

FuXi: A cascade machine learning forecasting system for 15-day global weather forecast

Lei Chen^{1†}, Xiaohui Zhong^{1†}, Feng Zhang^{2,3}, Yuan Cheng¹, Yinghui Xu¹, Yuan Qi^{1*} and Hao Li^{1*}

¹Artificial Intelligence Innovation and Incubation Institute,
Fudan University, Shanghai, 200433, China.

²Key Laboratory of Polar Atmosphere-ocean-ice System for
Weather and Climate, Ministry of Education, Department of
Atmospheric and Oceanic Sciences, Fudan University, Shanghai,
200433, China.

³Shanghai Qi Zhi Institute, Shanghai, 200232, China.

*Corresponding author(s). E-mail(s): qiyuan@fudan.edu.cn;
lihao_lh@fudan.edu.cn;

Contributing authors: cltpys@163.com; x7zhong@gmail.com;
fengzhang@fudan.edu.cn; cheng_yuan@fudan.edu.cn;
renji.xyh@vip.163.com;

†These authors contributed equally to this work.

Abstract

Over the past few years, due to the rapid development of machine learning (ML) models for weather forecasting, state-of-the-art ML models have shown superior performance compared to the European Centre for Medium-Range Weather Forecasts (ECMWF)'s high-resolution forecast (HRES) in 10-day forecasts at a spatial resolution of 0.25° . However, the challenge remains to perform comparably to the ECMWF ensemble mean (EM) in 15-day forecasts. Previous studies have demonstrated the importance of mitigating the accumulation of forecast errors for effective long-term forecasts. Despite numerous efforts to reduce accumulation errors, including autoregressive multi-time step loss, using a single model is found to be insufficient to achieve optimal performance in both short and long lead times. Therefore, we present FuXi, a cascaded ML weather forecasting system that provides 15-day global

forecasts with a temporal resolution of 6 hours and a spatial resolution of 0.25° . FuXi is developed using 39 years of the ECMWF ERA5 reanalysis dataset. The performance evaluation, based on latitude-weighted root mean square error (RMSE) and anomaly correlation coefficient (ACC), demonstrates that FuXi has comparable forecast performance to ECMWF EM in 15-day forecasts, making FuXi the first ML-based weather forecasting system to accomplish this achievement.

Keywords: weather forecast, machine learning, accumulation error, cascade, FuXi, transformer

1 Introduction

Accurate weather forecasts play an important role in many aspects of human society. Currently, national weather centers around the world generate weather forecasts using numerical weather prediction (NWP) models, which simulate the future state of the atmosphere. Running NWP models often requires high-performance computing systems, with some simulations taking several hours using thousands of nodes. The Integrated Forecast Systems (IFS) of the European Centre for Medium-range Weather Forecast (ECMWF) is widely regarded as the most accurate global weather forecast model [1]. The ECMWF's high-resolution forecast (HRES) runs at a horizontal resolution of 0.1° with 137 vertical levels for 10-day forecasts. However, uncertainty in weather forecasts is inevitable due to the limited resolution, approximation of physical processes in parameterizations, errors in initial conditions (and boundary conditions for regional models), and the chaotic nature of the atmosphere. Additionally, the degree of uncertainty and the magnitude of errors in weather forecasts increases with increasing forecast lead time. One way to address this uncertainty is to run an ensemble of forecasts by incorporating perturbations in initial conditions and physical parameterizations in the NWP model. The ECMWF ensemble prediction system (EPS) [2], which can provide forecasts for up to 15 days, includes one control member and 50 perturbed members and runs at 18 km spatial resolution with 91 vertical levels. Despite having lower spatial resolution than the deterministic forecast, the ensemble forecast is still computationally expensive. Furthermore, as an integral part of weather forecasting systems, postprocessing techniques, such as machine learning (ML) models, are commonly applied to calibrate the model outputs and improve forecast accuracy before operational applications [3].

In recent years, there have been increasing efforts to replace the traditional NWP models with ML models for weather forecasting [4]. ML-based weather forecasting systems have several advantages over NWP models, including faster speeds and the potential to provide higher accuracy than uncalibrated NWP models due to training with reanalysis data [4]. To facilitate inter-comparison between different ML models, the WeatherBench benchmark was

introduced to evaluate medium-range weather forecasting (i.e., 3-5 days) [5, 6]. WeatherBench was created by regridding ERA5 reanalysis data [7] from 0.25° resolution to three different resolutions (5.625° , 2.8125° and 1.40625°). Several studies have aimed to improve forecast performance on this dataset [8–10]. Rasp et al. [8] used a deep residual convolutional neural network (CNN) known as ResNet [11] to predict 500 hPa geopotential ($Z500$), 850 hPa temperature ($T850$), 2-meter temperature ($T2M$), and total precipitation (TP) at a spatial resolution of 5.625° for up to 5 days. They found that the ResNet model has similar performance compared to physical baseline models, such as IFS T42 and T63, with a comparable resolution. Meanwhile, Hu et al. [10] proposed the SwinVRNN model, which utilizes a Swin Transformer-based recurrent neural network (RNN) (SwinRNN) model coupled with a perturbation module to learn multivariate Gaussian distributions based on the Variational Auto-Encoder framework. They demonstrated the SwinVRNN model’s potential as a powerful ML-based ensemble weather forecasting system, with good ensemble spread and better accuracy compared to IFS in terms of $T2M$ and 6-hourly TP in 5-day forecasts with a 5.625° resolution.

While ML models have shown good performance in weather forecasting, their values are limited because of their forecasts’ low resolution (e.g., 5.625°). As a remarkable breakthrough, the FourCastNet model [12] is the first of its kind to provide high-resolution global weather forecasts of 0.25° for a time period of 7 days. It integrates the Adaptive Fourier neural operator (AFNO) [13] with a Vision Transformer (ViT) [14]. However, FourCastNet’s forecast accuracy is still worse than HRES’s. SwinRDM [15] distinguishes itself as the first ML-based weather forecasting system to outperform ECMWF HRES in 5-day forecasts at a spatial resolution of 0.25° . SwinRDM integrates SwinRNN+, an improved version of SwinRNN that surpasses ECMWF HRES at a spatial resolution of 1.40625° , with a diffusion-based super-resolution model that increases the resolution to 0.25° . Pangu-Weather [16] shows its superior performance compared to ECMWF HRES in 7-days forecasts at a resolution of 0.25° . Additionally, GraphCast [17], an autoregressive model that implements a graph neural network (GNN), outperforms HRES in 90% of the 2760 variable and lead time combinations in 10-day forecasts.

