Implementation of CNN Algorithm for Indonesian Hoax News Detection on Online News Portals

Clifansi Remi Siwi Hati [1*], Heni Sulistiani [2]**
* Informatika, Universitas Teknokrat Indonesia
**Magister Ilmu Komputer, Universitas Teknokrat Indonesia
clifansi_remi_siwi_hati@teknokrat.ac.id [1], henisulistiani@teknokrat.ac.id [2]

Article Info

Article history:

Received 2025-04-15
Revised 2025-05-03
Accepted 2025-05-06

Keyword:

Political Hoax News,
Convolutional Neural Network (CNN),
FastText.

ABSTRACT

The spread of hoax news in the Industrial Revolution 4.0 era has occurred in the world's society, including Indonesia. Therefore, an effective method is needed to detect it. The purpose of this research is to apply deep learning with the Convolutional Neural Network (CNN) algorithm in detecting text-based hoax news in Indonesian. The dataset is taken from Kaggle, which has been scraped from CNN Indonesia, Tempo, and Turnbackhoax, which will be labeled as valid and hoax. The implementation of the dataset goes through several processes that include input dataset, data pre-processing using pre-trained embedding FastText, data processing, model evaluation. Data is divided into 80% training data and 20% test data for CNN model development. The results show that the CNN model can achieve high accuracy in detecting hoaxes with training accuracy values reaching 95.77% and validation accuracy reaching 95.18% with a loss of 0.1146 and 0.1210, which means that the model is adequate in classifying text-based hoax news. The model is evaluated using 5 Stratified Fold Cross Validation, ROC AUC, confusion matrix with the visualization. For further research, it is recommended to explore other architectures and increase additional variations for training data so the model can understand patterns well.

I. INTRODUCTION

Information media in the era of the Industrial Revolution 4.0 increased as technology developed. Easy access to various information causes hoaxes, both content and news, to circulate easily [1]. Hoax news is defined as news whose truth has not been proven or can be called fake news based on KBBI [2]. The development of this hoax news has caused unrest and serious problems in various countries around the world, including Indonesia [3]. In Indonesia, the low level of media literacy is one of many factors that exacerbates the problem, because the community did not scrutinize the news that was found before it was disseminated [4].

Every citizen has the freedom to search and upload positive or negative content. They can also freely misuse information or spread hoaxes [5]. The Ministry of Communication and Digital successfully identified 1,923 fake information, fake news, and hoax content in Indonesia. In all the findings, hoax content was found including the political category with as much as 237 content, the government category with 214 content, the health category with 163 content, the disaster category with 145 content, and others with 84 content. Meanwhile, for false information in the international category and defamation as many as 50 contents, the trading category has 35 contents, the crime category with 33 contents, the religious and educational categories are 8 contents and the myth category is 6 contents [6].

The distribution of hoax news in Indonesia on online news portals can be explained by various related theories. Originally, propaganda theory was regarded as the hoax news that became this type of propaganda, which was designed to influence public opinion. That hoax news is often used to spread manuscripts that uphold certain agendas, whether they are political, community policy, or economic [7]. Addressing this problem requires an effective method of identifying hoaxes or validating a news item such as text classification with deep learning.

The Convolutional Neural Network (CNN) method is one of the effective approaches because CNN can process Indonesian text data well, identifying important patterns and features. Built-in models consist of embedding layers, and convolution layers, followed by global max pooling, and output layers (dense, sigmoid) [8]. In previous research conducted by [4], CNN was able to classify text data in the form of hoax news with results of achieving 92.53% accuracy in training data and 81.09% in testing data, with loss of 0.33 and 0.55. This suggests that previous research has shown good potential. However, that research took and studied only a small amount of data and the CNN model could memorize it well.

Therefore, we are interested in researching related CNN algorithms that are capable of maximizing text classification. This study contributed by combining word weighting through the pre-trained embedding of FastText, Synthetic Minority Oversampling Technique (SMOTE) techniques to prevent data imbalance, 5 Stratified Fold Cross Validation and ROC curve with AUC value for the model CNN evaluation. In hopes of improving the model's accuracy in detecting other hoax news and accurately reducing the negative effects of information.

II. METHOD

Based on the literature studies, a diagram of research flow was made to solve the problem so that this research is only focused on solving the problem [9]. The following is the research flow's diagram that can be seen in Figure 1.

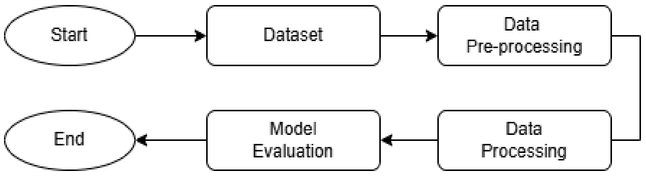


Figure 1. Research overview in diagram flow

Showing an explanation of the processes in Figure 1 so that it can clarify the flowchart.

A. Dataset

The raw data of valid and hoax news used, comes from the world's largest online data science community platform named Kaggle, on account with username linkgish that discusses Indonesian facts and hoax political news (https://www.kaggle.com/datasets/linkgish/indonesian-fact-and-hoax-political-news). Each valid news data is taken from Indonesia's online news portal, namely CNN Indonesia with 10.000 data, and Tempo with 6.952 data. Meanwhile, for hoax news data, 10.384 data were taken from Turnbackhoax web portal. Data used has a period from July 2015 to March 2023.

The stored datasets were still separate in xslx format, so first, we decided to label it with `df_turnbackhoax["label"] = 1`, `df_cnn["label"] = 0`, also `df_tempo["label"] = 0` and merge them with `df_combined = pd.concat([df_turnbackhoax, df_cnn, df_tempo], ignore_index=True)`. Label 0 indicates that the news in both CNN and Tempo datasets are considered as valid or non-hoax, then label 1 indicates that all news in the Turnbackhoax dataset is categorized as hoaxes. Those three data are combined to have 26.976 data.

B. Data Pre-Processing

Data pre-processing is the first step where the data is transformed in such a way that the machine can parse the data [10]. The collected data, first omit unnecessary words in the classification process [11]. It includes stopwords removal and tokenizing with NLTK, word weighting, also oversampling using SMOTE [12]. This step is important to ensure only the main information is used in the next process [13]. Followed by an explanation of the data pre-processing streaming sequence conducted in this study,

1) Stopword Removal

Statistical analysis of the natural text contained in the document was performed to find non-essential words (stopwords) in the text. The document contains valid news or hoaxes with different writing methods for each news item. This collection of data is considerable and includes in-depth political news [14]. The unimportant words (stopwords) that were removed made the writing denser and felt like it had a small role. The purpose of this removal is to find words that appear frequently in the document. These words have no value for further processing [15].

