layer is illustrated in Fig. 3.

3.3 Hierarchical Encoder-Decoder Architecture

Earthformer adopts a hierarchical encoder-decoder architecture illustrated in Fig. 2. The hierarchical architecture gradually encodes the input sequence to multiple levels of representations and generates the prediction via a coarse-to-fine procedure. Each hierarchy stacks D cuboid attention blocks. The cuboid attention block in the encoder uses one of the patterns described in Table 1, and each cuboid block in the decoder adopts the "Axial" pattern. To reduce the spatial resolution of the input to cuboid attention layers, we include a pair of initial downsampling and upsampling modules that consist of stacked 2D-CNN and Nearest Neighbor Interpolation (NNI) layers. Different from other papers that adopt Transformer for video prediction [19, 30], we generate the predictions in a non-auto-regressive fashion rather than an auto-regressive patch-by-patch fashion. This means that our decoder directly generates the predictions from the initial learned positional embeddings. We also conducted experiments with an auto-regressive decoder based on visual codebook [33]. However, the auto-regressive decoder underperforms the non-auto-regressive decoder in terms of forecasting skill scores. The comparison between non-auto-regressive decoder and auto-regressive decoder is shown in Appendix C.

4 Experiments

We first conducted experiments on two synthetic datasets, MovingMNIST and a newly proposed N-body MNIST, to verify the effectiveness of Earthformer and conduct ablation study on our design choices. Results on these two datasets lead to the following findings: 1) Among all patterns listed in Table 1, "Axial" achieves the best overall performance; 2) Global vectors bring consistent performance gain with negligible increase in computational cost; 3) Using a hierarchical coarse-to-fine structure can boost the performance. Based on these findings, we figured out the optimal design of Earthformer and compared it with other state-of-the-art models on two real-world datasets: SEVIR [43] and ICAR-ENSO[2]. On both datasets, Earthformer achieved the best overall performance. The statistics of all the datasets used in the experiments are shown in Table 2. We normalized the data to the range $[0, 1]$ and trained all the models with the Mean-Squared Error (MSE) loss. More implementation details are shown in Appendix A.

[2]Dataset available at `https://tianchi.aliyun.com/dataset/dataDetail?dataId=98942`

Table 2: Statistics of the datasets used in the experiments.

Dataset	Size			Seq. Len.		Spatial Resolution
	train	val	test	in	out	$H \times W$
MovingMNIST	8,100	900	1,000	10	10	64×64
N-body MNIST	20,000	1,000	1,000	10	10	64×64
SEVIR	35,718	9,060	12,159	13	12	384×384
ICAR-ENSO	5,205	334	1,667	12	14	24×48

Table 3: Ablation study on the importance of adopting a hierarchical encoder-decoder. We conducted experiments on MovingMNIST. "Depth D" means the model stacks D cuboid attention blocks and there is no hierarchical structure. "Depth $D1, D2$" means the model stacks $D1$ cuboid attention blocks, applies the pooling layer, and stacks another $D2$ cuboid attention blocks.

Model	#Param. (M)	GFLOPS	Metrics		
			MSE ↓	MAE ↓	SSIM ↑
Depth 2	1.4	17.9	63.80	140.6	0.8324
Depth 4	3.1	36.3	52.46	114.8	0.8685
Depth 6	4.9	54.6	50.49	110.0	0.8738
Depth 8	6.6	73.0	48.04	104.6	0.8797
Depth 1,1	1.4	11.5	60.99	135.7	0.8388
Depth 2,2	3.1	18.9	50.41	106.9	0.8805
Depth 3,3	4.9	26.3	47.69	**100.1**	**0.8873**
Depth 4,4	6.6	33.7	**46.91**	101.5	0.8825

4.1 Experiments on Synthetic Datasets

MovingMNIST. We follow [38] to use the public MovingMNIST dataset[3]. The dataset contains 10,000 sequences. Each sequence shows 2 digits moving inside a 64×64 frame. We split the dataset to use 8,100 samples for training, 900 samples for validation and 1,000 samples for testing. The task is to predict the future 10 frames for each sequence conditioned on the first 10 frames.

N-body MNIST. The Earth is a complex system in which an extremely large number of variables interact with each other. Compared with the Earth system, the dynamics of the synthetic MovingMNIST dataset, in which the digits move independently with constant speed, is over-simplified. Thus, achieving good performance on MovingMNIST does not imply that the model is capable of modeling complex interactions in the Earth system. On the other hand, the real-world Earth observation data, though experiencing rapid development, are still noisy and may not provide useful insights for model development. Therefore, we extend MovingMNIST to N-body MNIST, where N digits are moving with the N-body motion pattern inside a 64×64 frame. Each digit has its mass and is subjected to the gravity from other digits. We choose $N = 3$ in the experiments so that the digits will follow the chaotic 3-body motion [25]. The highly non-linear interactions in N-body MNIST make it much more challenging than the original MovingMNIST. We generate 20,000 sequences for training, 1,000 for validation and 1,000 for testing. Perceptual examples of the dataset can be found at the first two rows of Fig. 5. In Appendix D, we demonstrate the chaotic behavior of N-body MNIST.

Hierarchical v.s. non-hierarchical. We choose "Axial" without global vectors as our cuboid attention pattern and compare the performance of non-hierarchical and hierarchical architectures on MovingMNIST. The ablation study on the importance of adopting a hierarchical encoder-decoder is shown in Table 3. We can see that the hierarchical architecture has similar FLOPS with the non-hierarchical architectures while being better in MSE. This observation is consistent as we increase the depth until the performance saturates.

Cuboid pattern search. The design of cuboid attention greatly facilitates the search for optimal space-time attention. We compare the patterns listed in Table 1 on both MovingMNIST and N-body MNIST to investigate the effectiveness and efficiency of different space-time attention methods on spatiotemporal forecasting tasks. Besides the previously proposed space-time attention methods, we also include new configurations that are reasonable and not computationally expensive in our search space. For each pattern, we also compare the variant that uses global vectors. Results are summarized in Table 4. We find that the "Axial" pattern is both effective and efficient and adding global vectors improves performance for all patterns while having similar FLOPS. We thus pick "Axial + global" as the pattern in Earthformer when conducting experiments on real-world datasets.

[3]MovingMNIST: `https://github.com/mansimov/unsupervised-videos`

Table 4: Ablation study of different cuboid attention patterns and the effect of global vectors on MovingMNIST and N-body MNIST. The variant that achieved the best performance is in boldface while the second best is underscored. We also compared the performance of the cuboid attention patterns with and without global vectors and highlight the better one with grey background.

Model	#Param. (M)	GFLOPS	MovingMNIST			N-body MNIST		
			MSE ↓	MAE ↓	SSIM ↑	MSE ↓	MAE ↓	SSIM ↑
Axial	6.61	33.7	46.91	101.5	0.8825	15.89	<u>41.38</u>	<u>0.9510</u>
+ global ★	7.61	34.0	**41.79**	**92.78**	**0.8961**	**14.82**	**39.93**	**0.9538**
DST	5.70	35.2	57.43	118.6	0.8623	18.24	45.88	0.9435
+ global	6.37	35.5	52.92	108.3	0.8760	17.77	45.84	0.9433
Video Swin 2x8	5.66	31.1	54.45	111.7	0.8715	19.89	49.02	0.9374
+ global	6.33	31.4	52.70	108.5	0.8766	19.53	48.43	0.9389
Video Swin 10x8	5.89	39.2	63.34	125.3	0.8525	23.35	53.17	0.9274
+ global	6.56	39.4	62.15	123.4	0.8541	22.81	52.94	0.9293
Spatial Local-Global 2	6.61	33.3	59.88	122.1	0.8572	23.24	54.63	0.9263
+ global	7.61	33.7	59.42	122.9	0.8565	21.88	52.49	0.9305
Spatial Local-Global 4	6.61	33.5	58.72	118.5	0.8600	21.02	49.82	0.9344
+ global	7.61	33.9	54.84	115.5	0.8585	19.82	48.12	0.9371
Axial Space Dilate 2	8.59	41.8	50.11	104.4	0.8814	15.97	42.19	0.9494
+ global	10.30	42.4	46.86	98.95	0.8884	<u>15.73</u>	41.85	0.9510
Axial Space Dilate 4	8.59	41.6	47.40	99.31	0.8865	19.49	51.04	0.9352
+ global	10.30	42.2	<u>45.11</u>	<u>95.98</u>	<u>0.8928</u>	17.91	46.35	0.9440

Table 5: Comparison of Earthformer with baselines on MovingMNIST and N-body MNIST.

Model	#Param. (M)	GFLOPS	MovingMNIST			N-body MNIST		
			MSE ↓	MAE ↓	SSIM ↑	MSE ↓	MAE ↓	SSIM ↑
UNet [43]	16.6	0.9	110.4	249.4	0.6170	38.90	94.29	0.8260
ConvLSTM [36]	14.0	30.1	62.04	126.9	0.8477	32.15	72.64	0.8886
PredRNN [45]	23.8	232.0	52.07	108.9	<u>0.8831</u>	21.76	54.32	0.9288
PhyDNet [13]	3.1	15.3	58.70	124.1	0.8350	28.97	78.66	0.8206
E3D-LSTM [44]	12.9	302.0	55.31	101.6	0.8821	22.98	62.52	0.9131
Rainformer [3]	19.2	1.2	85.83	189.2	0.7301	38.89	96.47	0.8036
Earthformer w/o global	6.6	33.7	<u>46.91</u>	<u>101.5</u>	0.8825	<u>15.89</u>	<u>41.38</u>	<u>0.9510</u>
Earthformer	7.6	34.0	**41.79**	**92.78**	**0.8961**	**14.82**	**39.93**	**0.9538**

Comparison to the state of the art. We evaluate six spatiotemporal forecasting algorithms: UNet [43], ConvLSTM [36], PredRNN [45], PhyDNet [13], E3D-LSTM [44] and Rainformer [3]. The results are in Table 5. Note that the MovingMNIST performance on several papers [13] is obtained by training the model with on-the-fly generated digits while we pre-generate the digits and train all models on a fixed dataset. Comparing the numbers in the table with numbers shown in these papers are not fair. We train all baselines from scratch on both MovingMNIST and N-body MNIST using the default hyperparameters and configurations in their officially released code[4].

