Supporting Information for “Probabilistic diffusion model for stochastic parameterization – a case example of numerical precipitation estimation”

Baoxiang Pan[1], Leyi Wang[2], Feng Zhang[3], Qingyun Duan[4], Xin Li[5], Xiaoduo Pan[5], Xi Chen[1], Fenghua Ling[6], Shuguang Wang[7], Ming Pan[8], Ziniu Xiao[1]

[1]Institute of Atmospheric Physics, Chinese Academy of Sciences

[2]Chongqing Institute of Big Data, Peking University

[3]Department of Atmospheric and Oceanic Sciences, Fudan University

[4]National Key Laboratory of Water Disaster Prevention, Hohai University

[5]Institute of Tibetan Plateau Research, Chinese Academy of Sciences

[6]Institute for Climate and Application Research, Nanjing University of Information Science and Technology

[7]School of Atmospheric Sciences, Nanjing University

[8]Scripps Institution of Oceanography, University of California San Diego

Contents of this file

S1. Details of probabilistic diffusion model

The theory and practice of probabilistic diffusion models can be math-heavy and convoluted. We provide mathematical and implementation details of the deployed probabilistic diffusion model. For a friendly tutorial, see Luo (2022). For more information and useful learning materials, see Sohl-Dickstein et al. (2015), Ho et al. (2020), Kingma et al. (2021), and Song et al. (2020).

S1.1 Decomposition of $\log p(\mathbf{y})$ using latent $\mathbf{z}_{0:1}$

Diffusion model is an explicit likelihood based generative model. Its learning objective function is a factorization of data likelihood defined over latent variables $\mathbf{z}_{0:1} = \{\mathbf{z}_{t_0=0}, \mathbf{z}_{t_1}, \mathbf{z}_{t_2}, ..., \mathbf{z}_{t_T=1}\}$. This factorization stands at the core of variational view of diffusion models. A step-by-step derivation is given below.

First, for a pre-defined $p(\mathbf{z}_{0:1}|\mathbf{y})$, we have:

$$\log p(\mathbf{y}) = E_{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big[\log p(\mathbf{y})\Big] = E_{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big[\log \frac{p(\mathbf{z}_{0:1}, \mathbf{y})}{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big] \tag{1}$$

Given $p(\mathbf{z}_t|\mathbf{y}) := \mathcal{N}(\alpha_t \mathbf{y}, \sigma_t^2 \mathbf{I})$, $0 = t_0 < t_1 < t_2 < ... < t_T = 1$, we have:

$$\begin{aligned} p(\mathbf{z}_{0:1}, \mathbf{y}) &= p(\mathbf{y}|\mathbf{z}_{0:1}) p(\mathbf{z}_{t_0}|\mathbf{z}_{t_1:t_T}) p(\mathbf{z}_{t_1}|\mathbf{z}_{t_2:t_T})...p(\mathbf{z}_{t_{T-1}}|\mathbf{z}_{t_T}) p(\mathbf{z}_{t_T}) \\ &= p(\mathbf{y}|\mathbf{z}_{t_0}) p(\mathbf{z}_{t_0}|\mathbf{z}_{t_1}) p(\mathbf{z}_{t_1}|\mathbf{z}_{t_2})...p(\mathbf{z}_{t_{T-1}}|\mathbf{z}_{t_T}) p(\mathbf{z}_{t_T}) \\ &= p(\mathbf{y}|\mathbf{z}_{t_0}) p(\mathbf{z}_{t_T}) \prod_{i=1}^{T} p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}) \end{aligned} \tag{2}$$

and

$$\begin{aligned} p(\mathbf{z}_{0:1}|\mathbf{y}) &= p(\mathbf{z}_{t_0}|\mathbf{y}) p(\mathbf{z}_{t_1}|\mathbf{z}_{t_0}, \mathbf{y}) p(\mathbf{z}_{t_2}|\mathbf{z}_{t_0:t_1}, \mathbf{y})...p(\mathbf{z}_{t_T}|\mathbf{z}_{t_0:t_{T-1}}, \mathbf{y}) \\ &= p(\mathbf{z}_{t_0}|\mathbf{y}) p(\mathbf{z}_{t_1}|\mathbf{z}_{t_0}) p(\mathbf{z}_{t_2}|\mathbf{z}_{t_1})...p(\mathbf{z}_{t_T}|\mathbf{z}_{t_{T-1}}) \\ &= p(\mathbf{z}_{t_0}|\mathbf{y}) \prod_{i=1}^{T} p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}}) \end{aligned} \tag{3}$$

Plug Eq. 2 and Eq. 3 into Eq. 1, we have:

$$\begin{aligned}\log p(\mathbf{y}) =& E_{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big[\log \frac{p(\mathbf{z}_{0:1},\mathbf{y})}{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big] \\ =& E_{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big[\log \frac{p(\mathbf{y}|\mathbf{z}_{t_0})p(\mathbf{z}_{t_T})\prod_{i=1}^{T}p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})}{p(\mathbf{z}_{t_0}|\mathbf{y})\prod_{i=1}^{T}p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}})}\Big] \\ =& E_{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big[\log \frac{p(\mathbf{y}|\mathbf{z}_{t_0})p(\mathbf{z}_{t_T})}{p(\mathbf{z}_{t_0}|\mathbf{y})} + \sum_{i=1}^{T}\log \frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})}{p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}})}\Big]\end{aligned} \tag{4}$$

While Eq. 4 can be decomposed into differentiable terms, it involves estimating expectation over two random variables: $\{\mathbf{z}_{t_{i-1}}, \mathbf{z}_{t_i}\}$, which may have high variance (Luo, 2022). To achieve a robust estimate, we re-write $p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}})$ as:

$$p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}}) = p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}},\mathbf{y}) = \frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i},\mathbf{y})p(\mathbf{z}_{t_i}|\mathbf{y})}{p(\mathbf{z}_{t_{i-1}}|\mathbf{y})} \tag{5}$$

