International Journal of Information Management Data Insights 3 (2023) 100147

Contents lists available at ScienceDirect

International Journal of Information Management Data Insights

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

How AI Can be Used for Governance of Messaging Services: A Study on Spam Classification Leveraging Multi-Channel Convolutional Neural Network

Gopalkrishna Waja, Gaurang Patil*, Charmee Mehta, Sonali Patil

Department of Information Technology, K.J. Somaiya College of Engineering, Somaiya Vidyavihar University, Vidyanagri, Vidyavihar East, Mumbai 400077, India

ARTICLE INFO

Keywords:
Multi-channel
Convolutional neural network
Spam classification
Word embeddings
Natural language processing
SMS text messages

ABSTRACT

Over the past decade, there has been a meteoric evolution in Internet Messaging Services and although these services have become ingrained in our everyday life, SMS service remains an essential form of communication service. The omnipresence of SMS has also given rise to unsolicited and junk messages which has motivated researchers to use machine learning and deep learning to detect such spam messages. Studies using deep learning have shown promising results for spam classification, and in this paper, extending these studies, we have proposed a Multi-Channel CNN architecture with static and dynamic embeddings for SMS spam classification. UCI's SMS spam collection dataset along with several personally collected text messages are used to create a rich dataset for training the models. The proposed model has an accuracy of 96.12% and overcomes certain disadvantages associated with some of the state-of-the-art deep learning models.

1. Introduction

Short Message Service abbreviated as SMS is considered to be one of the quickest and most economical forms of communication. SMS has entrenched itself as both a formal and informal medium of communicating. Every day people receive a plethora of messages, some of which are solicited like messages from friends and acquaintances, bank transaction notifications, OTPs, and messages from other services, while others are unsolicited like promotional and phishing messages which are also known as spam messages. Sometimes these spam messages, although annoying, appear harmless but often they provide a medium to attackers for stealing users' private details, like card numbers, passwords, bank account details, etc., and hence spam message detection has become a necessity.

SMS spam classification is a problem that is still a subject of research for which lots of efforts have been made to present a simple yet robust solution. Many studies in the past have used machine learning techniques for text classification which although have given quite successful performance, have faced challenges like manual feature extraction by domain experts, difficulty in preserving word order and context, overfitting, and scalability (Minaee et al., 2021). For this reason, researchers have started to shift their attention towards deep learning approaches to obtain better performance, semantic and syntactic disambiguation, and scalability by the use of word vector representation and automated feature extraction.

Special networks are trained in a variety of Natural Language Processing (NLP) tasks to get words or sentence representations that are learned as vectors that encode the semantic and syntactic information about the input which are then used for automated extraction of higher-level features required for classification (Parwez, Abulaish, & Jahiruddin, 2019). Over the recent years, Convolutional Neural Network (CNN) models have portrayed tremendous potential for extraction of these high-level features and hence aid in text classification.

The aim of this research is to explore the possible application of Multi-channel CNN architecture (Kim, 2014) (and its variants with different orientations of convolution layers) for SMS spam classification and to compare its performance metrics and computational time with some of the state-of-the-art techniques present in the literature. We first start with the preparation of our dataset, where we try to overcome the issue of selection bias associated with the popular dataset used in the existing literature. The dataset is created by merging UCI's SMS Spam Collection Dataset (Almeida, Hidalgo, & Yamakami, 2011, 2012) and self-collected messages. The raw data collected is then pre-processed, followed by tasks like data augmentation, tokenization, stop-word

* Corresponding author.
E-mail addresses: gopalkrishna.w@somaiya.edu (G. Waja), gaurang.patil@somaiya.edu (G. Patil), charmee.m@somaiya.edu (C. Mehta), sonalipatil@somaiya.edu (S. Patil).

https://doi.org/10.1016/j.jjimei.2022.100147
Received 5 June 2022; Received in revised form 7 December 2022; Accepted 11 December 2022

Table 1
Dataset Description.

