


## Review Article

# A Novel Hybrid Deep Learning Approach to Code Generation Aimed at Mitigating the Real-Time Network Attack in the Mobile Experiment Via GRU-LM and Word2vec

Minjong Cheon,<sup>1</sup> Hyodong Ha,<sup>1</sup> Ook Lee,<sup>1</sup> and Changbae Mun <sup>2</sup>

<sup>1</sup>Department of Information Systems, Hanyang University, 222 Wangshimni-ro Seongdong-gu, Seoul 04673, Republic of Korea

<sup>2</sup>Department of Electrical, Electronic & Communication Engineering, Hanyang Cyber University, Seoul 04764, Republic of Korea

Correspondence should be addressed to Changbae Mun; changbae@hycu.ac.kr

Received 20 December 2021; Revised 30 June 2022; Accepted 9 August 2022; Published 29 September 2022

Academic Editor: Nicola Biccocchi

Copyright © 2022 Minjong Cheon et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As the use of devices in mobile environments increases, network attacks such as DDoS have a malicious attempt to flood the network's regular traffic to overload the target and surrounding infrastructure. This research proposed machine learning and deep learning approaches to dealing with DDoS attacks, and the results are described as follows. First, this research successfully detected DDoS attacks through an LGBM with a 100% accuracy score. Second, the proposed model (GRU-LM), which consists of a trained Word2vec layer with the Python dataset, is far more effective than the standard GRU model. Since Python is quite similar to English, language model-based GRU yields superior results. Various preprocessing steps were performed through the NLTK package, and each number was assigned to the tokenized one for constructing the GRU language model. The result reveals that the proposed model achieved an accuracy score of 87% for predicting the following words in the source code, while the rest achieved below 30% accuracy. This conclusion is significant because its relatively simple and light structure overcomes tradeoff problems between time and accuracy and is adaptable to the mobile setting. Discovering traffic patterns for the underlying data of DDoS assaults and retrieving them using statistical data analysis is the value of this research. Furthermore, since public cloud application vulnerability assaults are rising due to expanding cloud infrastructure, this finding could be used in such attacks.

## 1. Introduction

As the fourth industrial revolution has come, technologies, including artificial intelligence (AI), the Internet of things (IoT), and cloud systems, have been used widely in our society. Since many of those technologies are operated in the internet-based environment, the potential damage from DDoS attacks has also increased significantly. The DDoS attack is an abbreviation of distributed denial-of-service. It is a malicious attempt to overwhelm the target and surrounding infrastructure with Internet traffic by flooding the network's regular traffic [1]. It has been a persistent threat to various industries, regardless of geographic location or target market, and it is predicted to be increased dramatically in the upcoming years [2]. The graph below shows several DDoS attacks from 2018 to 2023 and can be

interpreted as an increasing trend of DDoS attacks over the years, including the future [3]. Furthermore, according to the latest research by Kaspersky Lab, a DDoS attack could cost an enterprise over \$1.6 million, a considerable sum for any business [4]. Seven different and efficient actions exist that a company could perform to prevent a DDoS attack by preparing a DDoS plan, improving network security, ensuring server redundancy, looking for warning signs, limiting network broadcasting, using cloud-based protection, and setting up continuous monitoring [5]. Until the late 2000s, a typical form of DDoS attack was achieved using malware in the network infrastructure. To detect this form of DDoS, it is necessary to see a specific pattern in a series of packets [6, 7]. If the amount of packets used for attack detection is insufficient, it is difficult for the system to perform a detection function for DNS amplification DDoS

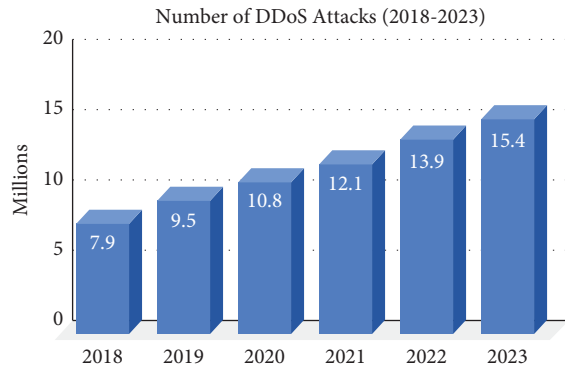


FIGURE 1: Cisco’s research and forecasting of DDoS total attack history (2018–2023) [3].

attacks. In particular, distributed reflection denial-of-service attack cannot perform structural modeling if the amount of packets used for attack detection is insufficient, so an appropriate pattern is not established. In this case, the cognitive value is not high even if a pattern is identified in reducing the data dimension. As a result, it has a characteristic that is very vulnerable to security. In addition, DDoS security issues are closely related to the network environment. In a cloud environment, an attack module such as DDoS docker is deployed on an external interlocking server to attack security devices. These attack techniques can find patterns through artificial intelligence analysis of data collected by intrusion detection systems (IDS) [8]. Large-scale DDoS attacks generate traffic much larger than the network bandwidth and occupy the line, making it impossible to defend only with DDoS equipment inside the network. Therefore, pre-emptive detection and pattern identification are essential for maintaining service stability. Artificial intelligence (AI) technologies, such as deep learning and machine learning, have proven effective and beneficial in various industries. Therefore, these technologies have been suggested as a solution for resolving DDoS attacks as they can assist in removing stubborn old networking obstacles and stimulating new network applications that make networking much more convenient. Representative examples of applications are intelligent network traffic management, inband network telemetry, and cyberattack identification and prevention. Intelligent network traffic management enables analytics in big data systems and large-area networks to discover complicated patterns. Network telemetry data gives fundamental network performance measurements although sometimes they are difficult to comprehend. Network behavior is an essential characteristic, and to detect threats, undiscovered viruses, and policy breaches, its principal to evaluate massive volumes of data in real time. Network telemetry data give fundamental network performance measurements although sometimes they are difficult to comprehend. Network telemetry data give fundamental network performance measurements although occasionally they are difficult to comprehend. Therefore, machine learning models have risen as a troubleshooter for those issues that traditional methods could not handle effectively [9].

