DEU FMD 27(79) (2025) 72-79

Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi
Dokuz Eylul University Faculty of Engineering Journal of Science and Engineering
Elektronik/Online ISSN: 2547-958X

RESEARCH ARTICLE / ARAŞTIRMA MAKALESI

Unleashing the Potential of Deep Learning Methods for Detecting Defective Expressions Using Hyperparameter Optimization Techniques

Anlatım Bozukluklarının Tespit Edilmesi için Hiperparametre Optimizasyon Teknikleri Kullanılarak Derin Öğrenme Yöntemlerinin Potansiyelinin Artırılması

Atilla Suncak [1*], Özlem Varlıklar [2]

[1] Department of Computer Technologies, Kastamonu Vocational Highschool, Kastamonu University, Kastamonu, TÜRKİYE
[2] Department of Computer Engineering, Faculty of Engineering, Dokuz Eylül University, İzmir, TÜRKİYE

Corresponding Author / Sorumlu Yazar *: atillasuncak@kastamonu.edu.tr

Abstract

Natural Language Processing (NLP) has emerged remarkable progress in the field of deep learning studies. Not only a superior alternative to rule-based NLP methods, deep learning-based techniques have also succeeded more accurate performances in various NLP tasks such as text classification, sentiment analysis or document clustering. Since the performance of a deep learning model undoubtedly depends on adjusting its hyperparameters ideally, tuning the most optimum hyperparameters determines the capability of the model learning in terms of meaningful pattern extraction from the input data. In this paper, hyperparameter optimization techniques of Bayesian Optimization, Random Search and Grid Search have been applied on the deep learning models of Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) for the purpose of detecting defective expressions in Turkish sentences. The hyperparameters of previously implemented LSTM and CNN models for this purpose have been adjusted using trial-and-error approach, which is time-consuming and cannot guarantee the most ideal model in general. After these hyperparameters have been adjusted using optimization techniques, the performances in terms of accuracy have been increased from 87.94% to 92.82% and from 84.33% to 89.79% for the models of LSTM and CNN respectively.

Keywords: *Bayesian optimization, Grid search, Hyperparameter optimization, NLP, Random search, Turkish*

Öz

Doğal Dil İşleme (DDİ), derin öğrenme çalışmaları alanında dikkat çekici ilerlemeler ortaya koymuştur. Derin öğrenme tabanlı teknikler, yalnızca kural tabanlı DDİ yöntemlerine üstün bir alternatif olmakla kalmayıp, aynı zamanda metin sınıflandırma, duygu analizi veya belge kümeleme gibi çeşitli DDİ görevlerinde de daha doğru performanslar elde etmeyi başarmıştır. Bir derin öğrenme modelinin performansı, şüphesiz ki hiperparametrelerinin ideal şekilde ayarlanmasına bağlı olduğundan, en ideal hiperparametrelerin ayarlanması, girdi verilerinden anlamlı örüntü çıkarma açısından model öğrenmesinin kapasitesini belirler. Bu makalede, Türkçe cümlelerdeki anlatım bozukluklarını tespit etmek amacıyla Uzun Kısa-Süreli Bellek (UKSB) ve Evrişimsel Sinir Ağları (ESA) derin öğrenme modelleri üzerinde Bayesian Optimization, Random Search ve Grid Search hiperparametre optimizasyon teknikleri uygulanmıştır. Bu amaçla daha önce geliştirilmiş UKSB ve ESA modellerinin hiperparametreleri, zaman alan ve genel olarak en ideal modeli garanti edemeyen deneme-yanılma yaklaşımı kullanılarak ayarlanmıştı. Bu hiperparametreler, optimizasyon teknikleri kullanılarak ayarlandıktan sonra ise, doğruluk açısından performansları UKSB ve ESA modelleri için sırasıyla %87,94'ten %92,82'ye ve %84,33'ten %89,79'a yükseltilmiştir.

Anahtar Kelimeler: *Bayesian optimization, Grid search, Hiperparametre optimizasyonu, Doğal dil işleme, Random search, Türkçe*

1. Introduction

Deep learning has made a revolutionary effect on the field of artificial intelligence. It enabled crucial advancements in image processing, computer vision, natural language processing, and various other domains [1]. The success of deep learning comes from the powerful neural network that is capable of learning complex representations from vast amounts of data. However, optimal performance of deep learning models depends on careful hyperparameter tuning of them since it plays an important role for their behavior in shaping and generalizing capabilities [2].

Hyperparameter optimization has become a critical component when building a deep learning model. It encompasses tuning ideal hyperparameters for the model in order to maximize model performance. By adjusting the optimum values for the hyperparameters such as dropout, learning rate, optimizers, batch size, activation functions and etc., deep learning models can reach their full potential and provide superior results [3, 4]. Hyperparameter techniques are highly benefited in several kinds of studies such as parkinson disease prediction [5], sensor-based human activity recognition [6], malware classification [7] and sarcasm recognition [8].