Dataset/Type	Spam	Non-Spam	Total
UCI Dataset	653	4516	5169
Self-Collected	2429	1153	3582
Total	3082	5669	8751

removal, and lemmatization to get an N-vector representation of the text. The N-vector is then fed to the variant models mentioned in Table 4 for classification. To gauge the model's effectiveness we have additionally trained and compared other common machine learning algorithms like logistic regression, SVM, decision trees, Naive Bayes, and variants of the proposed model by changing the number of channels and orientation. Experiments suggest that the model having 3 channels with convolutional layers arranged in parallel gives the best results- with an accuracy of 96.12%. The performance metrics are also compared with that of state-of-the-art techniques and the experiments suggest that the proposed model has an accuracy comparable to that of state-of-the-art techniques mentioned in the literature, with the added advantage of being computationally quicker with respect to training and testing, especially when compared to BERT-base-cased (Yaseen et al., 2021). The proposed Multi-Channel CNN architecture uses a combination of static pre-trained word embeddings and non-static random word embeddings (Kim, 2014). GloVe and Word2Vec are chosen as pre-trained embeddings for the network.

Further, we use the proposed model to experimentally highlight the need for using self-collected messages coupled with data augmentation as a means to build better models. These experiments are purely aimed at providing useful insights for future researchers who may want to understand the limitations associated with publicly available datasets for SMS spam classification.

The proposed model provides threefold benefits. Firstly, it uses multiple channels of word embeddings instead of a single or none which provides a varied non-sparse representation of the input required for better capturing the semantic information. Secondly, orienting the convolutional layers in parallel enables them to act as automatic n-gram feature detectors for each channel (Jacovi, Shalom, & Goldberg, 2020). Thirdly, the training and testing time associated with the model is significantly lesser when compared to state-of-the-art techniques like BERT, which makes it more suitable for deployment in real-time applications where low latency requirements are essential. Additionally, it is computationally lighter when compared to BERT.

The organization of the upcoming sections is as follows. Section 2.1 provides a brief review of the existing spam classification techniques and Section 2.2 outlines the importance of spam detection in information management and related work associated with the same. Following this, Section 3 provides details about the dataset and the proposed model. Then in Section 4 and 5 we have described the experimental setup and the experimental results along with the findings respectively. A discussion Section 6 talks about our contributions to literature and implications to practice. We end the paper by providing a conclusion in Section 7.

2. Literature Review

2.1. Methods For Spam Detection

The spam classification problem, over the past decade, has been studied widely by researchers over the world. The work done by them over these years can be broadly categorized into three approaches, namely rule-based techniques, machine learning techniques, and deep learning techniques.

Rule-based approaches are based on using pre-defined manually designed language rules, developed by domain experts, in form of regular expressions which identify characteristic phrases to distinguish spam from non-spam messages. So generally, in these systems, the corpus is scanned for these rules, and if matches are discovered, their weight is added to the overall score which is used for final classification (Kaddoura, Chandrasekaran, Popescu, & Duraisamy, 2022). In Luo, Liu, Yan, & He (2011), Luo Q has used rule-based extraction and filtering with dynamic adjustment of static rules on the Spam Assassin corpus containing 4150 spam and 1897 ham emails to optimize the spam filtering process. Using this approach gave them an accuracy of 98.5%, a false-positive rate of 0.42%, and a false negative rate of 4.7%. In Shrivastava & Bindu (2014), Shrivastava and Bindu have used a rule-based spam detection filter with pre-assigned weights in combination with the Genetic Algorithm for efficient spam detection which gave them an accuracy of 82.7% and a precision of 83.5%. Similarly, in Saidani, Adi, & Allili (2020), authors have utilized manually and automatically extracted rules from the Enron dataset with domain categorization to obtain accuracy and precision of 98.0% and 0.98 respectively. Although we can see that rule-based approaches provide good performance, they primarily depend on static rules and therefore cannot be generalized well to the ever-changing spam content, furthermore, it might not be always possible to identify all the rules required for spam detection. In addition to this, although automated rule generation systems are available to constantly modify rules based on changing spam content, their limitation is that they require a significantly larger amount of context-based features which increases time, analysis, and computational complexity. All these limitations associated with the rule-based approach have served as a motivation for the adoption of machine learning and deep learning for spam message classification.

