BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 4, June 2026 | Hal 1225-1232
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i4.996

Influence of Imbalanced Data on Text Classification Using Recurrent Neural Network

Rina Septiriana*, Tursina

Engineering Faculty, Informatics Study Program, Universitas Tanjungpura, Pontianak, Indonesia
Email: [1,*]rinaseptiriana@informatika.untan.ac.id, [2]tursina@informatika.untan.ac.id
Correspondence Author Email: rinaseptiriana@informatika.untan.ac.id

Abstract−Recurrent Neural Networks (RNNs) such as LSTM and GRU are designed for sequential data. However, their performance in emotion detection is often compromised by class imbalance. This study compares LSTM and GRU architectures for classifying emotional states using a dataset of 4,386 Indonesian tweets. The dataset exhibits a mild imbalance (approximately 1.7:1) across five classes: Anger, Happy, Sadness, Love, and Fear. However, the effectiveness of these models is often hindered by class imbalance in datasets, which biases predictions toward majority classes and compromises the reliability of standard metrics. This study aims to systematically evaluate the comparison of LSTM and GRU architectures in processing imbalanced Indonesian emotional tweet data. The methodology involves evaluating these models across various resampling techniques, including Random Oversampling, SMOTE, and Near-Miss. Key findings reveal that LSTM consistently outperforms GRU in capturing complex emotional patterns. Specifically, the LSTM model combined with Random Oversampling emerged as the most robust configuration, achieving a Macro-F1 score of 71% and an accuracy of 73%. While Random Oversampling effectively enhanced minority class recognition without overfitting, SMOTE and Near-Miss introduced significant performance trade-offs. These results provide actionable insights for selecting optimal architectures and resampling strategies to mitigate imbalance-related biases in sequential classification tasks.

Keywords: Deep Learning; Imbalanced Data; Recurrent Neural Networks; Gated Recurrent Unit; Resampling

1. INTRODUCTION

Deep learning is a branch of machine learning that using artificial neural networks. Deep learning outperforms shallower machine learning models and conventional data analysis techniques across various applications [1]. Deep Learning algorithms flow data through another layer. Every layer extracts features gradually and passes them to the next layer. The first layer extracts basic features, and the next layer forms a complete data representation. Examples of deep learning include unsupervised pre-trained networks, CNN for images, and RNN for sequential data like text [2].

Recurrent Neural Networks (RNNs) offer significant advantages for time-series analysis and speech recognition due to their inherent capacity to model contextual dependencies. Unlike standard feed-forward neural networks, which process inputs in isolation, RNNs maintain a recursive hidden state that captures sequential context, making them particularly suitable for tasks like Natural Language Processing (NLP) and video classification where temporal information is critical. However, standard RNNs often struggle to capture long-term dependencies because of the Vanishing Gradient Problem. This mathematical constraint occurs during backpropagation through time, where error signals diminish exponentially, effectively preventing the model from learning relationships between distant elements in a sequence. Advanced architectures, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), have successfully mitigated this constraint by employing gating mechanisms that allow the network to selectively retain or discard information over extended sequences [3].

Data for deep learning, especially on RNN, has a significant role in training a robust machine learning model. There are some types of data, for example, Structured Data, which is organized data like tables, spreadsheets, or databases, and then semi-structured data, like data that does not have a partial structure like email or HTML documents; the last one is unstructured data like images, videos, and audio [4].

To make a reasonable classification, data is used in a learning process. In classification, data is divided into the form of balanced data and imbalanced data [5]. The issue of data distribution on imbalanced data is a tendency of class with a significant number that can get chosen in the classification process compared to a limited number of data [6]. This matter causes a lack of accuracy in the classification process because of bias problems in imbalanced data, which decreases information from the data.

Classification trains data with a label to identify new data that do not have a label for class determination. Imbalanced data is an issue that causes imbalanced distribution between classes. Usually, data is dominated by one class (majority class) and another class with a tiny quantity (minority class). Therefore, the classification model tends to side with the majority class and dismiss the minority class. It can make model performance deteriorate when data is in imbalanced conditions. Some techniques handle imbalanced data, such as Resampling. Resampling is a method to balance data that adds (oversampling) or reduces (undersampling) from certain classes [7].

Class imbalance occurs when the distribution of categories within a dataset is significantly uneven, leading to a predictive bias where the model favors the majority class. This issue critically undermines model robustness. An analysis of the study by Handoyo et al. (2025) illustrates this challenge; their LSTM-based model, trained on 651,191 URLs, achieved high nominal accuracy and performed effectively on benign and defacement categories. However, the model struggled significantly with malware and phishing detection [8].

