Generalizing Weather Forecast to Fine-grained Temporal Scales via Physics-AI Hybrid Modeling

Wanghan Xu[*]
Shanghai Jiao Tong University
Shanghai AI Laboratory
xu_wanghan@sjtu.edu.cn

Fenghua Ling
Shanghai AI Laboratory
lingfenghua@pjlab.org.cn

Wenlong Zhang
Shanghai AI Laboratory
zhangwenlong@pjlab.org.cn

Tao Han
Shanghai AI Laboratory
hantao.dispatch@pjlab.org.cn

Hao Chen
Shanghai AI Laboratory
chenhao1@pjlab.org.cn

Wanli Ouyang
Shanghai AI Laboratory
ouyangwanli@pjlab.org.cn

Lei Bai[†]
Shanghai AI Laboratory
bailei@pjlab.org.cn

Abstract

Data-driven artificial intelligence (AI) models have made significant advancements in weather forecasting, particularly in medium-range and nowcasting. However, most data-driven weather forecasting models are black-box systems that focus on learning data mapping rather than fine-grained physical evolution in the time dimension. Consequently, the limitations in the temporal scale of datasets prevent these models from forecasting at finer time scales. This paper proposes a physics-AI hybrid model (i.e., WeatherGFT) which **G**eneralizes weather forecasts to **F**iner-grained **T**emporal scales beyond training dataset. Specifically, we employ a carefully designed PDE kernel to simulate physical evolution on a small time scale (e.g., 300 seconds) and use a parallel neural networks with a learnable router for bias correction. Furthermore, we introduce a lead time-aware training framework to promote the generalization of the model at different lead times. The weight analysis of physics-AI modules indicates that physics conducts major evolution while AI performs corrections adaptively. Extensive experiments show that WeatherGFT trained on an hourly dataset, effectively generalizes forecasts across multiple time scales, ***including 30-minute, which is even smaller than the dataset's temporal resolution***. Code is available at **https://github.com/black-yt/WeatherGFT** .

1 Introduction

Weather forecasting plays a vital role in modern society, impacting a wide range of human activities. For example, minute-level precipitation nowcasting is particularly valuable for short-term planning, such as outdoor activities, while medium-range forecasts that offer daily predictions play a crucial role in long-term strategic decisions like maritime trade. This field has witnessed remarkable advancements in recent years, largely attributed to the rapid progress of machine learning-based (ML) weather forecasting models [38], spanning from nowcasting to medium-range forecasts.

[*]This work was done during his internship at Shanghai Artificial Intelligence Laboratory.
[†]Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0734

Prior studies tackle the problem of weather forecasting by leveraging data-driven models trained on benchmark weather forecasting datasets, such as WeatherBench [47] and ERA5 [22]. Prevalent medium-range forecasting models (e.g., FourCastNet [32], GraphCast [33], and FengWu [7]) are commonly trained on the aforementioned hourly datasets to generate global forecasts with a time interval of 6-hour, can not offer finer predictions like 30-minute forecasts for nowcasting.

A significant limitation of current ML-based weather forecasting models [32, 3, 33, 7, 18] lies in their black-box training paradigm [53, 17], that is, primarily focusing on learning the mapping of data pairs with a fixed lead time (e.g., 6 hours), without explicitly incorporating the laws of atmospheric dynamics which govern finer-grained physical evolution processes. Consequently, this training paradigm brings a significant challenge for weather forecasting: ***existing black-box AI models are unable to generalize at finer temporal scales beyond the inherent time resolution of the training datasets due to the absence of fine-grained physics modeling***.

To address this challenge, we propose WeatherGFT, a physics-AI hybrid model capable of simulating weather changes on fine-grained time scales through a set of partial differential equations (PDEs) [50]. WeatherGFT consists of an encoder, multiple stacked **HybridBlocks** and a decoder. As the core of our model, HybridBlock contains two branches: One utilizes PDE kernels to conduct physical evolution over small time scales, while the other employs neural networks to learn unresolved atmospheric processes and perform bias correction on the physical evolution. These two branches are adaptively fused through a **learnable router** initialized as 0.5:0.5. Unlike existing models [32, 33, 7] trained with a fixed lead time, we introduce a **lead time-aware framework** through multi-lead time training strategy and a lead time conditional decoder [43, 1], enabling the model to generalize to finer-grained temporal scales. Experiments demonstrate that our method is capable of forecasting at different lead times within one single model and one unified framework, overcoming the limitations of the dataset's temporal resolution and ***enabling 30-minute forecasts with an hourly dataset***.

Additionally, we find two interesting insights by examining the learnable route weight of the hybrid physical-AI modules at different lead times, as depicted in Figure 1: **a)** The physical weight is consistently higher than the AI, indicating the significant role played by the PDE kernel. **b)** As the lead time increases, the weight of AI gradually increases. We attribute this increment to the errors accumulation of PDE kernel during the evolution process, necessitating more AI corrections. In summary, when there is training data available at the lead time, such as at 1:00:00, the fitting ability of AI is enhanced. Conversely, at the lead time without training data, such as at 0:30:00, the importance of physical evolution becomes more pronounced, which confirms our motivation: **WeatherGFT can benefit from both physics and AI adaptively.**

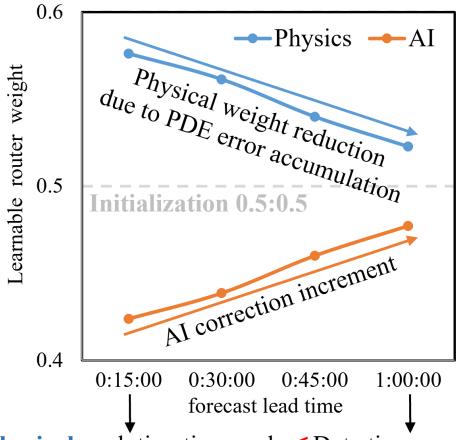

Figure 1: **Learnable router weight**. The role of physics and AI at different lead times: major evolution and adaptive correction (details in Sec. 4.4).

We summarize the contributions of this paper as follows:

- We propose a physics-AI hybrid model that incorporates physical PDEs into the networks, enabling the simulation of fine-grained physical evolution through its forward process.
- With the flexible PDE kernel and new lead time-aware training framework, our model performs multiple lead time forecasts, which bridges the nowcast and medium-range forecast.
- For the first time, our model extends the forecasting ability learned from an hourly dataset to make accurate predictions at a finer time scale, i.e., 30 minutes.
- Our model exhibits strong generalization ability while maintaining prediction errors comparable to those of pure AI and physical models.

https://doi.org/10.52202/079017-0734

2 Related Work

Data-driven Weather Forecasting. In recent years, data-driven weather forecasting models based on machine learning have developed rapidly [2], especially for medium-range weather forecast [54], which provides weather variables for the next few days. Clare et al.[11] propose a weather forecasting approach using stacked ResNets [21], but their model only considers geopotential and temperature, which is limited for real-world forecasting applications. FourCastNet [32] expands the model to include additional variables such as wind at different heights, and employs Adaptive Fourier Neural Operator (AFNO) [16] networks for prediction. Pangu-Weather [3] utilizes the 3D Swin Transformer [61] and introduces hierarchical temporal aggregation to minimize iterations in the autoregressive forecasting, followed by FengWu [7, 57], FuXi [9] and other Transformer-based [52] prediction models. Apart from Transformers, GraphCast [33] and Keisler [29] adopt a graph representation of the Earth and employ Graph Neural Network (GNN) [62] for weather prediction.

In addition to medium-range weather forecast, nowcast [4, 55] is another important field in weather forecast, which usually provides 30-minute forecasting of severe convective weather like thunderstorms. OFAF [44], Preciplstm [41], SimVP [12] use convolutions to capture spatial information and model temporal information through networks such as Long Short-Term Memory [60] or Recurrent Neural Network [59]. Earthformer [13] and CasCast [14] use Transformer-based models for nowcasting. The former proposes cuboid attention to efficiently model space-time information, and the latter uses the diffusion model [49] to address the problem of blur output. These nowcast models focus on minute-level forecasts for specific regions, and is difficult to forecast for long-term such as 5-day.

Consequently, there exists a significant gap (global vs. regional, day-level vs. minute-level, long-term vs. shot-term) between medium-range forecasts and nowcasts. Integrating AI models with physical guidance to make finer-grained predictions can bridge this gap.

