Hyperparameter Setting of LSTM-based Language Model using Grey Wolf Optimizer

Bilal Zahran Aufa
School of Computing
Telkom University
Bandung, Indonesia
bilalzahran@student.telkomuniversity.ac.id

Suyanto Suyanto
School of Computing
Telkom University
Bandung, Indonesia
suyanto@telkomuniversity.ac.id

Anditya Arifianto
School of Computing
Telkom University
Bandung, Indonesia
anditya@telkomuniversity.ac.id

***Abstract*— Hyperparameters is one of the most essential part of deep learning, because they can give big impact to the performance of the model. Recent works show that if the hyperparameters of a Long Short-Term Memory (LSTM) are carefully adjusted or optimized, its performance can achieve the same performance as the more complex LSTM model. Previously, several methods such as grid search and random search is introduced to solve this hyperparameters optimization problem, but it is still not effective enough. Hence, it opens opportunities for meta-heuristic nature-inspired approach like Swarm Intelligence (SI) method to solve this hyperparameter optimization problem. The main advantage of this method is the behaviour of the algorithm that has exploring-exploiting process in order to find the global optima solution in the search space. Algorithm such as Grey Wolf Optimizer (GWO) are one of the SI algorithms that have a promising performance in optimization problem in various field. The algorithm has balanced exploring-exploiting process, that can make the optimization are more effective. Therefore, in this paper the GWO is exploited to optimize the LSTM hyperparameters for a language modeling task. Evaluation for the Penn Tree Bank dataset shows that GWO is capable of giving an optimum hyperparameters of the LSTM.**

Keywords—Grey Wolf Optimizer, language modeling, Long Short-Term Memory

I. INTRODUCTION

Long Short Term Memory (LSTM) is a special kind of Recurrent Neural Network (RNN), which is one of many architectures of Neural Network (NN). It was first introduced by Hochreiter & Schmidhuber in 1997 [1] in order to tackle the problem of Long-Term Dependencies that happens in RNN [2]. Many researchers have been implemented the model to solve various tasks such as language modeling [3], image captioning [4], [5] and language translation [6] with excellent results.

Just like any other machine learning and deep learning models, LSTM has hyperparameters that can be optimized so that the model has a good performance. Some examples of these hyperparameters are the number of hidden units in the recurrent layers, dropout value, and learning rate value. Each of these hyperparameters can have a significant impact on LSTM performance. Several previous studies have shown that LSTM with well-tuned hyperparameters will produce an LSTM model that can compete with LSTM that has more complex architecture [7] [8].

However, the process of optimizing hyperparameters is not an easy task; one of the reasons is the need for a deep understanding and strong intuition about the underlying process of the model. Besides, it can be such a tedious and error-prone task with many episodes of trial and error until a good set of hyperparameters combination can be found, which consider as a barrier to the users of the model. Therefore, the need for a framework that can automate this hyperparameters optimization is a way that can be done in order to remove this barrier. In fact, there is a field called Automated Machine Learning (AutoML) [9] that researchers have a focus on automating all the streamline of machine learning processes.

There is two most common method to solve this automatic hyperparameters optimization: grid search and random search [10]. The grid search is basically a list of possible hyperparameters values that we want to evaluate. However, this method is not effective for optimizing high dimensional hyperparameters space because of the combinatorial complexity that needs to manage, and the behavior of this technique is an exhaustive search which we need to try all of the possibilities of hyperparameters combination set.

On the other hand, the random search method can perform more efficiently than the grid search because it can sample the important dimension of the search space. Nevertheless, it has drawbacks which the method cannot take previous evaluation result into account for the next iteration.

Therefore, based on the method described earlier, there is an alternative approach called nature-inspired meta-heuristic algorithms such as genetic algorithm (GA) [11], [12], [13], [14] and Swarm Intelligence [15], [16], [17], [18], [19], [20], [21], [22]. These method has been widely used to solve optimization problems with promising results, including in the field of machine learning and deep learning.

In this paper, we will use one of the SI Algorithm called GWO [23]. This algorithm tries to mimic the behavior of the grey wolf in searching for their prey in nature. In terms of our optimization, GWO will try to find the best possible combination of hyperparameters to produce the LSTM model with good performance in the Language Modeling task. The quality of the solution and the performance of the model produced by GWO were tested against the Penn Treebank dataset that has been preprocessed before [24].

