RAINFALL OCCURRENCE PREDICTION WITH CONVOLUTIONAL NEURAL NETWORK

Sunmin KIM¹, Tsuguaki SUZUKI² and Yasuto TACHIKAWA³

¹Member of JSCE, Dr. Eng., Associate Professor, Dept. of Civil and Earth Resources Eng., Kyoto University (C1-292, KyotoDaigaku-Katsura, Nishikyoku, Kyoto 615-8540, Japan)  
E-mail: kim.sunmin.6x@kyoto-u.ac.jp (Corresponding Author)
³Member of JSCE, Dr. Eng., Professor, Dept. of Civil and Earth Resources Eng., Kyoto University

A rainfall occurrence prediction model was developed using a convolutional neural network (CNN), a representative machine learning algorithm in image recognition. A spatiotemporal data array was created from the time series of related atmospheric variables from multiple ground gauge observation sites and used as the image data set. By feeding the atmospheric data array into the CNN algorithm as an input, the algorithm was trained to classify whether there will be rain in the next 30 min. The trained models demonstrate promising results for three different cities in Japan, with a 64 – 76 % detection ratio for a 30 min prediction lead time. The high false alarm ratio is an issue that should be addressed in further research, with additional input data. This paper presents the basic concept of the developed model and the results from modeling tests, with various model structures and input data combinations.

**Key Words:** Convolutional neural network, rainfall occurrence prediction, AMeDAS data

1. INTRODUCTION

Rainfall prediction is an essential prerequisite for flood forecasting in water resources engineering; however, it is one of the most challenging tasks in hydrological research. Extensive research efforts vary from conventional methods of numerical weather prediction¹ or radar image extrapolation² to recent challenges involving machine learning algorithms³⁻⁴.

Recent progress concerning machine learning algorithms with increased computing power and accumulated observation data has attracted the attention of many researchers, who are eager to improve rainfall forecasting accuracy. At present, machine learning algorithms are being actively tested in storm cell detection⁵, downscaling of satellite images⁶, and even mimicking numerical weather predictions⁷.

The convolutional neural network (CNN) is one of the most popular algorithms in rainfall forecasting⁵⁻⁷ owing to its specialized function in extracting features from spatiotemporal data. Suzuki et. al.⁸ successfully developed a rainfall occurrence detection algorithm using a CNN, and Park et. al.⁹ tested the algorithm with a varied input data array to utilize the algorithm for a limited observation site.

The CNN algorithm has been developed for image recognition by extracting features with a small set of parameters, known as a filter, by scanning the image several times. The algorithm can be applied to any form of data that can be expressed in two- or three-dimensional images with several channels (e.g., color images with RGB channels). Spatiotemporal data of atmospheric movement can be expressed as two- or three-dimensional data, and it can be successfully applied to CNN algorithms for rainfall detection in our previous study⁸⁻⁹. The developed CNN model is only for on-off forecasting using ground gauged atmospheric data, and thus the model set up is not complicated. Furthermore, once the model is properly trained, it does not ask a high computational resource as numerical weather prediction model does. However, the proposed model should be evaluated in various environments to confirm the availability of the algorithm in a practical sense; it is also necessary to examine the sensitivity of the algorithm to the input data format and model structure to determine its stability.

In this study, the original CNN model, developed by Suzuki et. al.⁸⁻⁹, will be extensively tested on multiple target areas in Japan, Kyoto, Osaka, and Tokyo, with various forms of input data and model structures. The main purpose of this study is to confirm the model performance in various environments and understand the sensitivity of the model to input data and model structures.

In this paper, Section 2 describes the basic concept of the CNN algorithm and the details of the developed model. Section 3 presents the test results, and Section 4 concludes the paper by summarizing the results.
2. DATA AND METHODOLOGY

2.1 Convolutional Neural Network

The CNN was originally introduced for document image recognition\(^{(9)}\) and has been upgraded to its current format for image recognition. The algorithm has three data processing layers: the convolution layer for extracting features; the pooling layer for summarizing the features; and the fully-connected layer to classify the features for the final output.

