Hybrid Neural Network Models for Postprocessing Medium-Range Forecasts of Tropical Cyclone Tracks over the Western North Pacific

HUNG MING CHEUNG,[a] CHANG-HOI HO,[a] AND MINHEE CHANG[a]

[a] *School of Earth and Environmental Sciences, Seoul National University, Seoul, South Korea*

(Manuscript received 12 November 2021, in final form 11 July 2022)

ABSTRACT: Tropical cyclone (TC) track forecasts derived from dynamical models inherit their errors. In this study, a neural network (NN) algorithm was proposed for postprocessing TC tracks predicted by the Global Ensemble Forecast System (GEFS) for lead times of 2, 4, 5, and 6 days over the western North Pacific. The hybrid NN is a combination of three NN classes: 1) convolutional NN that extracts spatial features from GEFS fields; 2) multilayer perceptron, which processes TC positions predicted by GEFS; and 3) recurrent NN that handles information from previous time steps. A dataset of 204 TCs (6744 samples), which were formed from 1985 to 2019 (June–October) and survived for at least six days, was separated into various track patterns. TCs in each track pattern were distributed uniformly to validation and test dataset, in which each contained 10% TCs of the entire dataset, and the remaining 80% were allocated to the training dataset. Two NN architectures were developed, with and without a shortcut connection. Feature selection and hyperparameter tuning were performed to improve model performance. The results present that mean track error and dispersion could be reduced, particularly with the shortcut connection, which also corrected the systematic speed and direction bias of GEFS. Although a reduction in mean track error was not achieved by the NNs for every forecast lead time, improvement can be foreseen upon calibration for reducing overfitting, and the performance encourages further development in the present application.

KEYWORDS: North Pacific Ocean; Tropical cyclones; Statistical forecasting; Neural networks; Postprocessing; Artificial intelligence

1. Introduction

Dynamical models have been used extensively for weather and climate predictions. They predict the forthcoming atmospheric states by numerically solving physical equations. Since the solutions are discrete in space and time, truncation errors are inevitable (Gerrity et al. 1972). Moreover, additional errors are produced as the processes at unresolved scales are approximated by parameterization (Jankov et al. 2005; Otkin and Greenwald 2008). Furthermore, inaccurate initial conditions will lead to deviation of the model simulation from the true future state (Lorenz 1963). Although dynamical models produce more accurate forecasts due to the constant improvement of computer resources and data assimilation methods, model output errors remain to be solved (Bauer et al. 2015; Grams et al. 2018; Rodwell et al. 2013).

Tropical cyclone (TC) track forecasts, which rely on dynamical models, inevitably inherit model errors. For example, the choice of cumulus parameterization and microphysics can significantly impact track forecasting. Bassill (2014) simulated Hurricane Sandy (2012) with two cumulus parameterizations—simplified Arakawa–Schubert and Tiedtke—and found that the former predicted that the TC would move toward the central North Atlantic, while landfall was predicted for the latter. Fovell et al. (2009) examined the impact of cloud microphysics on TC track simulations in idealized experiments. Three microphysics schemes [e.g., Kessler, Lin–Farley–Orville five-class, and the Weather Research and Forecasting (WRF) three-class single-moment] led to different distributions of virtual temperature, thus affecting TC motion. Conversely, Plu (2011) examined the predictability of TC tracks from 2006 to 2009 using global models from the European Centre for Medium-Range Weather Forecasts (ECMWF), Météo-France, and the Met Office (UKMO). He found that the length of time in which small track error doubles itself (i.e., doubling time), which is an estimate of the upper bound on predictability, was 30–50 h. Therefore, it can be deduced that TC track forecasts beyond five days that rely solely on the dynamic model are subjected to larger errors. Since more than one-third of the TCs formed in the western North Pacific (WNP) have lifetimes of at least six days (Cheung et al. 2021), improving TC track prediction in this region is important given the large East Asian population along the coastline (Lau et al. 2021; Mendelsohn et al. 2012; Park et al. 2016; Wang et al. 2019).

Artificial neural networks (ANN, or simply NN)—a subfield of both machine and deep learning—have been applied in weather and climate studies to reduce model errors, gaining popularity in recent years. Marzban (2003) used NNs to postprocess hourly surface temperature forecasts at 31 locations using the Advanced Regional Prediction System. Improvements were observed in the mean-squared error, bias, and variance. Rasp and Lerch (2018) developed an NN for postprocessing ECMWF ensemble forecasts of 2-m temperature over Germany. Their NN model, which included auxiliary

Supplemental information related to this paper is available at the Journals Online website: https://doi.org/10.1175/AIES-D-21-0003.s1.

Corresponding author: Chang-Hoi Ho, hoch@cpl.snu.ac.kr

DOI: 10.1175/AIES-D-21-0003.1 e210003

predictor variables and station-specific information as input, outperformed other benchmark methods. Ho et al. (2022) used a long short-term memory (LSTM) algorithm that included input from air pollutant concentrations predicted by the Community Multiscale Air Quality (CMAQ) model to forecast air pollutant concentrations in South Korea. Their LSTM model had an accuracy of 68%–77% for lead times of up to two days, which was higher than the CMAQ and comparable to forecaster-adjusted CMAQ predictions.

Using deep learning as a postprocessing tool for TC track forecasting is uncommon compared to weather forecasting and climate prediction. To the authors' knowledge, the only study of this kind has been done by Kim et al. (2020) where NNs were used to correct the TC position predicted by the WRF at lead times of 1, 2, and 3 days. Using the TC cases in 2015, reductions in track error of 10%–15% were obtained for forecasts at the three lead times when output selection was applied (an NN model was trained 10 times and unreasonable output among them were excluded). Conversely, NN is more commonly used for the direct prediction of future TC positions. Various architectures, such as convolutional NN (CNN; Giffard-Roisin et al. 2020), recurrent NN (RNN; Alemany et al. 2019; Chandra et al. 2015), and convolutional LSTM (Kim et al. 2018; Kim et al. 2019) have been implemented for TC track prediction within five days. In addition, application of NN for TC intensity prediction (Cloud et al. 2019; Hu et al. 2020; Xu et al. 2021) or estimation (Chen et al. 2019; Olander et al. 2021; Wimmers et al. 2019), genesis prediction (Zhang et al. 2019), and identification of TC center (Smith and Toumi 2021) or region of interest (Kumler-Bonfanti et al. 2020) are also becoming very popular.

As a proof of concept, this study aims to explore the possibility of developing and employing an NN algorithm for postprocessing medium-range TC track forecasts in the WNP. The three common classes of NN, namely multilayer perceptron (MLP), CNN, and RNN, were used concurrently. The former two types of NN were used for handling scalar and spatial input data, respectively; the latter was incorporated because TC trajectory can be considered as a time series. Feature selection and hyperparameter tuning were performed to optimize model performance. The introduction of a shortcut connection to the NN model was shown to improve the model training and performance.

The remainder of this paper is organized as follows. Section 2 describes the dataset and the methods used in this study. The architectures of the NN models are presented in section 3. In section 4, training of NN models, selection of input variables, and hyperparameters are elucidated, and their performance is evaluated in section 5. Finally, a discussion and summary are presented in sections 6 and 7, respectively.

2. Data and methodology

a. Historical TC observation

The TC best track data at 0000, 0600, 1200, and 1800 UTC from 1985 to 2019 (June–October) from the Regional Specialized Meteorological Center (RSMC) Tokyo Typhoon Center were used. The best track dataset archived the center location, central minimum pressure, maximum sustained wind speed, intensity grade, etc., of each TC formed over the WNP. Tropical depression and extratropical cyclones were excluded from the analysis.

TABLE 1. Two-dimensional variables and the corresponding pressure levels for the NN model input.

Variable name	Pressure level (hPa)
Vertical wind shear (VWSU, VWSV)	—
Steering flow (SFU, SFV)	—
Specific humidity (Q)	850, 500, 300
Temperature (T)	850, 500, 300
Geopotential height (Z)	500

b. Model forecast output

The global forecast datasets used in this study are the operational Global Ensemble Forecast System (GEFS) forecast (Hamill et al. 2011) and reforecast (Hamill et al. 2013). To maximize the number of samples produced by a model of the same version, the operational forecast dataset was used for the period 2012–14, and the reforecast dataset was used for the period 1985–2011 and 2015–19. These two GEFS datasets used model version 10 in their respective periods. The resolution of GEFS version 10 is T254 (~50-km horizontal resolution) out to 192 h with 42 vertical levels. The operational forecast dataset consists of 21 members, produced from 0000, 0600, 1200, and 1800 UTC initial conditions every day, while the reforecast dataset consists of 11 members only, and is produced once at 0000 UTC. The data available for download has a horizontal resolution of $1° \times 1°$. The five variables (vertical wind shear, steering flow, specific humidity, temperature, and geopotential height) and the corresponding pressure levels (850, 500, and 300 hPa) used for the NN input are listed in Table 1. These vertical levels were selected according to their association with the lower, middle, and upper troposphere. Wang and Holland (1996) and Zheng et al. (2007) established the influence of vertical wind shear on the motion of TC or TC-like vortex, which may depend on factors like the magnitude of shear and vertical structure of TC, using idealized model experiments. Steering flow, which is pressure-weighted and vertically averaged horizontal winds from 850 to 200 hPa, modulates the TC motion in the first order; large error in forecast flow field will result in large track error. It is often used for inferring to TC motion as such environmental flow can explain a large fraction of it (Elsberry et al. 1987). Subtropical high strongly governs the TC track in the WNP, and it is common to identify subtropical high in the 500-hPa geopotential height field (e.g., Ho et al. 2004). Yan et al. (2017) found that spatial distribution of moisture might alter TC intensification rate, and thus the interaction between TC circulation and environmental flow, which has a role in TC motion. According to Chan (2005), a change in temperature structure of the atmosphere is followed by a change in the distribution of potential vorticity (PV). Since TCs tend to move toward the location of maximum positive tendency of vorticity, any

change in temperature that leads to changes in PV and hence vorticity would have an impact on TC motion. Although sea surface temperature (SST) can have influence on TC movement (Katsube and Inatsu 2016), it is not available in GEFS and therefore could not be included as a predictor.

In late September 2020, GEFS was upgraded to version 12, which uses the finite-volume cubed-sphere dynamical core. We were uncertain how the behavior of GEFS varied with a different dynamical core and resolution, so data in 2020 and after were not included. Moreover, the reforecast dataset was obsolete.

c. Detection and tracking of TCs in GEFS

To obtain future TC positions predicted by GEFS and compare the performance of the present NN models against the dynamical forecast, vortex detection and tracking processes were required. The detection and tracking algorithm applied in this study was based on that adopted by UKMO (Heming 2017) with some modifications (Cheung et al. 2021). In this algorithm, relative vorticity at 850 hPa (RV850) was used to track a TC vortex in a model, as it helped provide a strong signal for the TC center position even at lower TC intensities. The nearest local minimum mean sea level pressure (MSLP) field to the highest value of RV850 was assigned as the model TC center. Some modifications and additional criteria were applied to search for a TC vortex for numerous cases. The details are provided in appendix A.

d. Clustering of TC tracks

During the development of a machine learning model, it is assumed that the training and test datasets are independent and identically distributed (Bickel et al. 2007; Wen et al. 2014); the samples in these two datasets are not correlated but are drawn from the same probability distribution. Violation of this assumption can result in poor generalization performance of the model. However, if validation and test datasets contain TCs selected randomly, the TCs in one dataset can be dissimilar to another one.