In the case of machine learning, researchers have studied the effect of using some of the most popular algorithms like Naive Bayes, decision trees, random forest, support vector machines, and ensemble learning for spam classification. In Mishra & Soni (2020), the authors proposed a model named "Smishing Detector" containing different modules to analyze different parts of the message from UCI's SMS spam dataset. In this paper, the authors have used the naive Bayes classifier to obtain precision and accuracy of 91.6% and 93%. In another paper (Sonowal & Kuppusamy, 2018), Sonowal has proposed a model, called "SmiDCA", for the detection of SMS spam messages from the UCI SMS spam dataset in which he uses correlation analysis for feature extraction followed by the application of 4 machine learning algorithms, random forest, decision tree, support vector machine, and AdaBoost to perform a comparative analysis. Here, the best accuracy of 96.4% was obtained for the random forest model. In Fattahi & Mejri (2021), authors have proposed "SpaML" model in which feature extraction was done using a Bag of words and TF-IDF, and classification was performed using a majority of the results of an ensemble of 7 models i.e. multinomial naive Bayes, SVM, nearest centroid center, logistic regression, XGBoost, KNN, and perceptron. This model was trained on UCI's Spam dataset and gave an accuracy of 97%. In Jain & Gupta (2019), the authors propose ten features that they claim to be able to distinguish false messages from ham. After this, they test the effectiveness of these features by training and comparing random forest, naive Bayes, SVM and Neural Network. This indicated that random forest gave the highest true positive rate of 94.20% and 98.74% of overall accuracy.

Having studied the prior work done by researchers on spam classification using machine learning we can understand that these techniques provide us the ability to deal with dynamic conditions and help to counter the limitations posed by rule-based systems. Despite this, they have their own set of limitations. In general they either extract features manually using domain knowledge or use techniques like Bag of words to provide a representation that is often sparse. In particular, algorithms like support vector machines that analyze and identify patterns in data for classification may perform well on a small dataset, but fail to generalize well and become computationally impractical on larger datasets (Kaddoura et al., 2022). Other algorithms which make up tree structures like decision trees suffer from difficulties like controlling tree growth and over-fitting of training data. In probabilistic models like naive Bayes

classifier makes a simplistic assumption that each word in a message is independent of others which is rarely the case in real-world SMS messages. Finally, studies indicate that ensemble algorithms are frequently cursed by high computational complexity and domain dependence. To avoid such deterrents by ML techniques, researchers have lately focused their attention on deep learning techniques.

Deep learning models have taken the world by storm due to their applicability to myriad domains, including natural language processing, the ability to deal with the scalability issue, and extract the features automatically (Kaddoura et al., 2022). The most prominent deep learning models among NLP researchers are Recurrent Neural networks, Long Short Tern Memory (LSTM) and Convolutional Neural Networks (CNN) networks. In Kim (2014), Yoon Kim proposed different types of CNN model variants for text classification in general which over the years have formed the benchmark for using CNN for natural language data. In Roy, Singh, & Banerjee (2020), authors have used single-channel CNN and LSTM on the UCI's spam dataset and obtained an accuracy of 99.4%. Popovac, Karanovic, Sladojevic, Arsenovic, & Anderla (2018) have proposed a CNN-based architecture having a single convolution and pooling layer SMS spam classification which achieved an accuracy of 98.4%. In Jain, Sharma, & Agarwal (2019), authors used the LSTM network to SMS spam for the same dataset with 200 LSTM nodes and pre-trained word embeddings, to achieve an accuracy of 99.01%. In Yaseen et al. (2021), authors have fine-tuned BERT-base-cased transformer model on the open source SpamBase dataset with 5569 messages to obtain accuracy and F1 score of 98.67% and 0.9866 respectively. In Zhuang, Zhu, Peng, & Khurshid (2021), Deep Belief Networks (DBN) are used on WEBSPAM-UK2006 and UK2007 datasets to obtain an accuracy of 94% and a precision of 0.95.

The presence of polysemous words is often one of the reasons for misclassification and using contextual embeddings and transfer learning architectures helps overcome this limitation. However, they generally cannot be used in real-world scenarios as they are slow to train and evaluate, and hence in Lester, Pressel, Hemmeter, Choudhury, & Bangalore (2020) authors find that by combining these embeddings, we can create richer representations of the word without the high computational overhead required by contextual alternatives and provide similar results. In Yin & Schtze (2016), authors propose Multichannel Variable-Size CNN (MVCNN), where they combine different pre-trained versions of word embeddings for richer text representation and also vary the size of the convolution filters, which provides promising results for sentiment classification and subjectivity classification. The promising results obtained suggest that similar architectures could possibly be researched for spam classification as well. Though there are many other studies where researchers have leveraged deep learning techniques for the spam classification problem, to our best knowledge, although they have studied spam classification using single word embedding and CNN, no one has yet explored the effect of using Multi-channel CNN architecture with different orientations of convolution layers for SMS spam classification which we aim to do in this paper.