NLTK, or Natural Language Toolkit, is one of the most popular Python libraries for Natural Language processing. It is a Python tool that enables the analysis of Natural Language text in an efficient and simple way. In addition, there is also a collection of text processing libraries for tasks such as classification, tokenization, stemming, et cetera. By collaborating with the standard Python library and other third-party libraries, NLTK can perform additional processing of already processed results (Meng Wang).

In this process, we used NLTK stopwords in Indonesian, made `stop_words` list. Then, used `text.lower()` to make the entire text equally small. Followed by `text = re.sub(r'[^\w\s]', '', text)` to delete all the conjunction that are inserted in text needed for next process. Table 1 shows how the comparison before and after the raw data received stopword removal.

2) Tokenization

Tokenization is a process in which sentences are divided into words, letters, and punctuation marks known as tokens. This separation is generally done based on the presence of spaces or punctuation marks. This process is important to eliminate unnecessary words in subsequent processing steps [16].

The NLTK library is used to perform word tokenization by means of `word_tokenize`, which separates the text into a list of separate words. This tokenization process allows each word to be analyzed individually, rather than as one long string. Then filtering process is carried out by removing stopwords, which are words that appear frequently but have no important meaning in text analysis such as “and”, “in”, and “which”.

This filtering process involves examining each token and retaining only words that are not in the `stop_words` list. The remaining words are then merged into one clean and relevant string, and stored in a new column called `clean_text` in the `df` (data frame). The result of this preprocessing stage makes the text data more ready for the next step of analysis.

3) Word Embedding

Word embedding is a step that converts text into numbers so that it can be understood by machine learning technologies and deep learning algorithms, such as Co-Occurrence Vector, Bag of Words, TF-IDF Vector, and LDA. With the application of word algorithms, continuous vector representations such as Glove, Word2Vec, and FastText in deep learning methods can convert terms into meaningful vectors [17]. FastText is a library for word representation developed by a research team from Facebook. It has 2 million crawled common words with a dimension of 300, and offers 600 billion word vectors. In addition, it also utilizes manually generated n-grams as a feature in addition to individual words [18].

We utilize `fasttext.util.download_model` with the `'id'` parameter to download the Indonesian model, and the `if_exists='ignore'` option keeps the model from being downloaded again if it already exists. After the model download process is complete, next one will loads the model from the `cc.id.300.bin` file using `fasttext.load_model`, so that it can be used to generate the embedding (representation vector) of the words in the processed text.

After that, we load the `get_fasttext_vector` function which takes the text as input. This function will break the text into words using the `split()` method. If the input text is empty (no tokens), then the function will provide a null vector of size 300 as a replacement, following the size of the FastText embedding dimension used. If tokens are present, the function will retrieve a representation vector for each word in the text using `ft.get_word_vector(word)`, then average all the word vectors with `np.mean` to produce a single vector representing the entire text.

The `get_fasttext_vector` function is used for each row in `clean_text` column of `df` via apply. Each result is a vector with a dimension of 300. Since each row produces one vector, all the vectors are then organized into one large two-dimensional array with `np.vstack`, thus forming a feature matrix `X_ft` that has a size of (`number_data`, 300). This matrix will serve as the numerical input for the machine learning model, where each news story is already represented in numeric form based on the meaning of its words.

4) Oversampling

Oversampling can be done through two main methods. The first method involves duplication of samples from the original minority group, such as by random sampling of excess samples. One of the most widely used techniques for over-sampling is the Synthetic Minority Oversampling Technique, otherwise known as SMOTE. The SMOTE method creates synthetic data by using linear interpolation between points of a minority class and one of the K-Nearest Neighbors. SMOTE is a highly effective over-sampling technique and has been widely applied in a variety of applications [19].

To solve the problem of class imbalance in the dataset, we use the SMOTE method. First, we created a SMOTE object with `random_state=42` to keep the results consistent. Next, we run the `fit_resample` function on the `X_ft` feature data and `y` label. This process automatically creates additional samples for fewer classes by creating new examples through interpolation of the existing minority samples. As a result, we get new `X_resampled` (feature) and `y_resampled` (label) data with a more balanced class distribution. This prevents the machine learning model to be trained from being biased towards more classes.

C. Data Processing

This stage receives clean data from the pre-processing stage, then enters the CNN model to be trained every word [20]. We chose to use the Convolutional Neural Network (CNN) algorithm over other deep learning algorithms—such as LSTM, Bert, et cetera, because it is able to extract local patterns from text, such as phrases or word combinations that often appear in hoax news, without relying too much on long word sequences.

CNN itself has a convolution layer formed from a number of combined convolutional layers, a pooling layer, and also a fully connected layer [21]. With its convolutional capabilities, CNN can recognize important features of news text efficiently, making it faster in training and computationally lighter. It can be seen in Figure 3, the standard architecture of CNN algorithms in text classification.

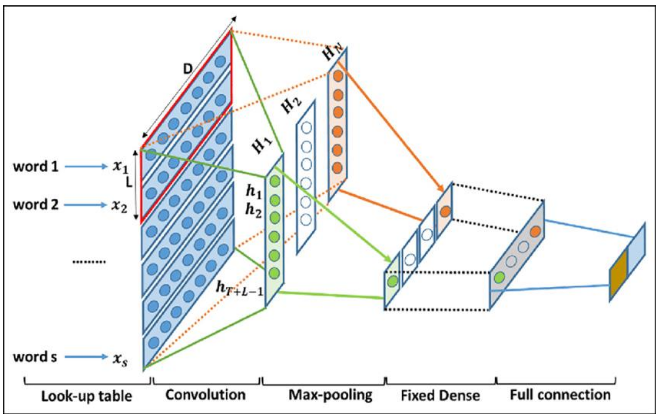


Figure 2. Illustration of standard CNN architecture on text classification (source: [22])

In addition, CNNs are more resistant to overfitting on medium-scale data and are easier to implement as a robust baseline for text classification tasks such as hoax news detection.

