Accredited by National Journal Accreditation (ARJUNA) Managed by
Ministry of Research, Technology, and Higher Education, Republic Indonesia with Second Grade (Peringkat 2, Sinta 2)
since year 2017 to 2021 according to the decree No. 10/E/KPT/2019.

Published online on the journal's webpage: **http://jurnal.iaii.or.id**

RESTI JOURNAL
(System Engineering and Information Technology)
Vol. 5 No. 3 (2021) 584 - 593 ISSN Electronic Media: 2580-0760

Identifying Emotion on Indonesian Tweets using Convolutional Neural Networks

Naufal Hilmiaji[1], Kemas M Lhaksmana[2], Mahendra D Purbolaksono[3]
[1,2,3]School of Computing, Telkom University
[1]naufalhilmiaji@student.telkomuniversity.ac.id, [2]kemasmuslim@telkomuniversity.ac.id,
[3]mahendradp@telkomuniversity.ac.id

Abstract

Identifying emotion out of text has become a research interest in natural language processing and other related fields, especially with the advancement of deep learning methods for text classification. Despite some effort to identify emotion on Indonesian tweets, its performance evaluation results have not achieved acceptable numbers. To solve this problem, this paper implements a classification model using a convolutional neural network (CNN), which has demonstrated expected performance in text classification. To easily compare with the previous research, this classification is performed on the same dataset, which consists of 4,403 tweets in Indonesian that were labeled using five different emotion classes: anger, fear, joy, love, and sadness. The performance evaluation results achieve the precision, recall, and F1-score at respectively 90.1%, 90.3%, and 90.2%, while the highest accuracy achieves 89.8%. These results outperform previous research that classifies the same classification on the same dataset.

Keywords: emotion, text classification, twitter, CNN.

1. Introduction

The rise of social media allows humans to express their emotions by sharing their moments with their fellows. Twitter is a social media with the highest user growth each year. In Indonesia, Twitter has ranked fourth by the most popular social media with at least seven million active users by the end of March 2021 [1]. It provides texts as their main feature to communicate, which are called tweets. As the media of expression, tweets can have various information, including human emotion. Users often post tweets whenever they have something emotional inside them. Likewise, Indonesian social media users express themselves through Twitter [2]. After all, when compared to other social media that are more concerned with visual images, on Twitter they can be more expressive because they only focus on text or tweets. Also, they mentioned that Twitter users tend to be more open-minded. With these advantages, they will never feel that when they wanted to reveal themselves, other users would respond carelessly or harshly to their remarks. The form of self-expression they do is what they feel at the time, such as anxiety and deep thoughts. Apart from that, at the same time, the emotions felt by the users were also expressed via tweets. These issues led researchers to find ways of gaining knowledge through user tweets. The information gained is about analyzing their sentiments, including their emotions.

Analyzing emotions is part of a psychology field, although, in natural language processing (NLP), emotion analysis or emotion mining has gone through years of research. Analyzing emotions can be done by learning the humans' nonverbal communication, such as facial expressions, body gestures, eye contact, touch, space, and voice [3]. Therefore, professional human resources in the field of psychology are needed to avoid misinterpretation. Due to the demanding cost of hiring psychologists, researchers are keen to find alternative ways. Machine learning is one of the most promising approaches for identifying emotion through text classification. It learns from the data (supervised) and predicts the corresponding classes of the text provided in the data. In addition, the ability of a machine or computer to understand emotions is critical to the success of several other applications. For instance, in the domain of customer service, Twitter gains prominence where customers are expected to have quick responses. Text emotion classification can be applied to monitor the cyberbullying, depression, and desperation in social media and prevent them from hurting themselves. It also helps companies to create an automated system of

Accepted by editor: 18-05-2021 | Final Revision: 24-06-2021 | Online Publication: 26-06-2021

classifying applicants' personalities through texts to seek talents who fit with their corporate culture in the talent recruitment process [4].

According to the past study of tweets emotion analysis, using Multinomial Naïve Bayes (MNB) with Laplace Smoothing has obtained good performance results. It combines MNB Classifier with multiple feature extraction methods, such as n-grams, POS Tag, and adjectives. Unigrams has the highest accuracy of 95% for the tweets and 67% for the SMS [5]. The same method was applied in another study of classifying sentiment based on movie review dataset by combining TF-IDF as the feature extraction method. It produced the best accuracy of 85.16%, meanwhile using SVM produced the best F1-Score of 84.9% [6, 7].

A study proposed the combination of two machine learning methods CNN and SVM to construct a text sentiment analysis model by using a Word2Vec as the feature extraction. This study used the data from NLPCC2014, an emotional analysis evaluation task data set based on the deep learning method. It consisted of 10000 rows of training data, with 50% labeled as positive and 50% labeled as negative. The testing data consisted of 2500 rows, with 50% labeled as positive and 50% labeled as negative. The result shows positive classes have an 89% F1-score, while negative classes have 88.6% [8].

In 2019, a study proposed a CNN model architecture for text sentiment classification using the English Movie Review dataset. The model consisted of two consecutive convolution layers. The first layer stores the local information to the second layer, while the second layer extracts features from the contextual information. They have produced relatively high-performance results on a relatively long text. The accuracy of binary and ternary classification respectively 81% and 68% [9].