Qualitative results on N-body MNIST. Fig. 5 shows the generation results of different methods on a sample sequence from the N-body MNIST test set. The qualitative example demonstrates that our Earthformer is capable of learning long-range interactions among digits and correctly predicting their future motion trajectories. Also, we can see that Earthformer is able to more accurately predict the position of the digits with the help of global vectors. On the contrary, none of the baseline algorithms that achieved solid performance on MovingMNIST gives the correct and precise position of the digit "0" in the last frame. They either predict incorrect motion trajectories (PredRNN and E3D-LSTM), or generate highly blurry predictions (Rainformer, UNet and PhyDNet) to accommodate the uncertainty about the future.

4.2 SEVIR Precipitation Nowcasting

Storm EVent ImageRy (SEVIR) [43] is a spatiotemporally aligned dataset containing over 10,000 weather events. Each event consists of $384\ \texttt{km} \times 384\ \texttt{km}$ image sequences spanning over 4 hours. Images in SEVIR were sampled and aligned across five different data types: three channels (C02,

[4]Except for Rainformer which originally has 212M parameters and thus suffers from overfitting severely.

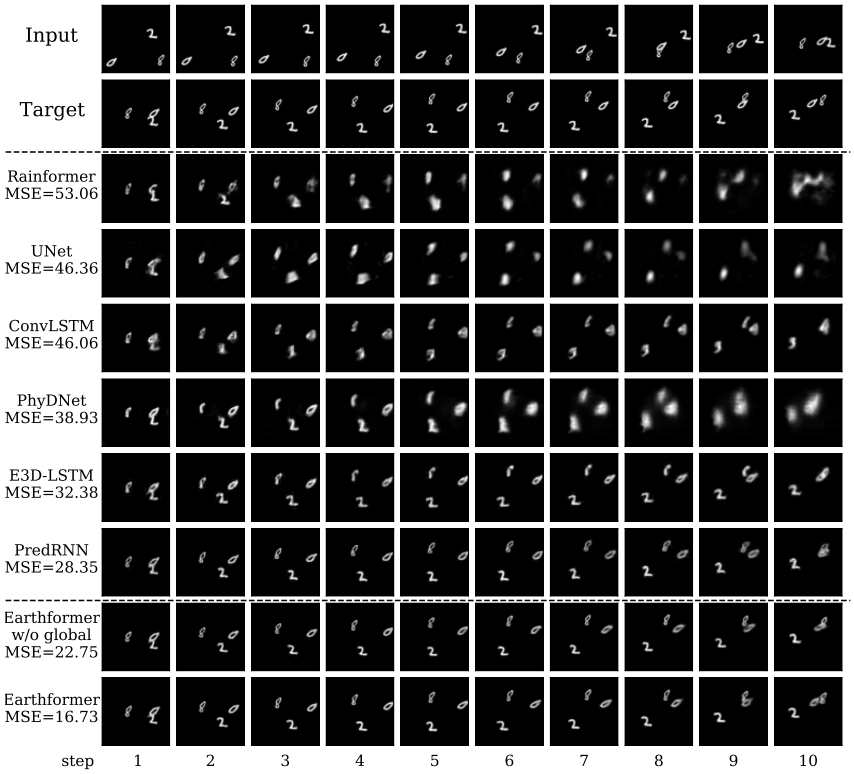


Figure 5: A set of examples showing the perceptual quality of the predictions on the N-body MNIST test set. From top to bottom: input frames, target frames, predictions by Rainformer [3], UNet [43], ConvLSTM [36], PhyDNet [13], E3D-LSTM [44], PredRNN [45], Earthformer without using global vectors, Earthformer. The results are sorted according to the MSE.

Table 6: Performance comparison on SEVIR. We include `Critical Success Index` (`CSI`) besides MSE as evaluation metrics. The `CSI`, a.k.a intersection over union (IOU), is calculated at different precipitation thresholds and denoted as `CSI`-$thresh$.

Model	#Param. (M)	GFLOPS	Metrics							
			CSI-M ↑	CSI-219 ↑	CSI-181 ↑	CSI-160 ↑	CSI-133 ↑	CSI-74 ↑	CSI-16 ↑	MSE (10^{-3}) ↓
Persistence	-	-	0.2613	0.0526	0.0969	0.1278	0.2155	0.4705	0.6047	11.5338
UNet [43]	16.6	33	0.3593	0.0577	0.1580	0.2157	0.3274	0.6531	0.7441	4.1119
ConvLSTM [36]	14.0	527	0.4185	0.1288	0.2482	0.2928	0.4052	0.6793	0.7569	3.7532
PredRNN [45]	46.6	328	0.4080	0.1312	0.2324	0.2767	0.3858	0.6713	0.7507	3.9014
PhyDNet [13]	13.7	701	0.3940	0.1288	0.2309	0.2708	0.3720	0.6556	0.7059	4.8165
E3D-LSTM [44]	35.6	523	0.4038	0.1239	0.2270	0.2675	0.3825	0.6645	0.7573	4.1702
Rainformer [3]	184.0	170	0.3661	0.0831	0.1670	0.2167	0.3438	0.6585	0.7277	4.0272
Earthformer w/o global	13.1	257	0.4356	0.1572	0.2716	0.3138	0.4214	0.6859	**0.7637**	3.7002
Earthformer	15.1	257	**0.4419**	**0.1791**	**0.2848**	**0.3232**	**0.4271**	**0.6860**	0.7513	**3.6957**

C09, C13) from the GOES-16 advanced baseline imager, NEXRAD Vertically Integrated Liquid (VIL) mosaics, and GOES-16 Geostationary Lightning Mapper (GLM) flashes. The SEVIR benchmark supports scientific research on multiple meteorological applications including precipitation nowcasting, synthetic radar generation, front detection, etc. We adopt SEVIR for benchmarking precipitation nowcasting, i.e., to predict the future VIL up to 60 minutes (12 frames) given 65 minutes context VIL (13 frames). Fig. 1 shows an example of VIL observation sequences in SEVIR.

Besides MSE, we also include the `Critical Success Index` (`CSI`), which is commonly used in precipitation nowcasting and is defined as $\texttt{CSI} = \frac{\#\texttt{Hits}}{\#\texttt{Hits}+\#\texttt{Misses}+\#\texttt{F.Alarms}}$. To count the $\#\texttt{Hits}$ (truth=1, pred=1), $\#\texttt{Misses}$ (truth=1, pred=0) and $\#\texttt{F.Alarms}$ (truth=0, pred=1), the prediction and the ground-truth are rescaled back to the range 0-255 and binarized at thresholds $[16, 74, 133, 160, 181, 219]$. We report `CSI` at different thresholds and also their mean `CSI-M`.

SEVIR is much larger than MovingMNIST and N-body MNIST and has higher resolution. We thus slightly adjust the configurations of baselines based on those for MovingMNIST for fair comparison. Detailed configurations are shown in Appendix A. The experiment results are listed in Table 6. Earthformer consistently outperforms baselines on almost all metrics and brings significant performance gain especially at high thresholds like `CSI`-219, which are more valued by the communities.

Table 7: Performance comparison on ICAR-ENSO. C-Nino3.4-M and C-Nino3.4-WM are the mean and the weighted mean of the correlation skill $\boldsymbol{C}^{\texttt{Nino3.4}}$ over $K = 12$ forecasting steps. C-Nino3.4-WM assigns more weights to longer-term prediction scores. MSE is calculated between the spatiotemporal SST anomalies prediction and the corresponding ground-truth.

Model	#Param. (M)	GFLOPS	Metrics		
			C-Nino3.4-M ↑	C-Nino3.4-WM ↑	MSE (10^{-4}) ↓
Persistence	-	-	0.3221	0.447	4.581
UNet [43]	12.1	0.4	0.6926	2.102	2.868
ConvLSTM [36]	14.0	11.1	0.6955	2.107	2.657
PredRNN [45]	23.8	85.8	0.6492	1.910	3.044
PhyDNet [13]	3.1	5.7	0.6646	1.965	2.708
E3D-LSTM [44]	12.9	99.8	0.7040	2.125	3.095
Rainformer [3]	19.2	1.3	0.7106	2.153	3.043
Earthformer w/o global	6.6	23.6	0.7239	2.214	2.550
Earthformer	7.6	23.9	**0.7329**	**2.259**	**2.546**

4.3 ICAR-ENSO Sea Surface Temperature Anomalies Forecasting

El Niño/Southern Oscillation (ENSO) has a wide range of associations with regional climate extremes and ecosystem impacts. ENSO sea surface temperature (SST) anomalies forecasting for lead times up to one year (12 steps) is a valuable and challenging problem. Nino3.4 index, which is the area-averaged SST anomalies across a certain area (170°-120°W, 5°S-5°N) of the Pacific, serves as a crucial indicator of this climate event. The forecast quality is evaluated by the correlation skill [15] of the three-month-moving-averaged Nino3.4 index $\boldsymbol{C}^{\texttt{Nino3.4}} = \frac{\sum_N (\boldsymbol{X}-\bar{\boldsymbol{X}})(\boldsymbol{Y}-\bar{\boldsymbol{Y}})}{\sqrt{\sum_N (\boldsymbol{X}-\bar{\boldsymbol{X}})^2 \sum_N (\boldsymbol{Y}-\bar{\boldsymbol{Y}})^2}} \in \mathbb{R}^K$ calculated on the whole test set of size N, where $\boldsymbol{Y} \in \mathbb{R}^{N \times K}$ is the ground-truth of K-step Nino3.4 index, $\boldsymbol{X} \in \mathbb{R}^{N \times K}$ is the corresponding prediction of Nino3.4 index.

