Earth and Space Science

RESEARCH ARTICLE
10.1029/2021EA002168

Key Points:
• Learning rate controls the forecasting accuracy of hourly typhoon rainfall
• The convolutional neural network is unable to forecast hourly typhoon rainfall in terms of too-large and too-small learning rate
• The convolutional neural network shows more accurate forecasts than the existing models

Supporting Information: Supporting Information may be found in the online version of this article.

Correspondence to: Y. Li, liyubin@muist.edu.cn

Citation:
Uddin, M. J., Li, Y., Sattar, M. A., Nasrin, Z. M., & Lu, C. (2022). Effects of learning rates and optimization algorithms on forecasting accuracy of hourly typhoon rainfall: Experiments with convolutional neural network. Earth and Space Science, 9, e2021EA002168. https://doi.org/10.1029/2021EA002168

Received 5 DEC 2021
Accepted 6 FEB 2022

Author Contributions:
Conceptualization: Yubin Li
Formal analysis: Md. Jalal Uddin
Investigation: Md. Jalal Uddin
Methodology: Yubin Li
Software: Md. Jalal Uddin, Zahan Most. Nasrin
Supervision: Yubin Li
Visualization: Md. Jalal Uddin
Writing – original draft: Md. Jalal Uddin
Writing – review & editing: Md. Abdus Sattar, Chunsong Lu

Abstract
The current study used seven optimization algorithms such as stochastic gradient descent (SGD), root mean square propagation (RMSprop), adaptive grad (AdaGrad), adaptive delta (AdaDelta), adaptive moment estimation (Adam), adaptive maximum (Adamax), and Nesterov-accelerated adaptive moment estimation (Nadam) and eight learning rates (1, 0.1, 0.01, 0.001, 0.0001, le-05, le-06, and le-07) to investigate the effects of these learning rates and optimizers on the forecasting performance of the convolutional neural network (CNN) model to forecast hourly typhoon rainfall. The model was developed using antecedent hourly typhoon rainfall within a 500 km radius from each typhoon center. Results showed that too-large and too-small learning rates would result in the inability of the model to learn anything to forecast hourly typhoon rainfall. The CNN model showed the best performance for learning rates of 0.1, 0.01, and 0.001 to forecast hourly typhoon rainfall. For long-lead-time forecasting (1–6 hr), the CNN model with SGD, RMSprop, AdaGrad, AdaDelta, Adam, Adamax, Nadam optimizers and learning rates of 0.1, 0.01, and 0.001 showed more accurate forecasts than the existing models. Therefore, this study recommends that future work may consider the CNN model as an alternative to the existing model for disaster warning systems.

1. Introduction
Accurate typhoon rainfall forecasting has become a great concern among the scientific community because typhoons' heavy rainfall results in coastal flooding that leads to loss of lives and properties. Typhoon In-Fa, a recent typhoon in China, for example, affected millions of people in Henan province by its heavy rainfall and floods (Guardian, 2021; Jazeera, 2021; NASA, 2021; News, 2021; The Washington Post, 2021). For disaster mitigation, therefore, accurate typhoon rainfall forecasting plays a vital role in the warning system (Lin et al., 2009, 2013). However, it is a challenging task to obtain accurate forecasts of typhoon rainfall. To solve this problem, the deep learning (DL) model can be an attractive alternative technique to forecast typhoon rainfall. Previous studies (e.g., Huang et al., 2018; Lin & Chen, 2005; Lin & Wu, 2009; Wei & Chou, 2020) based on the DL model have been proven to forecast typhoon rainfall more accurately than the model developed by the traditional neural network technique. It can be seen that the application of the DL model has been increasing in hydrological forecasting (Lin & Chen, 2004; Partal & Cigizoglu, 2008; Pulido-Calvo & Portela, 2007; Valverde Ramírez et al., 2005) because the DL model has powerful capability to model nonlinear systems without the need to make any assumptions (Lin & Wu, 2009).

Methods and results of previous studies related to typhoon rainfall forecasting are discussed herein. Lin and Chen (2005) and Lin and Wu (2009) were aimed to forecast typhoon rainfall within 1-hr lead-time. Lin's team in 2005 used typhoon features and spatial rainfall information as input to the artificial neural network (ANN) model, while Lin's group in 2009 used self-organizing map (SOM) and multilayer perceptron network (MLPN) to cluster typhoon rainfall and forecast typhoon rainfall respectively. Lin and Chen (2005) showed that the ANN model could only yield accurate forecasts 1 hr ahead of forecasting, whilst Lin and Wu (2009) found that their models forecasted typhoon rainfall more accurately than the model developed by the traditional neural network technique. Huang et al. (2018) used stepwise regression (SRM) and local linear embedding (LLE) algorithms to construct the input of the fuzzy neural network (FNN) model to forecast typhoon rainfall within a 1-day lead-time. The authors reported that the FNN-LLE model performed better than the FNN-SRM model to forecast typhoon rainfall. Finally, Wei and Chou (2020) applied the deep neural network (DNN) model to evaluate typhoon rainfall...
forecasting accuracy within 1–6 hr ahead of forecasting. They showed that this model can solve challenges in terms of real-time typhoon rainfall forecasting with high accuracy.

Optimization algorithms help the DL model to minimize the loss and maximize the efficiency of outputs. On the other hand, the learning rate is a tuning parameter of optimization algorithms, and it controls how much to change the DL model in response to the estimated error each time the model weights are updated (Brownlee, 2020). The effects of optimization algorithms and learning rates on the forecasting performance of DL models were confirmed by previous studies (e.g., Bengio, 2012; R. Chen et al., 2019; Goodfellow et al., 2016; Prasetya & Djamal, 2019; Saputri et al., 2020). These studies reported that optimization algorithms and learning rates determined the accuracy of the DL model significantly. A long training process would result from a small value of learning rate, while too fast or an unstable training process would result from a large value of learning rate (Brownlee, 2018a, 2018b, 2020; Goodfellow et al., 2016). To solve this problem, proper configuration of the DL model is necessary.

However, the selecting procedure or description of the learning rate is absent in the typhoon rainfall forecasting model based on the DL model (Huang et al., 2018; Lin & Chen, 2005; Lin & Wu, 2009; Wei & Chou, 2020). Lin and Chen (2005) selected a learning rate as 0.01 for their study without describing the procedure or method. On the other hand, Lin and Wu (2009) and Huang et al. (2018) just mentioned the learning rate in Equation 3 and flowchart 2 respectively (see their papers). While Wei and Chou (2020) used the trial and error method to select optimal learning rates. However, there was no description of how they tested the variation of learning rates regarding forecasting performance.

More importantly, optimization algorithms such as stochastic gradient descent (SGD), root mean square propagation (RMSprop), adaptive grad (AdaGrad), adaptive delta (AdaDelta), adaptive moment estimation (Adam), adaptive maximum (Adamax), and Nesterov-accelerated adaptive moment estimation (Nadam) are absent in the typhoon rainfall forecasting model based on the DL model (Huang et al., 2018; Lin & Chen, 2005; Lin & Wu, 2009; Wei & Chou, 2020). These optimization algorithms have successfully been applied in rainfall forecasting (Barrera-Animas et al., 2021; Fadhilah et al., 2021; Manoj & Ananth, 2020; Prasetya & Djamal, 2019; Sari et al., 2020; Zhang et al., 2018), spatial prediction of landslides (Bui et al., 2019), wind speed and wind direction forecasting (Puspita Sari et al., 2020; Saputri et al., 2020), evapotranspiration forecasting (Walls et al., 2020), run-off forecasting (Nath et al., 2021), air quality index prediction (H. He & Luo, 2020), river stage, flash flood susceptibility and streamflow forecasting (Hitokoto et al., 2017; Rahimzad et al., 2021; Tien Bui et al., 2020), water demand forecasting (Koo et al., 2021), temperature and global solar radiation prediction (Del & Starchenko, 2021; Ghimire et al., 2019).