Research on emotion classification on Indonesian Twitter data compared multiple machine learning methods, such as Logistic Regression, SVM, and Random Forest then combined them with multiple feature extraction methods, such as Bag-of-Words (BOW), Word2Vec (WV), and FastText (FT). The result is acceptable, getting an F1-score of 68.39%. Finally, they considered adding three more methods, Emotion Lexicons, Orthographic, and POS Tag. The result is slightly higher than the previous one, getting an F1-score of 69.73%. They also provided a dataset called "Indonesian Twitter Emotion Dataset" as the result of this study [10].

A later study reproposed the method used in the [10] by using Deep Learning LSTM-GloVe. Tweets are tokenized once it enters the input layer into a sequence of integers. The padding of zeros is added to match the length of each tweet. The system will then build an embeddings index to retrieve the index from the GloVe dictionary. The parameters used for experiments are: (1) Learning rate: 0.005 and 0.001, (2) Dropout: 0.25 and 0.5, (3) Optimizer: SGD and Adam's. With the use of 50 epochs and 100 batch size, precision, recall, and F1-score were obtained respectively at 33%, 38%, and 35%. The accuracy of the model is 46% [11].

Based on the research by [10, 11], with the same dataset, the Logistic Regression method leads the F1-score result at 69.73%, significantly higher than the LSTM-GloVe method. This may happen since deep learning methods are created to model the distribution that underlies the training data. All cases and possibilities are needed in the training process to achieve state-of-the-art accuracy. Therefore, deep learning methods often require a large number of data, much more than conventional ones. Meanwhile, both studies only used 4.403 rows of data. In addition, the architecture of the model can also affect the training performance.

Considering this research is using the deep learning CNN method, our main reference is from [11]. However, their results are still far below the average. The authors stated that it is caused by underfitting. Although, it is unclear the reason they allowed the model to underfit when the learning process is being done. Therefore, the purpose of this research is to optimize the deep learning CNN method by implementing some experiments on model architecture (e.g., hyperparameter tuning) to get the highest possible performance results and to avoid overfits and underfits. The results will then be compared to [11] as the baseline of our model prediction.

2. Research Method

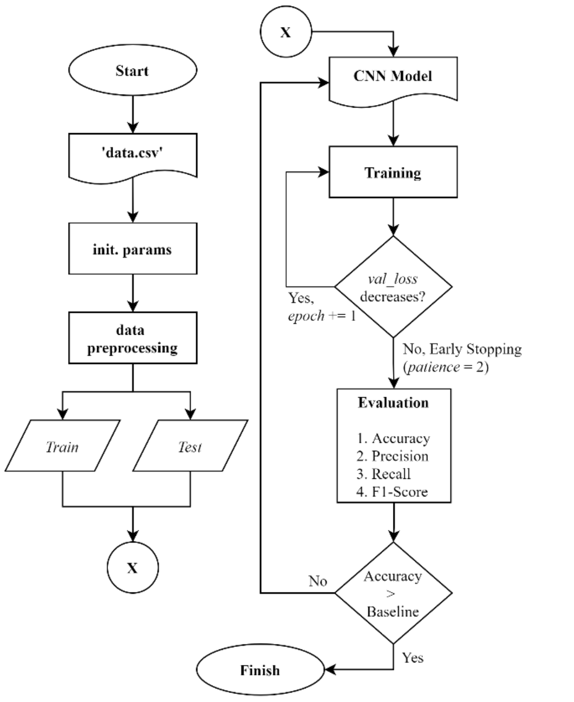


Figure 1. System Flowchart

In this research, the method used is Deep Learning Convolutional Neural Network (CNN). Figure 1 shows the flow of the system used in this research represented in a flowchart. The system received input data in CSV format and will first undergo preprocessing. Afterward, the data will be split into 70% training data and 30% testing data. The percentages of the data splitting are determined through experiments, and these values are considered to be the best for this research. However, having an excessive amount of training data will overfit the model performance. Meanwhile, the smaller the training data will be insufficient for the model to learn through processes and will cause underfitting.

After the data split, the system will continue to build the CNN model architecture and do the training process. Lastly, the CNN model will then predict the classes of every tweet in the test data and evaluate its performance results. The evaluation methods used are accuracy, precision, recall, and F1-Score. The CNN model will be evaluated by comparing it to the baseline whether or not it has surpassed them. If it failed to outperform them, the model needs to be reconstructed by tuning the learning rate, the hyperparameters, or using grid search method to find the optimal layer configurations until it has successfully produced higher results than the baselines.

2.1. Dataset

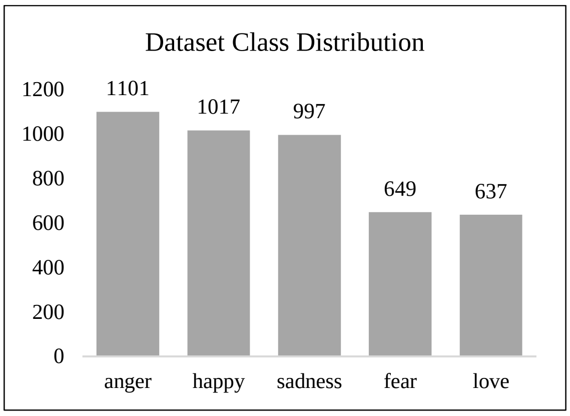


Figure 2. Dataset Class Distribution

The dataset used contains 4.403 rows of tweets that are labeled using five emotion classes, anger, fear, joy, love, and sadness provided by [10] in CSV format. [1] Each row consisted of a tweet and its respective emotion label separated by a comma (,). The first row is a header. For a tweet with a comma inside the text, there is a quote mark (" ") to avoid column separation. The tweets in this dataset have been pre-processed using the following criteria based on Table 1. Mentions have been replaced with “[USERNAME]”, URLs have been replaced with “[URL]”, and Numbers like phone number, invoice number, and a courier tracking number have been replaced with “[SENSITIVE-NO]”. However, these unnecessary strings still need to remove completely from the data, because it can lower the performance produced by the CNN model due to the learning complexity.

