Performance Analysis of Hybrid Architectures of Deep Learning for Indonesian Sentiment Analysis

Theresia Gowandi, Hendri Murfi(✉), and Siti Nurrohmah

Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Indonesia, Depok 16424, Indonesia
hendri@ui.ac.id

Abstract. Sentiment analysis is one of the fields of Natural Language Processing that builds a system to recognize and extract opinions in the form of text into positive or negative sentiment. Nowadays, many researchers have developed methods that yield the best accuracy in performing analysis sentiment. Three particular models are Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), which have deep learning architectures. CNN is used because of its ability to extract essential features from each sentence fragment, while LSTM and GRU are used because of their ability to memorize prior inputs. GRU has a more straightforward and more practical structure compared to LSTM. These models have been combined into hybrid architectures of LSTM-CNN, CNN-LSTM, and CNN-GRU. In this paper, we analyze the performance of the hybrid architectures for Indonesian sentiment analysis in e-commerce reviews. Besides all three combined models mentioned above, we consider one more combined model, which is GRU-CNN. We evaluate the performance of each model, then compare the accuracy of the standard models with the combined models to see if the combined models can improve the performance of the standard. Our simulations show that almost all of the hybrid architectures give better accuracies than the standard models. Moreover, the hybrid architecture of LSTM-CNN reaches slightly better accuracies than other hybrid architectures.

Keywords: CNN · Deep learning · GRU · LSTM

1 Introduction

In September 2019, GlobalWebIndex reported that 79% of Indonesia's internet users aged 16 to 64 made online purchases via mobile phones. The top mobile e-commerce applications are Tokopedia, Shopee, and Lazada [1]. Also, thousands of online opinions were written by customers every day on many platforms, e.g., Google Play. They posted about products they paid for, services that they used, and items they took into consideration. Also, customers will use these reviews before they make a decision. These applications developers got a new challenge to disclose what is being said about their application or the products and services they advertise. The standard technique would be checking thousands of reviews manually. However, nowadays, there is a popular technique to manage this huge amount of opinionated text data by using sentiment analysis. Sentiment analysis, also called opinion mining, is a field that analyzes people's opinions and sentiments toward certain entities, like products or services, to find its polarity between positive or negative sentiment [2].

Sentiment analysis can be done manually by looking through every single text one by one and decide its polarity on our own. But this approach is time-consuming and prone to human error. The solution to this problem is the machine learning approach for sentiment analysis. Machine learning develops methods that can automatically detect patterns in data and use them to predict future data [3]. Sentiment analysis is a classification problem in machine learning that needs to classify text into positive or negative sentiment. There are many studies of machine learning for text classification like Support Vector Machine and Naïve Bayes [4, 5]. However, deep learning is currently commonly used in sentiment analysis problems because it can extract features from unstructured data very well. Popular deep learning techniques that have been widely used in sentiment analysis are Convolutional Neural Network [6], Long Short-Term Memory [7], and Gated Recurrent Unit [8].

Previous researches have shown that CNN, LSTM, and GRU have a good performance for sentiment analysis. Also, Wang, Jiang, Luo [9] have shown that combined architecture CNN-LSTM and CNN-GRU models perform better than the CNN and LSTM models alone. They combined the model because they want to take advantage of both the local feature extraction from CNN and long-distance dependencies from RNN. Then, Sosa [10] proved that the combined LSTM-CNN model achieved better performance than the standard models and CNN-LSTM model.

In this paper, we analyze the performance of the hybrid architectures for Indonesian sentiment analysis in e-commerce reviews. Besides all three combined models mentioned above, which are LSTM-CNN, CNN-LSTM, CNN-GRU, we consider one more combined model, GRU-CNN. We evaluate the performance of each model, then compare the accuracy of the standard models with the combined models to see if the combined models can improve the performance of the standard. Our simulations show that almost all of the hybrid architectures give better accuracies than the standard models. Moreover, the hybrid architecture of LSTM-CNN reaches slightly better accuracies than other hybrid architectures.