Figure 1 shows that many cases of DDoS attacks are predicted to escalate by 2023, and people should prepare to detect or prevent them efficiently. In addition, coping with the real-time intrusion of DDoS has been a critical issue because DDoS attacks can cause a lot of damage, even for a short period. Therefore, this research aims to detect DDoS attacks, and develop a code generation model through deep learning methods, to decrease the time spent dealing with DDoS attacks. Several preprocessing methods from the natural language processing (NLP) library would be used for the analysis, and then various deep learning models would be applied to calculate the accuracy score. Word2vec was used to figure out the connection between the words in the Python corpus and as an embedding tool for constructing the model. Furthermore, considering a characteristic of the Python language, whose structure is quite similar to English, language models such as the gated recurrent unit (GRU) language model were utilized for the prediction of the next words, and usage of Word2vec and GRU language model is a distinct point from other related works. The following is a proposed contribution to this research study.

Applying this study to the web service industry can help reduce the complexity of applying security rules to logic. Cyberattacks are evolving, and its types are becoming more diverse. DDoS attacks can be executed with simple software, and protocols used for traffic attacks are also diversifying to LDAP and NTP. As cloud infrastructure expands, vulnerability attacks in public cloud applications are increasing. Since this study suggested a machine learning approach for detecting DDoS attacks and generating source code for that, it could be utilized for such attacks effectively in the future [10].

## 2. Related Works

Hung-Chi Chu and Chan-You Yan proposed a packet continuity-based model to calculate collecting consecutive packets, average time for detecting DDoS attacks, and classification scores based on the window size. The result demonstrates that the window size should be less than 50 for real-time collection, and when a scope is fixed to 10, it showed average accuracy of over 99.5% while consuming only 5 ms for computation time. This research contributes that they could gather the latest DDoS attack data with less computation time and speed for detecting DDoS attacks through the proposed model (5 ms) with high accuracy [11].

Vijayakumar and Ganapathy utilized ensemble algorithms through the random forest, SVM, and SWELL algorithms to detect DDoS attacks efficiently. The AWID dataset was used as a dataset, and the proposed algorithm achieved a 99.98% precision score [12].

Riyaz and Ganapathy suggested novel algorithms to detect intrusion. The proposed model first conducted feature selection through the conditional random field and linear correlation coefficient-based feature selection (CRF-LCFS) algorithms. The convolutional neural network (CNN) finally classified the target. These models yielded a 98.88% accuracy score for detecting intrusions [13].

Zengguang Liu and Xiochun Yin researched generating synthetic data for low-rate distributed DoS (LDDoS) attacks

via combining LSTM and CGAN. After generating fake datasets via CGAN, the authors found that the generated dataset is quite similar to the real ones through computing Euclidean distance. They compared again by conducting five different machine learning algorithms on them. The error rate, precision score, and recall scores were calculated through suggested models, and it was found that each score was very similar to the other, which verified the authenticity of the synthetic data. This research makes synthetic data to be utilized in the future for better analysis with a sufficient dataset [14].

Cruz-Benito et al. suggested a model for generating source code through various DNN designs, including AWD-LSTM, AWD-QRNN, and transformer, which were used to see which types of work are best in different tokenization models. This research discovered that tokenization with character-sized chunks performs better in tiny LM (like the AWD-LSTM and AWD-QRNN models) than in other tokenization models. Although accuracy was significantly reduced in larger models like the Transformer GPT-2, it performed better in source code generation tests [15].

Hu et al. suggested an algorithm intended to aid developers in comprehending the Java method's functions. DeepCom is seq2seq based, and a new technology is proposed in this study that automatically creates code annotations for Java language. Source codes from 9,714 Java projects from GitHub were used for training the model. Experiments are carried out on a large-scale Java corpus from GitHub's 9,714 open-source projects. The results are measured using machine translation metrics and the bilingual evaluation understudy (BLEU) score. DeepCom yielded 38.17%, while other algorithms such as CODE-NN and attention-based seq2seq showed 25.3% and 35.5%, each [16].

Chakraborty et al. suggested that character-based generation models are created using long short-term memory cell (LSTM). It can be concluded that expanding the number of LSTM cells at the start enhances the performance of the model. The increase in LSTM cells also has a consequence on the semantic association between letters, which has been linked to the issue of an extensive corpus. This problem is alleviated by decreasing the concentration of words in the corpus, and text generation becomes considerably more efficient [17].

Pang et al. proposed the application of convolutional neural networks and recurrent neural networks to the automatic generation of GUI codes. Through the network, which is the HGui2Code model as an attention-based deep neural code, the DSL code may be aligned with the relevant GUI pixel by using a hybrid attention approach. Comprehensive empirical findings indicate that the NAT model surpasses the newest techniques in online datasets, with the HGui2Code model enhancing accuracy by 5.5 percent and the SGui2Code model increasing accuracy by 1.5 percent, respectively [18].

Onan et al. performed keyword extraction for the thesis and the most-frequent method showed the best performance among various algorithms [19].

Onan conducted topic extraction in the paper's abstract and eliminated tasks such as preprocessing of data, feature

selection, and human interference in traditional machine learning by combining Word2vec, Pos2vec, word-position2vec, and LDA2vec schemes [20].

Onan and Korukoglu proposed an ensemble-applied feature selection model. Integrating the extracted lists through various filter-based techniques, such as information gain, into genetic algorithms showed superior performance over individual methodologies [21].

Onan presented a model that combines weighted GloVe word embedding based on CNN-LSTM architecture to perform sentiment analysis from Twitter's product review data, showing 93.85% accuracy and better performance than other models [22].

The related works yield that various pieces of research were performed on detecting DDoS attacks or code generation through deep learning algorithms. However, we suppose to suggest methods of combining them. We first utilize machine learning algorithms to detect attacks and construct source code for automatic code generation through deep learning algorithms. Therefore, this work could suggest a more realistic approach to responding well to DDoS attacks.

### 3. Materials and Methods

**3.1. Data Availability.** This research utilized two datasets from the Kaggle website, which are accessed through <https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv> [23], which is shown in Figure 2, and <https://www.kaggle.com/linkanjarad/coding-problems-and-solution-python-code> [24], as depicted in Figure 3. The University of New Brunswick proposed the first dataset. It contained some vital information for detecting DDOS attacks, such as DST port (destination port), protocol, flow duration, Tot Fwd Pkts (total forward packets), Tot Bwd Pkts (total backward packets), and label. The label column was selected as the target column for classifying the three different attacks. The second dataset consists of 3.3k+ coding problems and their corresponding source code in Python. These data were collected from various sources and involves codes and solutions for the following situations: (1) working with strings, lists, arrays, tuples, dictionaries, CSV, and JSON; (2) modules including NumPy, BeautifulSoup, Tkinter, Pandas, random, os, re, and DateTime; (3) file I/O; (4) loops and conditionals; (5) functions (including lambda) and classes; (6) OOP and DSA; (7) searching and sorting; and (8) pattern printing for the experiment; only source code in the dataset was utilized, as the objective of this research is to predict the next sentence from the given Python source code. The figure below shows the overall structure of the dataset.

