Prediction of Rainfall using Simplified Deep Learning based Extreme Learning Machines

Imam Cholissodin¹, Sutrisno²
¹,²Faculty of Computer Science, Computer Science, Brawijaya University, Malang, Indonesia
*Corresponding author, {¹imamcs*, ²trisno}@ub.ac.id

Received: 12 August 2018; Accepted: 28 October 2018

Abstract. Prediction of rainfall is needed by every farmer to determine the planting period or for an institution, eg agriculture ministry in the form of plant calendars. BMKG is one of the national agency in Indonesia that doing research in the field of meteorology, climatology, and geophysics in Indonesia using several methods in predicting rainfall. However, the accuracy of predicted results from BMKG methods is still less than optimal, causing the accuracy of the planting calendar to only reach 50% for the entire territory of Indonesia. The reason is because of the dynamics of atmospheric patterns (such as sea-level temperatures and tropical cyclones) in Indonesia are uncertain and there are weaknesses in each method used by BMKG. Another popular method used for rainfall prediction is the Deep Learning (DL) and Extreme Learning Machine (ELM) included in the Neural Network (NN). ELM has a simpler structure, and non-linear approach capability and better convergence speed from Back Propagation (BP). Unfortunately, Deep Learning method is very complex, if not using the process of simplification, and can be said more complex than the BP. In this study, the prediction system was made using ELM-based Simplified Deep Learning to determine the exact regression equation model according to the number of layers in the hidden node. It is expected that the results of this study will be able to form optimal prediction model.

Keywords: prediction, rainfall, ELM, simplified deep learning

1 Introduction
One of the regions in East Java Province which has high production level in agriculture and plantation sector is Malang Regency. Unfortunately, both sectors are vulnerable to crop failures when they enter rainy season with high rainfall (above 300 mm per month) and when entering the dry season with low rainfall (below 100 mm per month) [1][2]. So far, the efforts made by farmers to overcome this is just a reactive effort such as harvesting early. This effort is quite effective in reducing the magnitude of the loss, but it should be done proactively so that the failed harvest no longer occurs [3].

Planting calendar is one of the proactive efforts that farmers can use in determining the beginning of the best growing season, as has been done by Badan Penelitian dan Pengembangan Pertanian (Balitbangtan) of the Ministry of Agriculture every two times each year. In this case, Balitbangtan uses data forecasting rainfall every 10 days (“dasarian”) from Meteorology Climatology and Geophysics Agency (BMKG) to determine the entry and end of rainy or dry season [4]. Unfortunately BMKG in its operations often give a less accurate prediction [5], so consequently, the accuracy of Balitbangtan planting calendar is only reached 50% for the entire territory of Indonesia.
Some of the rainfall prediction methods that are often used by BMKG are Adaptive Neuro-Fuzzy Inference Systems (ANFIS) [7], wavelet transformation [8], and Autoregressive Integrated Moving Average (ARIMA) [9]. But the accuracy of some of the predicted methods mentioned above, BMKG said still not good about 70%.

In addition to the method often used BMKG. In this research proposed another popular method used for rainfall prediction is Deep Learning (DL) which is part of Neural Network (NN). However existing DL with backpropagation (BP) has a very high time of computing, so it is necessary to use another technique that can accelerate the learning speed DL without BP. Extreme Learning Machines (ELM) has a simpler structure, as well as non-linear approach capability and better convergence speed than BP [10][11][12]. So it’s suitable for use in Deep Learning [13][14]. The result of combining this method gives better performance than the conventional Deep Learning method. Therefore, in this research proposed method of Simplified Deep Learning-Based Extreme Learning Machine for rainfall prediction in Malang Regency in hopes can give more accurate rainfall result.

2 Method

2.1 Rainfall
Rainfall is the height of rainwater that collected in a place, non-flowing, non-volatile, and non-permeable. The unit of rainfall is millimeters (mm). One millimeter of rainfall means in one square meter in a flat place, collected water one millimeter or one liter [15]. Rainfall can be measured in various time periods. Short-term rainfall (hourly and day-to-day) is measured by the Meteorological Station, while the long-term (per 10 daily and per month) is measured by the Climatology Station. The Annual rainfall in Indonesia is shown in Fig. 1.

![Rainfall map in Indonesia](https://www.bmkg.go.id/?lang=EN)

2.2 Predictions
The difference between prediction and classification (in machine learning, classification is seen as one type of prediction). Based on Fig. 2, classification is used to predict class/category labels. Regression is building a model to predict the value (one target or multi-target) of the input data (with the feature length of the data). Then the difference between prediction versus forecasting (time period is the keyword to
distinguish between prediction and forecasting). And usually predictions are used to make short-term forecasts, while forecasting for the long term [16].