The rest of the paper is organized as follows: In Sect. 2, we present materials and methods. In Sect. 3, we discuss the results of the simulations. Finally, we give a conclusion of this research in Sect. 4.

2 Materials and Method

2.1 Dataset

There are three data sets used in this study. Three of them are Indonesian e-commerce application reviews extracted from the Google Play Store website, which are Tokopedia, Shopee, and Lazada. The detail of the data sets is listed in Table 1.

Table 1. Datasets used in this study.

Dataset	Positive sentiment	Negative sentiment	Total data
Tokopedia	1697	2663	4360
Shopee	2562	2435	4997
Lazada	2976	3304	6280

2.2 Preprocessing

In this process, the raw data set is processed from unstructured textual data to structured textual data ready for the model implementation. Firstly, the data need to be cleaned from unused punctuations and emoticons. Second, convert all alphabets to lowercase. Third, remove stop words or commonly used words. And last, the text needs to be modified into a vector of representation by tokenizing, sequencing, padding, and word embedding. Tokenizing is the process of dividing text in sentence form into tokens or parts per word. Sequencing is the process of creating an index for each unique word with an integer sorted based on the most used words in a dataset. Padding is the process of making all the lengths of each document in the dataset the same. Word embedding is the process of representing word indexes using vector representations. Last, one processing process will be carried out on label/sentiment, namely one-hot encoding. One hot encoding is the process of converting categorical variables into numeric variables. Table 2 shows the steps of preprocessing text data. Table 3 shows the preprocessing of labels.

Table 2. Preprocessing text data.

Steps	Processing Results
Data	Josss.. tingkatkan pelayanannya... dan perketat sailler yang curang/nakal... barang gak layak dijual... di jual.... yg di utamakan pelanggan puas... gak kecewa... 👍👍👍cek barang sebelum di kirim... Terimakasih
Data cleaning	tingkat layan ketat jual curang nakal barang layak jual jual utama langgan puas kecewa cek barang kirim terima kasih
Data tokenizing	tingkat, layan, ketat, jual, curang, nakal, barang, layak, jual, jual, utama, langgan, puas, kecewa, cek, barang, kirim, terima, kasih
Data sequencing	147, 51, 718, 61, 428, 329, 2, 682, 9, 9, 333, 107, 127, 22, 88, 2, 4, 12, 8
Data padding	0, 0, 0, 0, 0, 147, 51, 718, 61, 428, 329, 2, 682, 9, 9, 333, 107, 127, 22, 88, 2, 4, 12, 8

2.3 Convolutional Neural Network (CNN)

CNN is a deep learning model that performs well in text data classification using feature extraction layers. Feature extraction consists of convolutional layers and pooling layers

Table 3. Preprocessing labels.

Sentiment	Processing results
0	[0, 1]
1	[1, 0]

that will choose the best features. The first step is processing the input matrix with a convolutional layer which consists of several filters $\mathbf{W}$. Each row in the input matrix $X_{i:n}$ is an embedding vector with d dimension from the i-th word of a sentence that has n words. Filters play a role in learning the essential features of sentence fragments. The length of a sentence fragment is called the region size. Every region size has the same number of filters. The region size and number of filters will be the hyperparameter of the model and will be adjusted to get an optimal accuracy. The convolution operation involves a filter $\mathbf{W}$ that will be applied to a window of s words (region size) to produce a feature. Suppose, input matrix $X_{i:i+s-1}$ is the concatenation of words $\mathbf{x}_i, \ldots, \mathbf{x}_{i+s-1}$. Then, a feature c_i is defined as

$$c_i = f(X_{i:i+s-1} \cdot \mathbf{W}) \tag{1}$$

with f as the activation function. The filter will be applied to all possible sentence fragments, $X_{1:s}, X_{2:s+1}, \ldots, X_{n-s+1:n}$ to produce a feature map $\mathbf{c} = [c_1, c_2, \ldots, c_{n-s+1}]$. Then, feature map $\mathbf{c}$ will enter the pooling layer that will choose the representative value on each feature map. Max pooling layer will take the most important feature with the maximum value $\hat{c} = \max\{\mathbf{c}\}$ from each feature map [6].