Therefore, this paper aims to examine the effects of learning rate and optimization algorithms on the forecasting performance of the convolutional neural network (CNN) model to obtain effective hourly rainfall forecasts. The reason to use the CNN model is that this model has successfully been applied to forecast typhoon genesis (R. Chen et al., 2019; Matsuoka et al., 2018), typhoon track (Giffard-Roisin et al., 2019), and typhoon intensity (B. F. Chen et al., 2019). Moreover, typhoons and their precursors in the Northwestern Pacific Ocean were successfully detected by the CNN model (Matsuoka et al., 2018). This model has overcome the problem of overfitting by MLPN using some regularization methods such as dropout or skipped connections.

In this paper, we present a straightforward approach to adjust the learning rate for optimization algorithms during training the model. This process is known as grid search cross-validation, where we can provide a list of optimizers and ranges of learning rate to see the changes of loss over training epochs (see Section 2.4 for more details). The rest of the paper is organized as follows. Data and methods are described in Section 2. The third section represents the results and discussions. Finally, this paper is wrapped up with a conclusion.

2. Data and Methods

2.1. Data Quality Control

2.1.1. Select Data Within a 500 km Radius From the Typhoon Center

The Northwestern Pacific Ocean (NWPO) has been selected for this study because this area was affected by heavy rainfall and coastal flooding during typhoons events. The typhoon best track dataset was collected from the China Meteorological Administration (http://tcdata.typhoon.org.cn; Lu et al. [2021]; Ying et al. [2014]).
spatial positions together with the best tracks of the typhoons in the NWPO during 2019 are shown in Figure 1. Most of the typhoons were tended to form in the eastern part of the Philippine Sea and recurved and moved eastward. This study used Global Precipitation Measurement (GPM) data from the National Aeronautics and Space Administration (NASA; Huffman et al. [2014]).

For all typhoons in 2019 in the NWPO, the maximum rainfall was mainly located within a 500 km radius from each typhoon center (Figure 2). The amount of rainfall declines with increasing distance from the typhoon center. Less than 1 mm hr\(^{-1}\) azimuthally averaged rainfall was found in most cases from 500 to 1,000 km distance. Therefore, our study used antecedent typhoon rainfall within a 500 km distance from the typhoon center as input for the CNN model.

Similarly, to forecast typhoon rainfall, some studies applied this distance as input for statistical models (Kim et al., 2020; Lonfat et al., 2007; Tuleya et al., 2007). Moreover, several studies have defined typhoon rainfall events that occurred within a 500 km radius from the typhoon center to show the contribution of typhoon rainfall to total rainfall (F. J. Chen & Fu, 2015; Dare et al., 2012; Hernández Ayala & Matyas, 2018; Jiang & Zipser, 2010; Larson et al., 2005; Lau et al., 2008; Lavender & Abbs, 2013; Lee et al., 2010; Nogueira & Keim, 2011; Prat & Nelson, 2013a, 2013b; Schreck et al., 2011; Uddin et al., 2019).

Another reason to use a 500 km radius from the typhoon center to select typhoon rainfall is that the range of the typhoon primary wind circulation region (80–400 km radius) and of the curved typhoon cloud shield (550–600 km radius) is consistent with this distance (Englehart & Douglas, 2001).

2.1.2. Check for the Stationarity of Hourly Total Typhoons' Rainfall

The stationarity of a time series indicates that its mean, variance, and covariance do not change over time (Kwiatkowski et al., 1992). Therefore, a time series is not considered stationary if seasonal effects or white noise exist in it. The seasonal pattern indicates a repeated pattern in the time series. On the other hand, a time series is a white noise if the mean value is nearly zero, and the standard deviation is nearly one. In addition, the values of a non-stationary time series do not correlate with their lag values.

Kwiatkowski et al. (1992) reported that seasonality and white noise could affect the forecasting model. A recent study by Prabhakaran (2019) showed that the model could forecast easier and the outputs were more reliable when the time series was stationary. Therefore, it is necessary to check whether a time series is stationary or non-stationary. For data quality control, we used all typhoon rainfall events (3,768 hr) to check for stationarity (Figure 3). Please note that 10 typhoon events (2,328 hr) were used for training the model, and five typhoon events (1440 hr) were used for testing the model outputs (see Section 2.4 for more details). We can check whether
a time series is stationary or non-stationary in three ways: a) decompose a time series into its components, b) statistical test, and c) check for white noise.

1. Decompose a time series into its components.

In this study, we used the additive decompose method (i.e., additive method = observed + trend + seasonal pattern + residual) to decompose hourly total typhoons' rainfall into three components such as trend, seasonal pattern, and residual. Results showed that there were no seasonal patterns and errors in hourly total typhoons' rainfall (Figure 3). This output suggests that the entire data is a stationary time-series.

2. Augmented Dickey-Fuller (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test.

ADF (Dickey & Fuller, 1979) and KPSS (Kwiatkowski et al., 1992) are the unit root tests. The unit root is a characteristic of a time series which makes it non-stationary. For the ADF test, if the test statistic and p-value are smaller than the corresponding critical value and significant levels (0.05 and 0.01) respectively, the null
hypothesis is rejected. In this case, the alternate hypothesis is accepted, and the ADF test provides evidence that
the time series is stationary (Table 1). For the KPSS test, the null hypothesis is accepted if the test statistic is
smaller than the corresponding critical value and the p-value is greater than significant levels (0.05 and 0.01). In
this case, the KPSS test provides evidence that the time series is stationary. A detailed description of ADF and
KPSS tests can be found in articles of Dickey and Fuller (1979) and Kwiatkowski et al. (1992). In this study, we
used ‘statsmodels’ in Python (Seabold & Perktold, 2010) to calculate ADF and KPSS.

Results showed that test statistics (−11.01) for the ADF test is less than the critical values (−3.432, −2.862, and
−2.567) at 1%, 5%, and 10% confidence levels. Similarly, p-value (0.000) is less than significant levels (0.05
and 0.01; Table 2). For KPSS test, the critical values (0.739, 0.463, and 0.347) at 1%, 5% and 10% confidence
levels are greater than test statistics (0.329). Likewise, the p-value is greater than the significant levels (0.05 and
0.01). Finally, both ADF and KPSS tests confirmed that hourly total typhoons’ rainfall is a stationary time series
(Table 2).

3. Check for white noise in hourly total typhoons’ rainfall.

Hourly total typhoons’ rainfall will be considered a stationary time series if the mean value is non-zero, and the
standard deviation is not one. In addition, the values of hourly total typhoons’ rainfall should be correlated with
lag values. Here, this study used summary statistics and autocorrelation to check the white noise in hourly total
typhoons’ rainfall. Table 3 shows the mean, standard deviation, minimum, maximum as well as 25, 50, and 75
percentiles values of hourly typhoons’ rainfall. Results show that the mean value is 35 mm hr⁻¹, and the standard
deviation is 72 mm hr⁻¹. This result indicates that white noise is absent in hourly total typhoons’ rainfall, and the
data in this study is a stationary time series. On the other hand, the autocorrelation function (ACF; calculated
and plotted with pandas library in Python, McKinney, 2010) was used to check whether the values of hourly total

![Figure 3. Decompose time-series (hourly total typhoons' rainfall) into its components such as trend, seasonal pattern (a cycle structure that consistently repeats at the same frequency), and residual. The first figure represents the observed data.](image-url)

| Table 1 |
| --- |
| **Description of Augmented Dickey-Fuller and Kwiatkowski-Phillips-Schmidt-Shin Test** |
| Statistical test | ADF | KPSS |
| Null Hypothesis is accepted | The series has a unit root (non-stationary). | The process is trend stationary. |
| Alternate Hypothesis is accepted | The series has no unit root (stationary). | The series has a unit root. |
typhoons’ rainfall correlate with lag values (Figure 4). The 95% and 99% confidence levels are shown by the horizontal lines in Figure 4, with the dashed line that represents the 99% confidence level. Results show that hourly total typhoons’ rainfall is correlated with lag values. Therefore, hourly total typhoons’ rainfall is a stationary time series.

