ALBERT4Spam: A Novel Approach for Spam Detection on Social Networks

Araştırma Makalesi/Research Article

Rezan BAKIR*[1], Hasan Erbay[2], Halit BAKIR[3]

[1]Department of Computer Engineering , Sivas science and technology university, Sivas, Turkey
[2]Department of Computer Engineering, Ostim Technical University, Ankara, Turkey
[3]Department of Computer Engineering , Sivas science and technology university, Sivas, Turkey
rezan.bakir@sivas.edu.tr, hasan_erbay@yahoo.com, halit.bakir@sivas.edu.tr
(Geliş/Received:26.01.2024; Kabul/Accepted:04.03.2024)
DOI: 10.17671/gazibtd.1426230

***Abstract*—** Engaging in social media browsing stands out as one of the most prevalent online activities. As social media increasingly integrates into our daily routines, it opens up numerous opportunities for spammers seeking to target individuals through these platforms. Given the concise and sporadic nature of messages exchanged on social networks, they fall within the realm of short text classification challenges. Effectively addressing such issues requires appropriately representing the text to enhance classifier efficiency.Accordingly, this study utilizes robust representations derived from contextualized models as a component of the feature extraction process within our deep neural network model, which is built upon the Bidirectional Long Short-Term Memory neural network (BLSTM). Introducing ALBERT4Spam, the study presents a deep learning methodology aimed at identifying spam on social networking platforms. It harnesses the proven ALBERT model to acquire contextualized word representations, thereby elevating the effectiveness of the suggested neural network framework.The random search method was used to fine-tune the ALBERT4Spam model's hyperparameters, which included the number of BLSTM layers, neuron count, layer count, activation function, weight initializer, learning rate, optimizer, and dropout, in order to obtain optimal performance. The experiments conducted on three benchmark datasets demonstrate that our innovative model surpasses widely used methods in social network spam detection. The precision results stand at 0.98, 0.96, and 0.98 for Twitter, YouTube, and SMS datasets, respectively, showcasing superior performance outcomes.

Keywords—spam detection, word embedding, deep learning, BERT, ALBERT, BLSTM

ALBERT4Spam: Sosyal Ağlarda Spam Tespitinde Yeni Bir Yaklaşım

Özet: Sosyal medyada gezinmek, insanların katıldığı en popüler çevrimiçi etkinliklerden biridir. Sosyal medya, günlük hayatlarımıza daha fazla entegre oldukça, kurbanlarına sosyal ağ siteleri aracılığıyla ulaşmak isteyen spam göndericilere sayısız fırsat sunmaktadır. Sosyal ağlar üzerinden iletilen mesajlar genellikle kısa ve seyrek olduğu için, kısa metin sınıflandırma problemleri ortaya çıkmaktadır. Bu tür sorunların üstesinden gelmek için, sınıflandırıcının etkinliğini artırmak amacıyla metni uygun şekilde temsil etmek önemlidir. Bu amaçla, bu çalışma, sosyal medya platformlarında spam tanımlamak için derin öğrenme yaklaşımı olan ALBERT4Spam'i tanıtmaktadır. ALBERT modelinden gelen bağlamsal kelime temsilleri kullanılarak önerilen Çift Yönlü Uzun Kısa Süreli Bellek (BLSTM) sinir ağı mimarisinin performansı artırılmıştır. Önerilen ALBERT4Spam modelinde kullanılan BLSTM katman sayısı, nöron sayısı, katman sayısı, aktivasyon fonksiyonu, öğrenme oranı, ağırlık başlangıç, optimizer ve bırakma gibi hiperparametreler, en iyi performansa ulaşmak için rastgele arama yöntemi kullanılarak optimize edilmiştir. Üç farklı standart veri seti üzerinde yapılan deneysel sonuçlar, önerilen modelin mevcut modellere kıyasla sosyal medya platformlarındaki spam mesajlarını daha başarılı bir şekilde tespit ettiğini göstermektedir. Yapılan deneyler, Twitter, YouTube ve SMS veri setlerinde sırasıyla %98, %96 ve %98 kesinlik sonuçlarıyla daha üstün performans sergilediğini ortaya koymaktadır.

Anahtar Kelimeler— spam tespiti, kelime gömme, derin öğrenme, BERT, ALBERT, BLSTM

1. INTRODUCTION

Even as social networks have become integral to human connection and relationship development, enhancing lives, they have also introduced security concerns that pose serious threats to user privacy and security.

In this discourse, we draw attention to the pervasive issue of spam on social media, a common challenge for users across various networks. Spam messages, defined as unsolicited, low-quality content that disrupts users and wastes their time, encompass a range of nuisances, including advertisements, malicious links, trending hashtags, and images with concealed URLs. Because of the brief and sparse nature of social media text, as well as potential grammar or spelling problems, symbols, emoticons, and various patterns, handling text derived from social media presents particular issues. Addressing the problem of spam messages necessitates finding an effective representation of text to enhance the classifier's ability to discern messages accurately. Various word representation approaches have been proposed, including traditional methods like the Bag of Word model, classical word embedding methods such as word2vec [1], GloVE [2] , and fastText [3], and transformer-based methods like BERT [4]. Traditional word embedding models successfully handle regular-length texts, they encounter difficulties with short texts, particularly those sourced from social platforms due to sparsity and text ambiguity. Conventional word embedding approaches, which represent words with dense vectors, address data sparsity, However, despite their restrictions, they show limitations. Notably, the "out of vocabulary" problem arises when models unable to offer vector representations for words that are not in their dictionary. Another challenge is their context independence, where one vector is output for each word irrespective of its position in the sentence. To overcome these limitations, a dynamic word representation is essential. Contextual embedding, prevalent in transformer-based models, emerges as an ideal choice for short texts, generating word embeddings dependent on the surrounding words in a sentence. BERT [4], a prominent transformer-based model, achieves cutting-edge results in NLP. However, its full-sized models pose computational and memory challenges. The Google research team introduced ALBERT [5] (A Lite BERT) as a solution, incorporating parameter-reduction techniques such as factorized embedding parameterization and cross-layer parameter sharing. ALBERT, featuring a self-supervised loss for sentence-order prediction, enhances inter-sentence coherence.

This paper explores the dynamics of spam on social networks, delving into the challenges of short text representation and introducing innovative solutions like ALBERT for improved classification in the realm of spam detection.

Motivation and contribution

Learning natural language representations has proven valuable across various NLP tasks and has become a widespread practice. A notable shift in recent years involves transitioning from pre-training word embeddings—whether standard or contextualized—to comprehensive network pre-training followed by fine-tuning for specific tasks.