**3.2. Word2vec.** Word2vec is a word embedding model released by Mikolov et al. in 2013, and its basic assumption is that similar words in a given sentence or text would show close distance. It mainly consists of the Continuous Bag-of-Word (CBOW) model and a skip-gram model. The architecture of both models is similar, as seen in Figure 4, and the input layer receives  $w(t-2)$ ,

DsT Port	Protocol	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	
0	0	0	112641719	3	0	0	0	0	0.000000	...	0	0.0	0.0	0	0	56320859.5	
1	0	0	112641466	3	0	0	0	0	0.000000	...	0	0.0	0.0	0	0	56320733.0	
2	0	0	112638623	3	0	0	0	0	0.000000	...	0	0.0	0.0	0	0	56319311.5	
3	22	6	6453966	15	10	1239	2273	744	0	82.600000	...	32	0.0	0.0	0	0	
4	22	6	8804066	14	11	1143	2209	744	0	81.642857	...	32	0.0	0.0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
104B570	80	6	10156986	5	5	1089	1923	587	0	217.800000	...	20	0.0	0.0	0	0	
1044571	80	6	117	2	0	0	0	0	0	0.000000	...	20	0.0	0.0	0	0	
1044572	80	6	5095331	3	1	0	0	0	0	0.000000	...	20	0.0	0.0	0	0	
1044573	80	6	5235511	3	1	0	0	0	0	0.000000	...	20	0.0	0.0	0	0	
1044574	443	6	5807256	6	4	327	145	245	0	54.500000	...	20	291569.0	0.0	291569	291569	5515650.0

FIGURE 2: Overview of the first dataset obtained from the Kaggle website used for detecting DDoS attacks [23].

Unnamed: 0	Problem	Python Code
0	Write a NumPy program to repeat elements of an...	<code>import numpy as np\rx = np.repeat(3, 4)\rprint...</code>
1	Write a Python function to create and print a ...	<code>def printValues():\n\tl = list()\n\tfor i in r...</code>
2	Write a Python program to remove duplicates fr...	<code>import itertools\rnum = [[10, 20], [40], [30, ...</code>
3	Write a NumPy program to compute the x and y c...	<code>import numpy as np\rimport matplotlib.pyplot a...</code>
4	Write a Python program to alter a given SQLite...	<code>import sqlite3\rfrom sqlite3 import Error\rdef...</code>
...	...	...
3302	Python Program to Check Whether a Number is Po...	<code>\nn=int(input("Enter number:"))\nif(n&gt;0):\n ...</code>
3303	\n\nThe Fibonacci Sequence is computed based on ...	<code>\ndef f(n):\n if n == 0: return 0\n elif...</code>
3304	\n\n\nPlease raise a RuntimeError exception.\n:	<code>\nraise RuntimeError('something wrong')\n\n\n/n</code>
3305	Program to print inverted right triangle alpha...	<code>\nprint("Enter the row and column size:");\nro...</code>
3306	Program to find the sum of series 1+X+X^2/2.....	<code>\nprint("Enter the range of number:");\nn=int(i...</code>

FIGURE 3: Overview of the second dataset obtained from the Kaggle website [24].

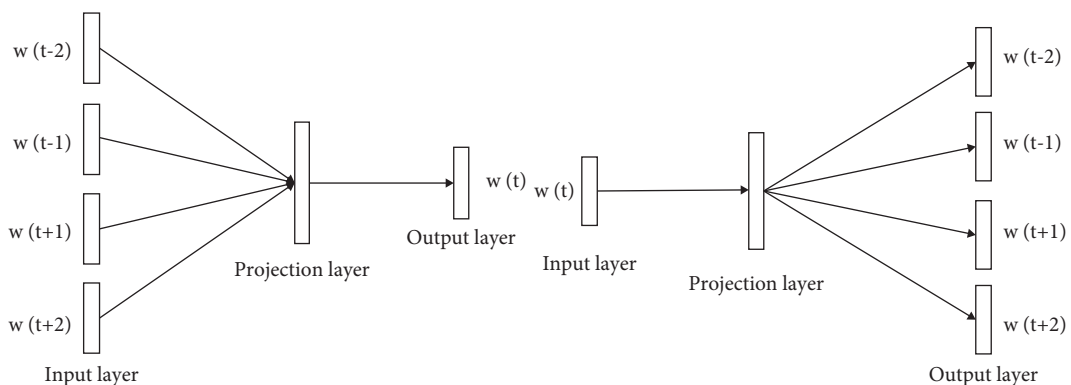


FIGURE 4: Overview of the CBOW (left) and skip-gram (right) structure [25].

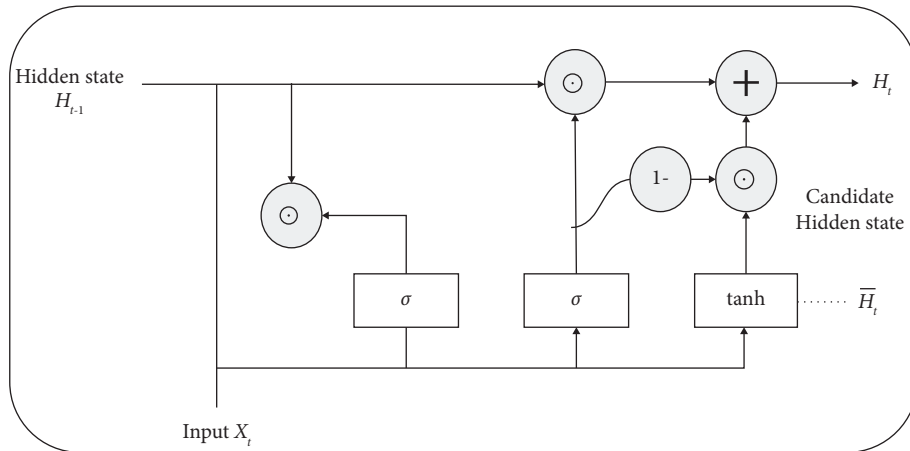


FIGURE 5: Overview of the GRU structure [28].

