Supplement of Geosci. Model Dev., 16, 3611–3628, 2023
https://doi.org/10.5194/gmd-16-3611-2023-supplement

Supplement of

Convective-gust nowcasting based on radar reflectivity and a deep learning algorithm

Haixia Xiao et al.

Correspondence to: Yangqiang Wang (yqwang@cma.gov.cn) and Yu Zheng (zhengyu@cma.gov.cn)

S1 Supplement figures

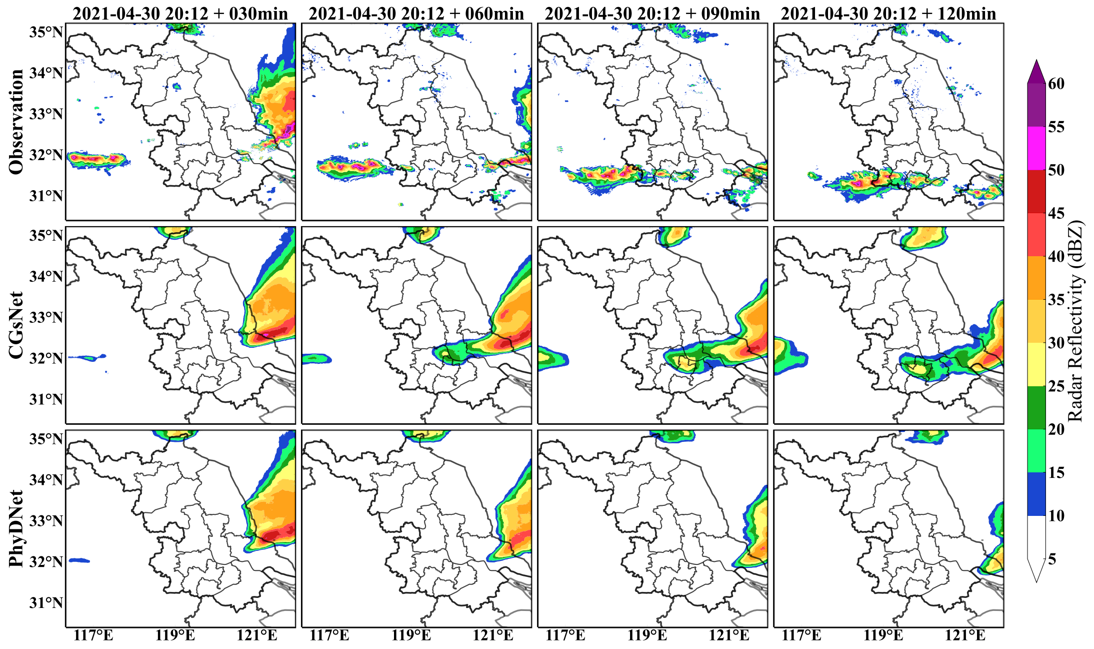

Figure S1. Observations (first row) and forecasts (second row) of ROMS in eastern China, 30 April 2021, 20:12-22:12 BJT. Note that forecasting started at 20:12 BJT, and the observations and forecasts are shown for every 30 min.