The inclusion of contextual word embeddings into neural architectures has improved the performance of cutting-edge NLP tasks dramatically. Given our focus on short text classification challenges, combining contextualized word embeddings with deep learning architectures emerges as an optimal solution for addressing the sparsity and other issues prevalent in social networks' texts. Consequently, we opted to utilize the ALBERT model as a feature extractor, integrating it with deep learning techniques to overcome challenges associated with traditional word representation methods.

The key contribution of our study is that we created a novel hybrid deep neural network model to identify spam on social platforms. This model combines a stacked bidirectional long short-term neural network architecture with contextualized word representations from the ALBERT model to provide more context. To the best of our knowledge, the combination of the bidirectional LSTM and the BERT model has never been applied to the domain of social network spam detection. The research involves an ablation investigation as well as hyperparameter tuning to improve performance. The model's usefulness in handling brief text data was confirmed by testing it on three benchmark datasets.

2. RELATED WORKS

Spam, in various forms, has long been associated with activities causing harm or disruption to users. While the word is not new, having appeared in contexts such as email and SMS spam, the advent of social networks has led to an increase in social spam. Consequently, spam detection on social networks has emerged as a prominent research area over the last decade.

This section provides an overview of significant contributions by various researchers in the realm of spam detection on social networks. Previous studies have employed diverse approaches, relying on features such as content-based features, account-based features, or a combination thereof, along with other innovative attributes. Some studies focused on identifying spam messages, while others aimed at detecting spammers' accounts. For instance, in [6] , the authors proposed a methodology to detect spam profiles on Twitter by extracting and analyzing public features, irrespective of the language used in tweets. And in the research study presented in [7], an unsupervised spam detection approach was utilized to differentiate spammers from legitimate users.

In another study [8], enhancements to three classification algorithms for detecting spammers on Twitter were introduced, leveraging features that capture similarities among spam accounts. On a different note, [9] research study presented an unsupervised method for identifying spam messages on Twitter, utilizing a fuzzy string-matching based technique.

Moreover, the paper [10] suggested a strategy for detecting spam accounts that combines text analysis, machine learning, and short link analysis. A dataset was developed to detect user communications, hyperlinks in messages, and spam accounts.

Furthermore, the [11] study investigated Twitter spam detection using a hybrid framework, SMOTE-ENN, which combines SMOTE with Edited Nearest Neighbors. The data is then fed into deep learning classification approaches, and the best framework is found through performance comparisons and simulations.

Addressing the spam issue on YouTube's social network, [12] study introduced a framework employing Support Vector Machine (SVM) and K-Nearest Neighbor (k-NN) for spam comment detection. In [13], similar classifiers, along with a multilayer perceptron, were used to combat spam on YouTube. Authors in [14] study suggested a method for distinguishing spammers from real users based on Twitter-specific features and a Naive Bayes classifier.

On the other hand, the [15] study used Artificial Intelligence algorithms to detect spam in YouTube comments, achieving a 91.65% accuracy rate. According to authors, the study outperforms existing technologies like Google Safe Search and YouTube Bookmaker, highlighting the complexity of spam and the effectiveness of existing AI techniques. Similarly, the [16] study used Naive Bayes classification to identify spam comments on the internet, achieving an accuracy of 92.78%. Furthermore, the [17] study proposes a method for detecting English spam comments on YouTube using a stacked ensemble model and a browser extension to remove spam comments by altering CSS.

Various word representation methods, including TF-IDF vectorizer [12], Bag of Words (BoW) [18] , and word2vec [19], have been applied to extract features from text in previous studies [20], [21], [22].

The application of neural networks extends across a myriad of fields, revolutionizing the way we approach complex problems and processes. For example, in the realm of healthcare, neural networks contribute to medical image analysis, disease diagnosis, and drug discovery, enhancing precision and efficiency [23], [24]. And in the realm of agriculture, the application of neural networks is transforming traditional practices and ushering in a new era of precision farming and resource management. Neural networks are being utilized to analyze vast amounts of agricultural data, ranging from weather patterns and soil conditions to crop health and yield predictions. [19].

Within the realm of cybersecurity, the application of neural networks has emerged as a pivotal area of study and implementation. Researchers and security experts are delving into the intricacies of utilizing artificial intelligence, particularly neural networks, to bolster defenses against evolving cyber threats [26], [27].

More recently, Neural networks have acquired favor in tackling spam on social networking sites. For instance, [28] research paper proposed a tweet-level spam recognition architecture based on convolutional neural networks. In other study [29] , on two social network datasets, cost-sensitive ensemble learning approaches with regularized deep neural networks was used in the proposed approach to detect spam messages. Moreover, [30] paper presented a deep learning-based architecture that incorporates knowledge bases such as WordNet and ConceptNet to provide semantic information to word representation.

What distinguishes the suggested approach from earlier researches is the use of the ALBERT model for word representation. This eliminates the need for extra knowledge bases, streamlines the process, and reduces complexity. Furthermore, the model efficiently solves the "out of vocabulary" problem by incorporating previously viewed text during training. This sets it apart from typical word embedding methods.
Moreover, the proposed model, which uses the tokenization strategy in the ALBERT model, is versatile in accepting any text. It was tested on three benchmark datasets, with comparable performance based only on text content.

3. METHODOLOGY

3.1. Used dataset

The experiments utilized three distinct benchmark datasets. The initial dataset, referred to as the Twitter dataset and introduced by Chen et al. in [31], originally comprised 100,000 tweet IDs with corresponding labels. To gather tweet information based on these IDs, a python script-based crawler was implemented. However, only 58,159 tweets were successfully obtained, as other tweets had been deleted either by Twitter or users. Due to the dataset's imbalanced nature, data augmentation was applied to enhance the classifier's performance. Subsequently, the dataset was divided into an 80% training set and a 20% testing set.

The second dataset originated from YouTube comments, as utilized in [18]. This dataset had five different sorts of comments, combining spam and normal comments. Finally, the SMS spam dataset, the third dataset, can be found in the UCI repository. Each dataset is fully described in Table 1.

Table 1. datasets summary

Dataset	The amount of spam	The amount of non-spam	Total number of messages
Twitter	38205	27822	66027
YouTube	941	935	1876
SMS spam	747	4827	5574

3.2. The proposed Model

In this work, we have innovatively combined the Bidirectional Long Short-Term Memory (BLSTM) neural network architecture with the ALBERT pre-trained model to address the challenge of spam detection in social networks. The proposed model comprises two distinct phases: the initial phase involves data preparation and feature extraction, while the subsequent phase focuses on classification. Each of these phases will be comprehensively discussed in the upcoming sub-sections.