Language Modelling is considered one of the crucial tasks in Natural Language Processing (NLP) because it is the core of other NLP tasks such as machine translation [25]. The language model task is predicting the next possible word from any given sentence. Other than that, we choose the language modelling task because of the previous study that was trying to optimize the hyperparameters for LSTM in this task. The study gives us motivation to do the optimization for the hyperparameters using swarm intelligence-based algorithm.

In general, this paper shows that Grey Wolf Optimizer is an algorithm that can tackle an optimization problem with a promising result because of the capability of the algorithm that can exploit areas in search space very well. Also specifically, this paper shows alternative method like swarm intelligence based algorithm such as GWO can optimize hyperparameters

978-1-7281-8235-3/20/$31.00 ©2020 IEEE

of LSTM in Language Modelling task and the result is promising in term of perplexity and the size of the model; and shows the result of the previous study that said if the hyperparameters of the model are optimized, the model can produce a good result is correct.

This paper is further divided into four parts. Section II discusses the previous researches. In Section III, the proposed method will be described in order to tackle the hyperparameters optimization problem of LSTM. Section IV will describe the experimental setup and the results. The conclusion of this research will be described in Section V.

II. Related Work

This publication is motivated by studies [7], which show that the selection of appropriate hyperparameters in LSTM can have a very big impact on model performance. Moreover, the model almost has the same or even exceeds the performance of models with more complicated architectures.

An attempt to optimize the hyperparameters of LSTM has also been carried out in studies [8]. In this study, an evaluation of a large number of hyperparameters is carried out in order to find the most impactful effect on the model. Same as [7], the studies show that properly-tuned LSTM achieves state-of-the-art results on a word-based language model.

As for the Grey Wolf Optimizer itself, it has been used to optimize the NN model [26]. However, instead of optimizing the hyperparameters, the author used the algorithm to train the model, which is finding the best value of weights and bias. The study shows that GWO can give a competitive result compared with any other algorithm such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) in terms of avoiding the local optima.

From the results of these studies, which shown the importance of hyperparameters for the LSTM performance and the promising performance of Grey Wolf Optimizer as an algorithm to solve optimization problems, we will use GWO as an optimizer of LSTM hyperparameters. Noteworthy that this study will analyze the performance of GWO in hyperparameters optimization tasks.

III. Research Method

To achieve our primary goal, GWO is proposed to optimize hyperparameter. There are four hyperparameters of LSTM to be optimized: the number of neurons in the recurrent layer, the recurrent dropout value, the output dropout, and the embedding dropout.

Every individual in GWO will be represented as a real-valued vector. Every vector will consist of four-element, which is the hyperparameters we will optimize. Every element in the vector will have an interval from [0,1] and will be mapped to the function corresponding with the actual value of the hyperparameters. Individual representations can be described as

$$x_i = (x_{i,0}, x_{i,1}, \dots, x_{i,j}) \quad (1)$$

where x_i denoted an individual in GWO, and $x_{i,j}$ are the elements of the vector. The function that mapped the real-valued vector to the actual value of hyperparameter can be described as

TABLE I. Lower and upper bounds of hyperparameter

Hyperparameters	Value
Number of Neurons in each Recurrent Layer	[300, 800]
Embedding Dropout, Recurrent Dropout, Output Dropout	[0.3, 0.6]

$$f(x_{i,j}) = x_{i,j} \times (ub_j - lb_j) + lb_j \quad (2)$$

where $x_{i,j}$ are the real-value from [0,1], ub_j is upper-bound of hyperparameter j, and lb_j is lower-bound of hyperparameter j. The upper-bound and the lower-bound values of the actual hyperparameters is described in Table 1.

As in every SI or metaheuristic nature-inspired algorithm, the fitness function is one of the most crucial components. For this problem, the fitness function will be the perplexity of the model that has been trained by Adam Optimizer [27].

Fig. 1. Show the process of hyperparameters optimization of LSTM using GWO. It may be seen that the GWO will provide the hyperparameters for the model, and the model will return the perplexity as a fitness function for GWO. Then GWO will try to find the best possible hyperparameters combination from the given perplexity based on samples of dataset.