2.2. Overview of the CNN Model

Here, we describe a common architecture of the CNN model. This model consists of the input layer, convolutional layer (Conv layer), fully connected layer (FC layer) and output layer. A specific filter or a set of shared weights on the input is convolved by the Conv layer. On the other hand, the FC layer maps the input vector linearly into another one.

2.3. Input and Output Variables to the CNN Model

In supervised learning, we can represent the input sequence and forecast sequence as $t-n \ldots t-1, t \text{ and } t+1 \ldots t+n$, respectively. This study employed previous time steps to forecast the next time step — which was known as the sliding window method or window method or lag method (Bontempi et al., 2013; Brownlee, 2016). The current study used the direct strategy method for multi-step forecasting, i.e., this method was applied to each forecast lead-time (Bontempi et al., 2013; Brownlee, 2018a, 2018b; Cheng et al., 2006; Makridakis, 1994; Sorjamaa et al., 2007).

We used a partial autocorrelation function (PACF) to determine the appropriate lag lengths of input. PACF can be calculated from the following equation:

$$\rho(k) = \frac{\frac{1}{n-k} \sum_{t=k+1}^{n} (R_t - \bar{R})(R_{t-k} - \bar{R})}{\sqrt{\frac{1}{n} \sum_{t=1}^{n} (R_t - \bar{R})^2} \sqrt{\frac{1}{n-k} \sum_{t=k+1}^{n} (R_{t-k} - \bar{R})^2}}$$

(1)

$$R_t = \emptyset_{21} R_{{t-1}} + \emptyset_{22} R_{{t-2}} + e_t$$

(2)

Where $R_t$ is the typhoon rainfall within 500 km radius, $k = 0, 1, 2 \ldots$ for $t = 1, 2, 3 \ldots n$, and $\bar{R}$ represents mean typhoon rainfall within a 500 km radius. $\emptyset_{22}$ is the value of the partial autocorrelation of order 2. To calculate and plot PACF, we used ‘statsmodels’ in Python (Seabold & Perktold, 2010).

It can be seen that the current hour is directly correlated with its previous hour for typhoon rainfall (Figure 5). Therefore, current time t and previous time t−1 were used as input to the CNN model. The input and output variables to the CNN model are shown in Table 4. A general form of the CNN model can be represented as:

$$R_{t+lead \ time} = f (R_t, R_{t-1})$$

(3)
Where lead time = 1–6 hr, $R_t$ and $R_{t-1}$ are typhoon rainfall within 500 km radius at current time $t$ and previous time $t-1$. The “series_to_supervised” function in Python (Brownlee, 2017) was employed to complete the process of supervised learning.

We used the “StandardScaler” package in Python (Pedregosa et al., 2011) to standardize hourly typhoon rainfall data. The reason to use standardized data as input for the model is that the data with large values cannot dominate the model (Yu et al., 2006). Equation 4 shows this method. Finally, we used the standardized data to feed the CNN model, and then, the model output was returned to the original scale.

$$z = \frac{(x - \mu)}{s}$$

(4)

Where $x$ is the training samples, $\mu$ is the mean of the training samples, and $s$ is the standard deviation of the training samples.

![Figure 4. Autocorrelation with lag variables.](image1)

![Figure 5. The correlation statistics of the lags for typhoon rainfall data.](image2)
Table 4
Input and Output of the CNN Model

| Lead-time (h) | Input | Output |
|---------------|-------|--------|
|               | R(t-1) | R(t)   |
| 1             | *      | ●      |
| 2             | *      | ●      |
| 3             | *      | ●      |
| 4             | *      | ●      |
| 5             | *      | ●      |
| 6             | *      | ●      |

Note. Here, R is the typhoon rainfall at current time t, previous time t-1 and 277 lead-time period t+1 to 6 [* = input, ● = output].

Table 5
Description of the Number of Typhoon Events for Training and Testing

| No. event | Date (2019) | Typhoon name | Note |
|-----------|-------------|--------------|------|
| 1         | 1–8 Aug     | Francisco    | Training |
| 2         | 3–14 Aug    | Lekima       |       |
| 3         | 5–17 Aug    | Krosa        |       |
| 4         | 31 Aug – 11 Sep | Lingling |       |
| 5         | 2 Sep – 11 Sep | Faxai   |       |
| 6         | 17 Sep – 23 Sep | Tapah  |       |
| 7         | 27 Sep – 05 Oct | Mitag |       |
| 8         | 05 Oct – 14 Oct | Hagibis |       |
| 9         | 15 Oct – 22 Oct | Neoguri |       |
| 10        | 18 Oct – 25 Oct | Bualoi  |       |
| 11        | 01 Nov – 10 Nov | Halong  | Testing |
| 12        | 04 Nov – 11 Nov | Nakri   |       |
| 13        | 11 Nov – 18 Nov | Fengshen |       |
| 14        | 11 Nov – 06 Dec | Kalmuaei |       |
| 15        | 20 Dec – 27 Dec | Phanfone |       |

2.4. Parameter Tuning

2.4.1. Grid Search Cross-Validation

Ten typhoon events were used for training the model, while 5 typhoon events were used for testing the model performance (Table 5). We used the grid-search with k-fold cross-validation method in Python (Pedregosa et al., 2011; and details could be found at https://scikit-learn.org/stable/modules/grid_search.html) to avoid overfitting the model. Here, the training set was divided into k-folds [k-1 sets were used for training and the remaining set (red box in Figure 6) was used for validation]. Fivefold cross-validation was employed in this study. The grid search algorithm has a significant advantage over time-consuming mitigation, and it is simple, robust, and time-saving (Hsu, 2003; Noori et al., 2011; C. L. Wu et al., 2008; J. Wu et al., 2019; Yu et al., 2006). To find the best parameters for the CNN model, we used the ‘KerasRegressor’ wrapper (Abadi et al., 2016; Chollet, 2015) in ‘scikit-learn’ software (Pedregosa et al., 2011).

2.4.2. Hyper-Parameters Tuning for the CNN Model

Filters, units, kernel size, kernel initializer, activation function, optimizers, learning rate, momentum, loss function, epochs, and batch size are the hyper-parameters for the CNN model. Table 6 shows the ranges and lists of these parameters and the selected optimal hyper-parameter for the CNN model, respectively.

2.4.2.1. Filters and Kernel Size

The dimensionality of the output space is known as filters for the CNN model. The ranges of filters were set from 10 to 64, and 50 was the best for this model. The length of the 1-dimensional convolution window for the CNN is specified by the kernel size. This parameter was set from 1 to 5, and the optimal kernel size was 1.