To create a more balanced validation and test datasets, TCs were assigned to these datasets so that both of them contain TC tracks in all parts of the spectrum. The 204 TCs in the entire dataset were clustered into various track patterns (or clusters). It is similar to putting them in different bins. Each track pattern contained tracks with similar geographical location and shape. Following Kim et al. (2011), fuzzy c-mean clustering was employed (Bezdek 1981). Although each track has a membership coefficient for each track pattern, which indicates the degree to which a track belongs to a certain track pattern, it was assigned to the track pattern where its membership coefficient was the largest, resulting in hard clusters. The number of track patterns chosen is explained in section 4b.

e. Feature selection

Feature selection is a process that seeks a subset of features/input variables that are relevant to a given problem for the construction of a machine learning model (Guyon and Elisseeff 2003; Leray and Gallinari 1999). Model performance can be improved by reducing the influence of noisy, correlated, or irrelevant features. To obtain the best/optimal subset, all combinations of variables must be evaluated exhaustively, which is computationally expensive and ineffective. Therefore, most selection methods perform searches such that a suboptimal subset is obtained instead of an ideal one. There are numerous feature-selection methods. This study used feature elimination following the computation of permutation feature importance. It is a model-dependent method: a model is first built, and feature selection is subsequently performed using the same model.

Permutation feature importance was first computed. This method estimates the importance of input features based on the increase in validation loss when the order of samples associated with a feature is randomly shuffled. It begins with a full set of features for model training, and the resulting validation loss is the baseline. This trained model is then run k times, where k is the number of features. In each of the k passes, the samples in the validation dataset of one feature is permuted, while those of other features are kept in place. Change in model performance, or the increase in validation loss from the baseline, is recorded. The features having larger increases in validation loss possess higher importance, and vice versa. A clear description and visualization of permutation feature importance can be found in McGovern et al. (2019) and their supplementary material. To obtain a more robust estimation, each feature was permuted, and validation loss was calculated 5 times in this study. The deviation of averaged validation loss from the baseline indicated feature importance.

Features that result in minor changes in validation loss can be considered as unimportant or redundant, so removal of these features can increase computational efficiency without significant reduction in model performance, or even can yield better performance. The features except the most important ones are eliminated, and the model is then retrained with these retained features for the next step, which is hyperparameter tuning. The number of retained features was determined arbitrarily to be six, which was half of the initial feature set.

f. Hyperparameter tuning

In machine learning, the parameters that control how a model is trained are called hyperparameters and their optimization for better model performance is called hyperparameter tuning. Like feature selection, searching for the optimal set of hyperparameters is prohibitive, necessitating a strategy for effective search. A grid search was adopted in the present study, where a subset of hyperparameters and several discrete values of each hyperparameter are specified. The model is trained with a value of each hyperparameter at one time, and all combinations are exhaustively attempted. Typical values of batch size, learning rate, and number of kernels were chosen for calibration, and they are listed in Table 2. Some of the hyperparameters are explained in appendix B. There was a total of 24 combinations.

g. Statistical analysis

A dependent sample t test (also called a paired t test) was applied to check if the mean track errors of NN prediction

TABLE 2. Hyperparameters and the value used for NN model training. When there are multiple values for a hyperparameter, the value used for feature selection is in bold; the value used by the NN1 and NN2 models with the lowest track error after tuning is indicated by a superscript. FC represents fully connected layers.

Hyperparameter	Values
Batch size	$8^{NN1,NN2}$, **16**, 32
Learning rate	0.01, **0.001**$^{NN1,\ NN2}$
Dropout rate (FC only)	**0.15**^{NN1}, 0.3^{NN2}
Optimizer	Adam
Kernel initializer	he_normal
Number of kernels (CNN only)	**10**^{NN2}, 20^{NN1}
Kernel size (CNN only)	(3, 3)
L2 regularization factor (CNN only)	0.001

were significantly larger than those of the GEFS forecast track at various confidence levels. Additionally, the Wilcoxon signed-rank test, which is the nonparametric counterpart of a paired *t* test, was used for comparison of the median. The null (alternative) hypothesis was that the mean or median of the track error of the NN prediction was greater than or equal to (smaller than) that of the GEFS track forecast.

h. Hardware and software for model development

Keras 2.3.1 and Tensorflow 1.15.0 were utilized to develop the NN models under Python 3.7.9. To make the results reproducible, random seeds were set (see Table S1 in the online supplemental material), and Tensorflow was patched with Tfdeterminism, version 0.3.0. Moreover, the NN models were trained using a computer with the following hardware: (CPU) Intel Xeon Silver 4116; (GPU) GeForce RTX 2080 Ti; (RAM) DDR4 16GB 2666 MHz × 18.

3. Architecture of neural network models

This section describes the configuration of the NNs. Since both two-dimensional (2D) data (e.g., model forecast fields) and scalar data (e.g., latitude and longitude of future TC positions) were utilized, it was necessary to use different NN architectures to handle the variety of data. Furthermore, TCs are reported at regular time intervals, so our task was regarded as a time series problem, where the application of RNN is suitable. The track error at a time step is dependent on TC location and the model environment where the TC is embedded at the previous time step(s), so it is necessary to use an algorithm that can pass earlier information to the latter part of prediction. Consequently, hybrid NN models were constructed by combining these three NN classes. Moreover, shortcut connection, which is a connection that skips one or multiple layers, is said to improve the ability to generalize and thus improve performance (Daliri and Fattan 2011; Rabuñal and Dorado 2006). Hybrid models without and with a shortcut connection were tested and named NN1 and NN2, respectively (Fig. 1).

a. Multilayer perceptron branch

The MLP is a vanilla (plain) NN. Sometimes it is simply called "NN," "deep NN," or "feedforward NN" when other NN architectures are not involved in the same context. A typical MLP has one input layer, one or multiple hidden layers, and one output layer, with each layer containing at least one node. Consider a node in the hidden layer: input signal x_i from node i in the previous layer is first multiplied by a weight w_i, and all the other n weighted input signals are summed to give a total input to the hidden node in the form

$$a = \sum_{i=1}^{n} w_i x_i + b, \tag{1}$$

where b is the bias. The output of this hidden node y is then subjected to an activation function f, and thus, $y = f(a)$. The activation function provides nonlinearity to an NN model. An NN model contains a collection of the two equations above, and weights and biases are updated to give the best fit to the input data. Further explanations and examples of MLP can be found in Gardner and Dorling (1998) and Bishop (1994). The MLP branch of the hybrid NN had an input layer and a hidden layer of eight nodes.[1] The input nodes corresponded to the scalar variables: TC position (latitude and longitude) predicted by the GEFS. The hyperbolic tangent (tanh) activation function was used.

b. Convolutional NN branch

CNN is an architecture that is common for image recognition or extraction of features from images. Essentially, it comprises of several layers: input, convolutional, pooling, fully connected, and output layers. The input layer contains multiple spatial fields that are stacked together. For the convolutional layer, multiplications (dot product) of the kernel (or filter, which is a set of weights) with an input image are performed. A kernel has the same "depth" as the input image, but the horizontal dimensions are much smaller. Since a kernel has a smaller size, it can slide over an image in the two horizontal dimensions. Here, the kernel represents a specific spatial feature (e.g., edges, curves), so convolution using a kernel allows us to search for certain types of features in the input image. Multiple kernels are used to discover more features from the input images. The output from a convolutional layer, called a feature map, is then downsampled by the pooling layer. This downsampling is performed by computing either the maximum or average value inside a small 2D window, which slides over the feature map similar to the kernel sliding over the input image. It reduces the resolution of the feature map, letting only large-scale structures of the input image remain. When there are multiple convolutional and pooling layers, complicated spatial features can be extracted. Finally, fully connected layers, which look like MLP, are used to perform the prediction task. Readers can consult Albawi et al. (2017) for a more in-depth description of the CNN. The CNN branch in our model contained an input layer comprising one or multiple spatial fields stacked together, and three repeated convolutional blocks. Each of the convolutional blocks consisted of a 2D convolutional, an activation [in which

[1] Strictly speaking, the MLP branch is not a complete MLP because it does not have an output layer.

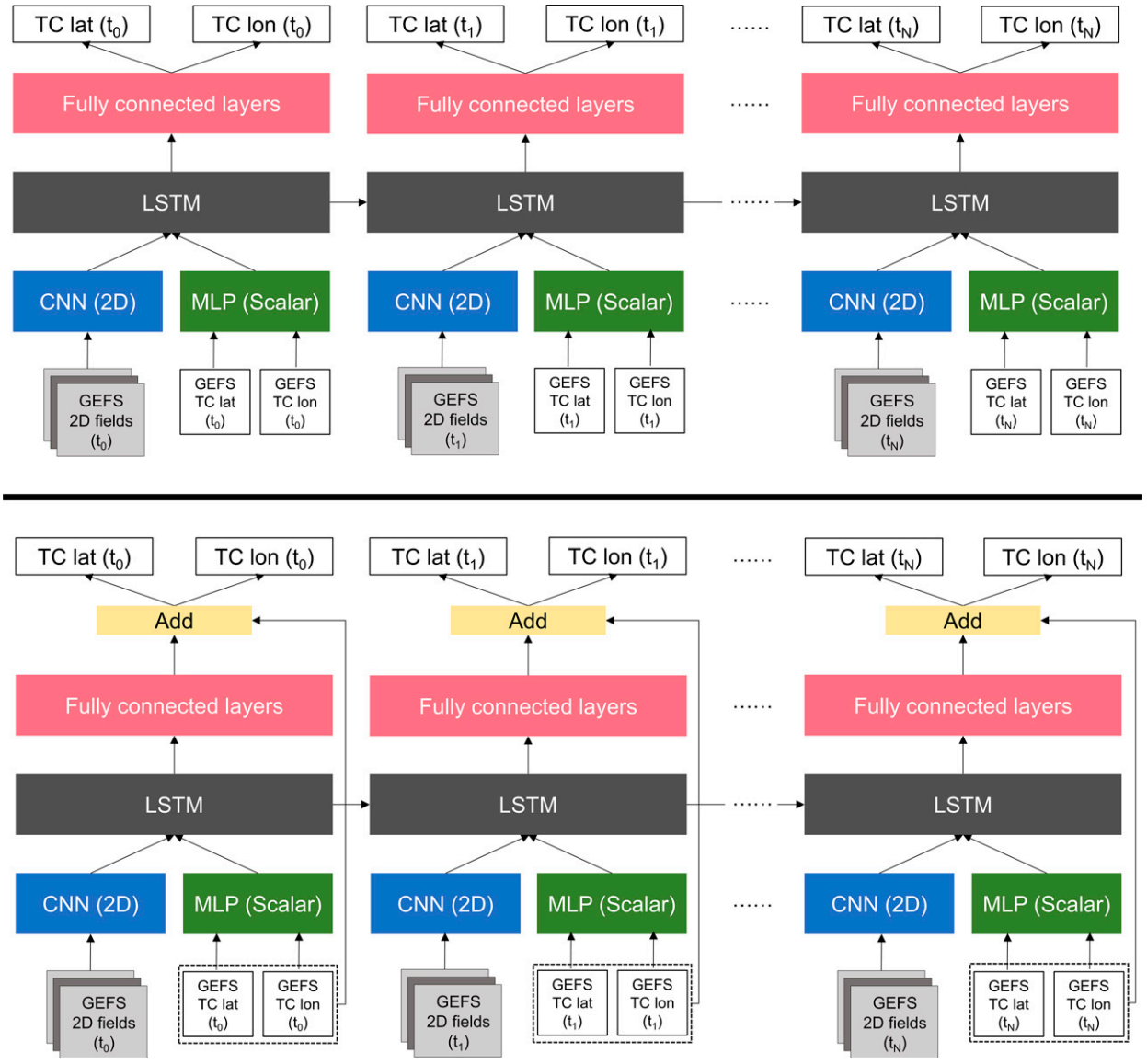

FIG. 1. The hybrid NN (top) without (NN1) and (bottom) with the shortcut connection (NN2).

a leaky rectified linear unit (leaky ReLU) function is used], a batch normalization, and 2 × 2 max pooling layers in sequence, except for the final block, which used global average pooling instead of max pooling.