Plug Eq. 5 into Eq. 4, we have:

$$\begin{aligned}\log p(\mathbf{y}) =& E_{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big[\log \frac{p(\mathbf{y}|\mathbf{z}_{t_0})p(\mathbf{z}_{t_T})}{p(\mathbf{z}_{t_0}|\mathbf{y})} + \sum_{i=1}^{T}\log \frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})}{p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}})}\Big] \\ =& E_{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big[\log \frac{p(\mathbf{y}|\mathbf{z}_{t_0})p(\mathbf{z}_{t_T})}{p(\mathbf{z}_{t_0}|\mathbf{y})} + \sum_{i=1}^{T}\log \frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})}{\frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i},\mathbf{y})p(\mathbf{z}_{t_i}|\mathbf{y})}{p(\mathbf{z}_{t_{i-1}}|\mathbf{y})}}\Big] \\ =& E_{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big[\log \frac{p(\mathbf{y}|\mathbf{z}_{t_0})p(\mathbf{z}_{t_T})}{p(\mathbf{z}_{t_0}|\mathbf{y})} + \sum_{i=1}^{T}\log \frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})}{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i},\mathbf{y})} + \sum_{i=1}^{T}\log \frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{y})}{p(\mathbf{z}_{t_i}|\mathbf{y})}\Big] \\ =& E_{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big[\log \frac{p(\mathbf{y}|\mathbf{z}_{t_0})p(\mathbf{z}_{t_T})}{p(\mathbf{z}_{t_0}|\mathbf{y})} + \sum_{i=1}^{T}\log \frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})}{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i},\mathbf{y})} + \log \frac{p(\mathbf{z}_{t_0}|\mathbf{y})}{p(\mathbf{z}_{t_T}|\mathbf{y})}\Big] \\ =& E_{p(\mathbf{z}_{0:1}|\mathbf{y})}\Big[\log p(\mathbf{y}|\mathbf{z}_{t_0}) + \sum_{i=1}^{T}\log \frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})}{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i},\mathbf{y})} + \log \frac{p(\mathbf{z}_{t_T})}{p(\mathbf{z}_{t_T}|\mathbf{y})}\Big] \\ =& E_{p(\mathbf{z}_{t_0}|\mathbf{y})}\log p(\mathbf{y}|\mathbf{z}_{t_0}) + E_{p(\mathbf{z}_T|\mathbf{y})}\log \frac{p(\mathbf{z}_{t_T})}{p(\mathbf{z}_{t_T}|\mathbf{y})} + \sum_{i=1}^{T}E_{p(\mathbf{z}_i|\mathbf{y})}\log \frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})}{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i},\mathbf{y})} \\ =& E_{p(\mathbf{z}_{t_0}|\mathbf{y})}\log p(\mathbf{y}|\mathbf{z}_{t_0}) + E_{p(\mathbf{z}_T|\mathbf{y})}\log \frac{p(\mathbf{z}_{t_T})}{p(\mathbf{z}_{t_T}|\mathbf{y})} + \sum_{i=1}^{T}E_{p(\mathbf{z}_i|\mathbf{y})}\log \frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})}{p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i},\mathbf{y})} \\ =& \mathbb{E}_{p(\mathbf{z}_{t_0}|\mathbf{y})}\log p(\mathbf{y}|\mathbf{z}_{t_0}) - D_{\mathrm{KL}}\big(p(\mathbf{z}_{t_T}|\mathbf{y})||p(\mathbf{z}_{t_T})\big) - \sum_{i=1}^{T}\mathbb{E}_{p(\mathbf{z}_{t_i}|\mathbf{y})}D_{\mathrm{KL}}\big(p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i},\mathbf{y})||p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})\big) \\ =& \mathbb{E}_{p(\mathbf{z}_0|\mathbf{y})}\log p(\mathbf{y}|\mathbf{z}_0) - D_{\mathrm{KL}}\big(p(\mathbf{z}_1|\mathbf{y})||p(\mathbf{z}_1)\big) - \sum_{i=1}^{T}\mathbb{E}_{p(\mathbf{z}_{t_i}|\mathbf{y})}D_{\mathrm{KL}}\big(p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i},\mathbf{y})||p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})\big)\end{aligned} \tag{6}$$

We now take a close examination of the three terms on the right side of Eq. 6:

- $\mathbb{E}_{p(\mathbf{z}_0|\mathbf{y})}\log p(\mathbf{y}|\mathbf{z}_0)$:] given that $\alpha_0 := 1, \sigma_0 := 0$, we have $p(\mathbf{z}_0|\mathbf{y}) = \begin{cases}\delta, \mathbf{z}_0 = y \\ 0, \text{else}\end{cases}$, and

$p(\mathbf{y}|\mathbf{z}_0) = \begin{cases}\delta, y = \mathbf{z}_0 \\ 0, \text{else}\end{cases}$, δ is Dirac function. Thus, we have $\mathbb{E}_{p(\mathbf{z}_0|\mathbf{y})}\log p(\mathbf{y}|\mathbf{z}_0) = 0$.

- $D_{\mathrm{KL}}\big(p(\mathbf{z}_1|\mathbf{y})||p(\mathbf{z}_1)\big)$: given that $\alpha_1 \coloneqq 0, \sigma_1 \coloneqq 1$, we have $p(\mathbf{z}_1|\mathbf{y}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, which is agnostic of $\mathbf{y}$, this leads to $p(\mathbf{z}_1) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, therefore, $D_{\mathrm{KL}}\big(p(\mathbf{z}_1|\mathbf{y})||p(\mathbf{z}_1)\big) = 0$.
- $\sum_{i=1}^{T} \mathbb{E}_{p(\mathbf{z}_{t_i}|\mathbf{y})} D_{\mathrm{KL}}\big(p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y})||p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}\big)$: for any i, we can derive an analytical Gaussian form of $p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y})$ (see below). For fine enough discretization of time, $p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})$ is also Gaussian, which is represented as a variational distribution parameterized by neural networks. Thus, we obtain an analytical form of $D_{\mathrm{KL}}\big(p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y})||p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}\big)$, suitable for stochastic gradient optimization.

Given the analysis above, to maximize $\log p(\mathbf{y})$ is approximately equivalent to minimizing the following time averaging Kullback–Leibler divergence term:

$$\log p(\mathbf{y}) \approx -\sum_{i=1}^{T} \mathbb{E}_{p(\mathbf{z}_{t_i}|\mathbf{y})} D_{\mathrm{KL}}\big(p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y})||p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})\big) \tag{7}$$

Below we derive analytical form of $p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y})$ and relate it with score estimate of $p(\mathbf{z}_{t_i}|\mathbf{y})$, which enables robust optimization and high-quality generative modeling.

S1.2 Analytical/parameterized form of $p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y})/p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}})$

To derive the analytical form of $p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y})$, we have:

$$\begin{aligned} p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y}) =& p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}}, \mathbf{y})\frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{y})}{p(\mathbf{z}_{t_i}|\mathbf{y})} = p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}})\frac{p(\mathbf{z}_{t_{i-1}}|\mathbf{y})}{p(\mathbf{z}_{t_i}|\mathbf{y})} \\ &\propto \exp\Big(-\frac{1}{2}\big(\frac{(\mathbf{z}_{t_i} - \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\mathbf{z}_{t_{i-1}})^2}{\sigma_{t_i}^2 - \frac{\alpha_{t_i}^2}{\alpha_{t_{i-1}}^2}\sigma_{t_{i-1}}^2} + \frac{(\mathbf{z}_{t_{i-1}} - \alpha_{t_{i-1}}\mathbf{y})^2}{\sigma_{t_{i-1}}^2} - \frac{(\mathbf{z}_{t_i} - \alpha_{t_i}\mathbf{y})^2}{\sigma_{t_i}^2}\big)\Big) \\ &= \exp\Big(-\frac{1}{2}\big(\frac{1}{\sigma_{t_{i-1}}^2(1 - \frac{\alpha_{t_i}^2\sigma_{t_{i-1}}^2}{\alpha_{t_{i-1}}^2\sigma_{t_i}^2})}\mathbf{z}_{t_{i-1}}^2 - 2\big(\frac{\frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\mathbf{z}_{t_i}}{\sigma_{t_i}^2 - \frac{\alpha_{t_i}^2}{\alpha_{t_{i-1}}^2}\sigma_{t_{i-1}}^2} + \frac{\alpha_{t_{i-1}}\mathbf{y}}{\sigma_{t_{i-1}}^2}\big)\mathbf{z}_{t_{i-1}} + C(\mathbf{z}_{t_i}, \mathbf{y})\big)\Big) \end{aligned} \tag{8}$$

