Expert Systems With Applications 217 (2023) 119594

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Hybrid ensemble framework with self-attention mechanism for social spam detection on imbalanced data

Sanjeev Rao *, Anil Kumar Verma, Tarunpreet Bhatia

Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab, India

ARTICLE INFO

Keywords:
Data resampling
Machine Learning
Natural Language Processing
Online Social Network
Self-Attention
Spam

ABSTRACT

Cybercriminals use social media platforms to disseminate spam, misleading facts, fake news, and malicious links. Blocking such deceptive social media spam is essential. However, extracting relevant features from social networks is challenging due to privacy and time constraints. Traditional frequency-based word representation techniques are time-consuming and inefficient in producing contextual word vectors. Word embeddings and deep learning models have recently shown good results in text classification. Also, most existing approaches assumed balanced class distribution, which is false for most real-world datasets. In this paper, an attempt is made to advance the performance of the social spam detection system by leveraging dataset balancing, advanced word embedding techniques, machine learning, and deep learning approaches with the self-attention mechanism. In the proposed framework, the datasets are balanced using NearMiss and SmoteTomek techniques to feed several machine-learning models. Later, the baseline ML models and proposed voting-based ensemble models are evaluated on imbalanced and balanced datasets. For the proposed deep learning-based hybrid approaches, embeddings are generated using GloVe and FastText word embeddings on the balanced *combined dataset* and passed into the deep neural network comprised of Conv1D and Bi-directional recurrent neural network layers with the self-attention mechanism for improved context understanding and effective results. This study examines hybrid approaches for detecting social spam using imbalanced social network data and picks the optimum combination. Besides, Machine learning ensembles, word embeddings, deep learning with hyper-parameter optimization, and a self-attention method are compared thoroughly. Experiments and comparisons with other techniques show that the proposed hybrid framework with deep learning-based approaches achieves better performance.

1. Introduction

Masses use online social networks (OSNs) to post text messages, news, consumer interactions, advertising, product reviews/promotions, etc. However, cybercriminals, bots, hackers, and third-party companies use public information to achieve nefarious goals. Spammers post malicious links, spam messages, porn, monetary claim content, and fake news from forged accounts utilizing automated tools to lure legitimate users. To grab the attention of online users, spammers choose trending keywords, tiny URLs, and hashtags in their messages that redirect online users to phishing sites to steal their data. Rumors and false information are disseminated quickly to millions of OSN users as huge campaigns, and social movements may conflict against any country or authority. The new era of innovation has introduced individuals to additional opportunities such as educational, informational, social networking, and advocacy through social media platforms, which can occasionally result in false communications, rage, prejudice, and hatred (Rao et al., 2020a, 2020b).

In the era of the internet and e-commerce, buyers routinely submit online evaluations of their purchases. Insights into the products or services are valuable for new customers. Online merchants can utilize review data to improve products, services, marketing strategies, and competitive analyses. However, spammers, bots, paid reviewers and dishonest businesses propagate a bulk of fake reviews to downgrade/uplift the reputation of particular products/businesses and sway online users' perceptions (Fei et al., 2017; Heydari et al., 2015; Hussain et al., 2019).

Besides, the fast spread of coronavirus illness has become one of the most devastating tragedies of the twenty-first century, affecting people globally. COVID-19, a pandemic, has revealed another ailment known as

* Corresponding author.
E-mail addresses: srao20_phdp17@thapar.edu, er.sanjeevrao@gmail.com (S. Rao), akverma@thapar.edu (A.K. Verma), tarunpreet@thapar.edu (T. Bhatia).

https://doi.org/10.1016/j.eswa.2023.119594
Received 13 August 2022; Received in revised form 22 December 2022; Accepted 19 January 2023
Available online 23 January 2023

misinformation, which appears to have spread through 21st-century civilizations. The World Health Organization (WHO) invented the term 'infodemic' to represent modernity's pandemic of misinformation. The COVID-19 pandemic and infodemic have profoundly changed our lives. Recently, for COVID-19 pandemic solutions and complications, misconceptions appear in the form of misinformation about drugs, vaccinations, surgical masks, etc., on social media, causing negative affect on one's mental health and triggering new difficulties for the government if not discovered in time. Nowadays, fake news and conspiracy theories are observed on mainstream media and the internet, primarily social media, due to the politicized environment and other influencing factors (Cinelli et al., 2020; Gallotti et al., 2020). The New York Times studied Covid-19 disinformation, hoaxes, and conspiracy theories. They found one million instances of deception in 38 million English-language articles in 3 % of Covid-19 conversations. Sometimes healthcare misinformation was not shared with malicious intent but as a health risk. Few of these claims include herbal eye drops treating the infection, disposable face masks that may be cleaned with steam, etc. (Frenkel et al., 2020). Hence, it's important and challenging to spot fake content in constantly-evolving social media.

In the past decade, researchers have extensively studied the detection of spam, fake reviews, and fake news in OSNs. Still, existing machine learning (ML) based social spam detection frameworks/approaches are not robust and efficient enough to detect spam content and block spammer accounts. In a recent review of social spam detection techniques (Rao et al., 2021), it has been observed that most researchers have tested their spam detection approaches on small and balanced datasets only. The existing frameworks and approaches cannot achieve better performance on the imbalanced dataset, i.e., one class has inadequate proportions compared to the other classes. This issue, known as class imbalance, is considered one of the main issues in ML classification (Tolba et al., 2021). Most ML algorithms assume that the distribution of data samples across class labels is roughly proportional, making learning difficult since they are biased towards the majority group and predict the majority group's class in most cases. The class imbalance problem can be solved using data resampling techniques. Moreover, various feature representation techniques, word embedding techniques, and deep learning (DL) models have shown promising results. DL is used to learn tasks from another dataset using the pre-trained model that need not be trained from abrasion, thus significantly improving the model's performance by reducing training time.

This paper provides a comparative analysis of applying ML/DL models/approaches on imbalanced and balanced datasets. Initially, the class imbalance problem is solved using data resampling techniques. In addition, the performance of existing approaches can be enhanced by incorporating ensemble models, efficient pre-trained word embeddings, and improved DL models with the self-attention mechanism. The paper provides the following contributions to determine the most efficient approaches for detecting spam for imbalanced data:

- Data pre-processing and exploratory data analysis (EDA) is done on the larger corpus built from three benchmark datasets: *SMS spam, Ling spam,* and *Twitter fake news*.
- Dataset balancing is done using NearMiss (under-sampling technique) and SmoteTomek (hybrid under-sampling and over-sampling technique).
- Two word representation techniques, Bag of words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), are used to represent text into numbers.
- Proposed two ML-based voting ensemble models that leverage the predicted results of individual ML models.
- Proposed two DL-based hybrid approaches. In the first approach, GloVe pre-trained word embedding is fed into (Conv1D + Bi-GRU + Bi-LSTM), while in the second FastText pre-trained word embedding is fed into the same hybrid sequential model.
- Besides, the self-attention mechanism is added in the proposed sequential neural network to pay special attention to extract the most significant inputs and aggregating their representations using attention weights and bias to form output as a sentence vector.
- Comparative evaluation of proposed approaches with respect to the existing approaches is done.

The rest of the paper is structured as follows: Section 2 summarizes related work. Section 3 provides the meticulous methodology used for ML and DL-based proposed framework. Section 4 provides the experimentation and results. Section 5 presents comprehensive discussions and future directions. Lastly, Section 6 concludes the work.

2. Related work

Traditional approaches for spam detection, such as URL list-based techniques and honey-*net*-based systems, have several shortcomings, including data collection and portability, legitimate spammer IP addresses and small URLs, attribute discrepancy, adaptability, performance, and proficiency. Most of these shortcomings have been fixed by ML/DL-based approaches (Rao et al., 2021). The following section briefly summarizes the recent research on textual social spam detection using various feature extraction techniques and ML/DL-based approaches.

For ML/DL-based social spam detection, the raw data is first collected using OSN APIs to form a dataset. The raw data is pre-processed to reduce noise and missing data. Feature extraction approaches are applied to retrieve only dominant features for ML/DL models. Common feature extraction techniques are TF-IDF, BoW, Word2Vec, etc. In BoW, *n*-gram text features are extracted from a group of words from the textual content. Aiyar & Shetty (2018) have detected spam comments on YouTube using character grams, word grams, and several ML algorithms such as Naïve Bayes (NB), random forest (RF), and support vector machine (SVM). Eventually, they found character grams are better than word grams to measure spamminess in YouTube comments and recommended using Word2Vec word embedding for better results. Alberto et al. (2016) have distinguished between spammers and legitimate users using the SVM classifier on YouTube using social and anti-social behavior features. Gupta et al. (2018) developed an ML framework with a BoW feature extraction technique to detect spam tweets. Mehmood et al. (2018) have used TF-IDF text features to build a spam comment prediction model using ensemble learning. Ahmed et al. (2018) have used *n*-gram analysis with TF-IDF to detect opinion spam using Metaheuristic Optimization Algorithms (MOA) and ML models. In addition, they implemented six ML classifiers on the fake news dataset and found that accuracy improved with the increase in number of features and declined with an increase in the size of *n*-gram. Adding dimensionality reduction to ML models enhanced their performance. Most spam detection research relies on feature extraction methods such as *n*-gram, BoW, TF-IDF, and Word2Vec. Some commonly used supervised learning (SL) models for spam detection are NB, SVM, RF, and Decision Trees (DT). Kardaş et al. (2021) have tested the performance of several ML models, such as NB, SVM, LR, and neural networks, on the Twitter dataset using pre-processing and word representation techniques. Saumya & Singh (2018) have implemented an opinion mining framework using RF, gradient boost (GB), and SVM models to detect spam reviews on balanced datasets generated using Smote and Adaptive synthetic sampling (ADASYN) as data balancing techniques. Few other relevant studies have worked on the class imbalance issue and resolved the problem using under-sampling and over-sampling approaches. Some of the data resampling techniques used are random over-sampling (ROS), NearMiss, Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002), ADASYN, Smote with boosting (SmoteWB) (Ghaderi Zefrehi & Altınçay, 2020; Sağlam & Cengiz, 2022; Tolba et al., 2021; Wang et al., 2019). Some studies have used semi-supervised learning (SSL) techniques over supervised and

unsupervised to train unlabelled data (Liu et al., 2020; Sedhai & Sun, 2018; Singh & Batra, 2018; Zhang et al., 2016).

Some studies have improved spammer detection systems using word embedding, optimization, and DL techniques. DL-based frameworks begin with word embedding. Low-dimensional vectors represent words in word embedding by understanding syntax and semantics. Words with similar meanings usually appear in similar contexts. Several recent pre-trained word embedding are Word2Vec, Global vectors (GloVe), FastText, Embeddings from Language Models (ELMo), and Bidirectional Encoder Representations from Transformers (BERT) (Albalawi et al., 2021). Feng et al. (2018) developed a multistage framework that uses the Word2Vec model for identifying spam messages and malicious URLs in Sina-Weibo OSN. Madisetty & Desarkar (2018) have built a CNN-based ensemble with two-word embeddings (Word2Vec and GloVe) to construct a feature-based model on balanced HSpam14 (Sedhai & Sun, 2015) and imbalanced 1KS10KN Tweets datasets. Jain et al. (2018) have developed a Semantic Convolutional Neural Network (SCNN) approach that leverage features from Word2Vec, wordNet, and conceptNet word embeddings. Their SCNN model used the CNN model with a semantic layer on Twitter and *SMS spam* datasets. Later the same authors, Jain et al. (2019), proposed CNN and LSTM-based sequential stacked (SSCL) models using the same word embeddings and datasets. Ban et al. (2019) have developed a Bi-LSTM approach that leverages DL models, Word2Vec, and statistical features from the Twitter text. Barushka & Hajek (2020) implemented a framework that used TF-IDF for feature extraction and a multi-objective evolutionary feature selection (MOEFS) with a regularised deep neural network (RDNN), to minimize the classifier miss-classification rate on two datasets Hyves and Twitter dataset. Lai et al. (2022) have built a neural network-based system using CNN and LSTM techniques along with TF-IDF and Word2Vec to detect fake news. ML models require a lot of data to build good NLP models. Extracting relevant features from large data for certain domains or languages becomes tedious and costly. Transfer learning methods are possible in such situations. Transfer learning models leverage pre-trained models such as ResNet50, GloVe, and DenseNet to rebuild and reuse DL models (Dargan et al., 2020; Rao et al., 2021). The summary of the existing techniques/approaches used for text-based spam detection in OSNs is given in Table 1.

Some of the challenges identified after an in-depth review of related articles are as follows:

- **Dearth of large datasets**: Often, ML/DL-based spam detection models were trained on small datasets.
- **Class imbalance problem**: ML algorithms work best for the majority class while neglecting the minority class. Hence, it is essential to balance the imbalanced dataset using under-sampling and over-sampling strategies to avoid biased ML model training and testing.
- **Spam drift**: Spammers often change tactics to evade spam detectors. This degrades the performance of ML/DL-based spam detection systems.

Table 1
Summary of existing techniques/approaches used for text-based spam detection in OSNs.

References	Datasets used	Feature extraction/ word embedding	ML/DL approach used	Algorithms used	Analysis
(Gupta et al., 2018)	*Hspam14 dataset*	BoW	SL, EL	SVM, NB, RF, and GB	Authors have used basic-level word representation to generate feature vectors.
(Mehmood et al., 2018)	*YouTube Spam collection dataset*	TF-IDF	SL, EL	RF, GBT, and DT	Authors have tested their framework on single and small datasets only with the TF-IDF technique.
(Ahmed et al., 2018)	*Dataset 1* (Ott et al., 2011), *dataset 2* from reuters.com and kaggle.com	*n*-gram	SL, MOA	SGD, SVM, LSVM, KNN, LR, and DT.	Their framework lacks semantics, statistics, and writer's style features
(Jain et al., 2018)	*Twitter Dataset, SMS spam dataset*	Word2Vec, WordNet and ConceptNet	SL, DL, EL	SCNN Framework, SVM, NB, ANN, RF, KNN	Authors have used semantic layers for feature learning and tested their approach on small and imbalanced datasets.
(Feng et al., 2018)	*Sina-Weibo dataset*	Word2Vec	SL, DL, EL	Multistage framework; LR, RF, GBT, and NB; DL	Authors have used small datasets
(Saumya & Singh, 2018)	Reviews from amazon.com	Cosine similarity	SL and oversampling techniques	RF, GB, SVM; Smote, ADASYN	The authors have used data-balancing techniques.
(Madisetty & Desarkar, 2018)	*HSpam14 dataset; 1KS10KN*	Word2Vec, GloVe	DL	CNN	Authors have used word embeddings (Word2Vec and GloVe) with the CNN model on two datasets. However, their system lacks user profile features.
(Ban et al., 2019)	*Twitter dataset*	Word2Vec	SL, DL, EL	Bi-LSTM technique; C4.5, J48, RF, KNN, NB, SVM, and NN	The proposed framework can be extended with advanced word embedding techniques.
(Jain et al., 2019)	*SMS spam; Twitter dataset;*	WordNet, ConceptNet	DL	CNN, LSTM, SSCL	Tested semantic vector representations with DL models.
(Barushka & Hajek, 2020)	*Hyves dataset; Twitter dataset*	TF-IDF	DL	MOEFS + RDNN	Miss classification of the classifier was reduced using MOEFS with cost-sensitive RDNN.
(Kardaş et al., 2021)	*Twitter dataset*	CV, TF-IDF	ML	MNB, LR, SVM	Authors have tested ML models with several pre-processing methods
(Lai et al., 2022)	*Fake news from Kaggle.com*	TF-IDF and Word2Vec	ML and DL	LR,NB,SVM, RF; CNN, LSTM	The proposed framework can be extended with advanced DL models.

- **Performance**: Building efficient and robust spam detector systems requires better data pre-processing, NLP-based pre-trained word embeddings, ensemble approaches, and advanced DL models.
- **User privacy and security**: It is necessary to build a safe and spam-proof system that examines text entering OSN users' profiles and stops it if it is spam to improve privacy and security.