c. *Recurrent NN (LSTM) and fully connected layers*

RNNs are a class of NN that consider historical information in the computation, and they are designed to handle sequential data. They accept inputs and produce an output of any given length. A typical RNN suffers from the vanishing gradient problem, so LSTM (Hochreiter and Schmidhuber 1997), a special version of RNN, was developed to overcome this problem. An LSTM cell receives three data streams: an input matrix at the current time step ($\mathbf{X}_t$), and a hidden state matrix and a cell state matrix from the previous time step ($\mathbf{H}_{t-1}$ and $\mathbf{C}_{t-1}$, respectively). The input matrix $\mathbf{X}_t$ represents new information, while $\mathbf{H}_{t-1}$ and $\mathbf{C}_{t-1}$ contain short-term and long-term memories, respectively. There are two outputs from an LSTM cell, $\mathbf{H}_{t-1}$ and $\mathbf{C}_{t-1}$. An LSTM cell is governed by the following six equations:

$$\mathbf{I}_t = \sigma(\mathbf{W}_{xi}\mathbf{X}_t + \mathbf{W}_{hi}\mathbf{H}_{t-1} + \mathbf{b}_i), \tag{2}$$

$$\mathbf{F}_t = \sigma(\mathbf{W}_{xf}\mathbf{X}_t + \mathbf{W}_{hf}\mathbf{H}_{t-1}\mathbf{b}_f), \tag{3}$$

$$\mathbf{O}_t = \sigma(\mathbf{W}_{xo}\mathbf{X}_t + \mathbf{W}_{ho}\mathbf{H}_{t-1}\mathbf{b}_o), \tag{4}$$

$$\mathbf{G}_t = \tanh(\mathbf{W}_{xc}\mathbf{X}_t + \mathbf{W}_{hc}\mathbf{H}_{t-1} + \mathbf{b}_c), \tag{5}$$

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \mathbf{G}_t, \quad \text{and} \tag{6}$$

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t), \tag{7}$$

where $\mathbf{W}_{xi}$, $\mathbf{W}_{hi}$, $\mathbf{W}_{xf}$, $\mathbf{W}_{hf}$, $\mathbf{W}_{xo}$, $\mathbf{W}_{ho}$, $\mathbf{W}_{xc}$, and $\mathbf{W}_{hc}$ are weight matrices, $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$, and $\mathbf{b}_c$ are bias vectors, σ is a sigmoid activation function, tanh is a hyperbolic tangent activation function, and $\odot$ indicates elementwise multiplication. The subscripts of the weight matrices and bias vectors (x, h, i, f, c, and o) indicate their association with certain matrices ($\mathbf{X}_t$ or $\mathbf{H}_{t-1}$) and/or gates ($\mathbf{I}_t$, $\mathbf{F}_t$, $\mathbf{G}_t$, $\mathbf{O}_t$). Since the output of a sigmoid function must be between 0 and 1, the activation functions in Eqs. (2)–(4) serve as gates for controlling the fraction of weighted input, and the hidden state can be retained for further calculation in Eqs. (6) and (7). Therefore, they are called the input ($\mathbf{I}_t$), forget ($\mathbf{F}_t$), and output gates ($\mathbf{O}_t$), respectively. The gate $\mathbf{G}_t$ is called candidate memory (or candidate gate), which scales information at the current time step with a hyperbolic tangent function.

This scaled input is then subjected to the input gate and updates the cell state. Moreover, the hidden state of each time step is also the output from a LSTM, but cell states are kept internal within a LSTM. In the present NN models, the input tensor (data array) for LSTM is formed by concatenating the output tensors of the MLP and CNN. There was one LSTM layer possessing 16 nodes, followed by two fully connected layers with 10 and 5 hidden nodes, and the output layer. Dropout layers were inserted after both fully connected (hidden) layers. None of the three layers following the LSTM layer used a nonlinear activation function. It should be noted that multiple hidden layers with linear activation function is equivalent to a single hidden layer. The output layer corresponds to postprocessed TC positions (top panels in Fig. 1).

d. *Shortcut connection*

In NN2 (bottom panels in Fig. 1), there is a shortcut connection linking the input layer of the MLP branch to an "Add" layer, which also receives the output from the fully connected layers following the LSTM layer. The "Add" layer performs an addition operation that sum up the two inputs to the layer. Consequently, the NN algorithm will try to learn the residual, which in our case is the deviation of the GEFS-predicted TC position from the true value. In contrast, NN1 was trained to directly predict the TC position.

4. Training of neural network models

a. *Data preparation*

The inputs to the NN models contained forecast fields in the domain (0°–50°N, 90°–170°E) and forecast TC positions at lead times of days 0, 2, 4, 5, and 6 (i.e., 0, 48, 96, 120, and 144 h), both from the GEFS, while ground truth labels were from the TC best track data of RSMC Tokyo. Days 1 and 3 were excluded to reduce the computation loading. Although not shown in the figure, the inclusion of these two days did not significantly affect the prediction results after 4 days. Each TC can contribute to multiple TC cases when it survives for longer than six days. For example, if a TC lasts for seven days and forecast is produced daily, two TC cases are obtained: the first starts at 0 h (i.e., day 0), and the other starts at 24 h (i.e., day 1). Since the GEFS dataset consists of multiple ensemble members (perturbations), they are treated as individual samples.

b. *Data split*

The samples in the entire dataset were split into three partitions: training, validation, and test datasets. The training dataset is responsible for fitting the model by generalizing any relationships or patterns within the data. The validation dataset is required when model selection by methods such as feature selection or hyperparameter tuning is performed. The test dataset is used for the final evaluation of the model performance. Ten percent of TCs (20 TCs) was assigned to the validation dataset, and another 10% (20 TCs) went to the test dataset. The remaining TCs were left for training. As explained in section 2d, clustering was performed prior to the selection of TCs for each dataset. For simplicity, the number of track patterns was designated to be 20, so a TC could be randomly picked from one track pattern. The 20 track patterns are shown in Fig. 2, and the resulting validation and test datasets contain similar distributions of tracks (Fig. 3). TCs in these two datasets are listed in Table S2. As a result, there were 5379, 411, and 954 samples, corresponding to 540, 57, and 79 TC cases.

The samples associated with the same TC must not exist in the training and validation/test datasets simultaneously (e.g., ensemble members 0–5 go to the training dataset while the remaining members go to the test dataset). If all the samples are randomly allocated to these datasets, it is possible that samples made up of different GEFS ensemble members of the same TC case are assigned to both the training and validation datasets simultaneously. Some of the ensemble members may be similar. When similar or exact copies of samples are found in the training and validation datasets, validation loss is close to that of the training dataset, and it weakens the model's generalization ability (Santos et al. 2018). Consequently, the TCs were partitioned instead of the TC cases or the samples.

c. *Data transformation*

Input data are usually transformed to have small magnitudes before training for faster training and less prediction error (Jin et al. 2015; Sola and Sevilla 1997). Moreover, when multiple variables with different range of magnitudes are selected as the model input, those having much larger variances can dominate. Therefore, data transformation is an integral part of the training of machine learning models.

Each of the input and output variables selected for the present study were scaled to values between 0 and 1 independently, using min-max normalization: $x_i' = (x_i - x_{\min})/(x_{\max} - x_{\min})$, where x_i' and x_i denote the normalized and original values of a variable at grid point i, respectively, and $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of the variable across all grid points, time steps, and ensemble members of the training dataset, respectively. The samples in the validation and test datasets were transformed

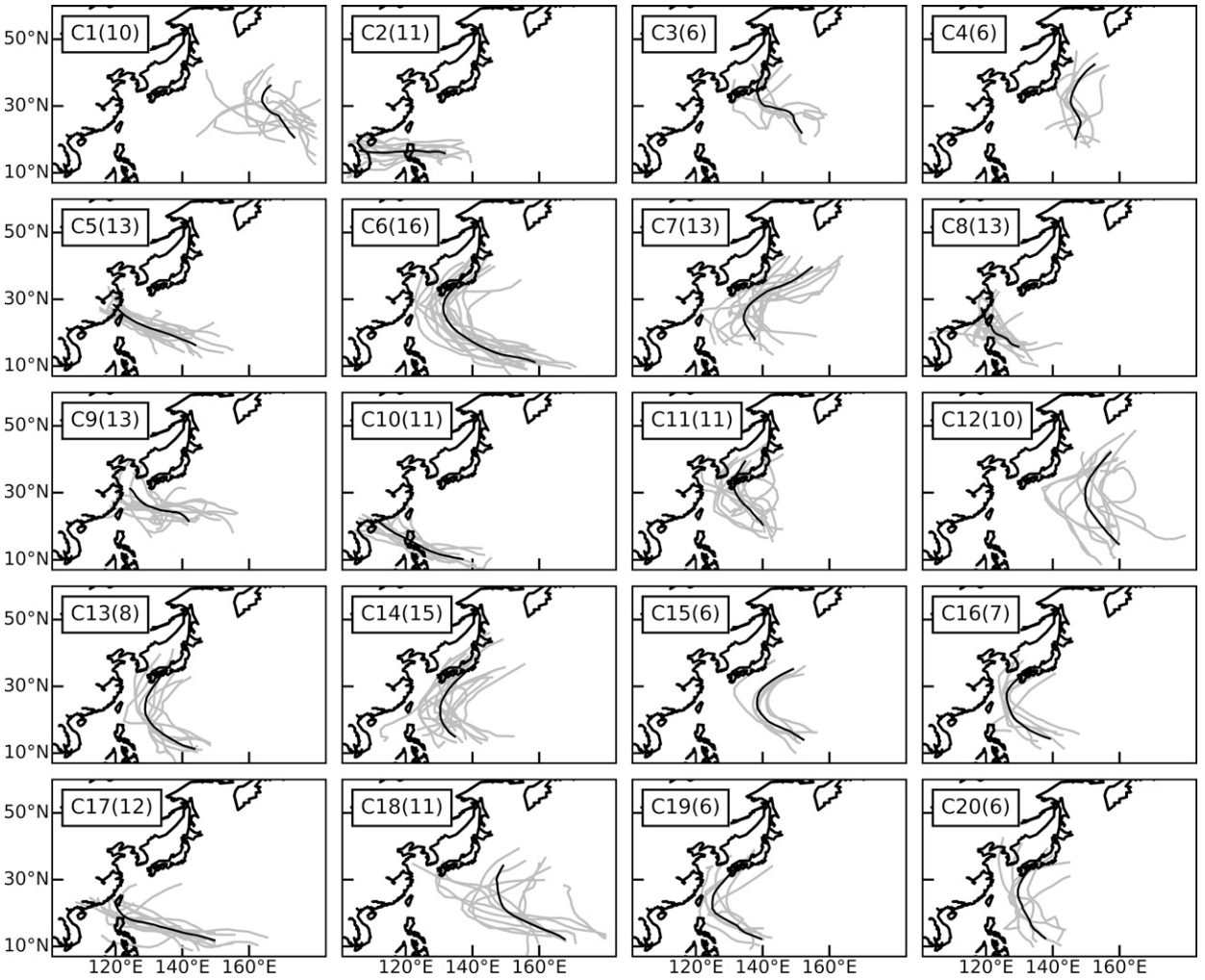

FIG. 2. The TC tracks (gray lines) of each track pattern for the entire dataset. The numbers inside the parentheses indicate the number of tracks contained in the particular pattern. The thick black line is the mean track of a track pattern.

using the minimum and maximum values calculated from the training dataset.

d. Loss function

The goal of training of a machine learning model is to minimize the loss function. For a regression task, mean absolute error or mean-squared error are common loss functions. However, neither of them is an accurate or typical measure of the distance between the true and predicted values because the outputs of the present NN models are latitude and longitude (in degrees), which are not location invariant. A custom loss function was therefore written to calculate the great-circle distance between the true and predicted TC positions (i.e., track error) for each forecast lead time.

e. Training

Training an NN model is equivalent to determining its weights to minimize loss. This is similar to finding the coefficients in a polynomial fit. The weights are updated continuously as the outputs of the NN do not match the ground truth labels. Backpropagation is an algorithm that computes the gradient of the loss function with respect to the weights of an NN for one sample, and one layer at one time. The weights in a NN are adjusted based on the above computation.