2.2. Importance of Spam Detection in Information Management

Automation of spam detection is a significant research problem in information management, owing to the overload of information that we face today, and often, from the perspective of the security of the users as well as businesses. Detecting spammers on social media is also crucial to ensure users get a good user experience. While some spammers often promote content on social media to gain traction, others spread misinformation to serve their personal interests. Spreading misinformation on social media poses a significant threat to businesses, governments, and individuals in multiple ways.

There has been a significant amount of work associated with the detection of spam and misinformation on social media. In Aswani, Kar, & Vigneswara Ilavarasan (2018) authors have analysed 18,44,701 tweets (sourced from 14,235 profiles) using social media analytics in tandem with bio-inspired algorithms to detect spammers with an accuracy of 97.98%. Authors in Aswani, Kar, & Ilavarasan (2019) give several insights into managing misinformation on social media. They conclude that the polarity and the emotions associated with the tweet are crucial factors helping determine if the tweet propagates misinformation. Ansar & Goswami (2021) find that supervised learning approaches to detect fake news have been used quite frequently in research (e.g. authors in Chauhan & Palivela, 2021 use LSTM neural network), with the disadvantage being that the models are domain-specific and may not generalize well if tested in other domains. This particular issue is addressed in Nasir, Khan, & Varlamis (2021), where the authors have proposed a hybrid CNN-RNN model which is tested on multiple datasets for fake news detection and provides good generalization. Their proposed approach involves, firstly, the feeding of word embeddings to the CNN for local feature extraction. The extracted features are then fed to the RNN layer, where the LSTM units model the long-term dependencies present in the extracted features. Authors in Ansari & Gupta (2021) interestingly pointed out that it is more important to understand a customer's perception of product reviews rather than simply detecting fake reviews by using artificial intelligence because a customer uses their own subjective perception to decide if a review is genuine or not.

Spam detection also finds its application in e-commerce. E-commerce sites often automate the segregation of authentic reviews from fake ones to ensure a fair shopping environment is provided to both- shoppers and sellers. Companies with vested interests tend to use fake reviews to damage the reputation of products built by other companies. Such unethical practices may also cause users to distrust the e-commerce site itself, thus making it extremely important for the sites to automate the detection of fake reviews. Models built in Xue, Wang, Luo, Seo, & Li (2019) indicate that user trustworthiness can be deciphered from the reviews provided by the user. Authors of the survey (Mewada & Dewang, 2021) observe that automated feature extraction using deep learning methods gives promising results on a variety of datasets used for fake review detection. Their study involves fake review detection in domains of hotel and e-commerce.

Our experimental findings discussed in Section 5 lead us to believe that though the present paper proposes a spam detection model trained on SMS text messages, the usage of similar Multi-channel CNN models can possibly be extended to the research problems discussed above and the reasoning for the same has been discussed later in Section 6.

3. Approach Followed

In this section, we will discuss the approach followed for the development of the spam message classifier. First, we describe the dataset which we have used for the spam classification task, the data sources whose amalgamation dataset is, and the composition of these individual data sources. After this, we elucidate the data pre-processing and augmentation pipeline through which the textual messages pass before they are fed to the spam classification model for training and testing. Finally, we have explained the proposed multi-channel CNN model containing static and non-static embeddings for spam classification.

3.1. Preparing Dataset

A closer look at the UCI's spam dataset (Almeida, Hidalgo, & Yamakami, 2012), which is popular and used in several existing research papers, suggests that most of the legitimate(non-spam) messages present in the dataset are sourced from informal conversations. However, in a realistic scenario, we know that important and formal legitimate messages might be sent by banks, network service providers, and other entities so as to provide alerts or updates to their customers/users. It is fair to hypothesize that the dataset used in most of the existing literature is prone to selection bias. Therefore, in addition to the existing data available in the dataset, we use several personally collected SMS text messages to create a rich dataset having a variety of messages like those