Although ML models have shown promising results in generating weather forecasts for up to 10 days, long-term forecasting remains challenging due to unforeseeable cumulative errors. The iterative forecasting method, which uses the model outputs as inputs for subsequent predictions, is a commonly used approach in developing ML-based weather forecasting systems. This approach is analogous to the time-stepping methods used in conventional NWP models [18]. However, as the number of iterations increases, errors in the model outputs accumulate, which may lead to significant discrepancies with the training data and unrealistic values in long-term forecasts. To improve the stability and accuracy of long-term forecasts, Weyn et al. [9] proposed a multi-time-step loss function to minimize errors over multiple iterated time steps. Rasp et al. [5] compared iterative forecasts with direct forecasts that predict specific lead

times and found the latter to be more accurate. However, one limitation of direct forecasts is that distinct models must be trained for each lead time. The FourCastNet [12] model underwent two training phases: pre-training, where the model is optimized to map one time step to the next, which is 6 hours apart, and fine-tuning to minimize errors in two-step prediction, similar to the multi-time-step loss function proposed Weyn et al. On the other hand, Bi et al. proposed a hierarchical temporal aggregation strategy for PangWeather’s forecasts, training four separate models for 1-hour, 3-hour, 6-hour, and 24-hour forecasts [16]. They demonstrated that running the 24-hour model 7 times is better than running the 1-hour model 168 times as it significantly reduces the accumulation errors for 7-day forecasts. However, they acknowledged that training a model directly predicting the lead time beyond 24 hours is challenging with their current model. Meanwhile, Lam et al. employed a curriculum training schedule following pre-training to improve GraphCast’s ability to make accurate forecasts for multiple steps [17]. Their study revealed that GraphCast’s performance decreases in short lead times and improves at longer lead times as the number of autoregressive steps increases. Thus, using a single model is insufficient for achieving the best performance for both short and long lead times.

As of now, ML-based weather forecasting systems have surpassed ECMWF HRES in 10-day forecasts; the next challenging objective is to perform comparably to ECMWF ensemble mean (EM) and extend the forecast lead time beyond 10 days. This work aims to reduce the accumulation error and provide 15-day ML-based weather forecasts that have comparable performance to ECMWF EM. Since a single model is shown to be incapable of achieving optimal forecast performance across various forecast lead times, we propose a novel cascade ML model architecture for weather forecasting based on pre-trained models, each optimized for specific forecast time windows. As a result, we present FuXi¹ weather forecasting system that generates 15-day forecasts at the spatial resolution of 0.25° . Figure 1 demonstrates that FuXi significantly outperforms ECMWF HRES and, for the first time, performs comparably to ECMWF EM, as evaluated using the globally-averaged latitude-weighted anomaly correction coefficient (ACC) of $Z500$ and $T2M$. FuXi is a cascade of models optimized for three sequential forecast time periods of 0-5 days, 5-10 days, and 10-15 days, respectively. The base FuXi model is an autoregressive model designed to efficiently extract complex features and learn relationships from a large volume of high-dimensional weather data. A total of 39 years of 6-hourly ECMWF ERA5 reanalysis data at a spatial resolution of 0.25° are used for developing the FuXi system.

Overall, our contribution to this work can be summarized as follows:

- We develop FuXi, the first ML-based weather forecasting system to have comparable performance to ECMWF EM.

¹FuXi, (Chinese: 伏羲), the first of ancient China’s mythological emperors, is said to be the first weather forecaster of China. He created bagua (八卦), eight diagrams, which were used to explain the constitution of the universe and predict weather.

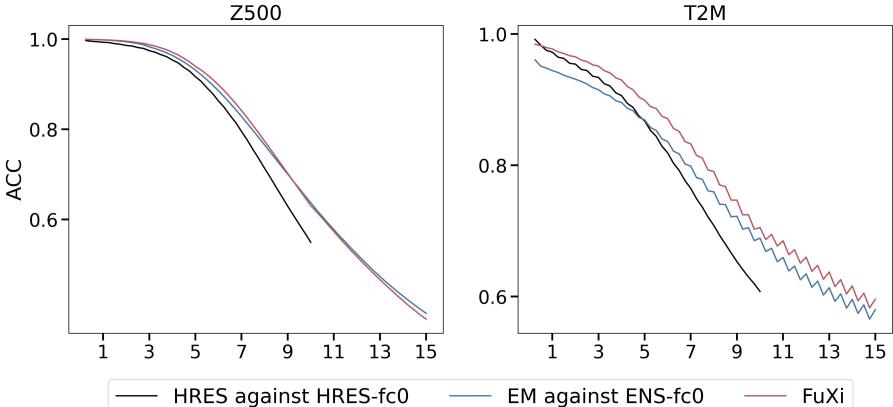


Fig. 1: Comparison of the globally-averaged latitude-weighted ACC of FuXi (red lines), ECMWF HRES (black lines), and ECMWF EM (blue lines) for $Z500$ (first column) and $T2M$ (second column) using testing data from 2018. FuXi is evaluated using the ERA5 reanalysis dataset as ground truth. ECMWF HRES and ECMWF EM are evaluated against ECMWF HRES (HRES-fc0) and ensemble control forecast (ENS-fc0) at the initial time step, respectively.

- We propose a novel cascade ML model architecture for weather forecasting, which aims to reduce accumulation errors and, for the first time, push the ML-based weather forecasts for up to 15 days.

2 Dataset

2.1 ERA5

ERA5 is the fifth generation of the ECMWF reanalysis dataset, providing hourly data of surface and upper-air parameters at a horizontal resolution of approximately 31 km and 137 model levels from January 1940 to the present day [7]. The dataset is generated by assimilating high-quality and abundant global observations using ECMWF's IFS model. Given its coverage and accuracy, the ERA5 data is widely regarded as the most comprehensive and accurate reanalysis archive. Therefore, we use the ERA5 reanalysis dataset as the ground truth for the model training.

We use a subset of the ERA5 dataset spanning 39 years, which has a spatial resolution of 0.25° (721×1440 latitude-longitude grid points) and a temporal resolution of 6 hours. In this work, we focus on predicting 5 upper-air atmospheric variables at 13 pressure levels (50, 100, 150, 200hPa, 250, 300, 400, 500, 600, 700, 850, 925, and 1000 hPa), and 5 surface variables. The 5 upper-air atmospheric variables are geopotential (Z), temperature (T), u component of wind (U), v component of wind (V), and relative humidity (R). Additionally, 5 surface variables are $T2M$, 10-meter u wind component ($U10$), 10-meter v

Table 1: A summary of variable definitions used in this paper.