Hence $p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y}) = \mathcal{N}(\tilde{\mu}_{t_i}, \tilde{\mathbf{\Sigma}}_{t_i})$, where:

$$\begin{aligned} \tilde{\mu}_{t_i} &= \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}\mathbf{z}_{t_i} - \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}(\sigma_{t_i}^2 - \frac{\alpha_{t_i}^2}{\alpha_{t_{i-1}}^2}\sigma_{t_{i-1}}^2)\frac{\mathbf{z}_{t_i} - \alpha_{t_i}\mathbf{y}}{\sigma_{t_i}^2} \\ &= \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}\mathbf{z}_{t_i} + \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}(\sigma_{t_i}^2 - \frac{\alpha_{t_i}^2}{\alpha_{t_{i-1}}^2}\sigma_{t_{i-1}}^2)\nabla \log p(\mathbf{z}_{t_i}|\mathbf{y}) \end{aligned} \tag{9}$$

and

$$\tilde{\mathbf{\Sigma}}_{t_i} = \sigma_{t_{i-1}}^2(1 - \frac{\sigma_{t_{i-1}}^2}{\sigma_{t_i}^2}\frac{\alpha_{t_i}^2}{\alpha_{t_{i-1}}^2})\mathbf{I} \tag{10}$$

Given fine enough discretization of time, $p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})$ is also Gaussian, i.e. $p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}) = \mathcal{N}(\mu_{t_i}, \mathbf{\Sigma}_{t_i})$. We parameterize μ_{t_i} in accordance with the analytical form of $\tilde{\mu}_{t_i}$:

$$\mu_{t_i} = \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}\mathbf{z}_{t_i} + \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}(\sigma_{t_i}^2 - \frac{\alpha_{t_i}^2}{\alpha_{t_{i-1}}^2}\sigma_{t_{i-1}}^2)\epsilon_{\mathrm{NN}}(\mathbf{z}_{t_i}) \tag{11}$$

where $\epsilon_{\mathrm{NN}}(\mathbf{z}_{t_i})$ is neural network parameterization of $\nabla \log p(\mathbf{z}_{t_i}|\mathbf{y})$. Following Ho & Salimans (2022), we consider the following simplied representation of $\mathbf{\Sigma}_{t_i}$:

$$\mathbf{\Sigma}_{t_i} = \frac{\sigma_{t_i}^{2v_{t_i}}}{\sigma_{t_{i-1}}^{2v_{t_i}}}\tilde{\mathbf{\Sigma}}_{t_i} \tag{12}$$

where $v_{t_i} \coloneqq 0.5$ for all time steps.

Given the analytical/parameterized form of $p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y})/p(\mathbf{z}_{t_i}|\mathbf{z}_{t_{i-1}})$, we have:

$$\begin{aligned}
\epsilon_{\mathrm{NN}}^*(\mathbf{z}_{t_i}) &= \underset{\epsilon_{\mathrm{NN}}(\mathbf{z}_{t_i})}{\operatorname{argmax}} \log p(\mathbf{y}) \\
&\approx \underset{\epsilon_{\mathrm{NN}}(\mathbf{z}_{t_i})}{\operatorname{argmin}} \mathbb{E}_{p(\mathbf{z}_{t_i}|\mathbf{y})} D_{\mathrm{KL}}\big(p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}, \mathbf{y})||p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})\big) \\
&= \underset{\epsilon_{\mathrm{NN}}(\mathbf{z}_{t_i})}{\operatorname{argmin}} \mathbb{E}_{p(\mathbf{z}_{t_i}|\mathbf{y})}\big[(\tilde{\mu}_{t_i} - \mu_{t_i})^T \Sigma_{t_i}^{-1}(\tilde{\mu}_{t_i} - \mu_{t_i})\big] \\
&= \underset{\epsilon_{\mathrm{NN}}(\mathbf{z}_{t_i})}{\operatorname{argmin}} \mathbb{E}_{p(\mathbf{z}_{t_i}|\mathbf{y})}\big[\tilde{\mathbf{\Sigma}}_{t_i}\frac{\sigma_{t_i}^{2(1-v_{t_i})}}{\sigma_{t_{i-1}}^{2(1-v_{t_i})}}\big\|\nabla \log p(\mathbf{z}_{t_i}|\mathbf{y}) - \epsilon_{\mathrm{NN}}(\mathbf{z}_{t_i})\big\|_2\big] \\
&\approx \underset{\epsilon_{\mathrm{NN}}(\mathbf{z}_{t_i})}{\operatorname{argmin}} \mathbb{E}_{p(\mathbf{z}_{t_i}|\mathbf{y})}\big\|\nabla \log p(\mathbf{z}_{t_i}|\mathbf{y}) - \epsilon_{\mathrm{NN}}(\mathbf{z}_{t_i})\big\|_2
\end{aligned} \tag{13}$$

We apply standard stochastic gradient descent to obtain $\epsilon_{\mathrm{NN}}^*(\mathbf{z}_{t_i})$. Thereafter, we can approximate $\{\mu_{t_i}, \mathbf{\Sigma}_{t_i}\}$ using Eq. 11 and 12. Based on $p_\theta(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i})$, we carry out iterative ancestral sampling: note that $p(\mathbf{z}_1) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ given the forward Gaussian Process setup. Therefore, starting from standard Gaussian samples, we iteratively generate samples of $\mathbf{z}_{t_{T-1}}, \mathbf{z}_{t_{T-2}}, ..., \mathbf{z}_{t_0}$, using the learned distributions of $p(\mathbf{z}_{t_{i-1}}|\mathbf{z}_{t_i}), i = T, T-1, ..., 1$. Finally, $p(\mathbf{y}|\mathbf{z}_0) = \begin{cases} \delta, & y = \mathbf{z}_0 \\ 0, & \text{else} \end{cases}$.