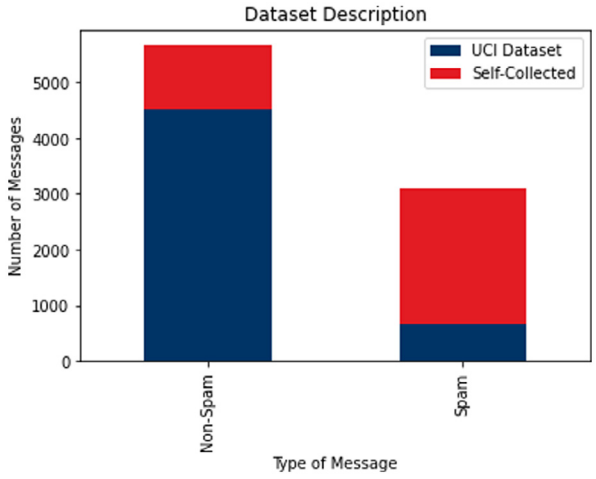

Fig. 1. Stacked bar plot for dataset label count.

from banks, network service providers, and other such entities which further reinforces the significance of our work. The dataset thus used is a combination of a pre-existing dataset and a self-created dataset formed by labeling messages collected from our mobile phones over 6 months.

The pre-existing dataset is UCI's public dataset containing 5169 unique labeled SMSs out of which, as visible in Fig. 1, 653 are spam and 4516 are non-spam. In addition to this our self-collected dataset contains 3582 unique messages with 2429 spam and 1153 non-spam messages. So, cumulatively the final dataset contains 8751 messages out of which 3082 are spam and 5669 are non-spam. The dataset details are tabulated in Table 1.

3.2. Data Augmentation and Processing

Pre-processing and cleaning the data involves performing operations on the raw text data to transform it into a proper format so that meaningful information can be extracted from it. In our case pre-processing involved the removal of leading and trailing blank spaces, extraction of URLs, and replacement of digits and URLs with blank spaces. Following this data augmentation is performed on the training messages. Text data augmentation is aimed at improving the generalizability of the model by increasing the number and variety of training examples which in turn prevents the model from overfitting the data. For this paper, text data augmentation is done by duplication and producing 2 synthetic sentences, by replacement of words by their synonyms, for each sentence in the training set facilitating an augmentation from 7875 training messages to 15,750 messages. Once augmentation is done tokenization followed by stop-word removal and lemmatization is performed. Tokenization is the process of decomposing a sentence into individual characters, words, or n-grams. For our study, we have used word tokenization using whitespace as delimiters. In stopword removal, we discard the most commonly used words in a language or corpus which generally do not provide any meaningful information required for the classification. Following this, in lemmatization, we aim to convert inflectional forms to root words. An SMS may contain different inflections like running, runs, ran, etc of the same base word ærung, lemmatization uses a dictionary and morphological analysis on these words, to eliminate inflections to obtain the root word.

Finally, once the lemmatization of the sentence is done, the sequence lemmas are mapped to a sequence of numbers unique to each lemma to obtain an N-vector representing the processed text.

3.3. Model Architecture

In this section, we will describe the multi-channel CNN model architecture which we have used for spam message classification. The different vector operations performed are shown in Fig. 2 and the overall architecture implemented along with layers of different CNN models is shown in Fig. 3. It can be seen that the model consists of three channels, corresponding to Word2Vec, Random, and GloVe embeddings respectively. The model takes a sequence of numeric tokens as input and uses trainable embeddings and high-level features extracted through convolution to classify the spam messages. An important feature of this architecture is that it uses a combination of pre-trained and dynamically generated word embeddings which are fine-tuned during training to make it more specific to the task at hand and provide a more meaningful representation of the tokens encountered in the spam messages. In addition to this, the model also arranges the convolution layers in parallel instead of arranging them in series. Specific details regarding each component of the model are as follows:-

3.3.1. Input Layer

The model takes an N-vector as the input which contains a sequence of numeric tokens obtained after processing and padding the text data. For our model N is equal to 76 and therefore the input layer accepts a 76-vector representing a text message as the input, as depicted in the Eq. (1).

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_N \end{bmatrix} \tag{1}$$

where $x_1, x_2, \ldots x_N$ are numeric tokens.