2.4 Long Short-Term Memory (LSTM)

LSTM proposed by Hochreiter & Schmidhuber in 1997 [11] is a particular type of Recurrent Neural Network because it can carry out learning on long-term dependencies. It means that LSTM can remember information in the long run. The cell state is where memory is stored in LSTM. LSTM can delete or add information to the cell state using the gate mechanisms, which are forget gate ($\mathbf{f}_t$), input gate ($\mathbf{i}_t$), dan output gate ($\mathbf{o}_t$). The gate is composed of one layer with a sigmoid activation function and a simple linear operation. The sigmoid activation function will map the results into a range of 0 and 1 values that control how much information can pass. Value 0 means to forget the information, while value 1 means to remember the information.

At time t, the first step taken in LSTM's is to determine the information that should be removed from the previous cell state ($\mathbf{c}_{t-1}$) and pass the necessary information. This decision was made by forget gate ($\mathbf{f}_t$). Then, LSTM will determine the new information that will be stored in the cell state ($\mathbf{c}_t$). This stage consists of 2 steps. First, we will determine the elements in the cell state to be updated. The input gate performs this step ($\mathbf{i}_t$). Second, the tanh function will form a candidate vector for cell state ($\tilde{\mathbf{c}}_t$) which may be added to the cell state. LSTM combines these two steps to update the cell state.

After processing the information with the previous steps, the old cell state ($\mathbf{c}_{t-1}$) will then be updated to the new state cell ($\mathbf{c}_t$). Finally, LSTM will determine the output

($\mathbf{h}_t$) based on the updated cell state. The output gate carries out this step ($\mathbf{o}_t$) using the sigmoid function will determine which part of the cell state will be used as the output [12]. The information of $\mathbf{h}_t$ then will be passed to the next unit. Figure 1 shows an illustration of LSTM's unit.

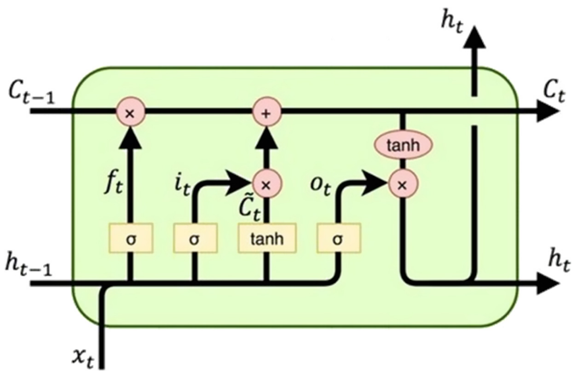

Fig. 1. Illustration of LSTM's unit [15]

$$\mathbf{f}_t = \sigma(\mathbf{w}^f \times \mathbf{x}_t + \mathbf{u}^f \times \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{2}$$

$$\mathbf{i}_t = \sigma(\mathbf{w}^i \times \mathbf{x}_t + \mathbf{u}^i \times \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{3}$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{w}^c \times \mathbf{x}_t + \mathbf{u}^c \times \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{4}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{5}$$

$$\mathbf{o}_t = \sigma(\mathbf{w}^o \times \mathbf{x}_t + \mathbf{u}^o \times \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{6}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{7}$$

2.5 Gated Recurrent Unit (GRU)

The GRU proposed by Cho et al. in 2014 [13] was another modification of the Recurrent Neural Network but simpler than the LSTM model, offered almost comparable performance to LSTM, was more computationally efficient, and was quite popular. The gate mechanism that GRU uses is the update gate ($\mathbf{z}_t$) and reset gate ($\mathbf{r}_t$), which are used to determine the output. The gate is composed of one layer with a sigmoid activation function that will map the results into a range of 0 and 1 values that control how much information can pass. Value 0 means to forget the information, while value 1 means to remember the information.