A. Overview of Grey Wolf Optimizer

First introduced by *Mirjalili et al.* [23]. It is a metaheuristic nature-inspired algorithm that mimics how grey wolves are hunting their prey. In nature, the grey wolves have a social hierarchy consisting of four levels of an individual. They are *alpha* (α), *beta* (β), *delta* (d), and the lowest individual in the hierarchy are the omega (**Ω**). In terms of optimization, the *alpha* will be the best fitness in the search space, the *beta* will be the second-best, and finally, the *delta* will be the third-best. For the other individual, the *omega*, it will be the 'scout' in search space to find other possible global optima.

There are two steps of a process of grey wolves to hunt their prey, the first one is encircling the prey, and the second one is hunting the prey. In order to describe this process in the mathematical model, we can see (3) and (4) for the encircling process and (10) and (11) for the hunting process,

$$\vec{D} = \left| C \cdot X_p(t) - X(t) \right| \quad (3)$$

where t indicates the current iteration $\vec{A}$ and $\vec{C}$ is the coefficient vector, $\vec{X}_p$ is the vector position of the prey and $\vec{X}$ is the grey wolf vector position

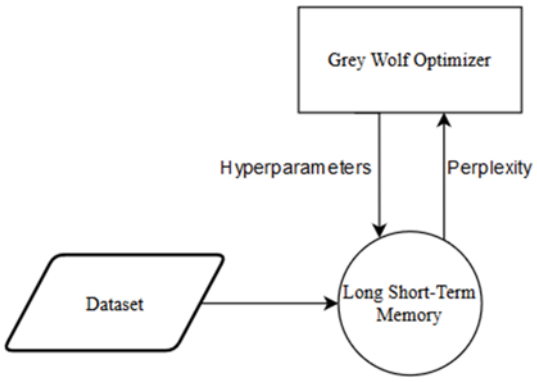

Fig. 1. The LSTM model will receive hyperparameters from GWO, evaluate on dataset samples and return the perplexity.

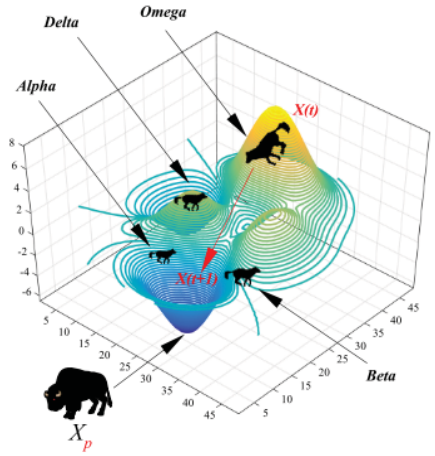

Fig. 2. Alpha, Beta Delta, and Omega definition on GWO [28].

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D} \quad (4)$$

The $\vec{A}$ and $\vec{C}$ can be described by (5) and (6) below:

$$\vec{A} = 2\vec{a} \cdot r_1 - a \quad (5)$$

$$\vec{C} = \vec{r}_2 \cdot 2 \quad (6)$$

where the $\vec{a}$ will linearly be decreased from 2 to 0 and the $\vec{r}_1$ and $\vec{r}_2$ will be a random number between [0,1]. As for the hunting process, the formula can be seen as follows:

$$X(t+1) = \frac{(\vec{X}1 + \vec{X}2 + \vec{X}3)}{3} \quad (7)$$

with $\vec{X}1$, $\vec{X}2$, and $\vec{X}3$ are calculated as

$$\vec{X}1 = \vec{X}_\alpha(t) - \vec{A}_1 \cdot \vec{D}_\alpha \quad (8)$$

$$\vec{X}2 = \vec{X}_\beta(t) - \vec{A}_2 \cdot \vec{D}_\beta \quad (9)$$

$$\vec{X}3 = \vec{X}_\delta(t) - \vec{A}_3 \cdot \vec{D}_\delta \quad (8)$$

For $\vec{D}_\alpha$, $\vec{D}_\beta$, and $\vec{D}_\delta$ are formulated as

$$\vec{D}_\alpha = \left| \vec{C}_1 \cdot \vec{X}_\alpha - \vec{X} \right| \quad (11)$$

$$\vec{D}_\beta = \left| \vec{C}_2 \cdot \vec{X}_\beta - \vec{X} \right| \quad (12)$$

$$\vec{D}_\delta = \left| \vec{C}_3 \cdot \vec{X}_\delta - \vec{X} \right| \quad (13)$$

All equations of the Grey Wolf Optimizer are summarized in Algorithm 1. The X_α is the best combination of hyperparameters set.

