4D-Var data assimilation using an adjoint model of a neural network surrogate model

Seiya Nishizawa

RIKEN Center for Computational Science, Kobe, Japan

March 4, 2022

Corresponding author: Seiya Nishizawa, RIKEN Center for Computational Science, 7-1-26 Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo 650-0047, Japan. E-mail: s-nishizawa@riken.jp

Abstract

Four-dimensional variational (4D-Var) data assimilation is an effective method for obtaining physically consistent time-varying states. In this study, a method using a neural network surrogate model obtained by machine learning is proposed to solve one of the most serious challenges in 4D-Var: to construct an adjoint model. The feasibility of the proposed method was demonstrated by a 4D-Var experiment using a surrogate model for the Lorenz 96 model. In the method, several effective procedures have been proposed to obtain an accurate surrogate model and the assimilated initial conditions, including two-stage learning (i.e., single- and multi-step learning) of neural networks, limiting the target states of the surrogate model to a small subspace of the state phase space, and updating the surrogate model during 4D-Var iterations.

Keywords Data assimilation; surrogate model; machine learning.

1. Introduction

Optimal initial conditions are crucial for accurate deterministic numerical simulations. Data assimilation is widely used to obtain initial conditions, e.g., it is an essential component of numerical weather forecasting systems. Four-dimensional variational (4D-Var) data assimilation has the advantage of obtaining a time evolution consistent with model physics. This is important, especially when obtaining four-dimensional analysis data of target phenomena to determine its mechanism. The 4D-Var method requires an adjoint model of the simulation model for the backward calculation of a cost function's gradient with respect to the initial conditions. Building the adjoint model and updating it with the simulation model is costly, which is an important challenge of the 4D-Var method. Despite this disadvantage, the method has been employed in several operational numerical weather forecasting systems. Research-purpose simulation models tend to have a shorter lifetime than operational models, and usually have multiple simulation paths (several different schemes) for individual physical processes, from which users can choose according to their objectives. Therefore, developing and managing adjoint models of research models may require more effort than operational models.

Machine learning techniques have developed rapidly and are used in an increasing range of domains. Data assimilation and machine learning have some similarities (Geer 2021); both minimize error (the cost or loss function) by optimizing target quantities, such as the state vector (data assimilation) and network parameters (machine learning). In neural network training, network parameters are updated according to their loss function's gradient. To obtain the gradient, a backward propagation algorithm is generally used. Recently, excellent machine learning frameworks, such as Pytorch (Paszke et al. 2019) and TensorFlow (Abadi et al. 2015), have been developed, which can easily compute gradients. Note that the procedure for learning network parameters is the same as that for updating the initial conditions with the adjoint model in 4D-Var. Therefore, once the forward simulation model is constructed, it is not necessary to manually build its adjoint model; the backward calculation of the gradient with respect to the initial conditions can be performed using the functionality of the framework. However, physics-based simulation models built using the framework generally require more computational resources, such as CPU time and memory usage, than conventional models written in C or Fortran, which may not be practical.

Replicating physics-based simulations with a neural network surrogate model is a possible solution. Surrogate models are not based on physical laws (e.g., governing equations), but on statistical relationships between the initial conditions and simulation results (Grzeszczuk et al. 1998; Dueben and Bauer 2018). Surrogate models are built by machine learning on inputs and outputs of physics-based simulations and can be designed to be computationally less expensive. The functionality of machine learning framework can be used to calculate the cost function's gradient of the neural network surrogate model with respect to the initial conditions. Even without the functionality, building a adjoint model of the neural network model manually is much easier than building a physics-based adjoint model because neural networks generally consist of a limited number of simple operations, such as weighted sums, and only a few nonlinear activation functions. Using a surrogate model's adjoint model may make 4D-Var data assimilation easier.

There are two major concerns with using surrogate models in 4D-Var data assimilation. (1) Can a surrogate model be obtained that provides sufficiently accurate simulation results? For systems with many degrees of freedom (e.g., atmospheric system), surrogate models must also have sufficient degrees of freedom (Dueben and Bauer 2018). The greater the degrees of freedom, the more difficult it is to build a surrogate model. Limiting the target space of the surrogate model in the phase space to a small subspace around the target state, building a surrogate model is expected to be more easier than when targeting the entire space. (2) Can the gradients be accurate enough to improve the initial conditions? Even if a surrogate model providing accurate forward computations can be obtained, its Jacobian may not be accurate (Chevallier and Mahfouf 2001; Aires et al. 2004). For example, if the resulting network overfits the training data, the gradients may be unrealistic, even if the results of the forward simulation appear reasonable.