S1.3 $\{\alpha_t, \sigma_t\}$: noise schedule

In diffusion model, we define a Gaussian process to map target distribution to a standard Gaussian. $\{\alpha_t, \sigma_t\}$ specifies the noise schedule, quantifying how fast the target distribution is diminished through the diffusion process. A better noise schedule allows efficient optimization and improved model likelihood, which may be achieved via optimization (Kingma et al., 2021). Here, for simplicity, we adopt a pre-defined noise schedule, following Ho & Salimans (2022) and Nichol & Dhariwal (2021). Specifically, we parameterize $\{\alpha_t, \sigma_t\}$ as a function of λ_t:

$$\alpha_t^2 = 1 - \sigma_t^2 = \frac{1}{1 + e^{-\lambda_t}} \tag{14}$$

where:

$$\lambda_t = -2 \log \tan(at + b) \tag{15}$$

Here $b = \arctan(e^{-\frac{\lambda_{\max}}{2}})$, $a = \arctan(e^{-\frac{\lambda_{\min}}{2}}) - b$, t is uniformly sampled from $[0, 1]$. $\{\lambda_{\min} = -20, \lambda_{\max} = 20\}$ are hyper-parameters. This noise schedule represents a hyperbolic secant distribution modified to be supported on a bounded interval (Ho & Salimans, 2022).

S1.4 Encoding of diffusion time step

Diffusion model is an iterative generative model, involving a hierarchy of neural network models $\epsilon_{\text{NN}}(\mathbf{z}_t)$ to approximate score functions $\nabla \log p(\mathbf{z}_t|\mathbf{y})$ at multiple noise levels. While this hierarchy of neural network models can be learned separately, in practice, we often adopt a time-dependent neural network, using an vector embedding of t to account for the impact of learning objective difference for different noise levels. Following Song et al. (2020), we incorporate the time information via Gaussian random features, i.e.: $\text{embedding}(t) = [\sin(2\pi\omega t); \cos(2\pi\omega t)]$, where $\omega \sim \mathcal{N}(\mathbf{0}, s\mathbf{I})$, $s = 1$ is a pre-defined scaling parameter.

S1.5 Model architecture and training details

The neural networks we apply for unconditional/conditional score estimates are time-dependent UNets with structures illustrated in Fig. S1 and Fig. S2. For now we do not include attention mechanism for computation efficiency. Both models take into input of noisified precipitation field and nosification scale, and outputs the score estimate. We embed the time information, and stack the time embedding as extra channel to all UNet blocks. Each contract block consists of a long chain of $\{C_{3\times3} + N + \mathrm{ReLU}\}_3$, and a short chain of $\{C_{1\times1}\}_1$, concatenated as a residual block, $C_{n\times n}$ is convolution layer with kernel receptive field of size $n \times n$, N is group normalization, ReLU is rectified linear unit function. Each expand block consists of a long chain of $\{R_2+C_{3\times3}+N+\mathrm{ReLU}\}_3$, and a short chain of $\{R_2, C_{1\times1}\}_1$, concatenating as a residual block, R_n resize the data by n times using linear interpolation. We start with channel size of 128, and double/shrink the channel size by 2 along each contract/expand block. For the conditional score estimating neural network, we includes the conditioning information. This conditioning information is deterministic precipitation estimation, offered by a separate UNnet that takes into input of dynamical field information. In this sense, the conditional score estimating neural network tries to recover and add details of the precipitation information discarded by the deterministic precipitation estimator.

We use data from 1979-2016/2017-2018/2019-2022 for training/validation/test. We keep same data splitting strategy for all data-driven models considered in this study. To train the unconditional model, we randomly crop precipitation field data of size 80×80 ($8^\circ \times 8^\circ$), add random scale noise to the data, and use the unconditional diffusion model to estimate the score. We use ADAM optimizer and an initial learning rate of 10^{-3}. We halve the learning rate if validation loss is not decreasing for 10 epochs. To train the conditional model, we include conditioning information from a UNet based deterministic precipitation estimate, the rest settings are same as the unconditional case.

S2. Baseline models

S2.1 UNet

We consider UNet as 1) a deterministic baseline and 2) the conditioning information extractor for all the generative models. UNet a unique convolutional neural network architecture suited for image-relevant tasks. Here, the model takes into input of resolved dynamical field information and static elevation information, and outputs a deterministic precipitation field estimate. The dynamical field information is provided by a 9-hour (including 3 previous/current/future hours), $8^\circ \times 8^\circ$ circulation field data, with 19 channels representing 19 dynamical variables, including key primitive variables (meridional and zonal wind velocity, temperature, specific humidity, and geopotential height) at 3 pressure levels (1000/850/500 hPa), and crucial surface level variables (sea level pressure, surface pressure, surface temperature, and total column precipitable water). This dynamical field information is first pre-processed through 3D convolution blocks, and concatenated with preprocessed elevation information, before feeding into a 2D UNet. The UNet applies a convolution based contracting path to capture precipitation relevant dynamical field information, and a symmetric transposed convolution based expanding path to gradually refine precipitation field estimates. Skip connections between symmetrical convolution and transposed convolution blocks are applied to force deeper neural network layers to learn meaningful representations that are not well captured by shallower layers. The learning objective is to minimize the squared error between estimated and observed precipitation. Underlying this objective function is the assumption that $p(\mathbf{y}|\mathbf{x})$ is Gaussian, with identical error covariance for any $\mathbf{x}$ and any grid point. See Fig. S3 for UNet model architecture.

S2.2 Conditional variational autoencoder (CVAE)

Conditional variational autoencoder (CVAE) is deep neural network powered probabilistic graphical model. To learn a non-linear latent variable model for the target conditional distribution $p(\mathbf{y}|\mathbf{x})$, CVAE constructs a bijective mapping between $p(\mathbf{y}|\mathbf{x})$ and a tractable latent distribution $p(\mathbf{z}|\mathbf{x})$, using an encoder-decoder neural network architecture. The encoder q_ϕ approximates $p(\mathbf{z}|\mathbf{x}, \mathbf{y})$ as a variational Gaussian distribution; the decoder p_ψ approximates $p(\mathbf{y}|\mathbf{x}, \mathbf{z})$ using the conditioning information $\mathbf{x}$ and the learned latent vector $\mathbf{z}$. To approximate the target conditional distribution, $\{q_\phi, p_\psi\}$ are jointly trained to maximize the following evidence lower bound (ELBO) of the data log likelihood:

$$\mathrm{ELBO} = \mathbb{E}_{\mathbf{z}\sim q_\phi} \log p_\psi - \beta D_{\mathrm{KL}}\big(q_\phi \| p(\mathbf{z}|\mathbf{x})\big) \tag{16}$$

Here $p(\mathbf{z}|\mathbf{x})$ is assumed to be standard Gaussian; β is a parameter balancing sample diversity and sample accordance to the conditioning information, similar to the functionality of ω in diffusion model. To train CVAE, we run mini-batches of $\{\mathbf{x}, \mathbf{y}\}$ samples through $\{q_\phi, p_\psi\}$, and update their parameters to maximize the ELBO, using stochastic gradient ascent. To generate novel samples of $p(\mathbf{y}|\mathbf{x})$, we draw $\mathbf{z}$ samples from $p(\mathbf{z}|\mathbf{x})$ and pass them together with $\mathbf{x}$ through the optimal p^*_ψ. See Fig. S4 for model architecture details.