$w(t-1)$ ,  $w(t)$ ,  $w(t+1)$ , and  $w(t+2) \dots$ , which refers to words as the input data, and the projection layer located between the input layer and the output layer is consistent with an array of multidimensional vectors which accumulate the sum of several vectors. The output layer yields the results of the vectors from the projection layer. Even though both models are pretty similar in overall architectures, an existing principal difference is that CBOW predicts target words from original context words. In contrast, skip-gram predicts actual context words from target words inversely [25].

**3.3. Gated Recurrent Unit.** Recurrent neural network (RNN) based algorithms, including gated recurrent unit (GRU), and long short term memory (LSTM), are mainly used in NLP-related studies because of the recurrent characteristics of the RNN [26]. The RNN maintained its state by utilizing its output as input from one iteration to the next, which makes a recurrent neural network. However, the RNN has a downside of gradient vanishing, which cannot retain the older previous layers, especially in long sequences. Therefore, LSTM and GRU were introduced to resolve the vanishing gradient problem of the RNN [27]. While there are three gates in the LSTM such as a forget gate, an input gate, and an output gate, only two gates are employed in the GRU such as a reset gate and an update gate. A single hidden state is also expressed by combining the cell and the hidden states. GRU consists of only two gates, so it can spend less memory and computation time than LSTM, as described in Figure 5 [28]. A single hidden state is also expressed by combining the cell and the hidden states.

**3.4. Proposed Approach: Code Generation.** Python is one of the most popular programming languages in use nowadays, and the main reason is that it is more accessible than other programming languages for beginners. As Python has quite a similar syntax to English, beginners can approach it straightforwardly [29]. Furthermore, this English-like syntax of Python allows beginners to understand the meaning of keywords in the programming language, compared to other C-like languages, including Java and Perl [30, 31].

As the previous research shows, Python has many similar structures to other programming languages; our research decided to utilize a language model for predicting the following sentence in the source code. Therefore, the language model was used to derive a better result. Statistical language modeling aims to discover a language's joint probability function of word sequences. It provides a probability  $P$  to the whole series given such a sequence of length  $m$ . The language model includes context for distinguishing between phonetically identical words and sentences [32].

Several preprocessing procedures to obtain better results are described in Figure 6. Firstly, the Python source code was separated into each sentence once received as input data. After fragmenting the Python code, the tokenizer from the NLTK package was used. The NLTK package was invented for conducting NLP tasks through Python. Many preprocessing procedures were applied to tokenized data, including converting to lowercase and deleting words from English stop-words. Word embedding was then generated using the Word2vec function to figure out the relationship between words in the source code, which could be helpful in code creation. Finally, a gated recurrent unit-based language model (GRU-LM) was used to calculate the accuracy score and predict upcoming sentences. GRU-LM utilized a language model approach to an ordinal GRU architecture, which could impose a characteristic of the language model.

**3.5. Proposed Model: Code Generation.** The proposed model consists of multiple steps, as described in Figure 7. Firstly, the given dataset is vectorized and tokenized for preprocessing, especially for assigning each number to the tokenized one. Because the language model emphasizes a sequence of words for a better prediction, in other words, it is vital to predicting the next term when the previous sentences/words were provided. In the second step, the Word2vec model is pretrained with our dataset (from Kaggle), which could help to contain multiple features for better performance. The pretrained Word2vec model was utilized for the embedding steps, and then the GRU model

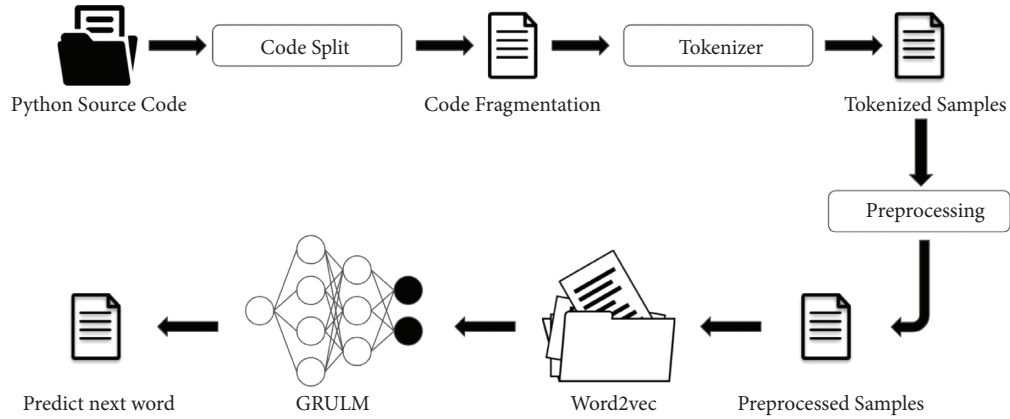


FIGURE 6: The pipeline of the experiment.