Before creating the CNN model, we split the dataset into two parts, training and validation data using the `train_test_split` function. This aims to avoid data leakage when the model is built. The dataset containing X features and y labels is split into 80% for training data (`X_train, y_train`) and 20% for validation data (`X_test, y_test`), according to the parameter `test_size=0`. We use the parameter `random_state=42` to keep the data split consistent and reproducible. This division is important to assess the performance of the model on data that has never been encountered before, so as to measure how well the model can be generalized objectively. After division, this data becomes a train set.

The used of CNN architecture in this research are consisting of input layer, Batch Normalization, max pooling, and fully connected.

1) Input Layer

The input layer in this CNN model is not shown directly, but is specified indirectly through `input_shape=(X_train. shape[1], 1)` in the first `Conv1D` layer. This indicates that the model receives input in the form of one-dimensional sequences that have a certain length (e.g. TF-IDF feature count or FastText embedding) and have 1 channel. This layer serves as an entry point for the data into the network, where each element in the sequence is treated as a feature point that will be locally processed by a convolution filter to find patterns or important features from the sequential data.

2) Batch Normalization Layer

This layer is used to normalize the output of previous layers to have a more stable distribution during training, which helps speed up convergence. Each x_i activation is normalized using the mean and variance derived from the mini-batch. This normalization process reduces the mean μ_B of all activations and divides them by the square root of the variance σ^2. It ensures that normalized activations have an average of zero and a variance of one. In addition, a small value ϵ is added to the denominator to maintain numerical stability, especially to avoid division by zero as has been shown in Equation 1.

$$\hat{x}_i = \left(\frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}} \right) \tag{1}$$

The normalized activation of x_i is then made larger using γ, which is a learnable parameter, and also shifted using β, another learnable parameter. In the presence of these parameters, models can learn about the best scaling and shifting of normalized activations, which provides more flexibility for the network [23]. It is applied mathematically in Equation 2.

$$y_i = \gamma \hat{x}_i + \beta \tag{2}$$

`BatchNormalization()` is applied after each `Conv1D` layer to balance the convolution results before going to pooling or dropout. The goal is to stabilize the distribution of activation values throughout the training process, so that the model can learn more quickly and stably. By balancing the output for each mini-batch, this layer supports the reduction of problems such as vanishing gradient and accelerates convergence. The application of batch normalization in each convolution block ensures that the data flow remains regular even though the model consists of many layers.

3) Max Pooling Layer

In deep learning, these layers are used to reduce the output dimensions of previous layers by selecting the highest value of any convolutional filter along the text sequence. `MaxPooling1D` is applied after `Conv1D` in the first two convolution blocks to shorten the sequence length by keeping the highest value of each pooling window. In this way, the data size is reduced and only the most striking features are noticed. On the other hand, `GlobalMaxPooling1D` at the end of the third convolution block is in charge of summarizing the entire sequence into one maximum value for each filter. This means that only the strongest value of the complete output is kept for each filter. This makes the model more efficient and encourages the network to focus on the most important features of the entire input.

4) Fully Connected Layer

This fully connected or else dense layer, in deep learning models plays an important role, in integrating features that have been taken from previous layers to make final decisions based on that information. Once the features have been extracted using convolution and pooling, the result will enter the fully connected layer (or dense layer) stage for classification. In your model, this starts with `Dense(64, activation='relu')`, which is a layer with 64 neurons and a ReLU activation function. The job of this layer is to integrate all the features from `GlobalMaxPooling1D` into a more compact and meaningful final representation. The neurons in this layer learn to recognize complex patterns based on a combination of previously extracted features.

After the dense layer, there is `Dropout(0.5)` which serves to avoid overfitting by randomly deactivating half of the neurons during the training process. Finally, the model adds a `Dense(1, activation='sigmoid')` layer, which acts as the output layer for binary classification. This layer has only one neuron because the model is tasked with predicting two categories (i.e. hoax news or non-hoax news), and the sigmoid activation function produces a probability value between 0 and 1. If the output is close to 1, then the input is most likely classified as a positive category; if it is close to 0, then as a negative category. In addition to utilizing Adam's optimization algorithm, the CNN was trained with batch size=64 and epoch=10.

D. Model Evaluation

After data processing, the model will be evaluated whether the model's performance can classify the news as a hoax or valid with evaluation, precision, recall, and F1-score metrics [23]. The results of the evaluation are represented by a confusion matrix [24]. This evaluation were also used

Receiver Operating Characteristic Area Under the Curve (ROC AUC) and 5 Stratified Fold Cross Validation.

Redistribute X (feature) and Y (label) data, with 80% of the data for training and 20% for testing, and insert random seeds that are then used as validation sets. Furthermore, previously trained models are used to forecast test data (X_test) using `model.predict`, producing probabilities as outputs (because the output activation function uses sigmoid).

This output is then flattened with `.ravel()` to be a one-dimensional array, and converted to a class label (0 or 1) by comparing each probability value against a threshold of 0.5: if more than 0.5, it is considered class 1; otherwise, it is considered class 0. The final result `y_pred_labels` contains the classification predicted by the model for the test data, and are ready to be evaluated with metrics such as accuracy or F1-score. Next, we calculate and demonstrate evaluation metrics to assess the performance of the classification model by comparing the actual labels (`y_test`) and predictions from the model (`y_pred_labels`).

The calculated metrics include accuracy (the overall percentage of correct predictions), precision (the ability of the model to avoid false positive predictions), recall (the ability of the model to find all correct positive data), and f1-score (the harmonic mean between precision and recall). All values will be displayed with four digits behind a comma, as well as printing a classification report that provides a breakdown of these metrics for each class, including support (number of examples per class), which helps in analyzing model performance in more detail. Precision is an evaluation metric that measures how many positive predictions are true compared to all positive predictions made by the model [25]. Precision is calculated based on the formula shown in Equation 3,

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

where TP (True Positive) is the number of completely positive and positively predicted samples by the model, while FP (False Positive) is the number of negative samples that are incorrectly predicted as positive. A high precision value indicates that the model has few errors in predicting positive classes. Then there is Recall, used to measure how many positive samples the model successfully detects compared to all the actual positive samples [25], with the calculation that can be seen in Equation 4,

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

where FN (False Negative) is the number of positive samples that are misclassified as negative. A high recall value means that the model managed to capture most of the existing positive samples, but it could be at the expense of lower precision. Lastly F1-Score, a combined metric that calculates the harmonic mean between precision and recall. This metric is used when a balance between precision and recall is required, especially if there is an imbalance in the number of samples between positive and negative classes [25]. Its formula can be seen in Equation 5,