DOI: https://doi.org/10.21205/deufmd.2025277910

Geliş Tarihi / Received: 11.03.2024
Kabul Tarihi / Accepted: 16.04.2024

Defective expression is the Turkish grammatical term which addresses the ambiguities in Turkish sentences. It can be caused by several semantic reasons such as using a redundant word, using a word in a wrong place, uncertainty in meaning of the sentence and etc. Furthermore, a defective expression may also be occurred by morphological reasons such as using wrong suffixes in a word (Since Turkish is an agglutinative language, suffixes are used in almost any grammatical issue.), missing element, conjunction errors and etc. [9, 10]. In education, mass-media, Turkish linguistic studies or literary works, defective expressions are always the problems that must be handled in Turkey. Considering other language ambiguities which are generally occurred by a synonym word which has several meanings, Turkish defective expressions show solid differences in terms of semantics and context.

Linguists rather than computer scientists have studied Turkish defective expressions throughout the literature. To give examples, the study of Büyükikiz [11] analyzed defective expressions in the essays, written by 8th grade students and the study of Özdem [12] analyzed local newspapers whether they have defective expressions or not. On the other hand, apart from the aforementioned manual analyze studies by linguists, the studies of Suncak and Aktaş [13-15] implemented deep learning models (LSTM [Long Short-Term Memory], CNN [Convolutional Neural Network] and Bi-LSTM [Bidirectional LSTM]) and machine learning classifiers (KNN [K-Nearest Neighbor], SVM [Support Vector Machine] and RF [Random Forest]) to detect defective expressions in Turkish sentences. The performance of deep learning models in terms of accuracy varies from 84% to 88%, while machine learning models provide 58% to 78% accuracy rates. However, the hyperparameters of these models have been adjusted using trial-and-error method, therefore it can be clearly declarable that these performances may be increased using the right optimization methods.

This article will discuss the hyperparameter optimization of CNN and LSTM deep learning models. This study's main objective and contribution are to provide more ideal performances for detecting defective expressions in Turkish sentences in comparison to the other models whose hyperparameters are empirically adjusted. Throughout the article, the impact of adjusting optimal values of key hyperparameters on model performances by using optimization techniques such as Bayesian Optimization (BO), Random Search (RS) and Grid Search (GS) will be analyzed in details. Each technique has its own strengths and limitations, therefore their applicability differs in consideration of amount of data, number of hyperparameters or other unforeseen reasons. By understanding the nuances of these optimization methods, the most suitable approach for a model can be selected [16].

In summary, hyperparameter optimization study in order to increase the performance of learning models to detect defective expressions in Turkish sentences have been served in this article. By understanding the importance of hyperparameter optimization, we aim to equip Turkish NLP (Natural Language Processing) researchers with the necessary tools and another point of view to unlock the full potential of deep learning models and push the boundaries of artificial intelligence. This article consists of five sections and the organization is as follows: Section Two explains dataset, data preparation and learning models and hyperparameter optimization methods used in this study. Section Three tells the results and performances of the models. Section Four provides discussion over the results and Section Five concludes the article.

2. Materials and Methods

In this article, the effectiveness of three popular hyperparameter optimization techniques, namely Bayesian optimization, random search, and grid search, have been analyzed for fine-tuning deep learning models. The implementation of the models has been performed using Python programming language with PyCharm IDE and its NLP and machine learning libraries such as Keras [17] and Tensorflow [18] have been benefited for the purpose.

2.1. Dataset

The data to train and test the models have been collected from various open-access web sources of courses, schools and education centers in addition to the official exam center of Turkey (ÖSYM). Each data is originally a sentence that belong to a multiple choice question related to defective expression. Thus, that sentence had already been determined by the expert of the institution whether it has defective expression or not since the answers have also been served related to each question. After a comprehensive search, 9710 Turkish sentences, 4299 of which consist of defective expressions and rest are grammatically proper, have been collected and each sentence according to having defective expression or not have been labelled as DEF or NON-DEF respectively, shown in Table 1.

Table 1. Sample of the sentences in dataset.

Sentence	Label
Dişçiye hiç ya da çok seyrek gidiyorlar.	DEF
Bu kursta, güzel konuşmanın inceliklerini öğreniyorum.	NON-DEF
Davete katılanların hemen hemen hepsini tanıyorum.	NON-DEF
Halk arasında da en iyi yaptığı işle sevilir sayılır duruma düştü.	DEF
Bu davranış insandan insana göre değişir.	DEF
Bu konuda yapılan açıklamaların anlaşılmayacak bir yanı bulunmuyor.	NON-DEF
Teknoloji ne kadar artarsa da el emeğinin önemi azalmıyor.	DEF
Toplumsal ve bireysel olaylara, yan tutmadan bakar.	NON-DEF
Yaptıklarını kendi ağzıyla itiraf etti.	DEF
İşe geç geleceğini hiç olmazsa bana haber verseydin bari.	DEF
İlgililer bu konuda görüş alışverişinde bulundular.	NON-DEF

Since that amount of data are inadequate for training an NLP-purposed learning model, the data have been augmented up to 29756 (13398 of them have defective expressions and 16358 of them are proper ones) using Turkish Synonym Dictionary [19], seen in Table 2.