Several studies have proposed a similar concept using machine learning for data assimilation. Brajard et al. (2020) combined data assimilation and machine learning without a physics-based model; the amount of training data was capped because they were limited to the observation data. This limitation can make program overfitting more serious. Hatfield et al. (2021) used an adjoint model obtained by machine learning for 4D-Var data assimilation. Training data were generated using simulations with a physics-based model and there was no limit to the amount of training data in principle. They demonstrated 4D-Var by replacing a parameterization scheme in the general circulation model with a neural network. The scheme was only a part of the model, and most parts of the adjoint model were derived manually, as in the conventional method. Nonnenmacher and Greenberg (2021) replaced a whole physics-based model with a neural network model, which output tendencies of the prognostic variables and was trained with the output tendencies of the physics-based model. For time integration, a conventional method, such as the Runge-Kutta method, was used. This scheme has advantages; tendencies, which can help understand the mechanisms of phenomena, and states after the arbitrary integration period can be obtained. However, even if the tendencies only have a tiny error, the error may accumulate (grow) during time integration, since the network is trained by instantaneous data. Alternatively, the neural network model can be designed to output a state after a certain time integration period. In this case, the model could be trained so that the error grown in a finite time integration period is reduced.

This study investigates the feasibility of using a neural network surrogate model to improve the initial conditions in a 4D-Var data assimilation, where the surrogate model is trained by simulation results from a physics-based model and outputs a state after certain time period. A simple dynamical system was used to study the feasibility. Several physics-based simulations were performed, then a neural network surrogate model was built using the simulations' output data. Efficient ways to train the network were also investigated. Using this surrogate model, a 4D-Var data assimilation experiment was conducted, and a promising method was proposed to efficiently improve the initial conditions during the assimilation iteration.

2. Model and Methodologies

2.1 Lorenz 96 model

The Lorenz 96 model is a dynamical system model (Lorenz 1996):

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, i = 1, 2, \cdots I, \tag{1}$$

where I is the number of grid points. The first, second, and last terms on the right-hand side correspond to the advection, diffusion, and forcing terms, respectively. The system exhibits chaotic behavior for a range of F values (Lorenz and Emanuel 1998).

Physics-based simulations were performed using the fourth-order Runge-Kutta scheme with a time step of $\Delta t = 0.01$. I and F were set to 40 and 8.0, respectively, at which the system was chaotic. Periodic boundary conditions were employed, and the initial conditions were $x_i = F + \epsilon_i$, where ϵ is a small random normally distributed perturbation with a standard deviation of 0.01. After a spin-up of 5,100 integration time steps, time integration of 100 steps from $t = 0$ to 1 was performed (hereafter referred to as the reference). Then, the ensembles were generated by adding random normally distributed perturbations, with a standard deviation of 0.1, to the reference state after the first spin-up 5,000 steps. After 100 integration steps for the second spin-up, 100 time-integration steps were performed for each ensemble.

The ensemble average of the mean squared error (MSE) from the reference grows exponentially, with a growth rate (Lyapunov exponent) of approximately 2.14. The states were output every five steps ($\Delta_{\text{output}} = 0.05$), and there were 21 outputs of 40-dimensional vectors $\boldsymbol{x}$, including the initial state, for each ensemble. These were used to train the neural network surrogate model.

The states of these ensembles lie within a limited region (a subspace) around the reference in the state phase space (hereafter referred to as the localized ensemble set). To examine the effect of extent of the training data's state in the phase space on the surrogate model trained from the data, another ensemble set was generated with a second spin-up of 1,000 steps (hereafter referred to as the spread ensemble set); the second spin-up was 100 steps for the localized ensemble set. The longer spin-up resulted in a wider spread; the states of the spread ensemble set are widely spread in the phase space with a large variance that is comparable in magnitude to the variance of a very long time series. The MSE of the spread ensemble set is approximately 24–29 throughout the integration period, whereas the MSE of the localized ensemble set is approximately 0.27 and 2.30 at the beginning and end of the integration period, respectively.

2.2 Surrogate model

Using the state vectors $\boldsymbol{x}$ of the physics-based simulation as both input and target data for training, a neural network surrogate model replicating the physics-based simulation was built. The calculations in the neural network were conducted using Pytorch.

a. Network architecture

In the physics-based model, the state at the next time step depends only on that of the previous step. To emulate this behavior, the network was designed as a recurrent neural network; an identical network module is connected recurrently, and each module corresponds to a time interval of 0.05 (Figure 1a). Each module consists of a stacked hourglass network (Newell et al. 2016) of two stages: down-sampling and up-sampling (Figure 1b). Through these stages, multiple horizontal scales are considered. In down-sampling, the grid size is halved at each step by the max-pooling layer; there are four steps and the grid size at each step is 40, 20, 10, or 5. In up-sampling, the grid size is doubled at each step, from 5 to 40, by the max-unpooling layer. The output of each down-sampling step is added to the input of the corresponding up-sampling step via skip connections. Each step consists of convolution layers, batch normalization layers, and rectified linear unit activation layers. In the convolution layers, the values of neighboring grid points interact, and the convolution and following activation layers are expected to replicate advection. The convolution kernel size for the hourglass network is three, and periodic boundary padding is applied before the convolution. Before the hourglass network, the number of channels is increased from one to four by convolution with a kernel size of one, and after the network, the number of channels is reduced from four to one.