Table 1. Preprocessing Criteria

Tweets	Preprocessed
*“Baca buku ini diawal senyum, ditengah-tengah senyum miris, diakhir senyum pasrah. Nuhun bung **@someone**”*	*“Baca buku ini diawal senyum, ditengah-tengah senyum miris, diakhir senyum pasrah. Nuhun bung [USERNAME]”*
*“sebagai supporter speak bola gue ga suka barca, tapi sebagai pecinta sepakbola gue suka ini film, keren **https://example-url.com/**”*	*“sebagai supporter speak bola gue ga suka barca, tapi sebagai pecinta sepakbola gue suka ini film, keren [URL]”*
*“Malam saya mau tanya kenapa benefit saya dibatalkan, saya bukan dropshiper dan saya rasa tidak ada aturan yang saya langgar **1234xxx**”*	*“Malam saya mau tanya kenapa benefit saya dibatalkan, saya bukan dropshiper dan saya rasa tidak ada aturan yang saya langar [SENSITIVE-NO]”*

2.2. Data Preprocessing

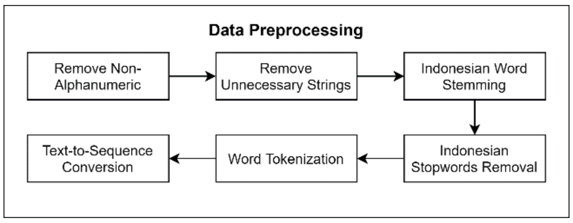


Figure 3. Data Preprocessing

To enhance the data quality, data must be preprocessed before getting into the training process. Data preprocessing in Machine Learning refers to the data mining technique of cleaning and organizing the raw data to make it suitable for building and training machine learning models. It is used to transform raw data into an understandable and readable format. Figure 3 shows the flowchart of the data preprocessing. The method used for word-stemming is from [12]. PySastrawi is a simple Python library to reduce inflected words in Indonesian to their base form. Table 2 shows how PySastrawi applied to Indonesian terms.

Table 2. Indonesian Word Stemming

Word	Stemmed
memainkan (playing)	*main* (play)
bercanda (joking)	*canda* (joke)
terkagum (amazed)	*kagum* (amaze)

In this research, we use a powerful Python tool for developing and evaluating deep learning models called Keras. Keras is a consistent and simple deep learning API, running on top of TensorFlow – a flexible data open-source flow-based programming model – designed to minimize the number of user actions required for

[1] https://github.com/meisaputri21/Indonesian-Twitter-Emotion-Dataset

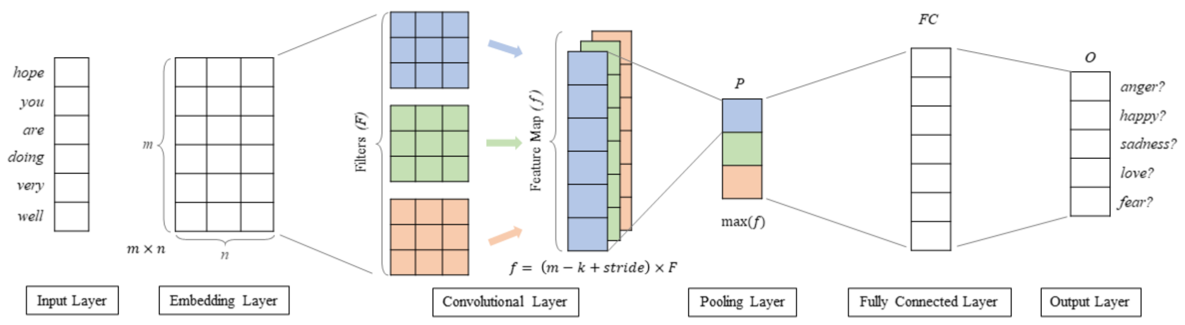


Figure 4. CNN Model Architecture

common use cases, and it provides clear and actionable error messages [13, 14]. In the preprocessing part of the system, Tokenizer from Keras will be applied to get word indexes by tokenizing words inside text documents, also to transform each text into a sequence of integers. This sequence will be used for computation inside the neural network since they were not designed to understand the inputs that are in non-numeric format.

2.3. Convolutional Neural Network (CNN)

This section provides the details of our model. CNN is a deep learning method that was initially used for analyzing and classifying digital images. Its purpose was to extract meaningful features of an image convolutionally by moving the kernel on the convolution layer through two-dimensional matrices [15]. Based on this concept, it can also be applied for text through one-dimension matrices. Figure 4 shows an image representation of the proposed model architecture. It consisted of an input layer consists of sequences of integers, an embedding layer, a one-dimensional convolutional layer, a pooling layer, a fully connected layer, and an output layer consisted of five classes, *anger*, *fear*, *joy*, *love*, and *sadness*. To get better understandings of CNN, we will go through the definition and the usage of every component that constructs CNN models.