Table 2. The number of sentences in dataset before and after data augmentation.

	Number of Sentences with Defective Expression	Number of Sentences without Defective Expression	Total
Before Augmentation	4299	5411	9710
After Augmentation	13398	16358	29756

In order to train and test a learning model for text operations, the data need the adequate preparation operations of NLP. For this reason, the text data have been applied some NLP techniques, seen in Figure 1, before feeding the models such as punctuation removal, normalization, stop-word removal and etc. However, we avoided stemming or lemmatization operations since suffixes may cause defective expression.

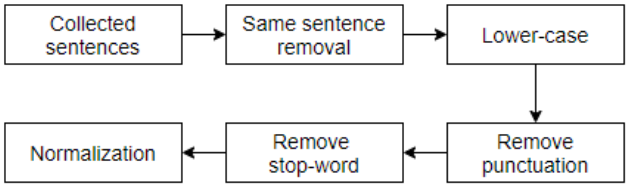

Figure 1. The flow diagram of data preprocess operations.

2.2. Word Vectors and Learning Models

Vectorising words or word embeddings refer to vectors for document vocabulary representation [20]. In this study, Word2vec technique has been used for word embedding extraction, introduced by Mikolov et al. [21] since word vectors are capable of providing better context information of the related word in comparison to the word itself. This technique considers the context of the word by its surrounding ones in the sentence for better semantic analysis [22]. In conclusion, each word of each sentence has been vectorised to train the learning models.

In order to detect defective expressions in Turkish sentences, the deep learning models of LSTM and CNN have been implemented. LSTM is known to be the most appropriate network for handling a long sequence of data among other RNN methods, introduced by Hochreiter and Schmidhuber [23]. Thanks to its memory cells, it has the capability of avoiding long term dependencies, which prevents vanishing gradient problem [24]. CNN is one of the most popular one among deep learning models for visual operations such as image classification or object detection, introduced by LeCun et al. [25]. On the other hand, CNN is also pretty applicable on text operations with considerably successful performances such as text classification, character level classification and etc.

The flow diagram of learning models is presented in Figure 2.

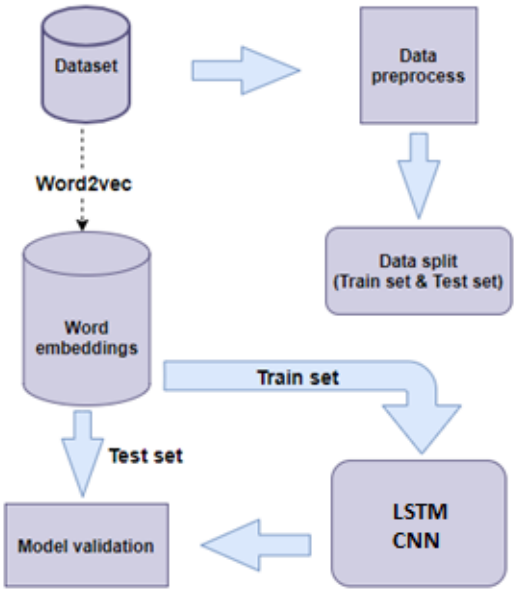

Figure 2. The flow diagram of learning models.

2.3. Evaluation Metrics

In order to measure the performance of each model; the metrics of accuracy, precision, recall and f1 score [26] have been applied, defined as Eqs. (1) - (2) - (3) - (4) respectively. The abbreviations of TP, TN, FP and FN refers to True-positive, True-negative, False-positive and False-negative respectively.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1\ Score = \frac{2 * precision * recall}{precision + recall} \quad (4)$$

Mean Squared Error (MSE) is the loss function of the models that calculates the average of the square of differences between original and predicted values of the data, defined as Equation (5). After several trials and errors with other loss functions, MSE resulted in better performances in terms of reducing loss.

$$MSE = \frac{1}{N}\sum_{i=0}^{n}(actual\ v. - predicted\ v.)^2 \quad (5)$$

In the equation of MSE, N refers to the total number of data and v means 'values'. etc.

