Journal of Soft Computing Exploration

Homepage: htttps://shmpublisher.com/index.php/joscex

p-ISSN: 2746-7686
e-ISSN: 2746-0991

`

Hoax Classification in Indonesian Language with Bidirectional Temporal Convolutional Network Architecture

Fajar Maulana[1], Tri Sri Noor Asih[2]
[1,2]Department of Mathematics, Universitas Negeri Semarang, Indonesia

Article Info

Article history:

Received Jan 3, 2023
Revised Jan 18, 2023
Accepted Jan 19, 2023

Keywords:

Deep Learning Model
Text Classification
Hoax
Bi-TCN

ABSTRACT

The increasingly massive rate of information dissemination in cyberspace has had several negative impacts, one of which is the increased vulnerability to the spread of hoaxes. Hoax has seven classifications. Classification problems such as hoax classification can be automated using the application of the Deep Learning model. Bidirectional Temporal Convolutional Network (Bi-TCN) is a type of Deep Learning architectural model that is very suitable for text classification cases. Because the architecture uses dilation factors in its feature extraction so it can generate exceptionally large receptive fields and is supported by Bidirectional aggregation to ensure that the model can learn long-term dependencies without storing duplicate context information. The purpose of this study is to evaluate the performance of Bi-TCN architecture combined with pre-trained FastText embedding model for hoax classification in Indonesian and implement the resulting model on website. Based on the research that has been done, the model with Bi-TCN architecture has satisfactory performance with an accuracy score of 92.98% and a loss value that can be reduced to 0.191. Out of a total of 13,673 data tested with this model, only 414 data or in other words around 3% of the total data were incorrect predictions.

Corresponding Author:

Fajar Maulana,
Department of Mathematics,
Universitas Negeri Semarang,
Sekaran, Gunungpati, Semarang, Indonesia.
Email: fajarmaulana_math@students.unnes.ac.id

1. INTRODUCTION

Along with the increase in internet service users in the world, the dissemination of information in cyberspace is also increasingly massive. It is easier for people to get information in the form of text, images, audio, and video through various online media. Social media is the easiest online media to use to exchange information. However, behind this convenience there are also problems that arise from the existence of social media, that is the vulnerability to spreading hoaxes from one person to another quickly. Based on data from the Ministry of Communication and Information Technology in 2017, there are 800,000 websites in Indonesia which are indicated to be spreading false information and hate speech [1]. This condition is even more concerning when a research results state that 44.19% of Indonesians cannot detect hoax news [2]. Until now, the detection of hoax news in Indonesia is still done manually by a social organization called Masyarakat Anti Fitnah Indonesia (MAFINDO) through news tracking and public complaints. However, the enormous number

of hoax news that is not matched by the number of detectors certainly makes the detection less than optimal. One solution that can be done to overcome this problem is to automate the task of hoax news detection.

Based on an analysis from First Draft, an international non-profit coalition in the field of journalistic ethics, misinformation and disinformation has seven classifications, namely satire/parody, misleading content, imposter content, fabricated content, false connection, false context, and manipulated content [3]. This classification problem can be solved automatically using Deep Learning models. There have been many studies on the classification of hoaxes in Indonesian, from those using word similarity measurement theories such as Text Rank and Dice Similarity [4], Extreme Gradient Boosting [5], classic machine learning methods such as Naive Bayes [6]–[9], K-Nearest Neighbor [10], Random Forest [7], Decision Tree [7], and Support Vector Machine [6], [7], [11], to deep learning methods such as Convolution Neural Network (CNN) [6], [12], Recurrent Neural Network (RNN) [13], Long Short-Term Memory (LSTM) [6], [14], and Gated Recurrent Unit (GRU) [6], [12], [14]. The deep learning methods that have been widely used to classify hoaxes in Indonesian language each have advantages and disadvantages. Convolutional architecture has an advantage in its feature extraction capabilities but does not have the ability to remember previous information, because the context analysis in the convolution architecture is based on patterns of several texts. Meanwhile, the recurrent architecture is designed with the ability to consider the order of information in a very sophisticated method, but its feature extraction capabilities are not as good as convolutional architecture. To overcome the shortcomings of each of these architectures, this study uses a new proposed architecture, namely Bidirectional Temporal Convolutional Network (Bi-TCN). TCN is a dilated-causal version of CNN and is an alternative to recurrent architectures that can handle input in the form of long sequential data without getting missing gradient problems and has longer memory than recurrent architectures with same capacity [15]. The convolution of each layer on TCN has an exponentially increasing dilation factor allowing the first layer to look for short-term connections in the data whereas the deeper layers show long-term dependencies based on the features extracted by the previous layer [16]. Because the architecture still uses the convolution layer, TCN also still has the properties of CNN in terms of feature extraction. However, TCN also uses a generic residual module instead of a convolution layer where a residual block leads to a series of transformations F whose output is added to the x input of the block $o = \text{Activation}(x + F(x))$ and effectively allows the layer to learn modifications to identity mapping of all transformations, which has repeatedly proven useful for very deep networks [17]. In addition, the TCN architecture has also been used in several studies [15], [18]–[21] and achieved good evaluation results. Meanwhile, the use of bidirectional aggregation is intended to ensure that the model can learn long-term dependencies without storing duplicate context information [22]. Based on this description, the TCN architecture with bidirectional aggregation can be a good combination to be applied to text classification cases such as Indonesian hoax classification.

This study proposes a multi-class classification that can produce more classification classes according to First Draft standard so that it can better explain why a narrative is classified as a hoax. In addition, the final result, which is no longer just a model, but a user-friendly website is intended as a continuation of previous research that wants to develop a simple web system to apply the model that have been made in subsequent studies [12]. So that the resulting hoax detector model can be easily used by anyone including ordinary internet users though.

2. METHOD

In the implementation of this study, there are several stages carried out from designing model, implementing model on website, to model improvement. The main steps involved in designing the model consist of data collection and cleaning, data preprocessing, word embedding, neural network architecture design, training process, and model filtration based on evaluation scores and model predictive ability. Meanwhile, the main steps involved in implementing model on website are creating backend applications, deploying backend applications, creating wireframe and mockup design of site, creating responsive website, developing site functionality, integrating site with model, and deploying site. While the improvement phase consists of usability testing and model improvement stages. Visualization of the process carried out in this study can be seen in Figure 1.