First, the convolutional layer extracts the image features by applying a set of weight factors, \(w_{p,q,k}\), where \((p, q, k) \in [0, H - 1] \times [0, H - 1] \times [1, K]\), for a certain input area, \(x_{i,j,k}\), where \((i, j, k) \in [1, L] \times [1, L] \times [1, K]\). Here, the input data has a three-dimensional array format, such as an image with \(L \times L\) pixels and \(K\) channels (e.g., RGB channel); the filter also has a three-dimensional array format.

\[
y_{l,j,k'} = F \left( \sum_{k=1}^{K} \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} w_{p,q,k} x_{i+p,j+q,k} + b_{k'} \right) 
\]

The extracted feature, \(y_{l,j,k'}\), will be estimated using Eq. (1). The process in function \(F(\cdot)\) is called a convolutional process and is applied to the whole input range by shifting \((i,j)\) by a certain stride (in general, the stride is one pixel). The result of the convolutional process is an \((L - H + 1) \times (L - H + 1) \times M\) feature map, where \(M\) is the number of convolutional processes. The size can be controlled as \(L \times L \times M\) by adding zero values surrounding the input data (zero padding). The function \(F(\cdot)\) is called an activation function and it converts the convolution results. The most common activation function in CNN is the rectified linear unit (ReLU).

Second, the pooling layer summarizes and reduces the size of the extracted features by taking the maximum values within a given window (max-pooling). As shown in Equation (2), max-pooling is taking only one maximum value within the given area, \(U_{s,t}\), where \(s\) and \(t\) are the vertical and horizontal size of the given area, respectively. If the given area of the input data is \(2 \times 2\) for \(L \times L \times M\), then the pooling layer provides an \((L/2) \times (L/2) \times M\) output. By conducting the max pooling process, only significant information is delivered to the next process and unnecessary features are eliminated, improving the model performance and reducing computational resources.

\[
y_{l,j,k} = \max_{(i,j) \in U_{s,t}} x_{l,i,j} 
\]

Third, a fully connected layer rearranges the three-dimensional feature map into a one-dimensional array and connects to the output layer for classification. The SoftMax function is often utilized as an activation function to emphasize the classification task and a cross entropy function is the most commonly used error function in the output layer. More details of the CNN algorithm can be found in O’Shea and Nash\(^{(11)}\).

The main advantage of using CNN compared to the conventional artificial neural network is the processing efficiency with a small number of parameters. The CNN algorithm can efficiently extract specific features from the input image by focusing on a limited region of the image using a filter.

There are several hyperparameters for maximizing the learning process in the CNN algorithm, such as the layer numbers, filter size, epoch numbers, and batch size. In this study, some parameters were fixed (e.g., epoch, batch size, and learning rate) and the sensitivity to input data and model structures was investigated. The details are given in Section 2.3.
2.2 Modeling Concept

The CNN algorithm has been developed for efficient image recognition, and once it is trained with enough number of data sets, the algorithm can classify images by extracting the necessary features. The application of the algorithm is not restricted to image recognition. If any form of data can be arranged into a two- or three-dimensional array and contains certain meaning within the information, the CNN can be trained to capture and extract the internal features.

We have developed a new concept for a rainfall forecasting model based on the CNN algorithm\(^5,9\), which can detect rainfall occurrence within a certain lead time; this is achieved by feeding the ground-observed atmospheric variables (e.g., rainfall amount, temperature, wind speed) as input data and training the CNN algorithm with the given result as to whether there is rainfall within a certain lead time (e.g., 30 min or 1 hr later).

Spatiotemporal data of a certain atmospheric variable from multiple observation points for the target area can be formulated into a two-dimensional data array by aggregating the time series of the atmospheric variables from multiple observation points. If there are multiple atmospheric variables available, by overlaying two-dimensional spatiotemporal data for several atmospheric variables, a three-dimensional data array can be formulated, as shown in Fig. 1.

This data array is the record of atmospheric movements for the target area, and there must be a signal of specific atmospheric movements if there is rainfall at some subsequent time. Once the signal is successfully captured as a feature map in the CNN algorithm, the algorithm can be utilized as a rainfall-detection model for the trained forecasting lead time. This modeling concept was initially tested in the Kyoto region, Japan, showing promising results\(^8\).

In this study, we test the performance of the model on additional target regions, with various inputs and model structures to confirm the model stability.

2.3 Input Data and Model Structures

<table>
<thead>
<tr>
<th>Table 1. Model Structure and Parameter Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CATEGORY</strong></td>
</tr>
<tr>
<td><strong>PREDICTION TARGET</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>MODEL STRUCTURE</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>TRAINING OPTION</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

The target regions are Kyoto, Osaka, and Tokyo, as shown in Fig. 2, and the prediction lead time is set as 30 min ahead, with 10 min intervals from the AMeDAS ground observation data. The threshold for classifying rainfall events is set as 1.0 mm/10 min.