3.2.1. Data preparation and feature extraction phase

The pivotal stages in constructing precise deep learning models are data cleaning, processing, and feature selection. Dealing with the condensed text extracted from social networking sites necessitates transforming the text into a more manageable format to enhance the performance of classification algorithms. Accordingly, several pre-processing steps have been implemented on the utilized datasets. These steps involve removing hyperlinks, punctuation, emojis, and special characters, as well as deleting stop phrases and switching the text to lowercase. The execution of data pre-processing was facilitated through the utilization of a purpose-built Python script crafted for this specific objective.

Feature extraction

Feature extraction is a technique that reduces the number of features in a dataset by generating new features from the existing ones. Its purpose is to transform the current features into a lower-dimensional space, thereby decreasing the overall computational cost, training time, and enhancing the model's accuracy. We adopted the ALBERT model as a feature extractor within the recommended deep neural network model in our research.

AlBERT model:

The ALBERT pre-trained model serves as a lightweight version of the renowned BERT model [5]. Given the computational intensity of the BERT base model, consisting of 110 million parameters, there was a demand for a more resource-efficient variant. ALBERT, with 12 million parameters, 768 hidden layers, and 128 embedding layers, achieves a reduction in parameters while maintaining high performance.

ALBERT leverages two important strategies to achieve this parameter reduction. The first method is cross-layer parameter sharing, in which only the first encoder's parameter is learned and applied evenly across all encoders. This avoids parameter expansion as the network's depth increases. The second technique is factorized embedding parameterization, which uses factorization to lower the embedding layer from 768 to 128 layers. The size of the hidden layers is separated from the size of the vocabulary embedding by dividing the huge vocabulary embedding matrix into two smaller matrices, allowing for an increase in hidden size without considerably growing the parameter size of vocabulary embeddings. Beyond its lightweight nature, ALBERT distinguishes itself from BERT through Sentence Order Prediction (SOP). SOP involves classifying whether two given sentences are in the correct order, in contrast to BERT's next sentence prediction (NSP) during training. NSP entails the model predicting if the second sentence is the next in the original text.

The present study employed the ALBERT pre-trained model for feature extraction because to its tokenization strategy. To handle words that are not in the dictionary, the ALBERT tokenizer uses a data-driven methodology. It is built on a Sentence Piece tokenizer, which is a sub-word level tokenizer. Its byte-level methodology sets it apart from existing tokenization approaches that employ [UNK] or [OOV] tokens by enabling the encoding of complicated characters, like emojis, by representing them as combinations of many bytes. In order to increase the model's efficiency, ALBERT presents parameter reduction strategies. It makes use of cross-layer parameter sharing and shrinks the size of the concealed layers. Let X be the input text, shown as a word sequence. An embedding layer converts every word into a high-dimensional vector. If *E* represents the embedding layer, then.

$$E(X)= [\ x_1,\ x_2,...,\ x_n]\ ,$$

Where *n* is the length of the sequence.

ALBERT employs a transformer architecture, consisting of multiple encoder layers. Each encoder layer consists of self-attention mechanisms and feedforward neural networks.

For a given layer *l*, the output H_l can be represented as

$$H_l=\text{TransformerLayer}\ (H_{l-1}),$$

where H_{l-1} is the output of the previous layer.

The model is able to assign varying weights to words in a sequence according to how relevant they are to one another thanks to the self-attention mechanism. The self-attention output is passed through a feedforward neural network to capture non-linear relationships.

Each sub-layer output is normalized and passed through a residual connection, aiding in the stable training of deep networks.

3.2.2. Classification phase

The deep learning techniques have been leveraged in this study to execute the classification task within the proposed model, with a specific emphasis on a neural network architecture based on the recurrent neural network (RNN). RNN networks' main advantage is their capacity to remember information from previous calculations and use it in the current one. As such, context dependencies in variable length inputs can be effectively modeled by RNN-based models, allowing for an efficient input composition. Nevertheless, architectures intended to tackle this difficulty have been introduced as a result of the well-documented vanishing gradient problem. Long Short-Term Memory is one such architecture (LSTM). Although LSTMs have demonstrated efficacy, their ability to interpret context in a single direction is limited. Bi-directional Long Short-Term Memory networks, commonly referred to as BLSTM (or Bi-LSTMs), have been developed to overcome this limitation. Unlike traditional LSTM models that process input sequences in only one direction, BLSTMs utilize two separate models during training. The first model learns from the input sequence in its original order, while the second model processes the sequence in reverse. This approach effectively captures contextual information from both the past and the present, ensuring that each segment of an input sequence benefits from comprehensive information.

Overfitting is a common challenge in deep neural networks, manifesting when a model excels on training data but performs poorly on testing data. Essentially, overfitting occurs when a model endeavors to perfectly match the training data, resulting in memorization of data patterns, noise, and random fluctuations. To mitigate overfitting, various techniques can be employed. One straightforward approach involves the utilization of dropout. Dropout serves as a regularization technique designed to prevent neural networks from succumbing to overfitting. The fundamental concept is to randomly deactivate some neurons during training. Standard dropout involves zeroing out a fraction of nodes in every layer before computing the next layer throughout each training iteration. This random dropout prevents the network from becoming overly reliant on certain nodes, allowing for more robust generalization to previously unknown data.

3.2.3. Tuning the parameters of the proposed model

Hyperparameters play a pivotal role in shaping the performance of deep learning models, underscoring the importance of optimizing them to attain optimal results. Various strategies exist to tackle hyperparameter optimization, each offering distinct advantages.

Grid Search, the most straightforward approach, partitions the hyperparameter space into a discrete grid and exhaustively evaluates each combination via cross-validation to gauge performance metrics. Random Search, akin to Grid Search, samples a random subset of grid points, often proving more effective, especially when optimizing with a limited number of hyperparameters.

Bayesian Optimization takes a probabilistic approach by constructing a model that maps hyperparameter values to performance metrics on a validation set. This method iteratively refines its understanding of the hyperparameter space, making informed decisions to explore promising regions efficiently.

Gradient-Based optimization involves computing gradients with respect to hyperparameters and optimizing them using gradient descent. Leveraging gradient information allows for efficient navigation of the hyperparameter space towards regions of improved performance.

Evolutionary Optimization harnesses evolutionary algorithms to explore the hyperparameter space. By emulating principles of natural selection, this method evolves a population of hyperparameter configurations over successive generations, favoring those exhibiting higher performance.

In our study, we opted for the random search method to fine-tune the hyperparameters of our proposed model due to its computational efficiency. Employing this approach, we optimized multiple hyperparameter values and selected the model yielding the most promising results. Detailed insights into the tuned hyperparameters and configurations are elaborated upon in the experimental results section. Figure 1 illustrates the suggested model's block diagram and Algorithm 1 illustrates the detailed process of the proposed method.