`

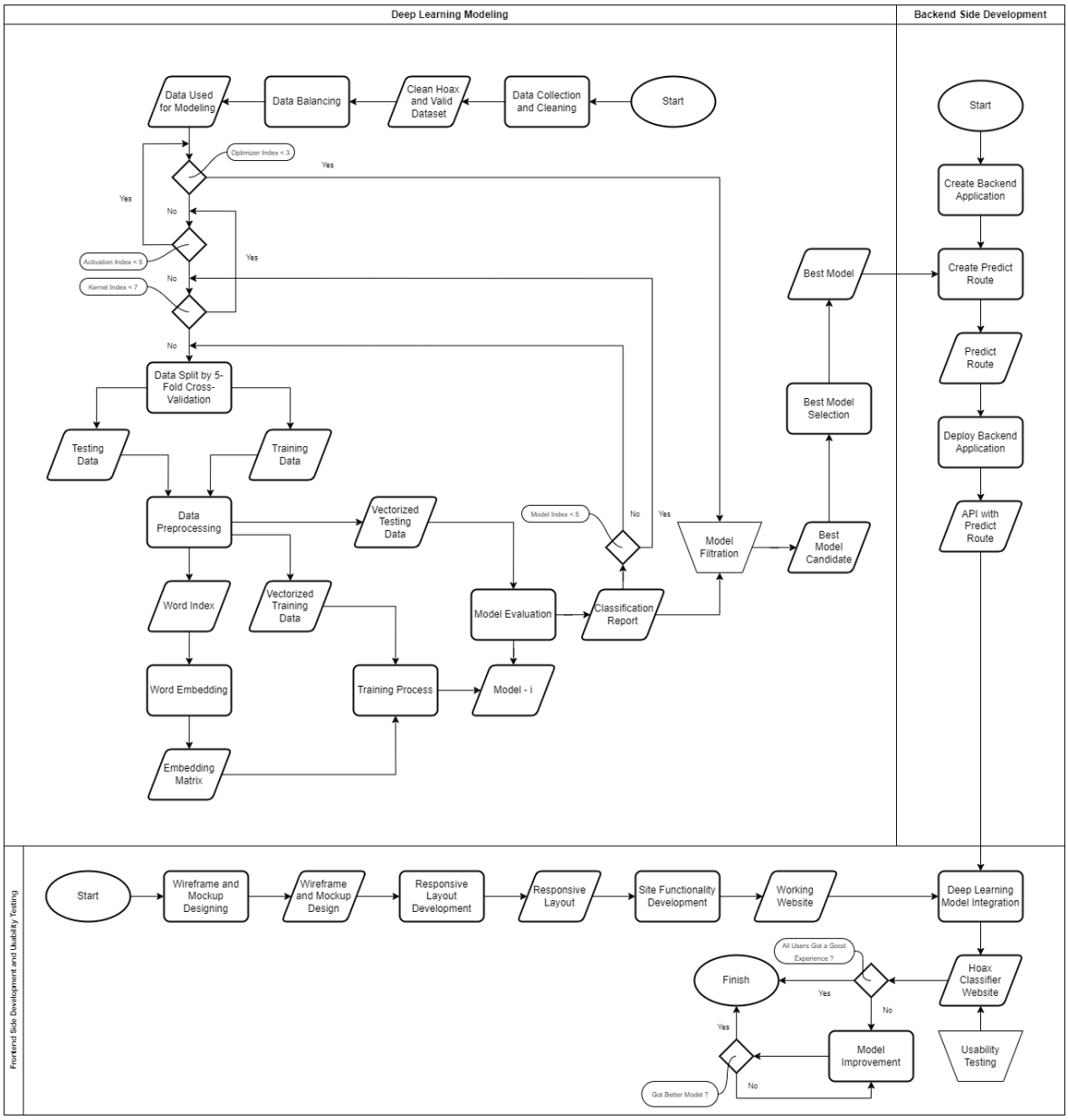

Figure 1. Visualization of research flow

2.1 Data Collection and Cleaning

The data used in this study is data obtained through web scraping at turnbackhoax.id and 20.detik.com where 20.detik.com is part of news.detik.com. The data taken is the post date of the article, the title of the article, the article narration, and the class hashtag (the class hashtag for the news.detik.com site is written as valid news). Web scraping is done with the help of the beautiful soup module in Python. The dataset obtained will consist of 8 classes, namely Valid, Satire/Parody, Misleading Content, Fabricated Content, False Context, Manipulated Content, Imposter Content, and False Connection. The data that has been obtained is then cleaned in several steps, starting from lowercasing, cleaning non-ascii characters, removing links, removing account names and hashtags, removing foreign language articles, normalizing non-standard words, removing stop-words to reduce noise on data [23], removing duplicate articles, and removing rows with empty narration due to the data cleaning process. After the data is clean, then the dataset is exported in (.csv) format.

2.2 Data Preprocessing

The purpose of data preprocessing is to determine the amount of data used at the training stage and to convert text and category labels into a series of numbers. The first thing that needs to be done, of course, is to load the dataset first. The dataset is then cleaned of data containing empty cells and duplicate narrations. After that, the Imposter Content and Fabricated Content classes were merged into a new class named Imposter Content/Fabricated Content. Misleading Content, False Context, and Manipulated Content classes were also

merged into a new class called Misleading Content/False Context/Manipulated Content. Meanwhile, False Connection and Satire/Parody class data are removed from the dataset. The merging and deletion of several classes is due to the unequal distribution of a lot of data on misinformation classes and to maximize the number of categories of misinformation that can be predicted based on the similarity of the characteristics of the classes combined. After the class merging stage, a new column is then created to display the number of words from each narrative data. Based on the number of words in each narrative data, then the data used for training is determined based on the desired range of words. In this study, data with a narrative that contains at least 3 words is used. After that, the data is then balanced based on the amount of data from each class. The data selection phase ends by creating a new data frame that contains data from the number of occurrences of each word in the dataset and the number of different words in the dataset.

The value of the many different words is later used as the padding size of the token vector sequence, while the value of the highest number of occurrences of the word is used as the maximum number of words that will be interpreted in the embedding matrix. Embedding matrix is a matrix containing random numbers where each cell represents the meaning of each word used for training and generated by the embedding layer. After that, each word in the selected data is converted into a token or word sequence and then converted again into a vector with the length of the vector according to the size of the padding. With padding, sentence vectors with a length less than the padding size will be filled with zeros at the beginning or end of the vector according to the type of padding used. The vector value of each token is based on the number of occurrences of the token in the dataset. If there are 10,000 different words in the dataset, then the vector value of the word with the highest frequency of occurrence is 2. Conversely, the vector value of the word with the least frequency of occurrence and is the last word in the dataset is 10001. Meanwhile, the value of vector 1 will later be used by the OOV token [24], that is words that are not interpreted in the embedding matrix. Each token and each vector value are then combined into a data object type called word index in a new variable, where the token becomes the key, and the vector value becomes the value. This word index will later be used as a reference in model prediction because the vector value of each token itself is used for meaning extraction of embedding matrix.

2.3 Word Embedding

For modeling that uses pre trained FastText embedding, it is necessary to create an embedding matrix before entering the model training stage. FastText is a development of Word2Vec where the Word2vec approach uses two neural networks, namely continuous bag of words (CBOW) and continuous skip-gram to create a high-dimensional vector of each word and is one of the most frequently used word embedding methods [25]. FastText also shows the better result for supervised text classification case than other text embedding method such as GloVe [13]. FastText vectors for Indonesian words can be downloaded via https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.id.300.vec.gz.