2.4. Hyperparameter Optimization Technique

The fact that both machine learning classifiers and deep learning models have a wide range of hyperparameters to be adjusted, it is undoubtedly crucial that tuning them in the most ideal of all the worlds in order to implement the optimum learning model and get the highest performance [27]. These techniques have both advantages and limitations according to the number of hyperparameter, the learning model that is applied on and etc. In this study, three popular hyperparameter optimization methods have been applied on the learning models, which are Grid Search (GS), Random Search (RS) and Bayesian Optimization (BO).

GS is a technique that has been widely used for hyperparameter tuning of learning models. It is known to be a straightforward approach that all combinations of the hyperparameters are defined in a grid and tried exhaustively [28]. Although it is simple, straightforward and exhaustive which guarantees that every hyperparameter combination will be calculated; there are important disadvantages such as being extremely computationally expensive and lack of flexibility which it cannot adapt to the observed result and limit the number of computations in addition to the fact that. In addition, the more the number of hyperparameters increases, the more the number of grid points grows exponentially, which results in inefficient performances [29, 30].

Despite being simple, RS is a powerful technique for hyperparameter optimization, introduced by Bergstra and Bengio [3]. This technique samples the parameter combinations randomly and discovers good configurations of hyperparameters by chance by exploring the search space in stochastic manner. Although GS is computationally expensive in high-dimensional spaces, RS is more effective in comparison. Despite the fact that it

may not guarantee to find the optimum values, it provides efficient and perfect results with fewer calculations [31, 32].

BO is a well-known technique for being powerful and efficient for tuning hyperparameters and global optimization, introduced by Mockus [33] and his co-researchers [34]. This technique explores the search space by balancing exploration by benefiting the power of probabilistic modelling and Gaussian process. BO selects the most suitable combination to evaluate by considering both previously observed data and gained knowledge. This process leads to the exploration of ideal solution with fewer calculations, which makes BO suitable for the models that are time-consuming and exhaustive [35, 36].

The learning process highly relies on the success of the hyperparameter optimization. The flow of hyperparameter optimization process is depicted in Figure 3.

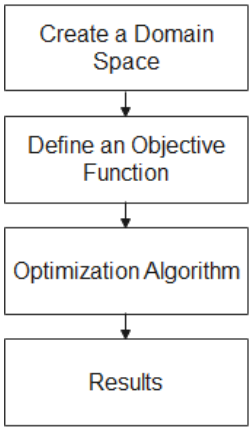

Figure 3. The flow diagram of learning models.

'Create a Domain Space' addresses the input values from the dataset. 'Define an Objective Function' represents the aim to increase the accuracy of learning models, therefore the objective function is the metric of 'accuracy'. 'Optimization Algorithm' is the method(s) used for hyperparameter optimization, therefore it addresses RS, GS and BO. Finally, 'Result' means the developed model after hyperparameter optimization technique is held.

3. Results

This section tells the performance results of the detection models when applying hyperparameter optimization techniques. It discusses them by comparing with the previous ones which the model hyperparameters have been adjusted with the approach of trial-and-error. Furthermore, each optimization technique has been applied using 10-fold cross validation in order to acquire more proper results and prevent overfitting. To start with, Table 3 depicts the hyperparameters of LSTM model with their ranges for each optimization technique applied.

Table 3. Hyperparameters of LSTM and their ranges for each optimization technique.

Hyperparameters	Bayesian Optimization	Random Search	Grid Search
Hidden Layer	32 to 512 (step = 32)	32 to 512 (step = 32)	64, 128, 256, 512
Dropout Rate	0.1 to 0.5 (step = 0.1)	0.1 to 0.5 (step = 0.1)	0.1, 0.3, 0.5
Activation Function	Softmax, Sigmoid, ReLu	Softmax, Sigmoid, ReLu	Softmax, Sigmoid, ReLu
Learning Rate	1e-4 to 1e-2 (log sampling)	1e-4 to 1e-2 (log sampling)	1e-4, 1e-3, 1e-2

Within the hyperparameters above, LSTM model have been optimized using each optimization technique and each technique resulted with top five best combinations. After that, each of these combinations have been applied on the model using 10-fold cross validation in order to prevent overfitting and get consistent performance. Table 4, Table 5 and Table 6 show the model performances when the best five hyperparameter combinations of Bayesian optimization, Random search and grid search are applied respectively.

CNN model, on the other hand, has been implemented using three convolution layers for the purpose of detecting defective expressions. Each layer has the hyperparameters of number of filters, kernel, pool size and dropout layer. The fact that there is too many hyperparameters with CNN model when multiplying by three, Grid search optimization has become a problem since it is originally an exhaustive approach; therefore, these too many hyperparameters made Grid search a grueling option and finally resulted in error due to the performance limitations of the device. As a result, only Bayesian optimization and Random search approaches could have been applied for hyperparameter optimization for CNN model. Table 7 tells the hyperparameters of CNN model with their ranges for each optimization technique applied.