Physical Neural Networks. Most data-driven models commonly neglect the incorporation of physics and treat networks as black-boxes. In order to enhance the consistency of predictions with respect to physical laws, PINNs [5], PINO [37], and DeepPhysiNet [35] add PDE loss to overall training loss. Nevertheless, these methods of changing loss functions often require balancing the weights between different PDEs, and the training results are heavily affected by hyperparameters. PI-HC-MoE [6], ClimODE [53] integrate physical processes into the networks, but they do not explicitly simulate the physical evolution of distinct variables based on PDEs. Instead, they implement the evolution using general kernels, such as Euler kernels [51]. NeuralGCM [31] employs neural networks to parameterize a dynamic core. However, it is primarily designed for medium-range forecasting. These works typically focus on forecasting at fixed lead times, rather than leveraging physical laws to generalize to finer-grained time scales beyond the training datasets.

3 Method

3.1 Problem Formulation

Weather forecasting aims to predict future weather states $\mathcal{X}_t$ given current weather states $\mathcal{X}_0$:

$$F_\theta(\mathcal{X}_0) = P(\mathcal{X}_t|\mathcal{X}_0) \tag{1}$$

where θ represents the parameters of the model and t denotes the lead time. The weather state $\mathcal{X} \in \mathbb{R}^{C\times H\times W}$ consists of C atmospheric variables across different pressure levels. Each variable is characterized by an $H \times W$ matrix that corresponds to the projection of the Earth's plane.

Assuming that the time resolution of the dataset is t_{data}, the lead time t for data-driven models can only be equal to or greater than t_{data}, because these models are trained using data pairs $(\mathcal{X}_0,\ \mathcal{X}_t)$ sampled from the dataset. Consequently, black-box AI models [32, 3, 33, 7, 20, 19, 15] are unable to forecast at finer lead times such as $\frac{1}{2}t_{data}$, indicating **a lack of temporal generalization ability**.

https://doi.org/10.52202/079017-0734

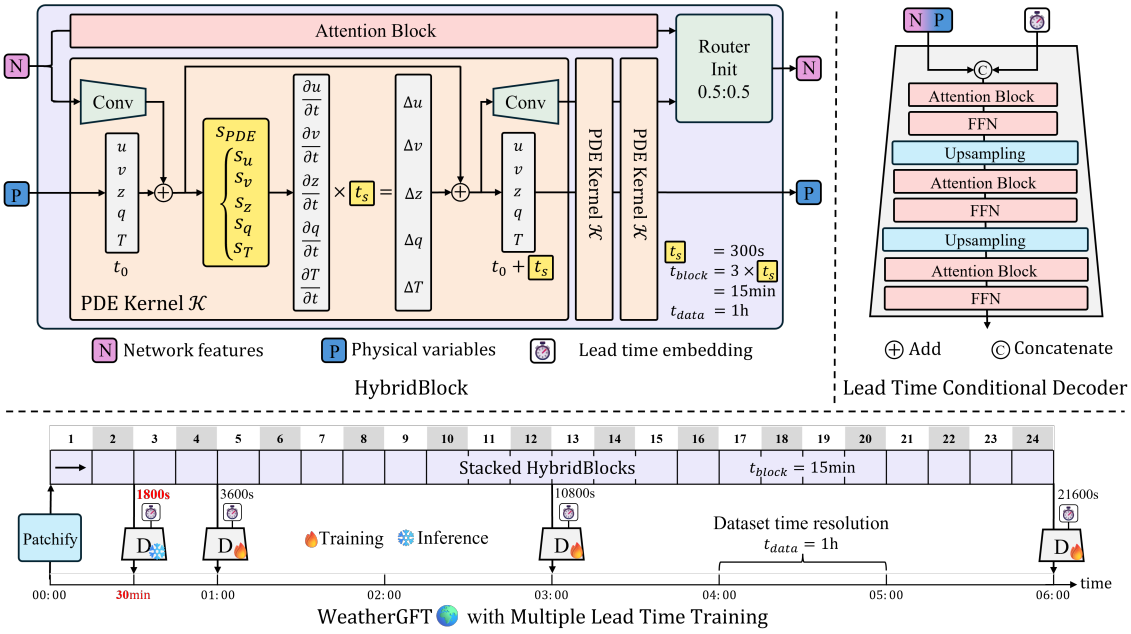

Figure 2: **Overview of WeatherGFT.** HybridBlock serves as the fundamental unit of the model, consisting of three PDE kernels, a parallel Attention Block, and a subsequent learnable router. A lead time conditional decoder is employed to generate forecasts for different lead times.

3.2 WeatherGFT Overview

As shown in Figure 2, our model consists of an encoder to patchify the weather states into tokens [52], multiple (specifically, 24) stacked HybridBlocks to preform weather evolution via PDE modeling, and a decoder to generate predictions under specific lead-time conditions.

Specifically, to enable our model to generalize at a finer-grained temporal resolution, we employ PDEs to model the evolution at a finer time scale:

$$\mathcal{X}_{t_s} = \mathcal{K}(\mathcal{X}_0), \text{ where } t_s = \frac{1}{m} t_{data},\ m \in \mathbb{Z}^+ \tag{2}$$

We simulate the physical evolution from $\mathcal{X}_0$ to $\mathcal{X}_{t_s}$ through a uniquely designed PDE kernel (details in Section 3.3), where t_s is much smaller than the time resolution t_{data} of the dataset, allowing model to capture fine-grained weather changes. By stacking PDE kernels $\mathcal{K}$, the longer evolution can be achieved like $\mathcal{X}_{t_{data}} = \mathcal{K}_m \ldots \mathcal{K}_2\ \mathcal{K}_1\ \mathcal{X}_0$. In this paper, we set m to 12, that is, $t_s = \frac{1}{12} t_{data}$.

To mitigate the issue of error accumulation as the number of evolutionary steps increases, we introduce a parallel Attention Block [52] that performs bias correction for every 3 iterations of $\mathcal{K}$. Additionally, a learnable router initialized as $0.5 : 0.5$, is employed to adaptively fuse features from PDE kernels and the Attention Block. We encapsulate three PDE kernels $\mathcal{K}$ and one parallel Attention Block within a HybridBlock, whose evolution time is $t_{block} = 3 \times t_s = \frac{1}{4} t_{data}$.

Our model can not only forecast at lead times equal to or greater than t_{data}, but also generalize to finer-grained time scale such as $\frac{1}{2} t_{data}$ even in the absence of corresponding training data pairs. This is achieved by modeling the physical evolution of $t_{block} = \frac{1}{4} t_{data}$, rather than simply learning from data pairs $(\mathcal{X}_0, \mathcal{X}_{t_{data}})$ sampled from the dataset. Notably, these generalized finer-grained predictions of our model outperform temporal interpolation on multiple metrics, as shown in Table 3, emphasizing the advantages of fine-grained physical evolution over black-box models.

3.3 PDE Kernel

We employ a set of five PDEs (7-11) including the motion equation, the continuous equation and others to establish a closed system, which simulate the physical evolution of 5 essential atmospheric variables: u (latitude-direction wind), v (longitude-direction wind), z (geopotential), q (humidity), T (temperature). The partial derivative of each atmospheric variable with respect to time can be separated mathematically (details in Appendix A), denotes as S_{PDE}, which takes current weather state as input and produces the derivative of each variable with respect to time. We define PDE kernel $\mathcal{K}$ as the evolution of the variables over a short period of time t_s, as demonstrated in Equation 3.

$$S_{PDE}\left(\mathcal{X}\right)=\begin{cases}\frac{\partial u}{\partial t}=S_u(u,v,z,q,T)16\\ \frac{\partial v}{\partial t}=S_v(u,v,z,q,T)16\\ \frac{\partial z}{\partial t}=S_z(u,v,z,q,T)21\\ \frac{\partial q}{\partial t}=S_q(u,v,z,q,T)23\\ \frac{\partial T}{\partial t}=S_T(u,v,z,q,T)18\end{cases},\quad \begin{array}{c}\text{PDE Kernel } \mathcal{K}\left(\mathcal{X}\right)=S_{PDE}(\mathcal{X})t_s+\mathcal{X}\\ \text{where } t_s=\frac{1}{12}t_{data}\end{array} \tag{3}$$

Calculating S_{PDE} requires the use of differential and integral operations. For example, for temperature T, its derivative with respect to time is shown in Equation 4. In order to efficiently calculate S_{PDE} and enable loss backward [25], we designed a fast implementation of differentiation and integration through convolution and matrix multiplication respectively. Equation 5 presents the implementation of the differential and integral of $\mathcal{X}$ in the x direction (latitude direction).