Thus, efficient spam detection systems must be developed to address the above challenges. Our proposed approach aims to solve some of the challenges mentioned above in the following ways:

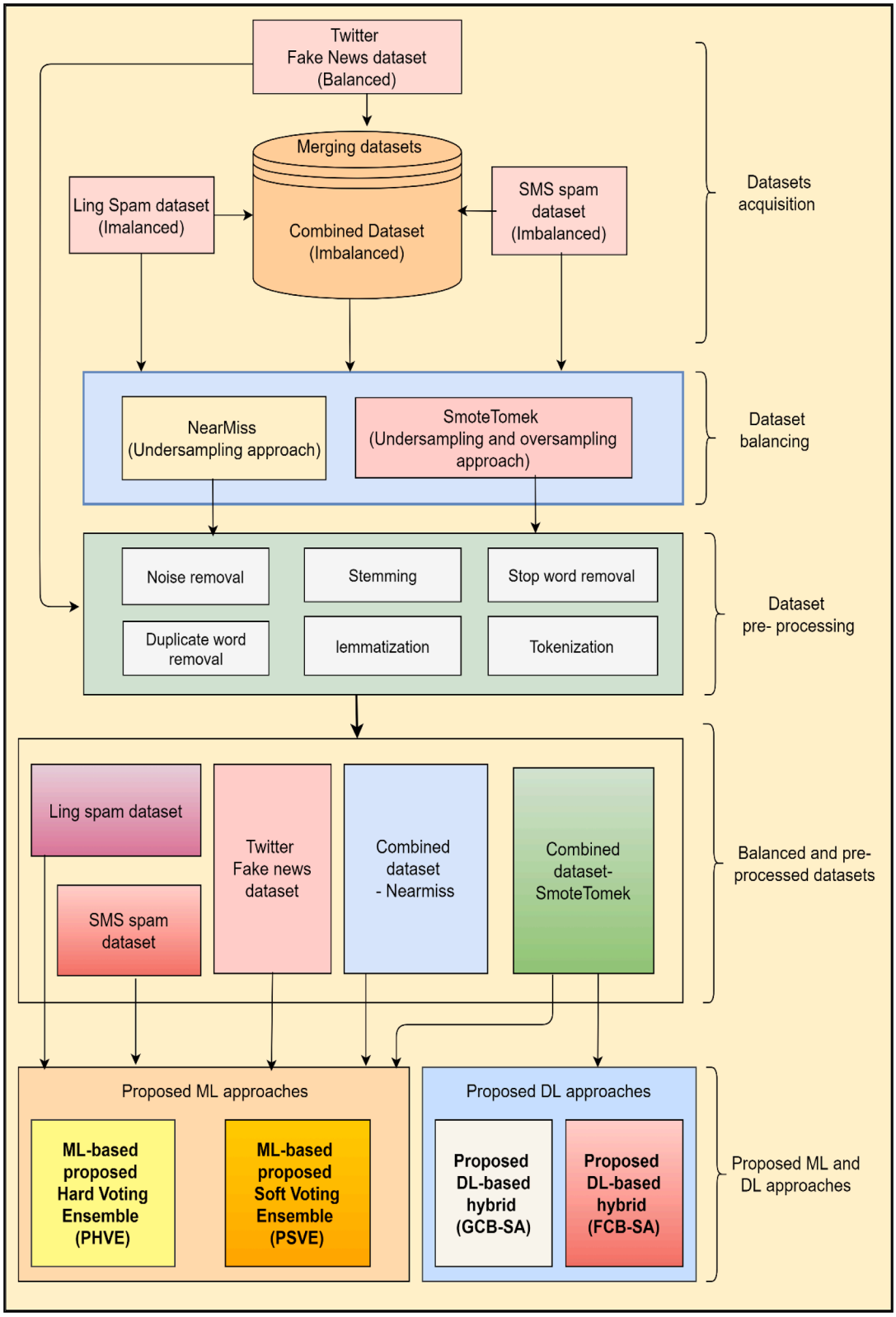


Fig. 1. Overall proposed framework with ML and DL approaches for social spam detection.

- Our proposed framework utilizes a larger corpus created by merging three benchmark datasets: *SMS spam, Fake news dataset (FND),* and *Ling spam*.
- Dataset balancing techniques such as NearMiss and SmoteTomek are employed to balance the imbalanced dataset.
- Several phases of pragmatic text pre-processing techniques are applied before feeding into ML/DL model training.
- Various feature representation and pre-trained word embedding techniques are employed and tested.
- For getting better performance, two ML-based voting ensemble models are proposed that leverage the predicted results of individual ML models.
- Two DL-based hybrid approaches are proposed for better contextual output and efficient results on the balanced *combined dataset*. In the first approach, GloVe pre-trained word embeddings are passed (Conv1D + Bi-GRU + Bi-LSTM) sequential model with the self-attention mechanism, whereas in the second FastText embeddings are given to (Conv1D + Bi-GRU + Bi-LSTM) sequential model with the self-attention mechanism.
- The proposed approaches are examined by comparing the results with existing approaches/frameworks using various performance measures.

3. Methodology

The proposed framework to categorize the text message as *Spam* and *Ham* is described in this section. Our social spam detection framework consists of the following main approaches:

a) Proposed hard voting ensemble (PHVE) and proposed soft voting ensemble model (PSVE).
b) Proposed (GloVe + Conv1D + Bi-GRU + Bi-LSTM) with Self-attention mechanism (GCB-SA) and (FastText + Conv1D + Bi-GRU + Bi-LSTM) with Self-attention mechanism (FCB-SA).

The methodology of the proposed framework consists of seven parts: (i) Dataset acquisition, (ii) Data balancing, (iii) Data pre-processing, (iv) Exploratory data analysis, (v) Word representation techniques, (vi) ML-based approaches, and (vii) DL-based approaches. The pictorial representation of the overall proposed framework with ML and DL-based approaches for social spam detection is shown in Fig. 1. The detailed schematic for proposed ML-based ensemble models (PHVE and PSVE) is presented in Fig. 5, and proposed DL-based hybrid approaches (GCB-SA and FCB-SA) are shown in Fig. 9. The following subsections describe all the parts of the proposed framework with ML and DL-based approaches:

3.1. Dataset acquisition

Due to the lack of a bigger dataset, we have built a larger text corpus of *spam* and *ham* text messages by merging the existing datasets from three different channels, i.e., mobile SMS, linguistics messages from email, and COVID-19 spam tweets from Twitter. Moreover, this will improve the training of DL-based models with a larger and more diverse dataset containing various forms of text spam on social networks. The three datasets merged are the *SMS spam dataset, Ling spam,* and *Fake news dataset (FND)* to form a larger dataset named the *combined dataset*. The *combined dataset* comprises various spam content, i.e., text messages spam, research spam, job posting spam, healthcare-related spam, lottery spam, malicious links, free software download spam links, etc.

The merging phase involves challenges such as data heterogeneity, data deduplication, eliminating irrelevant columns, and standardizing class labels. All of the mentioned challenges are handled by thorough data profiling, data cleaning, standardization, transformation, and deleting extra/irrelevant/empty columns such as the 'ID' column from the *Fake news dataset,* 'Source', and 'Subject' from *Ling spam dataset*. Later, the content aggregation is done by appending data from three datasets under appropriate columns. Standardizing class labels is another important challenge where the class labels are given in different forms in the datasets. For example, in the *SMS spam dataset*, class labels are shown as *'ham'/'spam'*; in the *Ling spam dataset*, the class labels are given as *'0'/'1',* and in the *Fake news dataset*, class labels are mentioned as *'real'/'fake'*. All the class labels are converted to the same standardized form of *'ham'/'spam'*. Later, the standardized labels are converted into *(0/1)* numeric values. Further, the dataset balancing for imbalanced datasets is done in the subsequent sub-section 3.2. The sample of the *combined dataset* is given in Table 2.

3.2. Dataset balancing

Machine learning is challenged by imbalanced dataset classification. Traditional supervised learning techniques bias toward the majority class in imbalanced datasets, leading to poor classification results on the minority class (Ghatasheh et al., 2020; Tao et al., 2022; Tolba et al., 2021). Thus, solving the class imbalance problem is essential before applying any ML model. To know the class imbalance for binary-class datasets, the class imbalance ratio (CIR) for datasets is computed as *(Size of minority class)/(Size of majority class)*. The summary of the datasets used, along with their CIR, is given in Table 3.

Table 3 shows an imbalanced distribution of *spam* and *ham* messages in the *SMS spam, Ling spam, and combined dataset* that must be balanced. The dataset balancing is essential to avoid biased model training, i.e., training on the majority class.

The data resampling techniques results in a significant performance improvement for some ML models by altering the composition of the training dataset on imbalanced data. The dataset can be balanced using under-sampling, over-sampling, and hybrid approaches that use both under-sampling and over-sampling approaches (Tolba et al., 2021). The concept behind the under-sampling technique is to lessen the number of samples of class containing more data equivalent to the class containing the minimum samples, whereas the over-sampling technique is just the inverse of the under-sampling technique that adds more data to the least sample containing class (Sağlam & Cengiz, 2022). The following data resampling techniques are used for balancing datasets:

Table 2
Sample of *combined dataset* formed from *SMS spam, Ling spam,* and *FND dataset*.

Text	Label	Label_text	Source
Free entry in 2 a weekly comp for a chance to win an ipod. Txt POD to 80182 to get entry (std txt rate) T&C's apply 08452810073 for details 18+	1	*spam*	*SMS spam dataset*
He telling not to tell any one. If so treat for me hi hi hi	0	*ham*	*SMS spam dataset*
hey there interested in some free xxx site 's?	1	*spam*	*Ling spam dataset*
i do n't know about eskimo words for snow, but scottish gaelic has a special word " turadh " for when it stops raining! compare: tha an *t*-uisge ann. is the water in it " it is raining " tha an turadh ann. is the dry-spell in it " it has stopped raining. " also: rinn e turadh san fheasgar. made it a dry-spell in the afternoon " it stopped raining (for a while) in the afternoon. kevin donnelly	0	*ham*	*Ling spam dataset*
Current understanding is #COVID19 spreads mostly from person to person through respiratory droplets produced when a person coughs or sneezes similar to how flu spreads. Learn more at https://t.co/VvIzx7O3mM https://t.co/MiHHHyCfTa	0	*ham*	*Twitter FND dataset*
Birx says COVID-19 outbreak not under control because â€people are on the moveâ€™ https://t.co/YFqi6f7Rvj	1	*spam*	*Twitter FND dataset*

Table 3
Summary of individual datasets and *combined dataset* used for the experimentation along with CIR.

Datasets	Reference	Year	Total instances	*#Ham(0)*	*Ham* (%)	*#Spam*(1)	*Spam* (%)	CIR
D1 *(SMS spam)*	(Almeida & Hidalgo, 2016)	2016	5574	4827	86.59	747	13.40	0.15
D2 *(Ling spam)*	(Gu, 2019)	2019	2892	2411	83.36	481	16.32	0.19
D3 *(Fake news)*	(Patwa et al., 2021)	2021	8560	4480	52.34	4080	47.66	0.91
Combined dataset (D1 + D2 + D3)			**17024**	**11716**	**68.82**	**5308**	**31.17**	**0.45**

- **NearMiss (NM):** We have used the NearMiss under-sampling technique from the *'Imblearn'* package for balancing using under-sampling.
- **SmoteTomek (ST):** The SmoteTomek technique performs both under-sampling and over-sampling. The ST technique from '*Imblearn. combine'* package combines two techniques. The SMOTE (Synthetic Minority Over-sampling Technique), built on the KNN algorithm, is used for upsampling, creating synthetic minority class data points to balance the dataset. In contrast, Tomek is a heuristic approach that performs down-sampling. Tomek links classify all the pairs of data points nearest to each other but belonging to different classes (Wang et al., 2019).

However, in addition to the benefits of under-sampling and over-sampling techniques, they also have limitations. Under-sampling approaches entail the loss of majority class data points to balance the dataset. In contrast, over-sampling techniques elude the limitation of under-sampling but produce additional samples inside the minority class, resulting in model overfitting. Table 4 shows the original dataset shape and the dataset shape after balancing using NM and ST techniques.

The resampled dataset shape of *SMS spam* and *Ling spam* retrieved using NM consists of small instances. Hence are ignored for ML and DL-based experimentation. However, the balanced *combined dataset* has comparatively large instances from NM under-sampling technique. Thus, the balanced *SMS spam dataset, Ling spam dataset,* and *combined dataset* using the ST technique, along with the balanced *combined dataset* retrieved using the NM technique, are considered to train and test several ML models, as shown in Fig. 5. However, it's observed that some ML models outperformed on balanced data using the ST technique. So, the balanced *combined dataset* retrieved using the ST technique is only used for the proposed hybrid DL-based experimentation, as shown in Fig. 9. However, the performance comparison of various ML models on imbalanced and balanced datasets is summarized in Section 4 to comprehend how dataset balancing affects ML models performance.

3.3. Dataset pre-processing

Data cleaning and data pre-processing are the pragmatic steps required for any NLP-based text classification. In our experimentation, the *Fake news dataset* is directly passed to the dataset pre-processing phase as it is already balanced. However, the imbalanced datasets (*SMS spam dataset, Ling spam dataset,* and *combined dataset*) are balanced using dataset balancing techniques and then sent to data pre-processing before feeding into any ML/DL model. The Dataset pre-processing involves the following steps:

- **Dataset cleaning:** The datasets were cleaned for any noise (white spaces, newlines, missing values), punctuation marks, special characters, and HTML tags, etc. using the *'re'* python library. The shortened words with apostrophes were commonly used in datasets and were corrected using the contractions python module.
- **Dataset normalization and standardization:** The cleaned data are converted into lowercase tokens, as some NLP libraries are case-sensitive. The irrelevant data, such as duplicate words and stop words, were removed, that will reduce the actual size of the data. Input features are converted into numerical tokens using the Keras tokenizer, followed by padding to create a dictionary of unique words. Then, tokens are converted into standard form by applying stemming and lemmatization. Lemmatization eliminates morphological information from words, whereas stemming removes prefixes and suffixes. We have used snowball stemmer for stemming and WordNetLemmatizer for lemmatization. The lower the noise, the higher the prediction performance of the models. The categorical output values ('*spam*'/'*fake*' and '*ham*'/'*real*') are converted into numerical form using the label encoder module of the *Scikit-learn* library.
- **Data split:** Further, the processed data is distributed into two sets, i. e., training data 80 % and testing data 20 %.

3.4. Exploratory data analysis

EDA is a process of analyzing data to find patterns, statistics, and graphical representations. The following EDA has been applied to the *combined dataset* to draw some relevant insights:

- **Word Clouds:** The word clouds formed from the *ham* and *spam* words is visualized using *wordcloud()*. The most common stop words, such as "of", "the", "a", "is" etc. are removed before plotting word clouds. The *ham* message word cloud shows that "language", "system", "english", "linguistic" and "people" are the most frequent words in *ham* messages, as shown in Fig. 2(a). The spam message Word cloud shows that "free", "https", "CO", "claim" and "bulk email" are the most commonly used words in spam messages shown in Fig. 2(b).

Table 4
The shape of imbalanced datasets before and after using data balancing techniques for *SMS spam, Ling spam,* and *combined dataset.*

Dataset	Class	Original dataset shape	Resampled shape retrieved using NM technique	Resampled shape retrieved using ST technique
SMS Spam dataset	*Ham (0)*	4827	747	4825
	Spam(1)	747	747	4825
Ling Spam dataset	*Ham (0)*	2411	481	2410
	Spam(1)	481	481	2410
Combined	*Ham (0)*	11716	5308	11714
Dataset	*Spam(1)*	5308	5308	11714

2(a). Word Cloud for *ham* words

2(b). Word Cloud for *spam* words

Fig. 2. Word Cloud for *ham* words and spam words.