After the file is extracted, the vector values contained in it are entered into a new variable so that later it can be used to create an embedding matrix. The final stage is creating the generator embedding matrix function. This function requires two parameters, that is word index and vocab size, where the value of vocab size is the length of the word index plus 1 and the added value of 1 represents the additional value of the OOV token.

2.4 Neural Network Architectural Design

At this stage, the architectural design of the neural network is carried out for further compilation and model creation. This neural network architecture consists of an input layer, embedding layer, spatial dropout layer, TCN layer, pooling layer, dropout layer, and finally the classification layer. One TCN block will be used bidirectionally with 64 filters and a dilation factor of 2. The formula for the dilated convolution layer written by [20] can be seen in equation 1.

$$G(x_t) = (x *_d g)(t) = \sum_{k=0}^{h-1} x_{t-d \cdot k} g^T(k) \quad (1)$$

As for the kernel size, it will be chosen between 2, 3, 4, 5, 6, 7, or 8 which can produce the best model. Five different activation functions namely tanh, relu, elu, selu, and swish will also be used as options in the training process [26]. In this architecture, an ensemble pooling approach will also be applied, in which two types of pooling layers are combined to achieve the best sub-sampling results and will then pass through the

dropout layer and classification layer with 3 neurons according to many classes and the softmax activation function according to the model criteria. The general architecture of TCN model can be seen in Figure 2. Meanwhile, the specific architecture used in this study can be seen in Figure 3.

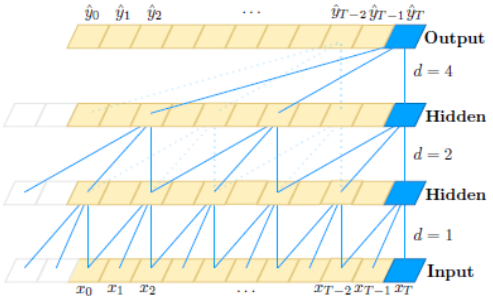

Figure 2. General architecture of Temporal Convolutional Network

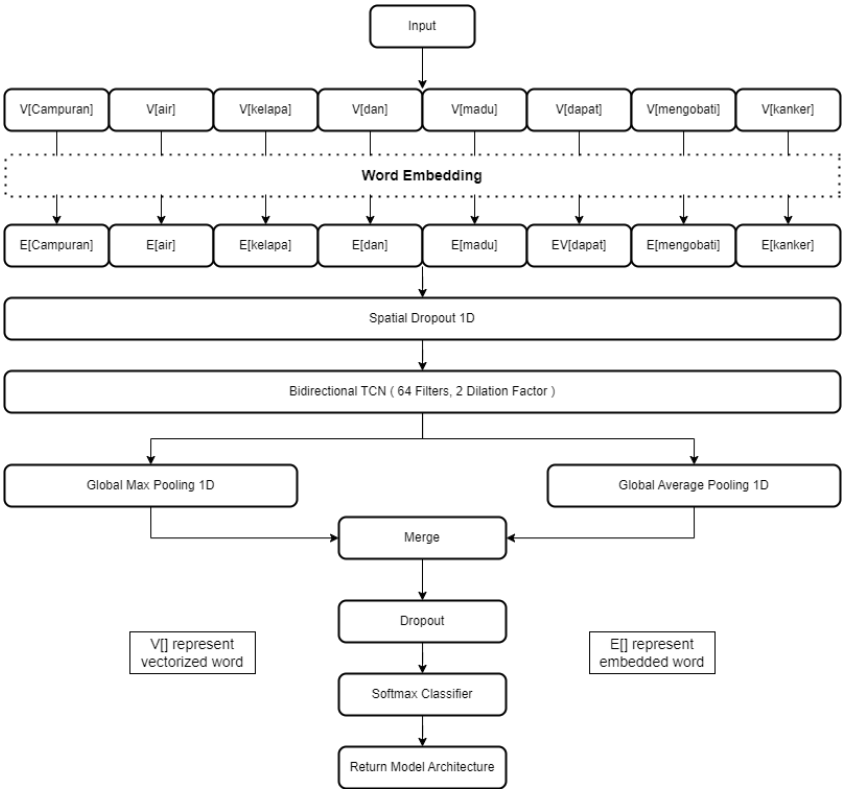

Figure 3. Specific architecture used in this study

2.5 Training Process

Model training uses the for-loop approach and 5-Fold Cross-Validation because this study will test many combinations of parameters as previously described. 5-Fold Cross Validation is the implementation of K-Fold Cross-Validation with $k = 5$. This method is used for automation the data distribution [27], [28]. The training will also use two different sizes of embedding matrix, that are the embedding matrix that is generated with all the words in the corpus and the embedding matrix that is generated with a corpus without words that appear only once in the corpus, or in other words considers a word with frequency one as an OOV token. Therefore, the model architecture is created first in a function to make it easier to use in loops. The callback function is also used to stop the training process when the model stops learning, that is, when the loss value starts to increase monotonically. Using the callback function can also automatically select the best weight. As for compilation, it uses 3 different optimizer options, namely Nadam [29], Adam [29], and Adamax [30]. And for the loss function, categorical cross entropy is used because the model created is a multi-class classification model and available in Keras API [31].

Furthermore, in the for loop, several variables are created to obtain information from each training process, where the information will later be downloaded and used for interpretation of results, model filtration, as well as making prediction logic and predictive explanation at the model testing and API development stages.

2.6 Model Filtration

When all the training processes have been completed, the model creation before the implementation of usability testing resulted 3150 different models. So, it is necessary to take a special approach using the model evaluation score to select the model with the best performance [32]. The first filter is based on the recall scores (for balanced data cases) or f1 scores (for unbalanced data cases) of the misinformation classes. Models that have a recall score or f1 score of more than 0.8 are then selected to be used at the model testing stage or selecting the best model.

2.7 Best Models Selection

At this stage, the previously selected models are tested against all data in the dataset. The model that correctly predicts the most data in the dataset is then chosen as the best model and downloaded along with the word index, classification report, and other supporting files that will be used in the backend application development stage.

2.8 Backend Side Development

Model implementation on website is done by converting the model into an Application Programming Interface (API). The API is a link between the frontend and the server which allows the frontend to be able to send requests to the server. The development of the backend side in this study uses one of the modules supported by the Python language, namely FastAPI. FastAPI supports creating functions with all http methods, namely Create, Read, Update, Delete (CRUD) and can respond properly according to client requests [33]. Furthermore, so that the API can be used on certain ports, additional modules are also used to support this, one of which is Uvicorn. As for the need to build backend applications, this research uses Docker. Docker is a platform that works with the container concept to bundle applications, associated dependencies, and system modules so that applications can work in any environment [34]. The steps involved in creating the backend application consist of initiating the creation of the application using the FastAPI function from the fastapi module, configuring middleware, creating functions for cleaning and normalizing input data and removing stopwords with the same scenario as cleaning the dataset. Then create functions for tokenization and pad sequences using the index word model that was downloaded earlier. The next stage is model import. The architecture used in this study is not an architecture that has been supported by Tensorflow or Keras, but this architecture is still being built using the Keras layer. So it is necessary to add the custom_objects={“TCN”: TCN} definition when load the model, so that the model can be read by hardware.