$$\frac{\partial T}{\partial t}=\frac{-L\frac{\partial z}{\partial p}w-\frac{\partial z}{\partial p}w}{c_p}-u\frac{\partial T}{\partial x}-v\frac{\partial T}{\partial y}-w\frac{\partial T}{\partial p},\ \text{where } w=-\int\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\mathrm{d}p \tag{4}$$

$$\begin{cases}\frac{\mathrm{d}\mathcal{X}}{\mathrm{d}x}=\frac{1}{12}Conv\left(\mathcal{X},K_x\right)\\ \int\mathcal{X}\mathrm{d}x=\mathcal{X}M_x\end{cases},K_x=\begin{bmatrix}0&0&0&0&0\\0&0&0&0&0\\1&-8&0&8&-1\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix},M_x=\begin{bmatrix}1&1&\cdots&1&1\\0&1&\cdots&1&1\\\vdots&\vdots&\ddots&\vdots&\vdots\\0&0&\cdots&1&1\\0&0&\cdots&0&1\end{bmatrix}\in\mathbb{R}^{W\times W} \tag{5}$$

Similarly, K_y and M_y can be constructed to perform differential and integral operations in the y direction (longitude direction). For differential and integral operations in the p direction (pressure level direction), we first reshape $\mathcal{X}\in\mathbb{R}^{C\times H\times W}$ to 3D space $\mathcal{X}_{3D}\in\mathbb{R}^{\frac{C}{P}\times P\times H\times W}$ based on the variables' pressure layers, and then implement corresponding operations through K_p and M_p.

3.4 HybridBlock with Adaptive Router

HybridBlock is a module that combines physics and AI. Firstly, it employs neural networks to address the issue of error accumulation resulting from the stacking of PDE kernel $\mathcal{K}$. Secondly, it utilizes the PDE kernel $\mathcal{K}$ to guide the neural networks to learn the physical evolution of a specific time step. The structure of HybridBlock consists of three PDE kernels $\mathcal{K}$ and one parallel Attention Block. Consequently, the time step corresponding to a HybridBlock is $t_{block}=3\times t_s=\frac{1}{4}t_{data}$.

HybridBlock has two branches, as depicted in Figure 2, one is physics and the other is AI. The neural networks features $\mathcal{X}_N$ are aligned with physical features $\mathcal{X}_P$ through a convolutional layer, followed by three PDE kernels. Subsequently, the PDE kernel output is projected back to the latent space of $\mathcal{X}_N$ through another convolutional layer. Finally, features fusion is performed through the learnable router shown in Figure 3.

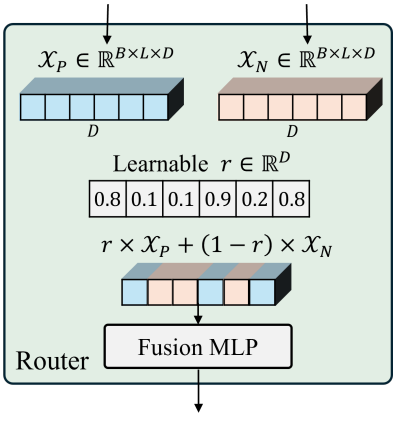

Figure 3: **Router in HybridBlock**. B represents batch size, L is the number of tokens with D dimension.

In the router, the features $\mathcal{X}_N$ obtained from the neural networks and the features $\mathcal{X}_P$ derived from the PDE kernels are initially linearly fused along the

https://doi.org/10.52202/079017-0734

feature dimension D, with the learnable factor r initialized as $0.5 : 0.5$. Subsequently, the preliminary fused features will go through an Multilayer Perceptron [40] layer containing a ReLU [36] activation function to accomplish nonlinear feature fusion.

3.5 Lead Time Conditional Decoder

HybridBlock provides the smallest time scale of model evolution, which is $t_{block} = \frac{1}{4}t_{data}$. Through $L\times$ HybridBlocks, we can predict the weather at a lead time of $\frac{L}{4}t_{data}$. To enable the model to generalize its prediction capabilities to finer-grained time scales, we design a lead time conditional decoder to generate forecasts varying lead times from the output of the corresponding HybridBlock.

In order to promote the expression of the condition, we embed the lead time t into a high-dimensional vector t_{emb} through learnable Fourier embedding [48], as shown in Equation 6.

$$t_{emb} = \sin(\pi \cdot t \cdot W) \oplus \cos(\pi \cdot t \cdot W) \oplus t, \text{ where } t \text{ is lead time} \tag{6}$$

where W is a learnable vector of size 16, and $\oplus$ denotes concatenation. Furthermore, t_{emb} will be concatenated with the output of HybridBlock and input to the decoder together. The decoder structure utilizes a Swin Transformer [39] with hierarchical upsampling, as illustrated in Figure 2.

3.6 Multiple Lead Time Training

For dataset like ERA5 [22] or WeatherBench [47], their time resolution is $t_{data} = 1\text{h}$. We set the time step of the PDE kernel to $t_s = \frac{1}{12}t_{data} = 300\text{s}$. Consequently, the time step of each HybridBlock is $t_{block} = 3 \times t_s = 900\text{s}$, equivalent to 15 minutes. By cascading 24 HybridBlocks, model can generate forecasts at a lead time of $24 \times 15\text{min} = 6\text{h}$. To encourage the model to learn evolution for different lead times and generalize forecasting to finer-grained time scales, during training, we not only use the output of the last HybridBlock but also include the outputs of the 4th and 12th HybridBlocks. These outputs are passed through the lead time conditional decoder with corresponding t_{emb} to predict the weather states at $4 \times 15\text{min} = 1\text{h}$ and $12 \times 15\text{min} = 3\text{h}$.

During inference, we can take the output of the second HybridBlock and pass it through the decoder with corresponding t_{emb} to get $2 \times 15\text{min} = 30\text{min}$ forecasts, which are not present in the dataset. In the Section 4.3, we provide a comprehensive demonstration showcasing the accuracy of these generalized prediction results for time scales smaller than the dataset's time resolution.

4 Experiment

Through the design of HybridBlock mixed with physics & AI and the multi-lead time training method, our model is capable of simultaneously conducting short-term forecasting and long-term forecasting without additional finetuning [42] on different forecasting tasks. In the experiments, we will showcase the superior performance of our model and try to answer the following questions:

(1) How does the model perform on the medium-range forecasting task?

(2) How does the model perform on the ***generalized 30-minute nowcasting*** task?

(3) As a hybrid expert model of AI and physics, what roles do they each play?

(4) How do PDE kernel and multi-lead time training contribute to the overall performance?

4.1 Experimental Setup

Dataset. We use WeatherBench [47] as our training dataset, whose time resolution is $t_{data} = 1\text{h}$ and spatial resolution is 128×256. The dataset spanning from 1980 to 2015 serves as training set, while the data of 2017 is the validation and test sets. Our model processes 4 surface variables and 5 upper-air variables across 13 pressure levels, as shown in Table 2.

Dataset	Train	Test	Time resolution
WeatherBench	✓	✓	1-hour
NASA	×	✓	30-minute

Table 1: **Datasets.** NASA dataset only contains precipitation, which will be used as the ground truth for precipitation nowcast.

Given that WeatherBench lacks data at finer temporal resolutions, we use the 30-minute satellite observations downloaded from NASA as ground truth to quantitatively assess the model's generalizability. NOTE: Data from NASA is only used for testing and not for model training.

Tasks. We conducted experiments on two typical weather forecasting tasks: medium-range forecasting and precipitation nowcasting. The forecast range for medium-range forecasting spans from 6 hours to 5 days, while the nowcasting is set to a range of 30 minutes to 2 hours.

Name	Description	Levels
u10	x-direction wind at 10m height	Single
v10	y-direction wind at 10m height	Single
t2m	Temperature at 2m height	Single
tp	Hourly precipitation	Single
z	Geopotential	13
q	Specific humidity	13
u	x-direction wind	13
v	y-direction wind	13
T	Temperature	13

Table 2: **Atmospheric Variables Considered.** The 13 levels are 50, 100, 150, 200, 250, 300, 400, 500, 600, 700, 850, 925, 1000 hPa.

Baseline Methods. We compare WeatherGFT with four forecast approaches: FourCastNet [32] uses AFNO [16] networks to simulate the nonlinear relationship between weather variables, Keisler [29] models global atmospheric data through GNN, ClimODE[53] adds ordinary differential equations (ODE) [26] to the neural networks, and ECMWF-IFS [46] is a physical dynamic model.

The above three data-driven models cannot generalize forecasting to finer-grained time scales due to the absence of 30-minute labels. Therefore, in nowcasting tasks, we interpolate the 30-minute forecast results through SOTA frame interpolation models Flavr [28] and UPR [27]. In contrast, our model can conduct 30-minute predictions inherently without interpolating.