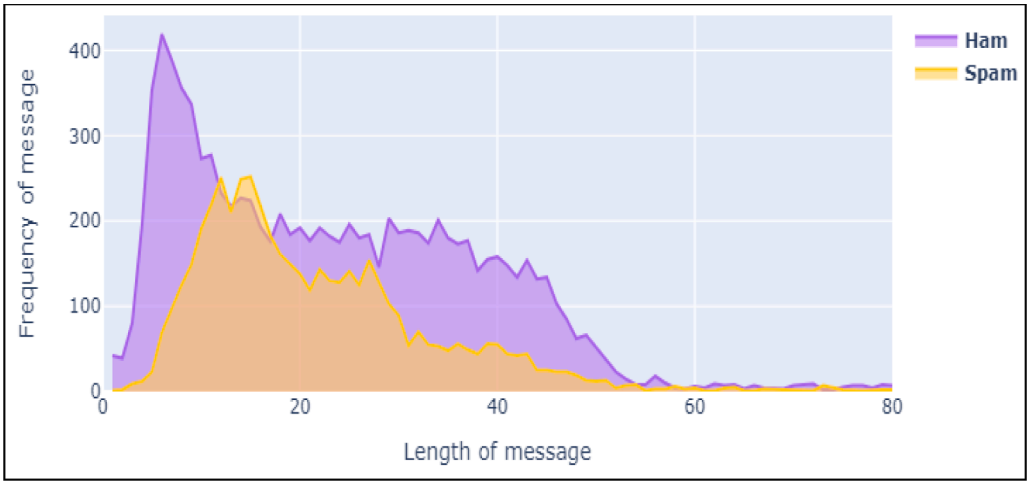


Fig. 3. Frequency of *ham* and *spam* messages over the length of message in the *combined dataset.*

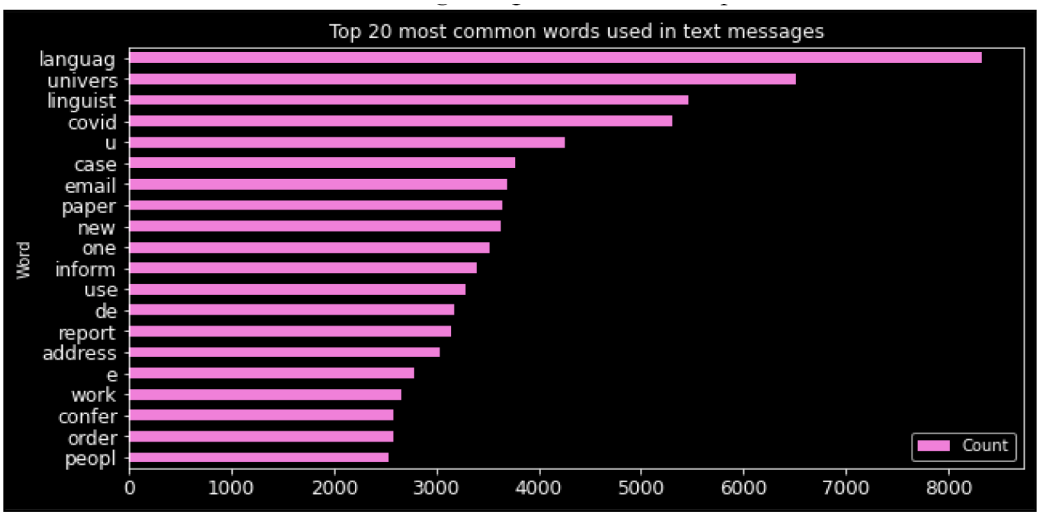


Fig. 4. Top 20 most common words used in text messages.

- **Frequency of *Ham* and *Spam* messages over the message length**: From Fig. 3, it is clear that most of the *ham* messages are of shorter length than spam messages length.
- **Top 20 most common words in text messages:** Fig. 4 shows the top 20 most common words used.
- **Dataset statistics:** The *describe()* method from the pandas library offers more statistics about the *combined dataset*, such as the count, mean, standard deviation (std), min, and max length of *spam* and *ham* messages. The dataset contains 17,024 messages with 16,584 unique messages and 440 duplicate messages. The most frequent *ham* message is 'Sorry, I'll call later' with a count of 30, whereas the most frequent *spam* message is 'this is not spam; you are receiving this mess…'. Table 5 describes the statistics for the *combined dataset.*

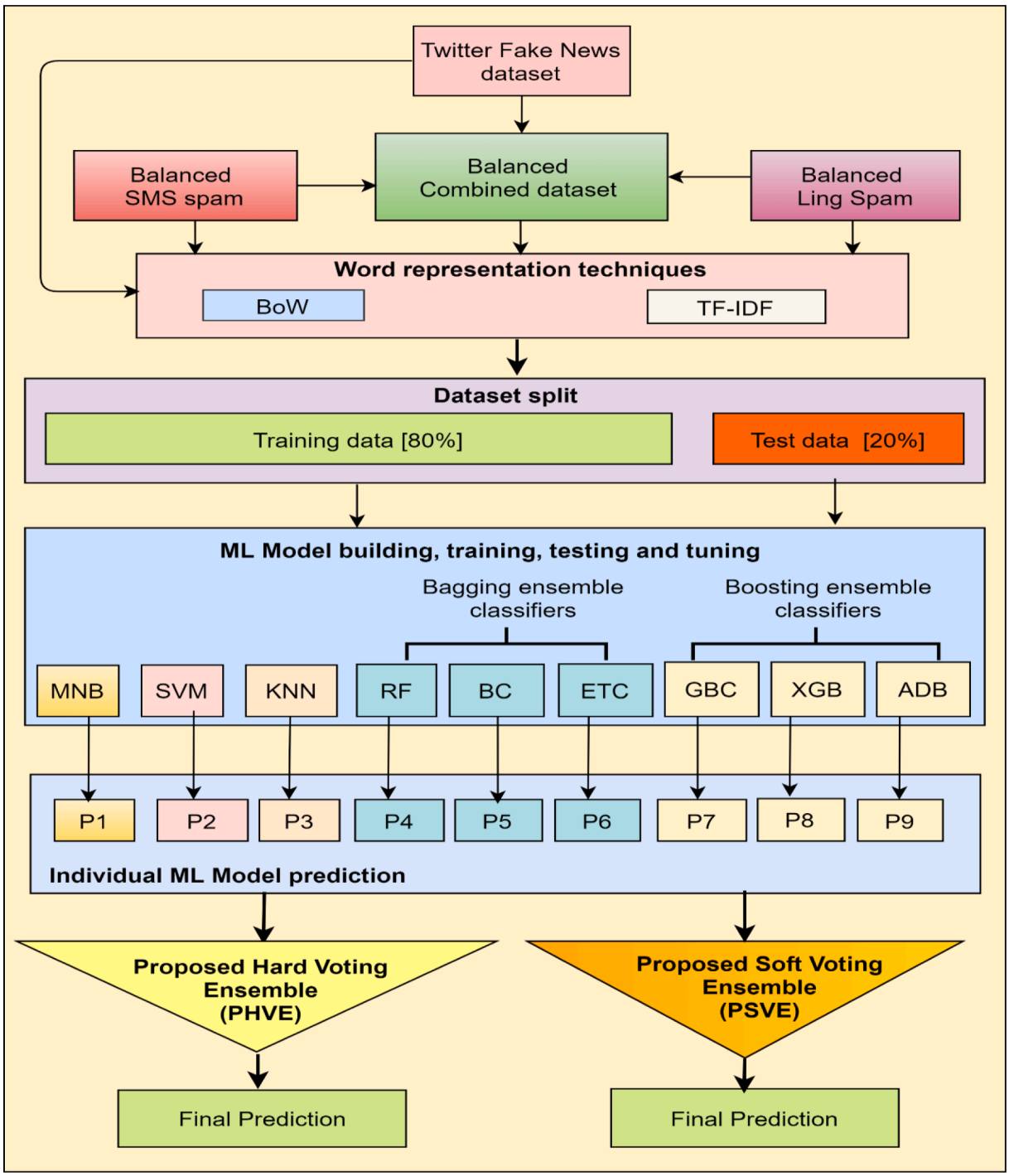


Fig. 5. Schematic for ML-based proposed hard voting ensemble (PHVE) and soft voting ensemble (PSVE) models.

Table 5
Statistics for the *combined dataset*.

Label	Count	Unique	Top frequent message	Frequency	Mean	Std	Min	25 %	50 %	75 %	Max
Ham (0)	11716	11395	Sorry, I'll call later	30	736.27	1889.71	2.0	65.75	162.0	291.0	28649
Spam (1)	5308	5189	this is not spam; you are receiving this mess…	5	460.64	1855.77	6.0	91.00	135.0	195.0	28571

3.5. Word representation techniques

As classification algorithms cannot work directly with textual data, the next step is to extract key features from textual data by converting the data into numeric representations. The most common word representation techniques are one-hot encoding, Count vectorizer (CV), and Hash vectorizer (HV). The following describes the frequency-based vectorization techniques and pre-trained word embeddings used for the experimentation:

a) Frequency-based vectorization techniques: BoW,

CountVectorizer (CV), and TF-IDF using TfidfVectorizer of *sklearn* library for frequency-based vectorization is used. In BoW, a sentence/document is characterized as a multiset of words with the frequency of each word that appeared in the document or sentence. The BoW excludes information such as phrases, syntax, and word order (Barushka & Hajek, 2018; Gupta et al., 2018). TF-IDF is an improvement over BoW as it multiplies the frequency of words within a document by the rarity with which they appear in the entire corpus. The TF denotes the frequency with which a word appears in a sentence concerning all other words, and the IDF denotes the downscaling of words often appearing in other comments/documents. Words with a higher frequency will have a higher weight, whereas words with a low frequency will have a lower weight (Lai et al., 2022; Mehmood et al., 2018). The TF-IDF weight of a word $w_{t,d}$ in a document/corpus can be calculated using Eq. (1):

$$w_{t,d} = tf_{t,d} \times log\left(\frac{N}{df_t}\right) \tag{1}$$

where t *is term/word*, d *is document*, $tf_{t,d}$ *is* number of occurrences of t in d, df_t *is number of documents containing* t, N *provides* total number of documents in the input corpus

The word representation techniques such as BoW and TF-IDF create sparse and high-dimensional feature vector space that makes training ML/DL models hard. Also, BoW and TF-IDF can't hold the word's semantic meaning. This problem can be solved using dense and low-dimensional feature representations using neural network-based pre-trained word embeddings. Besides, pre-trained word embeddings effectively identify a language's linguistic (syntactic and semantic) regularities (Verma et al., 2021).

b) Word embeddings: The feature extraction techniques retrieve important features from the textual data. In comparison, traditional word representation is helpful for long text messages and documents and is not useful for short text messages such as SMS, tweets, comments, news headlines, etc. Moreover, traditional techniques wouldn't improve the classification results and failed to capture the contextual relationship between the words. These issues are solved by word embedding techniques, which describe words using dense vector representations where similar words have similar vectors. It helps in catching semantic similarities. Word embedding maps each word or phrase from the vocabulary to an N-dimensional vector of real numbers. Its goal is to collect semantics and syntax on low-dimensional vectors. Several word embedding approaches have been employed to convert textual words into integers, known as word vectors, as an input to ML/DL models. Word embedding is trained from a large corpus. Generally, language or domain-specific word embeddings capture the statistical relations of all words in the corpus (Naseem et al., 2021). Employing existing pre-trained word embeddings is more viable than training them. In this paper, two-word embedding techniques, namely GloVe and FastText, were explored and applied as follows:

- **GloVe:** Global vectors were created at Stanford University. It's a count-based unsupervised learning approach for extracting word representations from a corpus by aggregating global word co-occurrence matrices (Pennington et al., 2014). The GloVe model's central idea is that word-to-word co-occurrence probability ratios can encode meaning. For this, consider a function F that models the ratio of probabilities relationship as mentioned in Eq. (2):

$$F\left(w_x, w_y, \widetilde{w}_z\right) = \frac{P_{xk}}{P_{yk}} \tag{2}$$

where w_x, w_y *are the words in context*, $\widetilde{w}_k$ *are the words out of context*, P_{xk}/P_{yk} depends on three words x, y, and z

The GloVe is better than Word2vec because the Word2Vec model is prediction based and captures the relations between words in a window. For the DL-based experimentation, the GloVe embedding '*glove.6B.300d.txt*' with 1.04 GB in size is used. The GloVe embedding has 300 dimension word vectors trained on 6 billion words with 400 k dictionary words (Pennington et al., 2014).

- **FastText:** FastText is a library developed by Facebook in 2016 as an extension of the Word2Vec approach for efficient learning of word representation and sentence classification (Bojanowski et al., 2016). FastText divides words into n-grams rather than providing individual words to the Neural Network (sub-words). FastText model training takes longer than other word embeddings since n-grams exceed the number of words. It outperforms Word2Vec and correctly represents rare words. Therefore, if a word was not seen during training, its embeddings can be determined by splitting the word into n-grams (Naseem et al., 2021). Word2Vec and GloVe are incapable of generating vector representations for words not included in the model dictionary, but this is a significant advantage of FastText. Moreover, FastText is fast, but the limitation is its huge size.

For DL-based experimentation, the FastText embedding (*wiki-news-*

Table 6
Comparison of various word representation and word embedding techniques.

Word representation and word embedding techniques	Type	Merits	Demerits
BoW (Traditional word representation)	Frequency-based	a) Computation is easy. b) Essential metric to extract terms. c) Compatible with the unknown word.	a) It can't capture the word's semantic, syntactic, and sentiment information.
TF-IDF (Traditional word representation)	Frequency-based	a) Computation is easy. b) Less memory. c) Ideal for text problems with larger words and document files. d) Basic technique to mine the descriptive terms. e) Similar terms do not influence results due to IDF.	a) It can't capture the word's semantic, syntactic, and sentiment information.
Word2Vec (Non-contextual word representation)(Mikolov et al., 2013)	Prediction based	a) Captures the semantics & syntactic information of words. b) Pre-trained Word2Vec trained on a large corpus.	a) It can't capture contextual information. b) It can't capture out-of-vocabulary (OOV) words. c) Requires a larger corpus to learn
GloVe (Non-contextual word representation)(Pennington et al., 2014)	Frequency-based	a) Use vectors to find sub-linear relationships b) Lower weight will not impact the training of stopwords.	a) It can't capture contextual information. b) Requires more memory. c) It can't capture OOV words. d) Requires a larger corpus to learn.
FastText (Non-contextual word representation)(Bojanowski et al., 2016)	Prediction based	a) Fast. b) More accurate in finding rare words. c) Compatible with rare words. d) Can handle OOV words.	a) Huge Size b) Requires longer training time. c) It can't capture contextual information. d) Requires more memory.

300d-1 M−subword.vec) having 2.04 GB size is used. The FastText has 300 dimension word vectors trained with subwords from Wikipedia 2017, UMBC webbase, and statmt.org news dataset (Mikolov et al., 2017). The comparison of various word representation and word embedding techniques is given in Table 6.

3.6. Machine learning approaches

3.6.1. Baseline ML models

From section 2, it is observed that most of the past studies have used basic SL, USL, SSL, EL, and MoA for social spam detection. In our ML-based experimentation, we have selected the best ML models from varied approaches such as Bayesian learning, decision-tree based, bagging, and boosting from the *scikit-learn* python library. The ML models used BoW and TF-IDF feature representation techniques. The ML models used for the experimentation are as follows:

- **Multinomial Naïve Bayes (MNB):** MNB is based on the Bayesian learning approach used to categorize discrete features (Ban et al., 2019; Feng et al., 2018; Gupta et al., 2018; Jain et al., 2018; Lai et al., 2022).
- **Support Vector Machine (SVM):** The linear support vector classification (LSVC) model from SVM provides the best-fit hyperplane to categorize the input data (Ban et al., 2019; ElSayed et al., 2018; Lai et al., 2022; Saumya & Singh, 2018). Some parameter values are changed for probability estimates, as mentioned in Table 9 (refer to experimentation and results section 4.2.1).
- **K nearest neighbors (KNN):** KNN uses closeness to categorize into classes (Ahmed et al., 2018; Ban et al., 2019; Jain et al., 2018).
- **Random Forest (RF):** RF model constructs decision trees on samples and uses the majority vote to classify (Ban et al., 2019; Feng et al., 2018; Lai et al., 2022; Mehmood et al., 2018; Saumya & Singh, 2018).
- **Bagging classifier (BC):** BC fits base classifiers on random subsets of the original dataset and then combines (votes or averages) their separate predictions to make a final prediction.
- **ExtraTree classifier (ETC):** ETC is a *meta*-estimator that uses randomized/extra decision trees on subsamples of the dataset to boost accuracy and control overfitting by aggregating the prediction of the individual decision tree.
- **Gradient Boosting classifier (GBC):** GBC is based on boosting technique that iteratively learns from weak classifiers to construct a robust model by leveraging an ensemble of weak classifiers (Feng et al., 2018; Gupta et al., 2018; Mehmood et al., 2018; Saumya & Singh, 2018).
- **Extreme Gradient Boosting (XGB):** XGB is a popular boosting model based on a regularizing gradient boosting approach. It generates an ensemble model of weak prediction models, generally decision trees, for classification stage-by-stage. However, allows the optimization of an arbitrary loss function.
- **AdaBoost classifier (ADB):** Adaboost, also known as Adaptive boost, is based on an ensemble approach. The weights are reallocated to each instance, with higher weights assigned to incorrectly classified instances.