The input data was formulated into two types. One, with an 80 min time series from 8 gauge stations for 6 atmospheric variables \((8 \times 8 \times 6)\), and the other, with a 160 min time series from 16 gauge stations for 6 atmospheric variables \((16 \times 16 \times 6)\). We selected 6 atmospheric variables, which are the rainfall amount, temperature, wind speed, and components of wind direction (as a direction vector), and sunshine ratio for every 10 min interval, because these variables are available from every AMeDAS station.

The gauge stations are \{_Sanda, Miyama, Sonobe, Nose, Kyoto, Otsu, Hirakata, Osaka, Nara, Ikomayama, Kyotanabe, Ueno, Shigarak, Tsuchiyama, Higashomi, Minami-komatsu\}_ for Kyoto, \{_Sanda, Kobe, Kumatori, Sakai, Osaka, Ikomayama, Hirakata, Nose, Kyoto, Kyotanabe, Nara, Hari, Gojo, Katsuragi, Miki\}_ for Osaka, and \{_Fuchu, Saitama, Nerima, Tokyo, Yokohama, Edogawa-rinkai, Funabashi, Kyoshigaya,\}_ for Tokyo.
Ome, Tokorozawa, Hachioji, Ebina, Kisarazu, Chiba, Sakura, Abiko} for Tokyo, where the italics are for 8 stations and the remaining are for 16 stations. The data set was prepared for the summer season from July to September during 2008 – 2017, and data from 2008 – 2015 were utilized for training the model; data from 2016 – 2017 were utilized for testing and validating the model.

In this study, a standard CNN structure is adopted with two convolutional layers, two pooling layers, and one fully-connected layer (C1-P1-C2-P2-FC). For the convolutional process, three different convolution numbers (8, 16, and 32 times) were tested; the neuron size in the fully connected layer was also tested, with several options, e.g., $N$, $N/2$, and $N/4$, where $N$ is the data size at the second pooling layer.

The filter size in every convolution process was set as $3 \times 3$ and the pooling window was set as $2 \times 2$ for every max-pooling process. Furthermore, the optimizer, learning rate, and minibatch size were unified as shown in Table 1. The maximum training epoch is set as 100 and the best training results were adopted before overfitting appears in every 10-epoch interval.

3. RESULTS AND DISCUSSION

In summary, three options (8, 16, and 32 filtering times) were tested for the two convolutional layers, and another three options ($N$, $N/2$, and $N/4$) were tested for the fully connected layer. Thus, 27 different model structures were tested with two different forms of input data ($8 \times 8 \times 6$ and $16 \times 16 \times 6$).

For evaluating the results, an accuracy ratio (ACC), critical success index (CSI), probability of detection (POD), and false alarm ratio (FAR) were defined as the evaluation criteria. The ACC is the overall ratio of correct prediction, including rainfall and no-rainfall events, while the CSI is the ratio of correct prediction ($FO$) among the sum of correct predictions ($FO$), false alarms ($FX$), and missed events ($XO$; excluding no-rain events). The POD is the ratio of correct predictions among the sum of correct predictions and missed events, while the FAR is the ratio of false alarms among the model predictions.

3.1 Prediction Accuracy

The testing results from 27 trained models with 16-gauge stations ($16 \times 16 \times 6$) are plotted with CSI and POD indices for Kyoto, Osaka, and Tokyo in Fig. 3; this shows that even though the POD is very high (around 0.75 – 0.85) for all the three target areas, the CSI value is limited to 0.395 (in Kyoto) because there is also a high FAR in most cases.

The models with a high POD also show high FAR, resulting in a low CSI. The results with a high CSI show a low FAR; however, the POD is also lower. It appears that there is a certain trade off in the prediction accuracy of the models.

Figure 3. CSI and POD for Kyoto (up), Osaka (middle), and Tokyo (down) with the model structure of Conv.1 (8: Red, 16: Blue, 32: Green), Conv.2 (8: Small, 16: Middle, 32: Big), and F.C ($N$: , $N/2$: O, and $N/4$: ×).
Including no more than 50.78 for Osaka, and 0.55 for Tokyo, 0.226 for Tokyo. The POD shows validation results from Table 2, testing results on model was selected from the 27 model structures demonstrate spatiotemporal data (appears respectively mean POD 16 stations are 0.358 and 0.348, respectively, while the to achieve high POD, but it is resulted in (0.763 vs. 0.680 slightly for (compared to stations Table 2. From all 27 the validation results are described to 16 × 16 × 6. The results from the input size of 8 × 8 × 6 were also evaluated and the best CSI models from these two different input data sets are compared in Table 2. In general, the prediction results with 16 gauge stations (16 × 16 × 6) show improved accuracy compared to the results with 8 gauge stations (8 × 8 × 6) from the aspect of the CSI (0.395 vs. 0.378, for Kyoto); however, the results with 8 stations show a slightly higher POD than the results with 16 stations (0.763 vs. 0.680, for Kyoto). It seems that the model with (8 × 8 × 6) of input tends to predict aggressively to achieve high POD, but it is resulted in a low CSI.