Variables	Definitions
C, H, W	Channel dimensions, and spatial dimensions in latitude and longitude directions, respectively.
D	A set containing all the forecast initialization times in the testing dataset.
c, i, j, t_0, τ	Indices for variables, latitude coordinates, and longitude coordinates, as well as forecast initialization time and forecast lead time steps added to t_0 , respectively.
X^t, \hat{X}^t, M	Ground truth and model predicted weather parameters at time step t , and climatological mean computed using ERA reanalysis data between 1993 and 2016.
Z, R, T, U, V, TP	They represent geopotential, relative humidity, temperature, u component of wind, v component of wind, and 6-hourly total precipitation, respectively.

wind component ($V10$), mean sea-level pressure (MSL), and TP^2 . In total, 70 variables are predicted and evaluated.

Following previous studies in splitting the data into training, validation, and testing set [12, 17], the training set consists of 54020³ samples spanning from 1979 to 2015. The validation set contains 2920 samples corresponding to the years 2016 and 2017, while out-of-sample testing is performed using 1460 samples from 2018.

2.2 HRES-fc0 and ENS-fc0 dataset

In this study, we evaluate our model against the ERA5 reanalysis data. Besides, we also created two reference datasets, HRES-fc0 and ENS-fc0, which consist of the first time step of each HRES and ensemble control forecast, respectively. We use these datasets to assess the performance of ECMWF HRES and EM. This approach aligns with that used by Haiden et al. [1] and Lam et al. [17] in evaluating ECMWF forecasts.

3 Methodology

3.1 Generating 15-days forecasts using FuXi

The FuXi model is an autoregressive model that leverages weather parameters (X^{t-1}, X^t) from two previous time steps as input to forecast weather parameters at the upcoming time step (X^{t+1}). t , $t - 1$, and $t + 1$ represent the current, the prior, and upcoming time steps, respectively. The time step considered in this model is 6 hours. By utilizing the model's outputs as inputs, the system can generate forecasts with different lead times.

Generating 15-day forecasts using a single FuXi model requires 60 iterative runs. Pure data-driven ML models, unlike physics-based NWP models, lack

²A summary of variable definitions can be referred to in Table 1

³54020 = 365 × 4 × 37, similarly, 2920 = 365 × 4 × 2, and 1460 = 365 × 4

physical constraints, which can result in significantly growing errors and unrealistic predictions for long-term forecasts. Using an autoregressive, multi-step loss effectively minimizes accumulation error for long lead times [17]. This loss is similar to the cost function applied in the four-dimensional variational data assimilation (4D-Var) method, which aims to identify the initial weather conditions that optimally fit observations distributed over an assimilation time window. Although increasing the autoregressive steps leads to more accurate forecasts for longer lead times, it also results in less accurate results for shorter lead times. Besides, larger autoregressive steps require more memory and computing resources for handling gradients during the training process, similar to increasing the assimilation time window of 4D-Var.

When making iterative forecasts, error accumulation is unavoidable as lead times increase. Also, previous studies indicate that a single model can not perform optimally across all lead times. In order to have optimal performance for both short and long lead times, we propose a cascade [19, 20] model architecture using pre-trained FuXi models, fine-tuned for optimal performance in specific 5-day forecast time windows, namely FuXi-Short (0-5 days), FuXi Medium (5-10 days), and FuXi-Long (10-15 days). As shown in Figure 2, FuXi-Short and FuXi Medium outputs from the 20th and 40th steps are used as inputs to FuXi-Medium and FuXi-Long, respectively. The cascaded FuXi model performs comparably to ECMWF EM in 15-day forecasts, as shown in Figure 1.

3.2 FuXi model architecture

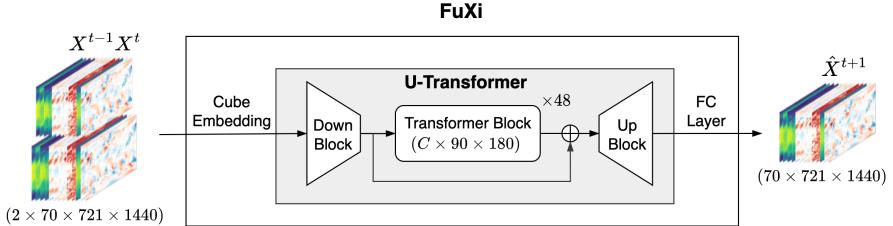
The model architecture of the base FuXi model consists of three main components, which are illustrated in Figure 2: cube embedding, U-Transformer, and a fully connected (FC) layer. The input data combines both upper-air and surface variables and creates a data cube with dimensions of $2 \times 70 \times 721 \times 1440$, where 2, 70, 721, and 1440 represent the total number of variables (C), the two preceding time steps ($t - 1$ and t), latitude (H) and longitude (W) grid points, respectively.

Firstly, the high-dimensional input data undergoes dimension reduction to $C \times 180 \times 360$ through joint space-time cube embedding. The primary purpose of cube embedding is to reduce the temporal and spatial dimensionality of input data, making it less redundant. Subsequently, the U-Transformer processes the embedded data, and prediction follows using a simple FC layer. The output is initially reshaped to $70 \times 720 \times 1440$, then restored to the original input shape of $70 \times 720 \times 1440$ by bilinear interpolation. The following subsections provide details for each component in the base FuXi model.

3.2.1 Cube embedding

To reduce the spatial and temporal dimensions of input and accelerate the training process, the space-time cube-embedding [21] is applied. A similar approach, patch embedding, which divides an image into $N \times N$ patches

a) The overall architecture of FuXi model



b) Cascade model architecture

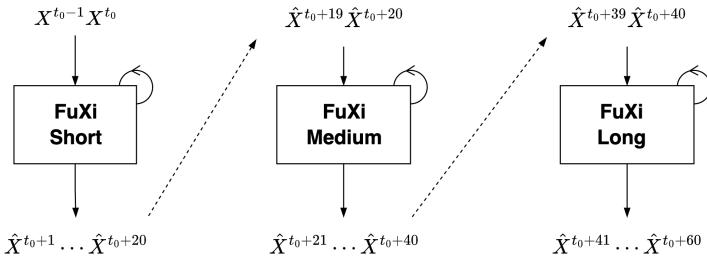


Fig. 2: Overall architecture of FuXi model. a) The FuXi model consists of three components: cube embedding, U-Transformer, and fully connected (FC) layer; b) FuXi-Short, FuXi-Medium, and FuXi-Long models cascade and produce 15-day forecasts, with each model generating 5 days forecasts.

with each patch being transformed into a feature vector, was used in the Pangu-Weather model [16]. The cube embedding applies a 3-dimensional (3D) convolution layer, with a kernel and stride of $2 \times 4 \times 4$ (equivalent to $\frac{T}{2} \times \frac{H}{4} \times \frac{W}{4}$), and output channels numbering C . Following cube embedding, a layer normalization (LayerNorm) [22] is utilized to improve training stability. The result is a data cube with dimensions $C \times 180 \times 360$.