Algorithm 1: Spam detection Algorithm

Input: *Raw text data R from social networking sites*
Output: *Probability score indicating the likelihood of spam.*
1: For *each raw in R* ***do****:*
Clean_Text=Remove_URL (raw)
Clean_Text = Remove_Punctioation (Clean_Text)

```
  Clean_Text = Remove_Emoji (Clean_Text)
  Clean_Text = Remove_Char (Clean_Text)
  Clean_Text = Remove_Stop_Words(Clean_Text)
  Clean_Text = Convert_To_Lower_Case (Clean_Text)
End for
2:  Function extract_features(Clean_Text):
  albert_tokenizer = AlbertTokenizer.from_pretrained('albert-base-v2')
  albert_model = AlbertModel.from_pretrained('albert-base-v2')
  tokenized_text = albert_tokenizer(Clean_Text)
  output = albert_model(tokenized_text)
return output
3:  Function hyperparameter_tuning():
  hyperparameters = {num_blstm_layers,dropout_existence, dropout_value , learning_rate}
 best_hyperparameters = random_search(hyperparameters)
return best_hyperparameters
4:Function Split_dataset (output,Ratio,train,test)
return train,test
5:Train model (model,train, best_hyperparameters)
6:Evaluate_model(model,test, best_hyperparameters)
End
```

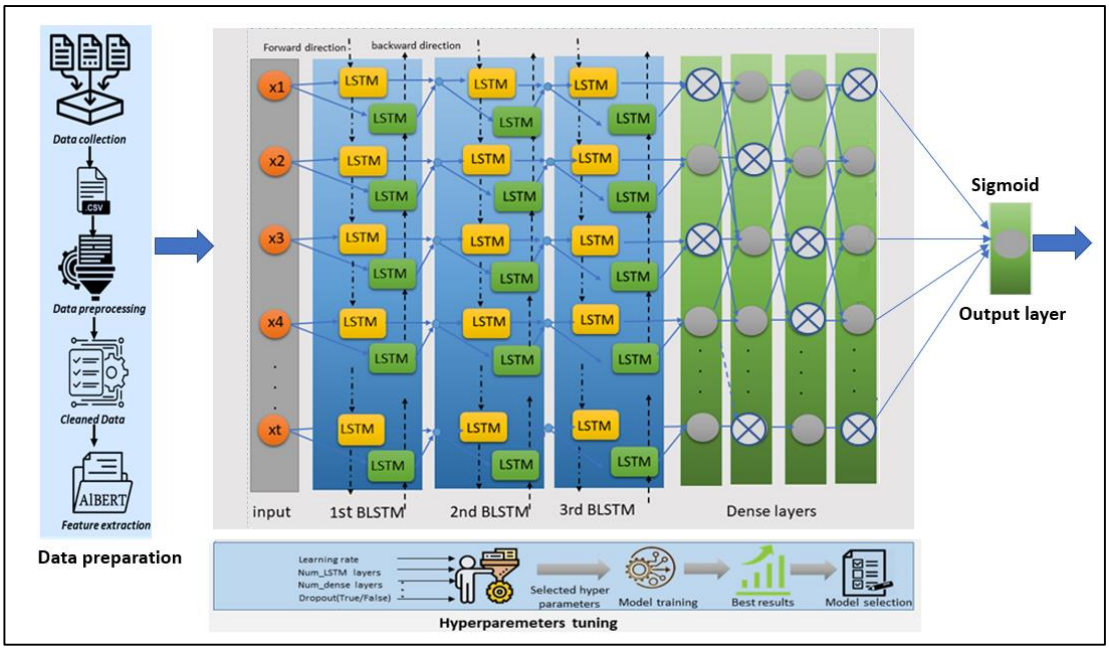


Figure 1. The block diagram of the suggested model

4. Experimental Results

All studies were conducted out with Google Colaboratory, Python3 and GPU runtime.

4.1. Evaluation metrics

The accuracy, Recall, F1-score, and precision metrics have been used as evaluation metrics in this study.

The accuracy metric is the main metric used to evaluate the model's performance. It describes the number of correct predictions over all predictions and can be computed by the following equations:

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TN} + \boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{TN} + \boldsymbol{FP} + \boldsymbol{FN}} \quad (\mathbf{3})$$

Precision is a measure of how many of the positive predictions made are correct (true positives) and can be computed by the following formula

$$\boldsymbol{Precision}\ (\boldsymbol{P}) = \boldsymbol{TP}/(\boldsymbol{TP} + \boldsymbol{FP}) \quad (\mathbf{4})$$

Recall, expressed in Equation 3, quantifies the classifier's accuracy in correctly identifying positive cases among all positive instances in the dataset.

$$\boldsymbol{Recall\ (R) = \frac{TP}{TP + FN}} \qquad (\mathbf{5})$$

F1-Score is a measure combining both precision and recall and can be computed by the following formula:

$$\boldsymbol{F1 - Score = (2 * P * R)/(P + R)} \quad (\mathbf{6})$$

Where TP stands for true positives, TN for true negatives, FP for false positives, and FN for false negatives. Here, TP stands for the total number of messages correctly classified as spam. TN displays the number of messages that have been correctly classified as non-spam. FP represents the number of non-spam messages that are labeled as spam, while FN represents the number of spam messages that are labeled as non-spam. It is worth mentioning that in the evaluation of our spam detection model, we employed a range of metrics to comprehensively assess its performance across different datasets. While metrics such as accuracy, precision, and recall provide valuable insights into overall model effectiveness, we acknowledge the significance of considering class imbalance, particularly in the context of the SMS spam dataset.

Given the imbalance between spam and non-spam classes in the SMS dataset, we recognized the importance of using evaluation metrics that are robust to such disparities. As such, we employed the F1-score as a primary evaluation metric alongside other standard metrics. The F1-score combines precision and recall, offering a balanced assessment of model performance that is less affected by class imbalance.

4.2. Hyperparemeters tuning

Herein this study, we used a random search strategy in order to tune each of the following hyperparameters: the number of BLSTM layers, dropout existence, dropout value, number of dense layers, number of neurons in each of dense layers, learning rate, and weight initializer, and activation functions. The value range of each of these hyperparameters is illustrated in Table 2. The random search algorithm was tuned to do 20 trials with four executions in each trial, and in each execution, the model will be trained for 5 epochs. The validation accuracy has been monitored during the algorithm execution and we have defined our objective function to select the maximum value of the validation accuracy. Table 3 shows the configuration parameters required to adopt the random search technique.