Fig. 1

b. Training and evaluation

Training the neural network was divided into two stages: single-step and multi-step learning. In single-step learning, a non-recurrent single network module was trained. The training data input were $\boldsymbol{x}(t)$ and the target data were $\boldsymbol{x}(t+0.05)$, where $t = 0, 0.05, \cdots, 0.95$. The loss function l_1 is defined as

$$l_1 = \frac{1}{I} \left| f(\boldsymbol{x}(t)) - \boldsymbol{x}(t+0.05) \right|^2, \tag{2}$$

where f is the operator corresponding to the single network module. With the 21-step output dataset obtained in each ensemble run, 20 training input-output pairs were available; from M ensemble runs, a training dataset of $20M$ pairs could be used.

In the multi-step learning process, the input data were $\boldsymbol{x}(t)$ and the target data were $(\boldsymbol{x}(t+0.05), \boldsymbol{x}(t+0.1), \cdots, \boldsymbol{x}(t+0.5))$, where $t = 0, 0.05, \cdots, 0.5$.

Each ensemble run contained 11 training data pairs, and a total of $11M$ training data pairs could be used from M ensemble runs. The loss function l_{10} is defined as

$$l_{10} = \frac{1}{10I} \sum_{n=1}^{10} \left| f^n(\boldsymbol{x}(t)) - \boldsymbol{x}(t + 0.05n) \right|^2 . \tag{3}$$

The network parameters obtained by single-step learning were used as the initial parameters for multi-step learning. In the multi-step learning process, the dropout layers were disabled, and the mean and standard deviations in the batch normalization layers were fixed to the values obtained in single-step learning.

For both the single- and multi-step learning processes, the error of the trained network was evaluated by the ensemble average of their loss functions, calculated using the evaluation data of another ensemble dataset of 100 runs, which was generated in the same manner as the training datasets, with the same number of spin-up steps. The batch size was swept and determined such that the error of the network was minimized. The size of the training dataset, which is proportional to the ensemble size, was also varied for sensitivity testing. The Adam optimization algorithm (Kingma and Ba 2014) was used to update the network parameters. The initial learning rates were set to 10^{-5} and 10^{-6} times batch size for single- and multi-step learning, respectively, with a decay every 1,000 epochs by a factor of 0.99.

2.3 4D-Var data assimilation

In the 4D-Var data assimilation experiment, the neural network surrogate model and its adjoint model were used for the forward and backward computations, respectively. The time window for assimilation was 0.5, and the observed data were assimilated at intervals of 0.05, for 10 time steps. Since the non-dimensional time unit in this system is roughly equivalent to 5 days (Lorenz 1996), the time window corresponds to 2–3 days. The cost function J_s was defined as follows:

$$\begin{aligned} J_s(\boldsymbol{x}(0)) &= \frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{x}_b\right)^T \mathrm{B}^{-1}\left(\boldsymbol{x}-\boldsymbol{x}_b\right) \\ &+ \frac{1}{2}\sum_{n=1}^{10}\left[H(f^n(\boldsymbol{x}(0)))-\boldsymbol{y}(0.05n)\right]^T \mathrm{R}^{-1}\left[H(f^n(\boldsymbol{x}(0)))-\boldsymbol{y}(0.05n)\right], \end{aligned} \tag{4}$$

where $\boldsymbol{x}_b$ is the first guess for $\boldsymbol{x}(0)$, $\boldsymbol{y}(t)$ is the observed state at time t, B is the background covariance matrix, R is the observation covariance matrix, and H is the observation operator. The first guess was the initial state of one of the ensemble runs in Section 2.1. B was calculated from the 1,000 ensembles, and the mean of the diagonal components was 0.27. Observations were generated to have a normally distributed error with a standard deviation of 0.1; the matrix R is diagonal and its diagonal components were 0.01. The observed data were located at 10 random grids, chosen from the 40 grids: $i = 3, 12, 17, 20, 25, 26, 27, 31, 32,$ and 33. The operator H reveals the data at these locations.

The gradient of J_s for $\boldsymbol{x}(0)$ was obtained using the surrogate model's adjoint model; the gradient was calculated automatically by Pytorch. The gradient was then used to update $\boldsymbol{x}(0)$. A one-dimensional golden-section search procedure (Kiefer 1953) was used for the update. $\boldsymbol{x}(0)$ was updated iteratively to reduce the cost function, and the maximum iteration count was 1,000. At each of the K iterations (hereafter referred to as the update interval), the physics-based simulation was performed using the latest $\boldsymbol{x}(0)$, and the network parameters of the surrogate model were updated using the simulation results in the same way as multi-step learning.

