Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 8 No. 4 (2024) 525 - 534 e-ISSN: 2580-0760

Optimized Hyperparameter Tuning for Improved Hate Speech Detection with Multilayer Perceptron

Muhamad Ridwan[1*], Ema Utami[2]
[1, 2]Magister Informatika, Universitas Amikom Yogyakarta, Yogyakarta, Indonesia
[1]rdwanmuhamad@students.amikom.ac.id, [2]ema.u@amikom.ac.id

Abstract

Hate speech classification is a critical task in the domain of natural language processing, aiming to mitigate the negative impacts of harmful content on digital platforms. This study explores the application of a Multilayer Perceptron (MLP) model for hate speech classification, utilizing Bag of Words (BoW) for feature extraction. The hypothesis posits that hyperparameter tuning through sophisticated optimization techniques will significantly improve model performance. To validate this hypothesis, we employed two distinct hyperparameter tuning approaches: Random Search and Optuna. Random Search provides a straightforward yet effective means of exploring the hyperparameter space, while Optuna offers a more sophisticated, optimization-based approach to hyperparameter selection. The study involved training the MLP model on a labeled dataset based on crawling results on the Twitter platform of hate speech and non-hate speech overall total dataset is 13.169, followed by evaluation using standard metrics. Our experimental results demonstrate the comparative effectiveness of these two hyperparameter tuning methods. Notably, the MLP model tuned with Optuna achieved a higher F1-score of 81.49%, compared to 79.70% achieved with Random Search, indicating the superior performance of Optuna in optimizing the hyperparameters. These results were obtained through extensive cross-validation to ensure robustness and generalizability. The findings underscore the importance of optimized hyperparameters in developing robust hate speech classification systems. The superior performance of Optuna highlights its potential for broader application in other machine learning tasks requiring hyperparameter optimization. This improvement enables more reliable and efficient automated moderation, which is crucial for the integrity and security of digital communication platforms such as Twitter.

Keywords: *hate speech; multilayer perceptro; bag of words; hyperparameter tuning; random search; optuna*

How to Cite: Muhamad Ridwan and Ema Utami, "An Optimized Hyperparameter Tuning for Improved Hate Speech Detection with Multilayer Perceptron", *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 8, no. 4, pp. 525 - 534, Aug. 2024.
DOI: https://doi.org/10.29207/resti.v8i4.5949

1. Introduction

The digital age has revolutionized communication, transforming how we interact by rapidly exchanging information and ideas. However, alongside these advancements, the spread of hate speech defined as any expression intended to insult or degrade individuals based on characteristics such as religious affiliation, sexual orientation, gender, or ethnicity has intensified, posing serious social challenges and consequences [1].

The rise of social media platforms like Twitter and Facebook has led to an exponential increase in user-generated content, facilitating both constructive discourse and the rapid dissemination of harmful content, including hate speech [2]. As more people turn to these platforms for news and discussion, the visibility of hate speech has grown, sometimes leading to conflicts among social groups [3], [4]. Specifically, Twitter, which has 63.6% of Indonesians aged 16 to 64 as active users, has become a focal point for public discourse and the spread of offensive language, making it an important platform for studying public opinion and conducting sentiment analysis [5], [6].

Despite the prevalence of offensive language on social media, the sheer volume of content makes manual detection inefficient. To address this, research has increasingly focused on automating the identification of hate speech through machine learning and deep learning techniques [7], [8]. Among the various methods available, Multilayer Perceptron (MLP) has been selected for this study due to its proven ability to model complex, non-linear relationships in data, making it well-suited for the nuanced task of hate speech detection. MLP's flexibility in learning representations

Received: 25-07-2024 | Accepted: 22-08-2024 | Published Online: 24-08-2024

from data, combined with hyperparameter tuning, allows for optimizing model performance, making it a compelling choice over more traditional approaches. Hyperparameter tuning, in particular, is crucial in this context as it ensures that the model is not only accurate but also robust, and capable of generalizing well to new, unseen data [9].

This study aims to improve hate speech detection using MLP, Bag of Words feature extraction, and Hyperparameter Tuning with Random Search and Optuna approaches. This study attempts to build a machine learning model through algorithm selection and optimization, which ultimately contribute to broader efforts to automate hate speech detection.

Research [10], [11] and [12] focused on detecting abusive language and hate speech in Indonesia on the Twitter platform, using different approaches. The research by [10] has successfully created a dataset based on crawling results on Twitter and used a combination of unigram word features, Random Forest the Decision Tree (RFDT), and Label Power-set (LP). This experiment showed an accuracy rate of 77.36% for multi-label classification without target, category, and hate speech level identification, and 66.12% when including these identifications. On the other hand, research [11] used a multi-label classification One-vs-All method with an Artificial Neural Network (ANN) classifier and a Bag of Words (BoW) approach, achieving the highest accuracy of 86.79%. This study emphasized the importance of text preprocessing steps, such as the elimination of non-formal words and the application of non-formal stemming, as well as balancing the number of tweets for each label to improve results. A combination of alternative feature selection and the use of deep learning models is also suggested for optimal classification performance. Research [12] evaluated the hate speech classification model using a Support Vector Machine (SVM) with linear and polynomial kernels, and DistilBERT feature extraction combined with PCA dimensionality reduction. The evaluation results showed that the polynomial kernel with 50 dimensions provided the highest average F1 Score of 78%. This study recommended focusing on the word cleaning methods in the preprocessing stage to improve analysis accuracy and consider dimensionality reduction methods in determining component values. In addition, the classification using DistilBERT with SVM can be optimized by extracting TF/IDF, Bag of Words, and Tweet Length features