2.4.2.2. Kernel Initializer

For the linear transformation of the inputs, we used a kernel initializer. Uniform, lecun_uniform (Klambauer 2017), normal, zero, glorot normal and uniform (Glorot & Bengio, 2010), and he normal and uniform (K. He et al., 2015) are initializers in Keras (defaults to ‘glorot_uniform’). These initializers were set to find the best initializer for this study (a detailed description can be found online at https://www.tensorflow.org/api_docs/python/tf/keras/initializers). Among eight initializers, glorot uniform was the best kernel initializer for this study. This initializer draws samples from a uniform distribution within [-limit, limit].

\[
\text{limit} = \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}} \tag{5}
\]

Where, \(n_{\text{in}}\) is the number of input units in the weight tensor, and \(n_{\text{out}}\) is the number of output units.

2.4.2.3. Activation Function

To transform outputs of the convolutional layer and the fully connected layer for nonlinearity, this study tested activation functions such as softmax, softplus, softsign, relu, tanh, sigmoid, hard_sigmoid, and linear. The relu was the best activation function for the CNN model. This activation function can be defined as relu = max(x, 0). Here, relu takes the element-wise rectified linear of the input (x), where all negative values of the input (x) are set to zero.

2.4.2.4. Loss Function

This study used mean absolute error (MAE) as a loss function. Equation 6 can be used to calculate MAE.

\[
\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{6}
\]
Where $\hat{y}_i$ = the predicted value of the $i$th sample, $y_i$ = the observed value of the $i$th sample, and $(\hat{y}_i - y_i) = \text{the difference between the predicted value and actual value.}$

### 2.4.2.5. Epochs and Batch Size

As we cannot pass the entire training dataset at once to the computer, we need to divide the training dataset into smaller batches for the CNN model and give it to the computer one by one. Therefore, the epoch can be defined by Equation 7.

$$1 \text{ epoch} = \frac{\text{training set size}}{\text{batch size}} \times \text{iterations} \quad (7)$$

| Hyper-parameters          | The ranges and lists of hyper-parameters | The selected optimal hyper-parameter |
|---------------------------|-----------------------------------------|-------------------------------------|
| Filters                   | 10, 30, 40, 50, 64                      | 50                                  |
| Kernel size               | 1, 2, 3, 5                              | 1                                   |
| Kernel initializer        | uniform, lecun_uniform, normal, zero, glorot_normal, glorot_uniform, he_normal, he_uniform | glorot_uniform                      |
| Activation function       | softmax, softplus, softsign, relu, tanh, sigmoid, hard_sigmoid, linear | relu                                |
| Loss function             | MAE                                     | MAE                                 |
| Epochs                    | 10, 50, 100                             | 50                                  |
| Batch size                | 10, 20, 40, 60, 72, 80, 100             | 72                                  |
| Optimizers                | SGD, RMSprop, AdaGrad, AdaDelta, Adam, Adamax, Nadam | SGD, RMSprop, AdaGrad, AdaDelta, Adam, Adamax, Nadam |
| Learning rate             | 1, 0.1, 0.01, 0.001, 0.0001, le-05, le-06, le-07 | 1, 0.1, 0.01, 0.001, 0.0001, le-05, le-06, le-07 |
| Momentum                  | 0.0, 0.2, 0.4, 0.6, 0.8, 0.9             | 0.4                                 |
For this study, epochs and batch size were tested from 10 to 100, and optimal values were found as 50 and 72, respectively.

2.4.2.6. Optimizers, Learning Rate, and Momentum

The CNN model uses optimizers to minimize the loss and maximize the efficiency of production, and this model uses momentum to accelerate the optimization process. On the other hand, the learning rate is a tuning parameter in an optimization algorithm. In other words, while moving toward a minimum of a loss function, the learning rate determines the step size at each iteration (Murphy, 2012). The learning rate can be calculated by using Equation 8 (Brownlee, 2018a, 2018b).

\[
\text{Learning rate (}\eta\text{)} = \text{initial learning rate} \times \frac{1}{1 + \text{decay rate} \times \text{iteration}} \tag{8}
\]

Where the learning rate is for the current epoch, the initial learning rate is the learning rate specified as an argument to optimization algorithms. The decay rate is the learning rate decay, and it starts training the CNN model with a large learning rate. After that, the learning rate slowly reduces until local minima are obtained. Finally, iteration is the epoch number.

Stochastic gradient descent (SGD), root mean square propagation (RMSprop), adaptive grad (AdaGrad), adaptive delta (AdaDelta), adaptive moment estimation (Adam), adaptive maximum (Adamax), and Nesterov-accelerated adaptive moment estimation (Nadam) optimizers are available in Keras (https://keras.io/api/optimizers/). Seven optimizers, eight learning rate values (1, 0.1, 0.01, 0.001, 0.0001, le-05, le-06, and le-07) and six momentum values (0.0, 0.2, 0.4, 0.6, 0.8, and 0.9) were tested in this study.

The formulas of SGD (Robbins & Monro, 1951), RMSprop (Hinton, 2012), AdaGrad (Duchi et al., 2012), AdaDelta (Zeiler, 2012), Adam and Adamax (Kingma & Ba, 2015), and Nadam (Dozat, 2016) are discussed herein.

SGD: Given a minibatch of m from the training set \{\{x^{(i)}\}, \ldots, \{x^{(m)}\}\} with corresponding targets \{y^{(i)}\} (i = 1, 2 \ldots m), the parameters of the CNN can be updated according to Equation 9.

\[
W_{\text{new}} = W_{\text{old}} - \eta \nabla L(W_{\text{old}}, x_{i}, y_{i}) \tag{9}
\]

Where \(W_{\text{new}}\) is the updated weight, \(W_{\text{old}}\) is the previous value of the weight, \(\eta\) is the learning rate and \(\nabla L(W_{\text{old}}, x_{i}, y_{i})\) is the gradient of loss function \(L(W_{\text{old}}, x_{i}, y_{i})\).

RMSprop keeps a moving average of the square of gradients and divides the learning rate by the root of this average (Hinton, 2012).

RMSprop:

\[
W_{\text{new}} = W_{\text{old}} - \frac{\eta}{\sqrt{\text{MeanSquare}(W, t)}} \nabla L(W_{\text{old}}) \tag{10}
\]

\[
\text{MeanSquare}(W, t) = \rho \text{MeanSquare}(W, t-1) + (1 - \rho)(\nabla L(W))^2 \tag{11}
\]

Where \(\rho\) = the forgetting factor = 0.9, \(t\) = the current time step.

AdaGrad: Based on the parameters, AdaGrad allows the learning rate to adapt (Zhao & Liu, 2018). This optimizer ignores the need to manually tune the learning rate (Do et al., 2019).

\[
W_{t+1, i} = W_{t, i} - \frac{\eta}{\sqrt{G_{t, i} + \epsilon}} \cdot g_{t, i} \tag{12}
\]

Where \(g_{t, i}\) is a partial derivative of the loss function, \(G_{t, i}\) is a diagonal matrix where each diagonal element \(i,\) and \(\epsilon\) is a smoothing term that avoids division by zero.
AdaDelta: To solve the problem of learning rate decay, AdaDelta is extended from AdaGrad (Do et al., 2019).

\[
W_{t+1} = W_t - \frac{RMS[\Delta W]_{t-1}}{RMS[g]_t} g_t
\]  

(13)

Where RMS is the root mean square error.

Adam: The first moment \((m_t)\) and the second moment \((u_t)\) of the gradients can be calculated by using the following equations:

\[
m_t = \rho_1 m_{t-1} + (1 - \rho_1) g_t
\]

(14)

\[
u_t = \rho_2 u_{t-1} + (1 - \rho_2) g_t^2
\]

(15)

Where \(\rho_1\) and \(\rho_2\) are exponential decay rates, \(g\) is the gradient.