Within the hyperparameters, given in Table 7, CNN model have been optimized using optimization techniques and each technique resulted with top five best combinations. After that, each of these combinations have been applied on the model using 10-fold cross validation in order to prevent overfitting and get consistent performance. Table 8, and Table 9 show the model performances when the best five hyperparameter combinations of Bayesian optimization and Random search are applied respectively. L.R. stands for Learning Rate and A.F. stands for Activation Function.

Table 4. Best five hyperparameter combinations of Bayesian optimization and their performances with LSTM

Hidden Layer	Dropout Rate	Learning Rate	Activation Function	Precision	Recall	F1 Score	Accuracy	Loss
416	0.1	0.004	Softmax	0.93	0.95	0.94	92.82	0.06
320	0.4	0.001	Softmax	0.94	0.94	0.94	92.78	0.06
192	0.4	0.0006	ReLu	0.93	0.94	0.93	91.85	0.06
320	0.3	0.002	Sigmoid	0.94	0.95	0.94	92.63	0.06
128	0.2	0.001	Sigmoid	0.93	0.93	0.93	91.57	0.07

Table 5. Best five hyperparameter combinations of Random search and their performances with LSTM.

Hidden Layer	Dropout Rate	Learning Rate	Activation Function	Precision	Recall	F1 Score	Accuracy	Loss
192	0.2	0.001	Sigmoid	0.94	0.93	0.93	92.14	0.07
480	0.4	0.003	Sigmoid	0.92	0.93	0.93	91.31	0.07
512	0.2	0.0002	Sigmoid	0.92	0.94	0.92	90.66	0.07
352	0.2	0.001	Softmax	0.94	0.95	0.94	92.49	0.06
352	0.5	0.008	Softmax	0.93	0.95	0.94	92.71	0.06

Table 6. Best five hyperparameter combinations of Grid search and their performances with LSTM.

Hidden Layer	Dropout Rate	Learning Rate	Activation Function	Precision	Recall	F1 Score	Accuracy	Loss
128	0.5	0.001	Softmax	0.92	0.92	0.92	90.45	0.08
64	0.5	0.0001	Softmax	0.84	0.89	0.86	82.92	0.14
352	0.2	0.001	Softmax	0.94	0.95	0.93	92.65	0.06
128	0.5	0.0001	Softmax	0.88	0.91	0.89	86.23	0.13
128	0.3	0.001	Sigmoid	0.93	0.94	0.93	91.74	0.08

Table 7. Hyperparameters of CNN and their ranges for each optimization technique.

Hyperparameters	Bayesian Optimization	Random Search
Number_of_filters_1	32 to 256 (step = 32)	32 to 256 (step = 32)
Kernel_Layer1	2 to 5 (step = 1)	2 to 5 (step = 1)
PoolSize_Layer1	2 to 3 (step = 1)	2 to 3 (step = 1)
Dropout_Layer1	0.1 to 0.5 (step = 0.1)	0.1 to 0.5 (step = 0.1)
Number_of_filters_2	32 to 256 (step = 32)	32 to 256 (step = 32)
Kernel_Layer2	2 to 5 (step = 1)	2 to 5 (step = 1)
PoolSize_Layer2	2 to 3 (step = 1)	2 to 3 (step = 1)
Dropout_Layer2	0.1 to 0.5 (step = 0.1)	0.1 to 0.5 (step = 0.1)
Number_of_filters_3	32 to 256 (step = 32)	32 to 256 (step = 32)
Kernel_Layer3	2 to 5 (step = 1)	2 to 5 (step = 1)
PoolSize_Layer3	2 to 3 (step = 1)	2 to 3 (step = 1)
Dropout_Layer3	0.1 to 0.5 (step = 0.1)	0.1 to 0.5 (step = 0.1)
Activation Function	Softmax, Sigmoid, ReLu	Softmax, Sigmoid, ReLu
Learning Rate	1e-4 to 1e-2 (log sampling)	1e-4 to 1e-2 (log sampling)

Table 8. Best five hyperparameter combinations of Bayesian optimization and their performances with CNN.