3.3.2. Embedding Layer

The embedding layer maps input vector $\mathbf{X} \in \mathbb{Z}^N$ containing sequences of tokens to a word embedding matrix $\mathbf{E} \in \mathbb{R}^{N,K}$ where $\mathbf{K}$ is the length of each word embedding and each row of the $\mathbf{E}$ is a word vector corresponding to the token in the input vector, as depicted in (2). Word embedding matrix provides a representation that is fed as input to the upcoming convolution layers which learn high-level features useful to determine the class of these messages. We have used the concept of multi-channel embedding proposed in Kim (2014), to obtain better word representation and improved performance by enabling better semantic and syntactic disambiguation. In our model, each input is mapped to 3 embeddings namely to GloVe, Random, and Word2Vec embeddings corresponding to the 3 channels respectively.

$$\mathbf{E_i} = \begin{bmatrix} e_{1,1} & e_{1,2} & . & . & e_{1,Ki} \\ e_{2,1} & e_{2,2} & . & . & e_{2,Ki} \\ . & . & . & . & . \\ . & . & . & . & . \\ e_{N,1} & e_{N,2} & . & . & e_{N,Ki} \end{bmatrix} \tag{2}$$

where,

i is the channel corresponding to the embedding

k_i is the dimension corresponding to the channel

Each row of E_i i.e $< e_{N,1}, e_{N,2}.e_{N,K_i} >$ is word vector corresponding to x_N.

The Word2Vec embeddings are 300-dimensional embeddings pre-trained on Google News data containing billions of words while GloVe embeddings are 100-dimensional embeddings pre-trained over millions of Wikipedia pages. Out of these millions of vector representations, the word vectors corresponding to the vocabulary of the text in the training set are extracted and used by the model. In case a particular word's GLoVe or Word2Vec vector representation is not found, a very common scenario in spam classification due to profuse usage of abbreviations and slang, then in such cases word is by default initialized with a null vector. The same initialization strategy is used for words encountered during testing which are not part of the vocabulary. In addition to this, the random embedding is also a 100-dimensional embedding initialized using a uniform distribution. Furthermore, it should be noted that although the random embedding is a non-static trainable embedding, the