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

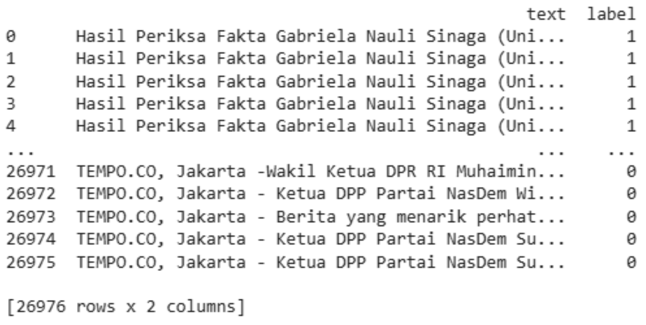
when the F1-Score is high if both precision and recall are high, while if one is slight, the F1-Score will also be low. Therefore, F1-Score is particularly useful in classification tasks where both False Positive and False Negative errors must be minimized. Followed by the evaluation of the Receiver Operating Characteristic (ROC) curve is a plot that shows the performance of the classification model at various decision levels.

It depicts the True Positive Rate (Recall) compared to the False Positive Rate (FPR), which is calculated by the formula FP / (FP + TN). As the graph gets closer to the upper left corner, it indicates the model's performance is getting better. And Area Under Curve (AUC), is a value between 0 and 1 that reflects the model's ability to distinguish positive from negative classes-a value closer to 1 indicates better performance. It ends with cross-validation, a method to assess the model that aims to evaluate how well the model can work on new data.

This model evaluation method is used to assess model performance more accurately by dividing the dataset into five parts or so-called “folds”. “Stratified” indicates that each fold will retain the same proportion of target classes (i.e., hoax versus valid) that existed in the original dataset. Thus, the class distribution within each fold will be similar to the class distribution in the entire dataset. This is very important when the dataset has an imbalance in class distribution.

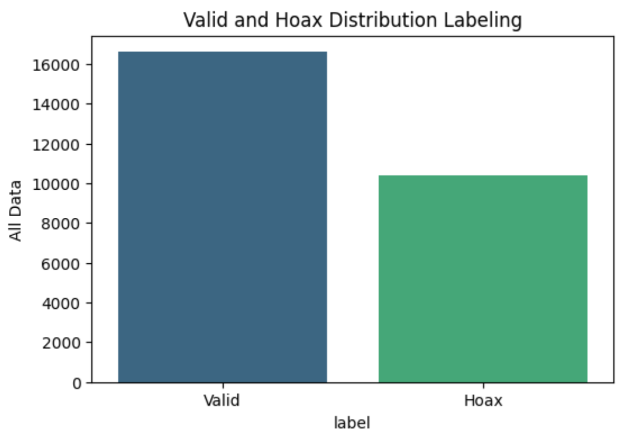
III. Result and Discussion

Raw data collection was obtained through the data science community website Kaggle (https://www.kaggle.com/) in February 2025 with the scope of discussion about hoaxes and valid news about political issues in Indonesia. This data was taken from online news portals like CNN Indonesia (https://www.cnnindonesia.com/), Tempo.co (https://www.tempo.co/), and Turnbackhoax (https://turnbackhoax.id/). The total data obtained from the three of them is 26976 data with information such as news titles, timestamps, news, tags, authors, and news links.



```
                                                    text  label
0      Hasil Periksa Fakta Gabriela Nauli Sinaga (Uni...      1
1      Hasil Periksa Fakta Gabriela Nauli Sinaga (Uni...      1
2      Hasil Periksa Fakta Gabriela Nauli Sinaga (Uni...      1
3      Hasil Periksa Fakta Gabriela Nauli Sinaga (Uni...      1
4      Hasil Periksa Fakta Gabriela Nauli Sinaga (Uni...      1
...                                                  ...    ...
26971  TEMPO.CO, Jakarta -Wakil Ketua DPR RI Muhaimin...      0
26972  TEMPO.CO, Jakarta - Ketua DPP Partai NasDem Wi...      0
26973  TEMPO.CO, Jakarta - Berita yang menarik perhat...      0
26974  TEMPO.CO, Jakarta - Ketua DPP Partai NasDem Su...      0
26975  TEMPO.CO, Jakarta - Ketua DPP Partai NasDem Su...      0

[26976 rows x 2 columns]
```



```
Dataset combined: 26976
```

Figure 3. Results after merging and labelling distribution on raw data

Shown in Figure 3, is the result of merging the three raw datasets in .xlsx format into .csv format. After merging, we labeled the datasets with 0 as valid or non-hoax news, and 1 as hoax news to make pre-processing easier.

Performing pre-processing to clean and prepare the raw text data before analysis. It starts with converting the entire text to lowercase, punctuation marks are removed using regular expressions, which are then split into word lists (tokenization). Furthermore, stopword removal stage is here, aims to remove words in Indonesian that do not provide important information, such as "*dan*", "*di*", "*yang*", etc. These words will be filtered from the token list, and then the results will be merged back into clean text without the word.

Five random examples of text data before and after removing unnecessary words are shown. Each text is also accompanied by its label. The "`text`" column displays the original version of the text, while the "`clean_text`" column is the text that has been stripped of punctuation, capitalization, and stopwords, making it more concise and ready for analysis or modeling. The "`label`" column indicates the category or class for each text, for example whether it is fake news or not, which can be seen in Table I.

TABLE I
BEFORE AND AFTER STOPWORD REMOVAL

text	clean_text	label
TEMPO.CO, Jakarta – Dukungan untuk Partai Gol…	tempoco jakarta dukungan untuk partai golongan…	0
Hasil Periksa Fakta Rizky Maulana (Univ…	hasil periksa fakta risky maulana universitas…	1
TEMPO.CO, Jakarta – TNI Angkatan Darat…	tempoco jakarta tni angkatan darat merespons…	0
Jakarta, CNN Indonesia – Pakar hukum pemilu…	jakarta cnn indoneisa pakar hukum pemilu univ…	0
TEMPO.CO, Jakarta – Lembaga survey indi…	tempoco jakarta lembaga survei indikator poli…	0

This is continued by capturing the meaning and semantic relationships between words with FastText, which converts each word into a vector that has 300 dimensions. Next, to generate one representative vector, the average of all word vectors contained in the text is calculated. After the text is converted into vectors, an oversampling process is carried out using the SMOTE technique. This method generates new data for minority groups in a synthetic way through interpolation between existing samples, so that the class distribution becomes more balanced. The results of the SMOTE technique can be seen in Figure 4.