After that, by using Equations 16 and 17, the correct biases in the first and second moment can be calculated as.

\[
\hat{m}_t = \frac{m_t}{1 - \rho_1 t}
\]

(16)

\[
\hat{u}_t = \frac{u_t}{1 - \rho_2 t}
\]

(17)

Finally, the parameters of the CNN can be updated according to the following formula:

\[
W_{new} = W_{old} + \Delta W
\]

\[
\Delta W = -\eta \frac{\hat{m}_t}{\sqrt{\hat{u}_t} + \epsilon}
\]

(18)

(19)

Where \(\epsilon = 10^{-8}\) is a small constant used to ensure numerical stability.

Adamax: This optimizer is based on infinity normalization, and it is an extension of Adam. Adamax calculates the biased first-moment moment \((m_t)\) and exponentially weighted infinity normalization \((\gamma_t)\) using Equations 20 and 21.

\[
m_t = \rho_1 m_{t-1} + (1 - \rho_1) g_t
\]

(20)

\[
\gamma_t = \max (\rho_2 \gamma_{t-1}, |g_t|)
\]

(21)

Finally, the CNN model parameters can be updated by using the following equation.

\[
W_{new} = W_{old} = \left(\frac{\eta}{1 - \rho_1^t}\right) \frac{m_t}{\gamma_t}
\]

(22)

Nadam: This optimizer modifies Adam with Nesterov’s accelerated gradient. The following two Equations 23 and 24 are similar to Adam to compute the gradient of first and second-moment variables.

\[
m_t = \rho_1 m_{t-1} + (1 - \rho_1) g_t
\]

(23)

\[
u_t = \nu u_{t-1} + (1 - \nu) g_t^2
\]

(24)

Equations 25 and 26 can be used to calculate the corrected moments.

\[
\hat{m} = \frac{\rho_{t+1} m_t}{1 - \prod_{i=1}^{t+1} \rho_i} + \frac{(1 - \rho_1) g_t}{1 - \prod_{i=1}^{t} \rho_i}
\]

(25)

\[
\hat{u} = \frac{\nu u_t}{1 - \nu_t}
\]

(26)
Parameters of the CNN model can be updated by using the following equation.

\[
W_{new} = W_{old} - \frac{\eta}{\sqrt{\hat{u}_i} + \varepsilon} \hat{m}_i
\]  

(27)

2.4.3. Performance Measure

The root mean square error (RMSE), MAE, Nash-Sutcliffe model efficiency coefficient (NSE), and coefficient of correlation (CC) were employed to evaluate the performance of the CNN model. The NSE ranges from the negative infinitive to 1, with 1 being a perfect match between forecasting results and observations. The better performance of the predicted results can be obtained from the smaller value of the RMSE and MAE and the higher value of NSE and CC.

The RMSE can be defined by Equation 28.

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2}
\]  

(28)

NSE can be calculated as:

\[
NSE = 1 - \frac{\sum_{i=1}^{N} (y_i - \bar{y})^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}
\]  

(29)

Equation 30 can be used to estimate the CC.

\[
CC = \frac{\sum_{i=1}^{N} (y_i - \bar{y})(\hat{y}_i - \bar{y})}{\sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2 \sum_{i=1}^{N} (\hat{y}_i - \bar{y})^2}}
\]  

(30)

Here, \(\hat{y}_i\) = the predicted value of the \(i\)th sample, \(y_i\) = the observed value of the \(i\)th sample, \(\bar{y}\) = mean observed value, and \(\bar{\hat{y}}\) = mean predicted value.

3. Results and Discussions

3.1. Effects of Learning Rates and Optimization Algorithms on Training the CNN Model Including Observation-Forecast Results

Loss over training epochs for SGD optimizer during training the CNN model is shown in Figure 7. The penalty for a bad prediction of the CNN model is defined by the loss that is a number. For the perfect prediction, this number is 0; otherwise, it is larger than 0. Results showed that a very high loss was found for a too-large learning rate of 1 and too-small learning rates from 0.0001 to 1E−7. A very small loss was found for only three learning rates: 0.1, 0.01, and 0.001. Similar results were found for the CNN model with RMSprop, Adam, AdaDelta, AdaGrad, Adamax, and Nadam algorithms (see supplementary Figures S1, S3, S5, S7, S9 and S11). Therefore, a too-large learning rate of 1 and too-small learning rates from 0.0001 to 1E−7 caused the inability of the CNN model with the SGD, RMSprop, Adam, AdaDelta, AdaGrad, Adamax, and Nadam algorithms. However, learning rates of 0.1, 0.01, and 0.001 were useful for the CNN model with these algorithms to learn the problem well. This result indicates that only three learning rates (0.1, 0.01, and 0.001) helped the CNN model to perform well on the training and testing dataset.

The observation-forecast results of the CNN model in terms of learning rates and the SGD algorithm are shown in Figure 8. For a too-large learning rate of 1, the CNN model with the SGD algorithm could not forecast anything, as the prediction line was straight. For too-small learning rates from 0.0001 to 1E−7, hourly typhoon rainfall was underestimated by the CNN model with the SGD algorithm. However, forecasting values were very close to the observed values for learning rates of 0.1, 0.01, and 0.001. The results for the CNN model with RMSprop, Adam, AdaDelta, AdaGrad, Adamax, and Nadam algorithms were similar (see Figures S2, S4, S6, S8, S10 and S12). Our results indicate that the CNN model is unable to forecast hourly typhoon rainfall with a learning rate of 1 and learning rates from 0.0001 to 1E−7. On the other hand, this model is able to forecast hourly typhoon rainfall with learning rates of 0.1, 0.01, and 0.001.
Figure 7. Loss over training epochs for stochastic gradient descent optimizer during training the convolutional neural network model. Mean absolute error is used for the loss function. The lead-time is 1 hr. The learning rate is denoted by ‘lrate’.

Figure 8. The observation-forecast results of the convolutional neural network model with the stochastic gradient descent optimizer. Here, the blue line represents the observed typhoon rainfall (mm h\(^{-1}\)), while the red line represents the forecasted typhoon rainfall (mm h\(^{-1}\)). The lead-time is 1 hr. The learning rate is denoted by ‘lrate’.
3.2. Statistical Performance of the CNN Model for 1–6 hr Lead-Time in Terms of Learning Rates and Optimization Algorithms

Figure 9 shows the forecasting performance of the CNN model with eight learning rates (1, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001, and 0.0000001) and seven optimization algorithms such as SGD, RMSprop, Adam, AdaDelta, AdaGrad, Adamax, and Nadam. For the too-large learning rate (learning rate = 1), both RMSE and MAE values were too high, and all algorithms provided the almost same result. For too-small learning rates (learning rate = 0.0001–1E−7), values of RMSE and MAE were also high and fluctuated in terms of optimization algorithms. However, both RMSE and MAE values showed that forecasting error was low and stable for learning rates of 0.1, 0.01, and 0.001. More importantly, forecasting error increased with increasing lead-time for these learning rates, but it was abnormal for a too-large learning rate of 1 and too-small learning rates from 0.0001 to 1E−7. It can be concluded that the CNN model shows the best performance for learning rates 0.1, 0.01, and 0.001, and the worst performance for the too-large and the too-small learning rates.