By applying different approaches, the research conducted by [13] and [14] used text classification with Bag of Words (BoW) feature extraction to address different issues, namely hate speech and suicide detection. The study [13] compared classification methods such as Decision Tree and Stochastic Gradient Boosting, using prominent feature extraction TF-IDF, Bag of Words, and Tweet Length to identify hate speech during COVID-19. The final results showed that Stochastic Gradient Boosting achieved an F1-Score of 98%, higher than the Decision Tree which achieved 97%. This research revealed that the classification that considers gender categories can enhance the generated information. Meanwhile, recent studies in suicide detection and prevention conducted by [14] showed a significant increase in the use of semi-supervised methods to populate the Life Corpus with bootstrapping techniques. There are two classifiers used namely Support Vector Machine (SVM) with Bag of Words (BoW) feature extraction and without TF-IDF. These approaches were applied to classify texts from social networks and forums related to suicide and depression. With five different data collections: Life, Reddit, Life+Reddit, Life_en, and Life_en+Reddit, the research results showed that the semi-supervised method increased the Life Corpus size from 102 to 273 samples with a macro F1 score of 0.80 in the combination of Life+Reddit+BoW Embeddings using SVM. Subsequently, the manual evaluation by annotators showed a Cohen's Kappa agreement level of 0.86. Although the semi-supervised method made significant contributions, there are limitations in recognizing certain contexts or nuances in the text that can affect detection accuracy. These findings contribute significantly to further development in hate speech detection and suicide prevention through text classification and feature extraction Bag of Word.

Additionally, research [15] - [18] proposed the use of deep learning models, particularly Multilayer Perceptron (MLP), for text classification and sentiment analysis with different yet complementary approaches. Research [15], [16], [17] used a combination of BiLSTM, CNN, and MLP models with word embedding techniques such as GloVe, TF-IDF, and transformer-based embeddings, achieving accuracies of over 95% in hate speech classification in English and Spanish. Research [15], [16], [17] also proposed MLP for sentiment analysis on two datasets, namely Twitter and movie reviews, with accuracies of 85% and 89%, respectively. Sentiment analysis is important for understanding public opinion from texts like messages or posts. Research [15], [16], [17] emphasized the importance of Explainable Artificial Intelligence (XAI) in hate speech detection, using BERT+ANN and BERT+MLP models with accuracies of 93.55% and 93.67% on the HateXplain dataset, and applying LIME interpretation methods. However, this method was less effective for all users due to insufficient depth in interpretation. Research [18] combined lexicon-based and machine-learning approaches to predict public market mood in BIST30, Borsa Istanbul, using 17.189 tweets. By excluding neutral labels, Support Vector Machine and MLP showed the best performance with accuracies of 0.89 and 0.88. All these studies indicated that MLP can achieve high accuracy in text classification, with potential improvements through parameter optimization and better preprocessing steps.

In another place, research [9] and [19] focused on hyperparameter optimization and imbalanced handling data in Multi-Layer Perceptron Neural Network (MLPNN) models. Research [19] proposed a new model to optimize the number of neurons in hidden layers, neuron connections, and connection weights to avoid overfitting or underfitting, enhancing the generalization capacity of MLPNN. Although methods like Random search and Grid Search take longer, both can yield good accuracy. Meanwhile, research [20] addressed the issue of imbalanced data in the context of fraud detection, achieving a test accuracy of 99.937% with the MLP model. This study explored resampling strategies such as under-sampling, oversampling, and SMOTE to reduce the number of false negatives. The results highlighted the importance of addressing class imbalance in the data preprocessing phase to improve the performance of predictive models.

2. Research Methods

This section focuses on the architectural description of the proposed system. The development of a hate speech text classification system is driven by the urgent need to address the rampant spread of hate speech across various digital platforms. This phenomenon does not only threaten social security and order but also disrupts peace and comfort in online interactions. Therefore, an effective solution is needed to identify and classify hate speech accurately.

The proposed system is built using a multilayer perceptron (MLP) approach, a type of artificial neural network known for its ability to process text data. Moreover, the hyperparameter tuning using two methods: Random Search and Optuna, is applied to enhance the performance and accuracy of the classification. Hyperparameter tuning is a crucial step that determines the best configuration for a machine learning model, thereby maximizing its performance.

The steps required to build this text classifier involve several stages which are started from the collection and preparation of hate speech text data, feature extraction using the Bag of Words (BoW) technique, and the construction and training of the MLP model. The BoW feature extraction is used to convert the text into a numerical representation that can be processed by the MLP model. Subsequently, the model is trained with the prepared dataset, and hyperparameter tuning is conducted to find the optimal parameters that enhance the classification performance.

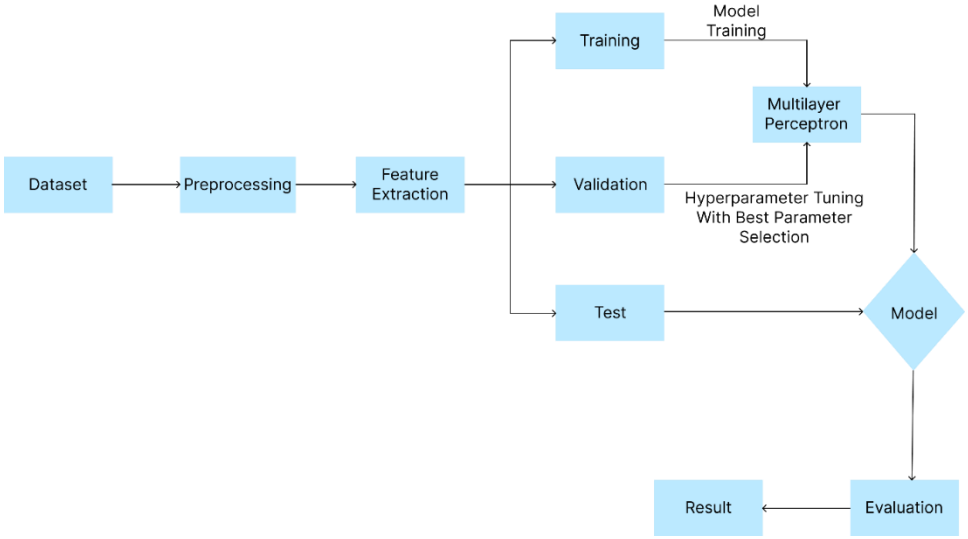

Figure 1. Research Flow

Figure 1 illustrates the research flow process involving the use of data preprocessing techniques, feature extraction, cross-validation, and machine learning, along with hyperparameter tuning to find optimal parameters.