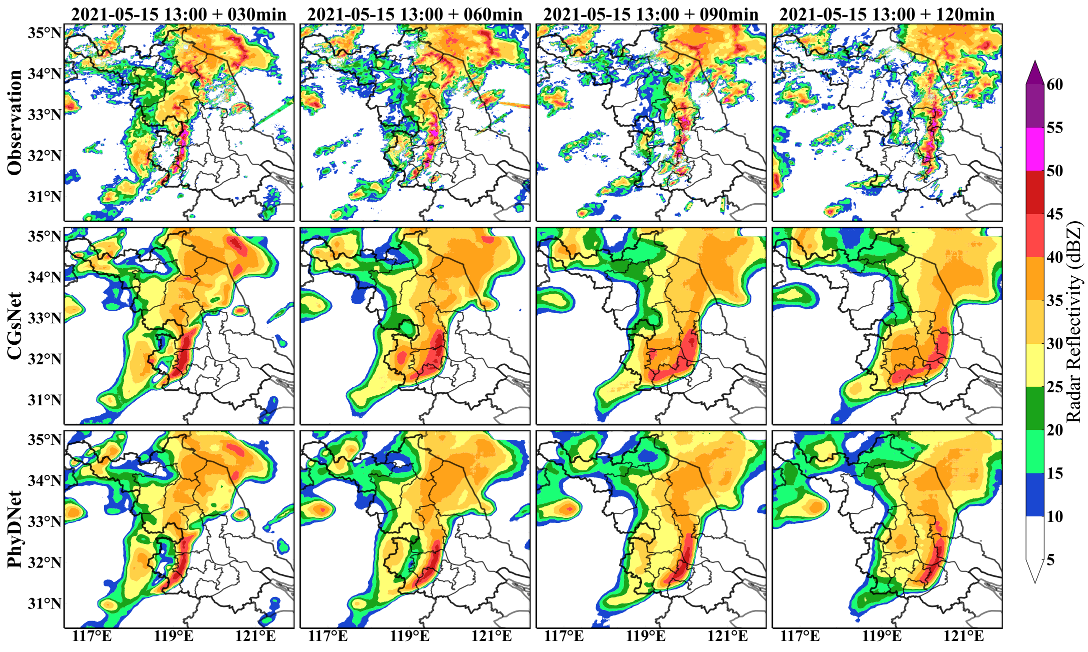

Figure S2. Same as Figure S1 but for 15 May 2021, 13:00-15:00 BJT.

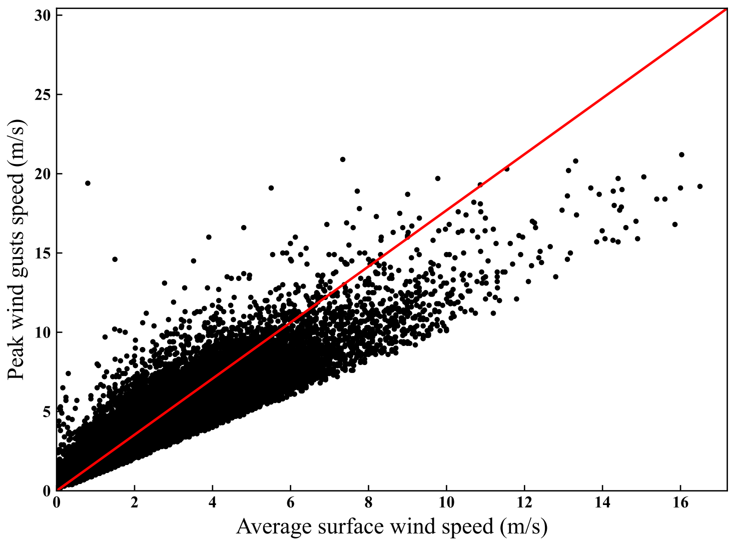

Figure S3. Scatter plot of the hourly maximum ASWS and PWGS. The red line represents the linear regression line for the GF.

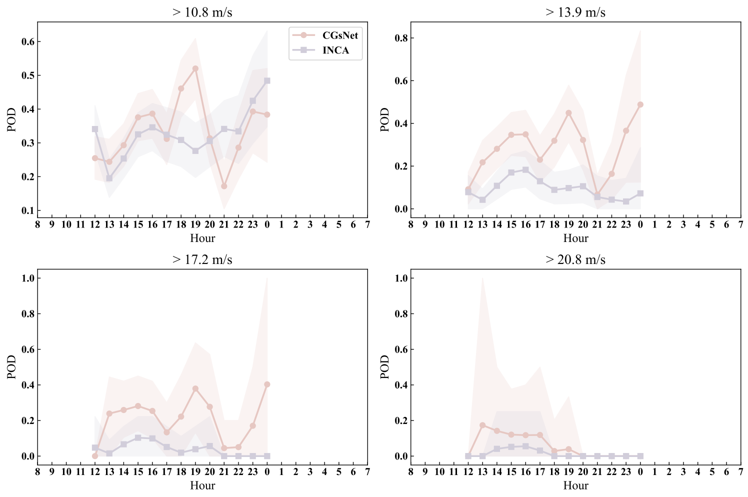

Figure S4. The POD results of PWGS forecasts from CGsNet and INCA for different hours of a day at thresholds of 10.8, 13.9, 17.2, and 20.8 m s^{-1}.The shaded pink and purple areas represent the 95% confidence intervals of the CGsNet and INCA indices, respectively.