To evaluate the assimilated initial state, an extended forecast simulation was conducted from the initial state using the physics-based model and the root mean squared error (RMSE) of the forecast state from the reference state at $t = 1$ was calculated. The learning rate in the surrogate model update swept between 10^{-5}, 3×10^{-5}, and 10^{-4}, and was determined such that the RMSE improvement was maximized.

The 4D-Var experiment was conducted with ten different first guesses for each parameter. In the analysis, the RMSE was averaged over the ten results.

3. Results

3.1 Obtaining the surrogate model

First, the neural network was trained using single-step learning. The error of the network depends on the size of the training data; the larger the size of the dataset, the smaller the error is (Figure 2a). The error is O(10^{-4})–O(10^{-2}), which is much smaller than the O(10) background variance of $\boldsymbol{x}$. The batch sizes with the smallest errors were 125, 250, 2000, 4000, and 4000 for ensemble sizes of 50, 100, 200, 400, and 800, respectively. Note that the ensemble size is proportional to the training dataset size. Then, using the surrogate model, a time integration experiment for $t = 0$–1 was conducted, i.e., the network obtained above was repeated 20 times. This time integration was calculated from 1,000 different generated initial states, as in Section 2.1. The accuracy of the surrogate model was evaluated by MSE from the physics-based model solution from the same initial states. Note that the MSE is only due to model error, since the initial conditions have no error. Figure 2c shows the temporal evolution of the ensemble average of the surrogate model's MSEs obtained from an ensemble set of $M = 800$. The MSE grows over time, with the growth rate initially gradually decreasing and then remaining nearly constant. Even later, the growth rate is still larger than the growth rate of the physical growth mode in Section 2.1.

Fig. 2

Next, multi-step learning was performed. Figure 2b shows the multi-step learning network error. Because the discrepancy between the surrogate model and the physics-based simulations tends to increase with time, the magnitude of the network error is larger than that in single-step learning. The error depends on the ensemble size, as in the case of single-step learning, but the dependency on batch size is smaller than in single-step learning. Using the multi-step learning surrogate model, time integration was performed as for single-step learning. The early growth rate of the error was improved compared to that of the single-step learning model (Figure 2c). As a result, the error at the end of the timespan is approximately 60% of that obtained by the single-step learning model. Conversely, the error after the first step ($t = 0.05$) is larger than that of the single-step learning model. This can be explained as follows. In single-step learning, the network learns such that the error after single-step integration is small, whereas in the multi-step learning process, the network learns such that the average error of 10 steps is small. This means that unstable modes with large Jacobian eigenvalues become smaller in single-step learning, and unstable modes with large singular values become smaller in the multi-step learning process. The fastest growing mode in terms of instantaneous temporal difference is represented by the eigenvector and the mode over a finite-time interval is represented by the singular vector. This is consistent with the expectation of Brenowitz and Bretherton (2018), that a multiple-time-step loss function penalizes a growing unstable mode.

To evaluate the efficacy of the two-stage learning process (single- and multi-step learning), one-stage learning (multi-step learning only) was also conducted. Randomly generated initial network parameters were used for multi-step learning. In this case, the network did not learn well; the loss function did not decrease significantly during the epoch iteration and saturated at the level of O(1). As a result, the error of the surrogate model obtained by one-stage learning was much larger than that obtained by two-stage learning. This is due to the total number of layers in the network being too deep. This may be solved by a more appropriate network design. Regardless of the case, the network was successfully trained by two-stage learning. This suggests that two-stage learning is an efficient way to build surrogate models.

To investigate the effect on the accuracy of the surrogate model by limiting the training data's state to a subspace of the phase space, the same training was performed using the spread ensemble set generated with the longer second spin-up of 1,000 steps, instead of the localized ensemble set of the 100-step spin-up. The errors of the networks obtained using the spread ensemble set were 0.40, 0.20, 0.088, 0.044, and 0.018 for ensemble sizes of $M = 50$, 100, 200, 400, and 800, respectively. These errors were approximately 6.5 to 10 times larger than the corresponding errors with the localized ensemble set (0.045, 0.024, 0.010, 0.0041, and 0.0028, respectively). This shows that limiting the state of the target space is effective in increasing the model's accuracy. A surrogate model targeting wider space may require larger training data size and/or more complex network architecture. This suggests that the difficulty of building a surrogate model can be reduced by limiting the target states to a small phase subspace.