Next is the creation of the prediction function. The prediction function contains some logic to generate data that will be displayed on the front end. The data generated from this function consists of normal narration from the results of data cleaning and normalization, narration without stopwords, final predictions, probability of each class, data on the number of occurrences of each word from the input sentence in each class, data on the number of words that dominate the most based on their occurrence in each class, the class that contains the most dominant words, data on the number of words from sentences contained in each class, and the name of the icon that is displayed according to the final prediction results. The next stage is the creation of the schema data and API route. At this stage a data schema is created from user input. Because the user input is a string type data which is likely to be quite long, the data schema that is suitable for use in API Route is an object type with one property of type string to be used in the body of the request. For the API Route function, it is enough to call the predict function that was created before and destruct all the values generated by the function in a new variable, and then return the value as an object with properties that contain all the values from the predict function. And the final stage of the backend application creation process is dynamic host and port configuration. Code can be run directly on localhost using uvicorn and any port. However, static ports cannot be used when the backend application has been deployed to the server. Therefore, before running the application, it is necessary to configure the host and dynamic port first with the help of the os module to obtain port values from the server environment and argparser to add and parse arguments. After the configuration is complete, the application can be run by adding the uvicorn.run() command to the main code.

`

2.9 Backend Side Deployment

The deployment process in this study uses the help of Docker to build code and Heroku as a hosting service. Once the rules are built in the Dockerfile, code can then be built as an image named “predict” by running the command “docker build -t predict .”. And to run predict images with containers, you can run the command “docker run predict” or “docker run -d -p 5000:5000 –name predictor predict” to run images in containers named custom “predictor” and on port 5000. After that, according to the documentation on heroku [35] , after first registering at https://heroku.com and creating a new application on heroku, to deploy a backend application to heroku, heroku-cli must first be installed by running the command “curl https: //cli-assets.heroku.com/install-ubuntu.sh | sh”. Then proceed by running the following commands sequentially.

1) heroku login
2) heroku container:login
3) heroku container:push web -a <app-name>
4) heroku container:release web -a <app-name>

After the process is complete, the endpoint generated by Heroku can be directly tested using Postman or used on the frontend side. However, to ensure that the API can be used by several people at once, it is necessary to scale up the disk on Heroku, because the slug size generated by the backend application with this deep learning model reaches around 700MB with normal use.

2.10 Wireframe Designing

At this stage, a wireframe of the site is made first as a rough description of the website that will be made to prepare the layout when entering the mockup design process.

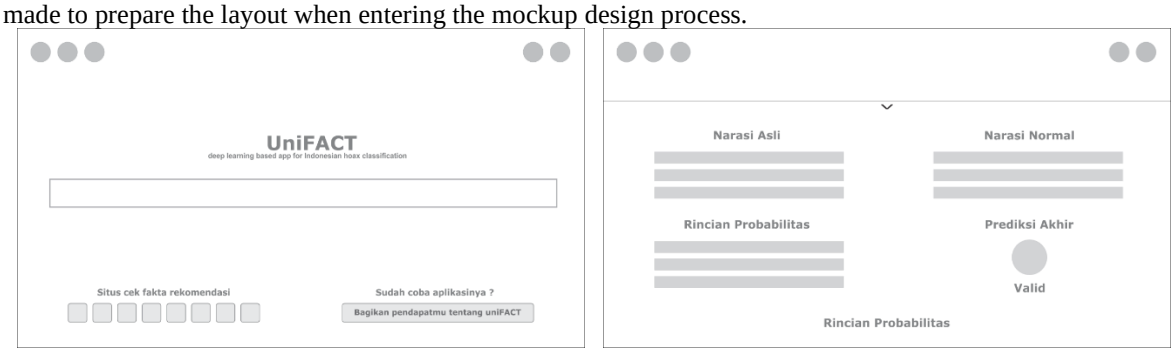

Figure 4. Wireframe design of the website

2.11 Mockup Designing

To make website more comfortable to view and easy to operate from the user experience aspect, the website with a defined layout of features needs to be designed with a mockup first. Mockup is used as a real and detailed description of the website that will be created along with its features as well as a reference for how the website will appear.

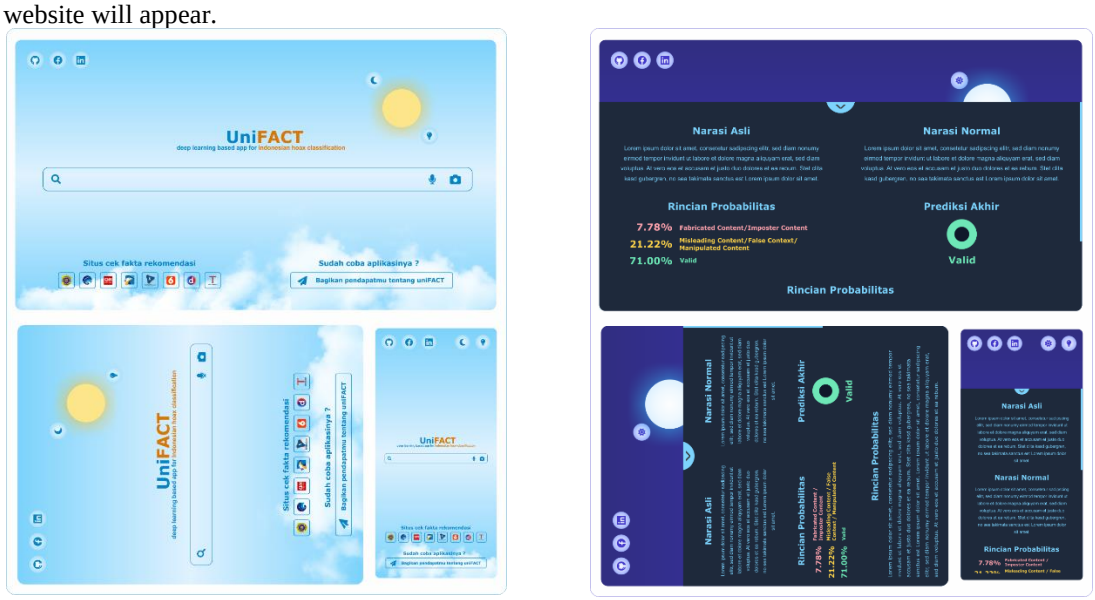

Figure 4. Mockup design of the website

2.12 Responsive Site Layout Development

From the existing mockup, localhost was created to carry out development on the frontend side. The framework used in the development is Vue which is further beautified using Tailwind CSS in color settings, layout, responsibility, and others. Vue is a progressive JavaScript framework for building user interfaces with Model-View-View-Model (MVVM) mode, where data and views are separated so that they require view model listeners to communicate with each other [36]. The reason for using Vue as a framework is because it is easier to use and lighter than other JavaScript frameworks such as React and Angular [37]. Vue which is a combination of concepts from React and Angular can increase website development efficiency, accommodate clearer coding logic, be flexible in dealing with changing needs, and the resulting system is easier to maintain and integrate to reduce development costs [38]. In addition, Vue can also display good data visualization from the prediction results of deep learning model [39]. Meanwhile, the programming language used in this study is TypeScript. TypeScript is a feature enhancement to JavaScript for more structured project development [40].