This performance gap demonstrates the Accuracy Paradox: in imbalanced datasets, high accuracy can be a "statistical lie." For instance, if 90% of the data is benign, a "lazy" model that classifies every input as benign would

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 4, June 2026 | Hal 1225-1232
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i4.996

achieve 90% accuracy while remaining 0% useful for detecting threats. Consequently, accuracy is an insufficient metric for evaluating imbalanced sequential data, as it fails to reflect the model's inability to identify critical minority classes. To address these biases and ensure reliability for real-time phishing detection, optimizations such as class weighting and resampling techniques are essential.The learning process on neural networks determines the best weight with the slightest error. However, in a learning process, the probability of selecting the slightest error that can happen in the class is high. So, the deep learning process will solve the problem above. However, the significant effect of imbalanced data on deep learning still needs further research because of another factor that can affect classification's accuracy used in deep learning.

Research by Thabtah et al. (2020), who used the Naive Bayes Algorithm, resulted in an F1-Score of 94% for imbalanced data without resampling, 94% for data that used random under-sampling, and 96% for SMOTE [9]. Meanwhile, Korkmaz (2020) trained a classification model for active compound detection using deep learning. Imbalanced data decreases classification accuracy from deep learning. This research observes imbalanced mitigation for five data tests from PubChem, which uses one undersampling method and three oversampling methods. The results show that data has been balanced but affects deep learning performance [10].

This study examines the impact of imbalanced data on deep learning models. A resampling approach is applied to a Recurrent Neural Network (RNN) to address this. The research employs various resampling techniques to assess its effectiveness in mitigating class imbalance, including random oversampling, SMOTE, and Near-Miss.

2. RESEARCH METHODOLOGY

2.1 Research Stages

This research focuses on analysis by experiment to see the influence of data imbalances to determine data classification using the RNN Algorithm with LSTM and GRU. Figure 1 shown the research stage that was used.

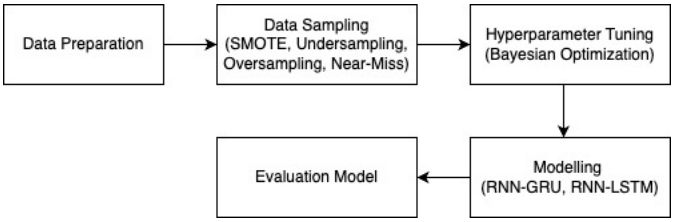

Figure 1. Research Stage

The first stage is data preparation and preprocessed data, and the second stage is data sampling to synthesize data with a sampling technique called SMOTE, undersampling. Oversampling dan Near-Miss, the third stage is Hyperparameter tuning for RNN with Bayesian Optimization, the fourth stage is modeling with RNN, RNN that will be used for modeling process comprised of RNN-GRU and RNN-LSTM. The last stage is the evaluation model based on the result of the modeling process.

2.2 Data Source

The data that will be used data with five labels is the tweet emotion dataset by Negara et al. (2021) about the classification of emotional tweets in Indonesian [11]. Distribution of the data that will be used represent on Figure 2.

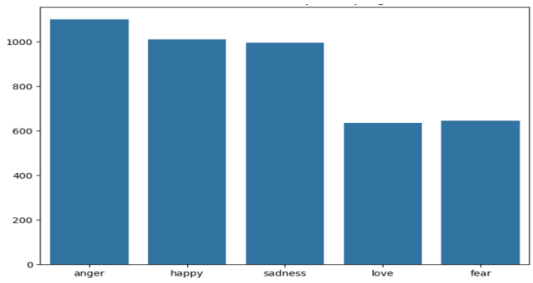

Figure 2. Dataset Distribution

Figure 2 illustrates the distribution of five emotional categories within the dataset: Anger (1101), Happy (1010), Sadness (996), Love (635), and Fear (644). While the numerical gap represents a statistically gentle imbalance, the decision to employ resampling stems from the specific data requirements of Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTMs). These models rely on observing diverse sequential patterns to effectively optimize their internal gating mechanisms.

Because of a slight distinction in some number datasets, for example, of 'Love' and 'fear' between 635 and 644. The models lack sufficient exposure to the linguistic variance and emotional nuances of these categories compared to the majority classes. Such a disparity can lead to biased gradient updates during training, in which the model prioritizes

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 4, June 2026 | Hal 1225-1232
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i4.996

features of more frequent emotions. To mitigate this, resampling methods such as SMOTE were used to expand the feature space of the minority classes synthetically. This ensures that the LSTM and GRU units can learn more robust representations, ultimately improving the model's performance in accurately recognizing less frequent emotional states.