3.2 4D-Var experiment

A 4D-Var data assimilation experiment was conducted using the neural network surrogate model. Figures 3a and 3b show the evolution of the cost function J_s with the number of iterations. As the number of iterations increases, the cost generally decreases. The same cost function using the physics-based model was also calculated (J_p). J_s and J_p were calculated from the time series integrated from the same initial conditions. J_p is due to the errors in the initial state and observation, while J_s is due to not only these errors, but also the model error. J_p is generally larger than J_s, since the initial conditions have been updated so that J_s decreases. Nevertheless, we see that J_p decreases with an increasing number of iterations. This indicates that 4D-Var data assimilation using an adjoint model of a neural network surrogate model is effective. The cost is smaller for a larger ensemble size, which corresponds to a smaller error of the surrogate model, suggesting that the accuracy of the surrogate model affects the accuracy of the assimilation. Additionally, updating the surrogate model during 4D-Var iterations improved the cost. We observed large improvements in the cost when the network was updated, e.g., at 100 iterations for the case of $M = 200$ and $K = 100$. We also observed that the smaller the update interval, the smaller the cost.

Fig. 3

To evaluate the accuracy of the assimilated initial conditions, the extended forecasts' RMSE at $t = 1$ from the assimilated initial states after 1,000 iterations was examined. Figure 4a shows the RMSE averaged over 10 samples; it was approximately 0.42 for large ensemble-size cases, and 1.59 for cases from the first guess. This indicates that assimilation using the surrogate model improved the accuracy of the initial state. The dependency of the RMSE on the ensemble size and update interval shows similar characteristics to those of the cost; the RMSE is likely to be small for large ensemble sizes and smaller update intervals. As a reference, a 4D-Var experiment with a manually constructed adjoint model of the physics-based model was also conducted. The forecast's RMSE from the assimilated initial state was 0.39. This shows that assimilation using the surrogate model can achieve similar accuracy to that using conventionally manually obtained adjoint model of the physics-based model.

Fig. 4

4. Conclusions

A 4D-Var assimilation method was proposed using an adjoint model of a neural network surrogate model. Additionally, several procedures were proposed to efficiently obtain an accurate surrogate model and assimilated initial conditions. As a feasibility study, a surrogate model was constructed and a 4D-Var assimilation experiment was conducted using the Lorenz 96 model.

Two-stage learning was efficient for obtaining an accurate surrogate model. In the first stage, the network was trained from a training dataset, with target data one step forward from the input data, obtained using the physics-based model (single-step learning). In the next stage, the network was trained using time-series data of multiple steps as the target data (multi-step learning). In this stage, the network parameter obtained in the first stage was used as the initial value. It is found that the neural network model trained by time-sequence data with a longer time period has better accuracy than that with shorter period. The neural network model which output tendencies, as proposed by Nonnenmacher and Greenberg (2021), is thought to have the same problem with models trained by shorter period data. We found that limiting the target states of the surrogate model to a state phase subspace around the target case was efficient for building an accurate surrogate model.

The 4D-Var assimilation experiment showed that the initial conditions were improved by assimilation by using the adjoint model of the surrogate model. It was also found that updating the surrogate model during the 4D-Var iterations was effective in improving the accuracy of the initial conditions. Even if the accuracy of the initial surrogate model is not very high (e.g., $M = 50$), accurate initial conditions could be obtained with frequent updates (small K) of the surrogate model during 4D-Var iterations. In general, more accurate data contribute to better training in machine learning. Therefore, learning during 4D-Var iterations is likely to be more efficient than the earlier two-stage learning because the training data used for updating the network are more accurate due to better initial conditions. On the other hand, frequent updates require large computational resources; updating the network requires physics-based simulation and training with the simulation data. The optimal values of the ensemble size and update interval must be determined by balancing the computational costs for each stage of training and assimilation.

Assimilation has an affinity for limiting states in the phase space to build a surrogate model. The states in a finite assimilation window generally occupy only a small subspace and, therefore, a surrogate model that covers all possible states is not needed; we can focus on the subspace around the target state to be assimilated. However, the surrogate model needs to be rebuilt for different cases. The learning speed of the network in other cases can be significantly improved by using the network obtained in one case as the initial parameters, i.e., transfer learning.

As simulation models become more sophisticated, they become more complex, which requires more effort from researchers. The effective use of data science techniques will become increasingly important for various aspects of simulation research.

Data Availability Statements

The source code of the programs used in this study are available from the Zenodo repository at `https://doi.org/10.5281/zenodo.6319009`. The datasets generated by the experiment are available from the Zenodo repository at `https://doi.org/10.5281/zenodo.6318869`.

Acknowledgments

This work was supported by JSPS KAKENHI (Grant Number JP19H01974) and JST AIP (Grant Number JPMJCR19U2). Some results were obtained using the Wisteria/BDEC-01 system at the Information Technology Center, The University of Tokyo. The calculations associated with the neural network were conducted using Pytorch (`https://pytorch.org/`). The diagrams were drawn using tools developed by the GFD-Dennou Club (`https://www.gfd-dennou.org/index.html.en`).