Table 2 Experimented hyperparameteres values during tuning

Hyperparameter	**Value range**
Number of the BLSTM layers	1-6
Dropout existence	[True, False]
Dropout value	[0.1,0.2,0.3,0.4,0.5]
Number of the Dense layers	1-4
Number of neurons in each the Dense layer	[32, 64, 96, 128]
Learning rate	[0.01, 0.001, 0.0001]
weight initializer	['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform']
Activation functions	[relu, tanh]

Table 3 Configuration settings used in random search

Number of trials	20
Executions per trial	4
Epochs per trial	5

In the final phase, the best model was identified, achieving optimal validation accuracy across the three datasets. This selected model comprises three Bidirectional Long Short-Term Memory layers, each with 256 units, followed by four dense layers, each

consisting of 32 units. A dropout is introduced, and the output layer concludes the architecture. The chosen dropout value is 0.2, the weight initializer is set to 'he_uniform', and the Rectified Linear Unit (ReLU) function serves as the activation function in the dense layers.

Figure 2 is a part of the parallel coordinates view illustrating the hyperparameter fine-tuning process using random search. The tuned hyperparameters are shown on the horizontal axes, while the corresponding selected values are shown on the vertical axes. This visual representation illustrates the results of the random search hyperparameter tuning process. Each line in the plot represents a different hyperparameter configuration, with the axes corresponding to the hyperparameters being tuned. The lines show how each hyperparameter configuration performs across multiple evaluation metrics. This visualization allows for a comprehensive understanding of how different combinations of hyperparameters impact model performance. Figure 3 also shows a preview of the random search algorithm's scatter plot matrix view, where each dot represents a trial. This visual depiction provides a comprehensive view of the relationships between pairs of hyperparameters during the random search process. Each point in the scatter plot matrix represents a hyperparameter configuration, with different plots showing the relationships between different pairs of hyperparameters. This visualization aids in identifying any correlations or patterns between hyperparameters, which can inform further optimization strategies. Figure 4 depicts the accuracy and loss plots for the best trial recorded during hyperparameter tweaking, with the three trials of the best experiment marked by the lines on the charts. This visual representation showcases the results of the hyperparameter tuning experiments conducted specifically on the YouTube dataset. Chart a illustrates the validation accuracy achieved by different hyperparameter configurations, while chart b displays the corresponding validation loss. These charts provide insights into how different hyperparameter settings influence the model's performance on the YouTube dataset, allowing for informed decision-making in selecting the optimal hyperparameter configuration. Table 4 summarizes the results of the random search in detail. The table presents the results of the hyperparameter tuning experiments conducted specifically on the YouTube dataset. Each row represents a different trial ID, with columns detailing the specific hyperparameters explored during the tuning process. The columns include information such as the dropout rate, the number of dense layers, the number of units in each dense layer, the number of BLSTM layers, the learning rate, and metrics such as train accuracy, validation accuracy, train loss, and validation loss.

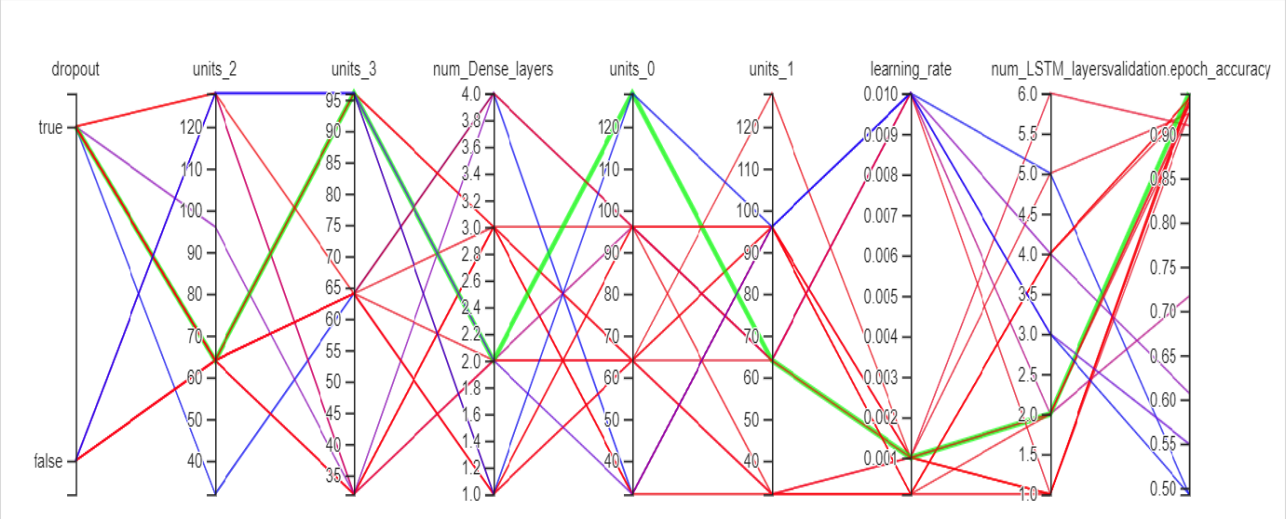


Figure 2. The parallel coordinates view for the conducted random search hyperparametres fine tuning