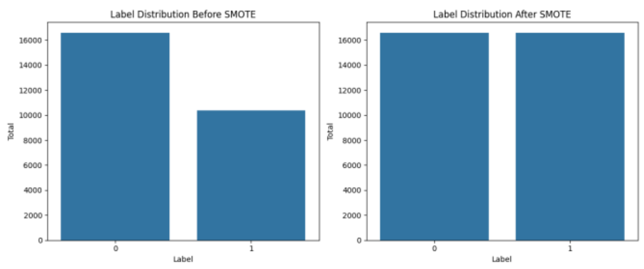
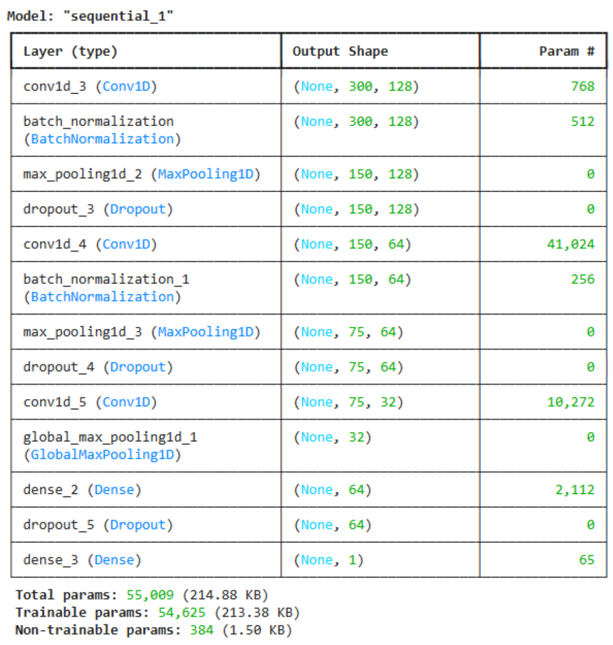


Figure 4. Label distribution after SMOTE technique been done

It is important to ensure that the classification model does not favor more classes and can learn fairly from all classes. Entering the processing part, the CNN model built consists of several main layers for the binary classification task. A summary of the CNN model can be seen in Figure 5.



```
Model: "sequential_1"

Layer (type)                            Output Shape            Param #
conv1d_3 (Conv1D)                       (None, 300, 128)            768
batch_normalization                     (None, 300, 128)            512
(BatchNormalization)
max_pooling1d_2 (MaxPooling1D)          (None, 150, 128)              0
dropout_3 (Dropout)                     (None, 150, 128)              0
conv1d_4 (Conv1D)                       (None, 150, 64)          41,024
batch_normalization_1                   (None, 150, 64)             256
(BatchNormalization)
max_pooling1d_3 (MaxPooling1D)          (None, 75, 64)                0
dropout_4 (Dropout)                     (None, 75, 64)                0
conv1d_5 (Conv1D)                       (None, 75, 32)           10,272
global_max_pooling1d_1                  (None, 32)                    0
(GlobalMaxPooling1D)
dense_2 (Dense)                         (None, 64)                2,112
dropout_5 (Dropout)                     (None, 64)                    0
dense_3 (Dense)                         (None, 1)                    65

Total params: 55,009 (214.88 KB)
Trainable params: 54,625 (213.38 KB)
Non-trainable params: 384 (1.50 KB)
```

Figure 5. Summary of CNN model

The model has several important components: there are three convolution blocks at the beginning, followed by a fully connected (dense) section. Each convolution block consists of a `Conv1D` layer that serves to extract features, `BatchNormalization` that helps stabilize the data distribution, `MaxPooling1D` that is used to shorten the sequence length, and `Dropout` that serves as a regularization method to prevent the model from overfitting. This layered structure reflects CNN's general approach in understanding the representation of sequential data such as text or signals. On this occasion, we also implemented a CNN model that does not use the Batch Normalization layer to compare how much difference there is between CNN models with and without Batch Normalization (BN). The visualization can be seen in Figure 6.

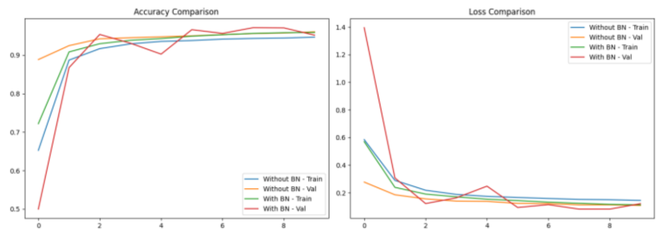


Figure 6. Comparison visualization of CNN model with and without Batch Normalization

In the accuracy graph, both models show a steady increase in accuracy as the epochs increase. However, the model using Batch Normalization appears to provide slightly better and more consistent validation accuracy results, especially after the second epoch. This indicates that BN helps the model become better at generalizing the validation data. In the loss graph, it can be seen that the model using Batch Normalization shows a faster and more stable decrease in loss than the model without BN.

The model without BN shows a higher validation loss initially and experiences smaller fluctuations afterwards, while the model with BN experiences a significant and more stable decrease in loss value, although slight fluctuations occur in some epochs. Overall, this figure indicates that the application of BN can accelerate the convergence process and improve the generalization performance of the model.

The three blocks create a multilevel feature extraction stage, where the number of filters progressively decreases (from 128 to 64 to 32), but each layer learns more specialized features. In the last block, `GlobalMaxPooling1D` is applied to summarize all the features from a single filter into a single maximum value, resulting in a short representation of the input. This is crucial for the efficient working of the model without having to keep the entire length of the original sequence, which can be lengthy and risks overfitting.

After the feature extraction process is executed, it is shown that the summarized features go to the dense layer with 64 neurons that apply ReLU activation, followed by `Dropout`. Finally, the output layer consists of 1 neuron that applies sigmoid activation to output the class probability. This diagram explains how the data moves from unprocessed input, through multiple stages of feature filtering, to outputting probabilities ready for classification. This structure reflects the balance between representativeness and the application of dropouts and normalization at each stage.