Both the NN1 and NN2 models underwent feature selection (permutation feature importance and feature elimination) and hyperparameter tuning. For feature selection, the batch size, learning rate, number of kernels, and kernel size were 16, 0.001, 10, and 3×3, respectively (Table 2). Adaptive moment estimation (Adam) was the optimization algorithm for the training of the two NN models. The kernel initializer was a He normal initializer. Each training lasted for 100 epochs, but only the set of model weights in the epoch that resulted in the minimal validation loss (which is an average across all lead times) within 100 epochs was saved. The validation loss in this epoch was an indicator of this model's performance.

Figure 4 shows the validation loss for each permuted feature of NN1 and NN2, and also the baseline (unpermuted loss). The

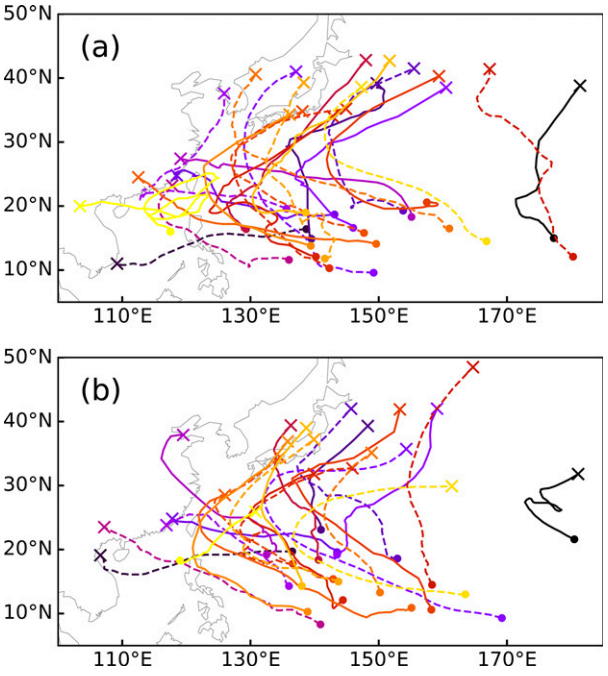

FIG. 3. TC tracks contained in the (a) validation and (b) test dataset. Dots and crosses represent the starting and ending location of each TC, respectively. Different line styles and colors are used for better identification of tracks.

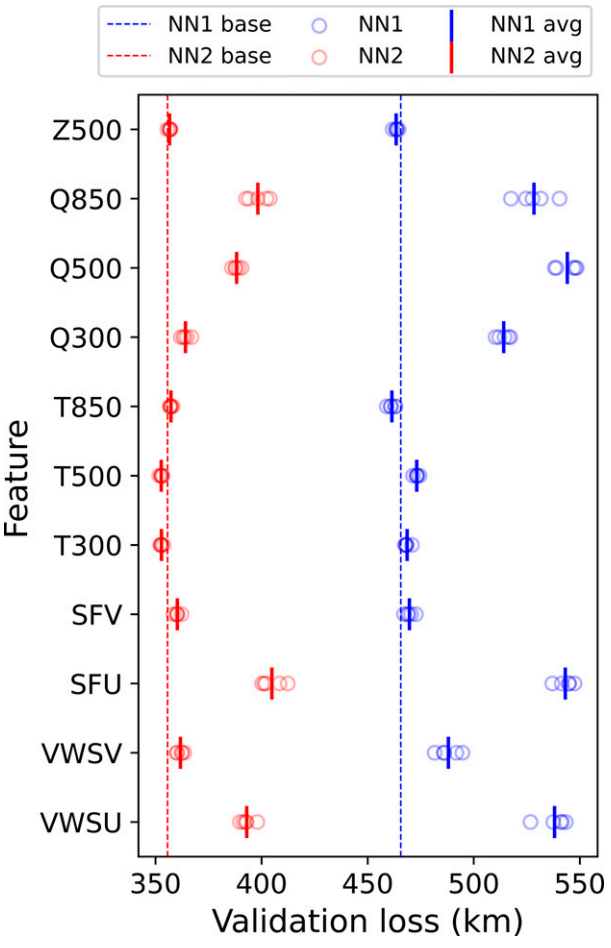

FIG. 4. The validation losses of NN1 (blue) and NN2 (red) for each permuted feature. A ring indicates one of the five permutations, while a solid line represents the average of all permutations for a feature. The loss for the baseline (unpermuted loss) is drawn as the dashed line.

baseline for NN1 and NN2 were 465.6 and 355.7 km, respectively. The training losses were 382.4 and 193.0 km for the two NN models. For NN1, the permutation of Q500 caused the greatest departure of validation loss from the baseline, followed by SFU and VWSU. Q850, Q300, and VWSV also had strong impacts on the validation loss. Temperature at all levels (T850, T500, and T300), SFV, and Z500, however, all had validation losses very close to baseline, indicating their influences on model performance were minimal. For NN2, a similar result was obtained. Q850, Q500, SFU, and VWSU had greater importance compared to other features. The importance of Q300 and VWSV were next to the four features mentioned above, but were not far from the baseline, and were only slightly larger than SFV. To preserve computational efficiency, we arbitrarily eliminated half of the features that were least important (or caused small changes in validation loss after permutation). Subsequently, the six features retained for both NN models were Q850, Q500, Q300, VWSU, VWSV, and SFU.

Hyperparameter tuning of NN1 and NN2 was performed using the features selected in the previous step. As shown in Fig. 5, the best model for NN1 used 0.001, 8, 20, and 0.15 for the learning rate, batch size, number of kernels, and dropout rate, while they were 0.001, 8, 10, and 0.3 for NN2. NN1 was slightly overfitted when the dropout rate was 0.15 but was underfitted when the dropout rate was increased to 0.3. Meanwhile, NN2 was overfitted in all sets of hyperparameters, but the overfitting was less severe when a higher dropout rate was used. Under the same hyperparameter set, the validation loss was generally lower when the model was less overfitted or underfitted. It can be seen from the NN1 models trained with dropout rate of 0.15 and NN2 models trained with dropout rate of 0.3. On the other hand, the validation loss (and training loss) of NN2 was less than that of NN1 for all sets of hyperparameters.

5. Model evaluation

The test dataset was applied to the best models of NN1 and NN2 selected after feature selection and hyperparameter tuning for the final evaluation of model performance at each lead time (except for day 0, because the exact observed TC position is known). For deterministic weather forecasts of continuous variables, the three commonly used verification measures are accuracy, association, and skill (Jolliffe and Stephenson 2011). Accuracy measures how well a forecast agrees with the corresponding observation, and it is indicated by the great-circle distance between actual and predicted TC position calculated using haversine formula. Along-track error and cross-track

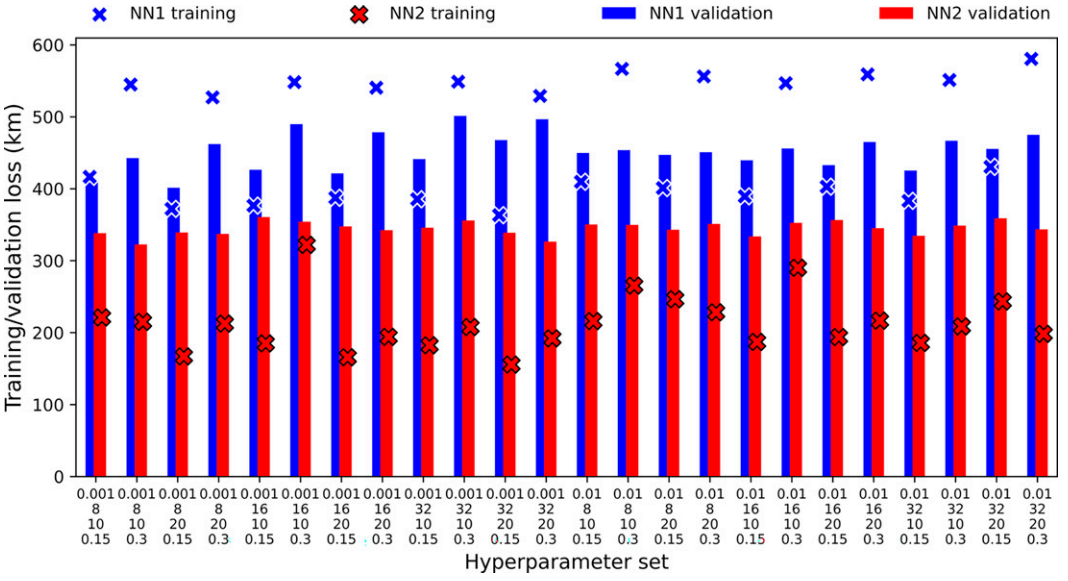

FIG. 5. Training and validation losses for each combination of hyperparameters. The tick label of the x axis indicates the learning rate, batch size, the number of kernels, and dropout rate. The training error is the one in the epoch where lowest validation loss is recorded.

error are also verified. Association quantifies the linear relationship (correlation) between forecast and observation. Since correlation coefficient can only be computed for one variable at one time, the predicted center latitude and longitude of a TC need to be evaluated separately. Finally, skill is the accuracy of the forecast relative to a zero-skill baseline model, which is calculated as $[(E_{\text{Base}} - E_M)/E_{\text{Base}}] \times 100\%$, where E_{Base} represents the track error from a baseline model, while E_M is that from either NN1, NN2, or GEFS. Climatology and persistence models are usually used to be the baseline reference, so CLIPER model, which is a multiple regression model, for TC track forecast is used. The details of the CLIPER model used here are explained in appendix C.

a. Accuracy

The boxplots in Fig. 6 and Table 3 show the distribution of the test loss from NN1, NN2, and GEFS. Only box and whiskers are plotted to preserve clarity. The figure with outliers is shown in Fig. S1 in the supplemental material. NN1 had greater test loss in terms of mean and median on days 2, 4, and 5 than GEFS, but the interquartile range (IQR) was shorter on the latter two days. On day 6, NN1 became better than GEFS for the three statistical measures, with a significantly smaller mean test loss. For NN2, the mean test losses on days 2, 4, and 5 were all significantly smaller than GEFS with shorter IQRs. Although the mean test loss and IQR of NN2 was still smaller on day 6, the difference in mean test loss was no longer significant. Although the median of NN2 on day 4 and 5 were greater than GEFS, they were still considered as smaller than GEFS from the Wilcoxon signed-rank test. It is because the samples in NN2 and GEFS must not be regarded as independent to each other, and each set of matched pair (same forecast) should be compared instead of the two distributions.

In addition to the track error as represented by the distance between observed and forecast position of TCs, the along-track error and cross-track error (Fig. 7), which are two components of the track error, were also analyzed. Evaluation of these two components is common for TC track forecasts (e.g., Fiorino et al. 1993; Heming 2017; Leonardo and Colle 2017; Tsui and Miller 1988). A positive along-track error indicates a fast bias in the forecast track, while slow bias is represented by a negative along-track error. For cross-track error, positive and negative values imply right and left bias, respectively. The example in Fig. 7 shows a forecast with positive along-track error (fast bias) and negative cross-track error (left bias). Although these two components are orthogonal, they are not independent because a small cross-track error follows a large along-track error (Fiorino et al. 1993).

The along- and cross-track bias are shown in Fig. 8. NN1 tended to have negative along- and cross-track bias, while NN2 and GEFS had positive biases. The statistics of the absolute magnitude of the three model forecasts were also compared (Table 4). For NN1, the mean of the along-track error was significantly smaller than GEFS on days 4, 5, and 6, but the mean of the cross-track error was larger on days 2, 4, and 5. As a result, the improvement in the along-track error on day 4 and 5 by NN1 was cancelled out by the decline in cross-track error. Conversely, NN2 improved both the mean/median of along- and cross-track error of GEFS on all lead times except the former one on day 6. It can also be observed from Table 4 that whenever the mean loss of the along-track error of NN1 or NN2 was greater than GEFS, the same was true for the median loss, but this relationship did not hold for the cross-track error.