S2.3 Conditional generative adversarial net (CGAN)

Conditional generative adversarial net (CGAN) approximates a target conditional distribution $p(\mathbf{y}|\mathbf{x})$ by setting up a "game" between two neural networks. The generator network G takes into input of the conditioning information $\mathbf{x}$ and random noise $\mathbf{z}$ to create samples that are intended to come from the target distribution; the discriminator network D is a binary classifier, optimized to differentiate between generated samples and true samples:

$$L_D = \mathbb{E}_{\mathbf{y}}\big(D(\mathbf{y})\big) - \mathbb{E}_{\mathbf{z}}\Big(D\big(G(\mathbf{x}, \mathbf{z})\big)\Big) \tag{17}$$

The generator network is optimized to 1) fool the discriminator network, and 2) draw the mean of generated samples close to ground truth observations ():

$$L_G = \mathbb{E}_{\mathbf{z}}\Big(D\big(G(\mathbf{x},\mathbf{z})\big)\Big) - \lambda \mathbb{E}_{\{\mathbf{y},\mathbf{z}\}}||G(\mathbf{x},\mathbf{z}) - \mathbf{y}||_2 \tag{18}$$

Here λ is a parameter that balances sample diversity and sample accordance to the conditioning information, similar to the functionality of ω/β in diffusion/CVAE model. To train CGAN, we run mini-batches of $\{\mathbf{x},\mathbf{y},\mathbf{z}\}$ samples through $\{G, D\}$, and apply stochastic gradient ascent to maximize L_D and L_G. We keep the optimal G^* if it offers best skill performance (measured by continuous ranked probabilistic skill score, as it is a proper scoring rule, see below) for the validation set. To generate novel samples of $p(\mathbf{y}|\mathbf{x})$, we draw $\mathbf{z}$ samples from random noise and pass them together with $\mathbf{x}$ through the optimal G^*. See Fig. S5 for model architecture details.

S2.4 Dynamical downscaling using WRF

We include comparison to a dynamical simulation approach for numerical precipitation estimation. Here the Advanced Research Version 4.2 of Weather Research and Forecasting (WRF-ARW V4.2, Skamarock et al. 2019) is deployed for simulation of a Typhoon precipitation case (Typhoon Lekima, 0000 UTC 04 August 2019 -0000 UTC 12 August 2019). WRF-ARW refines the coarsely resolved climate processes at regional scale, using high-resolution numerical geophysical fluid dynamics solver and a suite of accompanying parameterization schemes. We apply Global Forecast System reanalysis data to provide the initial and boundary condition for the considered precipitation cases. We apply spectral nudging of wind for the outer domain to ensure consistency between the simulated large-scale circulations and the analysis fields. The simulated domains are delineated in Fig. S6. The selected parameterization schemes as listed in Tab. S1.

S3. Evaluation metrics

S3.1 Human eYe Perceptual Evaluation (HYPE)

We apply a simplified Human eYe Perceptual Evaluation (HYPE) to assess the sample quality of models' precipitation estimates, relying on human climate scientists' and climate model end-users' perceptions. We measure human climate experts' error rate in detecting observations that are randomly mixed with model generated samples. We report the test takers' accuracy rate in five tests.

S3.2 Power spectral analysis

We inspect the spatial structure of precipitation estimates by computing their average spectrum power as function of spatial frequencies and orientations. The computation steps are as follows:

Step 1: Transform the precipitation field data to frequency domain, using Fast Fourier Transform.

Step 2: Compute the power spectrum by taking the squared magnitude of the Fourier Transform coefficients.

Next, for radial averaged power spectrum analysis:

Step 1: Define a set of concentric circles centered at the origin of the frequency domain, along each radial line, calculate the average power by averaging the power spectrum values corresponding to the points intersected by the line.

Step 2: Plot the average power values against the corresponding radial frequency.

For orientation averaged power spectrum analysis:

Step 1: Define a set of concentric circles centered at the origin of the frequency domain, along each orientation angle, calculate the average power by averaging the power spectrum values corresponding to the points through this orientation angle.

Step 2: Plot the average power values against the corresponding orientation angle.

S3.3 Spread-Skill Correlation (SSC)

The spatial correlation coefficient between the standard deviation of model's ensemble, and the mean absolute error of model's ensemble mean:

$$\mathrm{SSC} = \frac{\sum_{i=1}^{n}(\sigma_i - \bar{\sigma})(\epsilon_i - \bar{\epsilon})}{\sqrt{\sum_{i=1}^{n}(\sigma_i - \bar{\sigma})^2 \cdot \sum_{i=1}^{n}(\epsilon_i - \bar{\epsilon})^2}} \tag{19}$$

where σ_i is the standard deviation of model's ensemble at grid i:

$$\sigma_i = \sqrt{\frac{1}{J-1}\sum_{j=1}^{J}(\hat{y}_i^j - \bar{\hat{y}}_i)^2} \tag{20}$$

ϵ_i is the mean absolute error of model's ensemble mean at grid i:

$$\sigma_i = |\bar{\hat{y}}_i - y_i| \tag{21}$$

J is ensemble size; $\hat{y}_i^j$ is the jth ensemble estimate at grid i; $\bar{\hat{y}}_i$ is ensemble mean estimate; y_i is observation.

S3.4 Coverage Ratio (CR)

The percentage that grid observation falls into the coverage of ensemble spread.

S3.5 Pearson correlation coefficient (r)

The Pearson correlation coefficient (r) between ensemble mean prediction $\hat{y}$ and observation y is calculated as follows:

$$r = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2 \cdot \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{22}$$

S3.6 Root mean squared error (RMSE)

The root mean square error (RMSE) between ensemble mean prediction $\hat{y}$ and observation y is calculated as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{23}$$

S3.7 Continuous ranked probabilistic score (CRPS)

The continuous ranked probability score (CRPS) is defined as:

$$\mathrm{CRPS}(F, x) = \int_{-\infty}^{\infty} \left[F(\hat{y}) - \mathbb{I}(\hat{y} \geq y)\right]^2 \, dy \tag{24}$$

where $F(\hat{y})$ is the cumulative distribution function (CDF) of the predictive distribution, y is the observed value, and $\mathbb{I}(\cdot)$ is the indicator function.

References

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, 6840–6851.

Ho, J., & Salimans, T. (2022). Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Kingma, D., Salimans, T., Poole, B., & Ho, J. (2021). Variational diffusion models. *Advances in neural information processing systems*, *34*, 21696–21707.