2.3 Bayesian Optimization

Bayesian Optimization is an efficient method for optimizing models, particularly in cases where traditional approaches struggle with complex gradient computations. Unlike conventional methods, Bayesian Optimization constructs a model of the function based on probability to be optimized as the goal. According to Brochu et al. (2010), Gaussian Processes (GP) serve as a prior distribution in Bayesian Optimization, allowing for predictions of unknown function behavior [12] . Mathematically, a GP models a function which can be seen in Equation (1):

$$f(x^{(*)}) \geq f(x), \forall x \; s.t. \left\| x^{(*)} - x \right\| < \varepsilon \tag{1}$$

This formula states $x^{(*)}$ that it is a local maximum of the function f(x) because, within a small region around $x^{(*)}$, the function value at $x^{(*)}$ is at least as significant as the function values at nearby points. In other words, $x^{(*)}$ symbolizes a little neighborhood's peak. A local maximum is also a global maximum if the function's negative, −f(x), is convex.

For this research, bayesian optimization is used to targeting optimal configurations for learning rate, dropout, hidden units, and embedding dimensions across both LSTM and GRU architectures to maximize classification reliability. For the GRU architecture, the search space included optimizers (Adam, RMSprop, Nadam), dropout rates (0.1–0.5), hidden units (50–200), and batch sizes (32–64). Similarly, the LSTM model was optimized across embedding dimensions (32–128), LSTM units (32–128), dropout rates (0.1–0.5), and learning rates ranging from 0.0001 to 0.01.

2.4 Recurrent Neural Network

A recurrent Neural Network (RNN) is portion of Artificial Neural Network which simulates neuron connections in the human brain. In ANN, the connection between neurons sends a signal like synapses to the biological brain. An artificial process is a signal that is received and forwarded to another neuron through a weighted connection which can be fitted to strong or weakened signals along with learning processing. The RNN architecture includes an input, output and hidden layer, where the hidden layer is responsible for computing weight fitting to get an output. RNN has forward information flow between the input and hidden layer, also directed loop which compares errors between the recent hidden layer and past hidden to fit a weighted [13].

2.4.1 Long Short Term Memory (LSTM)

LSTM is RNN that has the capacity to learn long-term dependence. The main of LSTM is a cell state that able to include or remove the information through a gate mechanism (gate). Its framework is shown in Figure 3.

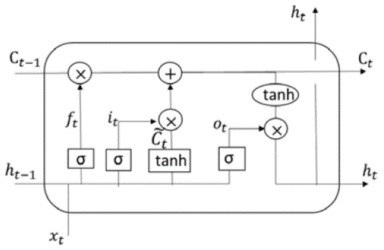

Figure 3. Architecture of LSTM [14]

An input gate, an output gate, and an forget gate are the three main gates of an LSTM. A network's ultimate output is determined by the output gate, which comes after the forget gate defines what details should be removed from the cell state and the input gate indicates what details should be updated [14]. The state of its node can be seen in Equation (2) to (4).

$$f_t = \sigma\left(W_f . [h_{t-1}, x_t] + b_f\right) \tag{2}$$

$$O_t = \sigma(W_o . [h_{t-1}, X_t] + b_o) \tag{3}$$

$$h_t = o_t * tanh(C_t) \tag{4}$$

The LSTM's current input, x_{t-1} passes through on h_{t-1} as hidden state of the preceding layer. W and b stand for weight and bias, σ for the sigmoid function, ft the forget gates output, it for the input gate's output. The following layer's cell state is shown by C_{t-1}, the output of the output gate is O_t, the brief intermediate state is indicated by $(\widetilde{C_t})^{\sim}$, and the hidden state is indicated by h_t.

2.4.2 Gated Recurrent Unit (GRU)

Cho et al. and Chung et al. (2014) introduced the GRU, a more straightforward and computationally efficient version of the LSTM. Two major distinctions account for its simplicity: (1) it combines the hidden state and the internal cell state,

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 4, June 2026 | Hal 1225-1232
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i4.996

and (2) it merges forget gates and the input into a single update gate. Equations (5) through (8) are used in Figure 3 to depict the GRU unit and its calculations.