The process to assess the performance of the CNN model will be performed using the 5-Fold Stratified Cross Validation method. This method divides the dataset into 5 sections (folds) while ensuring that the proportion of class labels remains balanced within each section. For example, if the dataset consists of 60% class 0 and 40% class 1, then each section will also have almost the same class distribution, i.e. 60:40.

Model is built with the `build_model(input_shape)` function which consists of three convolution blocks. Each block applies a `Conv1D` layer to draw features, followed by `BatchNormalization()` to stabilize the activation distribution, `MaxPooling1D()` to reduce the spatial dimension, and `Dropout(0.5)`. After three convolution blocks, the model applies `GlobalMaxPooling1D()` and is then augmented with two dense layers, with one of them serving as the output using sigmoid activation for binary classification.

The model was compiled using Adam's optimizer and the `binary_crossentropy` loss function. The CNN model in each fold of cross-validation is regenerated because each fold requires a model that has never been trained before (fresh model). This has to be done so that the evaluation results are not biased because the model weights have been influenced by previous data.

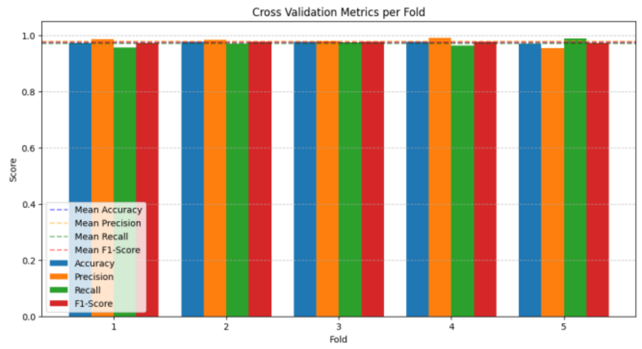


Figure 7. Visualization for the application of 5-Fold Stratified Cross Validation on the CNN model that has been built

In the cross-validation stage, the data is divided into five sections using `StratifiedKFold`, which ensures that the label distribution remains balanced in each section (results can be seen in the visualization of Figure 7). In each cycle, the model was rebuilt and trained for 10 epochs with the training data, then evaluated using the validation data. The input was reshaped into 3D to fit the `Conv1D` layer. After the training process, the predicted probabilities are converted into binary labels based on a threshold of 0.5. Next, evaluation metrics such as accuracy, precision, recall, and F1-score are calculated, which are then stored for averaging at the end. This method provides a deep understanding of the model's performance in the face of data variations, as well as helps guarantee that the model is not only precise, but also fair and consistent across data subsets.

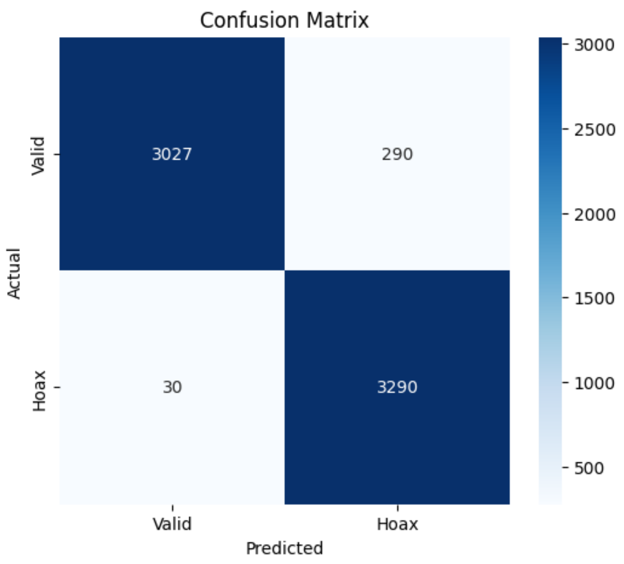
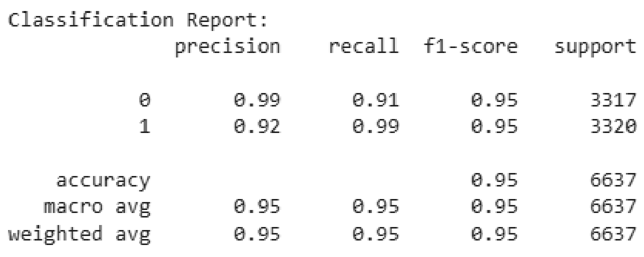


Figure 8. Confusion matrix visualization in heatmap form

The confusion matrix in Figure 8 of the model's classification results on the test data shows the predictive ability between two categories, namely Valid and Hoax. Out of the total data, the model correctly identified 3027 samples as Valid and 3290 samples as Hoax as well. However, there were 290 misclassifications where Valid samples were incorrectly identified as Hoaxes, and there were 30 errors for the reverse. This indicates that the model performs well, especially in detecting Hoaxes with few false negatives.

Next, a confusion matrix is created and displayed using `seaborn.heatmap`. This matrix is calculated with `confusion_matrix` without normalization (`normalize=None`), so the numbers that appear are absolute sums. The colour range on the heatmap is limited (`vmin, vmax`) by Valid rows only to adjust the visualization, so that the difference in colour intensity between cells is more visible. Labels on the axes and chart titles were added for clearer understanding of the matrix, and annotation numbers (`annot=True`) were used to indicate the number of predictions in each category.


```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.91      0.95      3317
           1       0.92      0.99      0.95      3320

    accuracy                           0.95      6637
   macro avg       0.95      0.95      0.95      6637
weighted avg       0.95      0.95      0.95      6637
```

Figure 9. Classification report for CNN model that has been built

Calculating basic metrics for classification evaluation such as accuracy, precision, recall, and f1-score can be done using the functions of `sklearn.metrics`. This is done based on the model's last predicted result (`y_pred_labels`) compared to the original label (`y_test`). Furthermore, the `classification_report` function is used to show a complete summary of the model's performance in tabular form. This includes the metrics for each class as well as the overall average. This is particularly useful for assessing the balance of performance between classes in binary classification situations.

Shows the classification report based on the evaluation of the binary classification model in Figure 9, which has labels 0 and 1. The precision, recall, and f1-score for each class are exceptional, proving that the model works well in identifying class 0 (Valid) and class 1 (Hoax). For class 0, the precision value reached 0.99, while the recall was slightly lower at 0.91. On the other hand, for class 1, recall was inordinate at 0.99, but precision dropped to 0.92. The F1-score for both classes remained consistent at 0.95, which is also reflected in the macro and weighted average values. This shows that the model performs great overall.