The aforementioned baseline models perform comparatively better than other ML models for detecting text spam. For training baseline ML models, the parameter tuning of ML is done to find the best predictive model. Later the models are tested on testing data (20 %) to predict the performance of each ML model. The ensemble models RF, ETC, BC, and XGB provided good results on imbalanced datasets. Yet, the results are significantly improved on balanced datasets. For enhancing the performance score further, the prediction result of each ML model is used in our proposed soft and hard voting ensemble models, i.e., PHVE and PSVE, as described in the subsequent sub-section.

3.6.2. Proposed hard and soft voting ensemble models

To get a better combined prediction from multiple classifiers, generally bagging, boosting, and stacking/voting ensembles are acknowledged by the research community. In our experimentation, we considered the voting classifier provided by the ensemble module of the *sklearn* library. The voting ensemble model combines the performances of multiple models to make predictions using a list of estimators as arguments and a voting method. After applying the ML models individually, the predictions from individual models are combined in two ensemble models, namely the proposed hard voting ensemble (PHVE) and the proposed soft voting ensemble (PSVE), to get the most accurate prediction results.

a) Proposed hard voting ensemble (PHVE): The PHVE model is based on hard voting that uses majority voting. The predicted output class has the most votes of being predicted by each classifier. Mathematically, the hard voting mechanism can be expressed as Eq. (3),

$$\widehat{y}_h = mode\{C_1(x), C_2(x), \cdots, C_m(x)\} \tag{3}$$

where C_j is classifier, $\widehat{y}_h$ predicts the final class label via majority voting of each classifier C_j.

In the PHVE model, an ensemble of nine ML models, i.e., MNB, SVM, RF, XGB, ETC, BC, KNN, ADB, and GDB, is built by passing the list of classifiers in the '*estimator*' parameter. The voting parameter is selected as '*hard*'. Each ML model is applied with uniform weight using the '*weights*' parameter. Then the final class prediction is given by the mode of each classifier's class prediction. The predicted class label for a particular sample is the class label that signifies the majority (mode) of the class labels predicted by each classifier. For example, if there are 3 classifiers- *clf1, clf2, clf3* for specific data with their predictions as (0,1,1), then for uniform weights assigned to the classifiers, the mode of the predictions is taken. Thus, the mode of (0,1,1) is 1. Hence, the predicted class to the particular data becomes *class 1*. The various parameter settings for building and training PHVE are provided in Table 9. Later, the performance of the PHVE model is validated on the testing dataset. After the model evaluation on test data, the best parameter values are extracted, and the best prediction model is finalized by adjusting the parameters. In PHVE, the final class prediction is made using the majority voting of each classifier.

b) Proposed soft voting ensemble (PSVE): In PSVE, the predicted probabilities for class labels are summed and predicted with the class label with the largest sum probability. In contrast to PHVE, PSVE returns the class label as argmax of the sum of predicted probabilities. Mathematically, the soft voting mechanism $(\widehat{y}_s)$ can be expressed as Eq. (4),

$$\widehat{y}_s = \underset{i}{argmax} \sum_{j=1}^{m} w_j p_{ij} \tag{4}$$

where w_j is the weight of jth classifier, i ∈ {0, 1} for binary classification, p_{ij} is the predicted probability of the jth classifier for class label i.

Similar to PHVE, in the PSVE model, an ensemble of nine ML models, i.e., MNB, SVM, RF, XGB, ETC, BC, KNN, ADB, and GDB, is built by passing the list of classifiers in the estimator parameter. The voting method is selected as '*soft*'. Each ML model is applied with uniform weight using the '*weights*' parameter. The predicted class probabilities for each classifier are collected, multiplied by the classifier weight, and averaged. The class label with the highest average probability is then used to determine the final class label. For example, if there are three binary classifiers- *clf1, clf2, and clf3*. For a particular data, the classifiers make the following predictions in terms of probabilities in favor of classes (*0,1*) as *clf1* (0.2, 0.8), *clf2* (0.1, 0.9), and *clf3* (0.8, 0.5). With uniform weights, the average probability of *class 0* = (0.2 + 0.1 + 0.8)/3 = 0.366 and the average probability of *class 1* = (0.8 + 0.9 + 0.5)/3 = 0.733. Then, the average probability predicted by the ensemble classifier for both classes will be [36.6 %, 73.3 %]. If the threshold is 0.5, the final predicted *class* $\widehat{y}_s = \underset{i}{argmax}$ [36.6 %, 73.3 %] = 1, i.e., most likely

be predicted with the class label with the largest sum probability. The various parameter settings for building and training PSVE are provided in Table 9. Later, the performance of the PSVE model is validated on the testing dataset. After the model evaluation on test data, the best parameter values are extracted, and the best prediction model is finalized by adjusting the parameters. The target label with the highest sum of weighted probability is chosen as the final prediction. Finally, Section 4.4 presents the performance evaluation of PHVE and PSVE models compared to other ML baseline models on various imbalanced and balanced datasets.

3.7. Deep learning approaches

ML and proposed voting models gave better results on the balanced dataset generated using the ST technique; thus, we have considered only the balanced *combined dataset* built using the ST technique for DL-based baseline architectures and proposed DL approaches. The pre-trained word embeddings used for DL-based approaches are GloVe and FastText. The Word2Vec and GloVe are the most popular, however, Facebook's FastText pre-trained word embedding is new and offers several benefits, as mentioned in Table 6. The pre-trained word embeddings are loaded to convert input data into tokens and generate word index vectors. In the next step, padding is done to generate same-length text sequences of word vectors. Later, various DL-based neural network architectures are trained using the generated embedding matrix from pre-trained word embeddings.

3.7.1. Baseline DL models

The following DL-based baseline neural networks are employed with GloVe and FastText word embeddings for the experimentation:

- **Recurrent neural network (RNN):** RNN is used for modeling sequential data. The architecture of RNN uses the current state at time step t as h_t, the previous state at time step t as h_{t-1}, output at time step t as o_t, and input at time step t as i_t. The dependency between the words in the text while making predictions but is incapable of generalizing to variable-length sequences due to short-term memory issues. For such problems, Long short-term memory (LSTM), Gated Recurrent Unit (GRU), bi-directional LSTM, and bi-directional GRU-based gating mechanisms are recommended (Alom et al., 2019; Minaee et al., 2020; Nasir et al., 2021). The simple architecture of RNN is shown in Fig. 6.
- **LSTM and Bi-LSTM:** LSTM is a unidirectional RNN that solves the vanishing gradient in backpropagation using a gating mechanism. The architecture of LSTM uses an input gate i_t, forget gate f_t, output gate o_t and cell state c_t. The historical information or output of the previous stage is denoted by h_{t-1} and current input by x_t as shown in Fig. 7.

The function of forget gate is to select which information to remove from the cell state and is mathematically described in Eq. (5):

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{5}$$

where W_f is weight matrix, b_f is the bias matrix of forget gate, and σ is

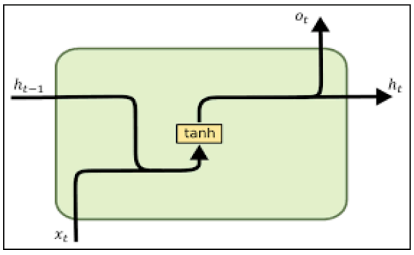


Fig. 6. Simple architecture of RNN.

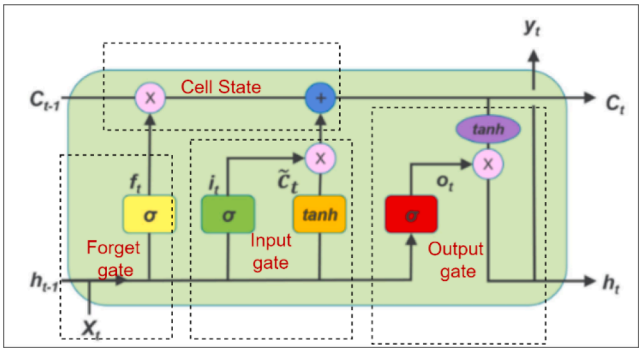


Fig. 7. Architecture of LSTM.

the activation function.

The output of f_t is 0 if old information is completely discarded. Otherwise, 1. The current cell c_t updates by forgetting some information and inserting some information from the current input x_t and previous hidden state h_{t-1}, which is decided by forget gate f_t, and input gate i_t respectively in each step. The input gate selects what information is stored in a cell using activation functions, the *sigmoid* layer, and the *tanh* layer. The sigmoid layer controls what value to update, and the *tanh* layer help in creating a new candidate value vector $\tilde{C}_t$ as mentioned in Eqs. (6), (7) and (8):

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{6}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{7}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{8}$$

Although the current hidden state h_t is determined by the present memory cell and output gate o_t. The output gate uses a sigmoid layer to decide which section of the cell state will be the output as described in Eq. (9). Eventually, the *tanh* layer with training matrix weights W_C and bias b_C provides candidate value through the input gate i_t that will be added to the new state of a memory cell C_t along with old state C_{t-1} provided by forget gate f_t. Finally, Eq. (10) represents the hidden state h_t based on the memory cell C_t as mentioned in Eq. (8).

$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o) \tag{9}$$

$$h_t = o_t * tanh(C_t) \tag{10}$$

Bi-LSTM is an extended version of LSTM that can work in both forward and backward directions to provide better performance than RNNs and LSTMs (Jain et al., 2019; Minaee et al., 2020; Tolba et al., 2021). The forward LSTM $h_{t(forward)}$ gathers the input sequences of past data information, whereas the backward LSTM $h_{t(backward)}$ receives input sequences of future data information. Later the output from both forward and backward layers are combined, as mentioned in Eq. (11).

$$h_t = h_{t(forward)} \oplus h_{t(backward)} \tag{11}$$

The LSTM and Bidirectional modules are imported from the sequential module of *Keras.models*. The hyper-parameters used for the Bi-LSTM model are mentioned in Table 10.

- **GRU and Bi-GRU:** GRU is an alternative to LSTM proposed by (Chung et al., 2014) that uses gates to control memorizing. The GRU is faster to train and less complex than LSTMs due to fewer tensor operations. GRU uses two gates: the update gate and the reset gate. It doesn't have an output gate but an update gate that combines the input and forget gate functionality, as shown in Fig. 8.

The function of the update gate z_t determines the amount of prior information that must be passed to the subsequent state. The function of

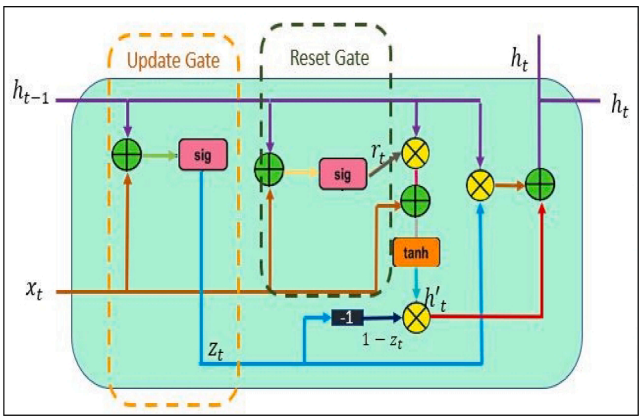


Fig. 8. Architecture of GRU.

the reset gate r_t is to determine the amount of past information required to be ignored. The update process of the hidden state h_t at time*(t)* is described in Eq. (14). Mathematically the update gate, reset gate, and hidden state functions used for computation are described in Eqs. (12), (13) and (14), respectively:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{12}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{13}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{14}$$

Similar to Bi-LSTM, Bi-GRU is an extended version of GRU that can work in both forward and backward directions to provide better performance. The GRU and Bidirectional modules are imported from the sequential module of *Keras.models*. The hyper-parameters used for the Bi-GRU model are mentioned in Table 10.

- **Convolutional Neural Network (CNN):** RNNs are trained to identify patterns sequentially, whereas CNNs learn to identify patterns across space (Lecun et al., 2015). CNN-based architectures are useful in text classification to detect patterns of a particular sentiment or topic (Minaee et al., 2020). The CNN architecture provides convolutional layers with dynamic pooling layers to generate a feature map over the sentence to explicitly capture short and long-range relations of phrases or words. In the proposed DL approaches, one-dimension convolutional neural network (Conv1D) layers are added. Initially, the embedding matrix generated from pre-trained word embeddings is passed as the sequential model's first layer. The parameters such as vocabulary length, embedding dimensions, and maximum input length of input sequences are computed in the pre-processing and tokenization phase. If the length of the text message is more than the specified maximum length then it will be truncated and if it is less than the maximum length specified then zero padding is done. Eventually, two Conv1D layers with max-pooling1D layers are added with optimized hyper-parameters as mentioned in Table 10.
- **Self-attention (SA) mechanism**: SA is used to get improved contextual output and effective results. The attention mechanism dynamically emphasizes some tokens in the input sequence by changing the token embeddings. In the SA mechanism, the multiple inputs are interacted with each other "*self*" to extract the most significant inputs and aggregate their representations based on attention weights and attention bias to form the output as a sentence vector (Niu et al., 2021; Vaswani et al., 2017). With the SA layer, the spam words such as "lottery", "promotion", "discount", "free", "offer" etc., can be given more attention. This will optimize the feature extraction process, hence increasing the overall performance of the neural network. The SA layer consists of *build()* and *call()* functions. The *build()* function is used to add attention weights and attention bias to build input shape, while the *call()* function is applied using *softmax()* across the attention scores. Later the attention scores received from *softmax()* are multiplied by the corresponding values to generate weighted values. The final output is given by summing the generated weighted values. Mathematically, the SA computation is done as mentioned in Eq. (15):

$$SA(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{15}$$

where *Q, K,* and *V* are the query vector matrix, key vector matrix, and value vector matrix, respectively, obtained from input and other parameters.

Initially, the query vector of the current input is multiplied by key vectors from other inputs. Later the score obtained is divided by $\sqrt{d_k}$. The *softmax()* function is used to normalize the results into probability distribution on all SA scores and multiplied by the value vector. Finally, we sum up the weighted value vectors to get the SA output for the word. Subsequently, the same process is applied to all input sequences (Vaswani et al., 2017).