2.3.1. Embedding Layer

Keras offers an embedding layer that can be used for text data on neural networks and it requires the data to be integer encoded. The embedding layer works similarly as a simple matrix multiplication that transforms words into their corresponding word embeddings or turns indexes into dense vectors of fixed size. Word embedding can be regarded as textual features so that it can be counted as a preprocessing step in more advanced NLP tasks [16]. This layer can only be used as the first layer in a model. The input data will be padded for each sentence to create a fixed size of sequences. If the sentence is too long, it will be trimmed to the maximum length, and if it is too short, it will be padded by zeros to match m. The embedding layer will receive these sequences and create an embedding matrix within the size of $m \times n$, where m is the output dimension. It consists of the correlation of each word index in every tweet to the whole document. The input dimension m should be equal to the number of unique vocabularies in the data and the output dimension m is the size of the vector space in which words will be embedded. It determines the size of the output vectors from this layer for each word. In our CNN model, the embedding layer will learn along with the model itself rather than using pre-trained word embedding models (e.g., Word2Vec). It is initialized with random weights and will learn an embedding for every word in the previously preprocessed data. An embedding layer learns then tries to find the optimal mapping of each of the unique words to a vector of real numbers.

Table 3. Word Embedding

Sentence	Embedding
"hope to see you soon"	[1,2,3,4,5]
"Nice meeting you"	[6,7,4,0,0] #zeros for padding

Table 3 shows how word indexes are assigned to each unique word by using a one-hot encoding method from Keras Tokenizer API. However, instead of using a large amount of one-hot encoded vectors, an embedding layer prefers to create an embedding matrix based on word indexes to keep the size of each vector much smaller as represented in Table 4.

Table 4. Embedding Index

Word Index	Embedding
0	[1.2, 3.1, 2.5]
1	[0.1, 4.2, 1.5]
2	[1.0, 3.1, 2.2]
3	[0.3, 2.1, 2.0]
4	[2.2, 1.4, 1.2]
5	[0.7, 1.7, 0.5]
6	[4.1, 2.0, 4.5]
7	[3.1, 1.0, 4.0]

Assuming we want to train a CNN model, and Table 4 is our training data, we should first specify our embedding layer. The number of unique vocabularies from the data is 8, so the input dimension m is 8, and 3 for the output dimension n. During the training process of the CNN, embedded vectors are getting updated and expecting that similar meaning words will have similar representations in a multi-dimensional space as shown in Table 4. Once the training has finished, the embedding layer will produce a matrix with the shape of (8, 3) as the output.

2.3.2. Convolution Layer

Convolution is a linear operation that performs the multiplication of inputs with a set of weights. Therefore, a convolution layer was designed to have the ability to apply such an operation to a sequence of input data and a filter or a kernel to produce a dot product. A dot product is the sum of a product of two sequences of numbers that correspond to each other and always resulting in a single value. As the layer convolves through the $m \times n$ matrix with an arbitrary stride, it will calculate the dot products of this matrix and filters (F) to create a feature map f and pass it to the next layer. During the convolution process of calculating the dot products, the feature map will gain values from it as the process continues to the last filter. These values can vary because the convolution layer naturally generates filters as it trains the data, and they do not have fixed values.

2.3.3. Pooling Layer

A pooling layer is a layer that reduces input spatially by using downsampling operations. The commonly used method of a pooling layer is max pooling. This method will take the maximum value for each patch of a feature map resulted from the convolution layer. The result of using a pooling layer is a summarized version of the feature map. In Keras, there is another type of pooling that is sometimes used called global pooling. In this research, the pooling method used is global max pooling. Instead of downsampling patches from the feature map, a global pooling layer downsamples the entire feature map to a single value. This would be the same as setting the pooling size to the size of the input feature map.

2.3.5. Fully Connected Layer

A fully connected layer is simply feed-forward neural networks which is an artificial neural network connection where the connection between nodes does not form a cycle [17]. The input to the fully connected layer is the output from the pooling layer, which has been flattened. After passing through the fully connected layers, the final layer will then get probabilities of the input being in an appropriate class.

2.3.6. Dropout

Small datasets have higher chances of causing overfits when using CNN models. This issue may also affect the lower performance of the model when the training process has done. An approach is needed to reduce the overfitting of every network and to average the predictions of the model. A dropout comes to regularize a fixed-size model by averaging predictions of the hyperparameters used in a certain layer by weighting each setting of its posterior probability given the training data [18].

Dropout can be applied with most layers in a neural network, whether it is a hidden or visible layer. During the training process, some layers are dropped temporarily across the connections. Dropout can noise the training process by forcing layers to drop their nodes. The dropout values are between 0 and 1. Assuming it is specified to 0.5, it will force layers to drop 50% of their nodes to reduce overfitting.