Algorithm 1. Grey Wolf Optimizer Algorithm

```
Initialize the population of Grey Wolf
Initialize a, A, and C value
Calculate the fitness of the population
Assign the first three best search agent to Xα, Xβ, and Xδ
While (t < Max Number of Iteration)
    For each search agent
        Update every position of search agent using (7)
    End For
    Update a, A, and C Value
    Calculate fitness of all search agent
    Update Xα, Xβ, and Xδ
    t = t + 1
End While
Return Xα
```

IV. Result and Discussion

A. Experimental Setup

The model will be trained and tested with the Penn Tree Bank dataset. A Word-based corpus consisting of 10k vocabulary words by Marcus et al. [29] that have been preprocessed by Mikolov et al. [24]. The dataset is divided into training, validation, and test sample with 929k, 73k, and 82k words, respectively. However, for the sake of computational cost, the dataset will be reduced into only 25% from the actual size by randomly picking the samples to find the optimal combination of hyperparameters. After the optimal combination of hyperparameters has been found, we use all of the data to testing the performance of the model.

In the hyperparameters optimization phase, 20 epochs are used to train the model with 20 learning rate values and will be annealing if there is no improvement for the validation samples. Besides, a batch size of 20 and two recurrent layers for the LSTM is used here.

As for the GWO, we will try to do two different sizes of population sizes. The first one will run with a population of 8, and the second will run with a population of 15. Population size is a parameter that can control the quality of the solution produced by every swarm intelligence algorithm. Therefore, we try to employ different population sizes in this work. Other than that, we choose not a very big population size because of limited computational resources.

Both experiments will run until 20 iterations. Other than that, we set the value of to 2 and will be decreasing to 0 over the iterations, and value of to a random number in the interval [2,0] for every search agent. The summary of the control parameters for the experiment is shown in Table 2.

As mentioned before, we try to optimize the value of recurrent dropout, output dropout, and embedding dropout. These dropouts usually have the same values for training the model. But, in this experiment, the various values of those dropout are performed to get the optimum ones.

TABLE II. CONTROL PARAMETERS FOR EXPERIMENT

Control Parameter Of GWO	
Parameter	**Value**
Population Size	8 & 15
Iterations	20
Control Parameters *a* (Initial Value)	2
Control Parameters *C*	[2,0]
LSTM Training Parameter	
Parameter	**Value**
Optimizer	Adam
Learning Rate	20
Number Of Layers	2
Batch Size	20

B. Result

The experimental results show that GWO with 15 search agents produces better performance than GWO with eight search agents. It can be seen from the perplexity value in the model with the best combination of hyperparameters in the first iteration. In the first iteration, the GWO with 15 agents has score 145 in terms of perplexity. However, GWO with eight agents gives a perplexity score of 162. The full performance of GWO along 20 iterations can be seen in Fig. 3 and Fig. 4, respectively.

From the graphic, we can see either GWO with 8 and 15 agents can exploit the global optima of the search space. For example, in $N = 8$ at iterations 3-5, we got the same performance at the perplexity of 159. However, in the next iterations, the performance decreases into perplexity of 157 and keep going down until the last iteration. It shows that GWO can exploit the global optima of the search space.

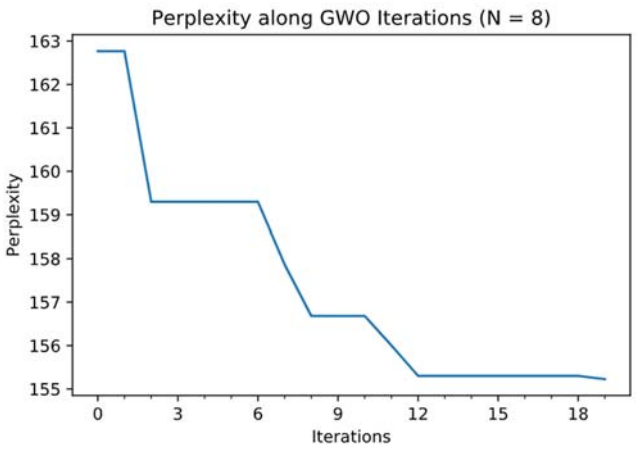

Fig. 3. Performance of GWO with 8 search agents. Lower are better.