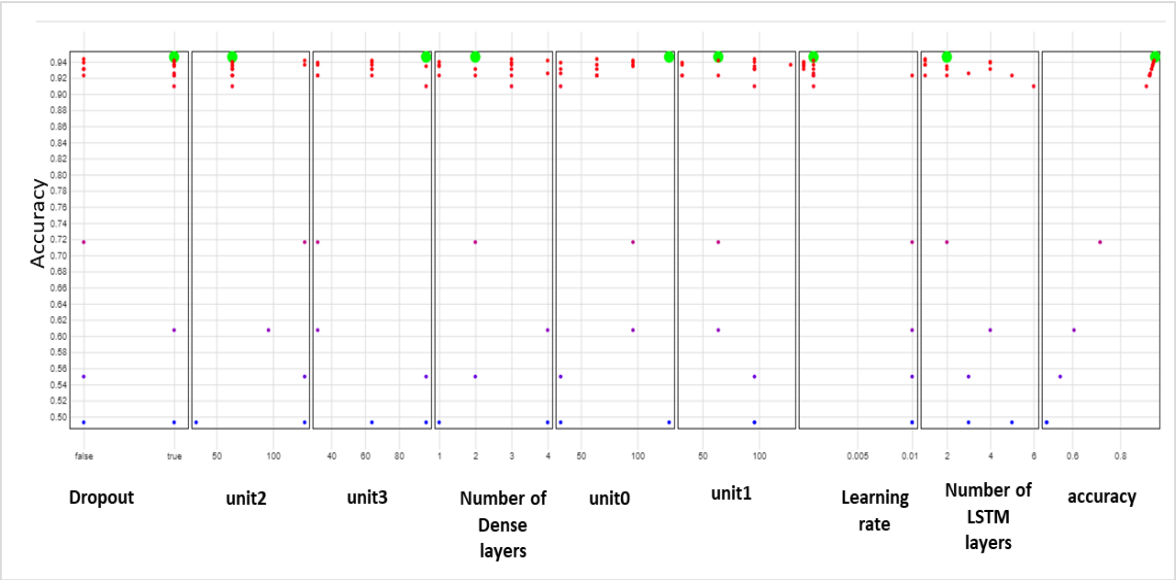


Figure 3. The scatter plot matrix view for the conducted random search

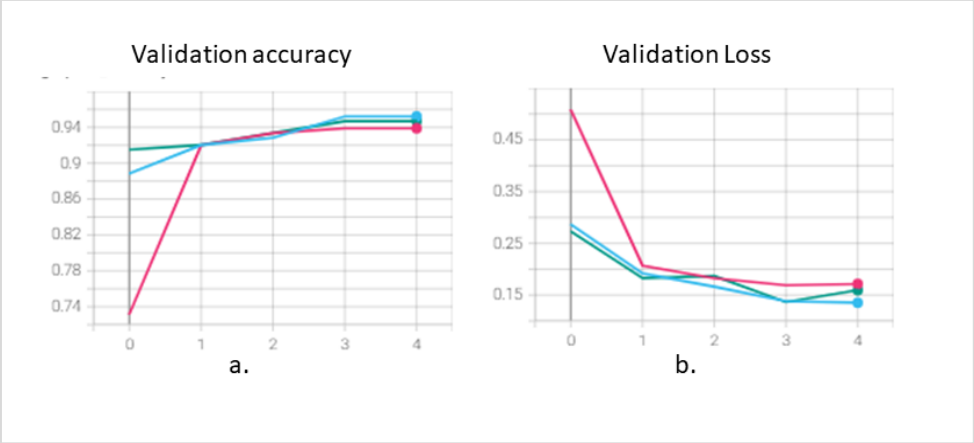


Figure 4 The best hyper parameters tuning experiments' results on YouTube dataset. chart a. represents the validation accuracy and chart b. represents validation loss

Table 4 hyper parameters tuning experiments' results on YouTube dataset

Trial ID	dropout	# of Dense layers	units_0	Unit_1	units_2	units_3	# of BLSTM layers	Learning rate	Train accuracy	validation accuracy	Train loss	Validation Loss
0	true	4	32	64	-	-	3	0.001	0.92516	0.92553	0.19694	0.19296
1	true	4	32	96	32	64	5	0.0001	0.50056	0.49291	0.70626	0.70383
2	true	3	32	96	64	-	6	0.001	0.87653	0.90957	0.34381	0.2838
3	false	3	64	32	64	-	1	0.01	0.95692	0.94326	0.11603	0.18043
4	true	3	64	64	64	-	5	0.01	0.92294	0.92287	0.21103	0.19901
5	false	2	64	64	-	-	4	0.01	0.94559	0.93085	0.15676	0.19206
6	false	3	32	32	64	-	2	0.001	0.94448	0.93085	0.15121	0.19475
7	false	3	32	32	64	-	4	0.01	0.94426	0.93883	0.16144	0.17078
8	true	3	96	64	128	-	1	0.001	0.94848	0.93617	0.14338	0.19642
9	true	1	64	-	-	-	1	0.001	0.94559	0.93617	0.14904	0.17147
10	true	4	96	32	96	32	4	0.01	0.61115	0.60727	0.64572	0.64864
11	false	2	96	96	-	-	2	0.0001	0.69776	0.71631	0.58824	0.58255
12	true	2	128	64	-	-	2	0.001	0.94581	0.94592	0.15504	0.15537
13	**true**	**4**	**32**	**32**	**32**	**32**	**3**	**0.001**	**0.9853**	**0.9521**	**0.0470**	**0.1944**
14	true	1	96	-	-	-	4	0.001	0.94715	0.93972	0.14341	0.19378
15	true	4	96	64	128	64	1	0.001	0.95514	0.94149	0.12171	0.1697
16	true	1	64	-	-	-	2	0.001	0.92449	0.92287	0.26203	0.26102
17	true	1	96	-	-	-	2	0.001	0.95203	0.9344	0.13678	0.18747
18	false	2	32	-	-	96	3	0.01	0.55008	0.54965	0.66753	0.65661
19	false	1	128	-	-	-	3	0.01	0.501	0.49291	0.69329	0.69345

4.3. Results and discussion

After employing the random search approach to determine the optimum hyperparameters that can be used for constructing a deep learning model that can give us the best results for the previously mentioned three datasets, we have gotten the model that has a structure illustrated in table 5. Then, after training the model for 10 epochs, the best results are reached as shown in Table 6. Furthermore, Figure 3 illustrates confusion matrix obtained by applying the proposed model on the used datasets.

According to the results, our proposed approach, which combines contextualized embedding (using ALBERT pretrained model as a feature extractor) with an BLSTM-based deep learning architecture, consistently outperforms the benchmark datasets. As depicted in Table 6, our model achieved impressive accuracies, surpassing 98%, 95%, and 99% on the Twitter, YouTube, and SMS datasets, respectively.

The tuning of hyperparameters proves instrumental in optimizing the performance of our proposed model. In this study, through meticulous hyperparameter tuning, we found that optimal performance is achieved by keeping the number of BLSTM layers at or below 3. Additionally, the optimal learning rate was determined to be 0.001, and a dropout value of 0.2 consistently led to superior results. In the dense layers, the ReLU function was chosen as the activation function during experiments. Given the binary classification nature of our problem, the default activation function in the output layer was the sigmoid function.

Finally, based on the Comparison study results on Table 7, the experimental results unequivocally demonstrated that the ALBERT-BLSTM based model surpassed existing methods from reference studies. These findings underscore the critical role of word representation in significantly improving classification performance, especially when dealing with short texts in social networking sites.