From all 27 Kyoto models, the mean CSIs with 8 and 16 stations are 0.358 and 0.348, respectively, while the mean PODs with 8 and 16 stations are 0.790 and 0.757, respectively (these are not shown in the table). It appears that the model with larger spatiotemporal data (16 × 16 × 6) is more stable than that with limited spatiotemporal data (8 × 8 × 6).

In Table 2, the validation results are described to demonstrate the performance objectively. The best model was selected from the 27 model structures based on the test results, and thus it should be verified with a different data set. In general, the performance of the testing results decreases during validation. In the validation results from Table 2, the developed model shows a CSI of 0.271 for Kyoto, 0.318 for Osaka, and 0.226 for Tokyo. The POD is still high, 0.665 for Kyoto, 0.78 for Osaka, and 0.55 for Tokyo, indicating that more than 50% of rain will be successfully detected. Including no-rainfall events, the prediction accuracy (ACC) is more than 97.5% for every target area.

The dominant model structures showing superior performance were not determined. Most of the models show similar patterns, i.e., {high POD but low CSI} or {high CSI but low POD}. Overall, more convolution processes at the first convolutional layer (Green color) demonstrate a slightly higher POD, whereas fewer convolutional processes (Red color) demonstrate a slightly higher CSI. Also, more neurons at the fully connected layer ( mark) demonstrate a slightly higher POD and fewer neurons ( × mark) demonstrate a slightly higher CSI.

The results in Fig. 3 are from an input size of 16 × 16 × 6. The results from the input size of 8 × 8 × 6 were also evaluated and the best CSI models from these two different input data sets are compared in Table 2. In general, the prediction results with 16 gauge stations (16 × 16 × 6) show improved accuracy compared to the results with 8 gauge stations (8 × 8 × 6) from the aspect of the CSI (0.395 vs. 0.378, for Kyoto); however, the results with 8 stations show a slightly higher POD than the results with 16 stations (0.763 vs. 0.680, for Kyoto). It seems that the model with (8 × 8 × 6) of input tends to predict aggressively to achieve high POD, but it is resulted in a low CSI.

From all 27 Kyoto models, the mean CSIs with 8 and 16 stations are 0.358 and 0.348, respectively, while the mean PODs with 8 and 16 stations are 0.790 and 0.757, respectively (these are not shown in the table). It appears that the model with larger spatiotemporal data (16 × 16 × 6) is more stable than that with limited spatiotemporal data (8 × 8 × 6).

In Table 2, the validation results are described to demonstrate the performance objectively. The best model was selected from the 27 model structures based on the test results, and thus it should be verified with a different data set. In general, the performance of the testing results decreases during validation. In the validation results from Table 2, the developed model shows a CSI of 0.271 for Kyoto, 0.318 for Osaka, and 0.226 for Tokyo. The POD is still high, 0.665 for Kyoto, 0.78 for Osaka, and 0.55 for Tokyo, indicating that more than 50% of rain will be successfully detected. Including no-rainfall events, the prediction accuracy (ACC) is more than 97.5% for every target area.

### Table 2. Evaluation indices for the validation results with the best CSI models in the test

<table>
<thead>
<tr>
<th>Target Area</th>
<th>Number of Input Stations</th>
<th>Testing Results</th>
<th>Validation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ACC</td>
<td>POD</td>
</tr>
<tr>
<td>Kyoto</td>
<td>8</td>
<td>0.972</td>
<td>0.763</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.976</td>
<td>0.680</td>
</tr>
<tr>
<td>Osaka</td>
<td>8</td>
<td>0.971</td>
<td>0.750</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.978</td>
<td>0.643</td>
</tr>
<tr>
<td>Tokyo</td>
<td>8</td>
<td>0.971</td>
<td>0.683</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.979</td>
<td>0.647</td>
</tr>
</tbody>
</table>

The results in Fig. 3 are from an input size of 16 × 16 × 6. The results from the input size of 8 × 8 × 6 were also evaluated and the best CSI models from these two different input data sets are compared in Table 2. In general, the prediction results with 16 gauge stations (16 × 16 × 6) show improved accuracy compared to the results with 8 gauge stations (8 × 8 × 6) from the aspect of the CSI (0.395 vs. 0.378, for Kyoto); however, the results with 8 stations show a slightly higher POD than the results with 16 stations (0.763 vs. 0.680, for Kyoto). It seems that the model with (8 × 8 × 6) of input tends to predict aggressively to achieve high POD, but it is resulted in a low CSI.