![Figure 2. Example visualization of regression vs. classification](image)

There are several approaches to prediction or forecasting, to build features as data patterns, for example on the exchange rate, i.e. [17][18]:

1. **Technical Analysis**
   - Involve exchange rate historical data to forecast future value.
   - The principle usually used by the technicalists, that the exchange rate has become a representative value of all relevant information affecting the exchange rate, the exchange rate will persist in a certain trend, and the exchange rate is a repetitive value repeatedly from the previous pattern.
   - But sometimes forecasting by technical analysis (technical forecasting) isn’t very helpful for long periods of time. Many researchers differ in opinion on the concept of that, whether to always use technical forecasting or not, although in general application in many cases, technical forecasting gives a good consistency.
   - Example:
     Initial data (Exchange rate data of IDR-USD in July 2015):
     | Date     | Exchange rate |
     |----------|---------------|
     | 5-Jul-15 | 13338         |
     | 6-Jul-15 | 13356         |
     | 7-Jul-15 | 13332         |
     | 8-Jul-15 | 13331         |
     | 9-Jul-15 | 13337         |
     | ...      | ...           |
     | 16-Jul-15| 13309         |

     The extraction results from initial data become, eg 2 data with 3 features (by technical analysis):

     | No | X1 (3 days ago) | X2 (2 days ago) | X3 (1 day ago) | Y (target) |
     |----|-----------------|-----------------|----------------|------------|
     | 1  | 13338           | 13356           | 13332          | 13331      |
     | 2  | 13356           | 13332           | 13331          | 13337      |

2. **Fundamental Analysis**
   - Based on the fundamental relationship between economic variables to the exchange rate, such as factors that affect the exchange rate, namely:
     - Inflation rate (INF)
     - Interest rates (INR)
     - Trade balance (log payment from the sale and purchase of goods and services between countries) (TB)
✓ Public Debt (PD)
✓ Ratio of Export Price and Import Price (REI), and
✓ Stability of Politics and Economics (SPE)

- Example:
The extraction results from initial data become, eg 2 data with 6 fundamental features (by fundamental analysis):

<table>
<thead>
<tr>
<th>No</th>
<th>X1 (INF)</th>
<th>X2 (INR)</th>
<th>..</th>
<th>X6 (SPE)</th>
<th>Y (target)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td>13338</td>
</tr>
<tr>
<td>2</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td>13356</td>
</tr>
</tbody>
</table>

2.3 Propose Method 1st: Modified feature extraction for each data of datasets like a time series or vector type to image matrix

Modified feature extraction for time series or vector data type to preprocessing data, so that data can be processed into the deep learning algorithm. There are several approaches to modified feature extraction, ie:

1. Repmat technique
   - The data vector (only features value) is repeated as much as the number of features, so it becomes a square matrix with size [num_of_features x num_of_features].
   - Example:
     Initial data:

     | No | X1 (3 days ago) | X2 (2 days ago) | X3 (1 day ago) | Y (target) |
     |----|-----------------|-----------------|---------------|------------|
     | 1  | 13338           | 13356           | 13332         | 13331      |

     The extraction results from initial data:

     | No | image matrix: a square matrix with size [num_of_features x num_of_features] | Y (target) |
     |----|-----------------------------------------------------------------------------|------------|
     | 1  | 13338 13356 13332 | 13338 13356 13332 | 13331  |

2. invS, and Spiral technique
   - The data vector (only features value) arranged following the pattern of the letter invS/Spiral on the square matrix with the size [num_of_features x num_of_features].
Example:
The extraction results \textit{invS} from initial data:

<table>
<thead>
<tr>
<th>No</th>
<th>image matrix: a square matrix with size [num_of_features x num_of_features]</th>
<th>Y (target)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13338 13356 13332</td>
<td>13331</td>
</tr>
<tr>
<td></td>
<td>13332 13356 13338</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13338 13356 13332</td>
<td></td>
</tr>
</tbody>
</table>

The extraction results \textit{Spiral} from initial data:

<table>
<thead>
<tr>
<th>No</th>
<th>image matrix: a square matrix with size [num_of_features x num_of_features]</th>
<th>Y (target)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13332 13356 13332</td>
<td>13331</td>
</tr>
<tr>
<td></td>
<td>13356 13338 13338</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13338 13332 13356</td>
<td></td>
</tr>
</tbody>
</table>

3. Custom technique

- The data vector (only features value) arranged following the pattern based set by user on the square matrix with the size [num_of_features x num_of_features] or on the specific matrix size.