Table 5. The structure of the best model

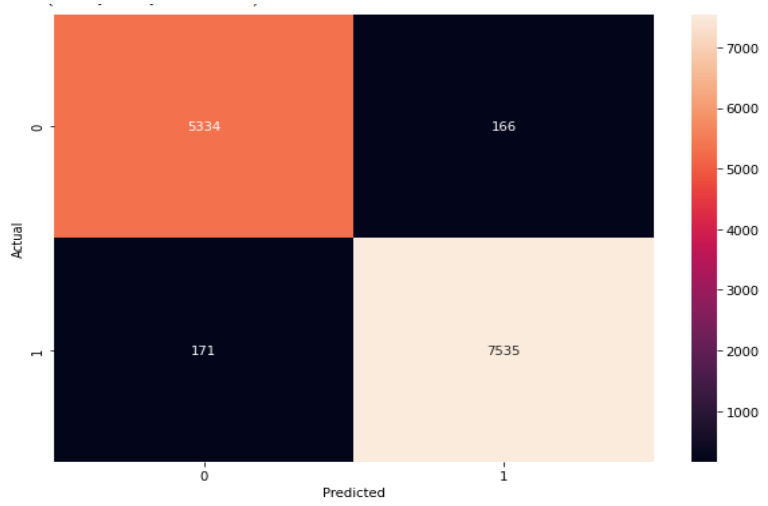
Parameter	**Value**
# of BLSTM layers	3
# of units in BLSTM layers	256→256→256
# of Dense layers	4
# of units in Dense layers	64→32→32→32
Activation function	ReLu
Weight initializer	he_uniform
Dropout	True
Dropout value	0.2
Output activation function	Sigmoid
Learning rate	0.001
Optimizer	Adam
Loss function	Binary crossentropy
Batch size	128
# of epochs	10

Table 6. Final results on the three datasets

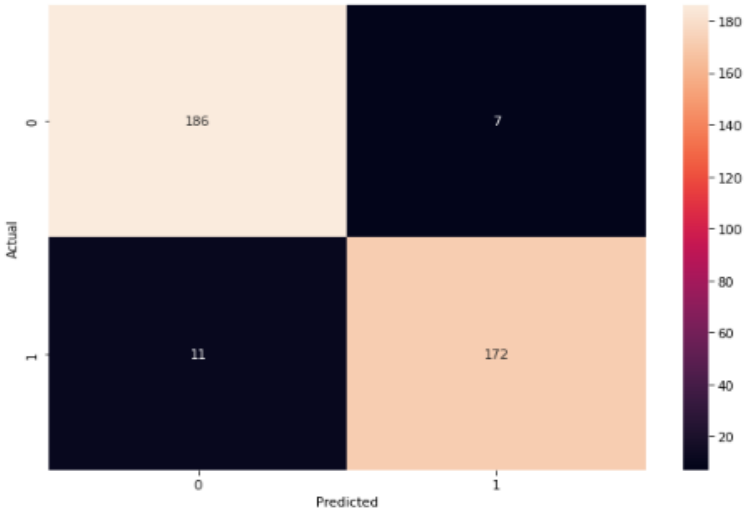
Dataset	**Accuracy**	**Precision**	**Recall**	**F1-score**
Twitter	0.982	0.98	0.98	0.98
YouTube	0.9548	0.96	0.94	0.95
SMS	0.9912	0.98	0.97	0.97

Table 7. Comparision with other Studies

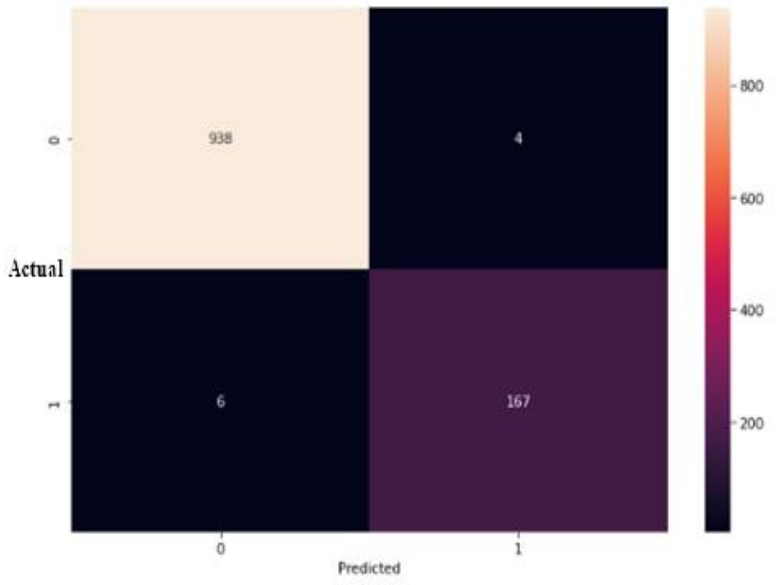
Study	Dataset	Accuracy
[28]	Twitter	0.975
Barushka et. al [29]	Twitter	0.956
[20]	Twitter	0.978
Jain et.al [32]	Twitter	0.9343
Our method	Twitter	0.982
Ghanem and Erbay [20]	SMS	0.99
Jain et.al [32]	SMS	0.978
Our method	SMS	0.99
Nagaraj et.al [33]	YouTube	0.93
[20]	YouTube	0.9521
Our method	YouTube	0.9548



a.



b.



c.

Figure 5. The confusion matrices obtained by applying the proposed model on the respective datasets: a. The confusion matrix for the Twitter dataset. b. The confusion matrix for the YouTube dataset. c. The confusion matrix for the SMS dataset.

5- CONCLUSION

In conclusion, the pervasive presence of spam messages on social networks poses a significant challenge, particularly in the realm of short text classification. The inherent brevity and ambiguity of text within social media platforms add layers of complexity to the task of spam detection in natural language processing. Acknowledging the pivotal role of robust text representation, our study introduced an innovative approach that integrates contextualized models and a deep learning architecture.

To Avoid the limitations in conventional word embedding models, characterized by struggles with data sparsity, out-of-vocabulary challenges, and context independence, our research proposed a novel methodology for spam detection. Leveraging contextualized word representations derived from the pre-trained ALBERT model, we augmented the performance of a stacked Bidirectional Long Short-Term Memory (BLSTM) based neural network architecture. This approach demonstrated efficacy in accurately classifying messages as either spam or non-spam, effectively navigating the intricate nuances of short text in social networks.

Furthermore, hyperparameter tuning played a crucial role in optimizing the proposed model's performance across three benchmark datasets. The utilization of the random search algorithm facilitated the fine-tuning of multiple hyperparameters, including the number of BLSTM layers, dense layers, neurons in each dense layer, dropout presence, activation function, and learning rate.

Looking ahead, the proposed method holds promise for applications in various trending studies such as sentiment analysis, authorship identification, hate speech detection, review spam detection, and more. Future endeavors could delve into these domains to further validate the versatility and effectiveness of the introduced approach.

ACKNOWLEDGMENT

The authors are grateful to the UCI repository, Chen et al. [26], Alberto et al. [13], for contributing their datasets.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare that there is no conflict of interest.