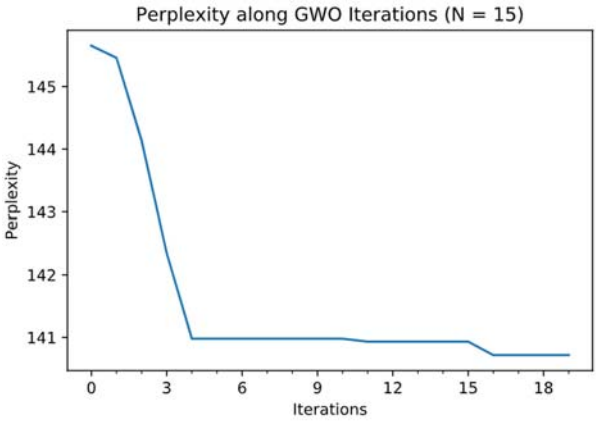

Fig. 4. Performance of GWO with 15 search agents. Lower are better.

TABLE III. MODEL PERFORMANCE USING BOTH STANDARD AND GWO-OPTIMIZED HYPERPARAMETERS

HP	**Hidden Units**	**Embedding Dropout**	**Output Dropout**	**Recurrent Dropout**	**Test PPL**
Standard	200	0.5	0.5	0.5	105.47
GWO, $N = 8$	398	0.346	0.6	0.562	**78.87**
GWO, $N = 15$	373	0.362	0.541	0.595	84.76

After we get the best hyperparameters from GWO, we train the LSTM model with 100% of dataset samples and run for 150 epochs. Then, we test hyperparameters that not optimized with GWO to see the performance. The results are described in Table 3.

When training with a full dataset, the perplexity of the LSTM Language Model gets a lower result. It can be a sign that GWO can adapt to the datasets and will try to find the best possible solution. Of course, using more computation resources, and an advanced data sampling technique, the computational cost of hyperparameters optimizations can be reduced.

These hyperparameters of LSTM that are configured by GWO produce a relatively small model, but its performance is better than the model in [30]. In that study, the model considered as a small model with 200 hidden units at the recurrent layer and with 0.7 dropout value has perplexity between 82.7 – 87.3. As for the model that produced by GWO has 398 hidden units and 0.348, 0.6, 0.562 for Embedding dropout, Output Dropout, and Recurrent Dropout has 78.87 perplexity.

Moreover, if we compare the model with optimized hyperparameters with the baseline model that we used, it shows a considerable improvement to the model performance. It proves that the optimized hyperparameters give a better performance for the model, and the GWO can optimize these hyperparameters well.

V. CONCLUSION

In this research, the GWO has been successfully applied for optimizing the hyperparameters of LSTM for a language modeling task. The experiment shows that GWO gives a promising performance to find the global optima of the search space. Once, the optimum hyperparameters are found, the LSTM is trained using 100% samples of the dataset for 100

epochs. It shows that GWO gives a promising solution for hyperparameters optimizations. Furthermore, the GWO can be optimized by adjusting both population size and iteration.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] Y. Bengio, P. Frasconi, and P. Simard, "Problem of learning long-term dependencies in recurrent networks," *1993 IEEE Int. Conf. Neural Networks*, no. February, pp. 1183–1188, 1993, doi: 10.1109/icnn.1993.298725.

[3] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," *13th Annu. Conf. Int. Speech Commun. Assoc. 2012, INTERSPEECH 2012*, vol. 1, pp. 194–197, 2012.

[4] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 3156–3164, 2015, doi: 10.1109/CVPR.2015.7298935.

[5] A. A. Nugraha, A. Arifianto, and Suyanto, "Generating Image Description on Indonesian Language using Convolutional Neural Network and Gated Recurrent Unit," in *2019 7th International Conference on Information and Communication Technology (ICoICT)*, Jul. 2019, pp. 1–6, doi: 10.1109/ICoICT.2019.8835370.

[6] Q. V. Le Ilya Sutskever Oriol Vinyals, "Sequence to Sequence Learning with Neural Networks," *Neural Inf. Process. Syst. Proc.*, 2014.

[7] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, pp. 1–10, 2018.

[8] S. Merity, N. Shirish, and K. Richard, "An Analysis of Neural Language Modeling at Multiple Scales," *CoRR*, 2018.

[9] F. Hutter, *Automated Machine Learning*. 2019.