Implementation Details. We implemented the model with PyTorch [25] and trained 50 epochs on 8 NVIDIA A100 GPUs [10] for 3 days, with a learning rate of cosine schedule starting from 5e-4.

4.2 Skillful Medium-Range Forecasts by WeatherGFT

Autoregression is commonly employed in medium-term forecasting, where the model output serves as the input for the subsequent forecast step, allowing for longer lead time predictions. However, prediction errors tend to accumulate during the autoregression, leading to an increase in the root mean square error (RMSE). As a result, a smaller RMSE indicates a more accurate prediction.

Figure 4 illustrates the changes in prediction RMSE of different weather variables as lead time increases. Our model demonstrates competitive performance across various lead times with AI or physical dynamics models, especially the prediction of surface temperature (t2m) and surface wind speed (u10) is significantly better than other models. The geopotential of the 500hpa pressure layer (z500) is a crucial weather variable in weather forecasting, as it reflects atmospheric circulation [45], subtropical high-pressure systems [34], and other significant phenomena. Due to the modeling of geopotential in the PDE 21, z500 prediction of our model outperforms the physical dynamic model ECMWF-IFS as visualized in Figure 5.

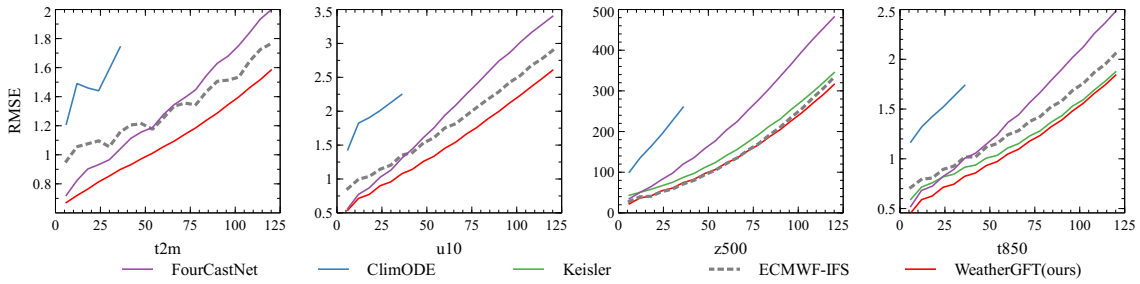

Figure 4: **Medium-Range Forecast.** The x-axis represents the lead time in hours, while the y-axis represents the RMSE for different variables. The smaller RMSE the better.

From the visualization in Figure 5, our model is more accurate in predicting the subtropical high, as indicated by the highlighted red box. In addition, the prediction error of our model at the lead time of 6-hour is significantly smaller than that of the physical dynamic model ECMWF-IFS.

https://doi.org/10.52202/079017-0734

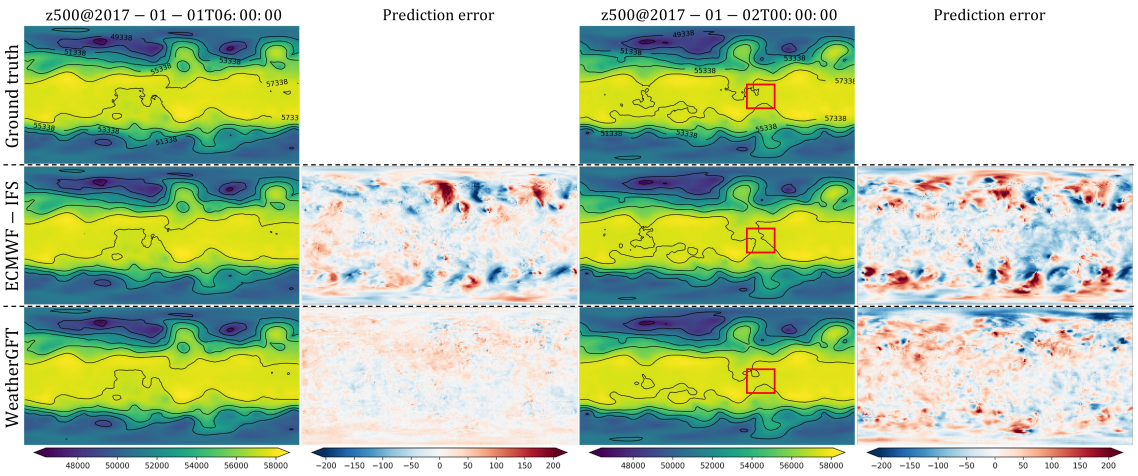

Figure 5: **Visualization of z500 Predictions.**

4.3 Generalizing to Fine-grained Time Scale for Nowcasting

In contrast to conventional black-box AI models [32, 29, 58] used in medium-range weather forecasting, WeatherGFT has the ability to break through the time scale limitations of the dataset, making the generalization to fine-grained temporal scales possible. This capability is facilitated by the dynamic progression of our PDE kernel modeling and multiple lead time training. Specifically, we use the second HybridBlock of the total 24 HybridBlocks to generate 30-minute generalized forecasts through the lead time conditional decoder, which is very important for precipitation nowcasting.

To quantify the accuracy of the model's generalized nowcasting, we utilize the NASA satellite precipitation observation dataset as the ground truth, which has a time resolution of 30-minute. We evaluate forecasts at 30, 60, 90, and 120 minutes. It is important to note that data of NASA were not used for training. For other comparison models that cannot directly produce half-hour forecasts, we use the frame interpolation models (i.e., Flavr [28] and UPR [27]) to generate 30-minute predictions.

	30-min			60-min			90-min			120-min		
	CSI↑ @0.5	CSI↑ @1.5	RMSE↓ tp1h	CSI↑ @0.5	CSI↑ @1.5	RMSE↓ tp1h	CSI↑ @0.5	CSI↑ @1.5	RMSE↓ tp1h	CSI↑ @0.5	CSI↑ @1.5	RMSE↓ tp1h
FourCast+Flavr	0.26	0.09	0.67	0.61	0.49	0.24	0.25	0.09	**0.65**	0.37	0.26	0.46
FourCast+UPR	0.20	0.10	0.76	0.61	0.49	0.24	0.11	0.05	1.49	0.37	0.26	0.46
Keisler+Flavr	0.25	0.09	**0.66**	0.59	0.48	0.23	0.25	0.08	0.66	0.41	0.29	0.35
Keisler+UPR	0.26	0.13	0.69	0.59	0.48	0.23	0.26	0.13	0.68	0.41	0.29	0.35
ClimODE+Flavr	0.26	0.09	0.67	0.62	**0.51**	0.22	0.25	0.09	0.66	0.47	0.34	0.32
ClimODE+UPR	0.25	0.12	0.67	0.62	0.49	0.21	0.25	0.11	0.66	0.46	0.32	0.31
WeatherGFT(ours)	**0.28**	**0.17**	0.72	**0.62**	0.50	**0.21**	**0.28**	**0.16**	0.71	**0.54**	**0.40**	**0.27**

Table 3: **Generalized Nowcast.** 60-min and 120-min are trained lead times, while 30-min and 90-min are generalized lead times. Gray represents the results obtained through the frame interpolation model, purple indicates the results obtained through our unified model without interpolating. For precipitation nowcasting, CSI (Critical Success Index) is the most important metric.

CSI@th (Critical Success Index) refers to the hit rate of the area that reaches the threshold precipitation value th. CSI@0.5 can reflect the overall forecast accuracy in rainy areas, and CSI@1.5 reflects the forecast accuracy in moderate rainy areas. Table 3 shows that our model surpasses others across different lead times, especially in forecasting regions of moderate rainfall, i.e., CSI@1.5.