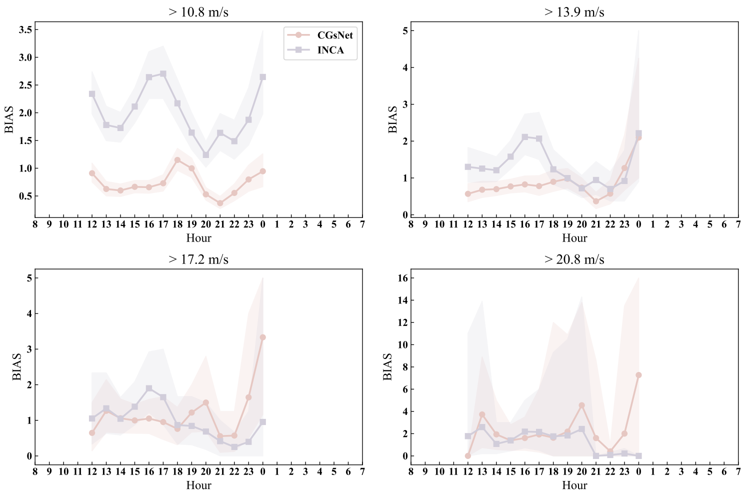

Figure S5. Same as Figure S4 but for bias.

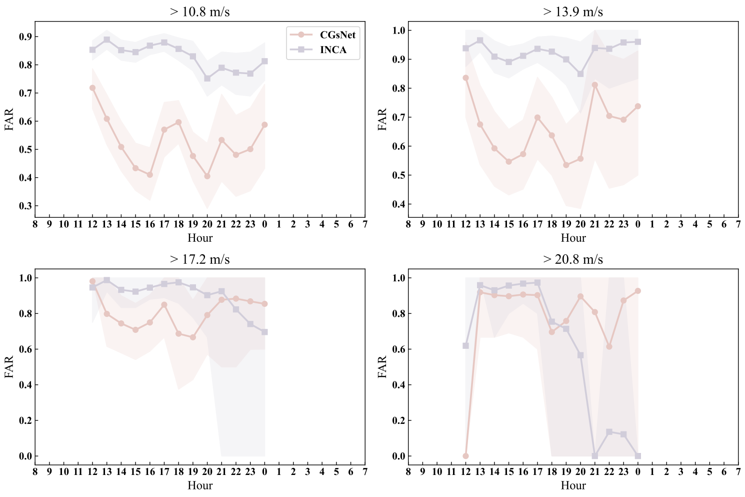

Figure S6. Same as Figure S4 but for FAR.

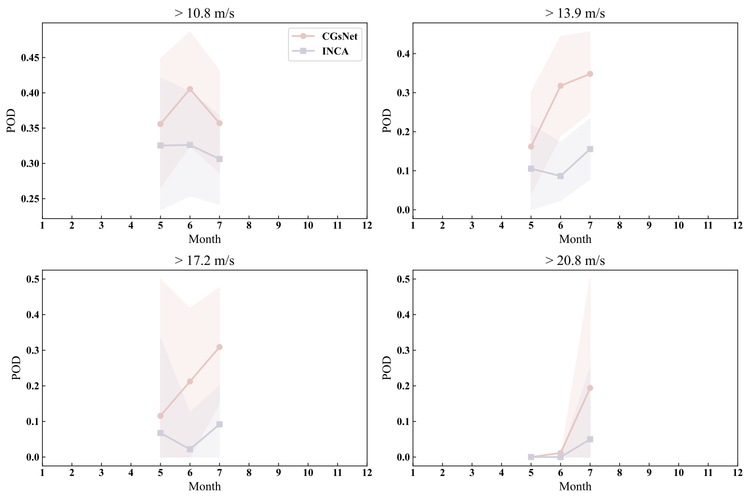

Figure S7. The POD results of PWGS forecasts from CGsNet and INCA for different months of a year at thresholds of 10.8, 13.9, 17.2, and 20.8 m s^{-1}.The shaded pink and purple areas represent the 95% confidence intervals of the CGsNet and INCA indices, respectively.