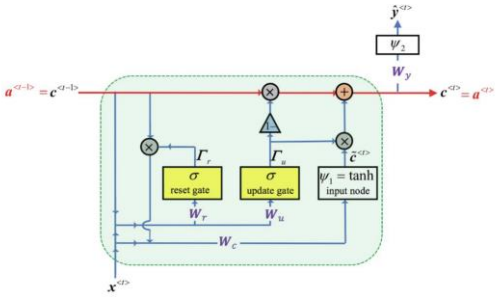

Figure 4 Architecture of GRU [15]

$$\tilde{C}^{<t>} = tanh(W_c[x^{<t>}; \Gamma_r * c^{<t-1>}] + b_c) \quad (5)$$

$$\Gamma_u = \sigma(W_u[x^{<t>}; c^{<t-1>}] + b_u) \quad (6)$$

$$\Gamma_r = \sigma(W_r[x^{<t>}; c^{<t-1>}] + b_r) \quad (7)$$

$$\hat{y}^{<t>} = \psi_2(W_y c^{<t>} + b_y) \quad (8)$$

The input data point in time step t is $x^{<t>}$, while the projected output at time step t is $\hat{y}^{<t>}$. The internal states at time steps t-1 and t, respectively, are denoted by $c^{<t-1>}$ and $c^{<t>}$. The candidate internal state at time step t is $\tilde{c}^{<t>}$. The update and reset gates' respective outputs are denoted by Γ_u and Γ_r. The weight matrix for the update and reset gates is denoted by W_u and W_r, respectively. The input and output weight matrices are denoted by W_c and W_y, respectively [15].

2.5 Resampling Data

Resampling techniques are methods used to address imbalanced data by adjusting amount of the data, that aim to attain a balanced distribution. These techniques leverage the original data through various approaches. One such method, random oversampling, involves duplicating randomly selected instances from the minority class to increase its representation, thereby improving model performance in recognizing underrepresented categories [16].

2.5.1 Oversampling Method

a. Random Oversampling
Random Oversampling is a method employed to balance class distributions in datasets by reproducing samples from the minority class at random. While this method can mitigate class imbalance, it has the drawback of potentially leading to overfitting, as the model may memorize these duplicated instances rather than learning generalizable patterns. There are some stage to do, that is (1) Identify Class Imbalance: Determine that the minority class has fewer instances compared to the majority class, (2) Duplicate Minority Instances: Randomly select and replicate existing minority class examples until the distribution of classes is balanced. and (3) Balanced Dataset. Result of random sampling is dataset that has an equal number of instances for both majority and minority classes, allowing machine learning algorithms to train without bias toward the majority class. For instances, a dataset contains 900 majority class samples and only 100 minority class samples, random oversampling would involve duplicating the minority class instances to reach 900, achieving a balanced class distribution [17].

b. SMOTE
A technique called the Synthetic Minority Over-sampling Technique (SMOTE) makes artificial samples for the class of minorities. with the objective to deal with data imbalance. This technique enhances the variety of the minority class, thereby enhancing the effectiveness of machine learning models in handling imbalanced datasets. There are some stage for it. Firstly, identify Minority Class Instances by selecting samples from the minority class. Secondly, Determine Nearest Neighbors for each selected minority class sample, the algorithm identifies its k-nearest neighbors within the feature space. Thirdly, generate Synthetic Samples: New synthetic instances are created by interpolating between the selected sample and its nearest neighbors. This is achieved by choosing a point along the portion of line segment that connects the original sample and a neighbor, thereby introducing new, plausible samples within the minority class feature space. Lastly, Repeat Until Balanced that is the process which is repeated until the minority class reaches a desired level of representation, effectively balancing the dataset [18].

2.5.1 Undersampling Method

a. Random Undersampling
Balancing the number of instances across classes by reducing the dominant class data, known as undersampling, is a more efficient method that accelerates the training process. However, the primary disadvantage of this method is the

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 4, June 2026 | Hal 1225-1232
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i4.996

potential destruction of important data from the removed data, which may affect the model's ability to recognize patterns in the dataset [19].

b. Near-Miss

Near-Miss is undersampling method designed to address data imbalance through reducing the quantity of cases of the majority class, giving an equal division of classes. This technique selects samples from the majority class according to their proximity to minority class samples within the feature space. Near-Miss employs various strategies that rely on distance measurements to determine which majority class examples should be retained [20]. The process involves reducing the majority class while preserving the most relevant instances that help distinguish between classes. By selecting samples that are either closer or farther from the minority class, Near-Miss ensures that only the most informative majority class instances are kept, enabling machine learning models to better learn the key differences between classes. To get the intended outcomes, parameters were sent through to control the near-miss technique. The near-miss method has three variations. Firstly, NearMiss - 1 indentifies the test data that, on average, is closest to the closest samples in the negative class. Secondly, NearMiss - 2 selects the positive samples closest to the farthest samples in the negative class on average. The final one is the two-step NearMiss - 3 technique. The first step is to keep the closest neighbors of each negative instances. They are then selected considering the typical distance between the positive samples and their closest neighbors.