3.2.2 U-Transformer

This subsection presents the design of the U-Transformer, which is visually represented in Figure 2 through its schematic diagram.

Recently, the ViT [14] and its variants have demonstrated remarkable performance in various computer vision tasks by using the multi-head self-attention, which enables the simultaneous processing of sequential input data. Nevertheless, global self-attention is infeasible for processing high-resolution inputs due to its quadratic computational and memory complexity with respect to the input size. Swin Transformer was proposed as a solution [23] to improve computational efficiency by limiting computation of self-attention only within the non-overlapping local windows. Besides, the shifted-window mechanism allows for cross-connections between windows. As a result, the Swin Transformer has shown superior performance on various benchmarks and is

frequently used as a backbone architecture in many vision tasks. Additionally, many researchers have developed ML-based weather forecasting models using Swin Transformer blocks [10, 12, 15, 16].

However, training and applying a large-scale Swin Transformer model for high-resolution inputs reveals several issues, including training instability. To address these issues, Swin Transformer V2 [24] was proposed, which upgrades the original Swin-Transformer (V1) [23] by using the residual post-normalization instead of pre-normalization⁴, scaled cosine attention instead of the original dot product self-attention⁵, and log-spaced coordinates instead of the previous linear-spaced coordinates. As a result, Swin Transformer V2 has 3 billion parameters and advances state-of-the-art performance on multiple vision task benchmarks.

The U-Transformer is constructed using 48 repeated Swin Transformer V2 blocks and calculates the scaled cosine attention as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = (\cos(\mathbf{Q}, \mathbf{K})/\tau + \mathbf{B})\mathbf{V} \quad (1)$$

where \mathbf{B} represents the relative position bias and τ is a learnable scalar, which is not shared across heads and layers. The cosine function is naturally normalized, which leads to smaller attention values.

The U-Transformer, as the name implies, also includes a downsampling and upsampling block from the U-Net model [25]. The downsampling block, referred to as the Down Block in Figure 2, reduces the data dimension to $C \times 90 \times 180$, thereby minimizing computational and memory requirements for self-attention calculation. The Down Block consists of a 3×3 2-dimensional (2D) convolution layer with a stride of 2, and a residual [26] block that has two 3×3 convolution layers followed by a group normalization (GN) layer [27] and a sigmoid-weighted linear unit (SiLU) activation [28, 29]. The SiLU activation is calculated by multiplying the sigmoid function with its input ($\sigma(x) \times x$). The upsampling block, known as Up Block in Figure 2, has the same residual block as used in the Down Block, along with a 2D transposed convolution [30] with a kernel of 2 and a stride of 2. The Up Block scales the data size back up to $C \times 180 \times 360$. Furthermore, a skip connection is included that concatenates the outputs from the Down Block with those of the transformer blocks before being fed into the Up Block.

3.3 FuXi model training

This section outlines the training process for FuXi models. The training procedure involves two steps: pre-training and fine-tuning, similar to the approach used for training GraphCast [17].

⁴The LN layer is moved from the beginning of each residual unit to the end, producing much milder activation values.

⁵This makes the computation irrelevant to amplitudes of block inputs so that attention values are less likely to fall into extremes.

3.3.1 One-step Pre-training

The pre-training step involves supervised training and optimizing the FuXi model to predict a single time step using the training dataset. The loss function used is the latitude-weighted *L1* loss, which is defined as follows:

$$L1 = \frac{1}{C \times H \times W} \sum_{c=1}^C \sum_{i=1}^H \sum_{j=1}^W a_i | \hat{X}_{c,i,j}^{t+1} - X_{c,i,j}^{t+1} | \quad (2)$$

where C , H , and W are the number of channels and the number of grid points in latitude and longitude direction, respectively. c , i , and j are the indices for variables, latitude and longitude coordinates, respectively. $\hat{X}_{c,i,j}^{t+1}$ and $X_{c,i,j}^{t+1}$ are predicted and ground truth for some variable and locations (latitude and longitude coordinates) at time step of $t + 1$. a_i represents the weight at latitude i and the value of a_i decreases as latitude increases. The *L1* loss is averaged over all the grid points and variables.

The FuXi model is developed using the Pytorch framework [31]. Pre-training of the model requires approximately 30 hours on a cluster of 8 Nvidia A100 GPUs. The model is trained with 40000 iterations using a batch size of 1 on each GPU. The AdamW [32, 33] optimizer is used with parameters $\beta_1=0.9$ and $\beta_2=0.95$, an initial learning rate of 2.5×10^{-4} , and a weight decay coefficient of 0.1. Scheduled DropPath [34] with a dropping ratio of 0.2 is employed to prevent overfitting. In addition, Fully-Sharded Data Parallel (FSDP) [35], bfloat16 floating point precision, and gradient check-pointing [36] are applied to reduce memory costs during model training.

3.3.2 Fine-tuning cascaded models

After pre-training, the base FuXi model is first fine-tuned for optimal performance for 6-hourly forecasts spanning from 0 to 5 days (0-20 time steps). This fine-tuning process is performed using an autoregressive training regime and curriculum training schedule to increase the number of autoregressive steps from 2 to 12, following the fine-tuning approach of the GraphCast model [17]. This fine-tuned model is referred to as FuXi-Short in Figure 2. With weights from FuXi-Short, the FuXi-Medium model is initialized and then fine-tuned for optimal forecast performance for 5 to 10 days (21-40 time steps). Implementing the online inference of FuXi-Short to get output at the 20th time step (5th day), which is required for input to the FuXi-Medium model during its fine-tuning process, is inappropriate due to significant memory consumption and the slowdown of the fine-tuning process for FuXi-Medium. To address this issue, the results of FuXi-Short for six years of data (2012-2017) are cached on a hard disk beforehand. The same procedure for fine-tuning FuXi-Medium is repeated for the fine-tuning of FuXi-Long, optimized for generating forecasts of 10-15 days. Finally, FuXi-Short, FuXi-Medium, and FuXi-Long are cascaded to produce the complete 15-day forecasts. As detailed in Appendix A, cascade helps to reduce accumulation errors and improve forecast performance for longer lead times.

During the fine-tuning process, the model was trained using a constant learning rate of 1×10^{-7} . It takes approximately two days to fine-tune each of the cascaded FuXi models on a cluster of 8 Nvidia A100 GPUs.