The model efficiency and correlation between observed and predicted values are shown in Figure 10. Here, the value 1 of NSE indicates a perfect match between prediction results and observations, and all values of CC are significant at 95% and 99% confidence levels. The model efficiency was too worse for the too-large learning rate (learning rate = 1), as NSE ranges from −3.5 to −2 for all lead-time forecasting. For the too-small learning rate (learning rates from 0.0001 to 1E−7), the value of NSE was close to 0 in most cases. However, the value of NSE was nearly 1 for learning rates 0.1, 0.01, and 0.001 in terms of 1–2 hr ahead of forecasting. After that, the model efficiency decreased with increasing lead-time for these learning rates. On the other hand, the value of CC was close to 0 for the too-large learning rate (1), while this value reached −1 for the too-small learning rate (0.0001–1E−7). However, a significant strong positive correlation was found for learning rates 0.1, 0.01, and 0.001. Overall, the lowest model efficiency and lowest correlation (p < 0.05, 0.01) between observed and prediction values were found for the too-large learning rate and the too-small learning rates. On the other hand, the highest model efficiency and highest correlation (p < 0.05, 0.01) between observed and forecasted values were found for learning rates 0.1, 0.01, and 0.001.
3.3. Model Comparison

We compared the results of the present study with previous studies in terms of optimization algorithm, learning rate, NSE (%), average relative RMSE (rRMSE = RMSE/mean observation value), and RMSE (Table 7). Lin and Chen (2005) and Lin and Wu (2009) used typhoon data from 10 rain gauges in the Tanshui River basin in northern Taiwan for forecasting typhoon rainfall within 1-hr lead-time. The model efficiency results of Lin and

![Image](image-url)

Figure 10. Nash-Sutcliffe model efficiency coefficient and coefficient of correlation within 1–6 hr lead-time in terms of learning rate and optimization algorithms for the convolutional neural network model.

| Method                  | Region | Lead-time | Optimization algorithms | Learning rate | NSE (%) | Average rRMSE | RMSE (mm)   |
|-------------------------|--------|-----------|-------------------------|---------------|---------|---------------|-------------|
| Lin and Chen (2005)     | NT     | 1 hr      | -                       | 0.01          | 30–40   | -             | -           |
| Lin and Wu (2009)       | NT     | 1 hr      | -                       | -             | 40–80   | -             | -           |
| Huang et al. (2018)     | NWPO   | 1 day     | -                       | -             | -       | 21.94 (FNN-LLE), 24.07 (ECMWF), and 25.22 (SRM) |
| Wei and Chou (2020)     | NT     | 1–6 hr    | -                       | 0.1–0.4       | -       | 1.98 (Case 1), 2.025 (Case 2), and 2.03 (Case 3) |
| Our study               | NWPO   | 1–6 hr    | SGD, RMSprop, AdaGrad, AdaDelta, Adam, Adamax, Nadam | 0.1, 0.01, and 0.001 | 90 (1h), 83 (2h), 78 (3h), 70 (4h), 65 (5h) and 59 (6h) | 0.96 | 14 (1h), 17–18 (2h), 20–21 (3h), 23–24 (4h), 25–26 (5h), and 28–29 (6h) |

Note. Here, NT, NWPO, FNN-LLE, ECMWF, and SRM represent northern Taiwan, Northwestern Pacific Ocean, fuzzy neural network locally linear embedding, European Center for Medium-Range Weather Forecasts, and stepwise regression method respectively. Note. Case 1 = 271 typhoons events that affected Taiwan, Case 2 = typhoons that passed through Taiwan, and Case 3 = typhoon events that reached the level of heavy rain (40 mm/hr).
Chen (2005) and Lin and Wu (2009) varied rain gauge wise (30%–40% and 40%–80%, respectively). For SGD, RMSprop, AdaGrad, AdaDelta, Adam, Adamax, Nadam optimizers with learning rates 0.1, 0.01, and 0.001, the efficiency of our model was higher (90%) for 1-hr lead-time than the model developed by Lin's team. For 2–6 hr lead-time, the model efficiency of this study was also higher (59%–83%) than Lin and Chen (2005) and Lin and Wu (2009). The rRMSE value (0.96) of our model was lower than the model (rRMSE value = 1.98 for Case 1, 2.025 for Case 2, and 2.03 Case 3) developed by Wei and Chou (2020). Here, the authors used cases 1, 2, and 3 for 271 typhoons events that affected Taiwan, typhoons that passed through Taiwan, and typhoon events that reached the level of heavy rain (40 mm/hr) respectively. Finally, the value of RMSE (mm) was lower for our study than Huang et al. (2018). It can be concluded that the current CNN model appears to be a powerful tool for forecasting the hourly typhoon rainfall, especially for the long lead-time forecasting (1–6 hr).

4. Conclusion

This study has investigated the effects of learning rates and optimization algorithms on the forecasting performance of the CNN model. This model used hourly typhoon rainfall within a 500 km radius from each typhoon center as input. Results show that the CNN model is very sensitive to learning rate because too-large and too-small learning rates result in the inability of the model to learn to forecast hourly typhoon rainfall. Only learning rates of 0.1, 0.01, and 0.001 are useful for the CNN model to forecast hourly typhoon rainfall. More importantly, the CNN model with SGD, RMSprop, AdaGrad, AdaDelta, Adam, Adamax, Nadam optimizers and learning rates of 0.1, 0.01, and 0.001 is capable of providing more accurate hourly typhoon rainfall forecasts than the existing models. Therefore, the CNN model is recommended as an alternative to the existing model for disaster prevention and mitigation.

Data Availability Statement

To run the CNN model, the Keras (Chollet, 2015) from TensorFlow (Abadi et al., 2016) can be used in Python (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv1D). The typhoon best track dataset can be downloaded from the China Meteorological Administration (https://tcdata.typhoon.org.cn/en/zjlsjji_zlhq.html; Lu et al., 2021; Ying et al., 2014). The rainfall data can be downloaded from the National Aeronautics and Space Administration (NASA; Huffman et al., 2014; https://disc.gsfc.nasa.gov/datasets/GPM_3IMERGHH_06/summary?keywords=%22IMERG%20final%22). Scripts in NCL, Python, and MATLAB for the corresponding figure of this article can be found in Md. Jalal Uddin’s Zenodo repository (https://doi.org/10.5281/zenodo.5915311).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating systems Design and Implementation, OSDI (Vol. 2016, pp. 265–283). USENIX Association.

Bontempi, G., Ben Taieb, S., & Le Borgne, Y. A. (2013). Machine learning strategies for time-series forecasting. Machine Learning with Applications, 7, 100204. https://doi.org/10.1016/j.mlwa.2021.100204

Brownlee, J. (2012). Practical recommendations for gradient-based training of deep architectures. Lecture notes in computer science (including Subseries Lecture notes in Artificial Intelligence and Lecture notes in Bioinformatics), 7700, 478 LECTU. https://doi.org/10.1007/978-3-642-35289-8_26

Bengio, Y. (2012). Understanding the dynamics of learning rate on deep learning neural networks. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Bui, D. T., Shahabi, H., Omidvar, E., Shirzadi, A., Pham, B. T., Chapi, K., et al. (2019). Functional machine learning algorithm. Remote Sensing, 11(931), 1–22. https://doi.org/10.3390/rs11080931

Chen, B. F., Chen, B., Lin, H. T., & Elsberry, R. L. (2019). Estimating tropical cyclone intensity by satellite imagery utilizing convolutional neural networks. Weather and Forecasting, 34(2), 447–465. https://doi.org/10.1175/WAF-D-18-0136.1
Chen, F. J., & Fu, Y. F. (2015). Contribution of tropical cyclone rainfall at categories to total precipitation over the Western North Pacific from 1998 to 2007. *Science China Earth Sciences*, 58(11), 2015–2025. https://doi.org/10.1007/s11430-015-5103-9