The visualization in Figure 6 reveals that when using frame interpolation to obtain 30-minute predictions, there is blurring occurring at different scales, resulting in the loss of extreme values, as indicated in the red box. Our model, which incorporates physical constraints, provides clearer predictions retaining extreme values without the need for frame interpolation.

https://doi.org/10.52202/079017-0734

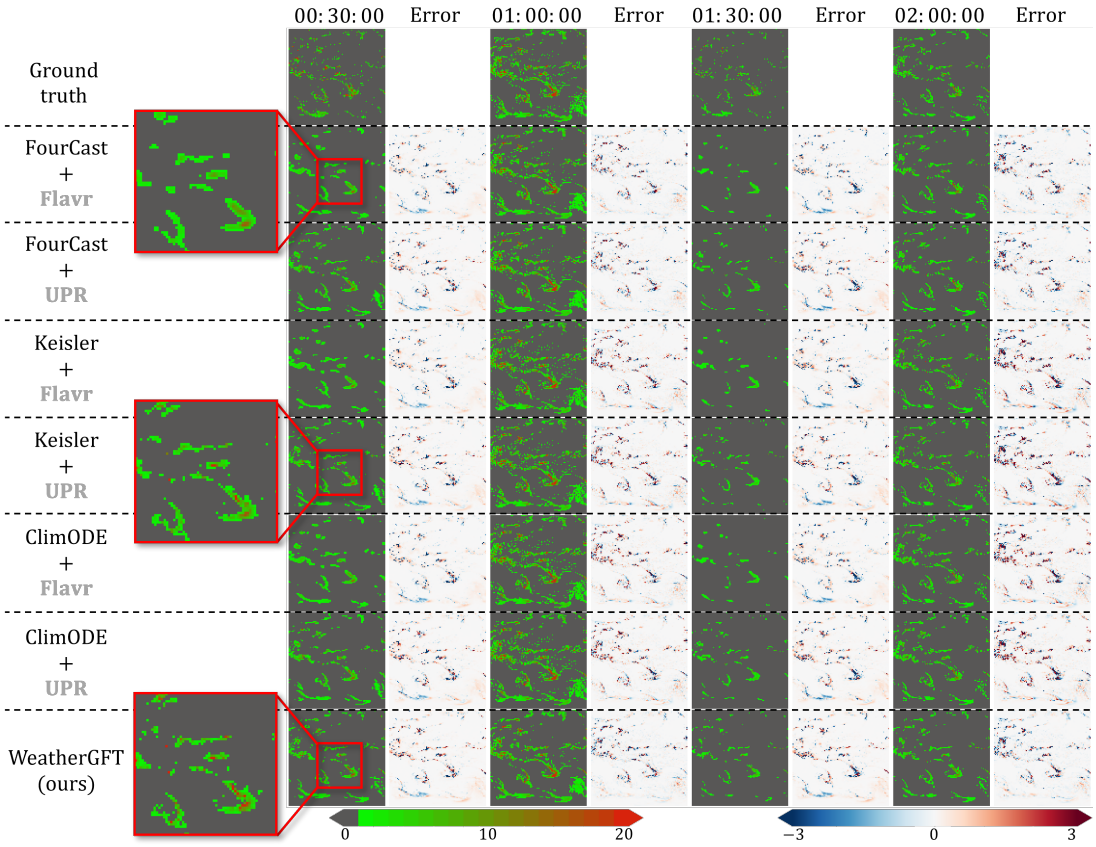

Figure 6: **Visualization of Precipitation Nowcast.** Precipitation in the area ranging from 34N to 50S and 148E to 128W during the time period from 00:00 to 02:00 on July 1, 2017.

4.4 Weather Forecasts can Benefit from Physics and AI via WeatherGFT

As a hybrid model combining both physics and AI components, it is crucial to analyze their contributions to the prediction process. We present insights into their respective proportions by visualizing the weight parameter r within the learnable router (refer to Figure 3). The visualization in Figure 7 reveals that the weights of the 24 HybridBlocks display a similar distribution:

a) The physical weight of the vast majority of HybridBlocks is significantly higher than the weight of AI, which shows that in the process of simulating time evolution, the PDE kernel plays a more important role, while the Attention Block only plays a supportive correction role. **b)** The physical weight gradually decreases while the weight of AI increases throughout each hour (dataset time resolution). This aligns with our underlying motivation, which acknowledges that errors may accumulate over time in the physics-based evolution. Consequently, a greater emphasis on AI corrections becomes necessary to compensate for these accumulated errors.

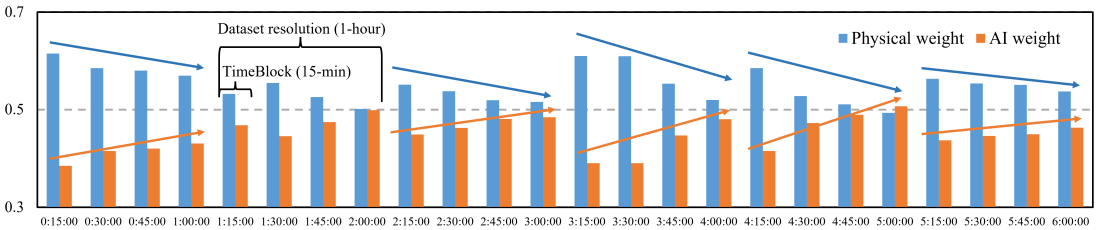

Figure 7: **The Weights in the Router of 24 HybridBlocks.**

https://doi.org/10.52202/079017-0734

By averaging the 4×6 HybridBlocks into 4 time steps, the average weight every 15-minute is obtained in Figure 1, which shows the above two conclusions more clearly. To summarize, physics plays the main evolutionary role in the model, while AI plays an dynamic corrective role.

4.5 Ablation Studies

We use Swin Attention Block [39] as the baseline for the ablation studies. For this baseline networks without PDE kernel constraints, as a black-box model, it will only learn the mapping of data pairs corresponding to the lead time. Consequently, its internal information between blocks is unexplainable, which also results in being unable to predict moments without data labels, such as 30-minute nowcasting.

	30-min nowcast	RMSE@1-h t2m↓ z500↓	RMSE@6-h t2m↓ z500↓	RMSE@3-d t2m↓ z500↓
Attent Block	×	0.52 18.76	0.73 24.21	1.23 157.9
+ PDE Kernel	✓	0.57 20.43	0.70 **21.78**	1.22 153.8
+ Muti Time	✓	**0.49** **16.66**	**0.67** 21.80	**1.14** **152.4**

Table 4: **Ablation Experiment.**

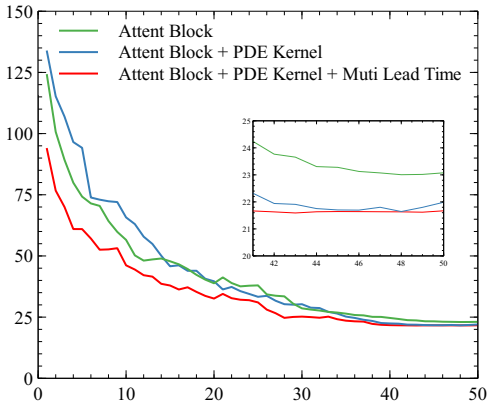

Figure 8: **RMSE z500 as Training Epochs.**

PDE kernel is crucial to the generalization of finer-grained predictions. Instead of simply learning the mapping between data, the model learns the evolution of the corresponding time step according to the physics laws, making information of each neural network layer explainable, thereby facilitating generalized 30-minute nowcasting. In addition, we find that the introduction of the PDE kernel also improved the prediction accuracy of the model.

Multiple lead time training accelerates convergence and improves the accuracy of model prediction, as shown in Figure 8. We hypothesize that this phenomenon can be attributed to the loss backward from different lead times, which alleviates the issue of vanishing gradients [23], allowing the parameters of different layers to quickly warm up and improve the expression of the model.

5 Conclusion

Most existing data-driven weather forecast methods which operated as black-box models via purely performing data mapping are unable to generalize at finer temporal scale beyond the inherent time resolution of the training datasets due to the absence of the fine-grained physics modeling. This paper proposes a physics-AI hybrid model to solve this problem. Through the exquisitely designed PDE kernel, each block in the networks can simulate the evolution of physical variables at finer-gained time step, while AI plays the role of adaptive correction, which makes our model capable of generalizing predictions to a finer time scale beyond dataset. By employing our proposed multi-lead time training strategy, our model trained on an hourly dataset exhibits remarkable ability of generalized 30-minute forecasts, while maintaining prediction errors that are competitive with those of pure AI and physical models in both medium-range forecast and precipitation nowcast.

The main limitation of our model is that only five important atmospheric equations are currently considered, which is still far from fully modeling the atmospheric motion process. Another limitation of this paper is that the experiments have been conducted solely at a spatial resolution of 128×256. As part of our future work, we plan to extend our experiments to higher resolutions such as 721×1440 to assess the model's performance under different settings. Additionally, while the minimum evolution time scale of our model is 15 minutes, we were unable to evaluate 15-minute generalized predictions due to the absence of corresponding validation data at that specific time scale. Therefore, we are currently only able to perform evaluations of 30-minute generalized predictions.

For future work, we plan to incorporate additional physical laws into our model and conduct higher-resolution experiments to ascertain the upper limit of its capabilities.

Acknowledgements

This work is supported by Shanghai Artificial Intelligence Laboratory.