3.4 Evaluation method

We follow [5] to evaluate forecast performance using latitude-weighted root mean square error (RMSE) and ACC, which are calculated as follows:

$$RMSE(c, \tau) = \frac{1}{|D|} \sum_{t_0 \in D} \sqrt{\frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W a_i (\hat{X}_{c,i,j}^{t_0+\tau} - X_{c,i,j}^{t_0+\tau})^2} \quad (3)$$

$$ACC(c, \tau) = \frac{1}{|D|} \sum_{t_0 \in D} \frac{\sum_{i,j} a_i (\hat{X}_{c,i,j}^{t_0+\tau} - M_{c,i,j}^{t_0+\tau})(\hat{X}_{c,i,j}^{t_0+\tau} - M_{c,i,j}^{t_0+\tau})}{\sqrt{\sum_{i,j} a_i (\hat{X}_{c,i,j}^{t_0+\tau} - M_{c,i,j}^{t_0+\tau})^2} \sum_{i,j} a_i (\hat{X}_{c,i,j}^{t_0+\tau} - M_{c,i,j}^{t_0+\tau})^2} \quad (4)$$

where t_0 is the forecast initialization time in the testing set D , and τ is the forecast lead time steps added to t_0 . M represents the climatological mean calculated using ERA5 reanalysis data between 1993 and 2016. Additionally, to improve the discrimination of the forecast performance among models with small differences, we use the normalized RMSE difference between model A and baseline B calculated as $(RMSE_A - RMSE_B)/RMSE_B$, and the normalized ACC difference represented by $(ACC_A - ACC_B)/(1 - ACC_B)$. Negative values in normalized RMSE difference and positive values in normalized RMSE difference indicate that model A performs better than the baseline model B .

To evaluate the performance of ECMWF HRES and EM, the verification method applied by ECMWF [1] is used where the model analysis, namely HRES-fc0 and ENS-fc0, serve as the ground truth for HRES and EM, respectively.

4 Results

For evaluating FuXi's performance, the study only uses the 2018 data and selects two daily initialization times (00:00 UTC and 12:00 UTC) to produce 6-hourly forecasts for 15 days.

4.1 Overall performance compared with ECMWF HRES and GraphCast

This subsection compares the forecast performance of FuXi with that of ECMWF HRES and GraphCast in 10-day forecasts. Figure 3 shows the time series of the globally-averaged latitude-weighted ACC and RMSE of FuXi, ECMWF HRES, and GraphCast for 4 surface variables (MSL , $T2M$, $U10$,

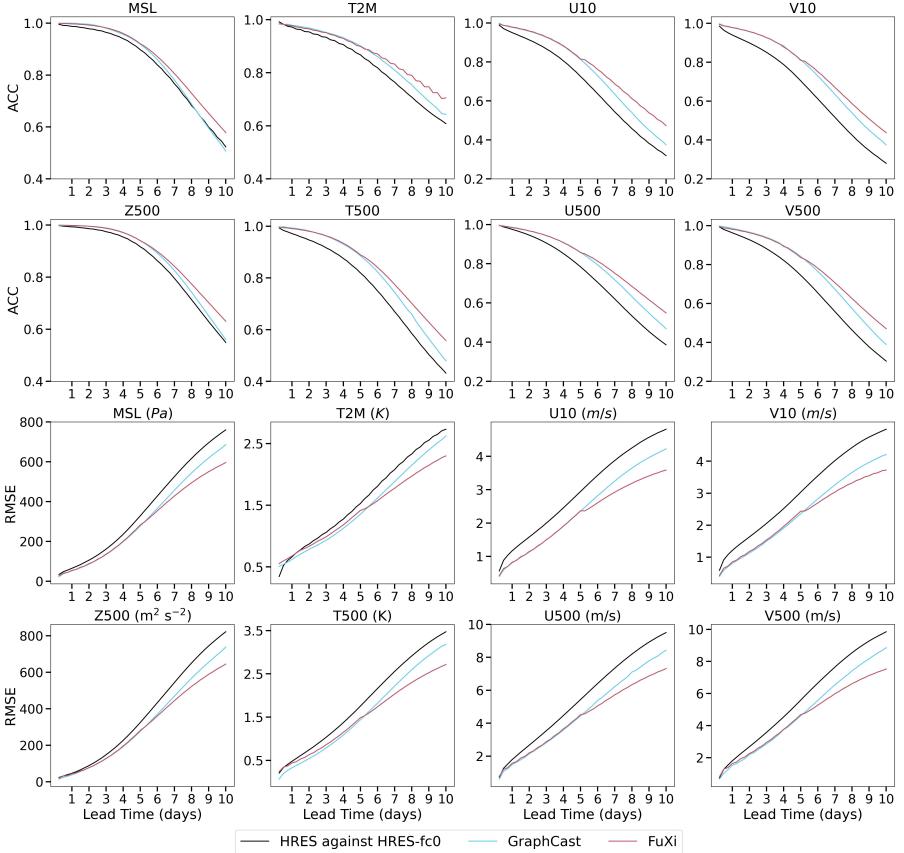


Fig. 3: Comparison of the globally-averaged latitude-weighted ACC (first and second rows) and RMSE (third and fourth rows) of the HRES (black lines), GraphCast (cyan blue lines, which is a state-of-the-art ML-based weather forecast model before FuXi), and FuXi (red lines) for 4 surface variables, such as MSL , $T2M$, $U10$, and $V10$, and 4 upper-air variables at the pressure level of 500 hPa, including $Z500$, $T500$, $U500$, and $V500$, in 10-day forecasts using testing data from 2018. FuXi and GraphCast are evaluated against the ERA5 reanalysis dataset, and ECMWF HRES is evaluated against HRES-fc0.

and $V10$) and 4 upper-air variables ($Z500$, $T500$, $U500$, and $V500$) at the 500 hPa pressure level. The figure illustrates that both FuXi and GraphCast significantly outperform ECMWF HRES. FuXi and GraphCast have comparable performance within forecasts of 7 days, beyond which FuXi shows superior performance, with the lowest values of RMSE and the highest values of ACC across all the variables and forecast lead times. Moreover, FuXi's superior performance becomes increasingly significant as lead times increase. It is worth noting that FuXi also outperforms ECMWF HRES and GraphCast for variables not shown in Figure 3.