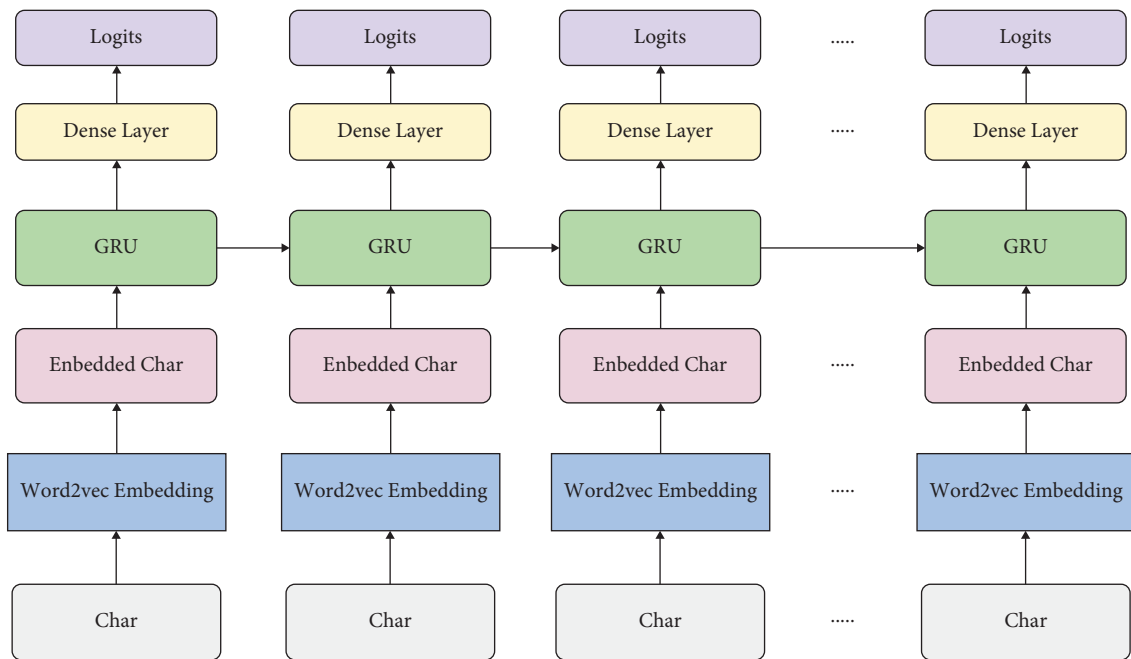


FIGURE 7: The pipeline of the proposed model: GRU-LM.

was applied to predict the labels. Then, a dense layer was added to the GRU model to construct a deeper model and predict the following words.

**3.6. Workflow of the Proposed System.** If it is determined that the system is subject to a specific network intrusion, it is first diagnosed whether the attack is a DDoS attack through a machine learning algorithm such as LGBM. Then, if it is determined to be a DDoS attack, it is possible to cope with the attack more efficiently through the code generation algorithm proposed in the paper.

## 4. Results and Discussion

**4.1. Experimental Setup.** As mentioned above, the experiment consists of two main stages. The first is to detect DDoS attacks through a machine learning approach, and the

second is about constructing an algorithm for generating source code automatically. The dataset was downloaded from the Kaggle website for the first experiment, as aforementioned in the Data Availability section. Then, preprocessing steps such as label coding were applied to the dataset. In the second experiment, since we should prove that our proposed model is superior to other deep learning models such as RNN, LSTM, or GRU, all algorithms were tested under the same conditions, such as GPU, batch size, or epochs.

**4.2. Detecting the DDoS Attacks via Machine Learning.** The light gradient boosting machine (LGBM) algorithm was applied to the dataset. Since the label column consists of three different values, multiclassification was conducted. The LGBM yielded a 100% accuracy score for the classification, which implied that detecting DDoS attacks with a machine



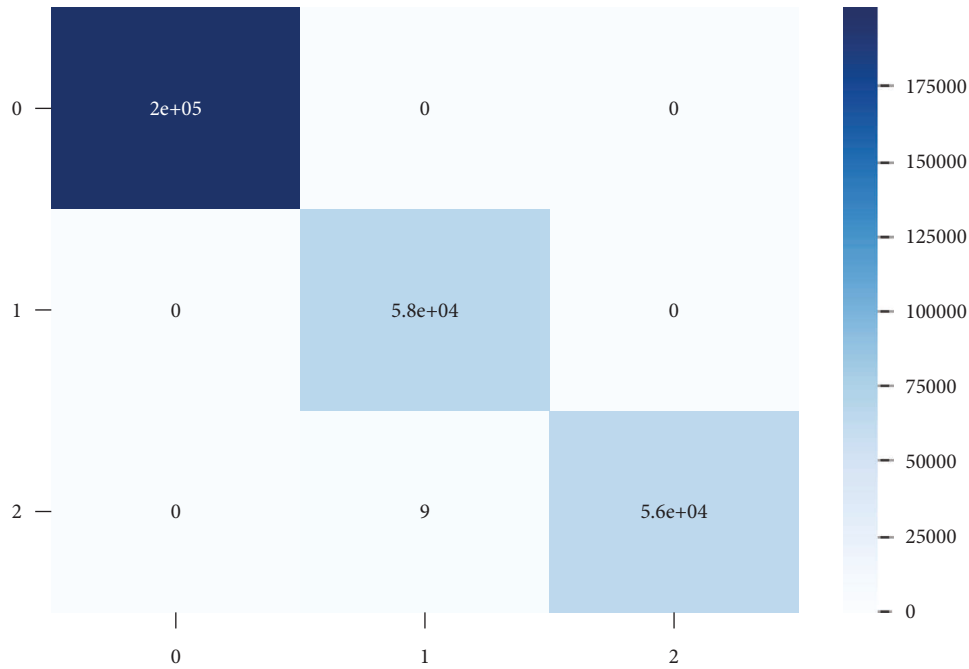


FIGURE 8: Confusion matrix of the results from the LGBM.

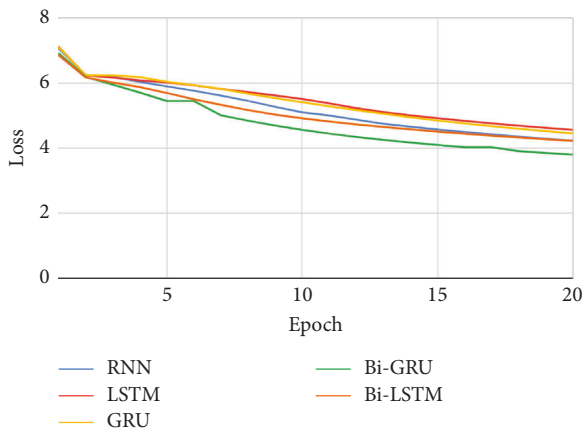


FIGURE 9: The training loss when epoch = 20.

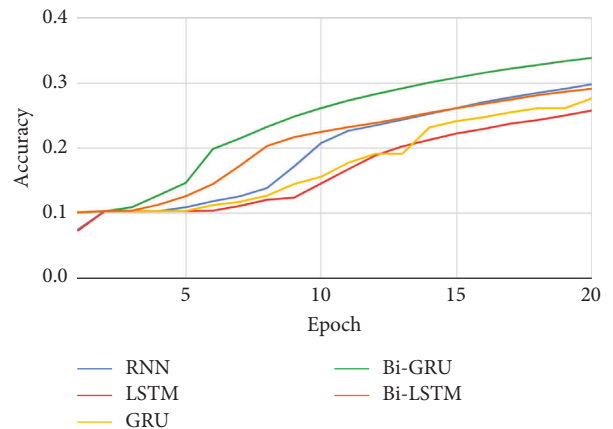


FIGURE 10: The training accuracy when epoch = 20.

learning algorithm is possible. Furthermore, it could also be concluded that if the source code could be generated automatically, we could prevent DDoS attacks efficiently. Figure 8 below yields the confusion matrix of the result from the LGBM.