2.1 Dataset

This research utilizes a dataset from previous research conducted by [10] which is accessible through GitHub. This reference dataset uses multilabel information to identify language containing insulting content and words that can trigger hateful feelings. The dataset contains the following columns: Religion/creed, Race/ethnicity, Physical/disability, Gender/sexual orientation, and other invective/slander. In Figure 2, the graph displays the number of Tweets containing hate and non-hate speech based on the HS label. The 0 number is used to indicate that the Tweet data does not contain hate or non-hate speech, with a total of 7.608. Then, the second bar with the value 1 is used for Tweet data containing hate or HS, with a total of 5.561, and the overall total of both is 13.169.

The hate speech tweets totalling 5.561 were mostly directed at individuals. In detail, the 3.575 tweets were specifically aimed at individuals, and 1.986 tweets were targeted to certain groups. The identified hate speech categories included 793 tweets related to religion/creed, 566 tweets related to race/ethnicity, 323 tweets related to physical/disability, 306 tweets related to

gender/sexual orientation, and 3.740 tweets related to other invective/slander. To assess the severity of hate speech, the data also included 3.383 cases of weak hate speech, 1.705 cases of moderate hate speech, and 473 cases of hate speech classified as strong.

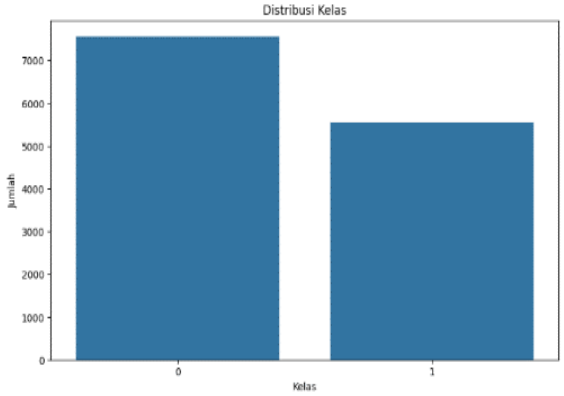

Figure 2. Class Distribution

2.2 Preprocessing

In the initial stage of the data preprocessing, this step significantly impacts the model's performance. The data sources such as Twitter or other online platforms are often filled with noise and may contain incomplete information, such as text, images, audio, video, and so on. This initial processing aims to clean the data, eliminate noise, and make it clear and meaningful [17]. The preprocessing steps include lowercase conversion, tweet attributes removal, non-alphanumeric characters removal, spell checking, stemming, and stop word removal. The following table is an example of a dataset that has undergone preprocessing at each step, shown in Table 1.

Table 1. Result Preprocessing

Tweet	Tweet_Clean
- disaat semua cowok berusaha melacak perhatian	cowok usaha lacak perhati lantas remeh perhati
RT USER: USER siapa yang telat ngasih tau elu?	telat tau edan sarap gaulcigax jifla cal licew
41. Kadang aku berfikir, kenapa aku tetap percaya	41 kadang pikir percaya tuhan jatuh kali kali
USER USER AKU ITU AKU\n\nKU TAU MATAMU SIPIT TAPI	ku tau mata sipit lihat
USER USER Kaum cebong kapir udah keliatan dongoknya	kaum cebong kafir lihat dongok dungu haha
…	…
USER jangan asal ngomong ndasmu. congor lu yg	bicara ndasmu congorsekata anjing
USER Kasur mana ena kunyuk'	kasur enak kunyuk
USER Hati hati bisu :(.g\n\nlagi bosan huft \	hati hati bisu bosan duh
USER USER USER USER Bom yang real mudah terdeteksi	sangat mudah _ deteksi lahir kubur dahsyat ledak
USER Mana situ ngasih(": itu Cuma foto ya kutil onta'	itu foto kutil di atas

The transformation of the text preprocessing stage is illustrated through word clouds. These changes can be observed by comparing tweet data before and after the preprocessing process. Figure 3 shows the visualization of the "Tweet" column containing data before the preprocessing process:

Figure 3. Wordcloud before initial processing

Figure 3 shows the frequency of words that often appear in the "Tweet" column. In that figure, some words in the tweet dataset contain meaningless phrases, such as (USER, URL, X9F, X98, XF0, etc), which can affect the analysis results.

Figure 4. Wordcloud after preprocessing

Next, Figure 4 shows the frequency of words that often appear in the "Tweet_Clean" column. This Figure provides an overview of the changes resulting from the preprocessing process. The comparison between Figures 3 and 4 allows an understanding of the overall changes after the preprocessing stage. The analysis expressed that there were several words like "WKWK," "HAHA," "EH," and other meaningless words, which indicate the need for improvements in the preprocessing stage in this study. The selection and creation of a "kamus alay" could be one factor that causes some meaningless words to be inadequately processed during preprocessing.

2.3 Feature Extraction

The feature extraction method applied in this research was Bag of Words. The training data in text form will be extracted into vectors. The resulting vectors can further be used as the implementation features in natural language processing and machine learning cases. Computers cannot directly understand text because it does not contain numerical information [20]. Thus, a method is needed to connect the computer with the text using a text-to-number converter.

CountVectorizer implementation bag-of-words approach, as previously mentioned, calculated the frequency of each feature and determined the importance of each word in the document. Table 2 shows the results of CountVectorizer.