From all 27 Kyoto models, the mean CSIs with 8 and 16 stations are 0.358 and 0.348, respectively, while the mean PODs with 8 and 16 stations are 0.790 and 0.757, respectively (these are not shown in the table). It appears that the model with larger spatiotemporal data (16 × 16 × 6) is more stable than that with limited spatiotemporal data (8 × 8 × 6).

In Table 2, the validation results are described to demonstrate the performance objectively. The best model was selected from the 27 model structures based on the test results, and thus it should be verified with a different data set. In general, the performance of the testing results decreases during validation. In the validation results from Table 2, the developed model shows a CSI of 0.271 for Kyoto, 0.318 for Osaka, and 0.226 for Tokyo. The POD is still high, 0.665 for Kyoto, 0.78 for Osaka, and 0.55 for Tokyo, indicating that more than 50% of rain will be successfully detected. Including no-rainfall events, the prediction accuracy (ACC) is more than 97.5% for every target area.

### 3.2 Missing & False Alarms

To understand the model performance in more detail, the prediction patterns of the 27 models in the Kyoto area (with 8 stations) were investigated for two events in the validation period, as shown in Fig. 4. In the figure, the solid black line in the upper panel is the amount of rainfall at the prediction target (Kyoto) and the dashed lines in the lower panel are the amount of rainfall at the other gauge stations in the input data. Furthermore, the red bar in the upper panel shows the number of models that missed the event and the blue bar in the lower panel shows the number of models that provided a false alarm.

![Figure 4. Model numbers of misses (red) and false alarms (blue) on events during the validation period.](image-url)
At the beginning of the event on June 21, 2017 (up in Fig. 4), all the 27 models missed the start of the event; however, immediately after the rain, two thirds of the models began to correctly predict the rainfall. During the event, all the 27 models successfully predicted rainfall 30 min ahead. At the end of the event, all the 27 models falsely predicted the rainfall events, lasting for several time steps. These continued false alarms (blue bars in the figure) are due to the atmospheric condition after the rainfall events. Even after the rain stops at the target point, the surrounding atmospheric condition still has a similar pattern to the condition during the rainfall, and thus the CNN model tends to falsely forecast after the event, resulting in a high FAR. With a short rainfall event on July 17 (down in Fig. 4), a similar pattern was found; there are continuous false alarms immediately after the event.

A strict perspective of the model performance can be considered as the model prediction is governed by rainfall on a target point. To investigate the prediction pattern based on the condition of rainfall at the target site, each prediction result (FO: correct prediction, XO: missed events, FX: false alarm) was classified into three cases: Case 1, rainfall at the target site; Case 2, no rain at the target site but rain at the other sites; Case 3, no rain at any site. Table 3 shows the classified results at Kyoto for the validation period (from the results of 8 station inputs).

First, the proportion of correct predictions (FO) for each case are as follows; Case 1: 92 events, Case 2: 51 events, and Case 3: 4 events. The CNN model is likely to predict correctly when there is rain already at the target or surrounding site. Very limited cases were successfully predicted if there was no rain at any site; i.e., a sudden rain at the target site.

The proportion of false alarm (FX) events for each case are as follows; Case 1: 81 events, Case 2: 182 events, Case 3: 61 events. A major proportion is from the case when there is no rain at the target site and there is rain at the other sites; this is also found in Fig. 4. Considering this prediction tendency of the model, minimizing the FAR is the main challenge to improving model accuracy. Since the largest proportion of false alarms came from Case 2 (no rain at the target site but rain at the other sites), avoiding this case will be our next challenge.

4. CONCLUDING REMARKS

The CNN based rainfall prediction model proposed by Suzuki et al.6 was extensively tested in three major cities in Japan, and it was confirmed that the model was able to detect rainfall occurrence stably with a 64 – 76% detection ratio for 30 min of forecasting lead-time. The developed model exhibits a high detection ratio; however, the high false alarm ratio levels remain an obstacle to improving model accuracy.

ACKNOWLEDGEMENTS

This research was supported by the KAKENHI of JSPS (Project No: 23760459).

REFERENCES


(Received June 30, 2020)
(Accepted August 28, 2020)