Luo, C. (2022). Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*.

Nichol, A. Q., & Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. In *International conference on machine learning* (pp. 8162–8171).

Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Liu, Z., Berner, J., ... others (2019). A description of the advanced research wrf version 4. *NCAR tech. note ncar/tn-556+ str*, *145*.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning* (pp. 2256–2265).

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

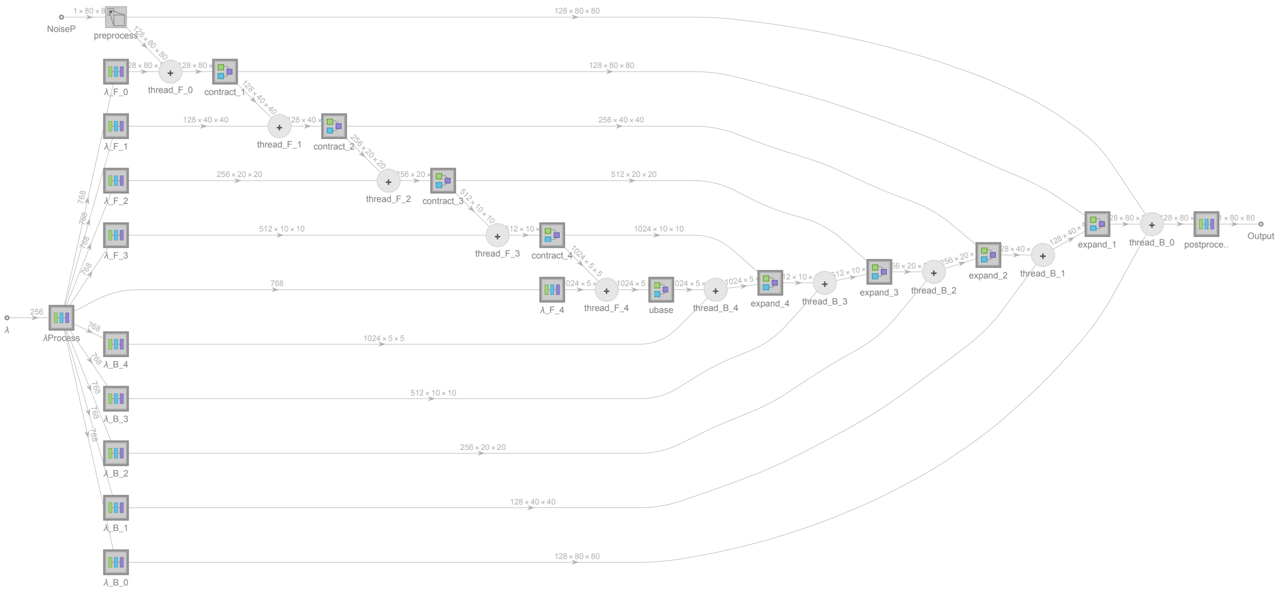

Figure S1. Model architecture of the unconditional score estimating neural network. We embed the time information, and stack the time embedding as extra channel to all UNet blocks. Each contract block consists of a long chain of $\{\mathrm{C}_{3\times3} + \mathrm{N} + \mathrm{ReLU}\}_3$, and a short chain of $\{\mathrm{C}_{1\times1}\}_1$, concatenated as a residual block. $\mathrm{C}_{n\times n}$ is convolution layer, with kernel receptive field of size $n \times n$, N is group normalization. Each expand block consists of a long chain of $\{\mathrm{R}_2 + \mathrm{C}_{3\times3} + \mathrm{N} + \mathrm{ReLU}\}_3$, and a short chain of $\{\mathrm{R}_2, \mathrm{C}_{1\times1}\}_1$, concatenating as a residual block, R_n resize the data by n times using linear interpolation. We start with channel size of 128, and double/shrink the channel size by 2 along each contract/expand block.

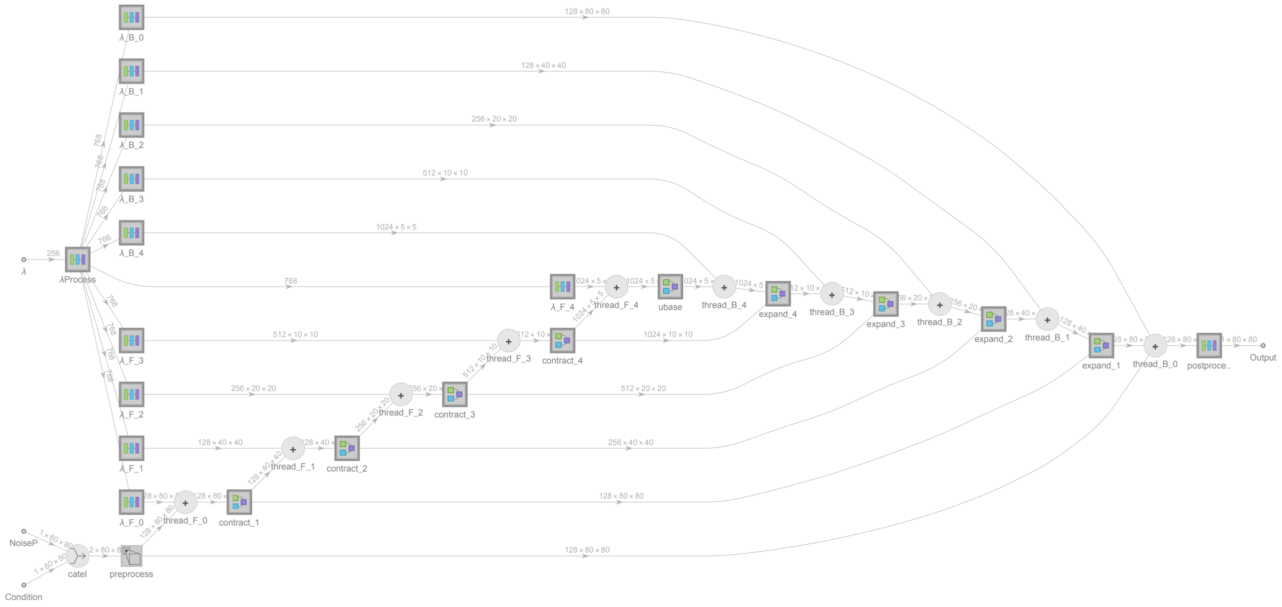

Figure S2. Model architecture of the conditional score estimating neural network, similar to the unconditional score estimating neural network, but includes the conditioning information from a UNet precipitation estimation using dynamical field as input.