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[11] A. Thengade and R. Dondal, "Genetic Algorithm – Survey Paper," *MPGI Natl. Multi Conf. Int. J. Comput. Appl.*, no. March, pp. 975–8887, 2012.

[12] Suyanto, *An informed genetic algorithm for university course and student timetabling problems*, vol. 6114 LNAI, no. PART 2. 2010, doi: https://doi.org/10.1007/978-3-642-13232-2_28

[13] R. M. Karim and S. Suyanto, "Optimizing Parameters of Automatic Speech Segmentation into Syllable Units," *Int. J. Intell. Syst. Appl.*, vol. 11, no. 5, pp. 9–17, 2019, doi: 10.5815/ijisa.2019.05.02.

[14] N. R. Emillia, Suyanto, and W. Maharani, "Isolated word recognition using ergodic hidden markov models and genetic algorithm," *Telkomnika*, vol. 10, no. 1, pp. 129–136, 2012, doi: 10.12928/telkomnika.v10i1.769.

[15] H. R. Ahmed and J. I. Glasgow, "Swarm Intelligence: Concepts, Models and Applications," *Queen's Univ. Sch. Comput. Tech. Reports*, vol. 585, no. February, pp. 1–50, 2012, doi: 10.13140/2.1.1320.2568.

[16] A. P. Pertiwi and Suyanto, "Globally Evolved Dynamic Bee Colony Optimization," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems,12-14 September 2011, Published in Part I, LNAI vol. 6881, pp. 52-61, Spriger-Verlag Berlin Heidelberg 2011, Print ISBN: 978-3-642-23*, 2011, no. 1, pp. 52–61, doi: https://doi.org/10.1007/978-3-642-23851-2_6.

[17] V. Clarissa and S. Suyanto, "New Reward-Based Movement to Improve Globally-Evolved BCO in Nurse Rostering Problem," in *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Dec. 2019, pp. 114–117, doi: 10.1109/ISRITI48646.2019.9034669.

[18] F. Ghaisani and S. Suyanto, "Discrete Firefly Algorithm for an Examination Timetabling," in *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Dec. 2019, pp. 1–4, doi: 10.1109/ISRITI48646.2019.9034668.

[19] U. Abdillah and S. Suyanto, "Clustering Nodes and Discretizing Movement to Increase the Effectiveness of HEFA for a CVRP," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 4, pp. 774–779, 2020, doi: https://dx.doi.org/10.14569/IJACSA.2020.01104100.

[20] G. K. Jati and S. Suyanto, "Evolutionary Discrete Firefly Algorithm for Travelling Salesman Problem," in *Adaptive and Intelligent Systems: Second International Conference, ICAIS 2011, Klagenfurt, Austria, September 6-8, 2011*, 2011, no. 1, pp. 393–403, doi: 10.1007/978-3-642-23857-4_38.

[21] G. K. Jati and S. Suyanto, "Discrete Cuckoo Search for Traveling Salesman Problem," in *2012 7th International Conference on Computing and Convergence Technology (ICCCT)*, 2014, pp. 993–997.

[22] G. K. Jati, R. Manurung, and Suyanto, *Discrete Firefly Algorithm for Traveling Salesman Problem: A New Movement Scheme*. Elsevier Inc., 2013.

[23] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[24] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. H. Černocký, "Empirical evaluation and combination of advanced language modeling techniques," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, no. August, pp. 605–608, 2011.

[25] P. Koehn, "Neural Machine Translation," no. 15, p. 66, 2015, doi: 10.5565/rev/tradumatica.203.

[26] S. Mirjalili, "How effective is the Grey Wolf optimizer in training multi-layer perceptrons," *Appl. Intell.*, vol. 43, no. 1, pp. 150–161, 2015, doi: 10.1007/s10489-014-0645-7.

[27] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," pp. 1–15, 2014.

[28] H. Faris, I. Aljarah, M. A. Al-Betar, and S. Mirjalili, "Grey wolf optimizer: a review of recent variants and applications," *Neural Comput. Appl.*, vol. 30, no. 2, pp. 413–435, 2018, doi: 10.1007/s00521-017-3272-5.

[29] M. MARCUS, B. SANTORINI, and M. MARCINKIEWICZ, "Building a Large Annotated Corpus of English: The Penn Treebank," *Comput. Linguist.*, vol. 19, no. 2, p. 313, 1993.

[30] H. Inan, K. Khosravi, and R. Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, pp. 1–13, 2019.