3.7.2. Proposed deep learning approaches

For implementing proposed DL approaches, embeddings generated from two pre-trained word embeddings are used. Several ensemble neural architectures with Conv1D and Bi-RNN models with SA mechanisms are built and used for experimentation. The description of the datasets used and experimental setup with various hyper-parameters settings used are mentioned in Section 4. The layered schematic for the proposed hybrid DL approaches with the SA mechanism for efficient detection of social spam is shown in Fig. 9. The proposed hybrid DL approaches are described as follows:

a) Proposed GCB-SA: In this hybrid DL approach, GloVe embeddings, multiple conv1D layers, and bi-directional GRU/LSTM layers with the SA mechanism are used. The GloVe embedding of 300-dimension word vectors trained on 6 billion words with 400k dictionary words is used in GCB-SA to generate vector representations for words. The parameter values used for the embedding layer are given in Table 10. Initially, the pre-processed data is passed to the GloVe embeddings to generate an embedding matrix on a balanced *combined dataset*. The embeddings are then passed as an input into the first layer of the sequential model. The sequential model comprises several conv1D and max pooling layers that work together to make a feature map over the sentence. The feature map explicitly captures short-range and long-range relationships between words or phrases. The captured feature maps are passed to Bi-GRU and Bi-LSTM layers. Eventually, the SA mechanism is applied to take inputs from Bi-GRU and Bi-LSTM layers to extract the most significant inputs and aggregate their representations using attention weights and bias to form output as a sentence vector. The *softmax()* function is used to normalize the results into probability distribution on all self-attention scores and multiplied by the value vector. The weighted value vectors are then summed up to get the SA output for the word. The same process is applied to all input sequences. Finally, flatten layer, and dense layers are added with activation functions to get the final predictions. The optimized hyper-parameters and model compilation parameters used for the GCB-SA approach are given in Table 10. The summary of the proposed GCB-SA approach used in the framework for social spam detection is given in Table 7.

b) Proposed FCB-SA: In this hybrid DL approach, FastText embeddings, multiple conv1D layers, and bi-directional GRU/LSTM layers with a SA mechanism are used. The FastText embedding is huge but comparatively fast to train than other pre-trained word embeddings. FastText embedding with 300-dimensional word vectors is used to generate word vectors. Besides, FastText can generate vector representations for words not included in the model dictionary, using sub-words (*n*-grams) instead of individual words. Compared with other embedding

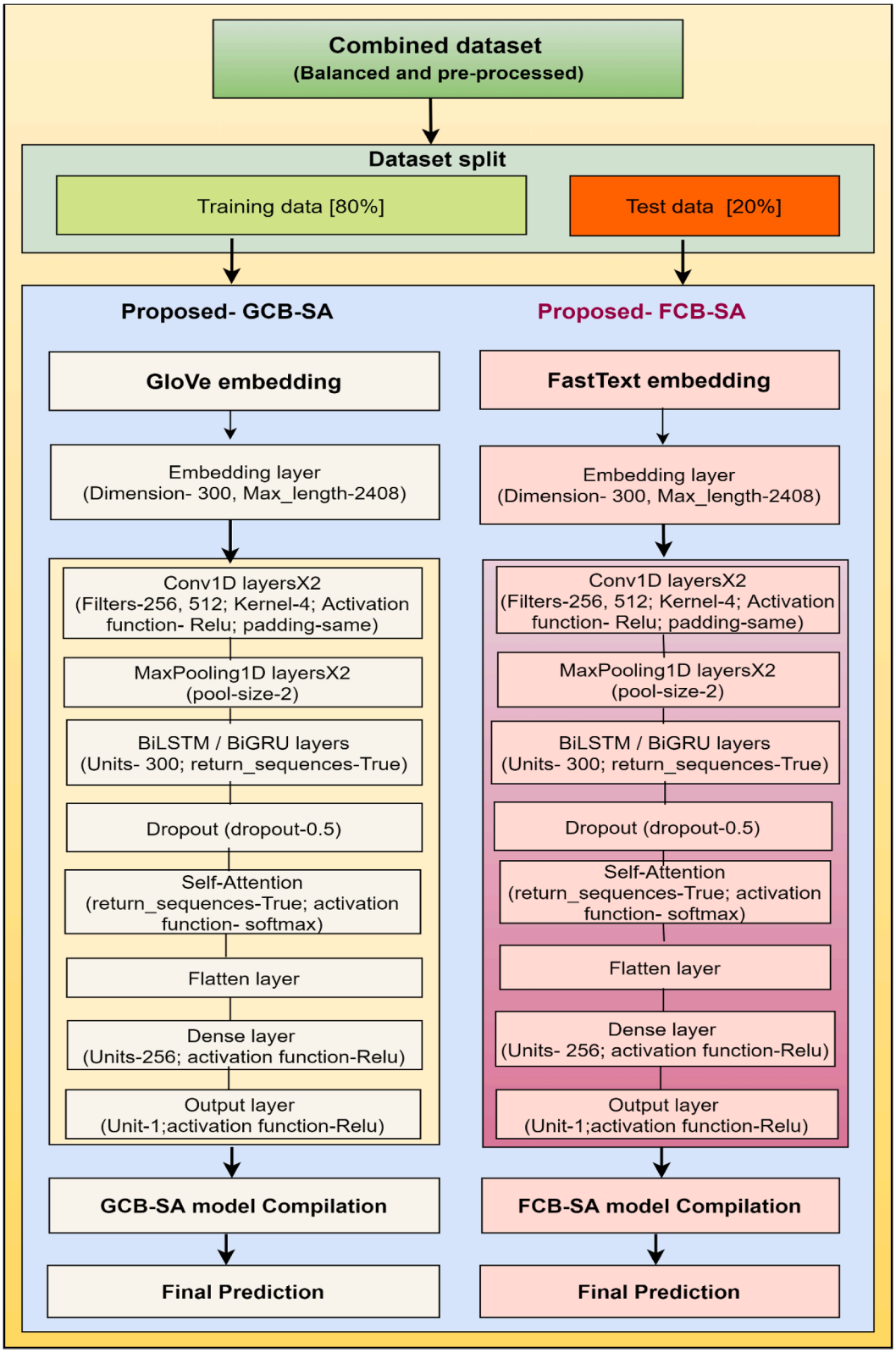


Fig. 9. Schematic for hybrid DL-based proposed GCB-SA and FCB-SA approaches.

techniques, FastText helps build vectors for unknown words since the *n*-gram exceeds the number of words. The parameter values used for the FastText embedding layer are given in Table 10. The embedding layer generated from FastText is considered as an input to the first layer of the sequential model. The layered architecture of the sequential model used in FCB-SA is same as in the GCB-SA approach. This approach also uses the SA mechanism to extract the most significant inputs. The representations based on attention weights and bias are aggregated to form an output sentence vector. The results are normalized into a probability distribution using the *softmax()* function to all the self-attention scores and then multiplying the results by the value vector. The SA output for the word is obtained by summing the weighted value vectors. All input

Table 7
Summary of the proposed GCB-SA approach used in the framework for social spam detection.

Model: GCB-SA		
Layer (type)	**Output Shape**	**Param #**
embedding_5 (Embedding)	(None, 2408, 300)	15582000
conv1d_10 (Conv1D)	(None, 2408, 512)	614912
max_pooling1d_10 (MaxPooling1D)	(None, 1204, 512)	0
conv1d_11 (Conv1D)	(None, 1204, 256)	524544
max_pooling1d_10 (MaxPooling1D)	(None, 602, 256)	0
bidirectional_10 (Bidirectional)	(None, 602, 600)	1336800
bidirectional_11 (Bidirectional)	(None, 602, 600)	1623600
dropout (Dropout)	(None, 602, 600)	0
attention_5 (Attention)	(None, 602, 600)	1202
flatten_5 (Flatten)	(None, 361200)	0
dense_10 (Dense)	(None, 256)	92467456
dense_11 (Dense)	(None, 1)	257
Total params: 112,150,771		
Trainable params: 96,568,771		
Non-trainable params: 15,582,000		

sequences undergo the same procedure. Finally, flatten layer, and dense layers are added with activation functions to get the final predictions. The optimized hyper-parameters and model compilation parameters used for the FCB-SA approach are given in Table 10. The summary of the proposed FCB-SA approach used in the framework for social spam detection is given in Table 8.

4. Experimentation and results

4.1. Dataset description

Due to a lack of a bigger social spam corpus, the *combined dataset* is merged from *SMS Spam, Ling Spam*, and *Fake news datasets*. The *combined dataset* is imbalanced. Thus, NM and ST are used to balance it as described in Sections 3.1 and 3.2, respectively. The balanced datasets for ML and DL experimentation are divided into training data (80 %) and testing data (20 %).

4.2. Experiment setting

The following sub-sections give information about experiment settings and optimized hyperparameters used for ML and DL experimentation.

4.2.1. Experiment setting for ML-based approaches

The ML experiments are performed with python 3.7. The GPU used for experimentation is Tesla T4. Parameter setting is done for individual ML models and proposed PHVE and PSVE models as described in Table 9.

Table 8
Summary of the proposed FCB-SA approach used in the framework for social spam detection.

Model: FCB-SA		
Layer (type)	**Output Shape**	**Param #**
embedding (Embedding)	(None, 2408, 300)	17931600
conv1d (Conv1D)	(None, 2408, 512)	614912
max_pooling1d (MaxPooling1D)	(None, 1204, 512)	0
conv1d (Conv1D)	(None, 1204, 256)	524544
max_pooling1d (MaxPooling1D)	(None, 602, 256)	0
bidirectional (Bidirectional)	(None, 602, 600)	1336800
bidirectional (Bidirectional)	(None, 602, 600)	1623600
dropout (Dropout)	(None, 602, 600)	0
attention (Attention)	(None, 602, 600)	1202
flatten (Flatten)	(None, 361200)	0
dense (Dense)	(None, 256)	92467456
dense (Dense)	(None, 1)	257
Total params: 114,500,371		
Trainable params: 96,568,771		
Non-trainable params: 17,931,600		

Table 9
Parameter values for ML models.

ML Classifier	Parameter value
MNB, KNN, RF, ETC, BC, GDB, ADB, XGB	Default
SVM (SVC)	probability = True, random_state = None, kernel='rbf', shrinking = True, tol = 0.001, verbose = False
PHVE (Proposed)	(estimators = [('MNB', clf1), ('SVM', clf2), ('KNN', clf3), ('RF', clf4), ('ETC', clf5), ('BC', clf6), ('GDB', clf7), ('ADB', clf8), ('XGB', clf9)], voting='hard', weights = None, n_jobs = None, flatten_transform = True, verbose = False)
PSVE (Proposed)	(estimators = [('MNB', clf1), ('SVM', clf2), ('KNN', clf3), ('RF', clf4), ('ETC', clf5), ('BC', clf6), ('GDB', clf7), ('ADB', clf8), ('XGB', clf9)], voting='soft', weights = None, n_jobs = None, flatten_transform = True, verbose = False)

Table 10
Optimised hyperparameter settings used for building and testing proposed hybrid DL approaches.

Parameter	Value	Layer
Embedding dimensions	300	GloVe and FastText embedding
vocab_length	51,940	GloVe embedding
vocab_length	59,772	FastText embedding
max_length	2408	GloVe and FastText embedding
Trainable	False	GloVe and FastText embedding
Filters	256	Conv1D
Filters	512	Conv1D
Kernel_size	4	Conv1D
Activation function	Relu	Conv1D
Padding	Same	Conv1D
MaxPooling1D (pool_size)	2	MaxPooling × 2
Bi-GRU units	300	Bi-GRU
Bi-LSTM units	300	Bi-LSTM
return_sequences	True	Bi-LSTM/Bi-GRU
Dropout	0.5	Dropout
return_sequences	True	Self-attention
Activation function	*Softmax()*	Self-attention
Dense units	256	Dense
Activation function	Relu	Dense
Dense units	1	Output layer
Output Layer activation function	Sigmoid	Output layer
Learning rate	0.001	Compile
Optimizer	Adam	Compile
Loss	Binary_cross_entropy	Compile
Batch size	128	Fit
Epochs	20	Fit
Validation split	0.1	Fit

4.2.2. Experiment setting for DL-based approaches

The individual DL models and proposed hybrid approaches are performed with python 3.7. The GPU used for experimentation is Tesla T4. Experiments are repeated with varied hyperparameter values until optimal performance is reached. The optimal hyperparameter settings for the proposed hybrid DL approaches are given in Table 10.

4.3. Performance evaluation measures

The ML and DL-based approaches are evaluated using the following performance measures (Eqs. (16), (17), (18), (19) and (20)):

$$\text{Accuracy (A)} = \frac{TP + TN}{(TP + FP + TN + FN)} \tag{16}$$

where TP is true positive, TN is true negative, FP is false positive, and FN is false negative.

$$\text{Weighted average F1 Score (F)} = \frac{2{*}Precision{*}Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \tag{17}$$

$$\text{Weighted average precision (P)} = \frac{TP}{TP + FP} \tag{18}$$

$$\text{Weighted average recall} \Big/ \text{Sensitivity} \Big/ \text{Hit Ratio (R)} = \frac{TP}{TP + FN} \tag{19}$$

Receiver Operating Characteristics (ROC): It is a 2-dimensional graph with Y-axis as the true positive rate (TPR) and the X-axis as the false positive rate (FPR).

Area under the ROC Curve (AUC): AUC under ROC gives a total performance across all possible classification thresholds.

Execution time (ET): It is computed as the sum of ML/DL model training and testing time in seconds.

Accuracy variation (AV): It is computed to examine the variation in performance of various ML/DL models on imbalanced and balanced datasets. It is the difference between ML/DL model's accuracy obtained on balanced datasets using the ST data resampling approach and the accuracy obtained on the imbalanced datasets.

$$AV = (Accuracy\ on\ the\ balanced\ dataset\ retrieved\ using\ ST) - (Accuracy\ on\ the\ imbalanced\ dataset) \tag{20}$$

4.4. Results of Machine learning approaches

The performance comparison of nine individual ML models with PHVE and PSVE is conducted on imbalanced and balanced datasets to evaluate social spam detection. The top five best performance scores from each dataset are highlighted in Table 11-14.

Table 11 provides the performance comparison of nine ML baselines with PHVE and PSVE on imbalanced *SMS spam* and balanced *SMS spam dataset* using the ST technique. The ML baseline models, PHVE and PSVE, improve on the balanced *SMS spam dataset*. Our proposed soft voting ensemble (PSVE) model achieves the highest accuracy of 98.21 % on a balanced *SMS spam dataset*. Some other ML models with good performance on balanced *SMS spam dataset* are PHVE, ETC, SVM, and XGB. The accuracy variation (AV) of the KNN model is maximum, i.e., 8.24 %, and minimum with the RF model, i.e., 1.14 %. Other models accuracy is also improved with a variation of approximately (1 ~ 4) %. The execution time (ET) of ML models is more on the balanced *SMS spam dataset* than on the imbalanced *SMS spam dataset*.

Table 12 provides the performance comparison of various ML baselines with PHVE and PSVE on imbalanced *Ling spam* and balanced *Ling spam datasets* using the ST technique. ML baseline models, PHVE and PSVE, improve on the balanced *Ling spam dataset*. The PHVE model achieves the highest accuracy of 93.09 % on the balanced *Ling spam dataset*. Some other ML models with good performance on balanced *Ling spam dataset* include PSVE, ETC, SVM, and RF. From the AV of Table 12, it is observed that the accuracy of the KNN model is decreased by 21.39 %, and accuracy for other ML models is improved within a range of (1 ~ 4)% in the balanced *Ling spam dataset.* The findings also reveal that the execution time of MNB, SVM, XGB, KNN, PHVE, and PSVE on the balanced *Ling spam dataset* is longer than on the imbalanced *Ling spam dataset*. The execution time of RF, ADB, BC, ETC, and GDB is shorter on the balanced *Ling spam dataset* than on the imbalanced *Ling spam dataset*.

The *Twitter Fake news dataset* is already balanced. Thus, a performance comparison of various ML baselines with PHVE and PSVE, only on the balanced *Fake news dataset,* is provided in Table 13. Our proposed soft voting ensemble (PSVE) model achieves the highest accuracy of 95.76 % and ROC-AUC score of 0.98 on the *Twitter Fake news dataset*. Some other ML models with good performance on the *Twitter Fake news dataset* include PHVE, ETC, SVM, and RF. The findings also reveal that the execution time of SVM, PHVE, and PSVE is longer than other ML models on the *Twitter Fake news dataset*.