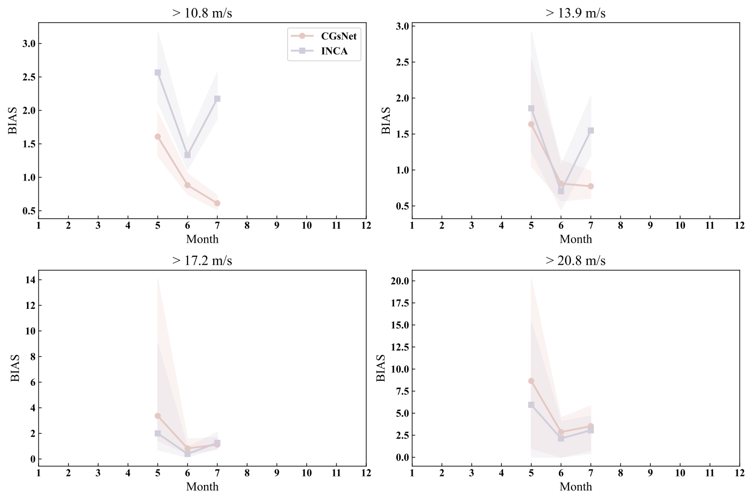

Figure S8. Same as Figure S7 but for bias.

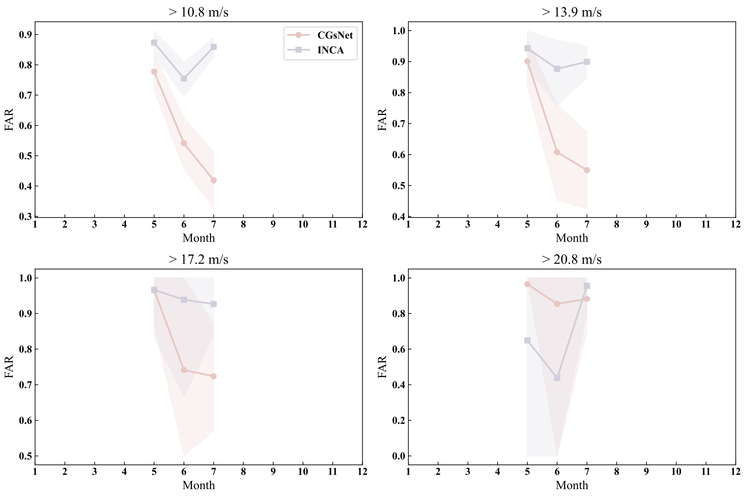

Figure S9. Same as Figure S7 but for FAR.

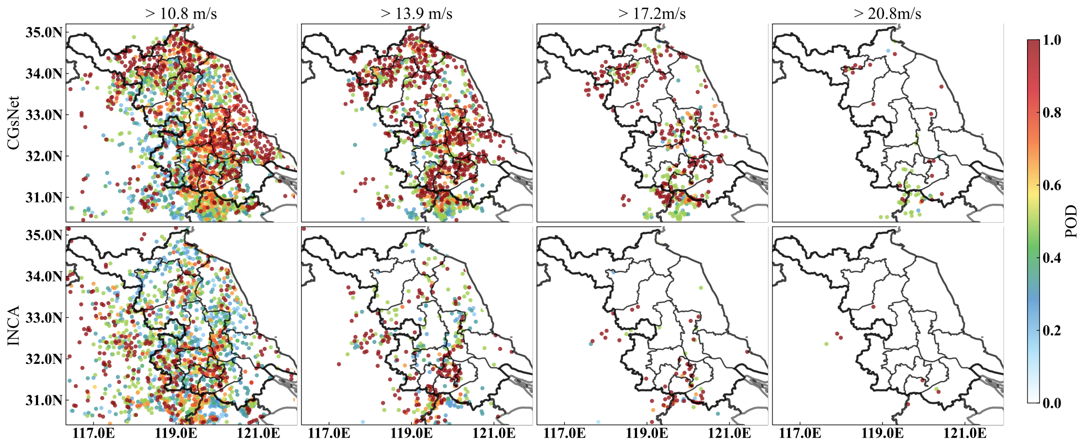

Figure S10. The POD results of PWGS forecasts from CGsNet and INCA for different areas at thresholds of 10.8, 13.9, 17.2, and 20.8 m s^{-1}.