2.13 Site Functionality Development

At this stage, the site's functionality is added so that it can respond to events in the form of input, clicks, or other things that are done by the user. At this stage several other modules such as tesseract.js will also be used for Optical Character Recognition (OCR) needs.

2.14 Deep Learning Model Integration

Integration of deep learning model through the API on the frontend side using the help of the axios module by leveraging the ability of axios to make http requests. To use the generated endpoint, you first need to create an instance containing the service function to make a post request with axios.post(), where the function will later store the feedback generated by the API into a constant. The value of this constant will later be an object generated by the prediction function on the backend side. Furthermore, the function can be imported into the (.vue) file which contains the html code of the application and destructuring it to certain constants. The function resulting from destructuring is placed in the input component and made to be triggered when the following scenario occurs.

1) The user presses the enter key with a constant state that accommodates user input which is a string type data with a length greater than zero.
2) Users use the OCR feature, where the function can be immediately triggered when the OCR process is complete.
3) The user uses the speech-to-text feature and says “remember”. Speech with a function that is the opposite of "remember" is "forget". If the user says the word "forget", what the user has said will be reset.
4) The user uses the speech-to-text feature, then says a few words, and then stops saying. After the application does not acquire voice data again for a certain period, the function can be triggered immediately.

To make the data generated by the function displayable on website, then the data also needs to be inserted into html components as needed.

2.15 Frontend Side Deployment

To make the site accessible to everyone, the website needs to be registered with the hosting service. In this study, Netlify will be used to deploy the frontend side and then also used the Namecheap service so that the default domain generated by Netlify, namely .netlify.app, can be simplified to .me.

2.16 Usability Testing

Usability testing is one of the methods used to test the effectiveness of applications. Several usability testings approach that can be used are questionnaires, task completion, "think aloud" protocol, interviews, focus groups, and heuristic testing [41]. In this study, a questionnaire and focus group approach were used with a target of 115 respondents to obtain usability testing results.

2.17 Model Improvement

At this stage, model improvements are carried out based on the results of usability testing and parameter combination analysis based on the previous model. The improvement result is then used as the result of the study. Parameter combination analysis is carried out to find the best combination of parameters that can be used in the model improvement stage so that the model improvement stage can be carried out more quickly

`

and as needed. The data used for parameter selection are accuracy and loss tables from all previous training. The following criteria and weighting are used to make the analysis easier.

1) For each accuracy value in the table, if the accuracy value is more than 85, then it is given a weight of 1. If the accuracy value is in interval of [82, 85], then it is given a weight of 0. If the accuracy value is less than 82, then it is given a weight of -1. And if the accuracy value is included in the 5 highest accuracy values, then it is given a weight of 2.
2) For each loss value in the table, if the loss value is less than 0.35, then it is given a weight of 1. If the loss value is in interval of [0.35, 0.42], then it is given a weight of 0. If the loss value is more than 0.42, then it is given a weight of -1. And if the loss value is included in the 5 lowest loss values, then it is given a weight of 2.
3) The scores obtained in each line are then added up.
4) For each kernel size used, then the highest weight is chosen.
5) For each activation function used, then the highest weight is chosen.
6) For each size of the embedding matrix and the word embedding approach used, then the highest weight is chosen.
7) For each word embedding approach used, then the highest weight is chosen.
8) For each optimizer used, the highest weight is then chosen.

3. RESULTS AND DISCUSSIONS

The results presented in this study consist of the performance of the model before the implementation of usability testing, the performance of the model after the implementation of usability testing, and the results of implementing deep learning model on website.

3.1 Model Performance Before Usability Testing Implementation

Model performance analysis at this stage is divided into 6 scenarios and 18 experiments according to the combination of parameters tested.

Scenario 1. Random Embedding and All Words in Corpus

Experiments 1, 2, and 3 use the Random Embedding approach to produce an embedding matrix. The first experiment using the adam optimizer produces the best model with an accuracy of 86.74% and a loss of 0.3707 and can predict 6006 data correctly. The second experiment using the adamax optimizer produces the best model with an accuracy of 86.74% and a loss of 0.3379 and can predict 6132 data correctly. While the third experiment using the Nadam optimizer produced an accuracy of 87.33% and a loss of 0.3557 and was able to predict 6029 data correctly.

Scenario 2. Non-Trainable FastText Embedding and All Words in Corpus

Experiments 4, 5, and 6 use Non-Trainable FastText Embedding approach to produce an embedding matrix. The fourth experiment using the adam optimizer produces the best model with an accuracy of 88.36% and a loss of 0.3591 and can predict 6089 data correctly. The fifth experiment using the adamax optimizer produces the best model with an accuracy of 87.48% and a loss of 0.3704 and can predict 6034 data correctly. While the third experiment with the Nadam optimizer produced an accuracy of 86.89% and a loss of 0.3708 and was able to predict 6236 data correctly.

Scenario 3. Trainable FastText Embedding and All Words in Corpus

Experiments 7, 8, and 9 use Trainable FastText Embedding approach to produce an embedding matrix. The seventh experiment using the adam optimizer produces the best model with an accuracy of 88.67% and a loss of 0.2862 and can predict 6619 data correctly. The eighth experiment using the adamax optimizer produces the best model with an accuracy of 87.33% and a loss of 0.3285 and can predict 6397 data correctly. While the ninth experiment with the Nadam optimizer produced an accuracy of 86.89% and a loss of 0.3612 and was able to predict 6314 data correctly.

Scenario 4. Random Embedding and Corpus without Words That Appear Only Once

Experiments 10, 11, and 12 use the Random Embedding approach to produce an embedding matrix. The 10th experiment using the adam optimizer produces the best model with an accuracy of 87.48% and a loss of 0.3473 and can predict 6371 data correctly. The 11th experiment using the adamax optimizer produces the best model with an accuracy of 88.06% and a loss of 0.3271 and can predict 6544 data correctly. While the 12th experiment using the Nadam optimizer produced an accuracy of 86.42% and a loss of 0.3643 and was able to predict 6175 data correctly.

Scenario 5. Non-Trainable FastText Embedding and Corpus without Words That Appear Only Once

Experiments 13, 14, and 15 use Non-Trainable FastText Embedding approach to produce an embedding matrix. The 13th experiment using the adam optimizer produces the best model with an accuracy of 87.93% and a loss of 0.3438 and can predict 5609 data correctly. The 13th experiment using the adamax optimizer produces the best model with an accuracy of 86.59% and a loss of 0.3470 and can predict 6015 data correctly.