As mentioned earlier, the *combined dataset* is balanced using two data resampling techniques. Thus performance comparison of various ML baselines with PHVE and PSVE is done on the imbalanced *combined dataset*, balanced *combined dataset* retrieved using NM, and balanced *combined dataset* retrieved using ST techniques is provided in Table 14. The ML baseline models, except KNN, improve on the balanced *combined dataset* retrieved using NM and on the balanced *combined dataset* retrieved using the ST technique. ML models' accuracy is improved within a range of (3 ~ 9) % on the balanced *combined dataset* retrieved using the ST technique. The accuracy of the KNN model is decreased by 14.38 % on the balanced *combined dataset* retrieved using the ST technique. The PSVE model achieves the highest accuracy of 95.88 % with a 0.98 ROC-AUC score on a balanced *combined dataset* retrieved using the ST technique. The PHVE model also performed well, with an accuracy of 94.46 % on balanced *combined dataset* retrieved using ST. Furthermore, the analysis reveals that the ETC, SVM, RF, XGB, and BC models also produced good results and contributed considerably to both PHVE and PSVE. The findings also reveal that the execution time of ADB, ETC, GDB, KNN, PHVE, and PSVE on balanced *combined dataset* retrieved using the ST technique is longer than the imbalanced *combined dataset*. MNB, SVM, RF, XGB, and BC have shorter execution times than the imbalanced *combined dataset*.

The consolidated accuracy analysis is carried out graphically by comparing PHVE and PSVE with other good-performing ML models in terms of their accuracy on different imbalanced and balanced datasets,

Table 11
Performance comparison of nine ML baselines with proposed voting ensemble models on imbalanced and balanced *SMS spam dataset* retrieved using ST.

	Imbalanced *SMS spam dataset*						Balanced *SMS spam dataset* retrieved using ST						~AV (%)
Performance parameters→ ML Models ↓	**A (%)**	**P (%)**	**R (%)**	**F (%)**	**ROC-AUC Score**	**ET (*sec*)**	**A (%)**	**P (%)**	**R (%)**	**F (%)**	**ROC-AUC Score**	**ET (*sec*)**	
MNB	89.36	89.34	89.36	89.34	0.93	0.16	93.29	93.32	93.29	93.29	0.96	0.77	3.93
SVM	92.04	92.11	92.04	92.04	0.94	357.36	**95.07**	**95.16**	**95.07**	**95.07**	**0.98**	657.43	3.03
RF	93.04	93.09	93.04	92.96	0.96	18.72	94.18	94.24	94.19	94.18	0.98	26.29	1.14
XGB	91.52	91.55	91.52	91.52	0.95	65.58	**94.70**	**94.71**	**94.70**	**94.70**	**0.98**	153.92	3.18
ADB	90.43	90.44	90.43	90.42	0.94	25.93	93.60	93.48	93.60	93.45	0.96	52.35	3.17
BC	89.64	89.65	89.64	89.54	0.93	17.01	92.23	92.17	92.23	92.18	0.95	28.78	2.59
ETC	93.69	93.74	93.71	93.68	0.96	77.24	**97.54**	**97.66**	**97.64**	**97.34**	**1.00**	77.28	3.85
GDB	89.23	89.23	89.23	89.21	0.93	120.10	91.08	91.15	91.08	91.06	0.94	184.21	1.85
KNN	66.82	66.85	66.81	66.82	0.80	4.22	75.06	75.06	75.08	75.06	0.85	7.75	8.24
PHVE	93.83	93.85	93.84	93.76	0.99	620.94	**97.87**	**97.89**	**97.87**	**97.35**	**1.00**	1189.91	4.04
PSVE	94.28	94.32	94.28	94.25	1.00	583.36	**98.21**	**98.25**	**98.22**	**98.14**	**1.00**	1197.86	3.93

Table 12
Performance comparison of nine ML baselines with proposed voting ensemble models on imbalanced and balanced *Ling spam dataset* retrieved using ST.

	Imbalanced *Ling spam dataset*						Balanced *Ling spam dataset* retrieved using ST						~AV (%)
Performance parameters→ ML Models ↓	**A (%)**	**P (%)**	**R (%)**	**F (%)**	**ROC-AUC Score**	**ET (*sec*)**	**A (%)**	**P (%)**	**R (%)**	**F (%)**	**ROC-AUC Score**	**ET (*sec*)**	
MNB	80.61	80.67	80.61	79.96	0.92	0.49	85.55	85.59	85.55	85.54	0.95	1.02	4.94
SVM	89.01	89.04	89.01	89.02	0.96	482.95	**91.68**	**91.70**	**91.68**	**91.66**	**0.97**	528.65	2.67
RF	88.58	88.65	88.58	88.50	0.96	15.82	**90.88**	**90.90**	**90.88**	**90.85**	**0.97**	8.22	2.30
XGB	87.74	87.78	87.74	87.75	0.96	100.18	89.69	89.70	89.69	89.69	0.96	216.65	2.45
ADB	81.91	81.92	81.91	81.90	0.92	105.32	83.48	83.54	83.48	83.24	0.94	77.76	1.57
BC	85.52	85.55	85.51	85.51	0.96	31.09	87.89	87.90	87.89	87.65	0.96	30.8	2.37
ETC	91.93	91.95	91.93	91.90	0.97	44.50	**95.01**	**94.98**	**95.01**	**95.01**	**0.97**	23.17	3.08
GDB	81.75	81.76	81.75	81.73	0.92	376.13	83.06	83.08	83.06	83.06	0.94	293.78	1.31
KNN	90.70	90.74	90.70	89.98	0.96	4.09	69.31	79.02	69.31	69.38	0.92	5.41	−21.39
PHVE	91.65	91.67	91.65	91.63	0.96	1016.81	**93.09**	**93.10**	**93.09**	**92.99**	**0.98**	1176.72	1.44
PSVE	88.97	88.98	88.97	88.92	0.97	1024.32	**90.05**	**90.08**	**90.05**	**90.04**	**0.97**	1223.48	1.08

Table 13
Performance comparison of nine ML baselines with proposed voting ensemble models on balanced *Fake news dataset.*

	Balanced Twitter spam dataset					
Performance parameters→ ML Models ↓	**A (%)**	**P (%)**	**R (%)**	**F (%)**	**ROC-AUC Score**	**ET (*sec*)**
MNB	90.36	90.36	90.36	90.36	0.89	0.65
SVM	**93.44**	**93.42**	**93.44**	**93.40**	**0.95**	2033.58
RF	**92.31**	**92.32**	**92.31**	**92.28**	**0.94**	61.24
XGB	89.67	89.67	89.67	89.67	0.93	228.86
ADB	86.84	86.84	86.84	86.84	0.90	169.75
BC	88.86	88.86	88.86	88.86	0.90	89.86
ETC	**95.57**	**95.57**	**95.57**	**95.57**	**0.98**	150.07
GDB	89.46	89.46	89.46	89.46	0.91	591.21
KNN	81.90	81.90	81.90	81.90	0.90	15.32
PHVE	**93.85**	**93.82**	**93.86**	**93.87**	**0.96**	1280.00
PSVE	**95.76**	**95.76**	**95.76**	**95.76**	**0.98**	971.82

as shown in Fig. 10. The ML models with 88 % or above accuracy are considered good and compared graphically with PHVE and PSVE. The performance of PHVE and PSVE models are highlighted with data labels to have a quick comparison with other ML models on imbalanced and balanced datasets. The PSVE model outperformed other models on almost all datasets except the imbalanced *Ling spam*, balanced *Ling spam*, and imbalanced *combined dataset*. The performance of the PHVE model is better on imbalanced *Ling spam* and balanced *Ling spam*. PHVE and PSVE perform better when baseline ML models improve. The major performance improvements in baseline models, PHVE and PSVE, are observed specifically on the balanced *SMS spam dataset* and balanced *combined dataset* using the ST technique than on imbalanced datasets. The KNN model's performance is relatively lower, but it produced better accuracy of 90.70 % on the imbalanced *Ling spam dataset*. The PHVE and PSVE provide considerably good results even if some baseline models are not performing well. The drawback of PHVE, PSVE, and SVM is that they took longer to train and test models than other baseline ML models.

The accuracy comparison of proposed ensemble models, i.e., PHVE and PSVE, with other existing ML-based studies/approaches is given in Table 15.

4.5. Results of deep learning approaches

DL-based experimentation is performed on a balanced *combined dataset* since ML models perform well on this dataset to improve social spam detection performance. Table 16 compares different DL baselines and proposed hybrid DL approaches (GCB-SA and FCB-SA) on the *combined dataset* with the top five best performance scores highlighted.

Table 16 shows that the performance of the proposed FCB-SA hybrid DL approach outperforms with an accuracy of 97.26 % on the balanced *combined dataset* compared to other DL-based approaches. Some other good-performing DL approaches are GCB-SA, FastText + Bi-GRU, GloVe + Bi-GRU, and FastText + Bi-LSTM. The unidirectional RNN performed better than LSTM/GRU but required a longer execution time. However, the performance gradually increased when pre-trained word embeddings from GloVe and FastText are passed in the hybrid DL approaches such as Bi-LSTM, Bi-GRU, and CNN. The Bi-GRU approach yields better accuracy on both GloVe and FastText word embeddings. Few DL models perform better with FastText than Glove word embeddings. The training and validation loss of baseline DL approaches and proposed approaches are also mentioned in Table 16. The difference between training and validation loss of DL approaches is less, which shows a good fit. The results are further improved with the self-attention mechanism as the emphasis is given to significant words or input vectors. Hence, the proposed DL approaches, i.e., GCB-SA and FCB-SA, are combined with the conv1D and attention layers. After several iterations, the best optimal hyperparameters for the proposed GCB-SA and FCB-SA approaches are gathered and presented in Table 10. The consolidated accuracy analysis for baseline DL approaches and proposed hybrid DL approaches (GCB-SA and FCB-SA) are shown in Fig. 13.

The graphical comparison between training accuracy and validation accuracy for proposed DL-based hybrid approaches, i.e., GCB-SA and FCB-SA on the balanced *combined dataset*, is presented in Fig. 11. From Fig. 11, we found that the training accuracies of both proposed approaches are nearly the same, whereas the validation accuracy of FCB-SA progressively improves with more epochs.

The training and validation loss observed on 20 epochs for the proposed DL-based hybrid approaches, i.e., GCB-SA and FCB-SA on the balanced *combined dataset* is presented in Fig. 12. From Fig. 12, we observe that training loss in both proposed approaches is almost the same, whereas validation loss is slightly less in GCB-SA than in FCB-SA with increasing epochs.

The consolidated accuracy analysis is carried out by graphically comparing GCB-SA and FCB-SA with other good-performing DL approaches on the balanced *combined dataset*, as shown in Fig. 13. The DL models with 90 % or above accuracy are considered good and compared with the proposed GCB-SA and FCB-SA. The performance of GCB-SA and FCB-SA hybrid DL approaches are highlighted with data labels to have a quick comparison with other DL approaches on the balanced *combined dataset*. Furthermore, the analysis reveals that the hybrid DL approaches such as Bi-LSTM, Bi-GRU, and CNN also produced considerably good results with pre-trained word embeddings from GloVe and FastText.

The performance comparison of proposed hybrid DL-based approaches with some existing approaches is given in Table 17.

5. Discussions and future directions

OSN spam is a critical problem that needs to be addressed. Existing

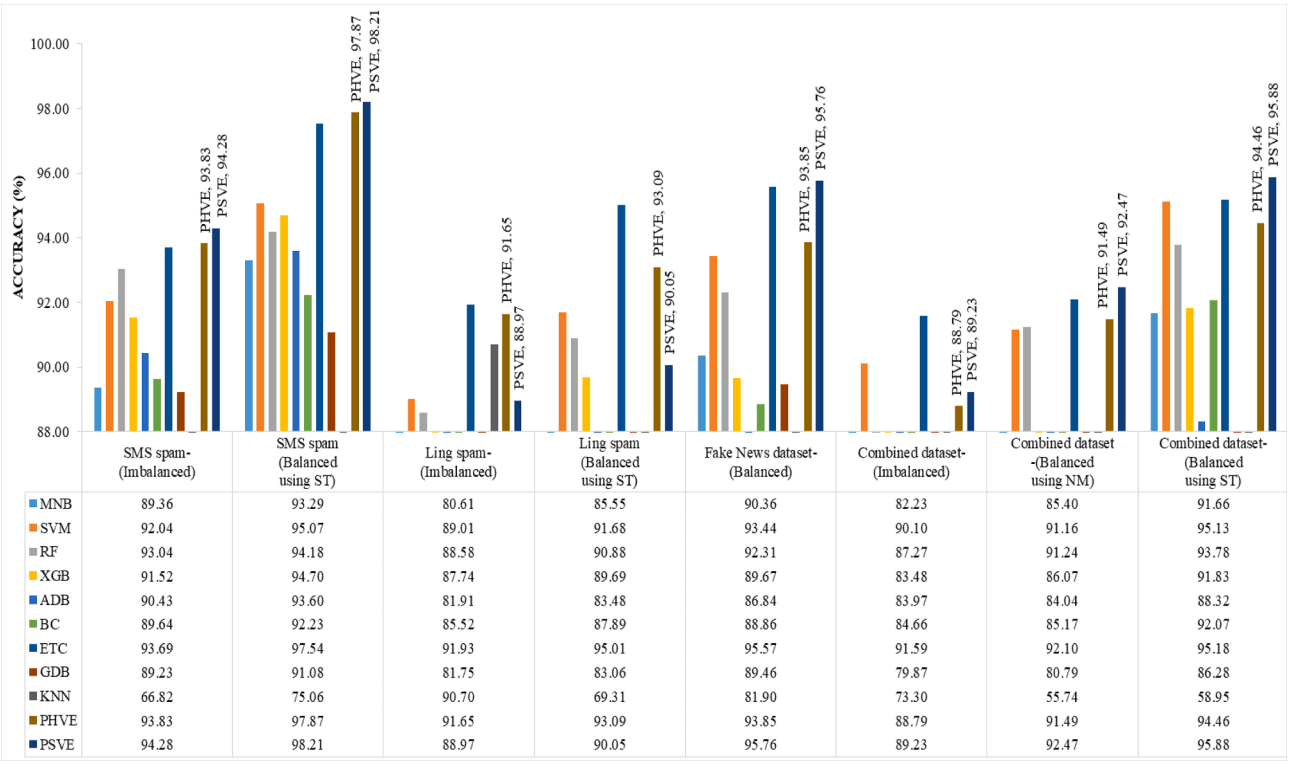
Table 14
Performance comparison of various ML baselines with proposed voting ensemble models on the imbalanced *combined dataset*, balanced *combined dataset* retrieved using NM, and balanced *combined dataset* retrieved using ST data resampling techniques.

Performance parameters→ ML Models ↓	Imbalanced Combined dataset					Balanced Combined dataset retrieved using NM					Balanced Combined dataset retrieved using ST					~AV (%)
	A (%)	P (%)	R (%)	F (%)	ROC-AUC Score	ET (*sec*)	A (%)	P (%)	R (%)	F (%)	ROC-AUC Score	ET (*sec*)	A (%)	P (%)	R (%)	F (%)	ROC- AUC Score	ET (*sec*)	
MNB	82.23	82.26	82.23	82.27	0.94	2.47	85.40	85.42	85.40	85.41	0.96	0.84	91.66	91.68	91.66	91.65	0.96	1.96	9.43
SVM	90.10	90.17	90.12	90.16	0.96	3527.09	**91.16**	**92.57**	**92.56**	**92.50**	**0.97**	1924.79	**95.13**	**95.15**	**95.13**	**95.14**	**0.98**	3349.70	5.03
RF	87.27	87.24	87.27	87.25	0.95	270.21	**91.24**	**91.25**	**91.24**	**91.23**	**0.97**	72.57	**93.78**	**93.80**	**93.78**	**93.77**	**0.99**	182.72	6.51
XGB	83.48	83.51	83.48	83.52	0.92	1264.18	86.07	86.09	86.07	86.07	0.96	468.2	91.83	91.84	91.83	91.84	0.98	1070.29	8.35
ADB	83.97	83.94	83.97	83.92	0.92	238.88	84.04	84.05	84.04	84.04	0.95	157.52	88.32	88.31	88.32	88.30	0.93	392.95	4.35
BC	84.66	84.67	84.66	84.64	0.92	645.30	85.17	85.11	85.17	85.12	0.96	209.38	92.07	92.05	92.07	92.04	0.98	490.01	7.41
ETC	91.59	91.55	91.59	91.54	0.96	365.34	**92.10**	**92.07**	**92.10**	**91.96**	**0.98**	175.62	**95.18**	**95.20**	**95.18**	**95.13**	**0.99**	540.20	3.59
GDB	79.87	79.83	79.87	79.85	0.93	947.09	80.79	80.77	80.79	80.73	0.94	553.02	86.28	86.25	86.28	86.27	0.95	1540.81	6.41
KNN	73.30	73.36	73.30	73.34	0.69	63.36	55.74	55.72	55.74	55.74	0.67	21.55	58.95	58.92	58.95	58.98	0.70	116.92	−14.38
PHVE	88.79	88.81	88.79	88.80	0.96	3626.14	**91.49**	**91.50**	**91.49**	**91.48**	**0.98**	2643.37	**94.46**	**94.42**	**94.46**	**94.48**	**0.99**	4212.88	5.67
PSVE	89.23	89.29	89.23	89.24	0.96	3663.58	**92.47**	**91.45**	**91.47**	**91.50**	**0.98**	2725.87	**95.88**	**95.70**	**95.88**	**95.84**	**0.99**	4378.57	6.65

ML/DL-based approaches are ineffective due to inadequacies such as class imbalance problems, spam drift, lack of larger labeled datasets, contextual features, performance issues, etc. Many existing studies have done their work with small and imbalanced datasets. Hence, in this study, we proposed a hybrid ensemble framework using ML and DL approaches on the larger and diverse dataset to solve a few important challenges.