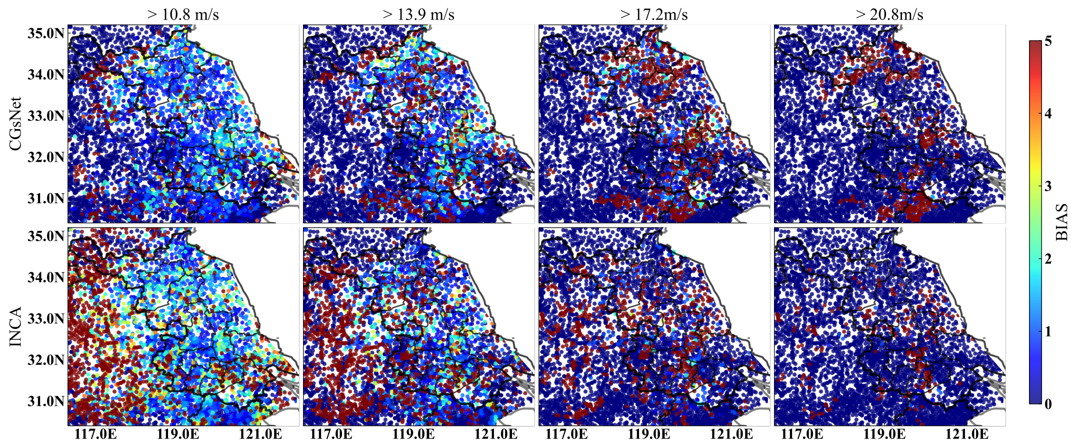

Figure S11. Same as Figure S10 but for bias.

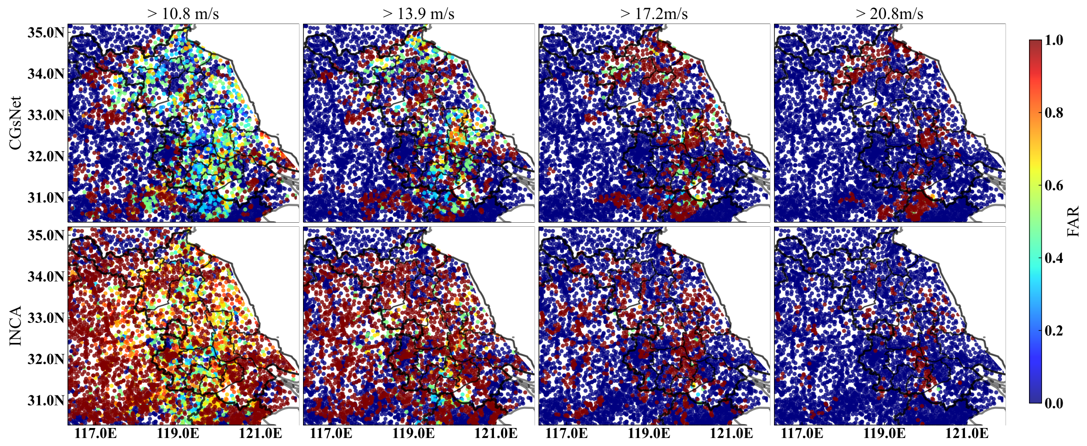

Figure S12. Same as Figure S10 but for FAR.

S2 The detailed structure of CGsNet

S2.1 The convolutional encoder and decoder

The detailed structures of the convolutional encoder and decoder are shown in Table S1. The input image shape is $batch\ size \times sequence\ length \times channel \times width \times height$. Specifically, batch size is set to 2, input sequence length is 10. The feature map shapes showed in the convolutional and deconvolutional layers are just $channel \times width \times height$. After each convolutional layer in the encoder, there is a Group Normalization followed by a LeakyReLU activation function. Similarly, in the decoder, a Group Normalization and a LeakyReLU activation function follow each deconvolution layer, except for the fourth deconvolution layer. Note the feature map shapes of $\mathbf{h}^E$, $\mathbf{h}^p$, $\mathbf{h}^c$ and $\mathbf{h}^m$ are both $128 \times 30 \times 35$.

Table S1. Parameter settings in encoder and decoder. "Conv" denotes convolutional layer in encoder, “Deconv” denotes deconvolutional layer in decoder.