Table 2. Converts Text Into Vector

CountVectorizer	Result
(0, 2427)	2
(0, 12073)	1
(0, 6306)	1
(0, 8737)	2
(0, 6386)	1
(0, 9559)	1
(0, 8737)	2
(0, 5493)	1
(0, 5843)	1
(0, 1353)	1

In research, hate speech detection using both deep and shallow learning methods and the Bag of Words (BoW) model is employed as a basic technique for text representation [21]. Bag of Word is commonly used in Natural Language Processing (NLP) to run tasks such as sentiment analysis, text classification, and information retrieval. This approach offers a straightforward and efficient means of converting textual data into structured format

2.4 Cross Validation

At this stage, the data would be divided into two parts, namely the training data and the test data. A portion of the data, called the training sample, is used for training, while the remaining portion, called the validation sample, is used for testing. The data split is illustrated in Figure 5.

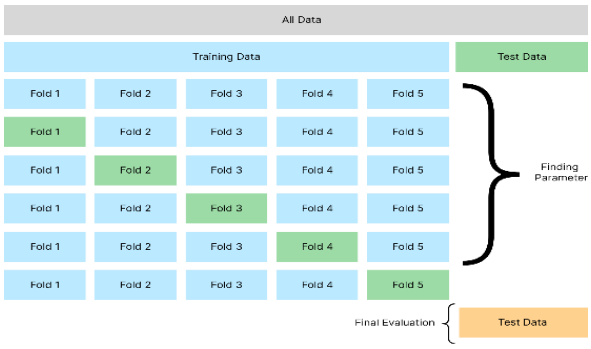

Figure 5. Cross Validation illustration

This research employed cross-validation, a common method for data partitioning in model selection. In this technique, the data was divided into k segments (known as k-folds). One segment serves as the validation set. The trained model was then tested on this validation set and its predictive performance was recorded. This process was repeated k times so that each data segment served as the validation set once [22].

After the preprocessing and data splitting process was completed, the data was trained using machine learning algorithms to build the model. This section provides a brief explanation of the chosen machine learning algorithms and hyperparameter tuning performed.

2.5 Multilayer Perceptron

This research employed a Multilayer Perceptron (MLP) as the learning module. MLP is a type of artificial neural network that consists of multiple layers of neurons, where each neuron in one layer is connected to neurons in the next layer.

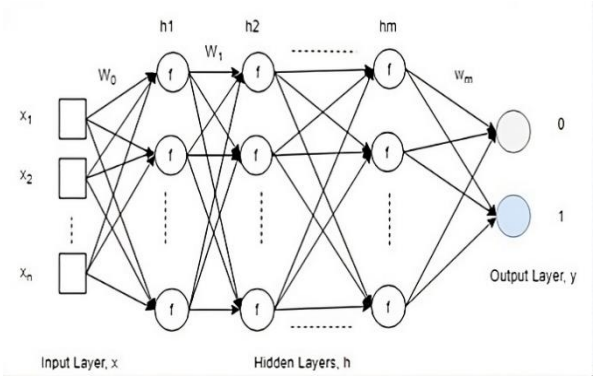

Figure 6. Multilayer Perceptron Architecture

Figure 6 illustrates the structure of a Multilayer Perceptron (MLP). MLP is known for its high classification capability and it consists of three types of layers that are input layer, hidden layers, and output layer [23]. In the input layer, each neuron represents an input variable or feature. The hidden layers are responsible for processing data and storing weights during training. Furthermore, the output layer consists of neurons that represent the output variables. The number of neurons in the input layer is equal to the number of features provided to the neural network, while the number of neurons in the output layer matches the number of classes to be predicted. The number of neurons in the hidden layers is a crucial architectural decision, with the primary goal of optimizing it with appropriate parameters to generalize well in the task of aggression detection classification. To minimize the difference between the desired network output and the actual output, MLP learning is based on weight adjustment, commonly using backpropagation techniques based on gradient descent methods.

Suppose we use n neurons as the input layer x as shown in Formula 1.

$$x = (x1, x2 \ldots, xn) \quad (1)$$

Rectified Linear Unit (ReLU) is used as the activation function in the hidden layer where ReLU(x) = max(x, 0) is shown in Formula 2

$$f(x) = \frac{1}{(1+e^{(-x)})} \quad (2)$$

Then the output of each layer is calculated to obtain the final output. Assume the set of hidden layers h = (h1, h2, ..., hm) and the number of neurons in each hidden layer hi is ni. The output of the first hidden layer h1 is calculated using Formula 3.

$$h_i^k = f\left(\sum_{j=1}^{n_i-1} w_{m,k}^0 \, x_j\right) k = 1 \cdots n_i \quad (3)$$

The output of the next hidden layer is calculated using Formula 4.

$$h_i^k = f\left(\sum_{j=1}^{n_{i-1}} w_{j,k}^{i-1} h_{i-1}^j\right) \; i = 2, \cdots N \; dan \; k = 1 \cdots n_i \quad (4)$$

In the equation w i−1 j, k represents the weight between the j-th neuron and (i − 1) in the next hidden layer, ni is the i-th neuron in the i-th hidden layer. The output hi can be calculated as in Formula 5.

$$h_i = (h_i^1, h_i^2, \cdots h_i^{n_i}) \quad (5)$$

The classification stage uses Multilayer Perceptron with the scikit-learn library as follows:

hidden_layer_sizes: array-like of shape (n_layers - 2,), default=(100,): This parameter specifies the architecture of the artificial neural network (MLP) by providing the number of units (neurons) in each hidden layer

activation{'logistic'}: This parameter specifies the activation function used in each unit (neuron) in the hidden layers and the output unit. 'logistic' refers to the sigmoid or logistic activation function, which produces output in the range (0.1).

batch_sizeint,: This parameter specifies the batch size used This parameter specifies the learning rate for the optimization algorithm used in training the model. The learning rate controls how much adjustment is made to the weights and biases of the network each iteration.

max_iterint,: This parameter specifies the maximum number of iterations and epochs performed during model training.