While the 15th experiment with the Nadam optimizer produced an accuracy of 87.18% and a loss of 0.3288 and was able to predict 5187 data correctly.

Scenario 6. Trainable FastText Embedding and Corpus without Words That Appear Only Once

Experiments 16, 17, and 18 use Trainable FastText Embedding approach to produce an embedding matrix. The seventh experiment using the adam optimizer produces the best model with an accuracy of 87.33% and a loss of 0.3521 and can predict 6611 data correctly. The eighth experiment using the adamax optimizer produces the best model with an accuracy of 88.52% and a loss of 0.3273 and can predict 6333 data correctly. While the ninth experiment with the Nadam optimizer produced an accuracy of 87.33% and a loss of 0.3374 and was able to predict 6439 data correctly.

First Best Model

Based on the results of all scenarios, the model chosen as the first best model is the model produced by the seventh experiment, because it has the best predictive ability when compared to other models, that is an accuracy of 88.67% and a loss of 0.2862 and can predict 6619 data correctly. The details of a lot of data in the Fabricated Content / Imposter Content class that was incorrectly predicted is 233 data, of which 3 data were valid data. The amount of data in the Misleading Content/False Context/Manipulated Content class that was incorrectly predicted was 269 data, of which 51 data were valid data. And the amount of data in the Valid class that was incorrectly predicted was 194 data. The prediction results of the first best models can be seen in Table 1.

Table 1. The prediction results of the first best models

Word Embedding Model	Optimizer	All Words in Corpus				Without Word that Appear Only Once			
		True	False-0	False-1	False-2	True	False-0	False-1	False-2
Random Embedding	Adam	6006	70	1036	203	6371	131	582	231
	Adamax	6132	136	946	101	6544	278	332	161
	Nadam	6029	77	1068	141	6175	118	928	94
Non-Trainable FastText Embedding	Adam	6089	226	952	48	5609	249	1430	27
	Adamax	6034	278	953	50	6015	144	1068	88
	Nadam	6236	274	549	256	5187	177	1814	137
Trainable FastText Embedding	Adam	6619	233	269	194	6611	123	495	86
	Adamax	6397	187	654	77	6333	220	620	142
	Nadam	6314	86	758	157	6439	143	562	171

Column “True” contains the number of correctly predictable data from the associated word embedding model and optimizer. Column “False-0” contains the amount of data from Fabricated Content/Imposter Content classes correctly predicted from the associated word embedding model and optimizer. Column “False-1” contains the amount of data from Misleading Content/False Context/ Manipulated Content classes correctly predicted from the associated word embedding model and optimizer. And column “False-2” contains the amount of data from the Valid class correctly predicted from the associated word embedding model and optimizer.

3.2 Model Performance After Usability Testing Implementation

Model improvement uses several parameters that have been selected and based on the results of the analysis of parameter combinations, that are:

1) The four kernel sizes used are 2, 4, 7, and 8.
2) The four activation functions used are tanh, relu, elu, and selu.
3) The two optimizers used are adam and adamax
4) Many words that will be used are all words in the corpus.
5) The word embedding approach used is Trainable FastText Embedding.

Based on the results of usability testing obtained from 22 google form responses and comments from 95 users in the selected Facebook group, it was concluded that the shortcomings of the application are on the user experience side. Most users suggest that the application provides instructions to the user regarding how to operate the application on first time the application is mounted. Meanwhile, the weakness of the initial model used is the lack of accuracy when predicting valid news narratives, especially for narratives that report about health.

`

Based on the results of the bug search and journal research that was carried out afterwards, a hypothesis was obtained that the distribution of narrative topics in the valid news dataset used needed to be adjusted to the distribution of hoax narrative topics [42]. Therefore, data collection was carried out again with a different approach than before. Valid data collection this time uses narration from articles on news.detik.com which are not part of articles on 20.detik.com, because scraping on the previous page does not allow scraping based on topic. The scraping process also considers the topic of each article so that it is balanced with the distribution of topics used in hoax narratives. This re-scraping produces a dataset with a total of 9624 rows of data. In addition, valid news scraping was also carried out on several other media, namely tempo.co, medcom.id, and cnnindonesia.com as test datasets to ensure that the models created could predict valid news outside the sources used for the training dataset. Comparison of many data used for training now also uses a ratio of 1 : 2 : 3. With this ratio, a 1 : 1 ratio is formed between the entire hoax narrative and valid narrative, to ensure that the distribution of narrative topics used in training between hoax narratives and valid narratives is roughly equal. And because a lot of the data used from each class is no longer balanced, the result is that for model filtration it is no longer relevant to use recall values. Instead, the f1-score value is used for filtration because it is more suitable for knowing the performance of models with unbalanced data.

After training for making the final model, the best model was obtained which had an accuracy of 92.98% and a loss of 0.191. After being tested on the entire training dataset, the model can predict as many as 9245 of the 9649 data correctly. With the details of a lot of data in the Fabricated Content / Imposter Content class that is incorrectly predicted, there are 212 data, of which 5 data are valid data. The amount of data in the Misleading Content/False Context/Manipulated Content class that was incorrectly predicted was 180 data, of which 63 data were valid data. And the amount of data in the Valid class that is incorrectly predicted is as much as 12 data. Furthermore, after the model was re-tested with the Valid dataset that was created previously, it was found that the number of valid data that was incorrectly predicted by the model was 202 out of a total of 4428 data or around 4.562%.

3.3 Deep Learning Model Implementation on Website

The implementation of the model on the resulting website can be accessed via https://unifact.me where the screenshot can be seen in Figure 5. The site has been updated with a model from the improvement stage. With a positive response from society based on the results of usability testing, using the model with a website has the following advantages.

1) Based on the aspect of prediction speed, model prediction via website has a speed that is roughly equal as prediction speed via interpreter, supported by Heroku server access speed with appropriate dyno resource selection and user signal quality. Overall, the prediction speed is faster because the user does not need to manually import the model and run other code that is outside the route function.
2) Based on the aspect of ease of use, users do not need to install any application to predict, just type in the narration they want to predict and press the enter button to get predictions. This causes ordinary users who do not understand programming at all to also use the model easily. There is also a "lamp" button which contains a guide for users to understand the use of the application. In addition, website is also equipped with speech-to-text and optical character recognition features, so that users can make predictions without the need to type if something is constrained.
3) Based on the aspect of the completeness of the prediction results, users can get many of the results presented, starting from the typed narration, the probability of the predicted results in each class, to an explanation that explains the explanation of the predicted results based on the data used in the corpus. This explanation is far more complete than the general model prediction results which are just a numpy array containing prediction probabilities.
4) Based on the aspect of portability, website can of course be accessed anytime, from anywhere, and with any device. This is because the website produced in this study has also been designed to be responsive on all devices.