3. RESULT AND DISCUSSION

3.1 Dataset

Five datasets are used in this research: the dataset without resampling, the undersampling dataset, the oversampling dataset, the Near-Miss dataset, and the SMOTE dataset. Figure 5 depicts the comparison of the distribution data.

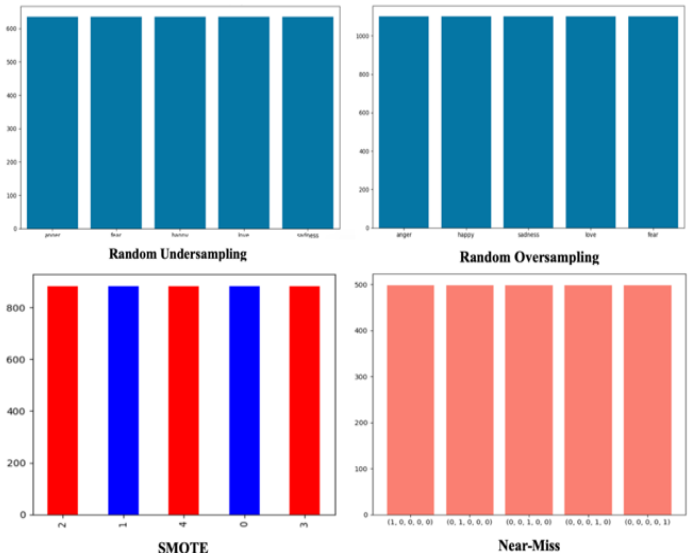

Figure 5 Results of Data Resampling

In figure 5, the results of each data resampling technique can be seen. It comprises Random Undersampling, which reduced the dataset to 635 samples; Random Oversampling, which increased it to 1101 samples; SMOTE (Synthetic Minority Over-sampling Technique), resulting in 1105 samples; and Near-Miss, which left the dataset with 498 samples. These diverse strategies show how various resampling techniques can alter the dataset's size and distribution, which can have an effect on the model training and assessment stages that follow.

3.2 Recurrent Neural Network Model

Deep learning models are trained by leveraging optimal configurations of units within RNN-GRU and RNN-LSTM architectures. To achieve these optimal configurations, Bayesian Optimization is an efficient method for identifying the most suitable hyperparameters. This method uses knowledge from earlier assessments to feed later searches, balancing exploration and exploitation to explore the hyperparameter space intelligently. This method works well as a surrogate model, especially when optimizing expensive processes like model performance evaluations, which produce better-performance models.

3.2.1 Long Short Term Memory (LSTM)

To improve the LSTM model's performance, an optimal parameter search was performed, focusing on Embedding Dimension, Dropout Rate, LSTM Units, Batch Size, and Learning Rate. This process was carried out to identify the most suitable parameter values for the given dataset. Table 1 shows an analogy of the obtained parameters.

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 4, June 2026 | Hal 1225-1232
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i4.996

Table 1 Result of Hyperparameter LSTM

	Embedding Dim	Dropout Rate	LSTM Units	Batch Size	Learning Rate
Non-Sampling	128	0,4859	96	32	0,0009
Random Undersampling	64	0,2640	32	32	0,0007
Random Oversampling	128	0,2418	32	64	0,0019
SMOTE	128	0,3378	64	64	0,0008
Near-Miss	128	0,3378	64	64	0,0008

Table 1 analyzes the optimized hyperparameters for an LSTM model trained using different data sampling techniques. Most sampling techniques (Non-Sampling, Random Oversampling, SMOTE, and Near-Miss) use embedding dimensions 128. Only Random Undersampling reduces this value to 64, possibly due to the reduced dataset size requiring fewer features to prevent overfitting. Non-Sampling has the highest dropout rate (0.4859), indicating that using the original imbalanced dataset requires a stronger regularization. Expanding the sample size of minority classes might decrease the requirement for regularization, as Random Oversampling has the lowest dropout rate (0.2418). SMOTE and Near-Miss share the same dropout rate (0.3378), implying a possible pattern in handling class imbalance.