Initially, the lack of a large labeled dataset is solved by merging the three benchmark datasets to build large *spam* and *ham* labeled corpus *combined dataset*. Another key issue of the class imbalance problem is solved, as unequal distribution substantially reduces the classifier's performance. The data balancing is done using NearMiss and Smote-Tomek data resampling techniques. Subsequently, several pre-processing and word representation techniques are applied to the imbalanced and balanced datasets to prepare them for ML/DL model training. Later, for the ML-based experimentation, the best models from varied approaches are selected, trained, and tested on imbalanced and balanced datasets. The comparative results of different ML models with PHVE and PSVE on various imbalanced and balanced datasets are presented in Tables 11, 12, 13, and 14, with the top five good performance scores highlighted. In addition, relevant discussions regarding the performance of ML models, accuracy variation, and execution time are also provided. Among various ML models, our proposed voting ensemble model, i.e., PSVE, outperformed with an accuracy of 95.88 % on balanced *combined dataset*. The consolidated accuracy comparison of good-performing baseline ML models with PHVE and PSVE on the imbalanced and balanced datasets is provided in Fig. 10. The findings reveal that accuracy improves on balanced datasets for almost all ML models except the KNN model. Other good-performing models are PHVE, ETC, SVM, and RF. However, the SVM, GDB, PHVE, and PSVE require more execution time. Most ML models performance increased on balanced datasets retrieved using the ST technique except KNN. The KNN model performed better on the imbalanced *Ling spam dataset* with an accuracy of 90.70 % than on the balanced *Ling spam dataset*. The performance of proposed voting ensemble models and other baselines considerably improved on balanced *combined dataset retrieved using the ST technique*. The ML models produced accurate and useful results. However, they cannot learn low-level features, which is not a problem for DL models. Therefore, DL-based experimentation is employed on the combined dataset retrieved using the ST technique to further improve social spam detection performance.

The DL model takes input data and extracts relevant features for faster learning. Therefore, GloVe and FastText pre-trained embeddings with individual DL models such as RNN, GRU, LSTM, Bi-GRU, Bi-LSTM, and CNN are experimented on the balanced *combined dataset*. Also, two hybrid DL approaches are proposed—GCB-SA and FCB-SA using GloVe and FastText word embeddings and a sequential ensemble model with the self-attention mechanism. The sequential model includes multiple layers of conv1D, Bi-GRU/Bi-LSTM model with the self-attention mechanism. The RNN provides better results than LSTM and GRU but takes a longer execution time. Uni-directional LSTM and GRU have nearly the same performance. The performance scores improved when pre-trained embeddings from GloVe and FastText are passed in Bi-GRU, Bi-LSTM, and CNN models. The conv1D layer slides different kernels across a sequence, generating 1D feature map kernel. Each kernel learned to detect a short sequential pattern. The short input sequences from Conv1D further helped Bi-GRU and Bi-LSTM layers to detect longer patterns. The Bi-GRU/LSTM have comparatively long-term memory than simple RNNs and can work in both forward and backward directions to provide better performance. One dropout layer is added to the sequential model to avoid overfitting. Bi-GRU with both GloVe and FastText yields better accuracy than other individual models. Bi-LSTM and Bi-GRU with FastText and GloVe have comparatively less training and validation loss, as shown in Table 16. Therefore, the proposed hybrid approaches are built by adding layers from CNN, Bi-GRU, and Bi-LSTM with optimized hyperparameters. DL models with FastText



	SMS spam-(Imbalanced)	SMS spam (Balanced using ST)	Ling spam-(Imbalanced)	Ling spam (Balanced using ST)	Fake News dataset-(Balanced)	Combined dataset-(Imbalanced)	Combined dataset -(Balanced using NM)	Combined dataset-(Balanced using ST)
MNB	89.36	93.29	80.61	85.55	90.36	82.23	85.40	91.66
SVM	92.04	95.07	89.01	91.68	93.44	90.10	91.16	95.13
RF	93.04	94.18	88.58	90.88	92.31	87.27	91.24	93.78
XGB	91.52	94.70	87.74	89.69	89.67	83.48	86.07	91.83
ADB	90.43	93.60	81.91	83.48	86.84	83.97	84.04	88.32
BC	89.64	92.23	85.52	87.89	88.86	84.66	85.17	92.07
ETC	93.69	97.54	91.93	95.01	95.57	91.59	92.10	95.18
GDB	89.23	91.08	81.75	83.06	89.46	79.87	80.79	86.28
KNN	66.82	75.06	90.70	69.31	81.90	73.30	55.74	58.95
PHVE	93.83	97.87	91.65	93.09	93.85	88.79	91.49	94.46
PSVE	94.28	98.21	88.97	90.05	95.76	89.23	92.47	95.88

Fig. 10. Consolidated accuracy analysis of proposed ensemble models (PHVE and PSVE) with other good-performing ML models with an accuracy of 88% or above on different imbalanced and balanced datasets.

Table 15
Performance comparison of PHVE and PSVE with existing ML-based approaches.

References	Feature extraction/ Optimization technique	ML/DL Classifier	Dataset/ Balancing technique	A (%)
(Gupta et al., 2018)	BoW	SVM, NB, RF, and GB algorithms	*Hspam14 dataset*	91.65
(Agarwal & Kumar, 2019)	PSO	NB	*Ling spam*	95.50
(Kardaş et al., 2021)	CV, TF-IDF	MNB, Neural Network, LR and SVM	*Twitter spam dataset*	93.02
PHVE (Proposed)	BoW, TF-IDF	MNB, SVM, RF, XGB, ADB, BC, ETC, GDB, KNN, PHVE	*SMS spam*	97.87
			Ling spam	93.09
			Twitter FND	93.85
			Combined dataset using NM	91.49
			Combined dataset using ST	94.46
PSVE (Proposed)	BoW, TF-IDF	MNB, SVM, RF, XGB, ADB, BC, ETC, GDB, KNN, PSVE	*SMS spam*	98.21
			Ling spam	90.05
			Twitter FND	95.76
			Combined dataset using NM	92.47
			Combined dataset using ST	95.88

embeddings perform better than DL models with GloVe, with a variation of approximately (1 ~ 2 %). The results are further improved with the SA mechanism as the emphasis is given to significant words or input vectors. The attention mechanism improved performance by dynamically emphasizing a few tokens in input sequences by transforming token embeddings (Bahdanau et al., 2014). Hence, the SA mechanism is applied in the sequential model to create an output sentence vector by extracting the most important self-inputs and aggregating their representations using attention weights and bias (Niu et al., 2021). FCB-SA outperforms other DL approaches with an accuracy of 97.26 % on balanced *combined dataset* retrieved using the ST technique. The proposed FCB-SA has a validation loss of 0.2327, which is acceptable. The FCB-SA and GCB-SA have comparably less execution time and are thus faster than the proposed ML-based PHVE and PSVE. Table 17 shows that our FCB-SA outperforms other existing frameworks/approaches.

DL-based approaches are found to be robust and effective. However, still, several improvements can be made. Acquainted with our work's limitations, we suggest several directions to advance research in social spam detection. The existing studies are hard to compare due to the lack of large-scale publically available benchmark datasets. The proposed work is evaluated on binary datasets. However, the proposed approaches can be tested on multiclass, multilingual, and multimodal datasets (Alam et al., 2021). The non-contextual word embeddings, such as GloVe and FastText, provided fairly good results. However, in the future, it can be observed how proposed approaches performed with contextual pre-trained word embeddings such as ELMO and BERT. The DL-based approaches performance can also be improved by incorporating advanced NLP-based transformer models.

6. Conclusions

This paper aims to empirically study social spam detection on

Table 16
Performance comparison of various DL baseline approaches with proposed DL models with SA mechanism on the balanced *combined dataset*.

Performance parameters→ Hybrid DL approaches ↓	A (%)	P (%)	R (%)	F (%)	ROC-AUC Score	Training Loss	Validation Loss	ET (*sec*)
GloVe + RNN	76.85	73.48	52.23	67.75	0.78	0.4123	0.4977	4360
GloVe + LSTM	69.19	67.22	68.10	55.32	0.71	0.6019	0.6176	1052
GloVe + GRU	69.73	68.64	69.43	54.98	0.72	0.6056	0.6175	816
GloVe + Bi-LSTM	94.30	91.23	93.17	91.46	0.97	0.1430	0.1980	1792
GloVe + Bi-GRU	**94.86**	**92.80**	**94.80**	**93.62**	**0.97**	0.1221	0.2012	1408
GloVe + CNN	92.74	90.64	91.64	90.52	0.95	0.0825	0.3344	572
GCB-SA (Proposed)	**96.19**	**94.28**	**95.36**	**96.10**	**0.99**	0.1675	0.2801	2080
FastText + RNN	74.22	73.15	76.24	66.35	0.77	0.5984	0.6179	3964
FastText + LSTM	69.54	70.38	72.45	64.44	0.72	0.6214	0.6181	812
FastText + GRU	70.47	71.56	72.69	68.53	0.73	0.6211	0.6172	820
FastText + Bi-LSTM	**94.56**	**91.70**	**93.81**	**92.70**	**0.97**	0.1512	0.1946	1704
FastText + Bi-GRU	**95.24**	**93.32**	**94.23**	**94.28**	**0.98**	0.1363	0.2621	1556
FastText + CNN	93.19	88.21	89.45	89.24	0.96	0.2763	0.2652	460
FCB-SA (Proposed)	**97.26**	**96.86**	**95.62**	**98.60**	**0.99**	0.0836	0.2327	1760

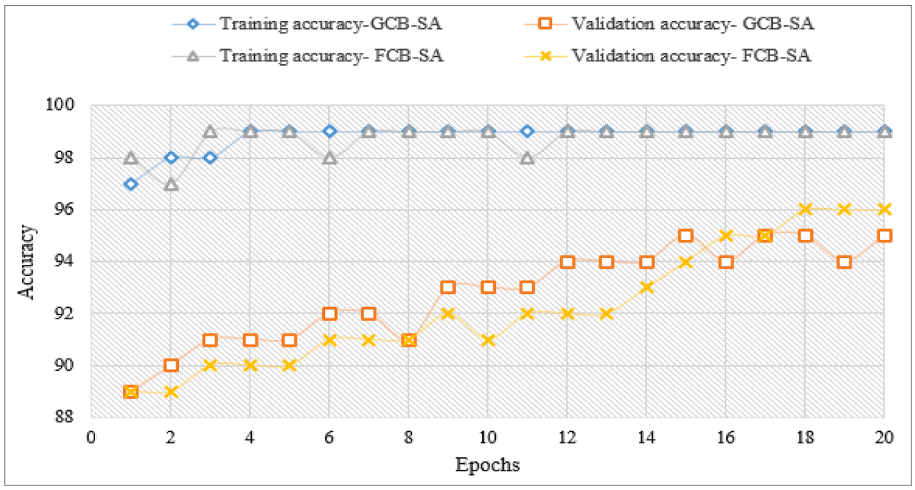


Fig. 11. Comparison of training accuracy and validation accuracy for proposed DL-based hybrid approaches, i.e., GCB-SA and FCB-SA on the balanced *combined dataset.*

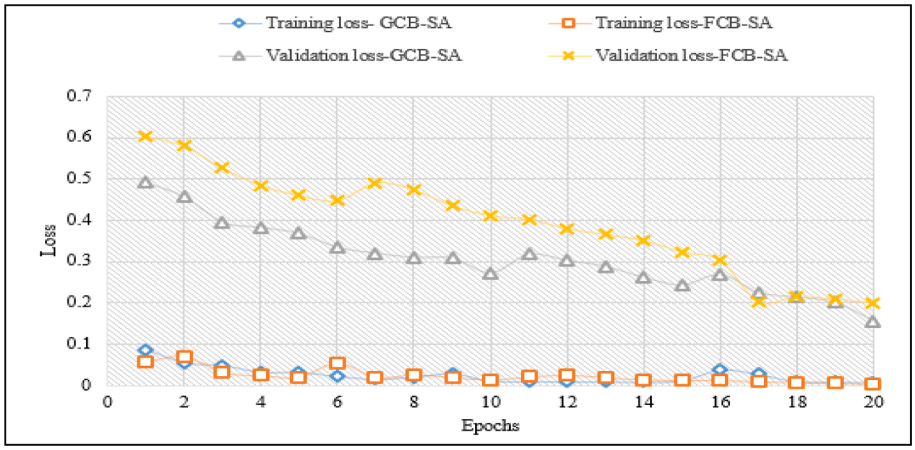


Fig. 12. Comparison of training loss and validation loss for proposed DL-based hybrid approaches, i.e., GCB-SA and FCB-SA on the balanced *combined dataset.*

individual datasets and a large corpus from several benchmark datasets. The exploratory data analysis and the dataset summary are provided to gain more insights into the *combined dataset*. The imbalanced datasets are balanced using two dataset-balancing techniques, ST and NM. Several ML models are trained and tested using word vectors obtained from basic BoW and TF-IDF techniques. Later, various ML models are used to give final results in the proposed soft and hard voting ensemble techniques. The effective results are obtained after balancing the datasets, particularly with the ST data resampling technique. In proposed hybrid DL-based approaches, GloVe and FastText pre-trained word embeddings provided improved results. Extensive DL experimentation is done to gain optimized values of hyperparameters for efficient performance. The self-attention mechanism is incorporated to enhance performance by emphasizing only the important words. From both ML and

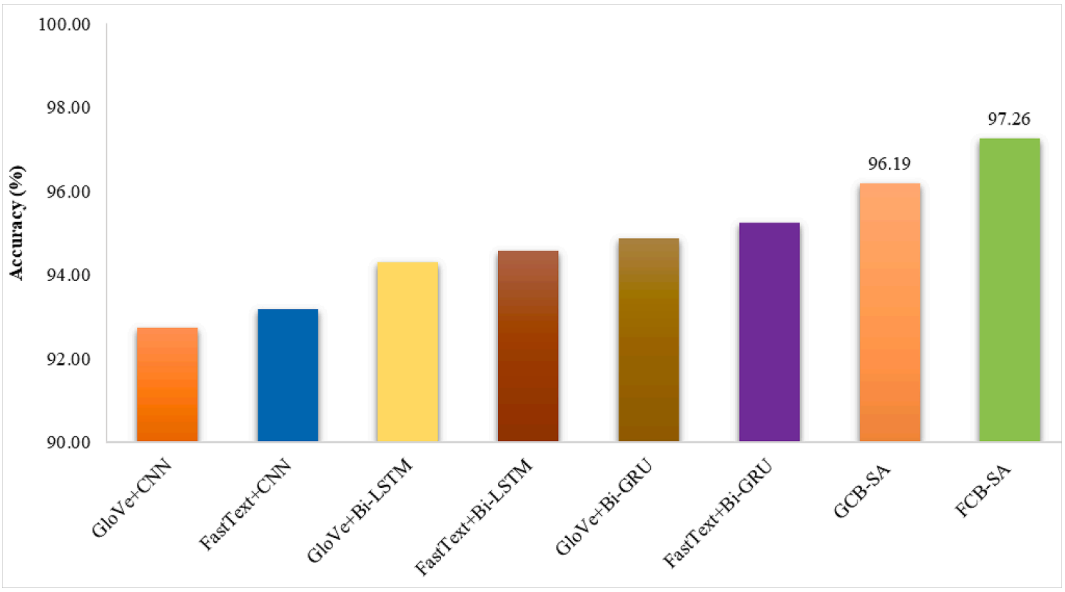


Fig. 13. Consolidated accuracy analysis of proposed hybrid DL approaches (GCB-SA and FCB-SA) with other good-performing DL approaches with an accuracy of 90% or above on the balanced *combined dataset.*

Table 17
Performance comparison of proposed hybrid DL-based approaches with existing approaches.