2.4 Propose Method 2nd: Simplified Deep Learning based ELM

The Simplified Deep Learning based ELM (SDL-ELM) combines the performance of feature abstractions from convolution neural network (CNN) and training speeds of the Extreme Learning Machines. In Figure 3, the structure of the SDL-ELM consists of an input layer, an output layer and several hidden layers arranged as a single unity convolution layer, followed by a pooling layer. The amount of convolution and pooling layer, depends on the complexity of the case. Convolution layer consists of several groups of feature and pooling layer consists like a summary of several groups of feature [19][20]. Here are the detailed steps of SDL-ELM:

1. Create relevant map SDL-ELM (it's designed by the user) by combining Convolution, Sig/ReLU, Pooling, and Fully Connected process, as in the Fig. 3.
2. Set Parameter value.
   a. To normalization process of the feature value, eg:
      maxActual (mac) = 300;  minActual (mic) = 0;
      maxNorm (mao) = 1;  minNorm (mio) = 0;
   b. To convolution process. Set, for example with 3 kinds of filters, eg:
      where, 1st (conv11) : average filter, 2nd (conv12) : max filter, and 3rd (conv13) : std filter, std (standard deviation).
numFilter = 3; and, % number of padding (k), filter matrix size (k x k) on the convolution
k = 3;
c. To pooling process, eg:
where, % filter matrix size [windows_size x windows_size] on the pooling windows_size = 2;

Figure 3. Map Simplified Deep Learning CNN based ELM

3. Training Process
 a. Preprocessing
[numData,...
numFeature,target,norm]=FnPreProses('datatrainForcast.xlsx',...
mac, mic, ma, mio);
 o 1. Load data training, get numData and numFeature.
 o 2. Create “image matrix” to each single data (only features value) from dataset, eg using Repmat technique.
 o 3. Normalization of all “image matrix” data.

norm{i} = ((a{i}-mic)/(mac-mic))*(mao-mio)+mio;
where a{i} is each element matrix data i-th, and norm{i} define a matrix with size [numFeature x numFeature], eg

<table>
<thead>
<tr>
<th>Pixel position</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>253</td>
</tr>
<tr>
<td>2</td>
<td>131</td>
</tr>
<tr>
<td>3</td>
<td>259</td>
</tr>
<tr>
<td>4</td>
<td>95</td>
</tr>
</tbody>
</table>

b. Feature Abstraction with CNN (based Fig. 3).
 o 1. Convolution Init.
hC=FnConvDL(norm,numData,k);
if k=3, then expand edge norm image matrix (padding) with zero value as much as pad_size = (k-1)/2 = (3-1)/2=1, where k is odd number ≥ 3.

For example,

where the size of the green box is [k x k]

the result of filter 1\textsuperscript{st}: average filter

\[
\begin{pmatrix}
1 & 2 & 3 & 4 \\
1 & 0.2844 & \\
2 & \\
3 & \\
4 & 
\end{pmatrix}
\]

the result of filter 2\textsuperscript{nd}: max filter

\[
\begin{pmatrix}
1 & 2 & 3 & 4 \\
1 & 0.8433 & \\
2 & \\
3 & \\
4 & 
\end{pmatrix}
\]

the result of filter 3\textsuperscript{rd}: std filter

\[
\begin{pmatrix}
1 & 2 & 3 & 4 \\
1 & 0.3667 & \\
2 & \\
3 & \\
4 & 
\end{pmatrix}
\]

2. Sigmoid/ReLU
hA=FnSigDL(hC,numFilter,numData);
For example using the activation function sigmoid:

\[
hA_{1,1} = 1/(1+\exp(-hC_{1,1}))
\]  

3. Convolution In.
hC=FnConvInDL(hA,numData,k,numFilter);

4. Sigmoid/ReLU
hA=FnSigDL(hC,numFilter,numData);

5. Pooling
hP=FnPoolDL(hA,windows_size,numFilter,numData);

Count pad, where mI, nI is number of rows and column of hA{1}{1}.