4.2 Overall performance compared with ECMWF EM

This subsection presents a comparison of the forecast performance between FuXi and ECMWF EM in 15-day forecasts. Figure 4 shows the time series of the globally-averaged latitude-weighted ACC and RMSE, along with the corresponding normalized difference in ACC and RMSE for 4 variables. The 4 variables include 2 upper-air variables ($Z500$ and $T850$) and 2 surface variables (i.e., MSL and $T2M$). Many combinations of variables and pressure levels are not included in the comparisons as they are unavailable from the ECMWF server. ECMWF EM is used as the baseline for computing the normalized difference in ACC and RMSE, as shown in the 2nd and 4th rows of Figure 4. FuXi outperforms ECMWF EM in 0-9 day forecasts, with positive values in normalized ACC difference and negative values in normalized RMSE difference. However, for forecasts beyond 9 days, FuXi shows slightly poorer performance than ECMWF EM. Overall, FuXi shows comparable performance to ECMWF EM in 15-day forecasts, with higher ACC and lower RMSE than ECMWF EM on 67.92% and 53.75% of the 240 variables, levels, and lead times in the testing set, which includes 2 surface variable and 2 upper-air variables over 15 days, with 4 steps each day. The higher percentage of ACC is probably because the climatological mean used in the computation of the ACC is based on ERA5 data, which is used as the ground truth for training FuXi.

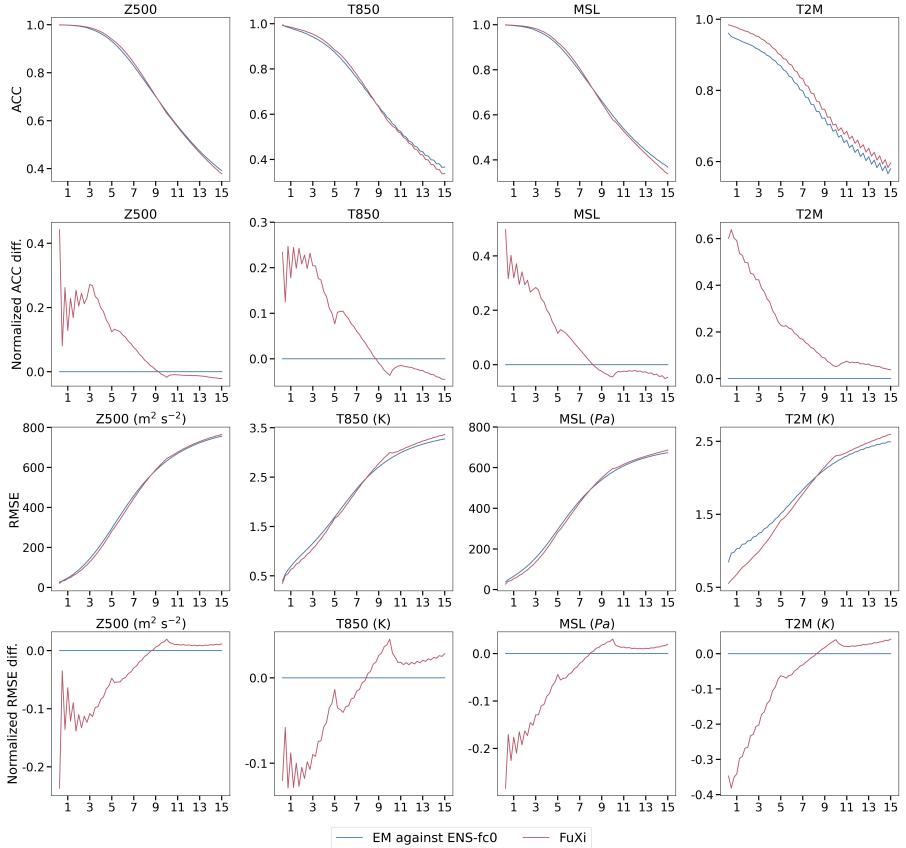


Fig. 4: Comparison of the globally-averaged latitude-weighted ACC (first row) and RMSE (fourth row) as well as normalized ACC (second row) and RMSE difference (fourth row) of ECMWF EM (blue lines) and FuXi (red lines) for 2 upper-air variables, including $Z500$ (first column) and $T500$ (second column), and 2 surface variables, such as MSL (third column) and $T2M$ (fourth column), in 15-day forecasts using testing data from 2018. FuXi is evaluated against the ERA5 reanalysis dataset, and ECMWF EM is evaluated against ENS-fc0.

4.3 Spatial error distribution

Figure 5 illustrates the spatial distributions of the average RMSE of FuXi, the RMSE difference between ECMWF HRES and FuXi, and the RMSE difference between ECMWF EM and FuXi for forecasts of $Z500$ and $T2M$ at lead times of 5 days, 10 days, and 15 days, respectively. All forecasts in the testing data from 2018 were averaged to produce the data. The RMSE difference is represented by red, blue, and white patterns indicating whether ECMWF HRES or ECMWF EM performs worse than, better than, or equally compared to

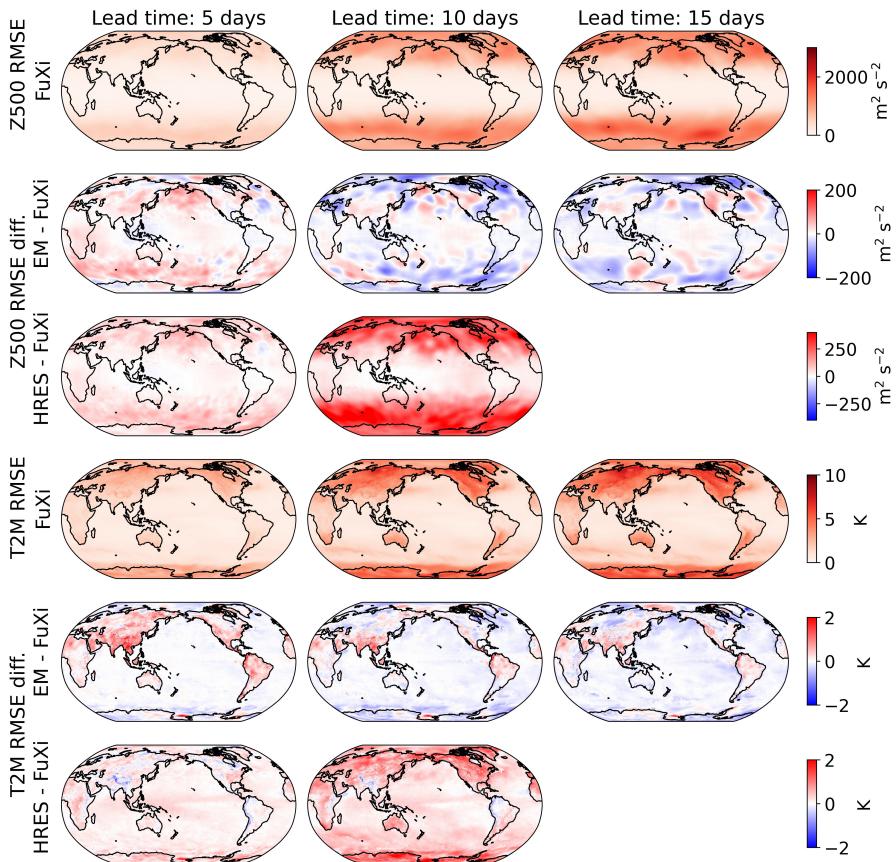


Fig. 5: Spatial map of average RMSE (not latitude-weighted) of FuXi (first and fourth rows), the difference in RMSE between ECMWF HRES (first and fourth rows) and FuXi, and the difference in RMSE between EM (second and fifth rows) and FuXi for $Z500$ (first to third rows) and $T2M$ (fourth to sixth rows) at forecast lead times of 5 days (first column), 10 days (second column), and 15 days (third column), using the 2018 testing data.