REFERENCES

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[2] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[3] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," arXiv preprint arXiv:1607.01759, 2016.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[5] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," arXiv preprint arXiv:1909.11942, 2019.

[6] A. Al-Zoubi, J. Alqatawna, H. Faris, and M. A. Hassonah, "Spam profiles detection on social networks using computational intelligence methods: the effect of the lingual context," J Inf Sci, vol. 47, no. 1, pp. 58–81, 2021.

[7] D. Niranjan Koggalahewa, Y. Xu, and E. Foo, "Spam detection in social networks based on peer acceptance," in Proceedings of the Australasian Computer Science Week Multiconference, 2020, pp. 1–7.

[8] K. S. Adewole, T. Han, W. Wu, H. Song, and A. K. Sangaiah, "Twitter spam account detection based on clustering and classification methods," J Supercomput, vol. 76, pp. 4802–4837, 2020.

[9] A. Kumar, M. Singh, and A. R. Pais, "Fuzzy string matching algorithm for spam detection in Twitter," in Security and Privacy: Second ISEA International Conference, ISEA-ISAP 2018, Jaipur, India, January, 9–11, 2019, Revised Selected Papers 2, Springer, 2019, pp. 289–301.

[10] O. ÇITLAK, M. DÖRTERLER, and İ. DOGRU, "A hybrid spam detection framework for social networks," Politeknik Dergisi, p. 1, 2023.

[11] C. Kumar, T. S. Bharti, and S. Prakash, "A hybrid Data-Driven framework for Spam detection in Online Social Network," Procedia Comput Sci, vol. 218, pp. 124–132, 2023.

[12] A. Aziz, C. F. M. Foozy, P. Shamala, and Z. Suradi, "YouTube spam comment detection using support vector machine and K–nearest neighbor," Indones. J. Electr. Eng. Comput. Sci, vol. 12, no. 2, p. 612, 2018.

[13] A. Ali and M. Amin, "An Approach for Spam Detection in YouTube Comments Based on Supervised Learning," 2016.

[14] A. T. Kabakus and R. Kara, "'TwitterSpamDetector': a spam detection framework for Twitter," International Journal of Knowledge and Systems Science (IJKSS), vol. 10, no. 3, pp. 1–14, 2019.

[15] P. Nagaraj, K. M. Sudar, P. Thrived, P. G. K. Reddy, S. B. Babu, and P. S. R. Krishna, "Youtube Comment Spam Detection," in 2023 International Conference on Computer Communication and Informatics (ICCCI), IEEE, 2023, pp. 1–6.

[16] H. Valpadasu, P. Chakri, P. Harshitha, and P. Tarun, "Machine Learning based Spam Comments Detection on YouTube," in 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), IEEE, 2023, pp. 1234–1239.

[17] L. Shabadi, P. Srikanth, V. Kumar, and U. Kashyap, "Youtube Spam Detection Scheme Using Stacked Ensemble Machine Learning Model," in 2023 International Conference on Network, Multimedia and Information Technology (NMITCON), IEEE, 2023, pp. 1–7.

[18] T. C. Alberto, J. V Lochter, and T. A. Almeida, "Tubespam: Comment spam filtering on youtube," in 2015 IEEE 14th international conference on machine learning and applications (ICMLA), IEEE, 2015, pp. 138–143.

[19] T. Wu, S. Liu, J. Zhang, and Y. Xiang, "Twitter spam detection based on deep learning," in Proceedings of the australasian computer science week multiconference, 2017, pp. 1–8.

[20] R. Ghanem and H. Erbay, "Spam detection on social networks using deep contextualized word representation," Multimed Tools Appl, vol. 82, no. 3, pp. 3697–3712, 2023.

[21] R. Ghanem and H. Erbay, "Context-dependent model for spam detection on social networks," SN Appl Sci, vol. 2, pp. 1–8, 2020.

[22] R. Ghanem, H. Erbay, and K. Bakour, "Contents-Based Spam Detection on Social Networks Using RoBERTa Embedding and Stacked BLSTM," SN Comput Sci, vol. 4, no. 4, p. 380, 2023.

[23] H. Bakir and G. Tarihi, "Using Transfer Learning Technique as a Feature Extraction Phase for Diagnosis of Cataract Disease in the Eye."

[24] H. BAKIR, S. OKTAY, and E. TABARU, "DETECTION OF PNEUMONIA FROM X-RAY IMAGES USING DEEP LEARNING TECHNIQUES," Journal of Scientific Reports-A, no. 052, pp. 419–440, Mar. 2023, doi: 10.59313/jsr-a.1219363.

[25] H. Bakır, "Evaluating the impact of tuned pre-trained architectures' feature maps on deep learning model performance for tomato disease detection," Multimed Tools Appl, pp. 1–22, 2023.

[26] H. Bakır and R. Bakır, "DroidEncoder: Malware detection using auto-encoder based feature extractor and machine learning algorithms," *Computers and Electrical Engineering*, vol. 110, p. 108804, 2023.

[27] E. Doğan and H. BAKIR, "Hiperparemetreleri Ayarlanmış Makine Öğrenmesi Yöntemleri Kullanılarak Ağdaki Saldırıların Tespiti," *in International Conference on Pioneer and Innovative Studies*, 2023, pp. 274–286.

[28] S. Madisetty and M. S. Desarkar, "A neural network-based ensemble approach for spam detection in Twitter," *IEEE Trans Comput Soc Syst*, vol. 5, no. 4, pp. 973–984, 2018.

[29] A. Barushka and P. Hajek, "Spam detection on social networks using cost-sensitive feature selection and ensemble-based regularized deep neural networks," *Neural Comput Appl*, vol. 32, pp. 4239–4257, 2020.

[30] G. Jain, M. Sharma, and B. Agarwal, "Spam detection in social media using convolutional and long short term memory neural network," *Ann Math Artif Intell*, vol. 85, no. 1, pp. 21–44, 2019.

[31] W. Chen, C. K. Yeo, C. T. Lau, and B. S. Lee, "A study on real-time low-quality content detection on Twitter from the users' perspective," PLoS One, vol. 12, no. 8, p. e0182487, 2017.

[32] G. Jain, M. Sharma, and B. Agarwal, "Spam detection in social media using convolutional and long short term memory neural network," *Ann Math Artif Intell, vol. 85, no. 1, pp. 21–44, 2019.*

[33] P. Nagaraj, K. M. Sudar, P. Thrived, P. G. K. Reddy, S. B. Babu, and P. S. R. Krishna, "Youtube Comment Spam Detection," *in 2023 International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, 2023, pp. 1–6.