**4.3. Result of Utilizing Several Ordinal Deep Learning Methods for the Dataset.** First, the preprocessed dataset was divided into train and test sets. Then, RNN, long short-term memory (LSTM), GRU, bidirectional GRU (Bi-GRU), and bidirectional LSTM (Bi-LSTM) were utilized to calculate the accuracy score. The two graphs below show how the accuracy and loss of models changed during training (epoch = 20). Loss from every model in Figure 9 exhibits decreasing trend, while accuracy from every model in Figure 10 yields an increasing trend. However, the accuracy score was far

smaller than the expectation, which was below 30% of accuracy except for Bi-GRU.

**4.4. Result of Utilizing the Proposed Approach: Word2vec + GRU-LM.** As the models mentioned, they did not achieve satisfactory results. The Word2vec model was trained with the preprocessed data, and the embedding was made through the model. Therefore, the embedding contained the relationship between the lexicons and could yield higher performance than the ordinal one hot encoding. Figure 11 shows the correlation between the tokenized words in the datasets.

Following creating the embedding layer from Word2vec, a GRU-based language model (GRU-LM) was made. Even though the model’s structure is relatively simple, with only a few layers, this model is expected to outperform the other

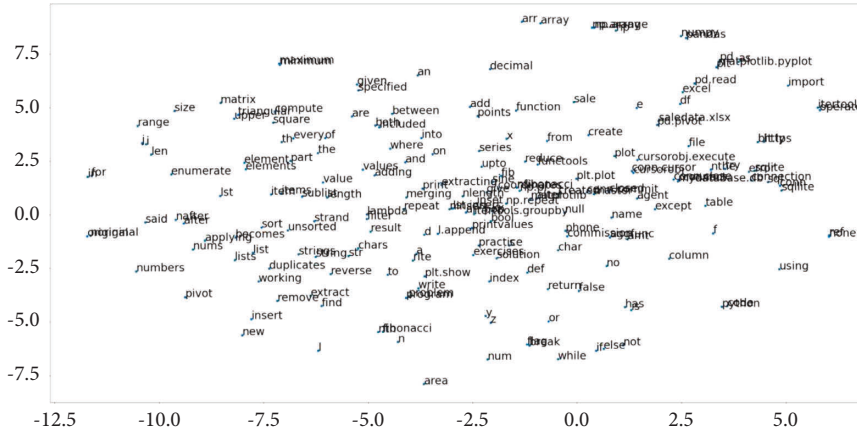


FIGURE 11: The training accuracy when epoch = 20.

	example	label
302	[[36, 38, 174, 872, 0, 0, 0, 0, 0, 0, 0, 0, ...	[[38, 174, 872, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
303	[[174, 379, 0, 0, 0, 235, 379, 38, 130, 872, 0, ...	[[379, 0, 0, 0, 235, 379, 38, 130, 872, 0, 0, ...
304	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
305	[[0, 36, 38, 174, 872, 0, 0, 130, 1476, 713, 3, ...	[[36, 38, 174, 872, 0, 0, 130, 1476, 713, 353, ...
306	[[0, 353, 52, 0, 36, 38, 130, 117, 0, 0, 0, 0, ...	[[353, 52, 0, 36, 38, 130, 117, 0, 0, 0, 0, 0, ...

FIGURE 12: Result of the proposed model with the dataset: example denotes predicted one, and label indicates the answer.

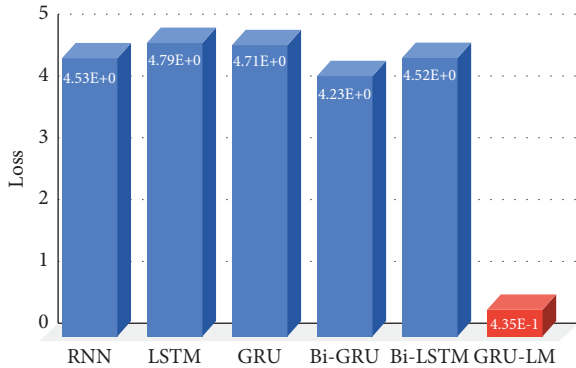


FIGURE 13: Loss results from the various deep learning models and the proposed model (test set).

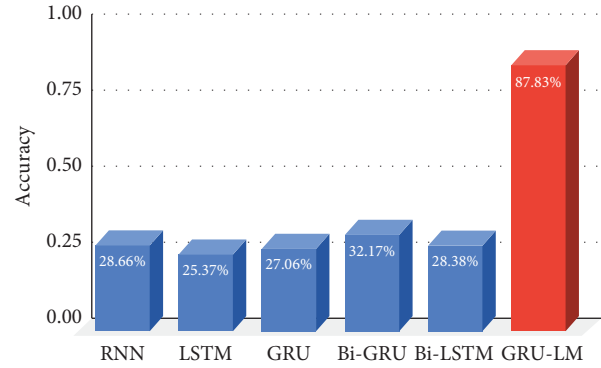


FIGURE 14: Accuracy results from the various deep learning models and the proposed model (test set).

models above due to the learned embedding layer and language model. Furthermore, using GRU-LM rather than LSTM-LM reduces computation memory and time. RNN-LM was not chosen since the RNN itself featured a gradient vanishing difficulty. Figure 12 shows a predicted output in the “example” column and the actual answer in the “label” column. The accuracy score was calculated via those two columns.

The below graphs present both loss and accuracy score, which were evaluated based on the test sets. Figure 13 shows the loss from the various models, and the proposed model, whose color is red, exhibits that its loss is far smaller than the other five ordinal deep learning algorithms. Furthermore, Figure 14 yields the accuracy score, and the proposed model

achieved 87.3%, almost three times higher than the other models except for Bi-GRU. RNN, LSTM, GRU, Bi-GRU, and Bi-LSTM yielded 28.66%, 25.37%, 27.06%, 32.17%, and 28.38%, respectively.