References

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, 2015: TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Aires, F., C. Prigent, and W. B. Rossow, 2004: Neural network uncertainty assessment using Bayesian statistics with application to remote sensing: 3. Network Jacobians. *J. Geophys. Res.*, **109**, D10305.

Brajard, J., A. Carrassi, M. Bocquet, and L. Bertino, 2020: Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the Lorenz 96 model. *J. Comput. Sci.*, **44**, 101171.

Brenowitz, N. D., and C. S. Bretherton, 2018: Prognostic validation of a neural network unified physics parameterization. *Geophys. Res. Lett.*, **45**, 6289–6298.

Chevallier, F., and J.-F. Mahfouf, 2001: Evaluation of the Jacobians of infrared radiation models for variational data assimilation. *J. Appl. Meteor.*, **40**, 1445–1461.

Dueben, P. D., and P. Bauer, 2018: Challenges and design choices for global weather and climate models based on machine learning. *Geosci. Model Dev.*, **11**, 3999–4009.

Geer, A. J., 2021: Learning earth system models from observations: machine learning or data assimilation? *Phil. Trans. R. Soc. A.*, **379**, 20200089.

Grzeszczuk, R., D. Terzopoulos, and G. Hinton, 1998: Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, Association for Computing Machinery, New York, NY, USA, 9–20.

Hatfield, S., M. Chantry, P. Dueben, P. Lopez, A. Geer, and T. Palmer, 2021: Building tangent-linear and adjoint models for data assimilation with neural networks. *J. Adv. Model. Earth Syst.*, **13**, e2021MS002521.

Kiefer, J., 1953: Sequential minimax search for a maximum. *Proc. American Math. Soc.*, **4**, 502–506.

Kingma, D. P., and J. Ba, 2014: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Lorenz, E. N., 1996: Predictability – a problem partly solved. In *Seminar on predictability*, ECMWF, Reading, UK, 1–18.

Lorenz, E. N., and K. A. Emanuel, 1998: Optimal sites for supplementary weather observations: Simulation with a small model. *J. Atmos. Sci.*, **55**, 399–414.

Newell, A., K. Yang, and J. Deng, 2016: Stacked hourglass networks for human pose estimation. , *Computer vision – ECCV 2016*, Springer International Publishing, Cham, 483–499.

Nonnenmacher, M., and D. S. Greenberg, 2021: Deep emulators for differentiation, forecasting, and parametrization in Earth science simulators. *J. Adv. Model. Earth Syst.*, **13**, e2021MS002554.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, 2019: Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, Eds., Curran Associates, Inc., 8024–8035.