Chen, R., Wang, X., Zhang, W., Zhu, X., Li, A., & Yang, C. (2019). A hybrid CNN-LSTM model for typhoon formation forecasting. *Geofisica Internacional*, 59(3), 375–396. https://doi.org/10.1007/s10707-019-00355-0

Cheng, H., Tan, P. N., Gao, J., & Scrip, J. (2006). Multistep-ahead time series prediction. *Lecture Notes in Computer Science*, 3918 LNAI, 765–774. https://doi.org/10.1007/11731139_89

Chollot, F. (2015). Keras. Retrieved from https://github.com/fchollet/keras

Dare, R. A., Davidson, N. E., & Mcbride, J. L. (2012). Tropical cyclone contribution to rainfall over Australia. *Monthly Weather Review*, 140(11), 3606–3619. https://doi.org/10.1175/MWR-D-11-00340.1

Del, I. V., & Starchenko, A. V. (2021). Forecast of the near ground air temperature based on the multilayer perceptron model. *Journal of Physics: Conference Series*, 1909(1), 012025. https://doi.org/10.1088/1742-6596/1909/1/012025

Dicker, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366a), 427–431. https://doi.org/10.1080/01621459.1979.10842531

Do, D. T. T., Lee, J., & Nguyen-Xuan, H. (2019). Fast evaluation of crack growth path using time series forecasting. *Engineering Fracture Mechanics*, 218(July), 106567. https://doi.org/10.1016/j.engfracmech.2019.106567

Dozat, T. (2016). Incorporating Nesterov momentum into Adam. *ICLR Workshop*, (1), 2013–2016.

Duchi, J. C., Bartlett, P. L., & Wainwright, M. J. (2012). Randomized smoothing for (parallel) stochastic optimization. *Proceedings - IEEE Conference on Decision and Control*, 12, 5442–5444. https://doi.org/10.1109/CDC.2012.6426698

Engelhath, P. J., & Douglas, A. V. (2001). The role of Eastern North Pacific tropical storms in the rainfall climatology of Western Mexico. *International Journal of Climatology*, 21(11), 1357–1370. https://doi.org/10.1002/joc.637

Fadliah, R., Djmal, E. C., & Byas, R. (2021). Rainfall prediction due to El Nino factors using recurrent neural networks. *Journal of Physics: Conference Series*, 1845(1), 012025. https://doi.org/10.1088/1742-6596/1845/1/012025

Ghimire, S., Deo, R. C., Raj, N., & Mi, J. (2019). Deep learning neural networks trained with MODIS satellite-derived predictors for long-term regional global solar radiation prediction. *Energies*, 12(2), 2407. https://doi.org/10.3390/en12122407

Giffard-Roisin, S., Yang, M., Charpiat, G., Bonnifait, C. K., Kgl, B., & Manteuleoni, M. (2019). Tropical cyclone track forecasting using fused deep learning from aligned reanalysis data. *ArXiv*, 3(January), 1–13. https://doi.org/10.48550/arXiv.2020.00001

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep models. *Journal of Machine Learning Research*, 9, 249–256.

Goldfellow, I., Bengio, Y., & Courville, A. (2016). Chapter 8 optimization for training deep models. In *Deep learning*. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Chapter 8 optimization for training deep models. In *Deep learning*. Guaridan, T. (2021). China floods: Thousands trapped without fresh water as rain moves north. Accessed August 12, 2021. Retrieved from https://www.theguardian.com/world/2021/jul/23/china-floods-thousands-trapped-without-fresh-water-as-rain-moves-north

He, H., & Luo, F. (2020). Study of LSTM air quality index prediction based on forecasting timeliness. *IOP Conference Series: Earth and Environmental Science*, 446(3), 032113. https://doi.org/10.1088/1755-1315/446/3/032113

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision*, 2015, 1026–1034. Inter. https://doi.org/10.1109/ICCV.2015.123

Hernández Ayala, J. J., & Matyas, C. J. (2018). Spatial distribution of tropical cyclone rainfall and its contribution to the climatology of Puerto Rico. *Physical Geography*, 39(1), 1–20. https://doi.org/10.1080/03014215.2017.1334416

Hinton, G., Srivastava, N., & Swersky, K. (2012). Neural networks for machine learning—Lecture 6e—Rmsprop: Divide the gradient by a running average of its recent magnitude. Retrieved from http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Hikotoko, M., Sakuraba, M., & Sei, Y. (2017). Development of the real-time river stage prediction method using deep learning. *Journal of JSCE*, 5(1), 422–429. https://doi.org/10.2208/journalofjsce.5.1_422

Hsu, C. W. (2003). A Practical Guide to support vector classification. Retrieved from https://www.csie.ntu.edu.tw/~cjlin/papers/guide.pdf

Huang, Y., Jin, L., Zhao, H. S., & Huang, X. Y. (2018). Fuzzy neural network and LLE algorithm for forecasting precipitation in tropical cyclones: Comparisons with interpolation method by ECMWF and stepwise regression method. *Natural Hazards*, 91(1), 201–220. https://doi.org/10.1007/s11069-017-3122-x

Huffman, G., Bolvin, D., Braithwaite, D., Hsu, K., Joyce, R., & Xie, P. (2014). Integrated multi-satellite Retrievals for GPM (IMERG), version 4.4. NASA’s Precipitation processing center. Retrieved from ftp://arthurhou.pps.eosdis.nasa.gov/gpmdata/

Jazeera, A. (2021). Fresh floods in central China as typhoon heads towards east coast. Accessed November 28, 2021. Retrieved from https://www.jazeera.net/news/2021/7/23/fresh-floods-in-central-china-as-typhoon-heads-towards-east-coast

Jiang, H., & Zipser, E. J. (2010). Contribution of tropical cyclones to the global precipitation from eight seasons of TRMM data: Regional, seasonal, and interannual variations. *Journal of Climate*, 23(6), 1526–1543. https://doi.org/10.1175/2009jcli3303.1

Kim, J. S., Chen, A., Lee, J., Moon, I. J., & Moon, Y. I. (2020). Statistical prediction of typhoon-induced rainfall over China using historical rainfall, tracks, and intensity of typhoon in the western north pacific. *Remote Sensing*, 12(24), 1–14. https://doi.org/10.3390/rs12244133

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. 3rd International Conference on Learning Representations. ICLR 2015 - Conference Track Proceedings, 1–15.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. *ArXiv*, 1(12), 1–20. https://doi.org/10.48550/arXiv.1706.02281

Lavender, S. L., & Abbas, D. J. (2013). Trends in Australian rainfall: Contribution of tropical cyclones and closed lows. *Climate Dynamics*, 40(1–2), 317–326. https://doi.org/10.1007/s00382-012-1566-y

Lee, C. S., Cheung, K. K. W., Fang, W. T., & Elsberry, R. L. (2010). Initial maintenance of tropical cyclone size in the Western North Pacific. *Monthly Weather Review*, 138(8), 3207–3223. https://doi.org/10.1175/2010MWR3203.1

Lin, G. F., & Chen, L. H. (2004). A non-linear rainfall-runoff model using radial basis function network. *Journal of Hydrology*, 289(1–4), 1–8. https://doi.org/10.1016/j.jhydrol.2003.10.015
Lin, G. F., & Chen, L. H. (2005). Application of an artificial neural network to typhoon rainfall forecasting. *Hydrological Processes, 19*(9), 1825–1837. https://doi.org/10.1002/hyp.5638

Lin, G. F., Chen, G. R., Wu, M. C., & Chou, Y. C. (2009). Effective forecasting of hourly typhoon rainfall using support vector machines. *Water Resources Research, 45*(8), 1–11. https://doi.org/10.1029/2009WR007911