The two graphs below show the accuracy score and loss from the suggested model throughout the training session. It could be concluded that the training was successful based on the growing accuracy graph and lowering loss graph in Figure 15. Overfitting, a common drawback of deep learning models did not occur.

Furthermore, an experiment on speed comparison was also conducted. All of the models were performed under the same condition, with the batch size for 1024 and 20 epochs per model. The result in Figure 16 showed that the RNN



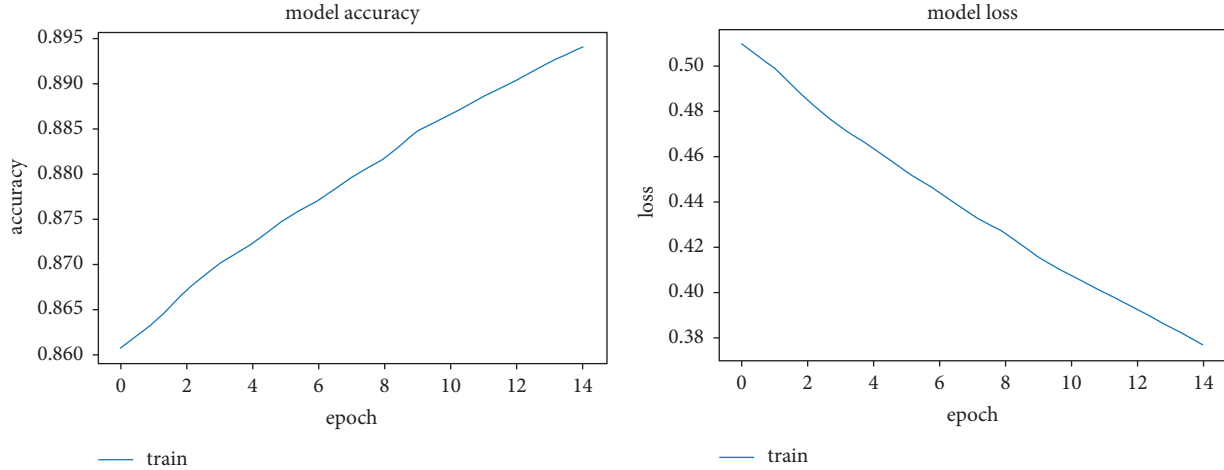


FIGURE 15: The training accuracy and loss from the proposed model during epoch = 15.

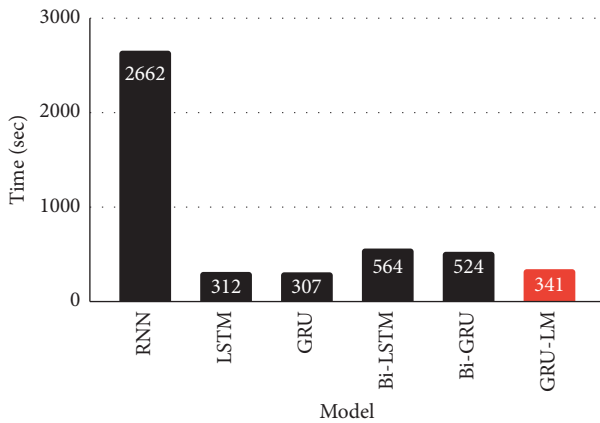


FIGURE 16: Speed comparison of each algorithm and our proposed model is marked as red (epoch = 20).

model consumed the longest time for the training, which was 2662 seconds, and the fastest model was GRU with 307 seconds. The proposed model required 341 seconds. Even though the proposed algorithm was not the fastest among different models, since the accuracy was far higher than others, which could be found in the figure above, it could be concluded that the proposed algorithm could be efficiently applied to the real world.

### 5. Discussion

The methodology of this work shows that malicious traffic patterns can be compared to characteristic forms of historical traffic to detect DDoS attacks in advance. In particular, when detecting attacks occurring at the application layer, the system can leverage historical traffic information to select resources to protect first and find a point in time to apply complementary logic automatically. In addition, using this technique in both cloud and on-premise environments can extend the resilience and stability of the system by using DDoS defense capabilities.

According to the experiment results, the LGBM-based detection intrusion detection model showed a possibility of

applying it to the real world since it yielded a 100% accuracy score. Furthermore, the suggested model in the second experiment, which comprises trained Word2vec and language model-based GRU, outperformed the other deep learning models. Aside from the excellent accuracy score, our suggested model has two additional benefits such as shorter calculation time and memory use since the number of parameters that should be calculated is less than other deep learning models. It is possible to estimate that because Python is comparable to English and language model-based GRU was highly efficient. Furthermore, by creating the Word2vec model, which had already been trained with the supplied dataset, we could save time and memory while achieving improved accuracy, which could be effective in the mobile environment. This finding could bring significant benefits in dealing with DDoS attacks as reducing time for correction is a vital issue.

### 6. Conclusions

Several deep learning algorithms were examined for efficient code creation to deal with real-time DDoS attacks. Several preprocessing processes were done to the supplied dataset, which may aid deep learning models in producing more exact results. The proposed model comprises the Word2vec embedding layer pretrained with the provided dataset and GRU-LM. Except for Bi-GRU, the accuracy score was obtained from them to evaluate each model’s performance. The suggested model achieved 87.3%, nearly three times higher than the other models. RNN, LSTM, GRU, Bi-GRU, and Bi-LSTM produced results of 28.66%, 25.37%, 27.06%, 32.17%, and 28.38%, respectively. Considering the outcomes, it is possible to infer that employing both the Word2vec embedding layer and GRU-LM is far more efficient than conventional techniques owing to the structure of Python being quite similar to that of English. This finding is meaningful that there is no tradeoff problem between time and accuracy, and it is worthwhile when correcting a code for defense in a mobile environment. Our proposed model and the two pretrained models, BERT and GPT, will be used

for code creation in future work with other programming languages such as Java. The value of this research lies in identifying traffic patterns for the underlying data of DDoS attacks and accessing them through statistical data analysis.

## Data Availability

This research utilized a dataset from the Kaggle website, which is accessed through <https://www.kaggle.com/linkanjarad/coding-problems-and-solution-python-code> [11]. The given dataset consists of 3.3k+ coding problems and their corresponding source code in Python language. These data were collected from various sources and involve codes and solutions for situations.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

Basic Science Research Program supported this research through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (2020R1G1A1006677).