2.3.7. Regularization

Regularization is a technique of modifying the model such that the model generalizes better. This will penalize the weight matrices of the nodes in the network. L1 and L2 are the most common types of regularization. It updates the general cost function by adding another term known as the regularization term. As a result, the weight matrix value decreases because it assumes that an artificial neural network with a smaller weight matrix produces a simpler model. Hence, it will also reduce overfitting to a great extent. In this research, the regularizer used is the L2 type of regularization for the experiment. L2 norm is represented by the term below [15]:

$$\Omega(W) = Loss\ Function + \frac{\lambda}{2} \times \sum \left|\left|W\right|\right|^2 \qquad (1)$$

Where the loss function used here is categorical cross-entropy, considering the model has five output classes, and the regularization parameter (λ) used is the hyperparameter whose values are optimized for better results and usually in logarithmic scale $(10^n,\ n < 1)$. L2 regularization is also known as weight decay as it forces the weights (W) to decay towards zero, but not exactly zero. Intuitively, the smaller weight reduces the impact of hidden neurons. In this case, the hidden neuron becomes negligible and the overall complexity of the neural network is reduced. Less complex models usually avoid modeling noise in the data, and therefore, there is no overfitting.

2.3.8. Learning Rates

Learning rate is a hyperparameter that controls how much the model should update the weights for every epoch and often has a value in the range between 0.0 and 1.0. The learning rate may be the most important hyperparameter when configuring neural networks [19]. It is required to analyze the appropriate value of learning rates for better performance. The lower value of the learning rate resulting the training process being longer and have more epochs, while the higher value will faster the learning process. The model will adapt more quickly to the problem to get less training time. If the learning rate is set too low, training will progress very slowly because the model will make very few updates on the weight on your network. However, a learning rate that is too high causes the model to obtain the solutions too quickly, so that it may skip the important features meant to be acquired.

2.3.9. Adam Optimizer

Adam Optimizer is an adaptive stochastic learning rate optimization algorithm designed for deep neural networks by computing individual learning rates for different parameters. It combined the Adaptive Gradient Algorithm and Root Mean Square Propagation. Both methods are using the same concept of stochastic optimization. Stochastic optimization is the process of optimizing an objective function in the presence of randomness. Adam optimizer handles the sparse gradients on noisy datasets efficiently by using a smaller amount of memory and it can be an advantage for this research, considering we use a small amount of data but with a more complex CNN model.

2.4. Model Testing

Ensuring the model has good performance is needed to get a better understanding of the problems. Therefore, it needs to be tested by doing predictions. The previously trained model will predict the labels from the test data. In this research, the methods used to evaluate the predictions are accuracy, precision, recall, and F-Measure. These methods are measured based on a confusion matrix represented in Table 5.

Table 5. Confusion Matrix

	Actual Class (+)	Actual Class (-)
Predicted Class (+)	TP	FP
Predicted Class (-)	FN	TN

Accuracy is a method that is often used in research related to binary classification and/or multiclass classification [20]. Accuracy is the number of correctly predicted data points out of all the data points. It can be calculated by dividing the number of correct predictions by the number of total predictions as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

On the formula above, *TP* is represented as True Positives, where the model correctly predicts the positive classes. *TN* means True Negatives, which means the model has correctly predicted negative classes. Meanwhile, *FP* and *FN* stand for False Positives and False Negatives. These denote that the model has incorrectly predicted both positive and negative classes. Precision is used to measure the positive class predicted upon the probability it is correct. When the model predicts positive, precision will calculate how often it predicted positive class correctly. Recall is the probability of positive labels labeled as positive. These methods can be calculated by using the following formulas:

$$Precision\ (P) = \frac{TP}{TP + FP} \tag{3}$$

$$Recall\ (R) = \frac{TP}{TP + FN} \tag{4}$$

The last evaluation method is F-Measure. F-Measure provides a single score that balances both the concerns of precision and recall in one number. It is the ratio of the number of positive classes that are predicted to be correct with the number of all samples that should be predicted as positive. This method is commonly used for evaluating many kinds of machine learning models. The F-Measure can be calculated by using the following formula.

$$F = \frac{2PR}{P + R} \tag{5}$$

From the formula above, the F-Measure was produced by having a trade-off between Precision and Recall. If the model is optimized to increase one and disfavor the other, the harmonic mean quickly decreases. However, when both precision and recall are equal or have slightly different values, the F-Measure will surely have a great result. With high precision but low recall, models are extremely accurate, but it misses a significant number of instances that are difficult to classify. This may not be very useful for the evaluation process.

3. Result and Discussion

3.1. Training Process

Based on Figure 4, the input layer receives a sequence of integers from the previously preprocessed data. The data consists of 14,648 unique vocabularies and a maximum of 83 words for a whole text document. The embedding layer takes every unique vocabulary as the input dimension, 128 output dimensions, and 83 maximum input lengths. Since the embedding layer is the first layer of the model, it needs to have more data from the input layer for the embedding matrix in terms of better learning processes. As a result, the model will have more time to learn, but the performance results will be significantly higher. Therefore, the more input dimension to have, the higher the performance of the CNN model.

Output from the embedding layer goes to the convolution layer. The convolution layer consists of 128 filters as well as 5 kernel sizes. These numbers were determined by tuning them repeatedly over time and have produced the best performance results. The activation function used here is Rectified Linear Unit (ReLU) that will produce the output directly if it is positive, otherwise, it produces zeros. The usage of

ReLU helps to prevent the exponential growth in the computation requirements to the CNN operations. As the CNN scales in size, the computational cost of adding extra ReLUs increases linearly.