ICAR-ENSO consists of historical climate observation and stimulation data provided by Institute for Climate and Application Research (ICAR). We forecast the SST anomalies up to 14 steps (2 steps more than one year for calculating three-month-moving-average) given a context of 12 steps of SST anomalies observations. Table 7 compares the performance of our Earthformer with baselines on the ICAR-ENSO dataset. We report the mean correlation skill $C\texttt{-Nino3.4-M} = \frac{1}{K}\sum_k C_k^{\texttt{Nino3.4}}$ and the weighted mean correlation skill $C\texttt{-Nino3.4-WM} = \frac{1}{K}\sum_k a_k \cdot C_k^{\texttt{Nino3.4}}$ over $K = 12$ forecasting steps[5], as well as the MSE between the spatiotemporal SST anomalies prediction and the corresponding ground-truth. We can find that Earthformer consistently outperforms the baselines in all concerned evaluation metrics and that using global vectors further improves the performance.

5 Conclusions and Broader Impact

In this paper, we propose Earthformer, a space-time Transformer for Earth system forecasting. Earthformer is based on a generic and efficient building block called *Cuboid Attention*. It achieves SOTA on MovingMNIST, our newly proposed N-body MNIST, SEVIR, and ICAR-ENSO.

Our work has certain limitations. The first one is that Earthformer is a deterministic model that does not model uncertainty. This may result in predicting the average of all plausible futures, causing the model to generate blurry predictions of low perceptual quality and be lack of valuable small-scale details. In fact, the community lacks appropriate metrics that measure the uncertainty component in Earth system forecasting models. Extending Earthformer to a probabilistic forecasting model can be an exciting future direction. We include more detailed discussions and preliminary experiments about handling uncertainty in Appendix C. The second one is that the model is purely data-driven and does not take advantage of the physical knowledge of the Earth system. Recent studies on adding physical constraints [20, 26] and ensembling the predictions from a data-driven model and a physics-based model [32] imply that it is an active and promising research direction to pursue. We plan to study how to incorporate physical knowledge into Earthformer in the future.

Acknowledgments and Disclosure of Funding

This work has been made possible by a Research Impact Fund project (R6003-21) and an Innovation and Technology Fund project (ITS/004/21FP) funded by the Hong Kong Government.

[5] $a_k = b_k \cdot \ln k$, where $b_k = 1.5$, for $k \leq 4$; $b_k = 2$, for $4 < k \leq 11$; $b_k = 3$, for $k > 11$.

References

[1] Tom R Andersson, J Scott Hosking, María Pérez-Ortiz, Brooks Paige, Andrew Elliott, Chris Russell, Stephen Law, Daniel C Jones, Jeremy Wilkinson, Tony Phillips, et al. Seasonal Arctic sea ice forecasting with probabilistic deep learning. *Nature communications*, 12(1):1–12, 2021.

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Luvcić, and Cordelia Schmid. ViViT: A video vision transformer. In *International Conference on Computer Vision (ICCV)*, 2021.

[3] Cong Bai, Feng Sun, Jinglin Zhang, Yi Song, and Shengyong Chen. Rainformer: Features extraction balanced network for radar-based precipitation nowcasting. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022.

[4] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding. *arXiv preprint arXiv:2102.05095*, 2(3):4, 2021.

[5] Boyu Chen, Peixia Li, Chuming Li, Baopu Li, Lei Bai, Chen Lin, Ming Sun, Junjie Yan, and Wanli Ouyang. GLiT: Neural architecture search for global and local image transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–21, 2021.

[6] Christian Schroeder de Witt, Catherine Tong, Valentina Zantedeschi, Daniele De Martini, Freddie Kalaitzis, Matthew Chantry, Duncan Watson-Parris, and Piotr Bilinski. RainBench: towards global precipitation forecasting from satellite imagery. In *AAAI*, 2021.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[9] Lasse Espeholt, Shreya Agrawal, Casper Sønderby, Manoj Kumar, Jonathan Heek, Carla Bromberg, Cenk Gazen, Jason Hickey, Aaron Bell, and Nal Kalchbrenner. Skillful twelve hour precipitation forecasts using large context neural networks. *arXiv preprint arXiv:2111.07470*, 2021.

[10] Veronika Eyring, Sandrine Bony, Gerald A Meehl, Catherine A Senior, Bjorn Stevens, Ronald J Stouffer, and Karl E Taylor. Overview of the coupled model intercomparison project phase 6 (CMIP6) experimental design and organization. *Geoscientific Model Development*, 9(5):1937–1958, 2016.

[11] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021.

[12] Steven J Goodman, Timothy J Schmit, Jaime Daniels, and Robert J Redmon. *The GOES-R series: a new generation of geostationary environmental satellites*. Elsevier, 2019.

[13] Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11474–11484, 2020.

[14] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021.

[15] Yoo-Geun Ham, Jeong-Hwan Kim, and Jing-Jia Luo. Deep learning for multi-year ENSO forecasts. *Nature*, 573(7775):568–572, 2019.

[16] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *NeurIPS*, 2021.

[17] William H Heiss, David L McGrew, and Dale Sirmans. NEXRAD: next generation weather radar (wsr-88d). *Microwave Journal*, 33(1):79–89, 1990.

[18] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[19] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[20] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

[21] Christophe Letellier. *Chaos in nature*, volume 94. World Scientific, 2019.

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[23] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021.

[24] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs created equal? a large-scale study. In *NeurIPS*, 2018.

[25] Valtonen MJ, Mauri Valtonen, and Hannu Karttunen. *The three-body problem*. Cambridge University Press, 2006.

[26] Geoffrey Négiar, Michael W Mahoney, and Aditi S Krishnapriyan. Learning differentiable solvers for systems with hard constraints. *arXiv preprint arXiv:2207.08675*, 2022.

[27] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. *arXiv preprint arXiv:2102.00719*, 2021.

[28] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. FourCastNet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[29] Mandela Patrick, Dylan Campbell, Yuki M. Asano, Ishan Misra Florian Metze, Christoph Feichtenhofer, Andrea Vedaldi, and João F. Henriques. Keeping your eye on the ball: Trajectory attention in video transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[30] Ruslan Rakhimov, Denis Volkhonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. Latent video transformer. *arXiv preprint arXiv:2006.10704*, 2020.

[31] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[32] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, et al. Skilful precipitation nowcasting using deep generative models of radar. *Nature*, 597(7878):672–677, 2021.

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

[34] Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, et al. Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195–204, 2019.

[35] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: A PixelCNN implementation with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

[36] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NeurIPS*, volume 28, 2015.

[37] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. In *NeurIPS*, volume 30, 2017.

[38] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using LSTMs. In *ICML*, pages 843–852. PMLR, 2015.

[39] Byron D Tapley, Srinivas Bettadpur, John C Ries, Paul F Thompson, and Michael M Watkins. GRACE measurements of mass variability in the earth system. *science*, 305(5683):503–505, 2004.

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

[41] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *CVPR*, 2021.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, volume 30, 2017.

[43] Mark Veillette, Siddharth Samsi, and Chris Mattioli. SEVIR: A storm event imagery dataset for deep learning applications in radar and satellite meteorology. *Advances in Neural Information Processing Systems*, 33:22009–22019, 2020.

[44] Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. Eidetic 3D LSTM: A model for video prediction and beyond. In *International conference on learning representations*, 2018.

[45] Yunbo Wang, Haixu Wu, Jianjin Zhang, Zhifeng Gao, Jianmin Wang, Philip Yu, and Mingsheng Long. PredRNN: A recurrent neural network for spatiotemporal predictive learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[46] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In *International Conference on Learning Representations*, 2019.

[47] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[48] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

[49] Shen Yan, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid. Multiview transformers for video recognition. In *CVPR*, 2022.

[50] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

[51] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. In *NeurIPS*, 2021.

[52] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34, 2021.

[53] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

[54] Yanyi Zhang, Xinyu Li, Chunhui Liu, Bing Shuai, Yi Zhu, Biagio Brattoli, Hao Chen, Ivan Marsic, and Joseph Tighe. Vidtr: Video transformer without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13577–13587, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Exploring Earth system forecasting has no negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Implementation Details

All experiments are conducted on machines with NVIDIA V100 GPUs. All models including Earthformer and the baselines can fit in a single GPU (with gradient checkpointing) and get trained without model parallelization.

A.1 Earthformer

Cuboid attention block. A cuboid attention block consists of L cuboid attention layers with different hyperparameters, as shown in 1. For example, as illustrated in the "Configuration Values" of Table 1, "Axial" has $L = 3$ since it has 3 consecutive patterns $(\mathtt{T}, 1, 1) \rightarrow (1, \mathtt{H}, 1) \rightarrow (1, 1, \mathtt{W})$, "Divided Space-Time" and "Video Swin $P \times M$" have $L = 2$. We also add Layer Normalization (LN) and Feed-Forward Networks (FFNs) in the same way as Pre-LN Transformer [48]. Given the input tensor $\mathcal{X}^{d-1}$ and global vectors $\mathcal{G}^{d-1}$, the output $(\mathcal{X}^d, \mathcal{G}^d)$ of the d-th cuboid attention block (there are D cuboid attention blocks in total for each hierarchy as illustrated in Fig. 2), is computed as Eqn. 10:

$$
\begin{aligned}
\mathcal{X}^{d,1} &= \mathtt{FFN}_{d,1}^{\mathtt{local}}\left(\mathcal{X}^{d-1} + \mathtt{CubAttn}_{d,1}^{\mathtt{local}}\left(\mathtt{LN}_{d,1}^{\mathtt{local}}(\mathcal{X}^{d-1}), \mathtt{LN}_{d,1}^{\mathtt{global}}(\mathcal{G}^{d-1})\right)\right), \\
\mathcal{G}^{d,1} &= \mathtt{FFN}_{d,1}^{\mathtt{global}}\left(\mathcal{G}^{d-1} + \mathtt{Attn}_{d,1}^{\mathtt{global}}\left(\mathtt{LN}_{d,1}^{\mathtt{global}}(\mathcal{G}^{d-1}), \mathtt{LN}_{d,1}^{\mathtt{local}}(\mathcal{X}^{d-1})\right)\right), \\
&\vdots \\
\mathcal{X}^{d,l} &= \mathtt{FFN}_{d,l}^{\mathtt{local}}\left(\mathcal{X}^{d,l-1} + \mathtt{CubAttn}_{d,l}^{\mathtt{local}}\left(\mathtt{LN}_{d,l}^{\mathtt{local}}(\mathcal{X}^{d,l-1}), \mathtt{LN}_{d,l}^{\mathtt{global}}(\mathcal{G}^{d,l-1})\right)\right), \\
\mathcal{G}^{d,l} &= \mathtt{FFN}_{d,l}^{\mathtt{global}}\left(\mathcal{G}^{d,l-1} + \mathtt{Attn}_{d,l}^{\mathtt{global}}\left(\mathtt{LN}_{d,l}^{\mathtt{global}}(\mathcal{G}^{d,l-1}), \mathtt{LN}_{d,l}^{\mathtt{local}}(\mathcal{X}^{d,l-1})\right)\right), \\
&\vdots \\
\mathcal{X}^{d} = \mathcal{X}^{d,L} &= \mathtt{FFN}_{d,L}^{\mathtt{local}}\left(\mathcal{X}^{d,L-1} + \mathtt{CubAttn}_{d,L}^{\mathtt{local}}\left(\mathtt{LN}_{d,L}^{\mathtt{local}}(\mathcal{X}^{d,L-1}), \mathtt{LN}_{d,L}^{\mathtt{global}}(\mathcal{G}^{d,L-1})\right)\right), \\
\mathcal{G}^{d} = \mathcal{G}^{d,L} &= \mathtt{FFN}_{d,L}^{\mathtt{global}}\left(\mathcal{G}^{d,L-1} + \mathtt{Attn}_{d,L}^{\mathtt{global}}\left(\mathtt{LN}_{d,L}^{\mathtt{global}}(\mathcal{G}^{d,L-1}), \mathtt{LN}_{d,L}^{\mathtt{local}}(\mathcal{X}^{d,L-1})\right)\right).
\end{aligned}
\tag{10}
$$

Here, $\mathtt{FFN}_{d,l}^{\mathtt{local}}, \mathtt{LN}_{d,l}^{\mathtt{local}}$ means the FFN an LN layers applied on the tensor $\mathcal{X}^{d,l-1}$, and $\mathtt{FFN}_{d,l}^{\mathtt{global}}, \mathtt{LN}_{d,l}^{\mathtt{global}}$ means those applied on the global vectors $\mathcal{G}^{d,l-1}$. $\mathtt{CubAttn}_{d,l}^{\mathtt{local}}$ is the combination of "decompose", "attend" and "merge" pipeline described in Eqn. 1,7,5. We omit cuboid_size, strategy and shift for brevity. $\mathtt{Attn}_{d,l}^{\mathtt{global}}$ is equivalent to Eqn. 8. For the l-th cuboid pattern in a cuboid attention block, the input $\mathcal{X}^{d,l-1}$ and $\mathcal{G}^{d,l-1}$ go through the LN layer, cuboid attention layer and FFN layer with residual connections sequentially, produce the output $\mathcal{X}^{d,l}$ and $\mathcal{G}^{d,l}$. The output of the final cuboid pattern $\mathcal{X}^{d,L}$ and $\mathcal{G}^{d,L}$ serve as the output of the whole cuboid attention block $\mathcal{X}^d$ and $\mathcal{G}^d$.

Hyperparameters of Earthformer architecture. The detailed architecture configurations of Earthformer are described in Table 8 and Table 9. We adopt the same configurations for MovingMNIST, N-body MNIST and ICAR-ENSO datasets, as shown in Table 8. We slightly adjust the configurations on SEVIR as shown in Table 9, due to its high resolution and large dataset size.

Optimization. We train all Earthformer variants using the AdamW optimizer. Detailed configurations are shown in Table 10. We train for 100 epochs on all datasets and early-stop model training according to the validation score with $\mathtt{tolerance} = 20$. We adopt 20% linear warm-up and `Cosine` learning rate scheduler that decays the lr from its maximum to zero after warm-up. We adopt data parallel and gradient accumulation to use a total batch size of 64 while the 16GB GPU can only afford a smaller batch size like 2 or 4.

A.2 Baselines

We train baseline algorithms following their officially released configurations and tune the learning rate, learning rate scheduler, working resolution, etc., to optimize their performance on each dataset. We list the modifications we applied to the baselines for each dataset in Table 11.

B More Ablation Analysis on Synthetic Datasets

As mentioned in Sec. 4.1, the design of cuboid attention greatly facilitates the search for optimal space-time attention. Table 4 summarizes the cuboid attention pattern search results with the model depth equal to 4 on both MovingMNIST and N-body MNIST. We further investigate if it is feasible to accelerate the pattern search using shallower models. The results with model depth equal to 2 are shown in Table 12. We find that the "Axial" pattern is still the optimal among the patterns listed in Table 1, and adding global vectors improves performance for all the patterns while having similar FLOPS. This observation implies that in the future work, it might be feasible to conduct NAS using smaller and shallower models with affordable cost and transfer the results to larger and deeper models.

Table 8: The details of the Earthformer model on MovingMNIST, N-body MNIST and ICAR-ENSO datasets. `Conv3` $\times$ `3` is the 2D convolutional layer with 3×3 kernel. `GroupNorm16` is the Group Normalization (GN) layer [47] with 16 groups. The negative slope in `LeakyReLU` is 0.1. The `FFN` consists of two `Linear` layers separated by a `GeLU` activation layer [18]. `PatchMerge` splits a 2D input tensor with C channels into N non-overlapping $p \times p$ patches and merges the spatial dimensions into channels, gets N 1×1 patches with $p^2 \cdot C$ channels and concatenates them back along spatial dimensions.

Block	Layer	Resolution	Channels
Input	-	64×64	1
2D CNN+Downsampler	`Conv3` $\times$ `3`	64×64	$1 \rightarrow 64$
	`GroupNorm16`	64×64	64
	`LeakyReLU`	64×64	64
	`PatchMerge`	$64 \times 64 \rightarrow 32 \times 32$	$64 \rightarrow 256$
	`LayerNorm`	32×32	256
	`Linear`	32×32	$256 \rightarrow 64$
Encoder Positional Embedding	`PosEmbed`	32×32	64
Cuboid Attention Block $\times 4$	`LayerNorm`	32×32	64
	`Cuboid(T, 1, 1)`	32×32	64
	`FFN`	32×32	64
	`LayerNorm`	32×32	64
	`Cuboid(1, H, 1)`	32×32	64
	`FFN`	32×32	64
	`LayerNorm`	32×32	64
	`Cuboid(1, 1, W)`	32×32	64
	`FFN`	32×32	64
Downsampler	`PatchMerge`	$32 \times 32 \rightarrow 16 \times 16$	$64 \rightarrow 256$
	`LayerNorm`	16×16	256
	`Linear`	16×16	$256 \rightarrow 128$
Cuboid Attention Block $\times 4$	`LayerNorm`	16×16	128
	`Cuboid(T, 1, 1)`	16×16	128
	`FFN`	16×16	128
	`LayerNorm`	16×16	128
	`Cuboid(1, H, 1)`	16×16	128
	`FFN`	16×16	128
	`LayerNorm`	16×16	128
	`Cuboid(1, 1, W)`	16×16	128
	`FFN`	16×16	128
Decoder Initial Positional Embedding	`PosEmbed`	16×16	128
Cuboid Cross Attention Block $\times 4$	`LayerNorm`	16×16	128
	`Cuboid(T, 1, 1)`	16×16	128
	`FFN`	16×16	128
	`LayerNorm`	16×16	128
	`Cuboid(1, H, 1)`	16×16	128
	`FFN`	16×16	128
	`LayerNorm`	16×16	128
	`Cuboid(1, 1, W)`	16×16	128
	`FFN`	16×16	128
	`CuboidCross(T, 1, 1)`	16×16	128
	`FFN`	16×16	128
	`LayerNorm`	16×16	128
Upsampler	`NearestNeighborInterp`	$16 \times 16 \rightarrow 32 \times 32$	128
	`Conv3` $\times$ `3`	32×32	$128 \rightarrow 64$
Cuboid Cross Attention Block $\times 4$	`LayerNorm`	32×32	64
	`Cuboid(T, 1, 1)`	32×32	64
	`FFN`	32×32	64
	`LayerNorm`	32×32	64
	`Cuboid(1, H, 1)`	32×32	64
	`FFN`	32×32	64
	`LayerNorm`	32×32	64
	`Cuboid(1, 1, W)`	32×32	64
	`FFN`	32×32	64
	`CuboidCross(T, 1, 1)`	32×32	64
	`FFN`	32×32	64
	`LayerNorm`	32×32	64
2D CNN+Upsampler	`NearestNeighborInterp`	$32 \times 32 \rightarrow 64 \times 64$	64
	`Conv3` $\times$ `3`	64×64	64
	`GroupNorm16`	64×64	64
	`LeakyReLU`	64×64	64
	`Linear`	64×64	$64 \rightarrow 1$

Table 9: The details of the Earthformer model on SEVIR dataset. `Conv3` $\times$ `3` is the 2D convolutional layer with 3×3 kernel. `GroupNorm16` is the Group Normalization (GN) layer [47] with 16 groups. The negative slope in `LeakyReLU` is 0.1. The `FFN` consists of two `Linear` layers separated by a `GeLU` activation layer [18]. `PatchMerge` splits a 2D input tensor with C channels into N non-overlapping $p \times p$ patches and merges the spatial dimensions into channels, gets N 1×1 patches with $p^2 \cdot C$ channels and concatenates them back along spatial dimensions.