The highest number of LSTM units (96) is found in Non-Sampling, meaning more complex patterns may need to be captured from the raw, imbalanced data. Other techniques converge to 32 or 64 units, indicating that balancing the dataset through sampling simplifies the model's complexity requirements. The batch size is either 32 or 64 across all configurations. Random Oversampling, SMOTE, and Near-Miss use 64, likely due to increased data volume after oversampling. Non-sampling and Random Undersampling use 32, likely due to limited data in undersampling or optimization requirements in the non-sampling case. Learning rates range between 0.0007 and 0.0019. The highest learning rate (0.0019) is seen in Random Oversampling, which could be necessary for faster convergence in a dataset with duplicate samples. The lowest learning rate (0.0007) appears in Random Undersampling, possibly to prevent instability caused by the reduced dataset.

3.2.2 Gated Recurrent Unit (GRU)

The Optimizer, Dropout Rate, GRU Units, Batch Size, and Epoch were among the parameters searched for to optimize the GRU model. This procedure aims to use the provided dataset to identify the best parameter values. A comparison of the generated parameters is shown in Table 2.

Table 2. Result of Hyperparameter GRU

	Optimizer	Dropout Rate	GRU Units	Batch Size	Epoch
Non-Sampling	nadam	0,4511	199	64	6
Random Undersampling	nadam	0,2154	176	32	7
Random Oversampling	nadam	0,4033	67	32	5
SMOTE	rmsprop	0,4382	63	64	6
Near-Miss	adam	0,2162	179	32	9

Different hyperparameter combinations used to train a deep learning model under various data sampling approaches are shown in Table 1. The frequent usage of the nadam optimizer across many techniques is demonstrated by its employment in Non-Sampling, Random Undersampling, and Random Oversampling. On the other hand, Adam is selected for Near-Miss, and rmsprop is used for SMOTE. Meanwhile, The highest dropout rate (0.4511) is observed in the Non-Sampling method, which might indicate a need for regularization when no resampling is applied. Meanwhile, Near-Miss and Random Undersampling have the lowest dropout rates (0.2162 and 0.2154, respectively), suggesting less reliance on dropout when handling imbalanced data through undersampling. SMOTE has the fewest GRU units (63), whereas non-sampling has the most (199). Moreover, SMOTE and Random Oversampling often need fewer GRU units, perhaps because of the tendency of the synthetic data to overfit.

The batch sizes range from 32 to 64, with different degrees of batch processing efficiency. Additionally, undersampling, oversampling, and Near-Miss use 32, whereas SMOTE and Non-Sampling use 64. Random Oversampling runs the shortest training (5 epochs), while Near-Miss runs the longest (9 epochs). Meanwhile, Random Undersampling and Non-Sampling show moderate training durations (7 and 6 epochs, respectively). Furthermore, SMOTE and Non-Sampling use 64, while undersampling, oversampling, and Near-Miss use 32. Near-Miss runs the most extended training (9 epochs), while Random Oversampling runs the shortest (5 epochs). Meanwhile, Random Undersampling and Non-Sampling show moderate training durations (7 and 6 epochs, respectively).

3.2.3 Discussion

To conduct a comprehensive evaluation of the GRU and LSTM models, a set of evaluation metrics will be employed to assess their performance. The primary metrics used for this evaluation will include Accuracy, F1-Score (macro average), and F1-Score (weighted average). These metrics will serve as the key tools to measure the effectiveness of each model in terms of overall correctness, class-wise balance, and the ability to handle imbalanced datasets. The evaluation process will involve training both models on the same dataset, followed by testing and comparing their results using the aforementioned metrics to determine which model performs better under specific conditions.

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 4, June 2026 | Hal 1225-1232
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i4.996

Table 3 Evaluation Metric Result of GRU and LSTM

	GRU			LSTM		
	Accuracy	F1-Score (macro avg)	F1-Score (weighted avg)	Accuracy	F1-Score (macro avg)	F1-Score (weighted avg)
Non-Sampling	0,58	0,59	0,59	0,61	0,62	0,61
Random Undersampling	0,57	0,57	0,57	0,60	0,60	0,60
Random Oversampling	0,71	0,71	0,71	0,73	0,72	0,72
SMOTE	0,49	0,48	0,49	0,51	0,51	0,51
Near-Miss	0,57	0,58	0,57	0,51	0,51	0,51