At time t, the first step taken in GRU is to determine how much past information still needs to be passed on to the next cell. The update gate made this decision ($\mathbf{z}_t$). Then, the reset gate ($\mathbf{r}_t$) is responsible for deciding how much past information should be forgotten. Next, we will calculate the candidate hidden state ($\mathbf{h}_t^{'}$) using the information from the reset gate to remember previous information that is useful. As $\mathbf{r}_t$ approaches 0, $\mathbf{h}_t^{'}$ will only process the current input. As $\mathbf{r}_t$ approaches 1, $\mathbf{h}_t^{'}$ will store the current input and the previous time output. This process is still a candidate because the results will be forwarded to calculate the final output values in the hidden state $\mathbf{h}_t$.

The final calculation is for the hidden state ($\mathbf{h}_t$), the vector that holds the information from the current unit will pass the information on to the next unit. The hidden state ($\mathbf{h}_t$) will use the information from the update gate ($\mathbf{z}_t$) to determine the required information from the previous hidden state ($\mathbf{h}_{t-1}$) and the candidate hidden state ($\mathbf{h}_t^{'}$). As $\mathbf{z}_t$ approaches 0, the value of $\mathbf{h}_t$ approaches the value of $\mathbf{h}_t^{'}$, in other words, there will be a change in the information on the output. As $\mathbf{z}_t$ approaches 1, the information from $\mathbf{x}_t$ is ignored and effectively ignores the time step t in the process so it will only retain the old information [12]. The information of $\mathbf{h}_t$ then will be passed to the next unit. Figure 2 shows an illustration of GRU's unit.

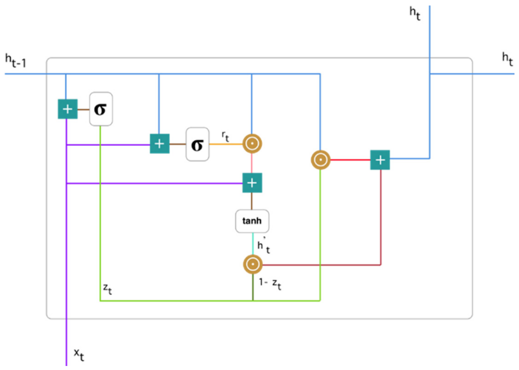

Fig. 2. Illustration of GRU's unit [16]

$$\mathbf{z}_t = \sigma(\mathbf{w}^z \times \mathbf{x}_t + \mathbf{u}^z \times \mathbf{h}_{t-1} + \mathbf{b}_z) \quad (8)$$

$$\mathbf{r}_t = \sigma(\mathbf{w}^r \times \mathbf{x}_t + \mathbf{u}^r \times \mathbf{h}_{t-1} + \mathbf{b}_r) \quad (9)$$

$$\mathbf{h}_t^{'} = \tanh(\mathbf{w}^h \times \mathbf{x}_t + \mathbf{r}_t \odot (\mathbf{u}^h \times \mathbf{h}_{t-1})) \quad (10)$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \mathbf{h}_t^{'} \quad (11)$$

2.6 RNN-CNN

The RNN-CNN model is a combined model of the RNN and CNN models. The combined RNN-CNN model architecture refers to both LSTM-CNN and GRU-CNN models. The first step is processing the input with the RNN layer to learn the feature representation and sequence on the data. Then, the output from the RNN layer will be used as input for the CNN layer, which will look for pairs of essential features in the data. Each of the standard models has its advantages. RNN pays attention to word order, while CNN can select important features of ordered word phrases. So, these advantages are expected to maximize the data learning process. Figure 3 shows an illustration of the combined RNN-CNN model's architecture.