Encoder	Input size	Kernel size /stride	Decoder	Input size	Kernel size /stride
Conv1	$2 \times 480 \times 560$	$3 \times 3/(2,2)$	Deconv1	$128 \times 30 \times 35$	$3 \times 3/(2,2)$
Conv2	$16 \times 240 \times 280$	$3 \times 3/(2,2)$	Deconv2	$64 \times 60 \times 70$	$3 \times 3/(2,2)$
Conv3	$32 \times 120 \times 140$	$3 \times 3/(2,2)$	Deconv3	$32 \times 120 \times 140$	$3 \times 3/(2,2)$
Conv4	$64 \times 60 \times 70$	$3 \times 3/(2,2)$	Deconv4	$16 \times 240 \times 280$	$3 \times 3/(2,2)$
Encoder output size: $128 \times 30 \times 35$			Decoder output size:$2 \times 480 \times 560$		

S2.2 The calculation process of PhyDNet and ConvLSTM

S2.2.1 PhyCell

PhyCell is a physical cell of PhyDNet (Guen and Thome, 2020b), whose dynamics are governed by the PDE response function $\boldsymbol{\mathcal{M}}_p(\mathbf{h}^{\mathbf{p}}, \mathbf{u})$:

$$\boldsymbol{\mathcal{M}}_p(\mathbf{h}, \mathbf{u}) := \Phi(\mathbf{h}) + \mathcal{C}(\mathbf{h}, \mathbf{u}) \tag{S1}$$

where $\Phi(\mathbf{h})$ represents a physical predictor modeling only the latent dynamics, and the $\mathcal{C}(\mathbf{h}, \mathbf{u})$ represents a correction term, modeling the interaction between input data and latent state. $\Phi(\mathbf{h})$ can be modeled as:

$$\Phi(\mathbf{h}(t, \mathbf{x})) = \sum_{i,j:i+j \leq q} C_{i,j} \frac{\partial^{i+j}\mathbf{h}}{\partial x^i \partial y^j}(t, \mathbf{x}) \tag{S2}$$

$\Phi(\mathbf{h}(t, \mathbf{x}))$ combines the spatial derivatives with coefficients $c_{i,j}$ up to a certain differential order q. A wide range of classical physical models, e.g., the wave equation and heat equation, can be subsumed in this generic class of linear PDEs.

The discrete time PhyCell with the standard forward Euler numerical scheme can be written as:

$$\mathbf{h}_{t+1} = (1 - \mathbf{K}_t) \odot (\mathbf{h}_t + \Phi(\mathbf{h}_t)) + \mathbf{K}_t \odot \mathbf{E}(\mathbf{u}_t) \tag{S3}$$

where $\odot$ denotes the Hadamard product; $\mathbf{K}_t$ is a gating factor; and $\mathbf{E}(\mathbf{u}_t)$ indicates the new observed input. Here we write the equivalent two-steps for Eq S3:

$$\tilde{\mathbf{h}}_{t+1} = \mathbf{h}_t + \Phi(\mathbf{h}_t) \qquad \textbf{Predition} \tag{S4}$$

$$\mathbf{h}_{t+1} = \tilde{\mathbf{h}}_{t+1} + \mathbf{K}_t \odot (\mathbf{E}(\mathbf{u}_t) - \tilde{\mathbf{h}}_{t+1}) \qquad \textbf{Correction} \tag{S5}$$

The Eq S4 represents the prediction step, which is a physically-constrained motion in latent space, and it computes the intermediate representation: $\tilde{\mathbf{h}}_{t+1}$. The correction step in Eq S5 incorporates the input data. The decoupling between prediction and correction can be used to robustly train the model in missing data contests and long-term forecasting. Besides, the trade-off between both steps is controlled by $\mathbf{K}_t$, which can be interpreted as the Kalman gain.

The physical predictor Φ in Eq S4 is implemented by using a convolutional neural network, based on the connection between convolutions and differentiations. The 1×1 convolutions are used to linearly combine the derivatives with $c_{i,j}$ coefficients in Eq S2. Moreover, the Kalman gain $\mathbf{K}_t$ is approximated by a gate with learned convolution kernels $\mathbf{W}_h$, $\mathbf{W}_u$ and bias $\mathbf{b}_k$:

$$\mathbf{K}_t = \tanh(\mathbf{W}_h * \tilde{\mathbf{h}}_{t+1} + \mathbf{W}_u * \mathbf{E}(\mathbf{u}_t) + \mathbf{b}_k) \tag{S6}$$

where * respent the convolutional operator. If $\mathbf{K}_t = 0$, the dynamic follows the physical predictor; if $\mathbf{K}_t = 1$, the latent will be reset and only driven by the input.