The output of the convolution layer will then undergo pooling. The commonly used pooling method is max pooling. In this CNN model, the global max-pooling layer downsamples the input representation by taking the global maximum value over the time dimension. Before getting into the fully connected layer, the dropout is added to the network. In this layer, the data will then be randomly dropped out 50% of the neurons during the training process to reduce overfitting and improve generalization errors.

The fully connected layer predicts whether the input data belongs to the appropriate class of the five emotion classes and it used the Softmax activation function. In the training process, the CNN model will go through epochs and set 0.0003 for the learning rate of Adam Optimizer. The loss function used is categorical cross-entropy, considering our model classifies five emotion classes. As the model fit, the early stopping with the patience value of 2 will monitor the validation losses. Whenever the validation loss keeps improving, it will gain more epochs. Otherwise, it will stop the training right away.

3.2. Text Emotion Classification

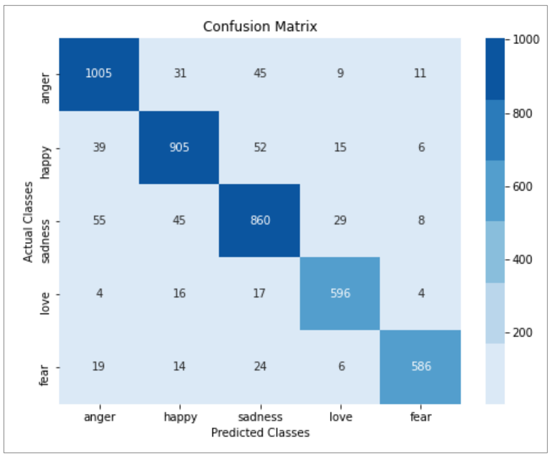


Figure 5. Multiclass Confusion Matrix

The predictions are done by classifying tweets into five classes. The confusion matrix in Figure 5 shows the correctly predicted tweets among their classes. 22.8% of the tweets are correctly classified as anger, 20.5% as happy, 19.5% as sadness, 13.5% as love, and 13.3% as fear. The remaining 10.4% of tweets are misclassified.

Table 6 shows the evaluation results of each predicted class. These results are base on the confusion matrix in Figure 5. The evaluation methods used are accuracy, precision, recall, and F1-score. In Table 6, the highest precision score is *love* class, by 93.6%. Meanwhile, the *fear* class has the highest score of recall by 95.3%. The *sadness* class has the lowest results of all. It indicates the system had more difficulties in classifying tweets to this class. It may be caused of the trends of tweets labeled as *sadness* are too complicated for the model to learn. Nevertheless, the results are not significant in terms of score interval compared to the other classes, so the model would still get high overall performance results.

Table 6. Classification Report

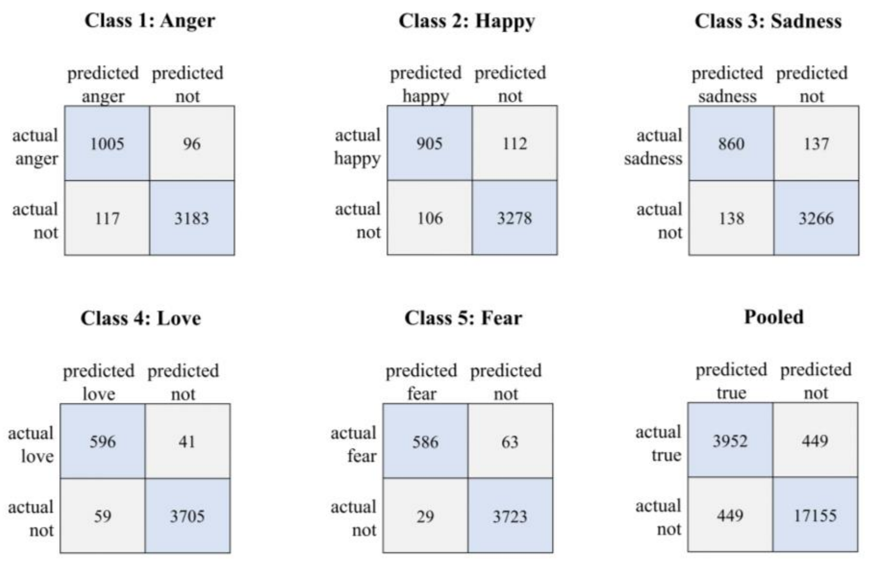
classes	precision	recall	f1-score
anger	0.913	0.895	0.904
happy	0.890	0.895	0.893
sadness	0.863	0.862	0.893
love	0.936	0.910	0.923
fear	0.903	0.953	0.927
micro-avg	0.898	0.898	0.898
macro-avg	0.901	0.903	0.902

Table 7. Comparison of Previous Studies

classes	precision	recall	f1-score
Logistic Regression [10]	0.720	0.682	0.697
LSTM-GloVe [11]	0.330	0.380	0.350
Our CNN Model	0.901	0.903	0.902

According to the definitions, precision calculates the number of positive predictions made correctly, and recall measures the number of correct positive predictions made from all potential positive predictions that could be made. The results show that the CNN model is more likely tends to identify a tweet as *love* of all tweets that have *love* class. Meanwhile, recall tells us that the model has the highest probability to correctly predict a tweet as *fear* over the other classes upon all tweets that had classified. These cases occur because of the dependency on keywords inside a tweet that help the CNN model to predict tweets to their appropriate classes. The ambiguity of keywords may affect the model to predict because the occurrences of them are rarely to be found in the whole document. People are often using autocorrection features on their devices before posting their tweets. The clearer a tweet is in terms of contexts, the easier the model recognizes it. However, assuming that the tweet consisted of unknown abbreviations or words, they may also affect the calculation. If they exist, then the model will try to learn from the other correlated and the adjacent words to calculate the probability instead of focusing on them. Table 8 shows how the representation of the keywords of a tweet affecting the predictions.