Figure 9 shows the evolution of along-track error and cross-track error of each model. Each data point represents the

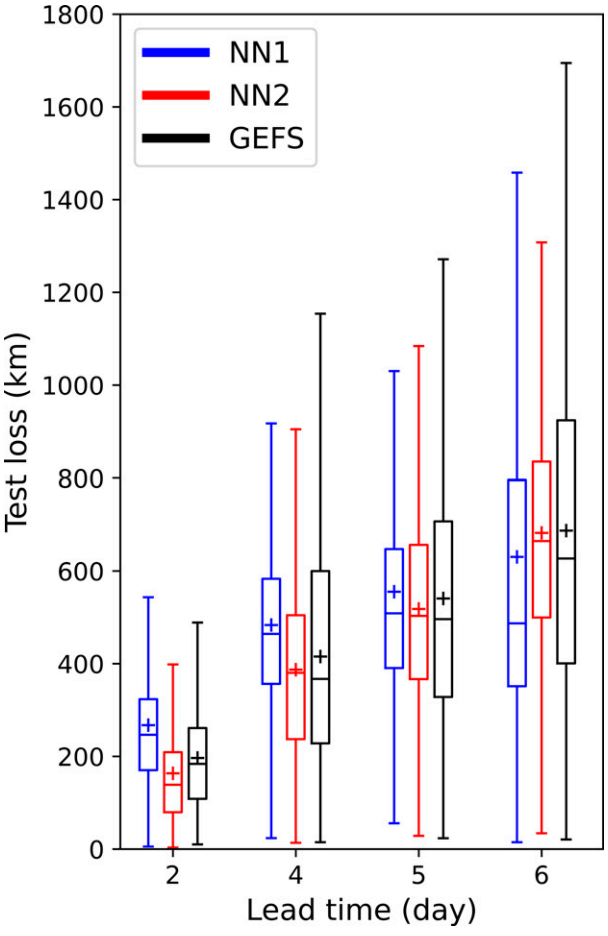

FIG. 6. Comparison of the test loss (km) of the prediction from NN1 (blue), NN2 (red), and GEFS (black) at each lead time. A cross inside a box represents the mean value, while a ring represents an outlier (defined as the data points that fall above the 75th percentile plus 1.5 interquartile range). The figure with outliers can be referred to in Fig. S1.

average across all samples in the test dataset at each lead time. GEFS tended to have an increasing along- and cross-track error (fast and right bias), although it started with a slow bias. NN1 had slow bias on all days. The slow bias was close to zero on day 4, but increased on the next two days. Right bias was observed on day 2, but it changed to another side on day 4. It approached zero after an increase on day 5. NN2 had an evolution similar to that of GEFS. A slow bias turned into a fast bias, and the right bias fluctuated with time. However, the magnitudes of the two types of bias were smaller than those of GEFS on all days. Given the similar trajectories of NN2 and GEFS, and smaller along- and cross-track errors of the former one, reduction in systematic speed and direction bias of GEFS prediction is likely to be corrected by NN2.

b. *Association*

Figure 10 shows the time series of correlation coefficient between observed and predicted latitude and longitude for each model. Regardless of lead time or variables (latitude and longitude), the correlation coefficients decreased with time. For latitude, GEFS had the strongest correlation between prediction and observation, followed by NN2 and NN1. The correlation coefficient of the three models started with 0.95 (NN1)–0.98 (GEFS) on day 2 and dropped to 0.65 (NN1)–0.79 (GEFS) on day 6. For longitude, the correlation coefficients were usually higher than those of latitude, and they decreased at a slower rate. NN2 had the greatest correlation coefficients among the three models.

TABLE 3. Mean, median, and IQR of test loss of NN1, NN2, and GEFS at each lead time. The unit of the values is kilometers. The asterisk indicates that the mean/median of NN test loss is significantly smaller than GEFS at the 99% confidence level.

Day	Statistics	NN1	NN2	GEFS
2	Mean	267	163*	197
	Median	247	139*	184
	IQR	153	130	153
4	Mean	483	387*	415
	Median	464	380*	367
	IQR	227	267	372
5	Mean	555	518*	541
	Median	509	503*	496
	IQR	257	289	379
6	Mean	630*	682	687
	Median	487*	664	627
	IQR	444	335	523

c. *Skill*

The relative forecast skill of NN1, NN2, and GEFS is shown in Fig. 11. The skill of NN1 on days 2–5 was rather low, which increased from 12.0% to 14.7%. It attained 20.3% on day 6. On the other hand, NN2 and GEFS had similar trends in forecast skill, although the former one was more skillful. On day 2, NN2 and GEFS have skills of 46.3% and 35.1%, respectively. The skills decreased continuously with time, and became less than 14% on day 6, which was lower than NN1.

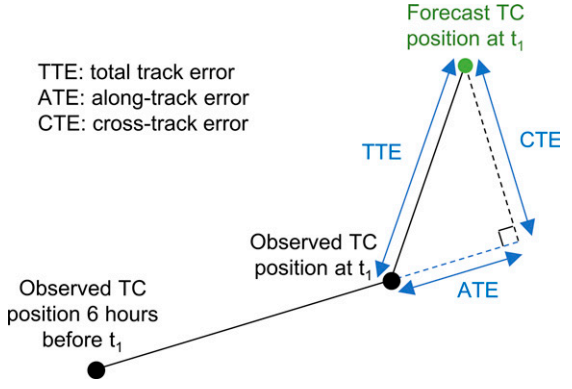

FIG. 7. Illustration of the along-track error (ATE) and the cross-track error (CTE).

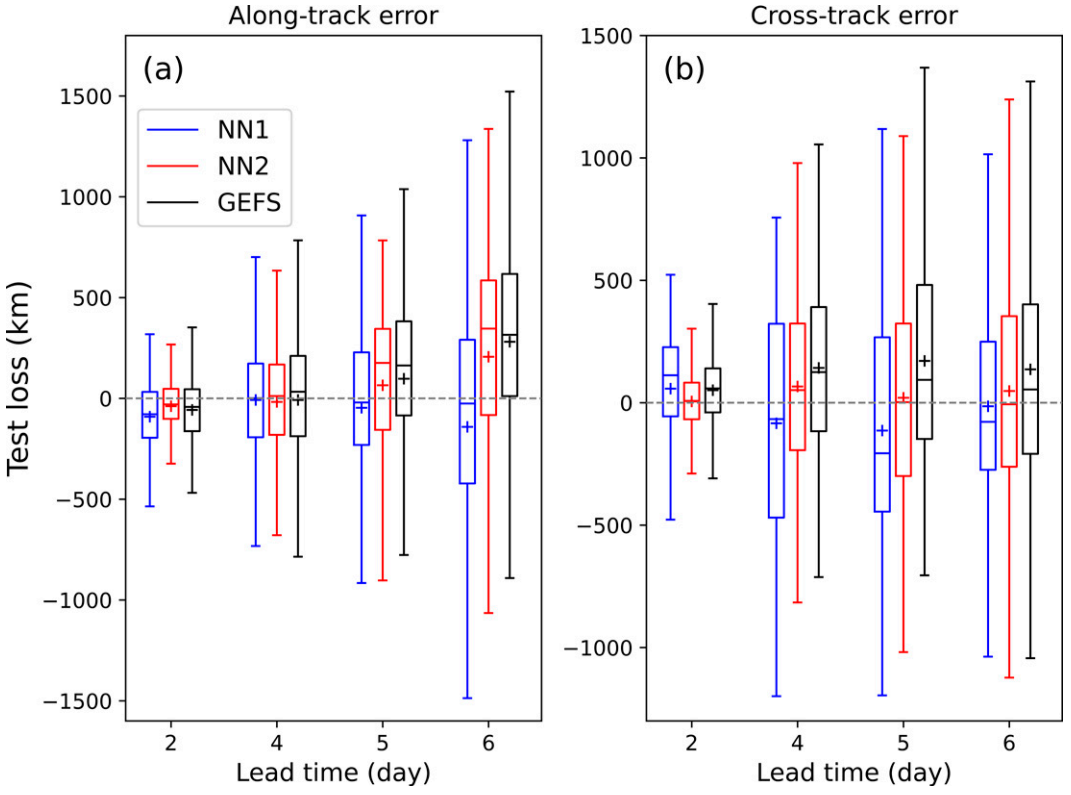

FIG. 8. As in Fig. 6, but for (a) ATE and (b) CTE.

d. Proportional of improved samples

Apart from the reduction in the mean track error of a large group of samples, the number of samples with reduced track error is also an important indicator of NN model performance. Figure 12 shows the scatterplot of the test loss of GEFS against NN1 and NN2. A sample located above the identity line (or line of equality) indicates a reduction in track error by NN. For NN1, 29%, 44%, 51%, and 60% of the samples on days 2, 4, 5, and 6 had smaller track errors than in GEFS (top panels of Fig. 12). The proportions of improved samples after postprocessed by NN2 were larger than those of NN1, except for day 4 (69%, 58%, 56%, and 53%; bottom panels of Fig. 12).

TABLE 4. As in Table 3, but for along-track error and cross-track error (in km) of test dataset. Absolute value of the test loss was used for calculation of statistics. The asterisk, ampersand, and pound indicate that mean/median of NN test loss is significantly smaller than GEFS at the 99%, 98%, and 95% confidence level, respectively.

		Along-track error			Cross-track error		
Day	Statistics	NN1	NN2	GEFS	NN1	NN2	GEFS
2	Mean	150	104*	126	193	104*	123
	Median	120	78*	101	186	76*	101
	IQR	141	101	136	170	111	118
4	Mean	232#	222*	246	379	269*	288
	Median	184	174*	204	382	247*	236
	IQR	241	244	278	285	280	302
5	Mean	292*	329*	347	412	335#	347
	Median	230*	308#	316	402	313#	280
	IQR	294	276	364	312	329	377
6	Mean	440*	500	487	361	357&	375
	Median	325*	482	432	265	302#	270
	IQR	345	409	488	342	394	395

The proportion of the samples above the 90th percentile of the GEFS test loss improved by NN1 and NN2 were also examined (Table 5). This analysis can specifically demonstrate the ability of the present NN models to improve GEFS predictions that contain large errors. The 90th percentile is indicated by the horizontal dashed lines in Fig. 12. For NN1, 69% of the samples were improved on day 2. The percentage rose to 95% on day 4, 96% on day 5 and 98% on day 6. For NN2, the fraction of samples with NN track errors smaller than that of GEFS for the four lead times were 62%, 94%, 95%, and 98%. The percentages were smaller than NN1 on days 2, 4, and 5, but greater after that.

e. Spatial distribution of track error

To understand the regions in which NN prediction can give better prediction than GEFS, the improvement caused by NN on each test sample were examined (Fig. 13). The majority of improved samples (blue) by NN1 were clustered mainly in two regions: south to Japan at 30°N, and farther south at 22°N. Most of the samples that had increased track error were found to the east of about 135°E. In comparison, the rises or

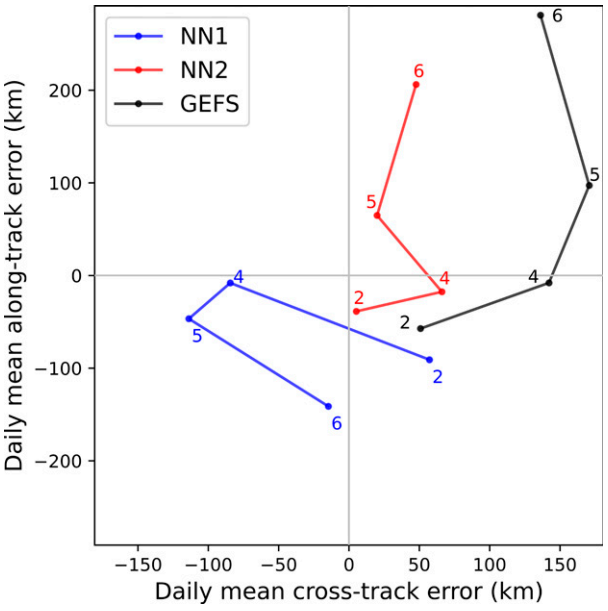

FIG. 9. Evolution of the ATE and CTE of NN1, NN2, and GEFS. The number next to a dot represents the lead time (days) of the corresponding model.

drops in test loss by NN2 were weaker than NN1. Two major clusters of improved samples by NN2 could be observed near the northern coast of Japan, and the region near 22°N, 135°E. Most of the samples at the southwest of Japan had increased test loss.