PhyCell is an atomic recurrent cell used for building physically-constrained RNNs. The PhyDNet here uses one layer of PhyCell, which can also be easily stacked to build more complex models.

S2.2.2 ConvLSTM

ConvLSTM is a variant of the long short-term memory network (Shi et al., 2015), which is a fundamental and effective spatiotemporal recurrent structure for spatiotemporal modeling. The ConvLSTM uses the forget gate, input gate, and output gate to update its cell and hidden states. The input gate controls how much of the new information will be added to the memory cell. The forget gate is used to control how much of the previous information will be forgotten from the memory cell, while the cell information which will be propagated to the new gate is controlled by the output gate. The calculation processes of the ConvLSTM are as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi} * \mathbf{x}_t + \mathbf{W}_{hi} * \mathbf{h}_{t-1} + \mathbf{W}_{ci} \odot \mathbf{c}_{t-1} + \mathbf{b}_i) \tag{S7}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf} * \mathbf{x}_t + \mathbf{W}_{hf} * \mathbf{h}_{t-1} + \mathbf{W}_{cf} \odot \mathbf{c}_{t-1} + \mathbf{b}_f) \tag{S8}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{xc} * \mathbf{x}_t + \mathbf{W}_{hc} * \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{S9}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo} * \mathbf{x}_t + \mathbf{W}_{ho} * \mathbf{h}_{t-1} + \mathbf{W}_{co} \odot \mathbf{c}_t + \mathbf{b}_o) \tag{S10}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_{t-1}) \tag{S11}$$

where σ is the sigmoid activation function. Besides, $\mathbf{x}_t$, $\mathbf{c}_t$, $\mathbf{h}_t$, $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ represent input, memory cell, hidden state, input gate, forget gate, and output gate, respectively. Bias $\mathbf{b}$ and weight $\mathbf{W}$ both represent learning parameters.

S3 The parameters choices

S3.1 The parameters of IDW calculation

Here we first briefly explain the IWD calculation steps, as follows: (1) Calculating the distance d from unknown points to all points;(2) Calculating the weight λ for each point, the weight is a function of the reciprocal of the distance:$\lambda_i = \frac{d_i^{-p}}{\sum\limits_{i=1}^{K} d_i^{-p}}$, where K is the total number of discrete points; Power (p) is a parameter used to calculate the influence weight of the K nearest discrete points on the interpolation site, controlling the effect of known points on the interpolation value based on their distance from the interpolated point.(3) Calculating interpolation results:$\hat{z}(x_0, y_0) = \sum\limits_{i=1}^{n} \lambda_i z(x_i, y_i)$, where (x_0, y_0) is the interpolation point coordinate and (x_i, y_i) represents the coordinates of discrete points.

To determine the optimal parameters for the IDW interpolation, we experimented with various combinations of the number of stations (K) and radius of influence (R) (Figure S13). In IDW interpolation, R refers to finding the K nearest discrete points

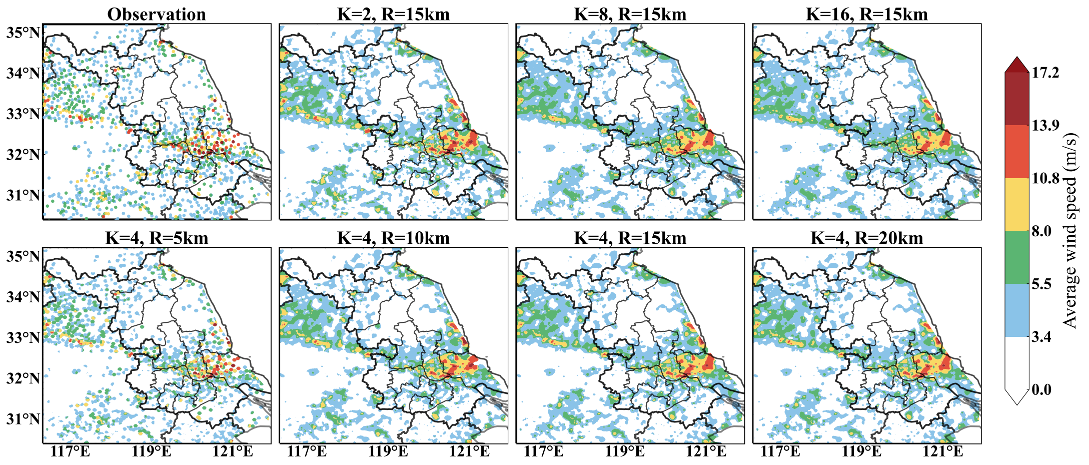

Figure S13. The various combinations of radius of influence, the number of stations for IDW interpolation.