Block	Layer	Resolution	Channels
Input	-	384×384	1
2D CNN+Downsampler	`Conv3` $\times$ `3`	384×384	$1 \rightarrow 16$
	`Conv3` $\times$ `3`	384×384	16
	`GroupNorm16`	384×384	16
	`LeakyReLU`	384×384	16
	`PatchMerge`	$384 \times 384 \rightarrow 128 \times 128$	$16 \rightarrow 144$
	`LayerNorm`	128×128	144
	`Linear`	128×128	$144 \rightarrow 16$
2D CNN+Downsampler	`Conv3` $\times$ `3`	128×128	$16 \rightarrow 64$
	`Conv3` $\times$ `3`	128×128	64
	`GroupNorm16`	128×128	64
	`LeakyReLU`	128×128	64
	`PatchMerge`	$128 \times 128 \rightarrow 64 \times 64$	$64 \rightarrow 256$
	`LayerNorm`	64×64	256
	`Linear`	64×64	$256 \rightarrow 64$
2D CNN+Downsampler	`Conv3` $\times$ `3`	64×64	$64 \rightarrow 128$
	`Conv3` $\times$ `3`	64×64	128
	`GroupNorm16`	64×64	128
	`LeakyReLU`	64×64	128
	`PatchMerge`	$64 \times 64 \rightarrow 32 \times 32$	$128 \rightarrow 512$
	`LayerNorm`	32×32	512
	`Linear`	32×32	$512 \rightarrow 128$
Encoder Positional Embedding	`PosEmbed`	32×32	128
Cuboid Attention Block $\times 2$	`LayerNorm`	32×32	128
	`Cuboid(T, 1, 1)`	32×32	128
	`FFN`	32×32	128
	`LayerNorm`	32×32	128
	`Cuboid(1, H, 1)`	32×32	128
	`FFN`	32×32	128
	`LayerNorm`	32×32	128
	`Cuboid(1, 1, W)`	32×32	128
	`FFN`	32×32	128
Downsampler	`PatchMerge`	$32 \times 32 \rightarrow 16 \times 16$	$128 \rightarrow 512$
	`LayerNorm`	16×16	512
	`Linear`	16×16	$512 \rightarrow 256$
Cuboid Attention Block $\times 2$	`LayerNorm`	16×16	256
	`Cuboid(T, 1, 1)`	16×16	256
	`FFN`	16×16	256
	`LayerNorm`	16×16	256
	`Cuboid(1, H, 1)`	16×16	256
	`FFN`	16×16	256
	`LayerNorm`	16×16	256
	`Cuboid(1, 1, W)`	16×16	256
	`FFN`	16×16	256
Decoder Initial Positional Embedding	`PosEmbed`	16×16	256
Cuboid Cross Attention Block $\times 2$	`LayerNorm`	16×16	256
	`Cuboid(T, 1, 1)`	16×16	256
	`FFN`	16×16	256
	`LayerNorm`	16×16	256
	`Cuboid(1, H, 1)`	16×16	256
	`FFN`	16×16	256
	`LayerNorm`	16×16	256
	`Cuboid(1, 1, W)`	16×16	256
	`FFN`	16×16	256
	`CuboidCross(T, 1, 1)`	16×16	256
	`FFN`	16×16	256
	`LayerNorm`	16×16	256
Upsampler	`NearestNeighborInterp`	$16 \times 16 \rightarrow 32 \times 32$	256
	`Conv3` $\times$ `3`	32×32	$256 \rightarrow 128$
Cuboid Cross Attention Block $\times 2$	`LayerNorm`	32×32	128
	`Cuboid(T, 1, 1)`	32×32	128
	`FFN`	32×32	128
	`LayerNorm`	32×32	128
	`Cuboid(1, H, 1)`	32×32	128
	`FFN`	32×32	128
	`LayerNorm`	32×32	128
	`Cuboid(1, 1, W)`	32×32	128
	`FFN`	32×32	128
	`CuboidCross(T, 1, 1)`	32×32	128
	`FFN`	32×32	128
	`LayerNorm`	32×32	128
2D CNN+Upsampler	`NearestNeighborInterp`	$32 \times 32 \rightarrow 64 \times 64$	128
	`Conv3` $\times$ `3`	64×64	128
	`GroupNorm16`	64×64	128
	`LeakyReLU`	64×64	128
2D CNN+Upsampler	`NearestNeighborInterp`	$64 \times 64 \rightarrow 128 \times 128$	128
	`Conv3` $\times$ `3`	128×128	$128 \rightarrow 64$
	`GroupNorm16`	128×128	64
	`LeakyReLU`	128×128	64
2D CNN+Upsampler	`NearestNeighborInterp`	$128 \times 128 \rightarrow 384 \times 384$	64
	`Conv3` $\times$ `3`	384×384	$64 \rightarrow 16$
	`GroupNorm16`	384×384	16
	`LeakyReLU`	384×384	16
	`Linear`	384×384	$16 \rightarrow 1$

Table 10: Hyperparameters of the AdamW optimizer for training Earthformer on MovingMNIST, N-body MNIST, SEVIR and ICAR-ENSO datasets.

Hyper-parameter	Value
Learning rate	0.001
β_1	0.9
β_2	0.999
Weight decay	0.00001
Batch size	64
Training epochs	100
Warm up percentage	20%
Learning rate decay	Cosine
Early stop	True
Early stop tolerance	20

Table 11: Implementation details of baseline algorithms. Modifications based on the officially released implementations are listed according to different datasets. "-" means no modification is applied. "reverse enc-dec" means adopting the reversed encoder-decoder architecture proposed in [37]. "2D CNN downsampler/upsampler ($8\times$) wrapper" means we wrap the model with a pair of 2D CNN downsampler and upsampler to downsample the spatial resolution 8 times in order to reduce the GPU memory cost as well as the FLOPS. The 2D CNN downsampler and upsampler are of the same designs as those used in Earthformer. Other terms listed are the hyperparameters in their officially released implementations.

Model	MovingMNIST	N-body MNIST	SEVIR	ICAR-ENSO
UNet [43]	-	-	-	num_layer=3
ConvLSTM [36]	reverse enc-dec [37] conv_kernels = [(7,7),(5,5),(3,3)] deconv_kernels = [(6,6),(4,4),(4,4)] channels=[96, 128, 256]	reverse enc-dec [37] conv_kernels = [(7,7),(5,5),(3,3)] deconv_kernels = [(6,6),(4,4),(4,4)] channels=[96, 128, 256]	reverse enc-dec [37] conv_kernels = [(7,7),(5,5),(3,3)] deconv_kernels = [(6,6),(4,4),(4,4)] channels=[96, 128, 256]	reverse enc-dec [37] conv_kernels = [(7,7),(5,5),(3,3)] deconv_kernels = [(6,6),(4,4),(4,4)] channels=[96, 128, 256]
PredRNN [45]	-	-	2D CNN downsampler/upsampler ($8\times$) wrapper	-
PhyDNet [13]	-	-	convcell_hidden = [256, 256, 256, 64]	-
E3D-LSTM [44]	-	-	2D CNN downsampler/upsampler ($8\times$) wrapper	-
Rainformer [3]	downscaling_factors=[2, 2, 2, 2] hidden_dim=32 heads=[4, 4, 8, 16] head_dim=8 -	downscaling_factors=[2, 2, 2, 2] hidden_dim=32 heads=[4, 4, 8, 16] head_dim=8 -	downscaling_factors=[4, 2, 2, 2] - - - -	downscaling_factors=[1, 2, 2, 2] hidden_dim=32 heads=[4, 4, 8, 16] head_dim=8 window_size=3

Table 12: Ablation results with depth equals to 2. We can see that the findings still hold for shallower models.

Model	Metrics on MovingMNIST			Metrics on N-body MNIST		
	MSE ↓	MAE ↓	SSIM ↑	MSE ↓	MAE ↓	SSIM ↑
Axial	50.41	106.9	0.8805	18.98	48.49	0.9391
+ global ★	**49.74**	105.2	**0.8838**	**18.49**	47.31	0.9416
DST	56.43	119.5	0.8620	21.47	54.06	0.9290
+ global	55.87	118.2	0.8635	21.16	53.03	0.9303
Video-Swin 2×8	59.31	124.4	0.8522	23.04	56.66	0.9226
+ global	60.05	125.9	0.8522	22.97	55.90	0.9241
Video-Swin 10×8	66.68	133.2	0.8387	25.91	62.49	0.9100
+ global	65.58	132.8	0.8383	25.56	61.52	0.9118
Spatial Local-Dilated 2	62.85	131.3	0.8441	25.98	64.24	0.9075
+ global	60.03	127.2	0.8489	24.90	60.13	0.9162
Spatial Local-Dilated 4	59.87	125.2	0.8514	23.68	57.11	0.9207
+ global	58.08	120.4	0.8611	22.93	56.22	0.9236
Axial Space Dilate 2	53.12	110.4	0.8747	19.52	48.77	0.9378
+ global	51.03	107.4	0.8782	18.53	**47.00**	**0.9419**
Axial Space Dilate 4	55.85	116.0	0.8645	20.30	50.51	0.9350
+ global	49.84	**104.9**	0.8815	19.38	49.32	0.9380

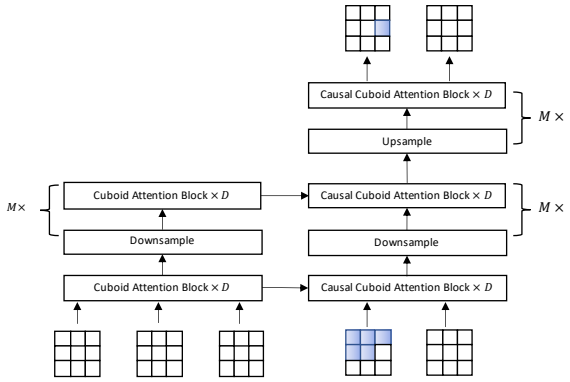


Figure 6: Illustration of the Earthformer variant in auto-regressive fashion (Earthformer AR). It is a hierarchical Transformer encoder-decoder based on cuboid attention. The input is a length-3 sequence of frames containing discrete visual codes and the target sequence has length 2. "$\times D$" means stacking D (causal) cuboid attention blocks with residual connection. "$M\times$" means M layers of hierarchies. A causal cuboid attention block is a cuboid attention block with causal mask. The "Downsample" and "Upsample" layers in the decoder follow the designs described in [35] and are hence also causal. Causal cuboid attention blocks and causal downsampling and upsampling layers prevent each element attending to the elements with larger indices in raster-scan ordering.