6. Discussion

a. *Influence of the application of the shortcut connection*

Although the test loss of NN2 was not significantly smaller than GEFS for all lead times (Fig. 6, Table 3), from the results obtained from feature selection and hyperparameter tuning with validation dataset (Figs. 4 and 5), or model evaluation with test dataset (Figs. 6 and 12), it is evident that the application of NN2 can result in lower track error averaged across all lead times than NN1. A possible reason for the superior performance of NN2 over NN1 in numerous situations can be the use of the shortcut connection. The "core" of NN2 was trained to learn the deviation from the normalized GEFS-predicted TC location, which should have magnitude much smaller than that of the normalized latitude and longitude value. It has the same positive effect as performing data normalization for data (section 4c). The difference in NN model characteristics caused by the inclusion of the shortcut connection can be speculated from Fig. 12. The scatterplots for NN1 had larger spreads than NN2, while the data points in the scatterplots for NN2 were closer to the identity line. It indicates that the predictions of NN2 were closer to those of GEFS, but NN1 was given the freedom to produce predictions in a larger range of magnitude. Figure 9 also shows that NN2 underwent evolution similar to GEFS but with smaller biases.

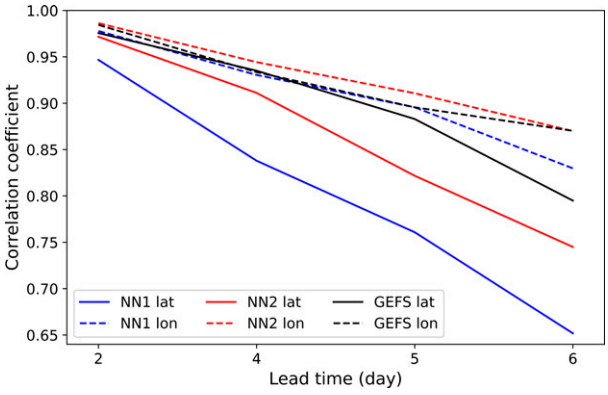

FIG. 10. Correlation coefficient between observed and predicted latitude or longitude of TC for NN1, NN2, and GEFS at each forecast lead time. A solid line represents latitude, while a dashed line represents longitude.

b. *Design of loss function*

The loss function is an important part of the development of an NN model because it can dictate the efficiency of training and final prediction. The current loss function is the average track error across all lead times. It is apparent that the improvement of track forecast on day 6 by NN2 was insignificant compared to other lead times (Fig. 6 and Table 3). Assigning heavier weight to loss to later forecast lead time can be a possibility to train a NN for focusing model performance less in the earlier part of track forecast. Alternatives such as the magnitude of improvement (difference between NN-predicted track error and GEFS-predicted track error) or the proportion of improved samples can also be considered alongside track error. Moreover, limiting the maximum reduction of track error may also be useful for preventing the NN model from exploiting some samples that are easy to improve.

c. *Suboptimal generalization of data*

Figure 5 shows that the variation in validation loss was small in all combinations of hyperparameters, while training loss exhibited apparent change according to hyperparameters.

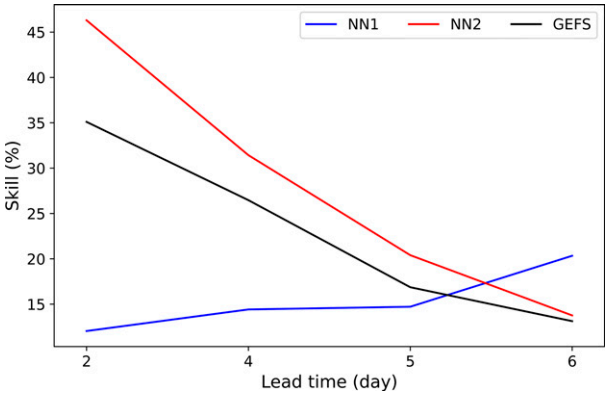

FIG. 11. Forecast skill of NN1, NN2, and GEFS relative to baseline model (CLIPER) at each forecast lead time.

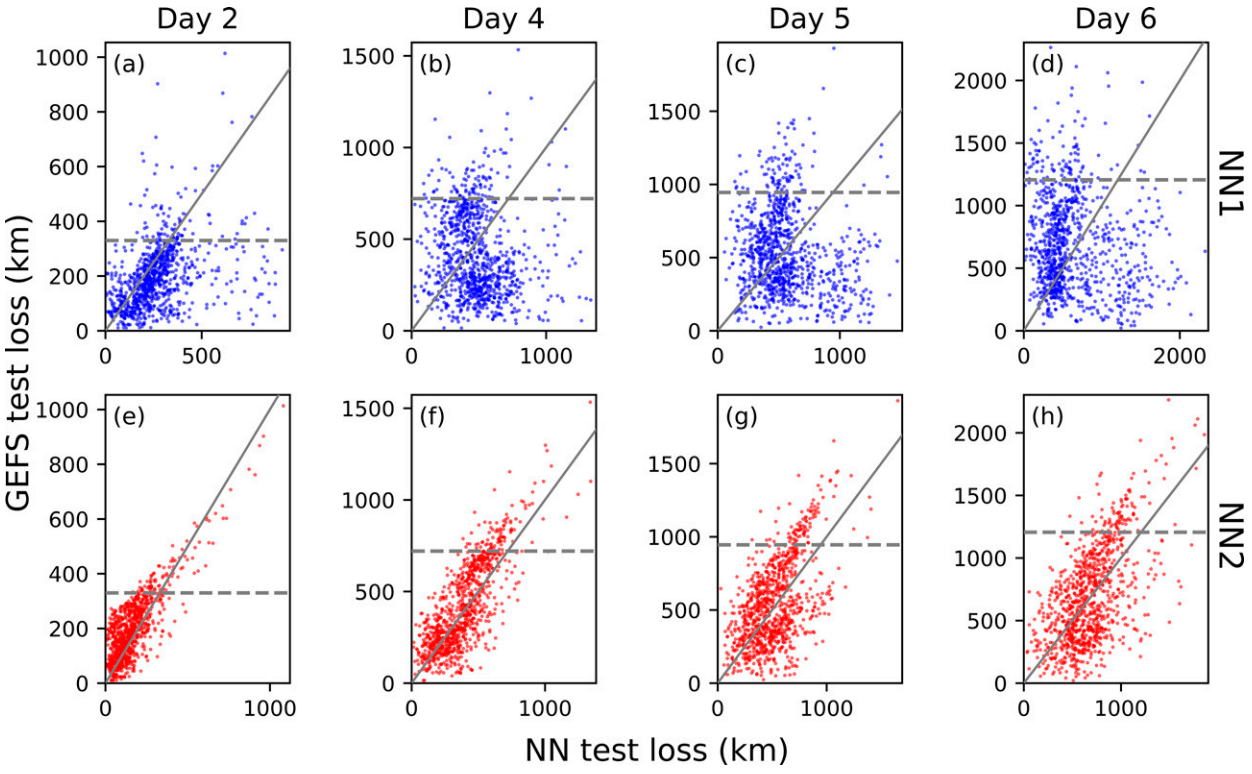

FIG. 12. Scatterplots of GEFS vs (top) NN1 and (bottom) NN2 test loss (km) for lead time of (a),(e) day 2, (b),(f) day 4, (c),(g) day 5, and (d),(h) day 6. Gray solid line and dashed line indicate identity line and the 90th percentile of GEFS track error, respectively.

A possible cause for the dissociation between training losses and validation losses is the strong dissimilarities among TC tracks in the present dataset, which consists of tracks that survive for at least six days. As seen in Fig. 2, the tracks within the same track pattern looked different to each other in most of the track patterns. As each of these long tracks possesses high uniqueness, the same problem will probably still exist even another validation/test dataset is created by resampling the dataset. Therefore, a good level of generalization of data can be very difficult to achieve.

Besides, NN2 was consistently overfitted in all combinations of hyperparameters. Selecting a combination of hyperparameters that leads to a severely underfitted or overfitted model is not desirable. A smaller dropout rate (0.15) usually led to more overfitting than a larger dropout rate (0.3) for NN2. It implies that stronger regularization (e.g., increase dropout rate, adding L2 regularization) may be needed to reduce overfitting.

TABLE 5. Percentage of samples with large GEFS test loss (above the 90th percentile) improved by NN1 and NN2 at each forecast lead time.

Lead time (day)	NN1	NN2
2	69%	62%
4	95%	94%
5	96%	95%
6	98%	98%

Since the grid search was only performed on a limited combination and value of hyperparameters, and the same grid search was applied for NN1 and NN2, it is unfortunate that this grid search caused consistent overfitting for NN2 but not NN1.

7. Summary

This study presents NN models for postprocessing TC track forecasts made by GEFS in the medium range (up to 6 days) for the WNP. The present NN models consist of a combination of three commonly used architectures: MLP, CNN, and RNN (LSTM), and was trained using dynamical model forecast as input data and was fitted to observation (best track data). The two NN models, in which one of them was incorporated with a shortcut connection, showed promising results. It was found that the use of a shortcut connection led to better performance under various training conditions. Extending the forecast length beyond six days is possible for the current algorithms, but the number of training samples is expected to decrease, and generalization of data can be more difficult.

The present NN models provide a prototype for encouraging further developments in TC track forecasts in the medium range, as seen from the evaluation of model performance with the test dataset. More than half of the samples in the test dataset had reduced track error after postprocessed by NN2, and the mean track errors for all but the last lead time were also smaller than those of GEFS. This indicates the potential for future operational applications if the model can be better

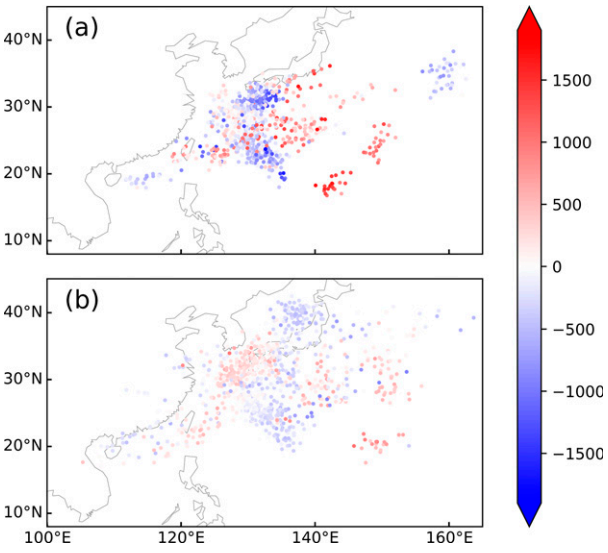

FIG. 13. TC location of samples in test dataset on day 6 predicted by (a) NN1 and (b) NN2. The color of each circle indicates the difference in track error between a NN model and GEFS (former minus latter; km).

calibrated. Figure 5 demonstrates the possibility of further improvement of NN2 if heavier regularization is imposed to reduce overfitting. Moreover, the NN is a convenient tool for TC forecasting. Although training an NN model can be time-consuming depending on its complexity, it is only required to be trained once. Retraining of a model can be performed before the next typhoon season, if additional data from the previous season can be included. Moreover, it takes very little time for making a real-time forecast (in addition to downloading the input forecast data of a dynamical model), which is negligible compared to that of running a regional (dynamical) model downscaled from a global model.