from the interpolation site within a radius range of R. As shown in the Figure S13, it is evident that when R is set to 5 km, the condition of having nearest discrete points/stations within a 5 km radius is not met, resulting in many 'nan' values. This is because the distance between meteorological observation stations is 10 km. Therefore, when R is less than 10 km, the observed field cannot be adequately interpolated. However, when R is set to 10 km or lager, the interpolation results do not significantly change. For K, when taking different values ranging from 2 to 16, the interpolation results are comparable. As K increases, the value of the interpolated point is progressively smoothed by the surrounding discrete points. Generally, when the value of R is not less than 10 km and the value of K is not too large, the interpolation results of the observed wind field are similar, and the results are effective. Thus using a R of 15 km and K= 4 staions are feasible. The choice of 15 km radius of influence allowed us to capture local variations in the data without being overly influenced by distant stations, which might not have the same meteorological conditions. Using 4 stations ensured that the interpolation incorporated sufficient data points.

For p, as the p value increases, the interpolated value will gradually approach the value of the nearest point. By specifying a small p value, the influence of points farther away will be great, resulting in a smooth plane. Based on a comprehensive analysis, we selected a common value of p=2. The p of 2 provided a balance between the weighting of nearby and distant stations, which has been widely used in many studies.

S3.2 The hyperparameters of CGsNet training

(1) Batch size: the batch size of 2 was chosen based on two factors. First, a small batch size also has good generalization ability since each batch is randomly selected, allowing the model to adapt better to new data. Second, a small batch size helps to reduce memory usage and accelerate training, as the samples occupy a significant amount of memory.
(2) Learning rate: the learning rate was set based on the settings of PhyDNet (Guen and Thome, 2020b), which involved selecting an initial learning rate of 0.001 and decreasing it if the loss function did not decrease after several epochs of training.
(3) Epoch: setting the number of epochs to 50 is to ensure that the model is fully trained. When saving the model, the one with the lowest validation loss was selected and saved.

S3.3 The parameters of loss fuction

The thresholds parameters of loss function was mainly based on the wind speed classification determined by the China Meteorological Administration (reference to this URL), as shown in the following Table S2. High weights were assigned to high

Table S2. The wind speed levels defined in China.

Wind level	**Wind speed (m s^{-1})**
0	[0.0-0.3)
1	[0.3-1.6)
2	[1.6-3.4)
3	[3.4-5.5)
4	[5.5-8.0)
5	[8.0-10.8)
6	[10.8-13.9)
7	[13.9-17.2)
8	[17.2-20.8)
9	[20.8-24.5)
......	

wind speeds and RMOS values since accurate predictions in these ranges are crucial for ensuring life safety and minimizing potential damage caused by extreme weather events. For example, wind speed greater than 20.8 m s^{-1} are considered to be particularly hazardous, and thus, predictions in this range were given a high weight. Besides, the issue of imbalance between high wind speed data and low wind speed data can be partially addressed by assigning different weights, as there is a scarcity of high wind speed data samples and an abundance of low wind speed data samples. This approach can help alleviate the problem of the insufficient number of high wind speed data samples and enable the model to forecast areas where strong gusts occur, which is a significant concern. The specific weights were set with reference to the settings of Shi et al (2015) and expert advice.

S4 The calculation of bootstrapping

(1) A new dataset is created by randomly selecting 100 observations (an area of 480×560) using sampling with replacement from the testing dataset.

(2) Compute the performance metrics such as HSS, TS, RMSE, and MAE based on the new dataset.

(3) Repeat steps 1) and 2) 1,000 times to generate multiple values of HSS, TS, RMSE, MAE, and other metrics.

(4) Sort the generated values for each metric in ascending order.

(5) Choose a 95% confidence level and calculating the 2.5_{th} and 97.5_{th} percentiles of the metric values, which provide a 95% confidence interval for the metrics.