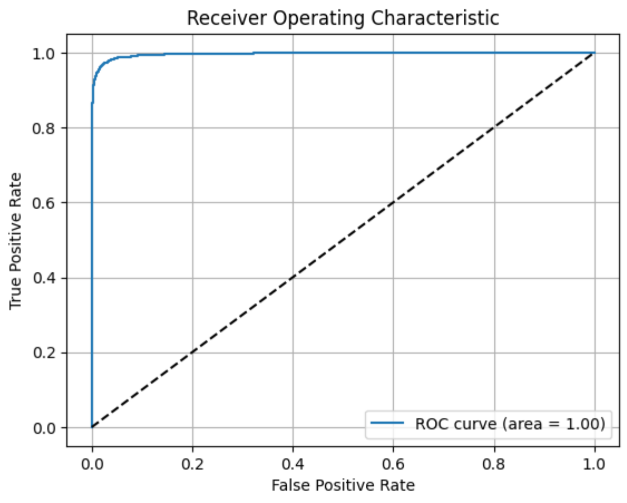


Figure 10. ROC curve visualization

Calculating and visualizing the ROC curve and AUC value of the model prediction is done in a specific way. To calculate the AUC value, `roc_auc_score` is used which compares the actual label (`y_test`) with the prediction from the model (`y_pred`). Furthermore, `roc_curve` generates values for the False Positive Rate (FPR), True Positive Rate (TPR), and threshold required to construct the ROC curve. Using `matplotlib`, the results are mapped, showing the diagonal dashed line as a reference (which indicates a random model), and the ROC curve is displayed with a label indicating the AUC value.

The ROC curve, known as Receiver Operating Characteristic, was used to evaluate classification model under test. The curve almost reaches upper left corner of the graph, indicating that the model is good in terms of predictive ability, with the AUC value almost reaching 1.00. This shows that the model is able to effectively distinguish between positive and negative classes, with a diminutive difference between True Positive Rate (sensitivity) and False Positive Rate.

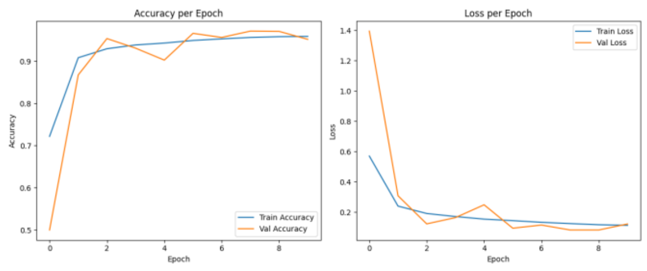


Figure 11. Visualization of CNN model training process

Figure 11 shows two graphs of the results from training the model for 10 epochs. On the left is a graph depicting accuracy, while on the right is a graph showing loss. In the accuracy graph, there is a rapid increase at the beginning, with the validation accuracy (`val_accuracy`) and training accuracy (`train_accuracy`) almost reaching 0.95, without showing any obvious signs of overfitting.

Meanwhile, the loss graph shows a consistent decrease in loss for both training and validation data. Although the validation loss value fluctuates, it remains low at the end of the epoch, indicating that the model is quite stable and able to generalize well. With these results, the CNN algorithm is proven to be effective for use in classifying text and distinguishing valid and hoax news. It could be used as a hoax news detection foundation or text classifying for accurate results.

IV. Conclusion

This research utilized an Indonesian language dataset consisting of 26,976 news articles, which were divided into valid news and hoaxes. This dataset was taken from CNN Indonesia, Tempo, and Turnbackhoax portals. Although the distribution is fairly balanced, labeling and merging are still done to facilitate analysis. Each news story holds the content on which the classification is based, so a data preparation stage is essential before modeling is carried out.

The pre-processing stage is done through several steps, namely converting all characters to lowercase, removing punctuation by using regular expressions, performing word splitting, and removing common words in Indonesian using NLTK. After that, each word is converted into a numerical representation using FastText to better capture the meaning of the word in context. To solve the class imbalance problem, an oversampling technique through SMOTE is applied so that the label distribution becomes more even and the model does not tend to the larger class.

In data processing, data that has been cleaned and converted into vector form is fed into a Convolutional Neural Network (CNN) model. This model has a structure consisting of three convolution blocks, and each block is equipped with BatchNormalization, MaxPooling1D, and Dropout. Next, the extracted features are summarized using GlobalMaxPooling1D and passed to the fully connected layer before exiting to the output layer which uses sigmoid activation for binary classification. The model was trained for ten epochs with 80% data split for training and 20% for testing, using Adam's optimizer and binary crossentropy loss function.

The model was thoroughly evaluated by measuring several metrics, such as accuracy, precision, recall, f1-score, and using confusion matrix. The final results showed that the model's training accuracy was 95.77% and validation accuracy reached 95.18%, with a training loss of 0.1146 and validation loss of 0.1210. For both classes, the f1-score value was 0.95, while the ROC AUC almost reached 1.00, indicating good classification ability. The confusion matrix shows that the model correctly predicted 3027 valid data and 3290 hoax data, with misclassification only occurring in 290 valid data that were incorrectly identified as hoaxes and 30 errors otherwise. In addition, the application of 5-Fold Stratified Cross Validation provides consistent evaluation results for each fold, indicating that the model can generalize well and not only depend on one group of data. Overall, the model performed well in detecting text-based hoax news.

Based on the results obtained, it can be concluded that the combination of CNN with FastText, BatchNormalization, SMOTE, and cross-validation is successful in detecting fake news well, precisely, and consistently. For future research, it is recommended to test other architectures such as LSTM, Bi-LSTM, or Transformer, and also to consider using additional data such as headline, date, or news source to improve accuracy. In addition, it is important to use a larger number of cross-validation folds and increase the variety of data from different topics so that the model can be more flexible to the changing patterns of hoax news.

References

[1] M. Athaillah, Y. Azhar, and Y. Munarko, "Perbandingan Metode Klasifikasi Berita Hoaks Berbahasa Indonesia Berbasis Pembelajaran Mesin," *REPOSITOR*, vol. 2, no. 12, pp. 1700–1705, 2020, [Online]. Available: www.trunbackhoax.id

[2] F. Diani *et al.*, "Prosiding the 15 th Industrial Research Workshop and National Seminar Bandung," 2024.