References

[1] Marcin Andrychowicz, Lasse Espeholt, Di Li, Samier Merchant, Alexander Merose, Fred Zyda, Shreya Agrawal, and Nal Kalchbrenner. Deep learning for day forecasts from sparse observations. *arXiv preprint arXiv:2306.06079*, 2023.

[2] Zied Ben Bouallègue, Mariana CA Clare, Linus Magnusson, Estibaliz Gascon, Michael Maier-Gerber, Martin Janoušek, Mark Rodwell, Florian Pinault, Jesper S Dramsch, Simon TK Lang, et al. The rise of data-driven weather forecasting: A first statistical assessment of machine learning-based weather forecasts in an operational-like context. *Bulletin of the American Meteorological Society*, 2024.

[3] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.

[4] KA Browning and CiG Collier. Nowcasting of precipitation systems. *Reviews of Geophysics*, 27(3):345–370, 1989.

[5] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[6] Nithin Chalapathi, Yiheng Du, and Aditi Krishnapriyan. Scaling physics-informed hard constraints with mixture-of-experts. *arXiv preprint arXiv:2402.13412*, 2024.

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

[8] Kun Chen, Lei Bai, Fenghua Ling, Peng Ye, Tao Chen, Kang Chen, Tao Han, and Wanli Ouyang. Towards an end-to-end artificial intelligence driven global weather forecasting system. *arXiv preprint arXiv:2312.12462*, 2023.

[9] Lei Chen, Xiaohui Zhong, Feng Zhang, Yuan Cheng, Yinghui Xu, Yuan Qi, and Hao Li. Fuxi: a cascade machine learning forecasting system for 15-day global weather forecast. *npj Climate and Atmospheric Science*, 6(1):190, 2023.

[10] Jack Choquette and Wish Gandhi. Nvidia a100 gpu: Performance & innovation for gpu computing. In *2020 IEEE Hot Chips 32 Symposium (HCS)*, pages 1–43. IEEE Computer Society, 2020.

[11] Mariana CA Clare, Omar Jamil, and Cyril J Morcrette. Combining distribution-based neural networks to predict weather forecast probabilities. *Quarterly Journal of the Royal Meteorological Society*, 147(741):4337–4357, 2021.

[12] Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z Li. Simvp: Simpler yet better video prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3170–3180, 2022.

[13] Zhihan Gao, Xingjian Shi, Hao Wang, Yi Zhu, Yuyang Bernie Wang, Mu Li, and Dit-Yan Yeung. Earthformer: Exploring space-time transformers for earth system forecasting. *Advances in Neural Information Processing Systems*, 35:25390–25403, 2022.

[14] Junchao Gong, Lei Bai, Peng Ye, Wanghan Xu, Na Liu, Jianhua Dai, Xiaokang Yang, and Wanli Ouyang. Cascast: Skillful high-resolution precipitation nowcasting via cascaded modelling. *arXiv preprint arXiv:2402.04290*, 2024.

 https://doi.org/10.52202/079017-0734

[15] Junchao Gong, Siwei Tu, Weidong Yang, Ben Fei, Kun Chen, Wenlong Zhang, Xiaokang Yang, Wanli Ouyang, and Lei Bai. Postcast: Generalizable postprocessing for precipitation nowcasting via unsupervised blurriness modeling. *arXiv preprint arXiv:2410.05805*, 2024.

[16] 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.

[17] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[18] Tao Han, Song Guo, Zhenghao Chen, Wanghan Xu, and Lei Bai. Weather-5k: A large-scale global station weather dataset towards comprehensive time-series forecasting benchmark. *arXiv preprint arXiv:2406.14399*, 2024.

[19] Tao Han, Song Guo, Fenghua Ling, Kang Chen, Junchao Gong, Jingjia Luo, Junxia Gu, Kan Dai, Wanli Ouyang, and Lei Bai. Fengwu-ghr: Learning the kilometer-scale medium-range global weather forecasting. *arXiv preprint arXiv:2402.00059*, 2024.

[20] Tao Han, Song Guo, Wanghan Xu, Lei Bai, et al. Cra5: Extreme compression of era5 for portable global climate and weather research via an efficient variational transformer. *arXiv preprint arXiv:2405.03376*, 2024.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

[23] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[24] Zhaoyang Huo, Yubao Liu, Yueqin Shi, Baojun Chen, Hang Fan, and Yang Li. An investigation on joint data assimilation of a radar network and ground-based profiling platforms for forecasting convective storms. *Monthly Weather Review*, 151(8):2049–2064, 2023.

[25] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, pages 87–104, 2021.

[26] Edward L Ince. *Ordinary differential equations*. Courier Corporation, 1956.

[27] Xin Jin, Longhai Wu, Jie Chen, Youxin Chen, Jayoon Koo, and Cheul-hee Hahm. A unified pyramid recurrent network for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1578–1587, 2023.

[28] Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. Flavr: Flow-agnostic video representations for fast frame interpolation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2071–2082, 2023.

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

[30] Ryuji Kimura. Numerical weather prediction. *Journal of Wind Engineering and Industrial Aerodynamics*, 90(12-15):1403–1414, 2002.

[31] Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, James Lottes, Stephan Rasp, Peter Düben, Milan Klöwer, et al. Neural general circulation models. *arXiv preprint arXiv:2311.07222*, 2023.

[32] Thorsten Kurth, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, and Anima Anandkumar. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the platform for advanced scientific computing conference*, pages 1–11, 2023.

[33] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[34] Hassan Lashkari and Zeinab Mohammadi. Study on the role of annual movements of arabian subtropical high pressure in the late start of precipitation in southern and southwestern iran. *Theoretical and applied climatology*, 137:2069–2076, 2019.

[35] Wenyuan Li, Zili Liu, Keyan Chen, Hao Chen, Shunlin Liang, Zhengxia Zou, and Zhenwei Shi. Deepphysinet: Bridging deep learning and atmospheric physics for accurate and continuous weather modeling. *arXiv preprint arXiv:2401.04125*, 2024.

[36] Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. *Advances in neural information processing systems*, 30, 2017.

[37] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 2021.

[38] Fenghua Ling, Lin Ouyang, Boufeniza Redouane Larbi, Jing-Jia Luo, Tao Han, Xiaohui Zhong, and Lei Bai. Is artificial intelligence providing the second revolution for weather forecasting? *arXiv preprint arXiv:2401.16669*, 2024.

[39] 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.

[40] Ian D Longstaff and John F Cross. A pattern recognition approach to understanding the multi-layer perception. *Pattern Recognition Letters*, 5(5):315–319, 1987.

[41] Zhifeng Ma, Hao Zhang, and Jie Liu. Preciplstm: A meteorological spatiotemporal lstm for precipitation nowcasting. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–8, 2022.

[42] Julieta Martinez, Jashan Shewakramani, Ting Wei Liu, Ioan Andrei Bârsan, Wenyuan Zeng, and Raquel Urtasun. Permute, quantize, and fine-tune: Efficient compression of neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15699–15708, 2021.

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

[44] Tengfei Nie, Xiang Ji, and YuYing Pang. Ofaf-convlstm: an optical flow attention fusion-convlstm model for precipitation nowcasting. In *2021 3rd International Academic Exchange Conference on Science and Technology Innovation (IAECST)*, pages 283–286. IEEE, 2021.

[45] Abraham H Oort and Eugene M Rasmusson. *Atmospheric circulation statistics*, volume 5. US Department of Commerce, National Oceanic and Atmospheric Administration âĂę, 1971.

[46] Anders Persson and Federico Grazzini. User guide to ecmwf forecast products. *Meteorological Bulletin*, 3(2), 2007.

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

[48] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

 https://doi.org/10.52202/079017-0734

[49] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[50] Walter A Strauss. *Partial differential equations: An introduction*. John Wiley & Sons, 2007.

[51] Shashank Reddy Vadyala, Sai Nethra Betgeri, and Naga Parameshwari Betgeri. Physics-informed neural network method for solving one-dimensional advection equation using pytorch. *Array*, 13:100110, 2022.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[53] Yogesh Verma, Markus Heinonen, and Vikas Garg. Climode: Climate and weather forecasting with physics-informed neural odes. *arXiv preprint arXiv:2404.10024*, 2024.

[54] A James Wagner. Medium-and long-range forecasting. *Weather and Forecasting*, 4(3):413–426, 1989.

[55] ZiDong Wang, Zeyu Lu, Di Huang, Tong He, Xihui Liu, Wanli Ouyang, and Lei Bai. Predbench: Benchmarking spatio-temporal prediction across diverse disciplines. *arXiv preprint arXiv:2407.08418*, 2024.