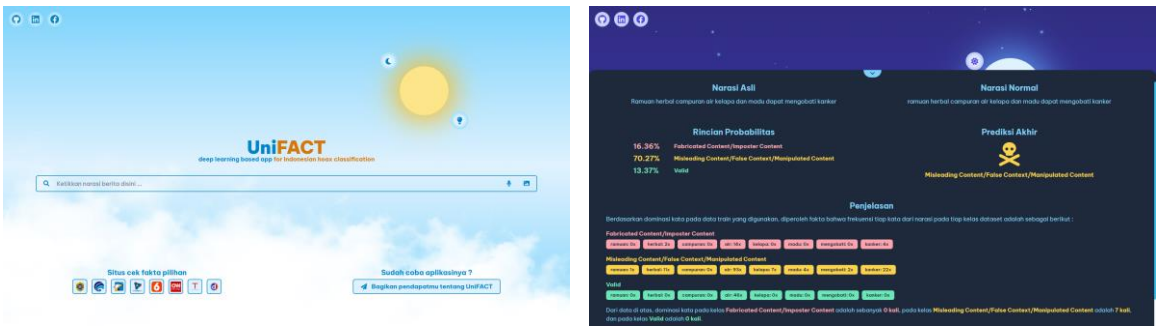

Figure 5. Screenshoot of the website

4. CONCLUSION

Based on the research that has been done, the results of training using a deep learning model with Bidirectional Temporal Convolutional Network architecture that created after implementing usability testing can obtain an accuracy of up to 92.98% and a loss of 0.191. The results of testing the model on all data in the dataset stated that the model could predict as many as 9245 of the 9649 data correctly. With the details of a lot of data in the Fabricated Content / Imposter Content class that is incorrectly predicted, there are 212 data, of which 5 data are valid data. The amount of data in the Misleading Content/False Context/Manipulated Content class that was incorrectly predicted was 180 data, of which 63 data were valid data. And the amount of data in the Valid class that is incorrectly predicted is as much as 12 data. The prediction results of the model on a valid dataset containing 4428 data where the data comes from 3 different mass media sources also states that the model only predicts incorrectly as many as 202 data or in other words only about 4.562% of the data is predicted incorrectly. Thus, it can be concluded that deep learning model with Bidirectional Temporal Convolutional Network architecture have good performance in classifying Indonesian narratives.

Meanwhile, based on the results of deploying backend applications on Heroku, backend applications with deep learning model can be implemented properly on website and the resulting website can be accessed via https://unifact.me. However, with a slug size that is large enough for applications with deep learning model, this causes more wasteful disk usage. Meanwhile, based on a comparison of the methods and results of using the model via website with the use of the model manually through an interpreter, the use of the model via website has advantages in terms of speed of prediction, ease of use, completeness of prediction results, and portability.

REFERENCES

[1] A. Yuliani, "Ada 800.000 Situs Penyebar Hoax di Indonesia," *Kominfo RI*, 2017. https://www.kominfo.go.id/content/detail/12008/ada-800000-situs-penyebar-hoax-di-indonesia/0/sorotan_media (accessed May 09, 2021).

[2] Badan Penelitian dan Pengembangan, "Riset: 44 Persen Orang Indonesia Belum Bisa Mendeteksi Berita Hoax," *Kemendagri RI*, 2018. https://litbang.kemendagri.go.id/website/riset-44-persen-orang-indonesia-belum-bisa-mendeteksi-berita-hoax-2/ (accessed Dec. 28, 2022).

[3] C. Wardle, "Fake news. It's complicated.," *First Draft*, 2017. https://firstdraftnews.org/articles/fake-news-complicated/ (accessed May 09, 2021).

[4] C. Khontoro, J. Andjarwirawan, and Yulia, "Penerapan Algoritma TextRank dan Dice Similarity Untuk Verifikasi Berita Hoax," *Jurnal Infra*, vol. 9, no. 1, pp. 98–102, 2021.

[5] J. P. Haumahu, S. D. H. Permana, and Y. Yaddarabullah, "Fake news classification for Indonesian news using Extreme Gradient Boosting (XGBoost)," *IOP Conf Ser Mater Sci Eng*, vol. 1098, no. 5, p. 052081, Mar. 2021, doi: 10.1088/1757-899x/1098/5/052081.

[6] B. P. Nayoga, R. Adipradana, R. Suryadi, and D. Suhartono, "Hoax Analyzer for Indonesian News Using Deep Learning Models," in *Procedia Computer Science*, 2021, vol. 179, pp. 704–712. doi: 10.1016/j.procs.2021.01.059.

[7] T. Trisna Astono Putri, H. S. Warra, I. Yanti Sitepu, and M. Sihombing, "Analysis and Detection of Hoax Contents in Indonesian News Based on Machine Learning," *Journal Of Informatics Pelita Nusantara*, vol. 4, no. 1, 2019.

[8] H. Mustofa and A. A. Mahfudh, "Klasifikasi Berita Hoax Dengan Menggunakan Metode Naive Bayes," *Walisongo Journal of Information Technology*, vol. 1, no. 1, p. 1, Nov. 2019, doi: 10.21580/wjit.2019.1.1.3915.

[9] I. Y. R. Pratiwi, R. A. Asmara, and F. Rahutomo, "Study of Hoax News Detection using Naïve Bayes Classifier in Indonesian Language," in *Proceedings of the 11th International Conference on Information and Communication Technology and System, ICTS 2017*, Jan. 2018, vol. 2018-January, pp. 73–78. doi: 10.1109/ICTS.2017.8265649.

[10] E. Zuliarso, M. T. Anwar, K. Hadiono, and I. Chasanah, "Detecting Hoaxes in Indonesian News Using TF/TDM and K Nearest Neighbor," in *IOP Conference Series: Materials Science and Engineering*, May 2020, vol. 835, no. 1. doi: 10.1088/1757-899X/835/1/012036.

[11] A. B. Prasetijo, R. R. Isnanto, D. Eridani, Y. A. A. Soetrisno, M. Arfan, and A. Sofwan, "Hoax Detection System on Indonesian News Sites Based on Text Classification using SVM and SGD," in

`

International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Oct. 2017, pp. 45–49.

[12] A. A. Kurniawan and M. Mustikasari, "Implementasi Deep Learning Menggunakan Metode CNN dan LSTM untuk Menentukan Berita Palsu dalam Bahasa Indonesia," *Jurnal Informatika Universitas Pamulang*, vol. 5, no. 4, pp. 544–552, Oct. 2020, doi: 10.32493/informatika.v5i4.7760.

[13] R. Adipradana, B. P. Nayoga, R. Suryadi, and D. Suhartono, "Hoax Analyzer for Indonesian News using RNNs with Fasttext and Glove Embeddings," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2130–2136, Aug. 2021, doi: 10.11591/eei.v10i4.2956.

[14] A. Apriliyanto and R. Kusumaningrum, "Hoax Detection in Indonesian Language using Long Short-Term Memory Model," *Sinergi*, vol. 24, no. 3, pp. 189–196, Jul. 2020, doi: 10.22441/sinergi.2020.3.003.