\[
\text{padX} = (\text{ceil}(nI/windows\_size)\times windows\_size) - nI;
\]

\[
\text{padY} = (\text{ceil}(mI/windows\_size)\times windows\_size) - mI;
\]

\[
\text{mpoolI} = \sqrt{(mI+padY)\times(nI+padX)/windows\_size^2};
\]

\[
\text{npoolI} = \text{mpoolI};
\]
if padX > 0 or padY > 0, then padding hA[1] = padX expand edge after last column of matrix hA[1], padY expand edge after last row of matrix hA[1], eg padX = 2, padY = 2

\[
\text{padding}(hA[1]) = \begin{pmatrix}
0.4574 & 0.4390 & 0.4357 & 0.4552 & 0 & 0 \\
0.4376 & 0.4065 & 0.4059 & 0.4340 & 0 & 0 \\
0.4376 & 0.4065 & 0.4059 & 0.4340 & 0 & 0 \\
0.4574 & 0.4390 & 0.4357 & 0.4552 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

where the size of the black box is [windows_size x windows_size]

\[
hP[1] = \begin{pmatrix}
0.4574 & \ldots \\
\ldots & \ldots
\end{pmatrix}
\]

6. Convolution In.
\[hC = \text{FnConvInDL}(hP, \text{numData}, k, \text{numFilter});\]

7. Sigmoid/ReLU
\[hA = \text{FnSigDL}(hC, \text{numFilter}, \text{numData});\]

8. Pooling
if size (hA{i}) = [2 x 2], then set windows_size = 1
\[hP = \text{FnPoolDL}(hA, \text{windows_size}, \text{numFilter}, \text{numData});\]

c. Fully Connected with ELM (based Fig. 3).

9. Fully connected 1st
Eg, num_neuron_hidden_layer=5;
\[\text{[hFC11,W11,Bias11,Beta11]} = \text{FnELMtrainForecast}(hP, \text{target},...\]
\[\text{num_neuron_hidden_layer, numData, numFilter});\]
Below is illustrate how to get X(1,:) as first data to Fully connected 1st,

\[
X(1,:) = \text{merge from } hP[1], hP[2], hP[3]
\]

\[
hP[1] = \begin{pmatrix}
0.4495 & 0.4495 \\
0.4495 & 0.4495
\end{pmatrix}
\]

\[
hP[2] = \begin{pmatrix}
0.3950 & 0.3950 \\
0.3950 & 0.3950
\end{pmatrix}
\]

\[
hP[3] = \begin{pmatrix}
0.4351 & 0.4351 \\
0.4351 & 0.4351
\end{pmatrix}
\]

For example, the element is given in the form of a variable:
\[hP[1] = \begin{pmatrix}
a1 & c1 & d1 \\
1 & c1 & d1
\end{pmatrix}
\]
\[hP[2] = \begin{pmatrix}
a2 & c2 \\
b2 & c2
\end{pmatrix}
\]
\[hP[3] = \begin{pmatrix}
a3 & c3 \\
a3 & c3
\end{pmatrix}
\]

Here's how to generate X (1, :)
\[X(1,:) = [a1 b1 c1 d1 a2 b2 c2 d2 a3 b3 c3 d3]
\]
\[X(1,:) = [0.4495 0.4495 0.4495 0.4495 0.3950 0.3950 0.3950 0.3950 0.4351 0.4351 0.4351 0.4351]
\]

10. Fully connected 2nd
Eg, num_neuron_hidden_layer=7;
\[\text{[hFC12,W12,Bias12,Beta12]} = \text{FnELMtrainForecast}(hP, \text{target},...\]
\[\text{num_neuron_hidden_layer, numData, numFilter});\]

11. Fully connected 3rd
Eg, num_neuron_hidden_layer=4;
\[\text{[hFC13,W13,Bias13,Beta13]} = \text{FnELMtrainForecast}(hP, \text{target},...\]
\[\text{num_neuron_hidden_layer, numData, numFilter});\]

4. Testing Process
a. Preprocessing
[numData2,...
numFeature2,target2,norm2]=FnPreProses('datatestForcast.xlsx',...
mac, mic, mao, mio);