C Non-Auto-Regressive v.s. Auto-Regressive

As mentioned in Sec. 3.3, besides generating the future predictions directly in a non-auto-regressive way, previous works generate the predictions patch-by-patch in raster-scan ordering [46, 50, 30]. In fact, the pros and cons of the auto-regressive and non-auto-regressive approaches are not clear under the setting of Earth system forecasting. Therefore, we replace the decoder of the non-auto-regressive Earthformer with the auto-regressive decoder shown in Fig. 6 and propose an Earthformer variant called *Earthformer AR*.

The auto-regressive approach uses discrete visual tokens as the input and target. Here, we pretrain a VQ-VAE [40] with a codebook $e \in \mathbb{R}^{Q\times C^{\text{code}}}$, where Q is the codebook size. The discrete latent space is hence Q-way categorical. The VQ-VAE encoder downsamples an input frame $\mathcal{X}_i \in \mathbb{R}^{H\times W\times C}$ to $\texttt{Enc}(\mathcal{X}_i) \in \mathbb{R}^{H'\times W'\times C^{\text{code}}}$, then maps each element to its nearest code e_i learned in the codebook, and produces the encoding $\texttt{Enc}^{\text{code}}(\mathcal{X}_i) \in [Q]^{H'\times W'\times 1}$. The decoder does it reversely: it takes the latent visual codes $\mathcal{Z}_i^{\text{code}} \in [Q]^{H'\times W'\times 1}$, indexes the codes in the codebook to convert the discrete codes to real-valued vectors $\mathcal{Z}_i \in \mathbb{R}^{H'\times W'\times C^{\text{code}}}$, and then upsamples the frame back to the pixel space $\mathcal{Y}_i \in \mathbb{R}^{H\times W\times C}$.

Fig. 6 illustrates the architecture of the Earthformer AR. The input is a sequence of frames containing discrete visual codes $[\texttt{Enc}^{\text{code}}(\mathcal{X}_i)]_i$, which are the output of the VQ-VAE encoder. The target sequence is $[\texttt{Enc}^{\text{code}}(\mathcal{Y}_i)]_i$. Earthformer AR generates the target code one-by-one in raster-scan ordering. The generation at the current step $\mathcal{Z}^{\text{code}}_{(a,b,c)}$ is conditioned on the context $\mathcal{X}$ and all the previously generated codes $[\mathcal{Z}^{\text{code}}_{<(a,b,c)}]$:

$$p(\mathcal{Z}^{\text{code}} \mid \mathcal{X}) = \prod_{(a,b,c)} p\left(\mathcal{Z}^{\text{code}}_{(a,b,c)} \mid \mathcal{Z}^{\text{code}}_{<(a,b,c)}, \mathcal{X}\right). \tag{11}$$

The VQ-VAE decoder decodes the generated visual codes $\mathcal{Z}^{\text{code}}$ to the final output in the pixel space.

We conduct preliminary experiments of Earthformer AR on the SEVIR dataset and find that Earthformer AR is able to give more perceptually satisfying predictions than Earthformer, but is inferior in terms of skill scores. Fig. 7 to Fig. 10 show the qualitative results of Earthformer and Earthformer AR on several challenging test cases. The outputs of Earthformer AR look more like "real" VIL images and do not suffer from producing blurry predictions. However, the performance of Earthformer AR is much worse than Earthformer (even worse than some simple baselines including

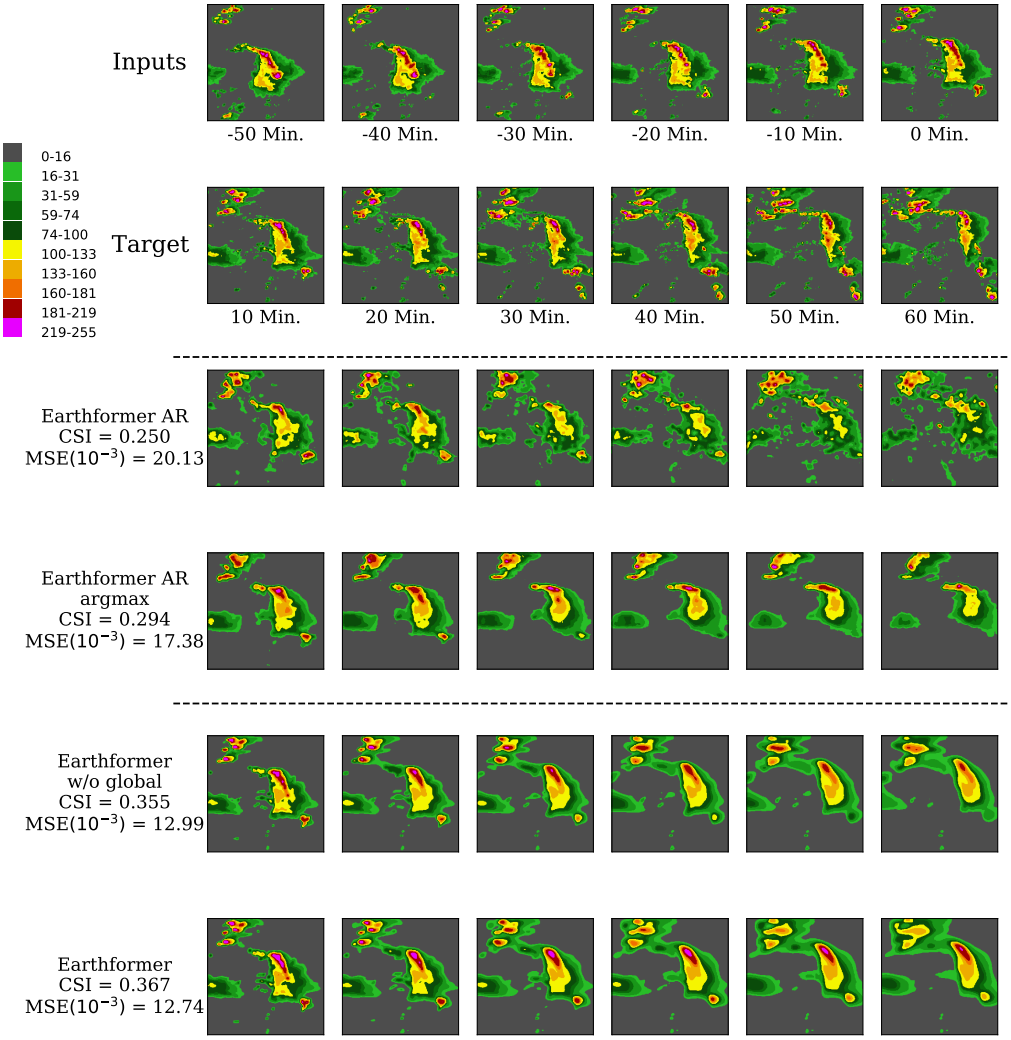


Figure 7: Non-Auto-Regressive v.s. Auto-Regressive on SEVIR: qualitative example 1.

UNet [43]) in the concerned evaluation metrics. We further investigate the effect of the sampling algorithm for Earthformer AR. We compared 1) generating the codes with `argmax` step-by-step and 2) randomly drawing samples from $p\left(\mathcal{Z}^{\text{code}}_{(a,b,c)} | \mathcal{Z}^{\text{code}}_{<(a,b,c)}, \mathcal{X}_i\right)$ specified by Eqn. 11. We denote the `argmax`-variant as *Earthformer AR argmax*. We find that argmax sampling can give better skill scores but the generated results have less perceptual similarity than the "real" VIL images.

Based on these results, we pick the non-auto-regressive decoder for experiments in the main paper. The fact that Earthformer AR gives more perceptually satisfying predictions than Earthformer while having worse skill scores triggers future work. So far, there is no well-established metric for evaluating the preceptual quality of the predictions generated by Earth system forecasting models. The Fréchet Inception Distance (FID) and Inception Score (IS) adopted in evaluating image GANs [24] rely on a pretrained network. We are able to pretrain Earthformer on Earth observations and adopt the pretrained network for both calculating the FID and initializing the GAN discriminator.