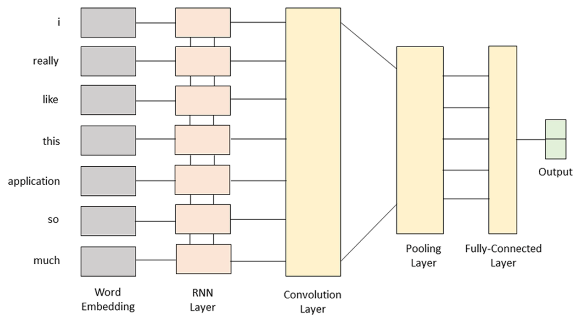

Fig. 3. The architecture of the combined RNN-CNN model [10]

2.7 CNN-RNN

In general, the combined CNN-RNN model has a similar concept to the combined RNN-CNN model. The difference lies in the orders of the model. The first step is processing the input with the CNN layer to select word phrases. Then, the output from the CNN layer will be used as input for the RNN layer, which will create a new representation for the data

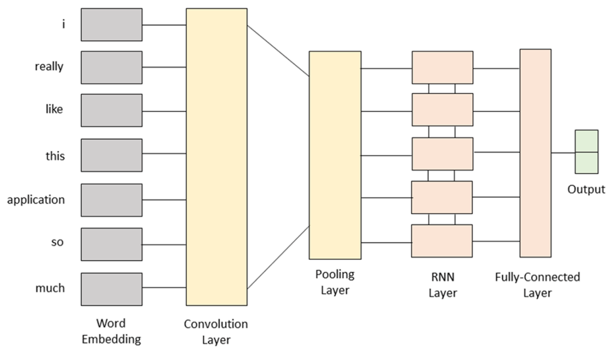

Fig. 4. The architecture of the combined CNN-RNN model [10]

sequence. The combined CNN-RNN model architecture refers to both CNN-LSTM and CNN-GRU models. Figure 4 shows an illustration of the combined CNN-RNN model's architecture.

3 Result and Discussion

3.1 Experimental Setup

After the data is being preprocessed, the following process is to implement each model using the prepared dataset to find the accuracy of each model. For each simulation, the dataset is divided into 80% training data and 20% testing data. Training data is used to build the model, and test data is used to find the model's accuracy. The training process will be done in 25 epochs and 32 batch sizes using the Adam optimization technique with a learning rate of 0.0001. The list of hyperparameters will be tuned to get the best accuracy in each simulation, and the hyperparameter's candidate can be seen in Table 4 [14].

After the data is being preprocessed, we implement the combined LSTM-CNN, CNN-LSTM, GRU-CNN, CNN-GRU model, and standard CNN, LSTM, GRU model with the optimized hyperparameter to get their accuracy. Then, compare and analyze the results.

Table 4. List of hyperparameter and the candidate values.

Layer	Hyperparameter	Candidate values
Word embedding	Embedding size	64; 100; 128
RNN	Unit	100; 150; 200
	regularization L2 kernel	0.01; 0.001
	regularization L2 recurrent	0.01; 0.001
CNN	Region size	200; 250; 300
	Number of filters	3; 4; 5
	regularization L2	0.01; 0.001
Fully connected	regularization L2	0.01; 0.001

3.2 Results

Results accuracy of the combined and standard models are listed in Table 5. Almost all combined models have better performance than the standard models. Only the CNN-GRU model has a lower accuracy than the CNN model in Tokopedia and Lazada datasets. The LSTM-CNN, CNN-LSTM, GRU-CNN performed better than the standard models.

Looking through the results, it seems that our intuition made earlier was correct. Almost all of our combined models are higher than the standard models. The combined model improved the model implementation because each model's benefit complements the other. While CNN worked to find the local features, RNN managed to use its memory ability to improve performance.