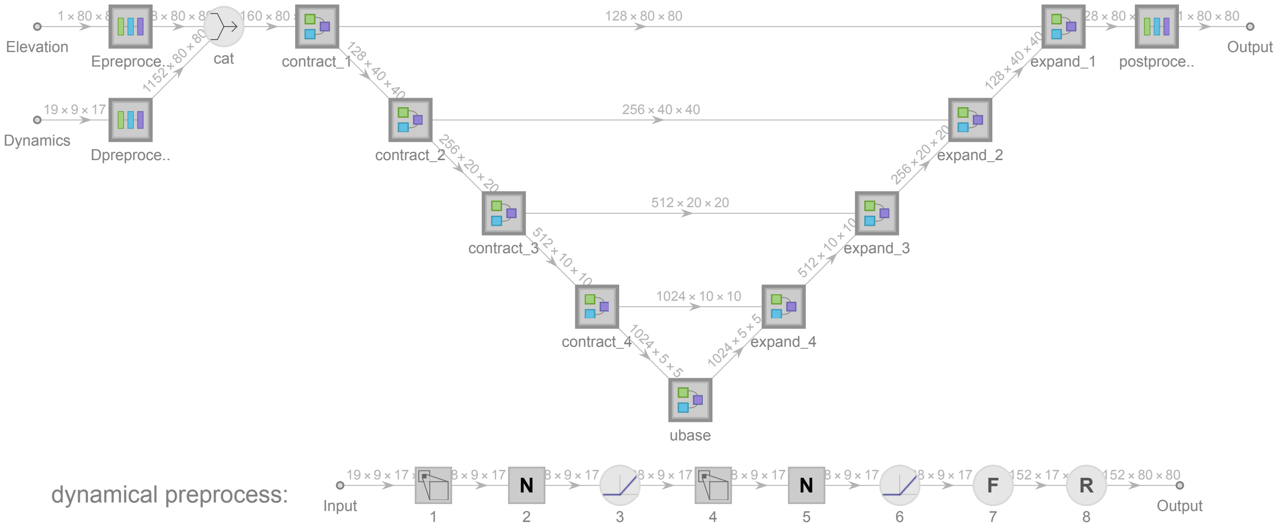

Figure S3. UNet architecture. The model takes into input of resolved dynamical field information and static elevation information, and outputs a deterministic precipitation field estimate. The dynamical field information is provided by a 9-hour (including 3 previous/current/future hours), 8° × 8° circulation field data, with 19 channels representing 19 dynamical variables. This dynamical field information is first pre-processed through 3D convolution blocks (bottom), and concatenated with preprocessed elevation information, before feeding into a 2D UNet. The UNet applies a convolution based contracting path to capture precipitation relevant dynamical field information, and a symmetric transposed convolution based expanding path to gradually refine precipitation field estimates. Skip connections between symmetrical convolution and transposed convolution blocks are applied to force deeper neural network layers to learn meaningful representations that are not well captured by shallower layers.

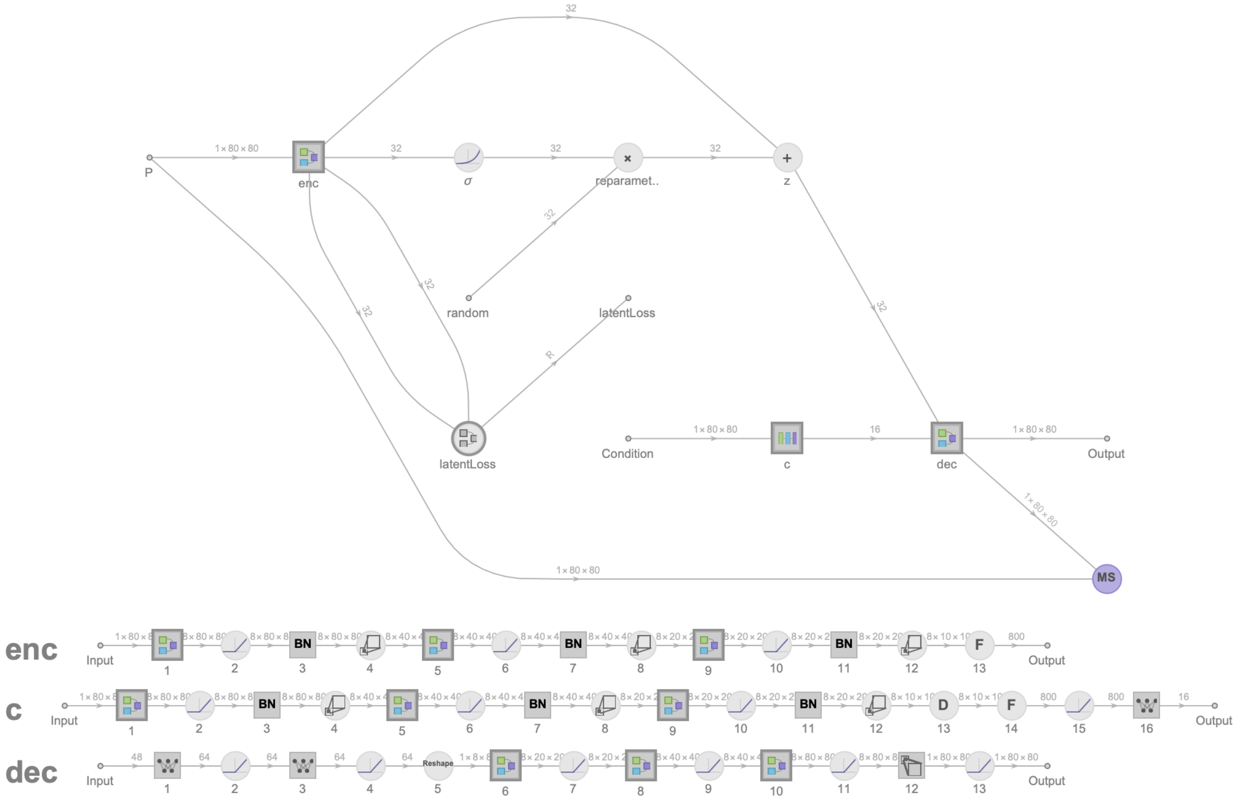

Figure S4. CVAE architecture. The encoder approximates $p(\mathbf{z}|\mathbf{x},\mathbf{y})$ as a variational Gaussian distribution; the decoder approximates $p(\mathbf{y}|\mathbf{x},\mathbf{z})$ using the conditioning information $\mathbf{x}$ and the learned latent vector $\mathbf{z}$. The dimension of $\mathbf{z}$ is 32.