Furthermore, there are still some issues to be investigated for the development of a more comprehensive model: 1) Model output across multiple versions: it is reasonable to assume that the systematic error of a model should change with model upgrade, so the simulation of the state of the same atmosphere is different. It is unknown whether the difference in systematic error contained in multiple GEFS model versions is significant enough to impact the learning of the NN model. If not, more data or samples are available for training; 2) forecast model: since both observation and forecast data are required for a model function as a postprocessing method, data augmentation is difficult if samples for training are insufficient. Dynamical models such as the WRF Model can be used to generate training samples, because model outputs are physically constrained such that resembling observation is possible. A similar approach has also been used in other studies (e.g., Kim et al. 2018; Ham et al. 2019), which increases the number of samples and their variety.

Acknowledgments. This work was funded by the Korea Meteorological Administration Research and Development Program under Grant KMI2020-00610. We thank Mr. Seungwoo Yoo for providing the code for calculation of along- and cross-track error.

Data availability statement. The best track data of the RSMC are available at https://www.jma.go.jp/jma/jma-eng/jma-center/rsmc-hp-pub-eg/besttrack.html. The operational GEFS forecast data are available at https://www.ncdc.noaa.gov/has/HAS.FileAppRouter?datasetname=GFS&subqueryby=STATION&applname=&outdest=FILE, but the GEFS reforecast dataset used in this study is no longer available as of 23 September 2020.

APPENDIX A

Details of TC Vortex Detection and Tracking Algorithm

The detection and tracking procedures are as follows:

1) Analysis time:
   - In the model analysis, the TC center was determined by initiating a search from the observed TC position within a radius of 220 km for the grid point that contained the highest value of RV850. The nearest grid point with a local minimum MSLP within a 450 km radius from the highest RV850 point was identified.
   - A closed isobar check was performed at a radius of 300 km from the grid point with the aforementioned local minimum MSLP. If the check fails, this low pressure is not regarded as a TC vortex.
   - When a vortex could not be detected, the analyzed (first) model TC position was set to the observed TC position.
   - To obtain a more precise TC center, a $5^\circ \times 5^\circ$ grid centered on the previously determined grid point at the lowest MSLP was interpolated to a $10^\circ \times 10^\circ$ grid.
2) Forecast position after six hours:
   - The search for the second TC position was similar to that for the analyzed position. It started from the observed TC position within a 450-km search radius.
3) Forecast position after 12 hours and beyond:
   - The first estimate was defined by extrapolating the forecast track for the previous two time steps. Locating the TC position was the same as in step 2 but was started at the first estimated position. The search radius was 450 km.
   - When a vortex could not be detected in the above step, the search was restarted using the preceding model's TC center position as the starting point with a 150-km search radius; this avoids missing a vortex when a model vortex suddenly accelerates or decelerates to be far from the original estimated position.

For a lead time longer than six hours, tracking of a model TC vortex could be continued if the following criteria were met:

1) For vortices detected at latitudes higher than 20°N, RV850 is greater than 5×10^{-5}, and 1×10^{-5} s^{-1} otherwise. This threshold was made to be region dependent as

TABLE C1. Predictors used for development of CLIPER model in the meridional and zonal direction, obtained from Aberson and Sampson (2003). LAT: initial latitude; LON: initial longitude; INT: initial intensity (maximum sustained wind speed); DAY: day of year at initial time; *U*: initial zonal moving speed; *V*: initial meridional moving speed.

Meridional				
V	*U*	INT × *V*	*U* × *V*	DAY × DAY
LON × DAY	INT × *U*	LAT × INT	LON × LON	LON
INT × INT	LAT × LAT	LAT × LON	DAY	*U* × *U*
LAT × U	LON × INT			
Zonal				
U	LAT	INT × *V*	INT × *U*	*U* × *U*
LAT × LON	LON × *V*	LON × *U*	LAT × LAT	LON × LON

in certain cases a model TC weakened at an early stage, occurring at lower latitudes.

2) The model vortex is located south of 45°N.
3) The MSLP decrease of a model TC from a previous time step is less than or equal to 20 hPa; it prevents false identification of a vortex center when two vortices in the model are in proximity.
4) The separation between the model TC center at the current and previous time steps is less than or equal to 500 km.

APPENDIX B

Explanation of Hyperparameters

Training of the hybrid NN was carried out for numerous cycles (*epoch*), and in one epoch, the entire training dataset was worked through. The *batch size* is the number of samples processed before the NN model is updated, and the number of batches is roughly the number of training samples divided by the batch size. Therefore, in one epoch, the model is updated multiple times.

Learning rate controls the speed of model weights change in training. When too low, the training can be more reliable, but it may take a much longer time to reach the optimal solution, or the model can become trapped in a local minimum of a loss function landscape. Conversely, when too high, the training can proceed much faster, but the model may converge toward a suboptimal solution.

Dropout rate is a probability that determines the proportion of nodes to be ignored randomly during the training stage. Dropout is commonly used as a regularization method to avoid overfitting. *L2 regularization*, which adds a penalty term to the loss function, can also be used for the same purpose.

Optimizers are algorithms that modify the weights and learning rate of NN models to reduce losses more efficiently. Gradient descent, Adagrad, and Adam are examples of commonly used optimizers. Adam is a type of stochastic gradient descent method with an adaptive learning rate, where the loss function is optimized by approximating the gradient with a randomly selected subset of data, and learning rate is made to decay with time.

Kernel initializer determines the method that randomly initializes the weights in a NN. The weights can be assigned values with different types of distributions (e.g., normal, uniform distribution, etc.). Training can be more efficient if the initializer is appropriately selected.

Kernel size and *number of kernels* are both associated with the CNN. The kernel size defines the size of a 2D window that slides over an input image during convolution. The number of kernels indicates the number of spatial features to be extracted/detected.

APPENDIX C

CLIPER Baseline Model

Aberson and Sampson (2003) used CLIPER for TC track forecasts in the WNP up to five days. The predictands are the displacement in zonal and meridional direction from the initial time at each forecast lead time. In our case, therefore, there are eight regression equations in total (two directions × four lead times). The predictors used in Aberson and Sampson (2003) were also applied in the development of the baseline model. They are listed in Table C1. The baseline model is developed using TCs formed in the period 1977–2019. Since the RSMC Tokyo best track dataset is used, utilization of TC data before 1977 is not possible because the maximum sustained wind speed was not recorded. The training and test datasets include samples of TCs that survive for at least six days in the WNP. The latter one is the same as the one for evaluation of NN models, while the former one has the remaining samples.

REFERENCES

Aberson, S. D., and C. R. Sampson, 2003: On the predictability of tropical cyclone tracks in the northwest Pacific basin. *Mon. Wea. Rev.*, **131**, 1491–1497, https://doi.org/10.1175/1520-0493(2003)131<1491:OTPOTC>2.0.CO;2.

Albawi, S., T. A. Mohammed, and S. Al-Zawi, 2017: Understanding of a convolutional neural network. *Proc. 2017 Int. Conf. on Engineering and Technology*, Antalya, Turkey, IEEE, 1–6, https://doi.org/10.1109/ICEngTechnol.2017.8308186.

Alemany, S., J. Beltran, A. Perez, and S. Ganzfried, 2019: Predicting hurricane trajectories using a recurrent neural network. *33rd AAAI Conf. on Artificial Intelligence*, Honolulu, HI, AAAI, 468–475.

Bassill, N. P., 2014: Accuracy of early GFS and ECMWF Sandy (2012) track forecasts: Evidence for a dependence on cumulus parameterization. *Geophys. Res. Lett.*, **41**, 3274–3281, https://doi.org/10.1002/2014GL059839.

Bauer, P., A. Thorpe, and G. Brunet, 2015: The quiet revolution of numerical weather prediction. *Nature*, **525**, 47–55, https://doi.org/10.1038/nature14956.

Bezdek, J. C., 1981: *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, 272 pp.

Bickel, S., M. Brückner, and T. Scheffer, 2007: Discriminative learning for differing training and test distributions. *ACM Int. Conf. Proc. Series*, Corvallis, OR, ICML, 81–88, https://doi.org/10.1145/1273496.1273507.

Bishop, C. M., 1994: Neural networks and their applications. *Rev. Sci. Instrum.*, **65**, 1803–1832, https://doi.org/10.1063/1.1144830.

Chan, J. C. L., 2005: The physics of tropical cyclone motion. *Annu. Rev. Fluid Mech.*, **37**, 99–128, https://doi.org/10.1146/annurev.fluid.37.061903.175702.

Chandra, R., K. Dayal, and N. Rollings, 2015: Application of cooperative neuro-evolution of Elman recurrent networks for a two-dimensional cyclone track prediction for the South Pacific region. *2015 Int. Joint Conf. on Neural Networks*, Killarney, Ireland, IEEE, 1–8, https://doi.org/10.1109/IJCNN.2015.7280394.

Chen, B.-F., B. Chen, H.-T. Lin, and R. L. Elsberry, 2019: Estimating tropical cyclone intensity by satellite imagery utilizing convolutional neural networks. *Wea. Forecasting*, **34**, 447–465, https://doi.org/10.1175/WAF-D-18-0136.1.

Cheung, H. M., C.-H. Ho, M. Chang, D. Kim, J. Kim, and W. Choi, 2021: Development of a track-pattern-based medium-range tropical cyclone forecasting system for the western North Pacific. *Wea. Forecasting*, **36**, 1505–1518, https://doi.org/10.1175/WAF-D-20-0102.1.

Cloud, K. A., B. J. Reich, C. M. Rozoff, S. Alessandrini, W. E. Lewis, and L. Delle Monache, 2019: A feed forward neural network based on model output statistics for short-term hurricane intensity prediction. *Wea. Forecasting*, **34**, 985–997, https://doi.org/10.1175/WAF-D-18-0173.1.

Daliri, M. R., and M. Fattan, 2011: Improving the generalization of neural networks by changing the structure of artificial neuron. *Malays. J. Comput. Sci.*, **24**, 195–204.

Elsberry, R. L., W. M. Frank, G. J. Holland, J. D. Jarrell, and R. L. Southern, 1987: *A Global View of Tropical Cyclones*. R. L. Elsberry, Ed., Naval Postgraduate School, 185 pp.

Fiorino, M., J. S. Goerss, J. J. Jensen, and J. E. J. Harrison, 1993: An evaluation of the real-time tropical cyclone forecast skill of the Navy Operational Global Atmospheric Prediction System in the western North Pacific. *Wea. Forecasting*, **8**, 3–24, https://doi.org/10.1175/1520-0434(1993)008<0003:AEOTRT>2.0.CO;2.

Fovell, R. G., K. L. Corbosiero, and H.-C. Kuo, 2009: Cloud microphysics impact on hurricane track as revealed in idealized experiments. *J. Atmos. Sci.*, **66**, 1764–1778, https://doi.org/10.1175/2008JAS2874.1.

Gardner, M. W., and S. R. Dorling, 1998: Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences. *Atmos. Environ.*, **32**, 2627–2636, https://doi.org/10.1016/S1352-2310(97)00447-0.

Gerrity, J. P., R. D. McPherson, and P. D. Polger, 1972: On the efficient reduction, of truncation error in numerical weather prediction models. *Mon. Wea. Rev.*, **100**, 637–643, https://doi.org/10.1175/1520-0493(1972)100<0637:OTEROT>2.3.CO;2.

Giffard-Roisin, S., M. Yang, G. Charpiat, C. K. Bonfanti, B. Kgl, and C. Monteleoni, 2020: Tropical cyclone track forecasting using fused deep learning from aligned reanalysis data. *Front. Big Data*, **3**, https://doi.org/10.3389/fdata.2020.00001.