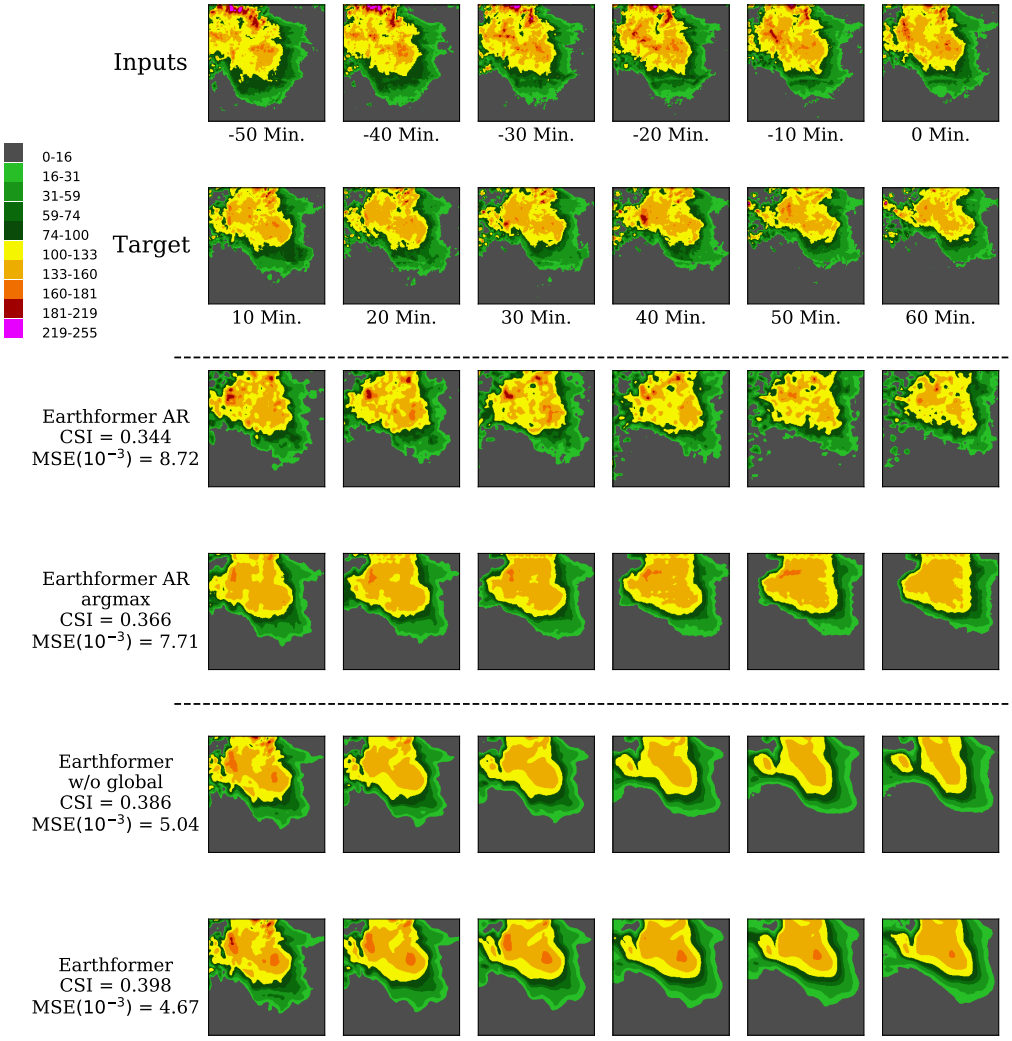


Figure 8: Non-Auto-Regressive v.s. Auto-Regressive on SEVIR: qualitative example 2.

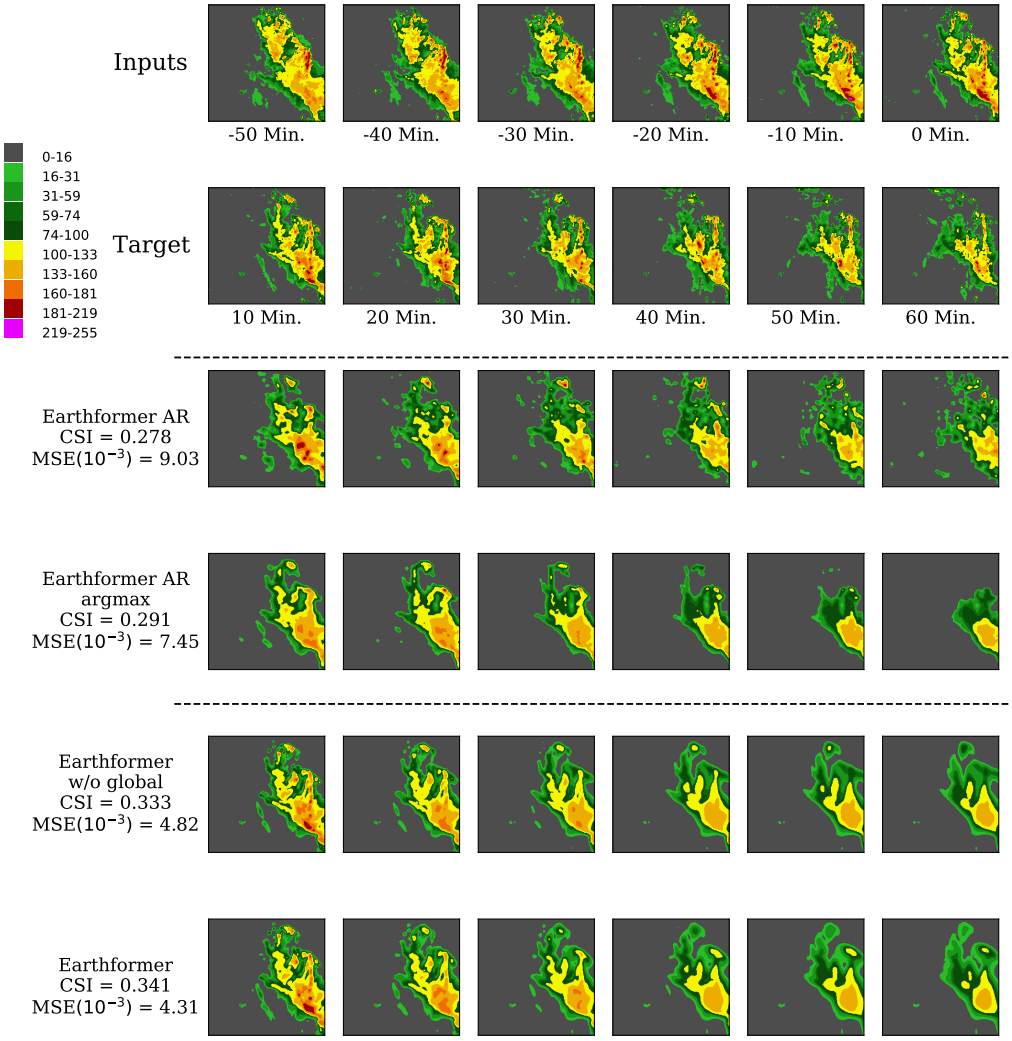


Figure 9: Non-Auto-Regressive v.s. Auto-Regressive on SEVIR: qualitative example 3.

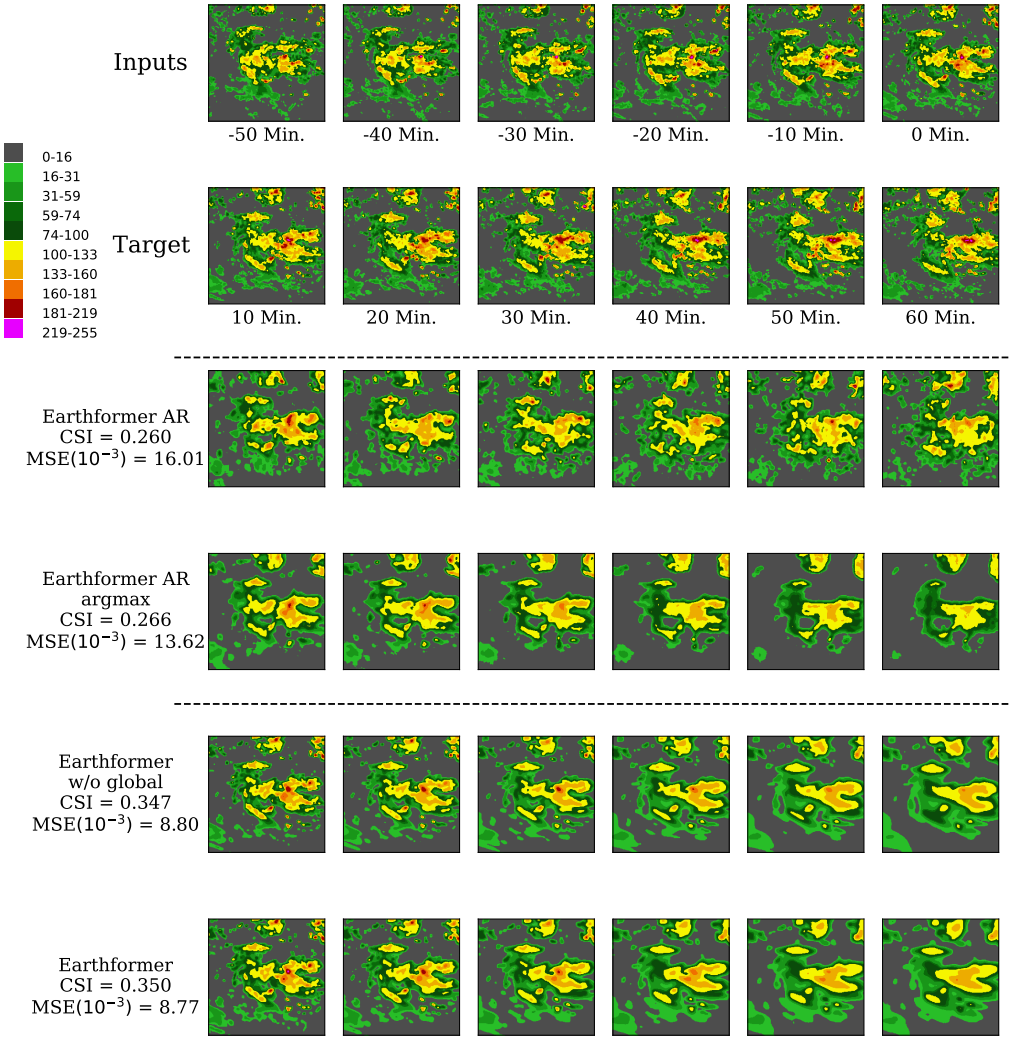


Figure 10: Non-Auto-Regressive v.s. Auto-Regressive on SEVIR: qualitative example 4.