2.6 Hyperparameter Tuning

Hyperparameter is a model to automate the hyperparameter tuning process and achieve [24], is shown in Formula 6.

$$x^* = arg \min_{x \in X} f(x) \quad (6)$$

f(x) is the objective function; x∗ is the hyperparameter configuration that yields the optimal value for f(x); and the hyperparameter x can take any element in the search space X. Hyperparameter optimization automates the adjustment of hyperparameter values to make the process more efficient. Overall, the goal of hyperparameter optimization is to achieve this efficiency [25]. Hyperparameters in the Multilayer Perceptron (MLP) model for classification tasks use two main approaches, namely Random Search and Optuna.

Random Search is a straightforward, simple and easy-to-use technique for hyperparameter optimization in machine learning [26], [27]. This approach involves randomly sampling hyperparameters from a predefined search space, and then evaluating the resulting model on the validation set to assess the performance of each hyperparameter combination [24].

Figure 7 illustrates a strategy where a set of random hyperparameters is used to find the optimal solution for a model [28]. The advantages of random search are its simplicity and ease of implementation. Additionally, random search can often outperform more complex optimization methods in low-dimensional hyperparameter spaces. However, the main drawback of random search is that each evaluation conducted during the search process is independent of one another. Consequently, this approach does not consider the results of previous evaluations when selecting the next set of hyperparameters to be tested. Random search is used to find the best hyperparameter combinations by using RandomizedSearchCV from scikit-learn.

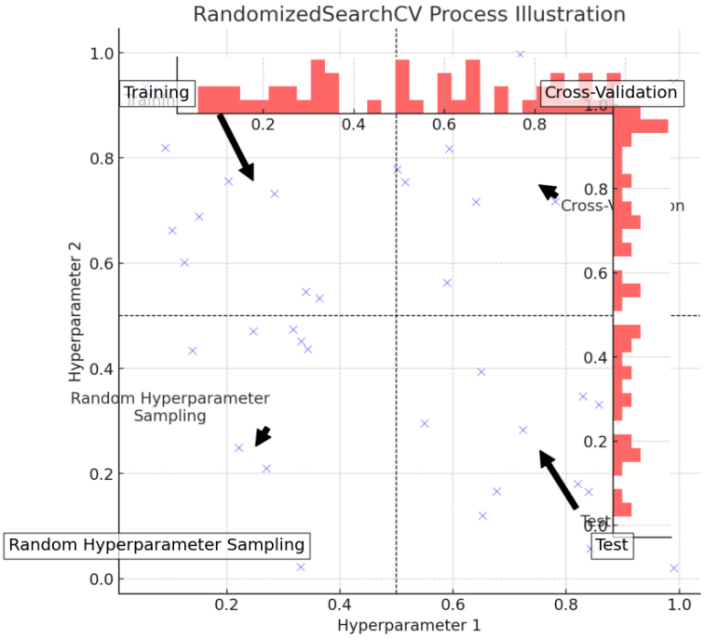

Figure 7. RandomSearchCV Process

RandomizedSearchCV is done by randomly sampling the hyperparameter configurations from a specified search space and then evaluating their performance using cross-validation. This approach allows for

broader exploration of hyperparameters, which can result in more optimal configurations [29]. The steps involved in RandomizedSearchCV are as follows:

Step 1: Define the possible range for each hyperparameter.

Step 2: Randomly select a set of hyperparameter configurations based on predetermined sampling time.

Step 3: Evaluate the model's performance using these configurations on the cross-validation set and calculate the objective function value.

Step 4: Select the hyperparameter configuration that yields the best performance, as determined by the objective function value.

Compared to traditional grid search methods, RandomizedSearchCV is more efficient because it can explore the hyperparameter space with random sampling. This allows it to find superior configurations with a limited number of samples. However, because it relies on random sampling, this method may not always find the globally optimal configuration.

Optuna uses the Hyperband method to perform optimization and test various hyperparameter combinations [30]. Optuna consists of several modules including Study, Storage, Trial, Sampler, and Pruner. The Study module is responsible for managing the objective function values for the best set of hyperparameters found, as well as setting the optimization method (Hyperband) and the number of trials to be conducted. Optuna helps us determine the optimal threshold values to maximize model accuracy [31]. It searches automatically for optimal hyperparameters by efficiently exploring its space using pruning and statistical modeling techniques. Optuna dynamically adjusts the search strategy based on the objective function and search space specified by the user and thus maximizes the model performance. This Optuna also supports various types of hyperparameters and integrates well with popular machine learning libraries, simplifying the hyperparameter optimization process and enhancing model performance.

The determination of hyperparameters or hyperparameter settings is essential in defining the structure of an Artificial Neural Network (ANN) model. The quality of the resulting model is highly dependent on the correct hyperparameter values [32]. The multilayer perceptron has hidden layers in this study, we investigate the influence of various hyperparameter settings on the effectiveness and efficiency of the artificial neural network model. The parameters to be used are shown in Table 3.

The model used has complex hidden layers with varying sizes from 5 to 100 neuron units in each layer, applied sequentially. Furthermore, the learning rate is set to (0.1, 0.01, 0.001, 0.0001). On the other hand, this method also uses batch sizes of (64, 128, 256, and 512) [33].

Table 3. Hyperparameter space

Parameter	Values
'batch_size':	64, 128, 256, 512
'hidden_layer_sizes':	(5 – 100)
'activation':	relu, tanh, logistic, identity
'solver':	sgd, adam, lbfgs
'learning_rate':	constant, adaptive, invscaling
'learning_rate_init':	(0.1 – 0.0001)
'alpha':	(0.1 – 0.0001)

3. Results and Discussions

This study presents various experiments conducted with data cleaning preparation and bag of words feature extraction. The machine learning method applied in this study was the multilayer perceptron. This research employed random search and Optuna as hyperparameter tuning approaches to optimize the parameters used by these algorithms. The algorithm's hyperparameters are listed in Table 3.