List of Figures

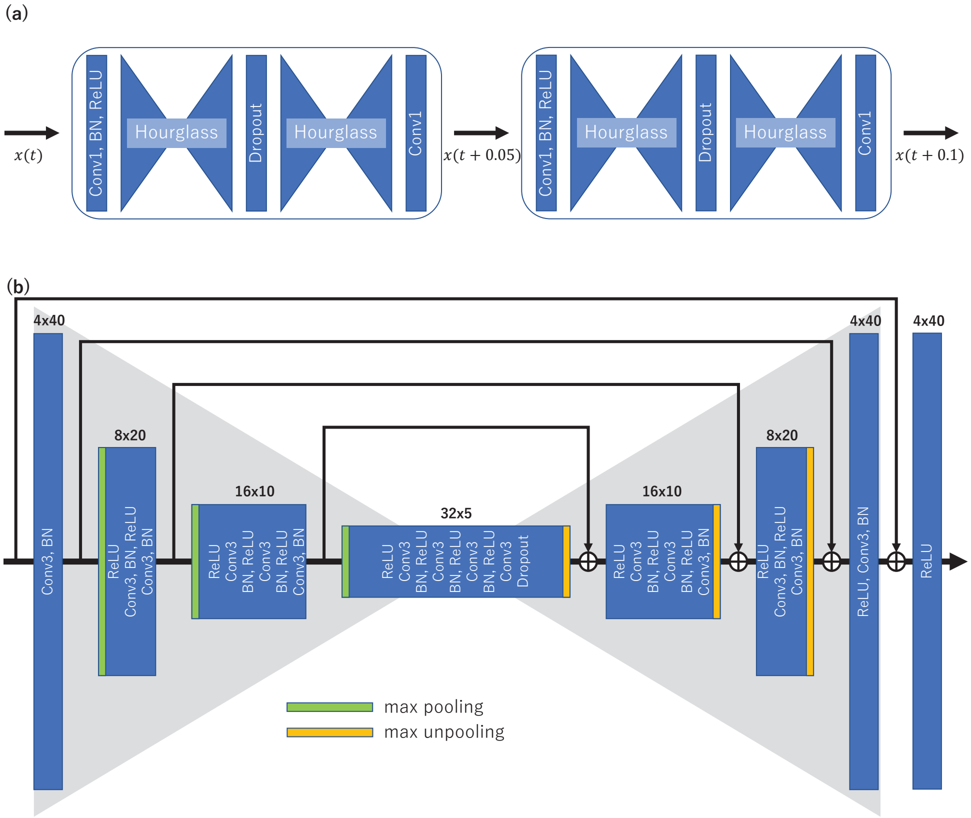

Fig. 1. Surrogate model network architecture: (a) recurrent network modules and (b) details of the hourglass network. “Conv,” “BN,” “ReLU,” and “Dropout” indicate convolution, batch normalization, rectified linear unit, and dropout layers, respectively. The number following the convolution indicates the kernel size. The number above each box in (b) is the channel size times the grid (neuron) size.

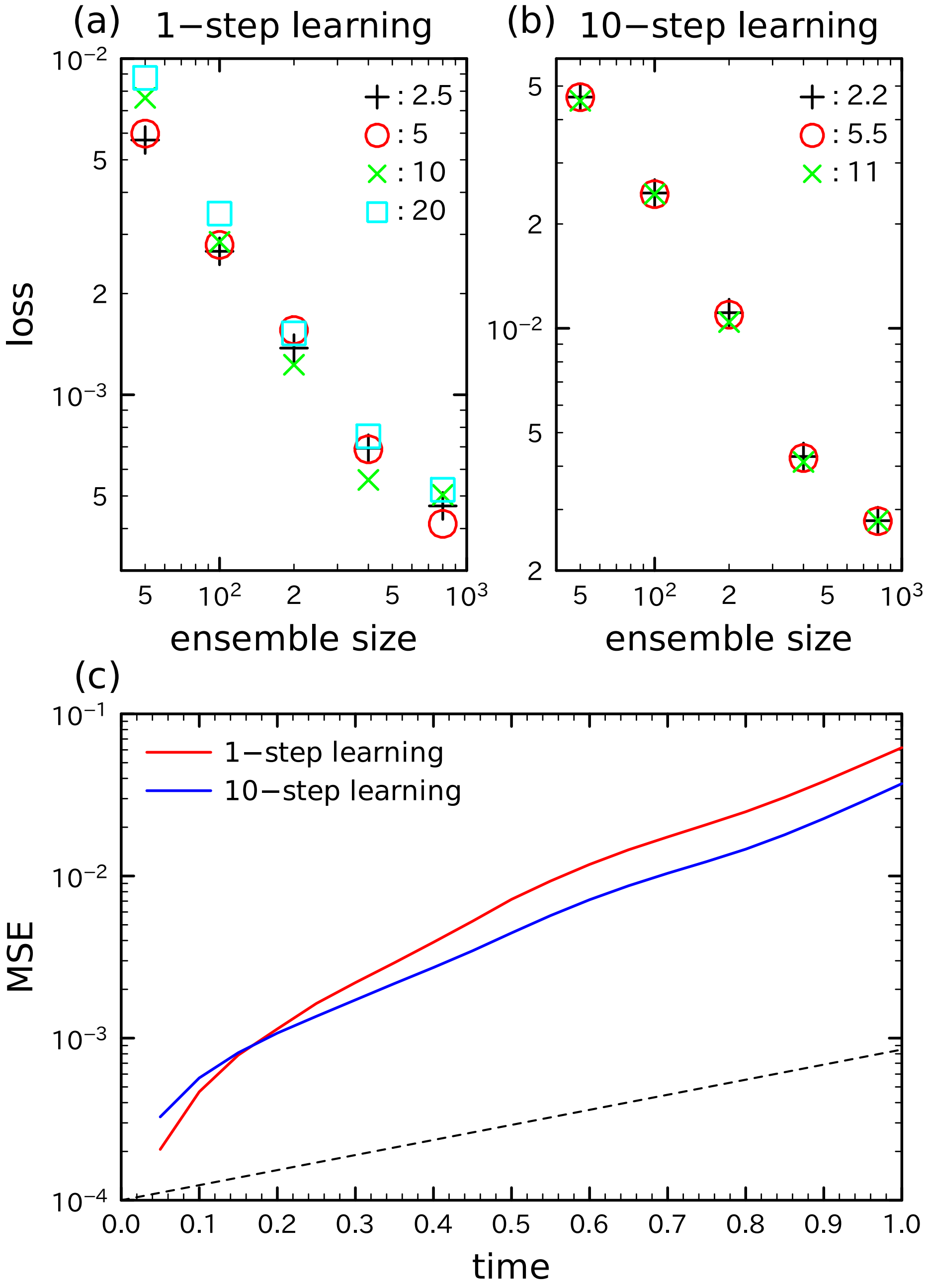

Fig. 2. The error of the network obtained by (a) the single- and (b) multi-step learning, and (c) the temporal evolution of the error of the surrogate model obtained with 800 ensembles by (red) single-step and (blue) multi-step learning. The symbols and colors in (a) and (b) represent the batch size normalized by the ensemble size. The broken line in (c) represents the error growth rate of the physical growth mode.