b. Feature Abstraction with **CNN** (based Fig. 3).
   o 1. Convolution Init.
      \( hC2=\text{FnConvDL}(\text{norm2},\text{numData2},k) \);
   o 2. **Sigmoid/ReLU**
      \( hA2=\text{FnSigDL}(hC2,\text{numFilter},\text{numData2}) \);
   o 3. Convolution In.
      \( hC2=\text{FnConvInDL}(hA2,\text{numData2},k,\text{numFilter}) \);
   o 4. **Sigmoid/ReLU**
      \( hA2=\text{FnSigDL}(hC2,\text{numFilter},\text{numData2}) \);
   o 5. Pooling
      \( hP2=\text{FnPoolDL}(hA2,\text{windows}_\text{size},\text{numFilter},\text{numData2}) \);
   o 6. Convolution In.
      \( hC2=\text{FnConvInDL}(hP2,\text{numData2},k,\text{numFilter}) \);
   o 7. **Sigmoid/ReLU**
      \( hA2=\text{FnSigDL}(hC2,\text{numFilter},\text{numData2}) \);
   o 8. Pooling
      if size \( (hA2[i][j]) = [2 \times 2] \), then set \( \text{windows}_\text{size} = 1 \)
      \( hP2=\text{FnPoolDL}(hA2,\text{windows}_\text{size},\text{numFilter},\text{numData2}) \);

c. Fully Connected with **ELM** (based Fig. 3).
   o 9. Fully connected 1\text{st}
      \[ \text{vEvaluation1,Ytest\_predict1}]=... \]
      \( \text{FnELMtestForcast}(hP2,\text{target2},... \]
      \( \text{W11,Bias11,Beta11,\text{numData2},\text{numFilter}}) ; \)
   o 10. Fully connected 2\text{nd}
      \[ \text{vEvaluation2,Ytest\_predict2}]=... \]
      \( \text{FnELMtestForcast}(hP2,\text{target2},... \]
      \( \text{W12,Bias12,Beta12,\text{numData2},\text{numFilter}}) ; \)
   o 11. Fully connected 3\text{rd}
      \[ \text{vEvaluation3,Ytest\_predict3}]=... \]
      \( \text{FnELMtestForcast}(hP2,\text{target2},... \]
      \( \text{W13,Bias13,Beta13,\text{numData2},\text{numFilter}}) ; \)

d. Voting to get final result
   Get \( \text{Ytest\_predict} \) by minimum \( \text{vEvaluation} \) from all “Fully Connected”
   \( \text{CompareEvaluasi}=[\text{vEvaluation1 vEvaluation2 vEvaluation3}] ; \)
   \( \text{[vMin, idxMin]}=\text{min(CompareEvaluasi)} ; \)
   So, the last step of SDLCNN-ELM algorithm get the best result from
   Fully connected \( \text{idxMin-} \text{th} \) with Mean absolute deviation (MAD) = \( \text{vMin} \).
   Link our full code project above for demo, please see at our webpage:

3 Results and Discussion

Based on Fig. 4, the SDLCNN-ELM algorithm on rainfall data with a limited amount is using 2 types of features to merger, namely the first feature extraction from CNN combined with the second feature extraction, namely the original features, so it is obtained the results of the majority of the minimum value of MAD are more dominant than using conventional ELM which only uses the original features. This shows that the
characteristics of feature extraction with CNN focus more on contributing to deeper hidden pattern recognition that cannot be quantized or represented by the original features. Feature extraction with CNN uses several filters, such as average filter, max filter, and STD filter, because this technique is a major part of the Deep Learning algorithm. While the original features are only visible from the outside. The improvement results of SDLCNN-ELM are able to reduce errors 1.117 from the average MAD value when compared to ELM standard.

**Figure 4. Test Result based MAD value, SDLCNN-ELM versus ELM**

**Figure 5. Time SDLCNN-ELM versus ELM**
Then, in Fig. 5 the comparison graph is shown, if the experiment is increasing, the two methods both require greater computational time, which can still be said to be comparable. This is because space memory as a resource used to process and store results for each iteration of the experiment is longer and larger. So that the computation speed is slower, it can be seen by the difference in the minimum average value of the computation time as 0.1194 seconds.

4 Conclusion

The SDLCNN-ELM algorithm is a collection of deep neural network families that have been proven to produce smaller error rate compared to pure ELM methods for prediction of rainfall. This hope for the future will be very helpful in solving wider and more complex problems. In future research can be more focused on exploring the hidden features of a feature that appears in any case with a variety of representative filtering techniques and combines the hidden features with features that appear outside. And also how to find the optimal map architecture as in Fig. 3, for example using Particle Swarm Optimization as in previous research [21]. Then related to computing time, in fact, this can be overcome by how the data structure is used, or involves parallel techniques or run them on server computers with very supportive specifications. While access from clients can be of any type of device, anywhere and anytime can process and monitor the results of the process.

References


p-ISSN: 2540-9433; e-ISSN: 2540-9824