Experiment 1: Hyperparameter Optimization with Random Search vs Optuna

In an attempt to set the hyperparameters in this experiment, the study applied tuning on the dataset using the multilayer perceptron algorithm by dividing the dataset into 10 parts using the cross-validation technique. The two hyperparameter tuning approaches used were Random Search and Optuna. The optimal hyperparameters obtained are shown in Table 4.

Table 4. Hyperparameter Optimal Random Search and Optuna

Best Parameter	Random Search	Optuna
'batch_size':	64	64
'hidden_layer_sizes':	80	95
'activation':	Relu	Relu
'solver':	Adam	Sgd
'learning_rate':	Constant	Adaptive
'learning_rate_init':	0.0001	0.1
'alpha':	0.1	0.1

As shown in Table 4, the optimal hyperparameter settings obtained from Experiment 1 for the multilayer perceptron with random search optimization are: batch_size: 64, hidden_layer_size: 80, activation: relu, solver: adam, learning_rate: constant, learning_rate_init: 0.0001, and alpha: 0.1. Then, for Optuna: batch_size: 64, hidden_layer_size: 95, activation: relu, solver: sgd, learning_rate: adaptive, learning_rate_init: 0.1 and alpha: 0.1.

Experiment 2: Evaluation Metrics of Random Search and Optuna

After finding the optimal hyperparameters for the multilayer perceptron algorithm using both random search and Optuna approaches, this study used the tuned hyperparameters to train and evaluate the model, as well as calculate performance metrics for the algorithms used. The results of both models are summarized in Table 5.

Based on the model evaluation results, in the second experiment, this study yielded the highest F1-Score of

81.49% on the Optuna model, while the Random Search model achieved 79.70%. Figure 8 shows the comparison results of the two approaches, namely random search and Optuna.

Table 5. Comparison Of Random Search and Optuna Results

Method	Hidden Layer	Accuracy	Precision	Recall	F1-Score
Random Search	80	0.8309	0.8105	0.7838	0.7970
Optuna	95	0.8450	0.8237	0.8063	0.8149

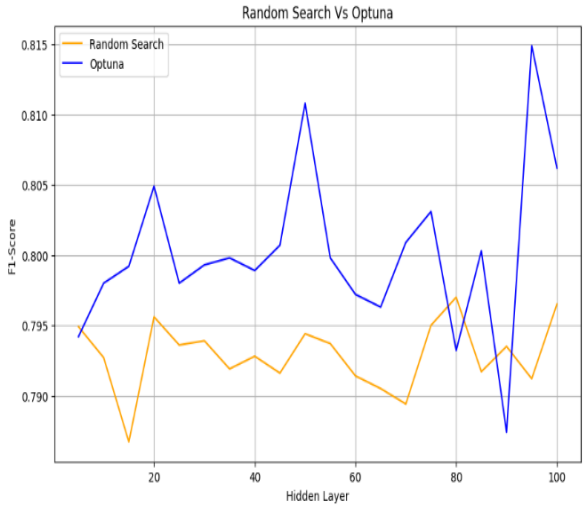

Figure 8. F1-Score Random Search and Optuna results

This improvement in performance using Optuna could be attributed to its more sophisticated hyperparameter optimization process, which explores a broader range of parameter combinations more effectively than Random Search. In the context of hate speech detection, this means that the Optuna-tuned model is better at balancing precision and recall, leading to more accurate identification of hate speech instances. For practical applications, this could translate to more reliable detection in environments where reducing false positives and capturing a higher number of actual hate speech occurrences is critical. The slightly higher computational cost may be justified in scenarios where accuracy is paramount.

From these results, this study concluded that Optuna allows the development of more efficient classifiers compared to the Random Search technique. This study also noted that both techniques achieved the highest classification accuracy levels. The results of Random Search tend to be closer to the results of Optuna. To deepen this comparison, this study proposes a third experiment focusing on the study of algorithm execution time with both techniques.

Experiment 3: Evaluation of Random Search and Optuna Execution Time

The third experiment of this study compares and measures the execution time in the hyperparameter tuning process for hate speech classification using multilayer perceptron and feature extraction with these two approaches, namely Random Search and Optuna. This study further summarizes the execution time results of both techniques, which can be seen in Table 6.

Table 6. Comparison Of Random Search And Optuna Execution Times

Random Search	Optuna
25.38 Hours	36.93 Hours

Random Search's best execution time took approximately 25.38 hours to complete. This method worked by randomly selecting hyperparameter combinations from a predefined search space. Although this method is simple and easy to implement, its drawback lies in its relatively low efficiency, as it does not consider information from previous iterations.

On the other hand, Optuna's best execution time of 36.93 hours. Optuna is a more sophisticated hyperparameter optimization method, using a Bayesian optimization approach to adaptively adjust the search space based on results from previous iterations. Although it requires a longer execution time, this method is expected to find more optimal hyperparameter combinations compared to Random Search.

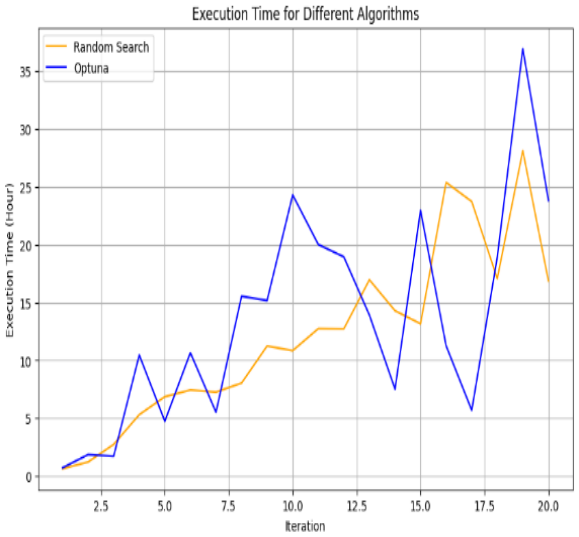

Figure 9. Algorithm time execution results

Based on the execution results illustrated in Figure 9, the Multilayer Perceptron algorithms show significant differences in execution time. Optuna's best execution time is 36.93 hours, while Random Search takes 25.38 hours. Although Optuna's execution time is notably longer, it achieves the highest F1-Score of 81.49%, compared to 79.70% for Random Search. This indicates that Optuna, despite its longer execution time, offers better performance. However, it is essential to consider whether this trade-off is practical for real-world applications, where the additional accuracy may or may not justify the increased execution time in a production environment.