The weighted-average, macro-average, and accuracy F1-scores evaluation metrics for the GRU and LSTM models using various sampling strategies are contrasted in Table 3. An analysis of the results, showing the performance patterns, advantages, and disadvantages of each model under various sampling techniques, is provided below:

a. The combinations show how much better the LSTM model performs than the GRU model. Accuracy, weighted F1 score, and macro F1 score emphasize its capability to spot complicated patterns and long-term sequential connections in the data. This indicates that the LSTM's memory cells and gated algorithms are better suited to handling the complexity of the information.
b. For both models, random oversampling turns out to be the best sampling method. The LSTM model's accuracy is 73%, whereas the GRU model is 71%. The replication of minority class samples, which successfully balances the dataset without causing appreciable overfitting, is responsible for this improvement. According to the results, a balanced class distribution can increase model performance.
c. Both models show average results when no sampling method is used (non-sampling). The accuracy of the GRU model is 58%, whereas the LSTM model is 61%. This shows that, while there is potential for improvement, each model can learn from data imbalances well.
d. Random Undersampling leads to a slight decline in performance for both models. This result of crucial information from the majority class being missed, which reduces the models' ability to generalize effectively. The results highlight the trade-off between balancing class distribution and retaining valuable data, emphasizing the need for careful consideration when applying undersampling techniques.
e. SMOTE (Synthetic Minority Oversampling Technique) yielded the weakest results, with the GRU and LSTM models reaching only 49% and 51% accuracy, respectively. This poor performance is likely due to 'feature space noise' introduced during word embedding interpolation. In NLP tasks, SMOTE may create synthetic vectors between semantically distinct neighbors such as 'I hate this' and 'I am so angry' that do not correspond to any real linguistic structure. Unlike Random Oversampling, which preserves the original tweet data, SMOTE introduces deformed signals that hinder model generality. These findings suggest that maintaining authentic human-generated sequences is more effective for recurrent architectures than relying on synthetic interpolations.
f. Near-Miss directed to poor performance, with the GRU achieving 57% accuracy and the LSTM dropping to 51%. This reduction is directly due to the severe data decrease, leaving only 498 samples, a volume insufficient for high-capacity architectures to converge effectively. Because LSTM and GRU require thousands of diverse patterns to optimize their internal gating mechanisms and capture complex contextual dependencies, this data scarcity led to underfitting. Yet, while Near-Miss balances class distribution, fatal information loss disproportionately hinders the learning capacity of recurrent neural networks.
g. In most cases, the macro-average F1-score and weighted-average F1-score are closely aligned, indicating that both models effectively balance precision and recall across classes. This alignment suggests that the models are not overly biased toward any particular class, even in imbalanced scenarios. However, an exception is observed in the Non-Sampling configuration for the LSTM model, where the macro F1-score (0.62) slightly exceeds the weighted F1-score (0.61). This discrepancy may be attributed to minor imbalances in the class distribution, the model's capacity to generalize uniformly across all classes.

4. CONCLUSION

This study systematically evaluates the performance of GRU and LSTM models across diverse sampling techniques, providing key insights into sequential data processing and class imbalance management. The results demonstrate that the LSTM model consistently outperforms the GRU, achieving 73% accuracy compared to the GRU's 71%.The superiority of the LSTM in this context is attributed to its separate cell state, which facilitates a more stable gradient flow over longer sequences than the GRU's combined hidden state. This architectural complexity proved particularly beneficial for capturing the subtle emotional patterns and semantic shifts inherent in Indonesian tweets. While both models demonstrated the ability to learn from unsampled data, the LSTM's gating mechanisms provided a more robust framework for maintaining long-term dependencies in the face of linguistic variability. Regarding resampling strategies, Random Oversampling (ROS) was found to be the most effective method for addressing class imbalance without compromising

ISSN 2774-3659 (Media Online)
Vol 6, No 4, June 2026 | Hal 1225-1232
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i4.996

model integrity. While ROS is often viewed as a simple technique due to its reliance on data duplication, it proved superior in this text classification task because it preserves exact word sequences. Unlike SMOTE, which introduces synthetic vector noise by interpolating between embeddings often resulting in structures that lack real linguistic meaning ROS maintains the original linguistic integrity of the emotional expressions. In conclusion, this study highlights the critical relationship between architecture and preprocessing. For sequential tasks like emotion detection, LSTM is recommended for its superior pattern retention, while Random Oversampling is preferred over synthetic methods to ensure the model learns from authentic, human-generated linguistic patterns rather than mathematical interpolations.