Layer 1	Layer 2	Layer 3	Others	Precision	Recall	F1 Score	Accuracy
Filters = 224 Kernel = 4 Dropout = 0.3 Pool Size = 2	Filters = 256 Kernel = 2 Dropout = 0.3 Pool Size = 2	Filters = 256 Kernel = 3 Dropout = 0.2 Pool Size = 3	L. R. = 0.0005 A. F. = ReLu	0.90	0.94	0.92	89.79
Filters = 160 Kernel = 4 Dropout = 0.1 Pool Size = 3	Filters = 224 Kernel = 4 Dropout = 0.3 Pool Size = 2	Filters = 160 Kernel = 4 Dropout = 0.1 Pool Size = 2	L. R. = 0.0001 A. F. = Softmax	0.89	0.93	0.92	88.89
Filters = 64 Kernel = 5 Dropout = 0.1 Pool Size = 2	Filters = 256 Kernel = 2 Dropout = 0.1 Pool Size = 3	Filters = 160 Kernel = 3 Dropout = 0.2 Pool Size = 3	L. R. = 0.002 A. F. = Softmax	0.89	0.93	0.91	88.40
Filters = 224 Kernel = 3 Dropout = 0.1 Pool Size = 3	Filters = 128 Kernel = 5 Dropout = 0.3 Pool Size = 3	Filters = 128 Kernel = 2 Dropout = 0.1 Pool Size = 2	L. R. = 0.0001 A. F. = Sigmoid	0.90	0.94	0.92	89.06
Filters = 64 Kernel = 2 Dropout = 0.1 Pool Size = 3	Filters = 256 Kernel = 2 Dropout = 0.1 Pool Size = 2	Filters = 128 Kernel = 3 Dropout = 0.2 Pool Size = 2	L. R. = 0.001 A. F. = ReLu	0.89	0.94	0.91	88.94

Table 9. Best five hyperparameter combinations of Random search and their performances with CNN.

Layer 1	Layer 2	Layer 3	Others	Precision	Recall	F1 Score	Accuracy
Filters = 128 Kernel = 5 Dropout = 0.2 Pool Size = 2	Filters = 224 Kernel = 4 Dropout = 0.5 Pool Size = 3	Filters = 64 Kernel = 2 Dropout = 0.2 Pool Size = 3	L. R. = 0.0002 A. F. = Softmax	0.88	0.93	0.92	89.08
Filters = 64 Kernel = 5 Dropout = 0.1 Pool Size = 3	Filters = 128 Kernel = 3 Dropout = 0.1 Pool Size = 2	Filters = 96 Kernel = 3 Dropout = 0.3 Pool Size = 3	L. R. = 0.001 A. F. = Sigmoid	0.89	0.91	0.91	88.80
Filters = 224 Kernel = 5 Dropout = 0.5 Pool Size = 2	Filters = 224 Kernel = 2 Dropout = 0.2 Pool Size = 2	Filters = 64 Kernel = 2 Dropout = 0.4 Pool Size = 2	L. R. = 0.0001 A. F. = ReLu	0.88	0.91	0.90	87.46
Filters = 160 Kernel = 4 Dropout = 0.4 Pool Size = 2	Filters = 128 Kernel = 3 Dropout = 0.3 Pool Size = 3	Filters = 224 Kernel = 3 Dropout = 0.3 Pool Size = 3	L. R. = 0.002 A. F. = Sigmoid	0.89	0.90	0.91	88.81
Filters = 128 Kernel = 3 Dropout = 0.2 Pool Size = 3	Filters = 32 Kernel = 2 Dropout = 0.2 Pool Size = 3	Filters = 256 Kernel = 2 Dropout = 0.5 Pool Size = 2	L. R. = 0.0006 A. F. = ReLu	0.91	0.92	0.92	89.21

4. Discussion

Applying optimization techniques on deep learning models has demonstrated a slight but important improvement in performance when comparing to existing results from trial-and-error technique. When comparing these two learning methods, LSTM comes into prominence in terms of performance due to the capabilities of long-term dependencies. On the other hand, CNN performed quite acceptable results and increased its performance. Table 10 shows the best performances of each hyperparameter optimization techniques applied on LSTM with the model, optimized by trial-and-error technique.

Even though the metric results of optimization techniques are all close to each other, it must be pointed out that BO provided completely different hyperparameters except activation function in comparison to RS and GS, which they adjusted exactly the same hyperparameters to get the ideal model. Due to the fact that BO benefits the power of probabilistic modelling and Gaussian process, it can implement the most ideal model with the hyperparameters in the search space that other techniques are not capable of. To conclude, all optimization techniques performed better than trial-and-error technique to provide the most ideal model of LSTM.

Table 10. Best performances using hyperparameter optimizations applied on LSTM model.

Optimization Technique	Hidden Layer	Dropout Rate	Learning Rate	Activation Function	Precision	Recall	F1 Score	Accuracy	Loss
Bayesian Optimization	416	0.1	0.004	Softmax	0.93	0.95	0.94	92.82	0.06
Random Search	352	0.2	0.001	Softmax	0.94	0.95	0.94	92.49	0.06
Grid Search	352	0.2	0.001	Softmax	0.94	0.95	0.93	92.65	0.06
Trial-and-error	256	0.3	0.001	Softmax	0.88	0.89	0.88	87.94	0.09

Table 11 depicts the best performances of each hyperparameter optimization techniques applied on CNN with the model, optimized by trial-and-error technique. As aforementioned, GS optimization could not be applied on CNN due to the fact that there are a lot of hyperparameters, thus thousands of combinations. The fact that GS performs an exhaustive search and the limitations of the machine to run the code, the computer could not handle the optimization. According to the results, it can be found out that the performances of CNN models, adjusted by BO and RS, showed close accuracy successes. BO comes into prominence with a slight difference for providing the most ideal model when comparing to RS technique. Both techniques, however, showed better performances in comparison to trial-and-error technique.