[3] C. Andreas, S. Priandi, A. N. M. B. Simamora, and M. F. F. Mardianto, "Analisis Hubungan Media Sosial dan Media Massa dalam Penyebaran Berita Hoaks berdasarkan Structural Equation Modeling-Partial Least Square," *MUST J. Math. Educ. Sci. Technol.*, vol. 6, no. 1, p. 81, Jul. 2021, doi: 10.30651/must.v6i1.8816.

[4] V. Ramadhan and A. Pambudi, "Implementasi Algoritma Convolutional Neural Network Untuk Mengidentifikasi Berita Hoaks Berbahasa Indonesia," 2024.

[5] C. S. Sriyano and E. B. Setiawan, "Pendeteksian Berita Hoax Menggunakan Naive Bayes Multinomial Pada Twitter dengan Fitur Pembobotan TF-IDF."

[6] "Kementerian Komunikasi dan Digital." Accessed: Apr. 14, 2025. [Online]. Available: https://www.komdigi.go.id/berita/siaran-pers/detail/komdigi-identifikasi-1923-konten-hoaks-sepanjang-tahun-2024

[7] "Deteksi Berita Hoax Dengan Pendekatan Lexicon Based Dan Lstm Thesis Oleh : Edwin Hari Agus Prastyo Nim. 220605220005 Program Studi Magister Informatika Fakultas Sains Dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang 2024."

[8] S. Imron, E. I. Setiawan, and J. Santoso, "Deteksi Aspek Review E-Commerce Menggunakan IndoBERT Embedding dan CNN," *J. Intell. Syst. Comput.*, vol. 5, no. 1, pp. 10–16, Apr. 2023, doi: 10.52985/insyst.v5i1.267.

[9] W. Hidayat, E. Utami, A. F. Iskandar, A. D. Hartanto, and A. B. Prasetio, "Perbandingan Performansi Model pada Algoritma K-NN

terhadap Klasifikasi Berita Fakta Hoaks Tentang Covid-19," *Edumatic J. Pendidik. Inform.*, vol. 5, no. 2, pp. 167–176, Dec. 2021, doi: 10.29408/edumatic.v5i2.3664.

[10] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," *Glob. Transitions Proc.*, vol. 3, no. 1, pp. 91–99, Jun. 2022, doi: 10.1016/j.gltp.2022.04.020.

[11] N. Alvi Hasanah, Nanik Suciati, and Diana Purwitasari, "Pemantauan Perhatian Publik terhadap Pandemi COVID-19 melalui Klasifikasi Teks dengan Deep Learning," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 5, no. 1, pp. 193–202, Feb. 2021, doi: 10.29207/resti.v5i1.2927.

[12] P. R. Gupte *et al.*, "A guide to pre-processing high-throughput animal tracking data," *J. Anim. Ecol.*, vol. 91, no. 2, pp. 287–307, Feb. 2022, doi: 10.1111/1365-2656.13610.

[13] F. T. Sabilillah, S. Winarno, and R. B. Abiyyi, "Implementasi BERT dan Cosine Similarity untuk Rekomendasi Dosen Pembimbing berdasarkan Judul Tugas Akhir," *Edumatic J. Pendidik. Inform.*, vol. 8, no. 2, pp. 585–594, Dec. 2024, doi: 10.29408/edumatic.v8i2.27791.

[14] S. Sarica and J. Luo, "Stopwords in technical language processing," *PLoS One*, vol. 16, no. 8 August, Aug. 2021, doi: 10.1371/journal.pone.0254937.

[15] R. E. kalaivani and R. MarivendanE, "The Effect of Stop Word Removal and Stemming In Datapreprocessing," 2021. [Online]. Available: http://annalsofrscb.ro

[16] A. Tabassum and R. R. Patil, "A Survey on Text Pre-Processing & Feature Extraction Techniques in Natural Language Processing," *Int. Res. J. Eng. Technol.*, 2020, [Online]. Available: www.irjet.net

[17] M. Kamyab, G. Liu, and M. Adjeisah, "Attention-Based CNN and Bi-LSTM Model Based on TF-IDF and GloVe Word Embedding for Sentiment Analysis," *Appl. Sci.*, vol. 11, no. 23, Dec. 2021, doi: 10.3390/app112311255.

[18] M. Umer *et al.*, "Impact of convolutional neural network and FastText embedding on text classification," *Multimed. Tools Appl.*, vol. 82, no. 4, pp. 5569–5585, Feb. 2023, doi: 10.1007/s11042-022-13459-x.

[19] D. Elreedy, A. F. Atiya, and F. Kamalov, "A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning," *Mach. Learn.*, vol. 113, no. 7, pp. 4903–4923, Jul. 2024, doi: 10.1007/s10994-022-06296-4.

[20] D. S. Silalahi, M. M. Santoni, and A. Muliawati, *Implementasi Convolutional Neural Network Untuk Klasifikasi Kata Pada Citra Teks*.

[21] "JEPIN (Jurnal Edukasi dan Penelitian Informatika) Penerapan Convolutional Neural Network (CNN) pada Pengenalan Aksara Lampung Berbasis Optical Character Recognition (OCR) Agus Mulyanto #1 , Erlina Susanti #2 , Farli Rosi #3 , Wajiran #4 , Rohmat Indra Borman #5", [Online]. Available: https://colab.research.google.com.

[22] V. Q. Nguyen, T. N. Anh, and H. J. Yang, "Real-time event detection using recurrent neural network in social sensors," *Int. J. Distrib. Sens. Networks*, vol. 15, no. 6, Jun. 2019, doi: 10.1177/1550147719856492.

[23] M. Fadli and R. A. Saputra, "Klasifikasi Dan Evaluasi Performa Model Random Forest Untuk Prediksi Stroke Classification And Evaluation Of Performance Models Random Forest For Stroke Prediction," vol. 12, [Online]. Available: http://jurnal.umt.ac.id/index.php/jt/index

[24] K. Kristiawan and A. Widjaja, "Perbandingan Algoritma Machine Learning dalam Menilai Sebuah Lokasi Toko Ritel," *J. Tek. Inform. dan Sist. Inf.*, vol. 7, no. 1, Apr. 2021, doi: 10.28932/jutisi.v7i1.3182.

[25] G. Ayu, V. Mastrika Giri, and L. Radhitya, "Musical Instrument Classification using Audio Features and Convolutional Neural Network," 2024. [Online]. Available: http://jurnal.polibatam.ac.id/index.php/JAIC