Experiment 4: Comparing Previous Models and Proposed Models

In the fourth experiment, similar dataset results from the study [10] were compared to detect hate speech on Twitter in Indonesia.

Table 7. Comparison Of Previous Research

Scenario	Accuracy	F1-Score
LP with RFDT and Unigram [10]	66.12%.	-
One-vs-All using ANN with BoW and Chi-Square [11]	86.79%	74.66%
DistilBERT with Support Vector Machine (SVM) [12]	-	78.5%
Hyperparameter Tuning Random Search and Optuna using MLP with BoW *This Research	84.50%	81.49%

In Table 7, this study conducts a comparative analysis between several classifications methods used in hate speech detection by combining Hyperparameter Tuning Random Search and Optuna with Multilayer Perceptron (MLP) using Bag of Words (BoW) as feature extraction. The study results were compared to several previous studies LP with RFDT and Unigram [10] This method achieves an accuracy of 66.12%. However, this accuracy result is relatively lower compared to the other methods, indicating that this combination may be less effective in the context of hate speech detection.

One-vs-All using ANN with BoW and Chi-Square [11] this method achieved an accuracy of 86.79% and an F1-Score of 74.66%. These results indicated a quite good performance in classifying hate speech, especially in terms of relatively high accuracy.

DistilBERT with Support Vector Machine (SVM) [12] This approach managed to achieve an F1-Score of 78.5%. By using more advanced deep learning models such as DistilBERT, these results show an advantage in terms of higher predictive accuracy compared to several other methods.

Hyperparameter Tuning Random Search and Optuna using MLP with BoW In this study, this method managed to achieve an accuracy of 84.50% and an F1-Score of 81.49%. These results indicated that the use of hyperparameter tuning techniques such as Random Search and Optuna can improve the performance of MLP in hate speech classification. The high F1-Score also shows a good balance between precision and recall, which is essential in hate speech detection.

4. Conclusions

Hyperparameter Tuning is a technique for optimizing the parameters provided by machine learning algorithms, allowing efficient model creation and accurate classification identification. This study aims to compare the effectiveness of two hyperparameter tuning techniques, namely Optuna and Random Search, in classifying hate speech on the social media platform Twitter. The results showed that Optuna outperforms Random Search in terms of accuracy, precision, recall, and F1-Score of 81.49%, but had a longer execution time when applied to Multilayer Perceptron and Bag of Words feature extraction. This study contributes to the literature by demonstrating the effectiveness of Optuna, which is not limited to a specific language or dataset, yet also applies to hate speech classification on Indonesian-language Twitter. Additionally, this research emphasizes the importance of hyperparameter tuning techniques to improve the performance of machine learning algorithms on Indonesian text data. Future work can add hybrid methods and explore other hyperparameter tuning techniques or investigate the effectiveness of various feature extraction methods in classifying Indonesian language news. Overall, this study provides valuable insights for researchers and practitioners working on Indonesian language text classification tasks.

References

[1] J. Kansok-Dusche *et al.*, "A Systematic Review on Hate Speech among Children and Adolescents: Definitions, Prevalence, and Overlap with Related Phenomena," Oct. 01, 2023, *SAGE Publications Ltd*. doi: 10.1177/15248380221108070.

[2] M. Anand, K. B. Sahay, M. A. Ahmed, D. Sultan, R. R. Chandan, and B. Singh, "Deep learning and natural language processing in computation for offensive language detection in online social networks by feature selection and ensemble classification techniques," *Theor Comput Sci*, vol. 943, pp. 203–218, Jan. 2023, doi: 10.1016/j.tcs.2022.06.020.

[3] B. Elisa Shearer, A. Mitchell, J. Research Elisa Shearer, R. Associate Hannah Klein, and C. Manager, "FOR MEDIA OR OTHER INQUIRIES," 2021. [Online]. Available: www.pewresearch.org

[4] B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal, and A. Mukherjee, "HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection," Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.10289

[5] "Kemp, S. (2021). https://datareportal.com/reports/digital-2021-global-overviewreport."

[6] Y. Wang, J. Guo, C. Yuan, and B. Li, "Sentiment Analysis of Twitter Data," Nov. 01, 2022, *MDPI*. doi: 10.3390/app122211775.

[7] R. Kumar, A. K. Ojha, S. Malmasi, and M. Zampieri, "Benchmarking Aggression Identification in Social Media," 2018. [Online]. Available: https://competitions.codalab.org/

[8] M. Subramanian, V. Easwaramoorthy Sathiskumar, G. Deepalakshmi, J. Cho, and G. Manikandan, "A survey on hate speech detection and sentiment analysis using machine learning and deep learning models," Oct. 01, 2023, *Elsevier B.V.* doi: 10.1016/j.aej.2023.08.038.

[9] T. Elansari, H. Bourray, and M. Ouanan, "Modeling of Multilayer Perceptron Neural Network Hyperparameter Optimization and Training," 2023, doi: 10.21203/rs.3.rs-2570112/v1.