References	Feature extraction/ Optimization technique	ML/DL Classifier	Dataset/ Balancing technique	A (%)
(Lai et al., 2022)	TF-IDF and Word2Vec	LR,NB,SVM, RF, CNN, LSTM	*Fake news dataset*	90.00
(Jain et al., 2018)	Word2Vec, WordNet and ConceptNet	SCNN Framework, KNN, NB, ANN, RF, SVM	*Twitter Dataset*	94.40
			SMS spam dataset	86.5
(Barushka & Hajek, 2020)	TF-IDF	MOEFS + RDNN	*Hyves dataset*	90.02
			Twitter dataset	95.60
GCB-SA (Proposed)	GloVe	Conv1D, Bi-GRU, Bi-LSTM, SA mechanism	*Combined dataset with ST*	96.19
FCB-SA (Proposed)	FastText	Conv1D, Bi-GRU, Bi-LSTM, SA mechanism	*Combined dataset with ST*	97.26

DL approaches, we gained satisfactory results. However, DL approaches are faster and outperform as compared to ML approaches. The experimentation details and the results recorded are presented in the experimentation and results section to provide a comparative analysis of the used approaches on various datasets. Finally, a comprehensive discussion section offers insights into the proposed work regarding reasoning, pros, cons, and future research directions. This paper resolves some prominent issues for social spam detection. However, still, improvements can be made by incorporating contextual word embeddings and transformer-based DL approaches.

Funding

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

CRediT authorship contribution statement

Sanjeev Rao: Visualization, Conceptualization, Methodology, Data curation, Investigation, Writing – original draft. **Anil Kumar Verma:** Supervision, Writing – review & editing. **Tarunpreet Bhatia:** Supervision, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Agarwal, K., & Kumar, T. (2019). Email Spam Detection Using Integrated Approach of Naïve Bayes and Particle Swarm Optimization. *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018*, 685–690. https://doi.org/10.1109/ICCONS.2018.8662957.

Ahmed, H., Traore, I., & Saad, S. (2018). Detecting opinion spams and fake news using text classification. *Security and Privacy, 1*(1), e9.

Aiyar, S., & Shetty, N. P. (2018). N-Gram Assisted Youtube Spam Comment Detection. *Procedia Computer Science, 132*, 174–182. https://doi.org/10.1016/J.PROCS.2018.05.181

Alam, F., Cresci, S., Chakraborty, T., Silvestri, F., Dimitrov, D., Da, G., Martino, S., Shaar, S., Firooz, H., & Nakov, P. (2021). *A Survey on Multimodal Disinformation Detection*. https://doi.org/10.48550/arxiv.2103.12541.

Albalawi, Y., Buckley, J., & Nikolov, N. S. (2021). Investigating the impact of pre-processing techniques and pre-trained word embeddings in detecting Arabic health information on social media. *Journal of Big Data, 8*(1). https://doi.org/10.1186/s40537-021-00488-w

Alberto, T. C., Lochter, J. V., & Almeida, T. A. (2016). TubeSpam: Comment spam filtering on YouTube. *Proceedings - 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA 2015, January*, 138–143. https://doi.org/10.1109/ICMLA.2015.37.

Almeida, T. A., & Hidalgo, J. M. Ga. (2016). *UCI Machine Learning Repository: SMS Spam Collection Data Set*. https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A. S., & Asari, V. K. (2019). A state-of-the-art survey on

deep learning theory and architectures. In *Electronics (Switzerland)* (Vol. 8, Issue 3, p. 292). MDPI AG. https://doi.org/10.3390/electronics8030292.

Bahdanau, D., Cho, K. H., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. https://doi.org/10.48550/arxiv.1409.0473.

Ban, X., Chen, C., Liu, S., Wang, Y., & Zhang, J. (2019). Deep-learnt features for Twitter spam detection. *International Symposium on Security and Privacy in Social Networks and Big Data (SocialSec), 2018*, 208–212. https://doi.org/10.1109/socialsec.2018.8760377

Barushka, A., & Hajek, P. (2018). Spam filtering in social networks using regularized deep neural networks with ensemble learning. In *IFIP Advances in Information and Communication Technology* (Vol. 519)Springer International Publishing. https://doi.org/10.1007/978-3-319-92007-8_4.

Barushka, A., & Hajek, P. (2020). Spam detection on social networks using cost-sensitive feature selection and ensemble-based regularized deep neural networks. *Neural Computing and Applications, 32*(9), 4239–4257. https://doi.org/10.1007/S00521-019-04331-5/TABLES/8

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics, 5*, 135–146. https://doi.org/10.1162/tacl_a_00051

Chawla, N.V., Bowyer, K.W., Hall, L.O., K. W. P. (2002). *SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research*. Journal of Artificial Intelligence Research. https://dl.acm.org/doi/10.5555/1622407.1622416.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. https://doi.org/10.48550/arxiv.1412.3555.

Cinelli, M., Quattrociocchi, W., Galeazzi, A., Valensise, C. M., Brugnoli, E., Schmidt, A. L., … Scala, A. (2020). The COVID-19 social media infodemic. *Scientific Reports, 10*(1), 16598. https://doi.org/10.1038/s41598-020-73510-5

Dargan, S., Kumar, M., Ayyagari, M. R., & Kumar, G. (2020). A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Archives of Computational Methods in Engineering, 27*(4), 1071–1092. https://doi.org/10.1007/s11831-019-09344-w

ElSayed, A., Kongar, E., Mahmood, A., Sobh, T., & Boult, T. (2018). *Neural Generative Models for 3D Faces with Application in 3D Texture Free Face Recognition*.

Fei, G., Li, H., & Liu, B. (2017). Opinion Spam Detection in Social Networks. In *Sentiment Analysis in Social Networks* (pp. 141–156). Elsevier Inc.. https://doi.org/10.1016/B978-0-12-804412-4.00009-7

Feng, B., Fu, Q., Dong, M., Guo, D., & Li, Q. (2018). Multistage and Elastic Spam Detection in Mobile Social Networks through Deep Learning. *IEEE Network, 32*(4), 15–21. https://doi.org/10.1109/MNET.2018.1700406

Frenkel, S., Alba, D., & Zhong, R. (2020). Surge of Virus Misinformation Stumps Facebook and Twitter. *The New York Times*. https://www.nytimes.com/2020/03/08/technology/coronavirus-misinformation-social-media.html.

Gallotti, R., Valle, F., Castaldo, N., Sacco, P., & De Domenico, M. (2020). Assessing the risks of 'infodemics' in response to COVID-19 epidemics. *Nature Human Behaviour*. https://doi.org/10.1038/s41562-020-00994-6

Ghaderi Zefrehi, H., & Altınçay, H. (2020). Imbalance learning using heterogeneous ensembles. *Expert Systems with Applications, 142*, Article 113005. https://doi.org/10.1016/J.ESWA.2019.113005

Ghatasheh, N., Faris, H., Abukhurma, R., Castillo, P. A., Al-Madi, N., Mora, A. M., … Hassanat, A. (2020). Cost-sensitive ensemble methods for bankruptcy prediction in a highly imbalanced data distribution: A real case from the Spanish market. *Progress in Artificial Intelligence, 9*(4), 361–375. https://doi.org/10.1007/s13748-020-00219-x

Gu, M. (2019). *Ling-Spam Dataset | Kaggle*. Com: Kaggle. https://www.kaggle.com/mandygu/lingspam-dataset/metadata.

Gupta, H., Jamal, M. S., Madisetty, S., & Desarkar, M. S. (2018). A framework for real-time spam detection in Twitter. *2018 10th International Conference on Communication Systems and Networks, COMSNETS 2018, 2018-Janua*(January), 380–383. https://doi.org/10.1109/COMSNETS.2018.8328222.

Heydari, A., Tavakoli, M. A., Salim, N., & Heydari, Z. (2015). Detection of review spam: A survey. *Expert Systems with Applications, 42*(7), 3634–3642. https://doi.org/10.1016/j.eswa.2014.12.029

Hussain, N., Turab Mirza, H., Rasool, G., Hussain, I., & Kaleem, M. (2019). Spam Review Detection Techniques: A Systematic Literature Review. *Applied Sciences, 9*(5), 987. https://doi.org/10.3390/app9050987

Jain, G., Sharma, M., & Agarwal, B. (2018). Spam Detection on Social Media Using Semantic Convolutional Neural Network. *International Journal of Knowledge Discovery in Bioinformatics, 8*(1), 12–26. https://doi.org/10.4018/ijkdb.2018010102

Jain, G., Sharma, M., & Agarwal, B. (2019). Spam detection in social media using convolutional and long short term memory neural network. *Annals of Mathematics and Artificial Intelligence, 85*(1), 21–44. https://doi.org/10.1007/S10472-018-9612-Z

Kardaş, B., Bayar, I. E., Özyer, T., & Alhajj, R. (2021). Detecting spam tweets using machine learning and effective preprocessing. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. https://doi.org/10.1145/3487351.3490968

Lai, C. M., Chen, M. H., Kristiani, E., Verma, V. K., & Yang, C. T. (2022). *Fake News Classification Based on Content Level Features. Applied Sciences (Switzerland), 12*(3). https://doi.org/10.3390/app12031116

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. In *Nature* (Vol. 521, Issue 7553, pp. 436–444). Nature Publishing Group. https://doi.org/10.1038/nature14539.

Liu, B., Sun, · Xiangguo, Ni, Z., Cao, J., Luo, · Junzhou, Liu, B., Fu, · Xinwen, Liu, A., Liu, G., Orgun, M. A., & Li, Q. (2020). *Co-Detection of crowdturfing microblogs and spammers in online social networks. 23*, 573–607. https://doi.org/10.1007/s11280-019-00727-4.

Madisetty, S., & Desarkar, M. S. (2018). A Neural Network-Based Ensemble Approach for Spam Detection in Twitter. *IEEE Transactions on Computational Social Systems, 5*(4), 973–984. https://doi.org/10.1109/TCSS.2018.2878852

Mehmood, A., On, B. W., Lee, I., Ashraf, I., & Sang Choi, G. (2018). Spam comments prediction using stacking with ensemble learning. *Journal of Physics: Conference Series, 933*(1). https://doi.org/10.1088/1742-6596/933/1/012012

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1St International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2017). Advances in Pre-Training Distributed Word Representations. In *LREC 2018–11th International Conference on Language Resources and Evaluation* (pp. 52–55).

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2020). *Deep Learning Based Text Classification: A Comprehensive Review., 1*(1), 1–43. https://doi.org/10.1145/3439726

Naseem, U., Khalid Khan, S., & Prasad, M. (2021). A Comprehensive Survey on Word Representation Models: From Classical to State-of-the-Art Word Representation Language Models. *ACM Transactions on Asian and Low-Resource Language Information Processing, 20*. https://doi.org/10.1145/3434237

Nasir, J. A., Khan, O. S., & Varlamis, I. (2021). Fake news detection: A hybrid CNN-RNN based deep learning approach. *International Journal of Information Management Data Insights, 1*(1), Article 100007. https://doi.org/10.1016/j.jjimei.2020.100007

Niu, Z., Zhong, G., & Yu, H. (2021). A review on the attention mechanism of deep learning. *Neurocomputing, 452*, 48–62. https://doi.org/10.1016/J.NEUCOM.2021.03.091

Ott, M., Choi, Y., Cardie, C., & Hancock, J. T. (2011). Finding Deceptive Opinion Spam by Any Stretch of the Imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 309–319).

Patwa, P., Sharma, S., Pykl, S., Guptha, V., Kumari, G., Akhtar, M. S., … Chakraborty, T. (2021). Fighting an Infodemic: COVID-19 Fake News Dataset. In *Communications in Computer and Information Science*. https://doi.org/10.1007/978-3-030-73696-5_3

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1532–1543. https://doi.org/10.3115/v1/d14-1162.

Rao, S., Kumar Verma, A., & Bhatia, T. (2021). A Review on Social Spam Detection: Challenges, Open Issues, and Future Directions. *Expert Systems with Applications, 115742*. https://doi.org/10.1016/j.eswa.2021.115742

Rao, S., Verma, A. K., & Bhatia, T. (2020a). Evolving Cyber Threats, Combating Techniques, and Open Issues in Online Social Networks. In *Handbook of Research on Cyber Crime and Information Privacy* (pp. 219–235). IGI Global. https://doi.org/10.4018/978-1-7998-5728-0.ch012.

Rao, S., Verma, A. K., & Bhatia, T. (2020b). Online Social Networks Misuse, Cyber Crimes, and Counter Mechanisms. In *Analyzing Global Social Media Consumption: Vol. i* (pp. 183–203). IGI Global. https://doi.org/10.4018/978-1-7998-4718-2.ch010.

Sağlam, F., & Cengiz, M. A. (2022). A novel SMOTE-based resampling technique trough noise detection and the boosting procedure. *Expert Systems with Applications, 200*, Article 117023. https://doi.org/10.1016/J.ESWA.2022.117023

Saumya, S., & Singh, J. P. (2018). Detection of spam reviews: A sentiment analysis approach. *CSI Transactions on ICT, 6*(2), 137–148. https://doi.org/10.1007/s40012-018-0193-0

Sedhai, S., & Sun, A. (2018). Semi-Supervised Spam Detection in Twitter Stream. *IEEE Transactions on Computational Social Systems, 5*(1), 169–175. https://doi.org/10.1109/TCSS.2017.2773581

Sedhai, S., & Sun, A. (2015). Hspam14: A collection of 14 million tweets for hashtag-oriented spam research. In *SIGIR 2015 - Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 223–232). https://doi.org/10.1145/2766462.2767701

Singh, A., & Batra, S. (2018). Ensemble based spam detection in social IoT using probabilistic data structures. *Future Generation Computer Systems, 81*, 359–371. https://doi.org/10.1016/j.future.2017.09.072

Tao, X., Zheng, Y., Chen, W., Zhang, X., Qi, L., Fan, Z., & Huang, S. (2022). SVDD-based weighted oversampling technique for imbalanced and overlapped dataset learning. *Information Sciences, 588*, 13–51. https://doi.org/10.1016/J.INS.2021.12.066

Tolba, M., Ouadfel, S., & Meshoul, S. (2021). Hybrid ensemble approaches to online harassment detection in highly imbalanced data. *Expert Systems with Applications, 175*(January), Article 114751. https://doi.org/10.1016/j.eswa.2021.114751

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

Verma, P. K., Agrawal, P., Amorim, I., & Prodan, R. (2021). WELFake: Word Embedding over Linguistic Features for Fake News Detection. *IEEE Transactions on Computational Social Systems, 8*(4), 881–893. https://doi.org/10.1109/TCSS.2021.3068519

Wang, Z., Wu, C., Zheng, K., Niu, X., & Wang, X. (2019). SMOTETomek-Based Resampling for Personality Recognition. *IEEE Access, 7*, 129678–129689. https://doi.org/10.1109/ACCESS.2019.2940061

Zhang, X., Bai, H., & Liang, W. (2016). A social spam detection framework via semi-supervised learning. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9794*(61272374), 214–226. https://doi.org/10.1007/978-3-319-42996-0_18