FuXi. Overall, all three forecasts have similar spatial error distributions, with the RMSE difference values much lower than the RMSE values. The highest RMSE values appear at high latitudes, while relatively small values are found in middle and low latitudes. The values of RMSE are higher over the land than over the ocean. The RMSE difference between ECMWF HRES and FuXi shows that FuXi outperforms ECMWF HRES in most grid points, as shown by the predominance of red color. In contrast, ECMWF EM shows comparable performance to FuXi in most areas, as indicated by the predominantly white color.

5 Conclusion and Future Work

It has been challenging for data-driven methods to compete with conventional physics-based numerical weather prediction models in weather forecasting due to the difficulty of reducing accumulation error. Recently, ML-based weather forecasting systems have witnessed significant breakthroughs, outperforming ECMWF HRES in 10-day forecasts with a temporal resolution of 6 hours and a spatial resolution of 0.25° [16, 17]. However, employing a single model proves insufficient to obtain optimal performance across various lead times. In order to generate accurate weather forecasts for 15 days, we first develop a powerful base ML model architecture, FuXi model based on the U-Transformer, capable of efficiently learning intricate relationships from vast amounts of high-dimensional weather data. Moreover, we propose a novel cascade ML model architecture for weather forecasting that utilizes three pre-trained FuXi models. Each model is fine-tuned for optimal forecast performance for one of the forecast time windows: 0-5 days, 5-10 days, and 10-15 days. And the models are cascaded to generate the complete 15-day forecasts. By implementing the above-mentioned methodologies, we created FuXi, an ML-based weather forecasting system that, for the first time, performs comparably to ECMWF EM in 15-day forecasts with a temporal and spatial resolution of 6 hours and 0.25° .

Constructing a cascaded model requires the fine-tuning of multiple models. Even though caching is utilized to reduce computational and memory costs, continuous improvement to the model architecture and training methodology is still crucial to accelerate the training process. Current research in large language models (LLM) introduces more efficient methods for fine-tuning large models, including prefix-tuning [37], adapter-tuning [38], and low-rank adaptation (LoRA) [39]. LoRA refers to a training method that accelerates the training of large models with less memory consumption. It introduces pairs of rank-decomposition weight matrices, referred to as update matrices to the existing weights, so that only newly added weights need to be trained. Using LoRA, the number of model parameters can be reduced by 1000 times, and faster training is achieved.

Furthermore, we plan to explore the potential of the cascade ML model architecture for sub-seasonal forecasting by fine-tuning additional models for forecast lead times ranging from 14 to 28 days. Sub-seasonal forecasting remains a challenge and is considered as a "predictability desert" [40]. Unlike medium-range weather forecasting, which can utilize deterministic methods, ensemble forecasts are necessary for sub-seasonal forecasting. In addition, research has identified various processes in the atmosphere, ocean, and land as sources of sub-seasonal predictability, such as the Madden-Julian Oscillation (MJO), soil moisture, snow cover, Stratosphere-troposphere interaction, and ocean conditions [41]. Therefore, more research is needed to develop an ML-based sub-seasonal forecasting system.

In addition, one limitation of current ML-based weather forecasting methods is that they are not yet completely end-to-end, as they still rely on analysis data generated by conventional NWP models for initial conditions. Thus, we

aim to develop a data-driven data assimilation method that uses observation data to generate initial conditions for ML-based weather forecasting systems. Looking forward to the future, we aim to build a truly end-to-end, systematically unbiased, and computationally efficient ML-based weather forecasting system.

Acknowledgements

We appreciate the researchers at ECMWF for their efforts in collecting, archiving, disseminating, and maintaining the ERA5 reanalysis dataset, HRES, and ensemble, without which this study would not have been feasible.

Competing interests

The authors declare no competing interests.

References

- [1] Haiden, T., Janousek, M., Vitart, F., Ben-Boualleuge, Z., Ferranti, L., Prates, F.: Evaluation of ecmwf forecasts, including the 2021 upgrade (2021). <https://doi.org/10.21957/90pgicjk4>
- [2] Magnusson, L., Bidlot, J.-R., Bonavita, M., Brown, A.R., Browne, P.A., Chiara, G.D., Dahoui, M., Lang, S.T.K., McNally, T., Mogensen, K.S., Pappenberger, F., Prates, F., Rabier, F., Richardson, D.S., Vitart, F., Malardel, S.: Ecmwf activities for improved hurricane forecasts. Bulletin of the American Meteorological Society **100**(3), 445–458 (2019). <https://doi.org/10.1175/BAMS-D-18-0044.1>
- [3] Frnáda, J., Durica, M., Rozhon, J., Vojteková, M., Nedoma, J., Martínek, R.: Ecmwf short-term prediction accuracy improvement by deep learning. Scientific Reports **12**(1) (2022). <https://doi.org/10.1038/s41598-022-11936-9>
- [4] Schultz, M.G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L.H., Mozaffari, A., Stadtler, S.: Can deep learning beat numerical weather prediction? Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **379**(2194), 20200097 (2021) <https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2020.0097>. <https://doi.org/10.1098/rsta.2020.0097>
- [5] Rasp, S., Dueben, P.D., Scher, S., Weyn, J.A., Mouatadid, S., Thuerey, N.: Weatherbench: a benchmark data set for data-driven weather forecasting. Journal of Advances in Modeling Earth Systems **12**(11), 2020–002203 (2020)