Grams, C. M., L. Magnusson, and E. Madonna, 2018: An atmospheric dynamics perspective on the amplification and propagation of forecast error in numerical weather prediction models: A case study. *Quart. J. Roy. Meteor. Soc.*, **144**, 2577–2591, https://doi.org/10.1002/qj.3353.

Guyon, I., and A. Elisseeff, 2003: An introduction to variable and feature selection. *J. Mach. Learn. Res.*, **3**, 1157–1182.

Ham, Y.-G., J.-H. Kim, and J.-J. Luo, 2019: Deep learning for multi-year ENSO forecasts. *Nature*, **573**, 568–572, https://doi.org/10.1038/s41586-019-1559-7.

Hamill, T. M., J. S. Whitaker, M. Fiorino, and S. G. Benjamin, 2011: Global ensemble predictions of 2009's tropical cyclones initialized with an ensemble Kalman filter. *Mon. Wea. Rev.*, **139**, 668–688, https://doi.org/10.1175/2010MWR3456.1.

——, G. T. Bates, J. S. Whitaker, D. R. Murray, M. Fiorino, T. J. Galarneau, Y. Zhu, and W. Lapenta, 2013: NOAA's second-generation global medium-range ensemble reforecast dataset. *Bull. Amer. Meteor. Soc.*, **94**, 1553–1565, https://doi.org/10.1175/BAMS-D-12-00014.1.

Heming, J. T., 2017: Tropical cyclone tracking and verification techniques for Met Office numerical weather prediction models. *Meteor. Appl.*, **24** (1), 1–8, https://doi.org/10.1002/met.1599.

Ho, C. H., J. J. Baik, J. H. Kim, D. Y. Gong, and C. H. Sui, 2004: Interdecadal changes in summertime typhoon tracks. *J. Climate*, **17**, 1767–1776, https://doi.org/10.1175/1520-0442(2004)017<1767:ICISTT>2.0.CO;2.

——, I. Park, J. Kim, and J. B. Lee, 2022: PM2.5 Forecast in Korea using the Long Short-Term Memory (LSTM) Model. *Asia-Pac. J. Atmos. Sci.*, https://doi.org/10.1007/s13143-022-00293-2, in press.

Hochreiter, S., and J. Schmidhuber, 1997: Long short-term memory. *Neural Comput.*, **9**, 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.

Hu, L., E. A. Ritchie, and J. S. Tyo, 2020: Short-term tropical cyclone intensity forecasting from satellite imagery based on the deviation angle variance technique. *Wea. Forecasting*, **35**, 285–298, https://doi.org/10.1175/WAF-D-19-0102.1.

Jankov, I., W. A. Gallus, M. Segal, B. Shaw, and S. E. Koch, 2005: The impact of different WRF Model physical parameterizations and their interactions on warm season MCS rainfall. *Wea. Forecasting*, **20**, 1048–1060, https://doi.org/10.1175/WAF888.1.

Jin, J., M. Li, and L. Jin, 2015: Data normalization to accelerate training for linear neural net to predict tropical cyclone tracks. *Math. Probl. Eng.*, **2015**, 931629, https://doi.org/10.1155/2015/931629.

Jolliffe, I. T., and D. B. Stephenson, 2011: *Forecast Verification: A Practitioner's Guide in Atmospheric Science*. 2nd ed. Wiley, 296 pp.

Katsube, K., and M. Inatsu, 2016: Response of tropical cyclone tracks to sea surface temperature in the western North Pacific. *J. Climate*, **29**, 1955–1975, https://doi.org/10.1175/JCLI-D-15-0198.1.

Kim, H.-S., J.-H. Kim, C.-H. Ho, and P.-S. Chu, 2011: Pattern classification of typhoon tracks using the fuzzy *c*-means clustering method. *J. Climate*, **24**, 488–508, https://doi.org/10.1175/2010JCLI3751.1.

Kim, K., D. Cha, and J. Im, 2020: Improvement of tropical cyclone track forecast over the western North Pacific using a machine learning method. M.S. thesis, Dept. of Urban and Environmental Engineering, Graduate School of the Ulsan

National Institute of Science and Technology, 56 pp., https://scholarworks.unist.ac.kr/handle/201301/31769.

Kim, S., J. K. Minho, and S. Song, 2018: DeepTC: ConvLSTM network for trajectory prediction of tropical cyclone using spatiotemporal atmospheric simulation data. *32nd Conf. on Neural Information Processing Systems*, Montreal, Canada, Association for Computing Machinery, https://openreview.net/pdf?id=HJlCVoPAF7.

Kim, S., H. Kim, J. Lee, S. Yoon, S. E. Kahou, K. Kashinath, and Prabhat, 2019: Deep-hurricane-tracker: Tracking and forecasting extreme climate events. *2019 IEEE Winter Conf. on Applications of Computer Vision*, Waikoloa, HI, IEEE, 1761–1769, https://doi.org/10.1109/WACV.2019.00192.

Kumler-Bonfanti, C., J. Stewart, D. Hall, and M. Govett, 2020: Tropical and extratropical cyclone detection using deep learning. *J. Appl. Meteor. Climatol.*, **59**, 1971–1985, https://doi.org/10.1175/JAMC-D-20-0117.1.

Lau, Y.-Y., K. Y. Chau, M. A. Dulebenets, Y. M. Tang, J. Guan, and T. K. Ying, 2021: Tropical cyclone research in Asia: Hong Kong and Macao. *IOP Conf. Ser. Earth Environ. Sci.*, **690**, 012044, https://doi.org/10.1088/1755-1315/690/1/012044.

Leonardo, N. M., and B. A. Colle, 2017: Verification of multimodel ensemble forecasts of North Atlantic tropical cyclones. *Wea. Forecasting*, **32**, 2083–2101, https://doi.org/10.1175/WAF-D-17-0058.1.

Leray, P., and P. Gallinari, 1999: Feature selection with neural networks. *Behaviormetrika*, **26**, 145–166, https://doi.org/10.2333/bhmk.26.145.

Lorenz, E. N., 1963: Deterministic nonperiodic flow. *J. Atmos. Sci.*, **20**, 130–141, https://doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

Marzban, C., 2003: Neural networks for postprocessing model output: ARPS. *Mon. Wea. Rev.*, **131**, 1103–1111, https://doi.org/10.1175/1520-0493(2003)131<1103:NNFPMO>2.0.CO;2.

McGovern, A., R. Lagerquist, D. J. Gagne, G. E. Jergensen, K. L. Elmore, C. R. Homeyer, and T. Smith, 2019: Making the black box more transparent: Understanding the physical implications of machine learning. *Bull. Amer. Meteor. Soc.*, **100**, 2175–2199, https://doi.org/10.1175/BAMS-D-18-0195.1.

Mendelsohn, R., K. Emanuel, S. Chonabayashi, and L. Bakkensen, 2012: The impact of climate change on global tropical cyclone damage. *Nat. Climate Change*, **2**, 205–209, https://doi.org/10.1038/nclimate1357.

Olander, T., A. Wimmers, C. Velden, and J. P. Kossin, 2021: Investigation of machine learning using satellite-based advanced Dvorak technique analysis parameters to estimate tropical cyclone intensity. *Wea. Forecasting*, **36**, 2161–2186, https://doi.org/10.1175/WAF-D-20-0234.1.

Otkin, J. A., and T. J. Greenwald, 2008: Comparison of WRF Model-simulated and MODIS-derived cloud data. *Mon. Wea. Rev.*, **136**, 1957–1970, https://doi.org/10.1175/2007MWR2293.1.

Park, D.-S. R., C.-H. Ho, J. Kim, K. Kang, and C. C. Nam, 2016: Highlighting socioeconomic damages caused by weakened tropical cyclones in the Republic of Korea. *Nat. Hazards*, **82**, 1301–1315, https://doi.org/10.1007/s11069-016-2244-x.

Plu, M., 2011: A new assessment of the predictability of tropical cyclone tracks. *Mon. Wea. Rev.*, **139**, 3600–3608, https://doi.org/10.1175/2011MWR3627.1.

Rabuñal, J. R., and J. Dorado, 2006: *Artificial Neural Networks in Real-Life Applications.* Idea Group Publishing, 375 pp.

Rasp, S., and S. Lerch, 2018: Neural networks for postprocessing ensemble weather forecasts. *Mon. Wea. Rev.*, **146**, 3885–3900, https://doi.org/10.1175/MWR-D-18-0187.1.

Rodwell, M. J., and Coauthors, 2013: Characteristics of occasional poor medium-range weather forecasts for Europe. *Bull. Amer. Meteor. Soc.*, **94**, 1393–1405, https://doi.org/10.1175/BAMS-D-12-00099.1.

Santos, M. S., J. P. Soares, P. H. Abreu, H. Araujo, and J. Santos, 2018: Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches. *IEEE Comput. Intell. Mag.*, **13**, 59–76, https://doi.org/10.1109/MCI.2018.2866730.

Smith, M., and R. Toumi, 2021: Using video recognition to identify tropical cyclone positions. *Geophys. Res. Lett.*, **48**, e2020GL091912, https://doi.org/10.1029/2020GL091912.

Sola, J., and J. Sevilla, 1997: Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Trans. Nucl. Sci.*, **44**, 1464–1468, https://doi.org/10.1109/23.589532.

Tsui, T. L., and R. J. Miller, 1988: Evaluation of western North Pacific tropical cyclone objective forecast aids. *Wea. Forecasting*, **3**, 76–85, https://doi.org/10.1175/1520-0434(1988)003<0076:EOWNPT>2.0.CO;2.

Wang, H., M. Xu, A. Onyejuruwa, Y. Wang, S. Wen, A. E. Gao, and Y. Li, 2019: Tropical cyclone damages in Mainland China over 2005–2016: Losses analysis and implications. *Environ. Dev. Sustainability*, **21**, 3077–3092, https://doi.org/10.1007/s10668-019-00481-7.

Wang, Y., and G. J. Holland, 1996: Tropical cyclone motion and evolution in vertical shear. *J. Atmos. Sci.*, **53**, 3313–3332, https://doi.org/10.1175/1520-0469(1996)053<3313:TCMAEI>2.0.CO;2.

Wen, J., C.-N. Yu, and R. Greiner, 2014: Robust learning under uncertain test distributions: Relating covariate shift to model misspecification. *31st Int. Conf. Machine. Learning*, Beijing, China, JMLR, 631–639, http://proceedings.mlr.press/v32/wen14.pdf.

Wimmers, A., C. Velden, and J. H. Cossuth, 2019: Using deep learning to estimate tropical cyclone intensity from satellite passive microwave imagery. *Mon. Wea. Rev.*, **147**, 2261–2282, https://doi.org/10.1175/MWR-D-18-0391.1.

Xu, W., K. Balaguru, A. August, N. Lalo, N. Hodas, M. DeMaria, and D. Judi, 2021: Deep learning experiments for tropical cyclone intensity forecasts. *Wea. Forecasting*, **36**, 1453–1470, https://doi.org/10.1175/WAF-D-20-0104.1.

Yan, Z., X. Ge, and B. Guo, 2017: Simulated sensitivity of tropical cyclone track to the moisture in an idealized monsoon gyre. *Dyn. Atmos. Oceans*, **80**, 173–182, https://doi.org/10.1016/j.dynatmoce.2017.10.008.

Zhang, T., W. Lin, Y. Lin, M. Zhang, H. Yu, K. Cao, and W. Xue, 2019: Prediction of tropical cyclone genesis from mesoscale convective systems using machine learning. *Wea. Forecasting*, **34**, 1035–1049, https://doi.org/10.1175/WAF-D-18-0201.1.

Zheng, X., Y. H. Duan, and H. Yu, 2007: Dynamical effects of environmental vertical wind shear on tropical cyclone motion, structure, and intensity. *Meteor. Atmos. Phys.*, **97**, 207–220, https://doi.org/10.1007/s00703-006-0253-0.