Furthermore, using the RNN model at the beginning of the combined model and continued with CNN provides higher accuracy than the combined CNN-RNN models for the Tokopedia and Lazada datasets. It looks like the CNN-RNN models lost some of the sequence information. Suppose the order from the output of the convolutional layer doesn't give any information. In that case, the LSTM layer won't be able to use its ability and works the same as the fully connected layer. On the contrary, the RNN-CNN models give a better result because LSTM and GRU managed to find information in the present and the past. Then, CNN work to find the local features.

Table 5. Results of the combined and standard models.

Model	Tokopedia	Shopee	Lazada
LSTM-CNN	**83.03%**	82.12%	**82.99%**
CNN-LSTM	82.82%	**82.40%**	82.85%
GRU-CNN	82.98%	81.82%	82.88%
CNN-GRU	82.13%	82.30%	82.32%
LSTM	82.71%	82.08%	82.83%
GRU	81.93%	81.72%	82.29%
CNN	82.45%	81.60%	82.69%

4 Conclusion

In this study, we have experimented to find each model's best performance and analyze whether the combined models are better than the standard models. As a result, the average accuracy produced by the combined LSTM-CNN, CNN-LSTM, GRU-CNN, CNN-GRU model on the three datasets is 82.71%, 82.69%, 82.56%, and 82.25%, respectively. Almost all of the combined models have better accuracy than the standard models. Based on our combined models, only the combined CNN-GRU model is slightly below the CNN model but has slightly better accuracy than the GRU model. Meanwhile, the other three combined models obtain slightly better accuracy than the standard models. The limitations of each of the data sets and tools developed in the research study have indicated a recommendation for further work, namely to implement the models on more data sets to be able to see a better learning on the model.

References

1. Kemp, S., Moey, S.: Digital 2019 Spotlight: E-commerce in Indonesia, https://datareportal.com/reports/digital-2019-ecommerce-in-indonesia. Accessed 01 Oct 2019
2. Liu, B.: Sentiment analysis and opinion mining. In: Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers (2012)

3. Murphy, K.: Machine Learning: A Probabilistic Perspective. MIT Press, London (2012)
4. Devika, M.D., Sunitha, C., Ganesh, A.: Sentiment analysis: a comparative study on different approaches. Procedia Comput. Sci. **87**, 44–49 (2016)
5. Mariel, W.C.F., Mariyah, S., Pramana, S.: Sentiment analysis: a comparison of deep learning neural network algorithm with SVM and Naïve Bayes for Indonesian text (2018)
6. Kim, Y.: Convolutional neural networks for sentence classification. In: Association for Computational Linguistics (ACL), Doha (2014)
7. Hassan, A., Mahmood, A.: Deep learning for sentence classification. In: IEEE Long Island Systems, Applications and Technology Conference (LISAT), New York (2017)
8. Biswas, S., Chadda, E., Ahmad, F.: Sentiment analysis with gated recurrent units. In: Advances in Computer Science and Information Technology (ACSIT), India, vol. 2, no. 11, pp. 59–63 (2015)
9. Wang, X., Jiang, W., Luo, Z.: Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In: COLING 2016 - 26th International Conference on Computational Linguistics, Osaka, pp. 2428–2437 (2016)
10. Sosa, P.M.: Twitter sentiment analysis using combined LSTM-CNN models. In: Academia Edu, pp. 1–9 (2017)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Zhang, A., Lipton, Z., Li, M., Smola, A.: Dive into Deep Learning (2020)
13. Cho, K., Merrienboer, B., van Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder-decoder approaches, pp. 103–111 (2015)
14. Wijaya, M.: Analisis Kinerja Modifikasi Model Gabungan Long Short-Term Memory dan Convolutional Neural Network untuk Analisis Sentimen Berbahasa Indonesia. Universitas Indonesia, Indonesia (2020)
15. Olah, C.: Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed 30 Oct 2019
16. Kostadinov, S.: Understanding GRU Networks. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be. Accessed 31 Oct 2019