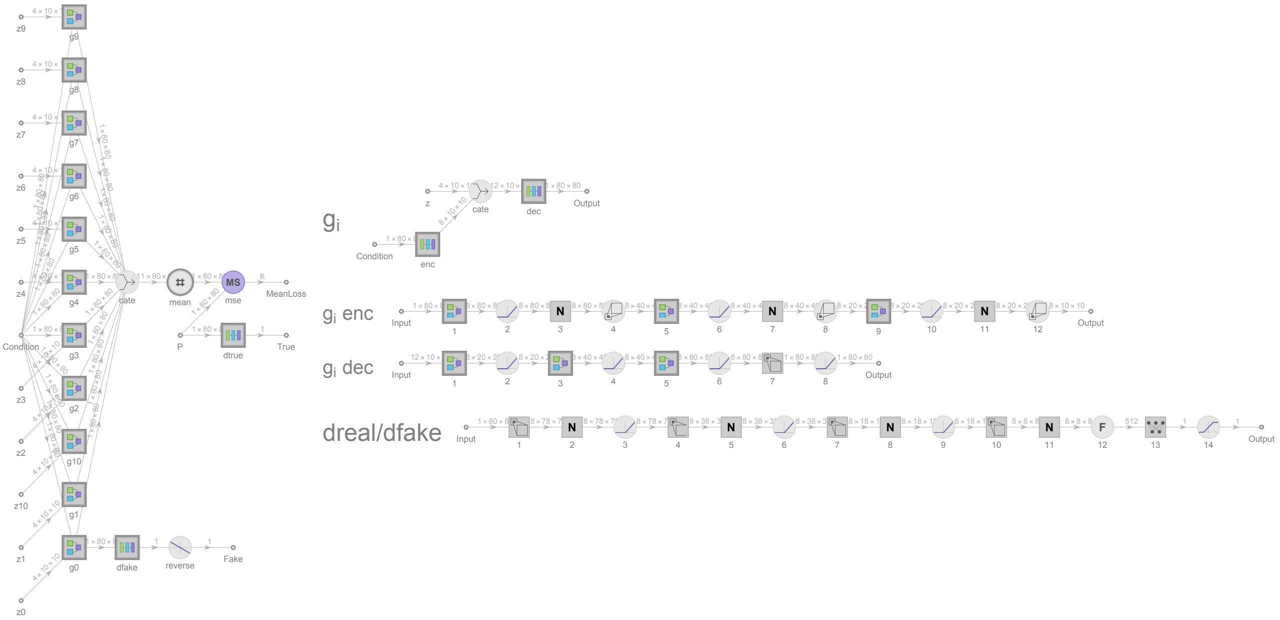

Figure S5. CGAN architecture. We replicate the generator network G for 10 times, each receiving a different $\mathbf{z}$ and a same conditioning information $\mathbf{x}$ to create an ensemble member. The generator network G is trained to fool the discriminator network D, while drawing the ensemble mean close to the realized observation. The discriminator network D is a binary classifier, optimized to differentiate between generated samples and true samples.

Physical process	Option
Cloud microphysics	Lin (Purdue)
Cumulus	Zhang and McFarlane
Radiation	Rapid Radiative Transfer Model
Boundary layer	Yonsei University (YSU) PBL scheme
Surface	Noah Land Surface Mode

Table S1. Physics options for WRF simulation.

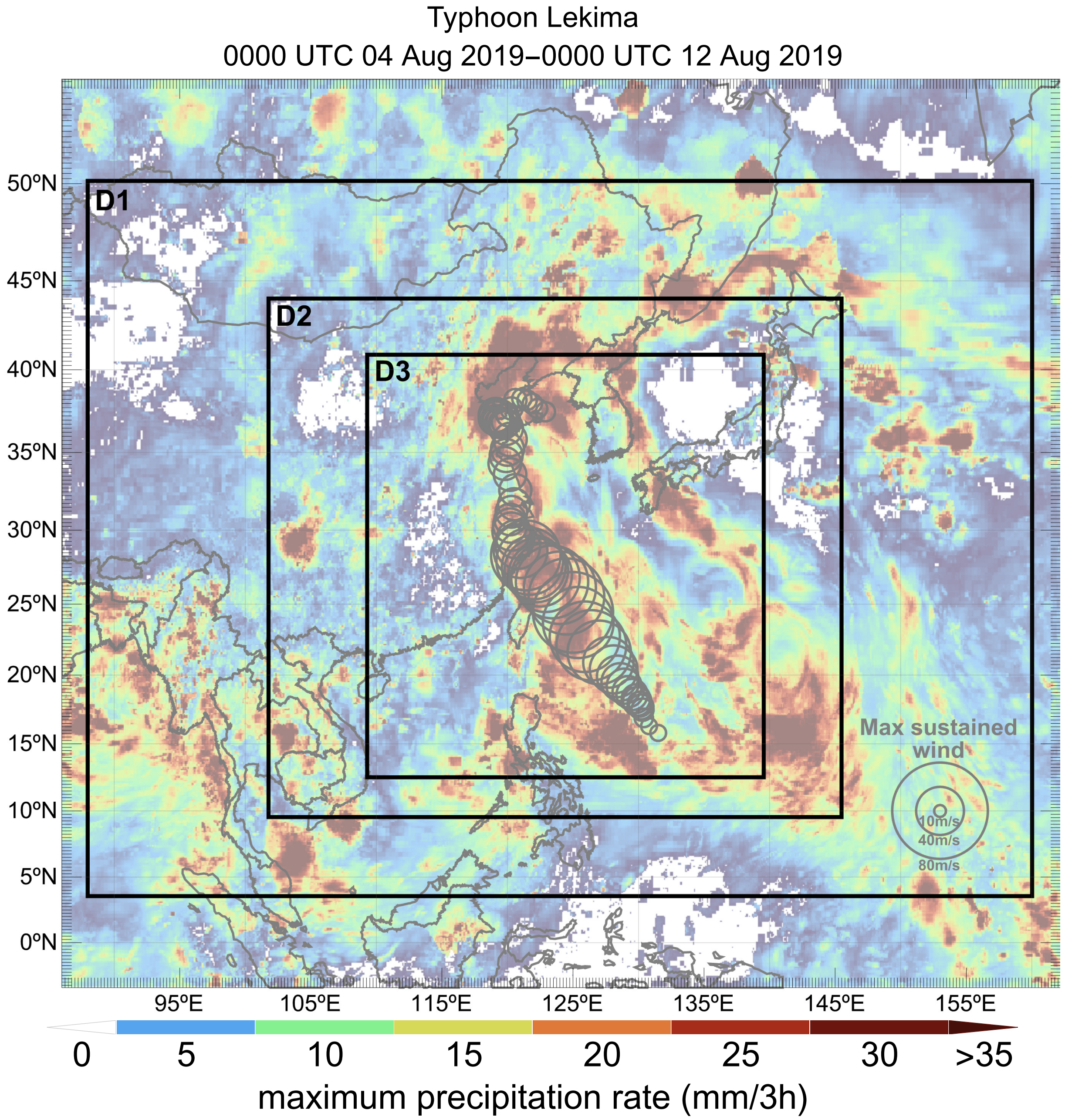

Figure S6. WRF nested Domains (27km/9km/3km) for Typhoon Lekima simulation, from 0000 UTC 04 August 2019 to 0000 UTC 12 August 2019. Color denotes maximum precipitation rate (mm/3h) through the simulation period.