Table 11. Best performances using hyperparameter optimizations applied on CNN model.

Optimization Technique	Layer 1	Layer 2	Layer 3	Others	Precision	Recall	F1 Score	Accuracy	Loss
Bayesian Optimization	Filters = 224 Kernel = 4 Dropout = 0.3 Pool Size = 2	Filters = 256 Kernel = 2 Dropout = 0.3 Pool Size = 2	Filters = 256 Kernel = 3 Dropout = 0.2 Pool Size = 3	L.R. = 0.0005 A.F. = ReLu	0.90	0.94	0.92	89.79	0.10
Random Search	Filters = 128 Kernel = 3 Dropout = 0.2 Pool Size = 3	Filters = 32 Kernel = 2 Dropout = 0.2 Pool Size = 3	Filters = 256 Kernel = 2 Dropout = 0.5 Pool Size = 2	L.R. = 0.0006 A.F. = ReLu	0.91	0.92	0.92	89.21	0.11
Trial-and-Error	Filters = 128 Kernel = 3 Dropout = 0.3 Pool Size = 3	Filters = 128 Kernel = 3 Dropout = 0.3 Pool Size = 2	Filters = 128 Kernel = 2 Dropout = 0.3 Pool Size = 2	L.R. = 0.0001 A.F. = Softmax	0.80	0.88	0.84	84.33	0.12

5. Conclusion and Future Works

In this paper, we applied hyperparameter optimization techniques of Random search, Grid search and Bayesian optimization in order to adjust the learning models of LSTM and CNN to detect defective expressions in Turkish sentences. Previous studies showed that the hyperparameters of these models have been adjusted using trial-and-error technique, which requires an excessive time, knowledge and luck for providing the most ideal model. However, hyperparameter optimization techniques reduces this excessive spent time and selects the best hyperparameter combination out of search space, that has been created by each technique according to their own background algorithm.

Bayesian optimization benefits the power of probabilistic modelling and Gaussian process. Random search samples the parameter combinations randomly and discovers good configurations of hyperparameters in stochastic manner. Grid search, on the other hand, performs an exhaustive search, which guarantees the most ideal model; however, consumes an excessive time and requires a powerful machine. After these techniques are applied on learning models, both LSTM and CNN models performed more accurate than the previous models adjusted using trial-and-error technique.

This study contributes to Turkish NLP and is a great source for the researchers who study this area. In future, a more powerful machine must be considered for providing a larger search space using a wider hyperparameter interval, which will result higher number of combinations to implement more ideal models. Since more hyperparameters require more calculations, the number of process increases dramatically, therefore regular computers cannot handle and halts the process. What is more, this study can be turned into an application and serviced to the students and teachers who deals with Turkish education. Furthermore, this application also be used in mass-media workers. They can check their writings before publication whether there are any defective expressions in their writings or not.

Ethics committee approval and conflict of interest statement

This article does not require ethics committee approval and has no conflicts of interest with any individual or institution.

Acknowledgment

The authors would like to thank the editors and anonymous reviewers for providing insightful suggestions and comments to improve the quality of the research paper.

Author Contribution Statement

Author 1 (corresponding author) conducted the literature review, wrote the manuscript focusing on conceptualization and results presentation, and developed and implemented the code used in this study. Author 2 supervised the results from the learning models, contributed to writing and editing processes,

and conducted a critical review offering feedback for improvement.

References

[1] LeCun, Y., Bengio, Y., & Hinton, G., 2015. Deep learning. Nature, Vol. 521(7553), pp. 436-444.

[2] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

[3] Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. Journal of Machine Learning Research, Vol. 13(2).

[4] Smith, L.N., 2018. A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay. arXiv preprint arXiv:1803.09820.

[5] Lilhore, U.K., Dalal, S., Faujdar, N., Margala, M., Chakrabarti, P., Chakrabarti, T., Velmurugan, H., 2023. Hybrid CNN-LSTM model with efficient hyperparameter tuning for prediction of Parkinson's disease. Scientific Reports, Vol. 13(1), p. 14605.

[6] El Ghazi, M., Aknin, N., 2024. Optimizing Deep LSTM Model through Hyperparameter Tuning for Sensor-Based Human Activity Recognition in Smart Home. Informatica, Vol. 47(10).

[7] Mehta, R., Jurečková, O., Stamp, M., 2024. A natural language processing approach to Malware classification. Journal of Computer Virology and Hacking Techniques, Vol. 20(1), pp. 173-184.