## References

- [1] R. V. Deshmukh and K. K. Devadkar, "Understanding DDoS Attack & its Effect in Cloud Environment," *Procedia Computer Science*, vol. 49, pp. 202–210, 2015.
- [2] D. Warburton, *DDoS Attack Trends for 2020*, F5 Labs, Washington, DC, USA, 2021, <https://www.f5.com/labs/articles/threat-intelligence/ddos-attack-trends-for-2020>.
- [3] P. Nicholson, *Five Most Famous DDoS Attacks and Then Some*, A10 Networks, San Jose, CA, USA, 2020, <https://www.a10networks.com/blog/5-most-famous-ddos-attacks>.
- [4] D. Kaspersky, *Denial of Service: Anatomy and Impact of DDoS Attacks*, USA.Kaspersky.Com, Moscow, Russia, 2022, <https://usa.kaspersky.com/resource-center/preemptive-safety/how-does-ddos-attack-work>.
- [5] V. Andreja, *How to Prevent DDoS Attacks: 7 Tried-And-Tested Methods*, PhoenixNAP, Phoenix, AZ, USA, 2021, <https://phoenixnap.com/blog/prevent-ddos-attacks>.
- [6] M. Suresh and R. Anitha, "Evaluating machine learning algorithms for detecting DDoS attacks," in *Advances in Network Security and Applications*, pp. 441–452, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [7] J. Wu, X. Wang, X. Lee, and B. Yan, "Detecting DDoS attack towards DNS server using a neural network classifier," in *Artificial Neural Networks - ICANN 2010*, pp. 118–123, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [8] A. M. Lonea, D. E. Popescu, and H. Tianfield, "Detecting DDoS attacks in cloud computing environment," *International Journal of Computers, Communications & Control*, vol. 8, no. 1, pp. 70–78, 2012.
- [9] K. Rastey, *Real world application of machine learning in networking*, IoT for all, Woodbine, MD, USA, 2021, <https://www.iotforall.com/real-world-application-of-machine-learning-in-networking-2>.
- [10] ALLOT, *DDoS Glossary: Common DDoS Attack Types You Should Know*, ALLOT, Hod Hasharon, Israel, 2022, <https://www.allot.com/ddos-attack-glossary/>.
- [11] H. C. Chu and C. Y. Yan, "DDoS Attack Detection with Packet Continuity Based on LSTM Model," in *Proceedings of the 2021 IEEE 3rd Eurasia Conference on IOT Communication and Engineering (ECICE)*, Yunlin, Taiwan, October 2021.
- [12] D. S. Vijayakumar and S. Ganapathy, "Multistage ensemble classifier for wireless intrusion detection system," *Wireless Personal Communications*, vol. 122, no. 1, pp. 645–668, 2021.
- [13] B. Riyaz and S. Ganapathy, "A deep learning approach for effective intrusion detection in wireless networks using CNN," *Soft Computing*, vol. 24, no. 22, Article ID 17265, 2020.
- [14] Z. Liu and X. Yin, "LSTM-CGAN: Towards generating low-rate ddos adversarial samples for blockchain-based wireless network detection models," *IEEE Access*, vol. 9, pp. 22616–22625, 2021.
- [15] J. Cruz-Benito, S. Vishwakarma, F. Martin-Fernandez, and I. Faro, "Automated source code generation and auto-completion using deep learning: comparing and discussing current language model-related approaches," *A&I*, vol. 2, no. 1, pp. 1–16, 2021.
- [16] X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, "Deep code comment generation," in *Proceedings of the 2018 IEEE/ACM 26th International Conference on Program Comprehension*, May 2018, Article ID 20010.
- [17] S. Chakraborty, J. Banik, S. Addhya, and D. Chatterjee, "Study of dependency on number of LSTM units for character based text generation models," in *Proceedings of the 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, Gunupur, India, March 2020.
- [18] X. Pang, Y. Zhou, P. Li, W. Lin, W. Wu, and J. Z. Wang, "A novel syntax-aware automatic graphics code generation with attention-based deep neural network," *Journal of Network and Computer Applications*, vol. 161, Article ID 102636, 2020.
- [19] A. Onan, S. Korukoğlu, and H. Bulut, "Ensemble of keyword extraction methods and classifiers in text classification," *Expert Systems with Applications*, vol. 57, pp. 232–247, 2016.
- [20] A. Onan, "Two-Stage topic extraction model for bibliometric data analysis based on word embeddings and clustering," *IEEE Access*, vol. 7, Article ID 145614, 2019.
- [21] A. Onan and S. Korukoğlu, "A feature selection model based on genetic rank aggregation for text sentiment classification," *Journal of Information Science*, vol. 43, no. 1, pp. 25–38, 2016.
- [22] A. Onan, "Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 23, 2020.
- [23] N. D. solarmainframe, *IDS 2018 intrusion csvs (CSE-CIC-IDS2018)*San Francisco, CA, USA, Article ID Kaggle., 2022.
- [24] Kaggle, *Natural language to Python code*, Kaggle, San Francisco, CA, USA, 2021, <https://www.kaggle.com/linkanjarad/coding-problems-and-solution-python-code>.
- [25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013, <https://arxiv.org/abs/1301.3781>.
- [26] O. Lee, H. Joo, H. Choi, and M. Cheon, "Proposing an integrated approach to analyzing ESG data via machine learning and deep learning algorithms," *Sustainability*, vol. 14, no. 14, p. 8745, 2022.
- [27] D. Wei, B. Wang, G. Lin et al., "Research on unstructured text data mining and fault classification based on RNN-LSTM with malfunction inspection report," *Energies*, vol. 10, no. 3, p. 406, 2017.
- [28] S. Yang, X. Yu, and Y. Zhou, "LSTM and GRU neural network performance comparison study: Taking yelp review dataset as an example," in *Proceedings of the 2020 International*

*Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, Shanghai, China, 2020, June.

- [29] F. Zehra, M. Javed, D. Khan, and M. Pasha, “Comparative Analysis of C++ and Python in Terms of Memory and Time,” *Comparative Analysis of C++ and Python in Terms of Memory and Time*, 2020.
- [30] A. Stefik and S. Siebert, “An empirical investigation into programming language syntax,” *ACM Transactions on Computing Education*, vol. 13, no. 4, pp. 1–40, 2013.
- [31] V. T. Norman and J. C. Adams, “Improving non-CS major performance in CS1,” in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, February 2015.
- [32] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.