[56] Yi Xiao, Lei Bai, Wei Xue, Hao Chen, Kun Chen, Tao Han, Wanli Ouyang, et al. Towards a self-contained data-driven global weather forecasting framework. In *Forty-first International Conference on Machine Learning*, 2023.

[57] Yi Xiao, Lei Bai, Wei Xue, Kang Chen, Tao Han, and Wanli Ouyang. Fengwu-4dvar: Coupling the data-driven weather forecasting model with 4d variational assimilation. *arXiv preprint arXiv:2312.12455*, 2023.

[58] Wanghan Xu, Kang Chen, Tao Han, Hao Chen, Wanli Ouyang, and Lei Bai. Extremecast: Boosting extreme value prediction for global weather forecast. *arXiv preprint arXiv:2402.01295*, 2024.

[59] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[60] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[61] Yonghong Zhang, Xuquan Ji, Wenyong Liu, Zhuofu Li, Jian Zhang, Shanshan Liu, Woquan Zhong, Lei Hu, Weishi Li, et al. A spine segmentation method under an arbitrary field of view based on 3d swin transformer. *International Journal of Intelligent Systems*, 2023, 2023.

[62] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

A PDE Solver

We constrain 5 atmospheric variables, that is, u (latitude-direction wind), v (longitude-direction wind), z or ϕ (geopotential), q (humidity), T (temperature), through the following set of five partial differential equations (PDEs) [30]:

$$\frac{\mathrm{d}\mathbf{V}}{\mathrm{d}t} + f\mathbf{k} \times \mathbf{V} = -g\nabla_p z + \mathbf{F}_h \tag{7}$$

$$\frac{\partial \phi}{\partial p} = -\frac{1}{\rho} \tag{8}$$

$$\nabla_p \cdot \mathbf{V} + \frac{\partial w}{\partial p} = 0 \tag{9}$$

$$c_p \frac{\mathrm{d}T}{\mathrm{d}t} - \frac{1}{\rho} w = Q \tag{10}$$

$$p = \rho R T \tag{11}$$

The expansion of $\frac{\mathrm{d}}{\mathrm{d}t}$ is as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t} = \left(\frac{\partial}{\partial t}\right)_p + \mathbf{V} \cdot \nabla_p (\,) + w\frac{\partial}{\partial p} \tag{12}$$

The PDE above is in the pressure coordinate system, which is aligned with the input to our model, as the input to the model comes from 13 pressure layers. In the air pressure coordinate system, the following equation is also satisfied:

$$\frac{\partial p}{\partial t} = 0 \tag{13}$$

w represents the vertical wind speed and is not directly included as one of the input variables in our model. However, it can be derived from u and v using following equation:

$$\begin{aligned} \frac{\partial w}{\partial p} &= -\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} \\ w &= -\int \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \mathrm{d}p \end{aligned} \tag{14}$$

After getting w, we can get $\frac{\partial u}{\partial t}$ and $\frac{\partial v}{\partial t}$ according to Equation 7.

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} + w\dfrac{\partial u}{\partial p} - fv = -\dfrac{\partial \phi}{\partial x} \\ \dfrac{\partial v}{\partial t} + u\dfrac{\partial v}{\partial x} + v\dfrac{\partial v}{\partial y} + w\dfrac{\partial v}{\partial p} + fu = -\dfrac{\partial \phi}{\partial y} \end{cases} \tag{15}$$

$$\begin{cases} \dfrac{\partial u}{\partial t} = -u\dfrac{\partial u}{\partial x} - v\dfrac{\partial u}{\partial y} - w\dfrac{\partial u}{\partial p} + fv - \dfrac{\partial \phi}{\partial x} \\ \dfrac{\partial v}{\partial t} = -u\dfrac{\partial v}{\partial x} - v\dfrac{\partial v}{\partial y} - w\dfrac{\partial v}{\partial p} - fu - \dfrac{\partial \phi}{\partial y} \end{cases} \tag{16}$$

where $f = 7.29e-5$ is a constant.

According to Equation 10, we can get $\frac{\partial T}{\partial t}$:

https://doi.org/10.52202/079017-0734

$$\begin{cases} c_p\left(\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}+w\frac{\partial T}{\partial p}\right)-\frac{1}{\rho}w=Q \\ Q=-L\frac{\partial \phi}{\partial p}w \end{cases} \tag{17}$$

$$\frac{\partial T}{\partial t}=\frac{-L\frac{\partial \phi}{\partial p}w-\frac{\partial \phi}{\partial p}w}{c_p}-u\frac{\partial T}{\partial x}-v\frac{\partial T}{\partial y}-w\frac{\partial T}{\partial p} \tag{18}$$

where $L=2.5e6$ and $c_p=1005$ are constants.

According to Equations 8 and Equations 11, we can get $\frac{\partial \phi}{\partial t}$:

$$\frac{\partial \phi}{\partial p}=-\frac{1}{\rho}=-\frac{RT}{p} \tag{19}$$

$$\begin{aligned} \frac{\partial^2 \phi}{\partial p \partial t}&=-\frac{\partial \frac{RT}{p}}{\partial t} \\ &=-R\left(\frac{1}{p}\frac{\partial T}{\partial t}-\frac{T}{p^2}\frac{\partial p}{\partial t}\right) \\ &=-\frac{R}{p}\frac{\partial T}{\partial t} \end{aligned} \tag{20}$$

$$\begin{aligned} \frac{\partial \phi}{\partial t}&=\int \frac{\partial^2 \phi}{\partial p \partial t}\mathrm{d}p \\ &=-\int \frac{R}{p}\frac{\partial T}{\partial t}\mathrm{d}p \end{aligned} \tag{21}$$

where $R=8.314$ is a constant.

Finally, according to the water vapor equation 22, we can get $\frac{\partial q}{\partial t}$:

$$\begin{cases} \frac{\mathrm{d}q}{\mathrm{d}t}=\frac{\delta F}{RT}\frac{\mathrm{d}\phi}{\mathrm{d}t} \\ \delta=\begin{cases}0, \frac{\mathrm{d}\phi}{\mathrm{d}t}<0\ and\ q\geq q_s \\ 1, else\end{cases} \\ F=q_sT\frac{LR-c_pR_vT}{c_pR_vT^2+L^2q_s} \\ e_s=6.112\times exp\left(\frac{17.67T'}{T'+243.5}\right) \\ T'=T-273.15 \\ q_s=\frac{0.622e_s}{p-0.378e_s} \end{cases} \tag{22}$$

$$\frac{\partial q}{\partial t}=\frac{\delta F}{RT}\left(\frac{\partial \phi}{\partial t}+u\frac{\partial \phi}{\partial x}+v\frac{\partial \phi}{\partial y}+w\frac{\partial \phi}{\partial z}\right)-u\frac{\partial q}{\partial x}-v\frac{\partial q}{\partial y}-w\frac{\partial q}{\partial z} \tag{23}$$

where $R_v=461.5$ and $R_d=287$ are constants.

B Implementation of Integrals and Differentials

Integral in p-direction (pressure levels direction) is implemented with PyTorch [25] as follows:

```
def integral_z(input_tensor):
    # Pressure-direction integral
    B, pressure_level_num, H, W = input_tensor.shape
    input_tensor = input_tensor.reshape(B, pressure_level_num, H*W)
    output = M_z.to(input_tensor.dtype).to(input_tensor.device) @
                                        input_tensor
    output = output.reshape(B, pressure_level_num, H, W)
    return output
```

M_x obtains the integral through matrix multiplication. Given the input matrix x below, the result of xM_x is:

$$x = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, \; xM_x = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1+4 \\ 2 & 2+5 \\ 3 & 3+6 \end{bmatrix}. \tag{24}$$

Differentials in x-direction (latitude direction) is implemented with PyTorch as follows:

```
def d_x(input_tensor):
    # Latitude-direction differential
    B, C, H, W = input_tensor.shape
    conv_kernel = torch.zeros([1,1,1,5], device=input_tensor.device,
                                        dtype=input_tensor.dtype,
                                        requires_grad=False)
    conv_kernel[0,0,0,0] = 1
    conv_kernel[0,0,0,1] = -8
    conv_kernel[0,0,0,3] = 8
    conv_kernel[0,0,0,4] = -1

    input_tensor = torch.cat((input_tensor[:,:,:,-2:],
                              input_tensor,
                              input_tensor[:,:,:,:2]), dim=3)
    _, _, H_, W_ = input_tensor.shape

    input_tensor = input_tensor.reshape(B*C, 1, H_, W_)
    output_x = F.conv2d(input_tensor, conv_kernel)/12
    output_x = output_x.reshape(B, C, H, W)
    output_x = output_x/pixel_x.to(output_x.dtype).to(output_x.device)

    return output_x
```

K_x is the convolution kernel. Assume a one-dimensional input data $x = [-2, -1, 0, 1, 2]$. It gradually increases from left to right by 1, that is, its gradient is 1. Applying convolution kernel K_x to x, the result is: $Conv(x, K_x) = \frac{(-2)\times1+(-1)\times(-8)+0\times0+1\times8+2\times(-1)}{12} = 1$. By using this convolution kernel, the data gradient can be determined.