Based on Table 8, we may see that the first tweet has two keywords, which are *aman* (safe) and *nyaman* (cozy). Meanwhile, the second tweet has longer words to describe what emotion would fit the tweet. The term "safe" and "cozy" are straightforward indicating *love* by their means.



Class 1: Anger

	predicted anger	predicted not
actual anger	1005	96
actual not	117	3183

Class 2: Happy

	predicted happy	predicted not
actual happy	905	112
actual not	106	3278

Class 3: Sadness

	predicted sadness	predicted not
actual sadness	860	137
actual not	138	3266

Class 4: Love

	predicted love	predicted not
actual love	596	41
actual not	59	3705

Class 5: Fear

	predicted fear	predicted not
actual fear	586	63
actual not	29	3723

Pooled

	predicted true	predicted not
actual true	3952	449
actual not	449	17155

Figure 6. Pooled Confusion Matrix

Table 8. Keywords Representation

Tweets	Keywords	Class
"Setiap kesempatan yg pernah hadir tuk dapat membuatmu selalu merasa ***aman*** *dan* ***nyaman****, kini jadi suatu kehormatan yg pernah didapat."* Translated: *"Every opportunity that has ever been present to always feel safe and comfortable, has now become an honor that has been obtained before."*	*aman* (safe), *nyaman* (cozy)	love
"Pulang udah H-4 lebaran dilema sekali. Seperti ***tidak bisa melakukan apa2 di rumah*** *sebelum lebaran. Buka puasa bareng cuman 3 hari sama keluarga begitu juga sahur"* Translated: *"Going home on D-4 of Eid is such a dilemma. It is like I cannot do anything at home before D-day. To break my fast with family within only 3 days as well as having a pre-dawn meal"*	*tidak bisa melakukan apa2 di rumah* (cannot do anything at home)	sadness

Hence, the precision score of *love* class is higher than other classes. Meanwhile, on the second tweet, there are no individual keywords that can be observed. Preferably, the model uses word phrases to recognize the appropriate class. Referring to these cases, we as human beings can be easily predicting both tweets to their classes. However, the system would need to learn from the trends of the whole document before predicting. The same technique may also apply to *fear* class

Figure 6 shows how the confusion matrix from Figure 5 is pooled by splitting it into the matrix of each class. The micro-average method can be calculated by using the summed version of the multiclass confusion matrix in Figure 5, called pooled confusion matrix (Figure 6) [22]. Meanwhile, the macro-average method is calculated by averaging precision, recall, and F1-score on every class. The micro-averaged precision, recall, and F1-score were obtained by the same value of 89.8%. Meanwhile, the macro-averaged precision, recall, and F1-score were obtained respectively 90.1%, 90.3%, and 90.2%. Since the pooled confusion matrix has been created, the accuracy can be calculated by using the formula (2), resulting in 89.8% accuracy.

In this research, the CNN model will learn the trends through the embedding layer. Since tweets are transformed into a sequence of integers, each vocabulary in every tweet will be paired based on their adjacent and correlated word indexes and calculate the probability of the occurrences of the whole document. It produces the embedding matrix, which will then be used for the model to classify tweets based on their language structure in Indonesian.

3.3. Discussion

A study by [10] using Logistic Regression tends to be more suitable for classifying a small amount of data. Some experiments on feature extractions were applied to the model. However, their results from the proposed classifier model are not quite enough to fully meet the research standard. In the other study by [11], they had obtained very low-performance results by using the LSTM-GloVe method. They have also attempted experiments by applying multiple hyperparameter combinations, including lowering the learning rates, adding dropouts, and changing the optimizers, but still could not optimize the LSTM model performance results. It may occur because of the incompleteness of data preprocessing on the raw data since it only undergoes preprocessing inside the layers in the LSTM model that causes the lack of data quality. For instance, they had missed word-stemming, which is a necessary part of data preprocessing. It converts every term into its base form. In this research, word-stemming played an immense role in the data preprocessing process that helps the model to more easily recognize each term in base form.

Based on the analysis, both researches had experienced difficulties in optimizing their models. In the first research, the Logistic Regression method worked more effectively than the succeeding. However, despite having done some experiments on the model, the results are still low. This issue may indicate that the base model classifier is the key to getting high-performance results. In the research by [11], the LSTM method encountered very low performance. Since LSTMs are deep learning-based models, the quantity of the data matters. The more data used, the better the performance. Although, the data used only consisted of 4.403 rows of data which is insufficient for deep learning models to learn. In another way, the model complexity can improve the learning process despite having various amounts of data. For instance, adding more nodes to the layers will help the model extracting more information from the data. This information is used as blueprints when the model starts to predict.