[15] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.01271

[16] L. Nanni, A. Lumini, A. Manfè, R. Rampon, S. Brahnam, and G. Venturin, "Gated Recurrent Units and Temporal Convolutional Network for Multilabel Classification," 2021.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/

[18] C. Zhong, L. Jiang, Y. Liang, H. Sun, and C. Ma, "Temporal Multiple-convolutional Network for Commodity Classification of Online Retail Platform Data," in *ACM International Conference Proceeding Series*, Feb. 2020, pp. 236–241. doi: 10.1145/3383972.3383989.

[19] Y. Liang, J. Kang, Z. Yu, B. Guo, X. Zheng, and S. He, "Leverage Temporal Convolutional Network for theRepresentation Learning of URLs," in *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*, Jul. 2019, pp. 74–79.

[20] J. Sun, X. Luo, H. Gao, W. Wang, Y. Gao, and X. Yang, "Categorizing Malware via A Word2Vec-based Temporal Convolutional Network Scheme," *Journal of Cloud Computing*, vol. 9, no. 1, Dec. 2020, doi: 10.1186/s13677-020-00200-y.

[21] Y. Zuo *et al.*, "Short Text Classification Based on Bidirectional TCN and Attention Mechanism," in *Journal of Physics: Conference Series*, Dec. 2020, vol. 1693, no. 1. doi: 10.1088/1742-6596/1693/1/012067.

[22] B. Jang, M. Kim, G. Harerimana, S. U. Kang, and J. W. Kim, "Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism," *Applied Sciences (Switzerland)*, vol. 10, no. 17, pp. 5841–5854, Sep. 2020, doi: 10.3390/app10175841.

[23] M. Aziz Muslim, Y. Dasril, and T. Mustaqim, "Accuracy of Malaysia Public Response to Economic Factors During the Covid-19 Pandemic Using Vader and Random Forest," *Journal of Information System Exploration and Research*, vol. 01, no. 01, pp. 49–70, 2023, doi: 10.00000/joiser.0000.00.00.000.

[24] K. He, Y. Yan, and W. Xu, "Learning to Tag OOV Tokens by Integrating Contextual Representation and Background Knowledge," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Jul. 2020, pp. 619–624.

[25] M. Heidarysafa, K. Kowsari, D. E. Brown, K. J. Meimandi, and L. E. Barnes, "An Improvement of Data Classification using Random Multimodel Deep Learning (RMDL)," *Int J Mach Learn Comput*, vol. 8, no. 4, pp. 298–310, Aug. 2018, doi: 10.18178/ijmlc.2018.8.4.703.

[26] O. Calin, *Deep Learning Architectures*. Cham: Springer Nature Switzerland AG, 2020. doi: 10.1007/978-3-030-36721-3.

[27] A. D. Lestari, N. A. Syarifudin, and Y. J. Nurriski, "Application of pest detection on vegetable crops using the cnn algorithm as a smart farm innovation to realize food security in the 4.0 era," *Journal of Soft Computing Exploration*, vol. 3, no. 2, Sep. 2022, doi: 10.52465/joscex.v3i2.72.

[28] H. Aji Prihanditya and N. Hestu Aji Prihanditya, "The Implementation of Z-Score Normalization and Boosting Techniques to Increase Accuracy of C4.5 Algorithm in Diagnosing Chronic Kidney Disease," *Journal of Soft Computing Exploration*, vol. 1, no. 1, pp. 63–69, 2020.

[29] H. El-Amir and M. Hamdy, *Deep Learning Pipeline*. New York: Apress Media LLC, 2020. doi: 10.1007/978-1-4842-5349-6.

[30] B. Xiao, Y. Liu, and B. Xiao, "Accurate State-of-charge Estimation Approach for Lithium-ion Batteries by Gated Recurrent Unit with Ensemble Optimizer," *IEEE Access*, vol. 7, pp. 54192–54202, 2019, doi: 10.1109/ACCESS.2019.2913078.

[31] A. Vieira and B. Ribeiro, *Introduction to Deep Learning Business Applications for Developers: From Conversational Bots in Customer Service to Medical Image Processing*. New York: Apress Media LLC, 2018. doi: 10.1007/978-1-4842-3453-2.

[32] C. Zong, R. Xia, and J. Zhang, *Text Data Mining*. Singapore: Tsinghua University Press, 2021. doi: 10.1007/978-981-16-0100-2.
[33] S. Yamaguchi *et al.*, "Web Services for Collaboration Analysis with IoT Badges," *IEEE Access*, vol. 10, pp. 121318–121328, 2022, doi: 10.1109/ACCESS.2022.3222562.
[34] A. M. Potdar, D. G. Narayan, S. Kengond, and M. M. Mulla, "Performance Evaluation of Docker Container and Virtual Machine," *Procedia Comput Sci*, vol. 171, pp. 1419–1428, 2020, doi: 10.1016/j.procs.2020.04.152.
[35] Heroku, "Deploying with Docker | Heroku Dev Center." https://devcenter.heroku.com/categories/deploying-with-docker (accessed Jan. 18, 2023).
[36] N. Li and B. Zhang, "The Research on Single Page Application Front-end Development Based on Vue," *J Phys Conf Ser*, vol. 1883, no. 1, Apr. 2021, doi: 10.1088/1742-6596/1883/1/012030.
[37] A. Safonyk, M. Mishchanchuk, V. I. Lytvynenko, and V. Lytvynenko, "Intelligent Information System for The Determination of Iron in Coagulantsbased on a Neural Network," in *2nd International Workshop on Intelligent Information Technologies and Systems of Information Security (IntelITSIS)*, Mar. 2021, pp. 142–150. [Online]. Available: https://www.researchgate.net/publication/351844460
[38] Y. Quan, "Design and Implementation of E-commerce Platform based on Vue.js and MySQL," *3rd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2019)*, vol. 90, pp. 449–454, 2019.
[39] Y. Bao, C. Zhang, and Q. Shi, "Mining and Analysis Based on Big Data in Public Transportation," *11th International Symposium on Intelligence Computation and Applications (ISICA 2019)*, vol. 1205, pp. 681–688, 2020, doi: 10.1007/978-981-15-5577-0.
[40] E. Wohlgethan, "Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React, and Vuejs," PhD Thesis, Hamburg University of Applied Sciences, Hamburg, 2018.
[41] I. Maramba, A. Chatterjee, and C. Newman, "Methods of Usability Testing in The Development of eHealth Applications: A Scoping Review," *Int J Med Inform*, vol. 126, pp. 95–104, Jun. 2019, doi: 10.1016/j.ijmedinf.2019.03.018.
[42] C. Juditha and J. J. Darmawan, "Infodemik Di Masa Pandemi: Analisis Peta Hoaks Covid-19 Tahun 2020," *Pekommas*, vol. 6, no. The, pp. 66–67, 2020, doi: 10.30818/jpkm.2021.2060307.

`