REFERENCES

[1] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Mach. Learn. Deep Learn. Christ.*, vol. 31, pp. 685–695, 2021, doi: 10.1515/9783110791402-004.
[2] A. Mathew, P. Amudha, and S. Sivakumari, "Deep learning techniques: an overview," in *Advances in Intelligent Systems and Computing*, Springer, 2021, pp. 599–608. doi: 10.1007/978-981-15-3383-9_54.
[3] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, "A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU," *arXiv Prepr. arXiv2305.17473*, 2023, [Online]. Available: http://arxiv.org/abs/2305.17473
[4] G. Ian, Y. Bengio, and A. Courville, *Deep Learning (Adaptive Computation and Machine Learning series)*. Cambridge: The MIT Press, 2016.
[5] P. Kumar, R. Bhatnagar, K. Gaur, and A. Bhatnagar, "Classification of Imbalanced Data:Review of Methods and Applications," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1099, no. 1, p. 012077, Mar. 2021, doi: 10.1088/1757-899x/1099/1/012077.
[6] A. Amin, A. Adnan, and S. Anwar, "An adaptive learning approach for customer churn prediction in the telecommunication industry using evolutionary computation and Naïve Bayes," *Appl. Soft Comput.*, vol. 137, p. 110103, Apr. 2023, doi: 10.1016/j.asoc.2023.110103.
[7] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," in *2020 11th International Conference on Information and Communication Systems (ICICS)*, IEEE, Apr. 2020, pp. 243–248. doi: 10.1109/ICICS49469.2020.239556.
[8] T. F. Handoyo, M. Pajar, and K. Putra, "Optimasi Bobot Kelas LSTM untuk Deteksi URL Phishing pada Dataset Tidak Berimbang," *JPIT (Jurnal Penelit. Inform. dan Teknol.*, vol. 10, no. 1, pp. 20–36, 2025, doi: 10.30591/jpit.v10i1.8128.
[9] F. Thabtah, S. Hammoud, F. Kamalov, and A. Gonsalves, "Data imbalance in classification: Experimental evaluation," *Inf. Sci. (Ny).*, vol. 513, pp. 429–441, Mar. 2020, doi: 10.1016/j.ins.2019.11.004.
[10] S. Korkmaz, "Deep Learning-Based Imbalanced Data Classification for Drug Discovery," *J. Chem. Inf. Model.*, vol. 60, no. 9, pp. 4180–4190, Sep. 2020, doi: 10.1021/acs.jcim.9b01162.
[11] A. B. P. Negara, H. Muhardi, and F. Sajid, "Perbandingan Algoritma Klasifikasi terhadap Emosi Tweet Berbahasa Indonesia," *J. Edukasi dan Penelit. Inform.*, vol. 7, no. 2, p. 242, Aug. 2021, doi: 10.26418/jp.v7i2.48198.
[12] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," *arXiv Prepr. arXiv1012.2599*, 2010, [Online]. Available: http://arxiv.org/abs/1012.2599
[13] P. S. Muhuri, P. Chatterjee, X. Yuan, K. Roy, and A. Esterline, "Using a long short-term memory recurrent neural network (LSTM-RNN) to classify network attacks," *Inf.*, vol. 11, no. 5, pp. 1–21, 2020, doi: 10.3390/INFO11050243.
[14] Y. Luan and S. Lin, "Research on Text Classification Based on CNN and LSTM," *Proc. 2019 IEEE Int. Conf. Artif. Intell. Comput. Appl. ICAICA 2019*, pp. 352–355, 2019, doi: 10.1109/ICAICA.2019.8873454.
[15] S. Nosouhian, F. Nosouhian, and A. K. Khoshouei, "A review of recurrent neural network architecture for sequence learning: Comparison between LSTM and GRU," *Preprints.org*, pp. 1–7, Jul. 2021, doi: 10.20944/preprints202107.0252.v1.
[16] H. He and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*. New Jersey: John Wiley \& Sons, 2013.
[17] C. Yang, E. A. Fridgeirsson, J. A. Kors, J. M. Reps, and P. R. Rijnbeek, "Impact of random oversampling and random undersampling on the performance of prediction models developed using observational health data," *J. Big Data*, vol. 11, no. 1, 2024, doi: 10.1186/s40537-023-00857-7.
[18] F. R. A. Pratama and S. I. Oktora, "Synthetic Minority Over-sampling Technique (SMOTE) for handling imbalanced data in poverty classification," *Stat. J. IAOS*, vol. 39, no. 1, pp. 233–239, 2023, doi: https://doi.org/10.3233/SJI-220080.
[19] M. S. Shelke, P. R. Deshmukh, and V. K. Shandilya, "A Review on Imbalanced Data Handling Using Undersampling and Oversampling Technique," *Int. J. Recent Trends Eng. Res.*, vol. 3, no. 4, pp. 444–449, May 2017, doi: 10.23883/IJRTER.2017.3168.0UWXM.
[20] A. Tanimoto, S. Yamada, T. Takenouchi, M. Sugiyama, and H. Kashima, "Improving imbalanced classification using near-miss instances," *Expert Syst. Appl.*, vol. 201, no. November 2021, p. 117130, 2022, doi: 10.1016/j.eswa.2022.117130.