This research compares the methods used in the previous studies to our CNN method using the same dataset provided by [10], as represented in Table 7. Due to the limitations of the data used, it is necessary to add more nodes within layers to gather more information from the data through the networks. However, our CNN model implements different feature extraction methods. The methods used are within the CNN layers, e.g., word embeddings in the embedding layer. The embedding layer forms the data into the embedding matrix used for the model to digest its contexts. In the training process, the embedding matrix is used for calculating the adjacency and the correlation of each term in every tweet within a whole document. Once the training is done, the model predicts the data and evaluates the results. The model produced great final results and showed significant differences in terms of model performance.

4. Conclusion

In this research, the CNN method is implemented to identify emotion by classifying tweets in Indonesian by setting the parameter combinations, such as adding 50% dropout, applying L2 regularization, and lowering the Adam Optimizer’s learning rate to 0.0003. The performance results are pretty high, considering this research used a small dataset. The precision, recall, and F1-score were obtained respectively 90.1%, 90.3%, and 90.2%, while the accuracy is 89.8%. It is worth mentioning that the parameter combinations played a huge role in optimizing the model.

CNN works more effectively on a comprehensive user-generated suite of tasks and datasets (e.g., user reviews) in a much faster computation and has shown better performance results compared to the LSTMs [23, 24]. Therefore, CNN is more suitable and applicable for datasets consist of tweets considering they are user-generated texts. In this research, the CNN method has shown significant differences in performance results that outperformed LSTM. Hence, it is concluded that CNN has proven to be a better method to use for cases of text classification.

Reference

[1] Statcounter, "Statcounter GlobalStats," Statcounter, March 2021. [Online]. Available: https://gs.statcounter.com/social-media-stats/all/indonesia. [Accessed 22 April 2021].

[2] M. Zaskya, A. Boham and L. J. H. Lotulung, "Twitter Sebagai Media Mengungkapkan Diri Pada Kalangan Milenial," ACTA DIURNA KOMUNIKASI, vol. 3, 2021.

[3] A. Mehrabian, Nonverbal communication, Routledge, 2017.

[4] N. Y. Hutama, K. M. Lhaksmana and I. Kurniawan, "Text Analysis of Applicants for Personality Classification Using Multinomial Naïve Bayes and Decision Tree," JURNAL INFOTEL, vol. 12, p. 72–81, 2020.

[5] J. K. Rout, K.-K. R. Choo, A. K. Dash, S. Bakshi, S. K. Jena and K. L. Williams, "A model for sentiment and emotion analysis of unstructured social media text," Electronic Commerce Research, vol. 18, p. 181–199, 2018.

[6] J. W. Simanullang, A. Adiwijaya and S. Al Faraby, "Klasifikasi Sentimen Pada Movie Review Dengan Metode Multinomial Naïve Bayes," eProceedings of Engineering, vol. 4, 2017.

[7] M. Yulietha, S. A. Faraby, W. C. Widyaningtyas and others, "An implementation of support vector machine on sentiment classification of movie reviews," in Journal of Physics: Conference Series, 2018.

[8] Y. Chen and Z. Zhang, "Research on text sentiment analysis based on CNNs and SVM," in 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2018.

[9] H. Kim and Y.-S. Jeong, "Sentiment classification using convolutional neural networks," Applied Sciences, vol. 9, p. 2347, 2019.

[10] M. S. Saputri, R. Mahendra and M. Adriani, "Emotion classification on indonesian twitter dataset," in 2018 International Conference on Asian Language Processing (IALP), 2018.

[11] A. Larasati, B. Harijanto and F. Rahutomo, "Implementasi Deep Learning Untuk Deteksi Ekspresi Emosi Pada Twitter," in Seminar Informatika Aplikatif Polinema, 2020.

[12] H. A. Robbani, "PySastrawi," 28 September 2018. [Online]. Available: https://github.com/har07/PySastrawi.

[13] F. Chollet and others, Keras, 2015.

[14] M. Abadi and others, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015.

[15] Goodfellow, Y. Bengio, A. Courville and Y. Bengio, Deep learning, vol. 1, MIT press Cambridge, 2016.

[16] Y. Li and T. Yang, "Word embedding for understanding natural language: a survey," in Guide to big data applications, Springer, 2018, p. 83–104.

[17] A. Zell, N. Mache, R. Hübner, G. Mamier, M. Vogt, M. Schmalzl and K.-U. Herrmann, "SNNS (stuttgart neural network simulator)," in Neural network simulation environments, Springer, 1994, p. 165–186.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," The journal of machine learning research, vol. 15, p. 1929–1958, 2014.
[19] Y. Bengio, R. Ducharme, P. Vincent and C. Janvin, "A neural probabilistic language model," The journal of machine learning research, vol. 3, p. 1137–1155, 2003.
[20] M. Hossin, M. N. Sulaiman, A. Mustapha, N. Mustapha and R. W. Rahmat, "A hybrid evaluation metric for optimizing classifier," in 2011 3rd Conference on Data Mining and Optimization (DMO), 2011.
[21] Y. Sasaki, "The truth of the F-measure," Teach Tutor Mater, 1 2007.
[22] D. Jurafsky and J. Martin, "Naive bayes and sentiment classification," Speech and language processing, p. 74–91, 2017.
[23] X. Zhang, J. Zhao and Y. LeCun, "Character-level convolutional networks for text classification," arXiv preprint arXiv:1509.01626, 2015.
[24] S. Bai, J. Z. Kolter and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," arXiv preprint arXiv:1803.01271, 2018.