Differentials in y-direction (longitude direction) is implemented with PyTorch as follows:

```
def d_y(input_tensor):
    # Longitude-direction differential
    B, C, H, W = input_tensor.shape
    conv_kernel = torch.zeros([1,1,5,1], device=input_tensor.device,
                                        dtype=input_tensor.dtype,
                                        requires_grad=False)
    conv_kernel[0,0,0] = -1
    conv_kernel[0,0,1] = 8
    conv_kernel[0,0,3] = -8
    conv_kernel[0,0,4] = 1
```


https://doi.org/10.52202/079017-0734

```
    input_tensor = torch.cat((input_tensor[:,:,:2],
                              input_tensor,
                              input_tensor[:,:,-2:]), dim=2)
    _, _, H_, W_ = input_tensor.shape

    input_tensor = input_tensor.reshape(B*C, 1, H_, W_)
    output_y = F.conv2d(input_tensor, conv_kernel)/12
    output_y = output_y.reshape(B, C, H, W)
    output_y = output_y/pixel_y

    return output_y
```

Differentials in p-direction (pressure levels direction) is implemented with PyTorch as follows:

```
def d_z(input_tensor):
    # Pressure-direction differential
    conv_kernel = torch.zeros([1,1,5,1,1], device=input_tensor.device,
                                           dtype=input_tensor.dtype,
                                          requires_grad=False)
    conv_kernel[0,0,0] = -1
    conv_kernel[0,0,1] = 8
    conv_kernel[0,0,3] = -8
    conv_kernel[0,0,4] = 1

    input_tensor = torch.cat((input_tensor[:,:2],
                              input_tensor,
                              input_tensor[:,-2:]), dim=1)

    input_tensor = input_tensor.unsqueeze(1) # B, 1, C, H, W
    output_z = F.conv3d(input_tensor, conv_kernel)/12
    output_z = output_z.squeeze(1)
    output_z = output_z/pixel_z.to(output_z.dtype).to(output_z.device)

    return output_z
```

C Hyperparameter Details

Hyperparameter	Value
Max epoch	50
Batch size	4x8 (GPUs)
Learning rate	5e-4
Learning rate schedule	Cosine
Patch size	4x4
Embedding dimension	1024
MLP ratio	4
Activation function	GLUE
Input (0-hour)	[4, 69, 128, 256]
Output (1, 3, 6-hour)	[4, 3, 69, 128, 256]

Table 5: **Hyperparameters of the Model**

Datasets	Training set	Validation set	Test set	Time resolution	Variable
WeatherBench	1980-2014	2015	2017-2018	1h	tp, t2m, u10, v10, z, q, u, v, t
NASA	None	None	2017-2018	30min	tp

Table 6: **Datasets Information**

D Additional Experiments

D.1 Prediction Bias Evaluation

Bias [2, 8, 56] indicates the disparity between the model's predictions and the ground truth. Negative bias indicates underestimation, a prevalent issue in forecasting models. Although the PDE kernel was not specifically designed to address bias underestimation, experimental results indicate that its usage helps ameliorate underestimation.

$$\text{bias} = \overline{pred - gt} \tag{25}$$

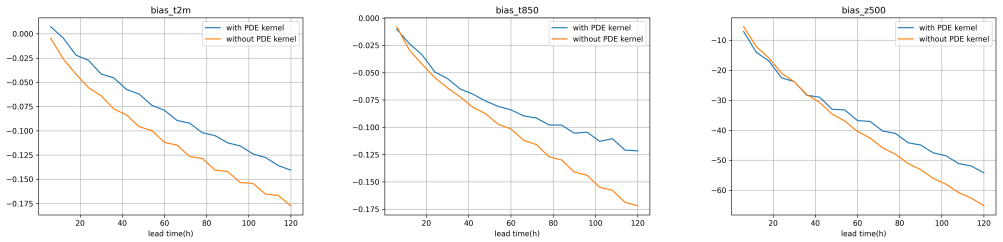

Figure 9: **Bias.** The closer to 0 the better.

D.2 Prediction Energy Evaluation

This assesses the energy [24] changes in the model's predictions. The experiments reveal that employing the PDE kernel aids in energy preservation.

$$\text{energy} = \frac{1}{2}\left(u^2 + v^2\right) + \frac{c_p}{2T_r}T^2 \tag{26}$$

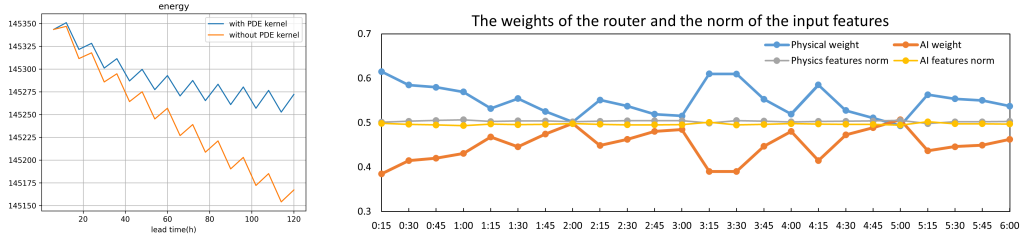

Figure 10: **Left.** Energy: the more consistent the better. **Right.** The norms of the outputs from the two networks are similar and stable. This indicates: a) The two networks produce outputs on the same scale. b) The router is decoupled and dynamically selects the more crucial features from the two branches without affecting the scale of the two networks.

D.3 Router Weights and Features Norm Change

Figure 10 complements Figure 7 in the paper. It illustrates that physical and AI features are on a comparable scale, with the router dynamically selecting the more effective aspects from each. The router's weight adjustments do not impact the output of the AI or physical branches, highlighting the router's decoupling characteristics.

E Code Of Ethics and Broader Impacts

Our research is ethical. The physical and AI hybrid model proposed in this paper can be used for global weather forecasting, which can serve many fields such as transportation and agriculture, and bring huge benefits to society.

https://doi.org/10.52202/079017-0734

The dataset used in this paper is public and there are no issues of infringement or privacy leakage. The experiments conducted in this paper are fair and reproducible. The resource consumption during the experiments is minimal and will not have an impact on the environment and society.

The model we propose is free of bias and discrimination issues. We open-source the model code and checkpoints on GitHub.

F Safeguard of Model

This paper presents a hybrid physics-AI model for global weather forecasting. It is important to acknowledge that all models inherently carry a certain degree of forecasting error. Hence, the model proposed in this paper should not be solely relied upon as the sole basis for predicting significant events. Instead, it is recommended to integrate the findings from this model with other models and expert insights to draw comprehensive and informed conclusions.

G Assets

Our study adheres to the licenses governing the usage of existing assets, as the data utilized in this paper are publicly available and permitted for academic research purposes.

The model introduced in this paper represents a novel contribution and is considered a new asset.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We demonstrate the contribution and scope of the paper in the Abstract and Introduction 1.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We show the limitations of our paper in Section 5.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

https://doi.org/10.52202/079017-0734

Justification: We present the assumptions and proofs in Section 3.3 and Appendix A.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We show the configuration required to reproduce the experiment in 4.1. We show part of the model code in Appendix B, and the complete model code is released on GitHub.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: This paper uses the public datasets, there is no requirement to release the data. We show part of the model code in Appendix B, and the complete model code is released on GitHub.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so âĂIJNoâĂİ is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: We present the experimental setup and details in Section 4.1.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: We show a detailed and correct error assessment in Section 4.2 and Section 4.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

https://doi.org/10.52202/079017-0734

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We present the computational resources of our experiments in Section 4.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: We explain in Appendix E that our research is ethical.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: We show Broader Impacts in Appendix E.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [Yes]

    Justification: We explain the safeguards of the model in Appendix F.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We explain the usage of the existing assets in Appendix G.

    Guidelines:

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

https://doi.org/10.52202/079017-0734

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [Yes]

    Justification: We explain the new assets in Appendix G.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: This study does not include crowdsourcing experiments and research with human subjects.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: This research is not related to Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects.

    Guidelines:

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

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-0734