- [6] Garg, S., Rasp, S., Thuerey, N.: Weatherbench probability: A benchmark dataset for probabilistic medium-range weather forecasting along with deep learning baseline models. arXiv preprint arXiv:2205.00865 (2022)
- [7] Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., et al.: The era5 global reanalysis. Quarterly Journal of the Royal Meteorological Society **146**(730), 1999–2049 (2020)
- [8] Rasp, S., Thuerey, N.: Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for weatherbench. Journal of Advances in Modeling Earth Systems **13**(2), 2020–002405 (2021)
- [9] Weyn, J.A., Durran, D.R., Caruana, R.: Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. Journal of Advances in Modeling Earth Systems **12**(9), 2020–002109 (2020)
- [10] Hu, Y., Chen, L., Wang, Z., Li, H.: Swinrvnn: A data-driven ensemble forecasting model via learned distribution perturbation. Journal of Advances in Modeling Earth Systems **15**(2), 2022–003211 (2023) <https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2022MS003211>. <https://doi.org/10.1029/2022MS003211>. e2022MS003211 2022MS003211
- [11] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
- [12] Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., et al.: Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. arXiv preprint arXiv:2202.11214 (2022)
- [13] Guibas, J., Mardani, M., Li, Z., Tao, A., Anandkumar, A., Catanzaro, B.: Adaptive Fourier Neural Operators: Efficient Token Mixers for Transformers (2022)
- [14] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021). <https://openreview.net/forum?id=YicbFdNTTy>
- [15] Chen, L., Du, F., Hu, Y., Wang, F., Wang, Z.: SwinRDM: Integrate Swin-RNN with Diffusion Model Towards High-Resolution and High-Quality Weather Forecasting. (2023). <https://doi.org/10.48448/zn7f-fc64>

- [16] Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., Tian, Q.: Pangu-Weather: A 3D High-Resolution Model for Fast and Accurate Global Weather Forecast (2022)
- [17] Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., Ravuri, S., Ewalds, T., Alet, F., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Stott, J., Vinyals, O., Mohamed, S., Battaglia, P.: GraphCast: Learning skillful medium-range global weather forecasting (2022)
- [18] Dueben, P.D., Bauer, P.: Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development* **11**(10), 3999–4009 (2018). <https://doi.org/10.5194/gmd-11-3999-2018>
- [19] Ho, J., Saharia, C., Chan, W., Fleet, D.J., Norouzi, M., Salimans, T.: Cascaded Diffusion Models for High Fidelity Image Generation (2021)
- [20] Li, H., Lin, Z., Shen, X., Brandt, J., Hua, G.: A convolutional neural network cascade for face detection. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5325–5334 (2015). <https://doi.org/10.1109/CVPR.2015.7299170>
- [21] Tong, Z., Song, Y., Wang, J., Wang, L.: VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training (2022)
- [22] Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer Normalization (2016)
- [23] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012–10022 (2021)
- [24] Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., Guo, B.: Swin Transformer V2: Scaling Up Capacity and Resolution (2022)
- [25] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention, pp. 234–241 (2015). Springer
- [26] He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition (2015)
- [27] Wu, Y., He, K.: Group Normalization (2018)

- [28] Elfwing, S., Uchibe, E., Doya, K.: Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning (2017)
- [29] Ramachandran, P., Zoph, B., Le, Q.V.: Searching for Activation Functions (2017)
- [30] Zeiler, M.D., Krishnan, D., Taylor, G.W., Fergus, R.: Deconvolutional networks. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2528–2535 (2010). <https://doi.org/10.1109/CVPR.2010.5539957>
- [31] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: NIPS 2017 Workshop on Autodiff (2017). <https://openreview.net/forum?id=BJJsrnfCZ>
- [32] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (2017)
- [33] Loshchilov, I., Hutter, F.: Decoupled Weight Decay Regularization (2017)
- [34] Larsson, G., Maire, M., Shakhnarovich, G.: FractalNet: Ultra-Deep Neural Networks without Residuals (2017)
- [35] Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C.-C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., Desmaison, A., Balioglu, C., Nguyen, B., Chauhan, G., Hao, Y., Li, S.: PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel (2023)
- [36] Chen, T., Xu, B., Zhang, C., Guestrin, C.: Training Deep Nets with Sublinear Memory Cost (2016)
- [37] Li, X.L., Liang, P.: Prefix-Tuning: Optimizing Continuous Prompts for Generation (2021)
- [38] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-Efficient Transfer Learning for NLP (2019)
- [39] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-Rank Adaptation of Large Language Models (2021)
- [40] Vitart, F., Robertson, A.W., Anderson, D.: Subseasonal to seasonal prediction project: bridging the gap between weather and climate. (2012)

- [41] Robertson, A.W., Vitart, F., Camargo, S.J.: Subseasonal to seasonal prediction of weather to climate with application to tropical cyclones. *Journal of Geophysical Research: Atmospheres* **125**(6), 2018–029375 (2020)
<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2018JD029375>.
<https://doi.org/10.1029/2018JD029375>. e2018JD029375
10.1029/2018JD029375

Appendix

A Effectiveness of the cascade model architecture

This section discusses the effect of the cascade ML model architecture in reducing accumulation errors for weather forecasting. We use a single FuXi model (FuXi-short) to generate 15-day forecasts and compare its performance with FuXi. Figure A.1 shows the comparable performance for forecasts of $Z500$ and $T2M$ between the single base FuXi model and FuXi for lead times ranging from 0 to 7 days. However, the performance of the single FuXi base model decreases considerably as lead times increase due to accumulation error.

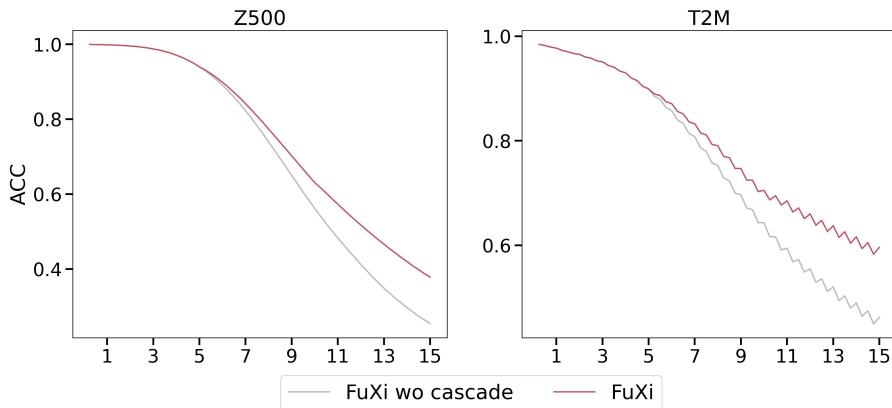


Fig. A.1: Comparison of the globally-averaged latitude-weighted ACC of the FuXi system (red lines) and FuXi without cascade (gray lines) using testing data from 2018.