[10] M. O. Ibrohim and I. Budi, "Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter," 2019. [Online]. Available: https://www.komnasham.go.id/index.php/

[11] E. Utami, Rini, A. F. Iskandar, and S. Raharjo, "Multi-Label Classification of Indonesian Hate Speech Detection Using One-vs-All Method," in *Proceedings - 2021 IEEE 5th International Conference on Information Technology, Information Systems and Electrical Engineering: Applying Data Science and Artificial Intelligence Technologies for Global Challenges During Pandemic Era, ICITISEE 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 78–82. doi: 10.1109/ICITISEE53823.2021.9655883.

[12] N. Azmi Verdikha, R. Habid, and A. Johar Latipah, "Analisis DistilBERT dengan Support Vector Machine (SVM) untuk Klasifikasi Ujaran Kebencian pada Sosial Media Twitter,"

METIK JURNAL, vol. 7, no. 2, pp. 101–110, Dec. 2023, doi: 10.47002/metik.v7i2.583.

[13] A. M. U. D. Khanday, S. T. Rabani, Q. R. Khan, and S. H. Malik, "Detecting twitter hate speech in COVID-19 era using machine learning and ensemble learning techniques," *International Journal of Information Management Data Insights*, vol. 2, no. 2, Nov. 2022, doi: 10.1016/j.jjimei.2022.100120.

[14] R. W. Acuña Caicedo, J. M. Gómez Soriano, and H. A. Melgar Sasieta, "Bootstrapping semi-supervised annotation method for potential suicidal messages," Apr. 01, 2022, *Elsevier B.V.* doi: 10.1016/j.invent.2022.100519.

[15] "Retracted: Analysing Hate Speech against Migrants and Women through Tweets Using Ensembled Deep Learning Model," *Comput Intell Neurosci*, vol. 2023, pp. 1–1, Oct. 2023, doi: 10.1155/2023/9781063.

[16] K. Shaker, "Optimizing Sentiment Big Data Classification Using Multilayer Perceptron," *Anbar Journal of Engineering Sciences*, vol. 13, no. 2, pp. 14–21, Nov. 2022, doi: 10.37649/aengs.2022.176353.

[17] H. Mehta and K. Passi, "Social Media Hate Speech Detection Using Explainable Artificial Intelligence (XAI)," *Algorithms*, vol. 15, no. 8, Aug. 2022, doi: 10.3390/a15080291.

[18] H. Cam, A. V. Cam, U. Demirel, and S. Ahmed, "Sentiment analysis of financial Twitter posts on Twitter with the machine learning classifiers," *Heliyon*, vol. 10, no. 1, Jan. 2024, doi: 10.1016/j.heliyon.2023.e23784.

[19] I. de Zarzà, J. de Curtò, and C. T. Calafate, "Optimizing Neural Networks for Imbalanced Data," *Electronics (Switzerland)*, vol. 12, no. 12, Jun. 2023, doi: 10.3390/electronics12122674.

[20] W. F. Satrya, R. Aprilliyani, and E. H. Yossy, "Sentiment analysis of Indonesian police chief using multi-level ensemble model," in *Procedia Computer Science*, Elsevier B.V., 2022, pp. 620–629. doi: 10.1016/j.procs.2022.12.177.

[21] A. Toktarova *et al.*, "Hate Speech Detection in Social Networks using Machine Learning and Deep Learning Methods." [Online]. Available: www.ijacsa.thesai.org

[22] B. Morris, "The components of the wired spanning forest are recurrent," *Probab Theory Relat Fields*, vol. 125, no. 2, pp. 259–265, Feb. 2003, doi: 10.1007/s00440-002-0236-0.

[23] S. Sadiq, A. Mehmood, S. Ullah, M. Ahmad, G. S. Choi, and B. W. On, "Aggression detection through deep neural model on Twitter," *Future Generation Computer Systems*, vol. 114, pp. 120–129, Jan. 2021, doi: 10.1016/j.future.2020.07.050.

[24] H. T. Vo, H. T. Ngoc, and L. Da Quach, "An Approach to Hyperparameter Tuning in Transfer Learning for Driver Drowsiness Detection Based on Bayesian Optimization and Random Search," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 4, pp. 828–837, 2023, doi: 10.14569/IJACSA.2023.0140492.

[25] L. Yang and A. Shami, "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice," Jul. 2020, doi: 10.1016/j.neucom.2020.07.061.

[26] Z. B. Zabinsky, "Random Search Algorithms," 2009.

[27] J. Bergstra, J. B. Ca, and Y. B. Ca, "Random Search for Hyper-Parameter Optimization Yoshua Bengio," 2012. [Online]. Available: http://scikit-learn.sourceforge.net.

[28] I. Jamaleddyn, R. El ayachi, and M. Biniz, "An improved approach to Arabic news classification based on hyperparameter tuning of machine learning algorithms," *Journal of Engineering Research (Kuwait)*, vol. 11, no. 2, Jun. 2023, doi: 10.1016/j.jer.2023.100061.

[29] Y. Zhao, W. Zhang, and X. Liu, "Grid search with a weighted error function: Hyper-parameter optimization for financial time series forecasting," *Appl Soft Comput*, vol. 154, Mar. 2024, doi: 10.1016/j.asoc.2024.111362.

[30] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," Jul. 2019, [Online]. Available: http://arxiv.org/abs/1907.10902

[31] O. Dib, Z. Nan, and J. Liu, "Machine learning-based ransomware classification of Bitcoin transactions," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, no. 1, Jan. 2024, doi: 10.1016/j.jksuci.2024.101925.

[32] Z. Car, S. Baressi Šegota, N. Anđelić, I. Lorencin, and V. Mrzljak, "Modeling the Spread of COVID-19 Infection Using a Multilayer Perceptron," *Comput Math Methods Med*, vol. 2020, 2020, doi: 10.1155/2020/5714714.

[33] R. Marco, S. S. S. Ahmad, and S. Ahmad, "An Improving Long Short Term Memory-Grid Search Based Deep Learning Neural Network for Software Effort Estimation," *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 4, pp. 164–180, 2023, doi: 10.22266/ijies2023.0831.14.