Lin, G. F., Zhong, B. C., & Chang, C. C. (2013). Development of an effective data-driven model for hourly typhoon rainfall forecasting. *Journal of Hydrology, 495*, 52–63. https://doi.org/10.1016/j.jhydrol.2013.04.056

Lin, G. F., & Wu, M. C. (2009). A hybrid neural network model for typhoon-rainfall forecasting. *Journal of Hydrology, 375*(3–4), 450–458. https://doi.org/10.1016/j.jhydrol.2009.06.047

Lonfat, M., Rogers, R., Marchok, T., & Marks, F. D. (2007). A parametric model for predicting hurricane rainfall. *Monthly Weather Review, 135*(9), 3086–3097. https://doi.org/10.1175/MWR3433.1

Lu, X., Yu, H., Ying, M., Zhao, B., Zhang, S., Lin, L., et al. (2021). Western north pacific tropical cyclone database created by the China meteorological administration. *Advances in Atmospheric Sciences, 1*. https://doi.org/10.1007/s00376-020-0211-7

Makridakis, S. (1994). Time series prediction: Forecasting the future and understanding the past. *International Journal of Forecasting, 10*(3), 463–466. https://doi.org/10.1016/j.ijforecast.2009.07.007

Manci, S., & Ananth, J. P. (2020). MapReduce and optimized deep network for rainfall prediction in agriculture. *Computer Journal, 63*(6), 900–912. https://doi.org/10.1093/comjnl/bzx164

Matsuoka, D., Nakano, M., Sugiyama, D., & Uchida, S. (2018). Deep learning approach for detecting tropical cyclones and their precursors in the simulation by a cloud-resolving global nonhydrostatic atmospheric model. *Progress in Earth and Planetary Science, 5*(1), 1–16. https://doi.org/10.1186/s40645-018-0243-y

McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference, J(Scipy)*, 56–61. https://doi.org/10.25080/majora-92f81922-00a

Murphy, K. P. (2012). Machine learning—a probabilistic perspective-MIT Press. A Probabilistic Perspective. NASA. (2021). Flooding in Central China. Accessed August 12, 2021. Retrieved from https://earthobservatory.nasa.gov/images/148623/flooding-in-central-china

Nath, A., Barman, D., & Saha, G. (2021). Gated Recurrent Unit: An effective tool for runoff estimation. *Lecture Notes in Networks and Systems, 170* LNNS, 145–155. https://doi.org/10.1007/978-981-33-4084-8_14

News, A. P. (2021). More rain to hit China’s flood-ravaged Henan province. Accessed November 28, 2021. Retrieved from https://apnews.com/article/china-floods-flooding-in-central-china

Nogueira, R. C., & Keim, B. D. (2011). Contributions of Atlantic tropical cyclones to monthly and seasonal rainfall in the eastern United States 1960-2007. *Theoretical and Applied Climatology, 103*(3), 213–227. https://doi.org/10.1007/s10955-010-0292-9

Noori, R., Karbassi, A. R., Moghaddamnia, A., Han, D., Zokaei-Ashtiani, M. H., Farokhnia, A., & Gousheh, M. G. (2011). Assessment of input variables determination on the SVM model performance using PCA, Gamma test, and forward selection techniques for monthly stream flow prediction. *Journal of Hydrology, 40*(3–4), 177–189. https://doi.org/10.1016/j.jhydrol.2011.02.021

Porta, T., & Gigizolu, H. K. (2008). Estimation and forecasting of daily suspended sediment data using wavelet-neural networks. *Journal of Hydrology, 358*(3–4), 317–331. https://doi.org/10.1016/j.jhydrol.2008.06.013

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*.

Prahlakaran, S. (2019). Time series analysis in Python – a Comprehensive Guide with Examples. Retrieved from https://www.machinlearning-plus.com/time-series/time-series-analysis-python-

Prasetya, E. P., & Djamal, E. C. (2019). Rainfall forecasting for the Natural disasters Preparation using Recurrent neural networks. *Proceedings of the International Conference on Electrical Engineering and Informatics, 52–57*. https://doi.org/10.1109/ICEEI47359.2019.8988838

Prat, O. P., & Nelson, B. R. (2013a). Mapping the world’s tropical cyclone rainfall contribution over land using the TRMM Multi-satellite Precipitation Analysis. *Water Resources Research, 49*(11), 7236–7254. https://doi.org/10.1002/wrcr.20527

Prat, O. P., & Nelson, B. R. (2013b). Precipitation contribution of tropical cyclones in the southeastern United States from 1998 to 2009 using TRMM satellite data. *Journal of Climate, 26*(3), 1047–1062. https://doi.org/10.1175/JCLI-D-11-00736.1

Pulido-Calvo, I., & Portela, M. M. (2007). Application of neural approaches to one-step daily flow forecasting in Portuguese watersheds. *Journal of Hydrology, 332*(1–2), 1–15. https://doi.org/10.1016/j.jhydrol.2006.06.015

Puspita Sari, A., Suzuki, H., Kitajima, T., Yasuno, T., Arman Prasetya, D., & Rabi’, A. (2020). Prediction of wind speed and direction using a stochastic approximation method. *The Annals of Mathematical Statistics, 22*(3), 400–407. https://doi.org/10.1214/aoms/1177729586

Rahimnazad, M., Moghadamnia, N., Zolfouoon, H., Sohtani, J., Dananbehr Mohajer, A., & Kwon, H. H. (2021). Performance comparison of an LSTM-based deep learning model versus conventional machine learning algorithms for streamflow forecasting. *Water Resources Management, 35*(12), 4167–4187. https://doi.org/10.1007/s11269-021-02937-w

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics, 22*(3), 400–407. https://doi.org/10.1214/aoms/1177729586

Saputri, I. V., Djamal, E. C., Nugraha, F., & Ilyas, R. (2020). Wind Speed Forecasting toward El Nino Factors Using Recurrent Neural Networks (pp. 123–128). https://doi.org/10.1109/2e50715.2020.9274686

Sari, Y. R., Djamal, E. C., & Nugraha, F. (2020). Daily rainfall prediction using one dimensional convolutional neural networks. *International Conference on Computer and Informatics Engineering, ICUCE2020*, 90–95. https://doi.org/10.1109/IC2E50715.2020.9274572

Schreck, C. J., Molinari, J. C., & Mohr, K. I. (2011). Attributing tropical cyclogenesis to equatorial waves in the western north pacific. *Journal of the Atmospheric Sciences, 68*(2), 195–209. https://doi.org/10.1175/2011JAS3396.1

Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. In 9th Python in science Conference. https://doi.org/10.25080/majora-92b19222-011

Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., & Lundasé, A. (2007). Methodology for long-term prediction of time series. *Neurocomputing, 70*(16–18), 2861–2869. https://doi.org/10.1016/j.neucom.2006.06.015

Tien Bui, D., Hoang, N. D., Martinez-Alvarez, F., Ngo, P. T. T., Hoa, P. V., Pham, T. D., et al. (2020). A novel deep learning neural network approach for predicting flash flood susceptibility: A case study at a high frequency tropical storm area. *The Science of the Total Environment, 707*, 134413. https://doi.org/10.1016/j.scitotenv.2019.134413

Tuleya, R. E., DeMaria, M., & Kugiumtzis, R. J. (2007). Evaluation of GFDL and simple statistical model rainfall forecasts for U.S. landfalling tropical storms. *Weather and Forecasting, 22*(1), 56–70. https://doi.org/10.1175/WAF972.1