[8] Palaniammal, M.A., Anandababu, P., 2024. Enhancing Sarcasm Recognition Using Chicken Swarm Optimization Algorithm with Graph Neural Network on Social Media.

[9] Demir, C., 2020. Lexical and structural ambiguities in student writing: An assessment and evaluation of results. Academic Education Research Journal, Vol. 8, pp. 100-108. DOI: 10.30918/AERJ.8S3.20.077.

[10] Göksel, A., Kerslake, C., 2004. Turkish: A comprehensive grammar. Routledge.

[11] Büyükikiz, K.K., 2007. İlköğretim 8. sınıf öğrencilerinin yazılı anlatım becerilerinin söz dizimi ve anlatım bozukluğu açısından değerlendirilmesi. Gazi University, Ankara, Turkey.

[12] Özdem, A., 2012. Çanakkale'deki yerel gazetelerin anlatım bozuklukları açısından incelenmesi. Çanakkale Onsekiz Mart University, Çanakkale, Turkey.

[13] Suncak, A., Aktaş, Ö., 2021. A novel approach for detecting defective expressions in Turkish. Journal of Artificial Intelligence and Data Science (JAIDA), Vol. 1, pp. 35-40.

[14] Suncak, A., 2022. Developing a new approach in natural language understanding to defect defective expressions in Turkish sentences. Dokuz Eylül University, İzmir, Turkey.

[15] Suncak, A., Aktaş, Ö., 2022. Detecting Defective Expressions in Turkish Sentences Using a Hybrid Deep Learning Method. Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi, Vol. 24(72), pp. 825-834.

[16] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B., 2011. Algorithms for hyper-parameter optimization. Advances in Neural Information Processing Systems, Vol. 24.

[17] Chollet, F., 2015. Keras. GitHub. Retrieved from https://github.com/fchollet/keras.

[18] Abadi, M., et al., 2016. TensorFlow. GitHub. Retrieved from https://github.com/tensorflow.

[19] Aktaş, Ö., Birant, Ç.C., Aksu, B., Çebi, Y., 2013. Automated synonym dictionary generation tool for Turkish (ASDICT). Bilig, Vol. 65, p. 47.

[20] Muhammad, P.F., Kusumaningrum, R., Wibowo, A., 2021. Sentiment analysis using Word2vec and long short-term memory (LSTM) for Indonesian hotel reviews. Procedia Computer Science, Vol. 179, pp. 728-735.

[21] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, pp. 3111-3119.

[22] Fang, G., Zeng, F., Li, X., Yao, L., 2021. Word2vec based deep learning network for DNA N4-methylcytosine sites identification. Procedia Computer Science, Vol. 187, pp. 270-277.

[23] Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation, Vol. 9(8), pp. 1735-1780.

[24] Gers, F.A., Schmidhuber, J., Cummins, F., 2000. Learning to forget: Continual prediction with LSTM. Neural Computation, Vol. 12(10), pp. 2451-2471.

[25] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, Vol. 86(11), pp. 2278-2324.

[26] Hossin, M., Sulaiman, M.N., 2015. A review on evaluation metrics for data classification evaluations. International Journal of Data Mining & Knowledge Management Process, Vol. 5(2), p. 1.

[27] Feurer, M., Hutter, F., 2019. Hyperparameter optimization. Automated Machine Learning: Methods, Systems, Challenges, pp. 3-33.

[28] Michalski, R.S., Carbonell, J.G., Mitchell, T.M., 1984. Machine learning: An artificial intelligence approach. Springer-Verlag Berlin Heidelberg.

[29] Pontes, F.J., Amorim, G.F., Balestrassi, P.P., Paiva, A.P., Ferreira, J.R., 2016. Design of experiments and focused grid search for neural network parameter optimization. Neurocomputing, Vol. 186, pp. 22-34.

[30] Ensor, K.B., Glynn, P.W., 1997. Stochastic optimization via grid search. Lectures in Applied Mathematics, Vol. 33, pp. 89-100.

[31] Andradóttir, S., 2006. An overview of simulation optimization via random search. Handbooks in Operations Research and Management Science, Vol. 13, pp. 617-631.

[32] Andradóttir, S., 2014. A review of random search methods. Handbook of Simulation Optimization, pp. 277-292.

[33] Močkus, J., 1975. On Bayesian methods for seeking the extremum. In Optimization Techniques IFIP Technical Conference, pp. 400-404.

[34] Mockus, J., Mockus, J., 1989. The Bayesian approach to local optimization. Springer Netherlands, pp. 125-156.

[35] Frazier, P.I., 2018. Bayesian optimization. In Recent Advances in Optimization and Modeling of Contemporary Problems, pp. 255-278.

[36] Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical Bayesian optimization of machine learning algorithms. Advances in Neural Information Processing Systems, Vol. 25.