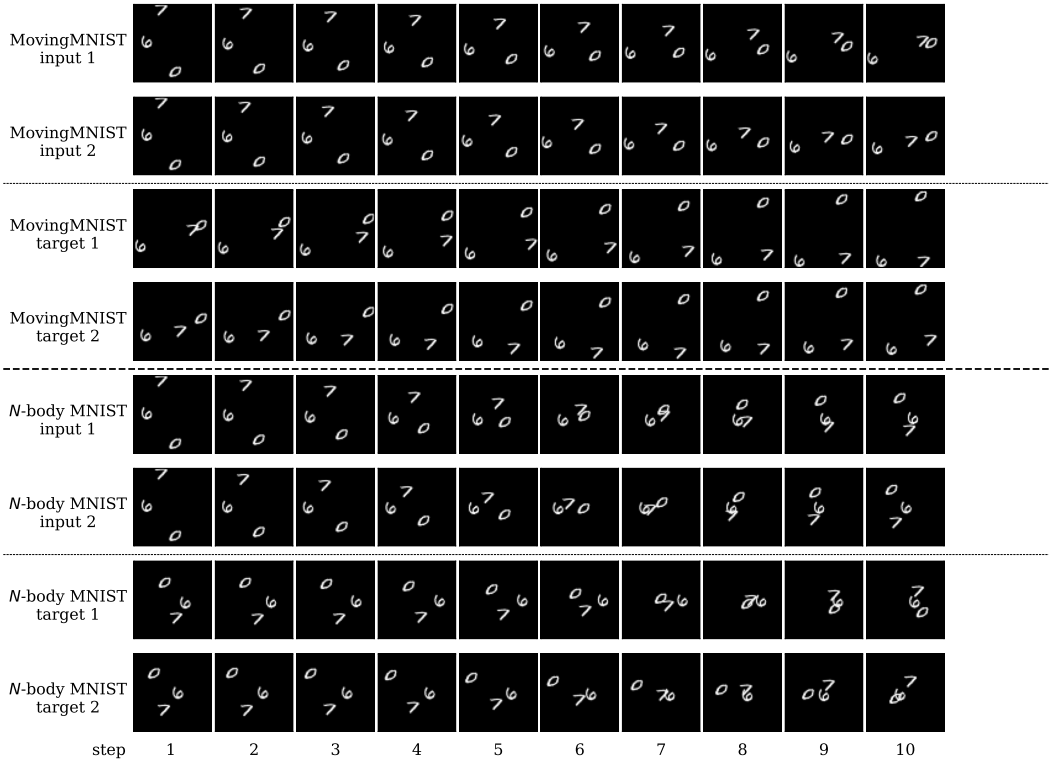


Figure 11: Chaos in N-body MNIST: the effect of a slight disturbance on the initial velocities is much more significant on N-body MNIST than on MovingMNIST. The top half are two MovingMNIST sequences, where their initial conditions only slightly differ in the the initial velocities. The bottom half are two N-body MNIST sequences. N-body MNIST sequence 1 has exactly the same initial condition as MovingMNIST sequence 1. N-body MNIST sequence 2 has exactly the same initial condition as MovingMNIST sequence 2. The final positions of digits in MovingMNIST after 20 steps evolution only slightly differ from each other, while the differences are much more significant in the final frames of N-body MNIST sequences.

D Chaos in N-Body MNIST Dataset

The Earth system is chaotic [39], meaning that the future is very sensitive to the initial conditions. In Fig. 11, we illustrate the chaotic effect of N-body MNIST. By only slightly changing the initial velocities of the digits, the positions of the digits after 20 steps change significantly. N-body MNIST is thus much more challenging than MovingMNIST.

E Evaluation Metrics in SEVIR

In Sec. 4.2 and Table 6, we follow [43] to use the `Critical Success Index` (`CSI`) for prediction quality evaluation[6]. Besides [43], the SEVIR team holds the *SEVIR Dataset Challenge*[7], where the data involved are exactly the same while the evaluation metrics are slightly different[8].

Specifically, [43] used six precipitation thresholds which correspond to pixel values $[219, 181, 160, 133, 74, 16]$. The prediction and the ground-truth are rescaled back to the range 0-255. The $\#\texttt{Hits}(\tau)$ (truth$\geq \tau$, pred$\geq \tau$), $\#\texttt{Misses}(\tau)$ (truth$\geq \tau$, pred$< \tau$) and $\#\texttt{F.Alarms}(\tau)$ (truth$< \tau$, pred$\geq \tau$) at certain threshold τ are counted over all test pixels as shown in Eqn. 12.

$$
\begin{aligned}
\#\texttt{Hits}(\tau) &= \sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{h=1}^{H}\sum_{w=1}^{W} \texttt{Hits}(\tau)[n,t,h,w] \\
\#\texttt{Misses}(\tau) &= \sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{h=1}^{H}\sum_{w=1}^{W} \texttt{Misses}(\tau)[n,t,h,w] \\
\#\texttt{F.Alarms}(\tau) &= \sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{h=1}^{H}\sum_{w=1}^{W} \texttt{F.Alarms}(\tau)[n,t,h,w] \\
\texttt{CSI-}\tau &= \frac{\#\texttt{Hits}(\tau)}{\#\texttt{Hits}(\tau) + \#\texttt{Misses}(\tau) + \#\texttt{F.Alarms}(\tau)} \\
\texttt{CSI-M} &= \frac{1}{6} \sum_{\tau \in [219,181,160,133,74,16]} \texttt{CSI-}\tau
\end{aligned}
\tag{12}
$$

$\tau \in [219, 181, 160, 133, 74, 16]$ is one of the thresholds. $\texttt{Hits}(\tau), \texttt{Misses}(\tau), \texttt{F.Alarms}(\tau) \in [0,1]^{N \times T \times H \times W}$, where N is the test dataset size, T is the forecasting horizon, H and W are the height and width, respectively. We denote the average `CSI-`τ over the thresholds $[219, 181, 160, 133, 74, 16]$ as `CSI-M`. The results using the above evaluation metrics are demonstrated in Sec. 4.2 and Table 6.

The calculation of `CSI` is slightly different in the SEVIR Dataset Challenge, as shown in Eqn. 13.

[6]The implementation details of `CSI` used in Sec. 4.2 and Table 6 follow `https://github.com/MIT-AI-Accelerator/neurips-2020-sevir`

[7]Challenge website at `https://sevir.mit.edu/nowcasting`

[8]The implementation details of `CSI` used in the SEVIR Dataset Challenge are available at `https://github.com/MIT-AI-Accelerator/sevir_challenges`.

Table 13: Performance comparison on SEVIR using metrics in SEVIR Dataset Challenge. The CSI is calculated at precipitation thresholds $[219, 181, 160, 133, 74, 16]$. Different from Table 6, $\texttt{CSI-}\tau = \frac{1}{T}\sum_t^T \texttt{CSI-}\tau(t)$ is the mean of $\texttt{CSI-}\tau(t)$ over forecasting horizon T. CSI-M3 and CSI-M6 are the average of CSI-τ over thresholds $[133, 74, 16]$ and $[219, 181, 160, 133, 74, 16]$, respectively.

Model	#Param. (M)	GFLOPS	Metrics									
			CSI-219 ↑	CSI-181 ↑	CSI-160 ↑	CSI-133 ↑	CSI-74 ↑	CSI-16 ↑	CSI-M3 ↑	CSI-M6 ↑	MSE (10^{-3}) ↓	MAE (10^{-2}) ↓
Persistence	-	-	0.0575	0.1056	0.1374	0.2259	0.4796	0.6109	0.4386	0.2695	11.5283	4.4349
UNet [43]	16.6	33	0.0521	0.1364	0.1887	0.3032	0.6542	0.7444	0.5673	0.3465	4.1119	2.7633
ConvLSTM [36]	14.0	527	0.1071	0.2275	0.2829	0.4155	0.6873	0.7555	0.6194	0.4126	3.7532	2.5898
PredRNN [45]	46.6	328	0.1164	0.2246	0.2718	0.3873	0.6744	0.7544	0.6054	0.4048	3.9014	2.6963
PhyDNet [13]	13.7	701	0.1041	0.2123	0.2583	0.3755	0.6627	0.7176	0.5853	0.3884	4.8165	3.1896
E3D-LSTM [44]	35.6	523	0.1125	0.2218	0.2637	0.3847	0.6674	0.7591	0.6037	0.4015	4.1702	**2.5023**
Rainformer [3]	184.0	170	0.0745	0.1580	0.2087	0.3397	0.6599	0.7308	0.5768	0.3619	4.0272	3.0711
Earthformer w/o global	13.1	257	0.1429	0.2643	0.3086	0.4215	0.6885	0.7671	0.6257	0.4321	3.7006	2.5306
Earthformer	15.1	257	**0.1480**	**0.2748**	**0.3126**	**0.4231**	**0.6886**	**0.7682**	**0.6266**	**0.4359**	**3.6702**	2.5112

$$
\begin{aligned}
\#\texttt{Hits}(\tau, t) &= \sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} \texttt{Hits}(\tau)[n, t, h, w] \\
\#\texttt{Misses}(\tau, t) &= \sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} \texttt{Misses}(\tau)[n, t, h, w] \\
\#\texttt{F.Alarms}(\tau, t) &= \sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} \texttt{F.Alarms}(\tau)[n, t, h, w] \\
\texttt{CSI-}\tau(t) &= \frac{\#\texttt{Hits}(\tau, t)}{\#\texttt{Hits}(\tau, t) + \#\texttt{Misses}(\tau, t) + \#\texttt{F.Alarms}(\tau, t)} \\
\texttt{CSI-}\tau &= \frac{1}{T}\sum_{t}^{T} \texttt{CSI-}\tau(t) \\
\texttt{CSI-M3} &= \frac{1}{3}\sum_{\tau\in[133,74,16]} \texttt{CSI-}\tau \\
\texttt{CSI-M6} &= \frac{1}{6}\sum_{\tau\in[219,181,160,133,74,16]} \texttt{CSI-}\tau
\end{aligned}
\tag{13}
$$

The CSI-τ, at certain threshold, is the mean of $\texttt{CSI-}\tau(t)$ over forecasting horizon T. We denote the average CSI-τ over the three thresholds $[133, 74, 16]$ used in the SEVIR Dataset Challenge as CSI-M3, and the average CSI-τ over the six thresholds $[219, 181, 160, 133, 74, 16]$ used in [43] as CSI-M6. The experiment results evaluated using the metrics in the SEVIR Dataset Challenge are listed in Table 13. Earthformer still consistently outperforms the baselines in almost all the metrics.