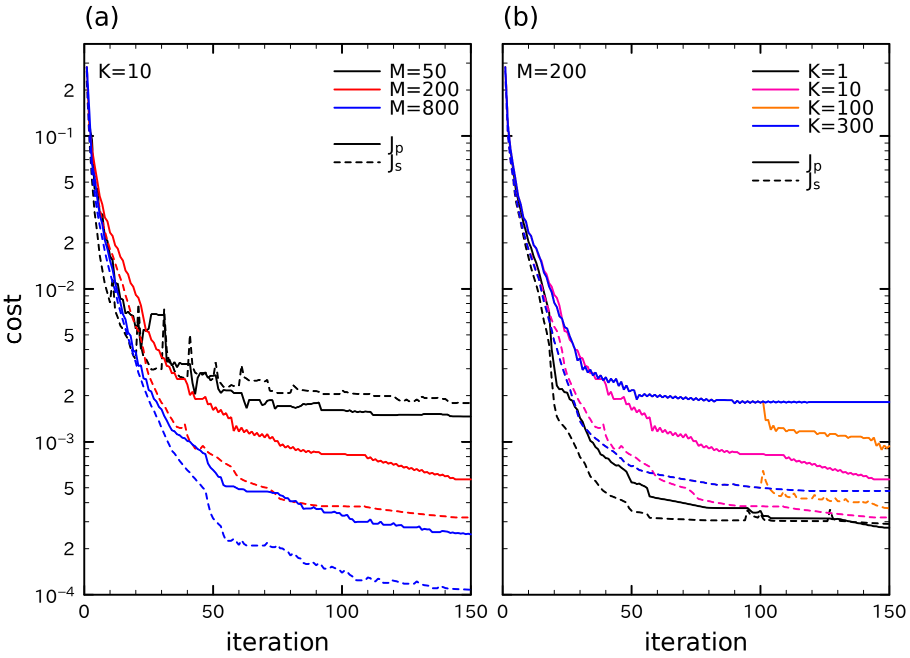

Fig. 3. The temporal evolution of the cost as a function of the iteration count in the 4D-Var assimilation with (a) the update interval $K = 10$ and (b) the ensemble size $M = 200$. The solid and broken lines represent J_p and J_s, respectively.

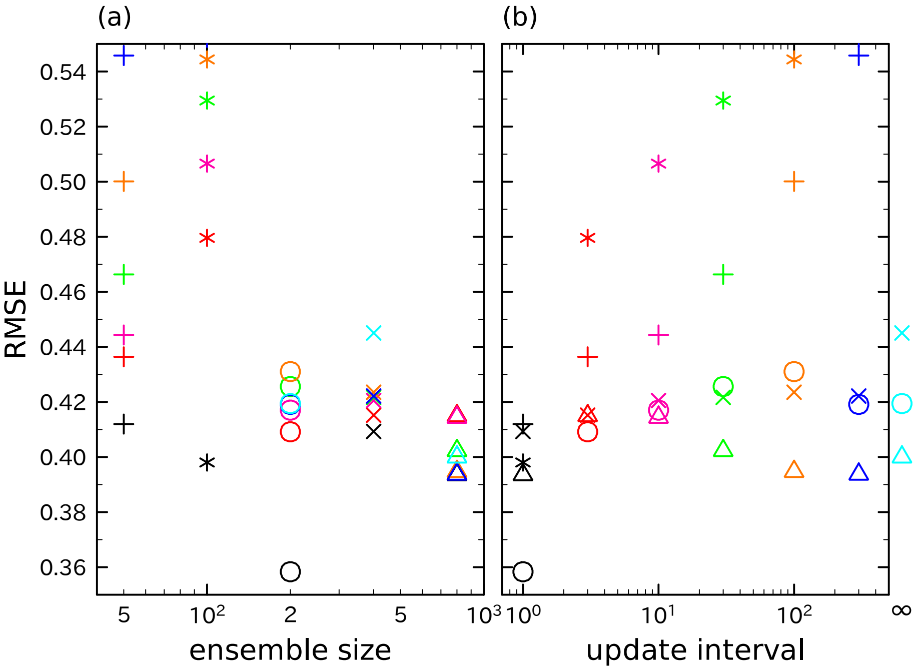

Fig. 4. RMSE dependency of the extended forecast at $t = 1$ on (a) the ensemble size and (b) the update interval. The symbols and colors represent the ensemble size and update interval, respectively.