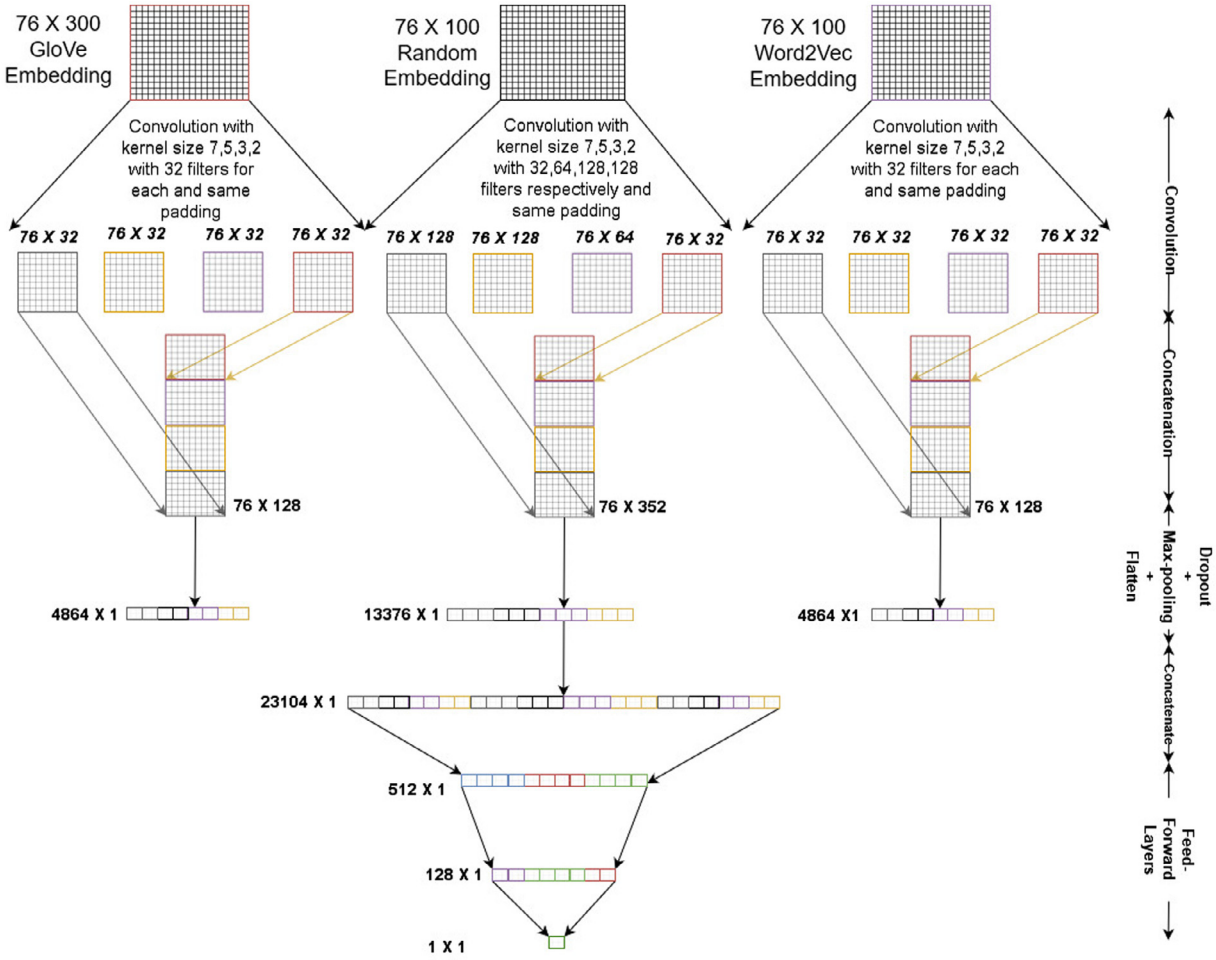

Fig. 2. CNN Model Vector Representation.

GloVe, and Word2Vec embeddings, keeping in mind the small size of the dataset, are kept static to prevent overfitting and also decrease computational complexity.

3.3.3. Convolution Layer and Concatenation

In the case of text classification, we need to perform 1D convolution to extract high-level n-gram patterns required for classification. Convolution filters act as heterogeneous n-gram detectors where each filter is trained to detect one or several families of closely related n-grams depending on the filter width (Jacovi et al., 2020). Thus using multiple convolution layers, with different kernel widths, in parallel to each other in such a manner that they act as a single layer with multiple kernels of varying sizes will allow the detection of n-gram phrases of different lengths required for classification. In our model, we have used 4 convolution layers in parallel per channel. Each channel uses convolution layers with kernel size $F_{i,j} \in \mathbb{R}^{w_j \cdot K}$, where $w_j \in 2, 3, 5, 7$ is the width of the kernel of jth layer, and $f_{i,j} \in \mathbb{Z}$ number of filters, where i is the channel number. Over here same convolution operation is performed, meaning the shape of the input remains the same after convolution. In order for this to happen the input matrix E_i must be P-zero padded to obtain $EP_i \in \mathbb{R}^{N+2P,K}$ where P, considering stride as 1, is as depicted below in Eq. (3).

$$\mathbf{P_{i,j}} = \frac{F_{i,j} - 1}{2} \tag{3}$$

Once the padding is done the output activation map $A_{i,j,z} \in \mathbb{R}^{N,1}$ where z is the filter number which produces the activation and each individual element a_n of $A_{i,j,z}$ is calculated as follows in Eqs. (4) and (5).

$$A_{i,j,z} = [a_1, a_2, \ldots, a_n] \tag{4}$$

$$a_n = g\left(EP_{n:n+w_j-1} * F_{i,j} + b_n\right) \tag{5}$$

where,

$EP_{n:n+w_j-1}$ represents using w_j rows at a time
$*$ is the convolution operator
b_n is the biased term
$g(x)$ is the ReLu activation function

Since at each layer there are $f_{i,j}$ filters of the same size $f_{i,j}$ such activation are produced which are stacked side-by-side to give a feature map of shape $(N, f_{i,j})$. After this, the feature maps of all 4 convolution layers are concatenated together to obtain C_i having final dimensions as mentioned below in (6).

$$\left(N, \sum_{j=1}^{4} f_{i,j}\right) \tag{6}$$

3.3.4. Dropout Layer

The dropout layer randomly masks out a set of neurons with the specified probability during training which helps to reduce co-adaptation of hidden units and regularize the network to provide better validation score and hence better generalizability.

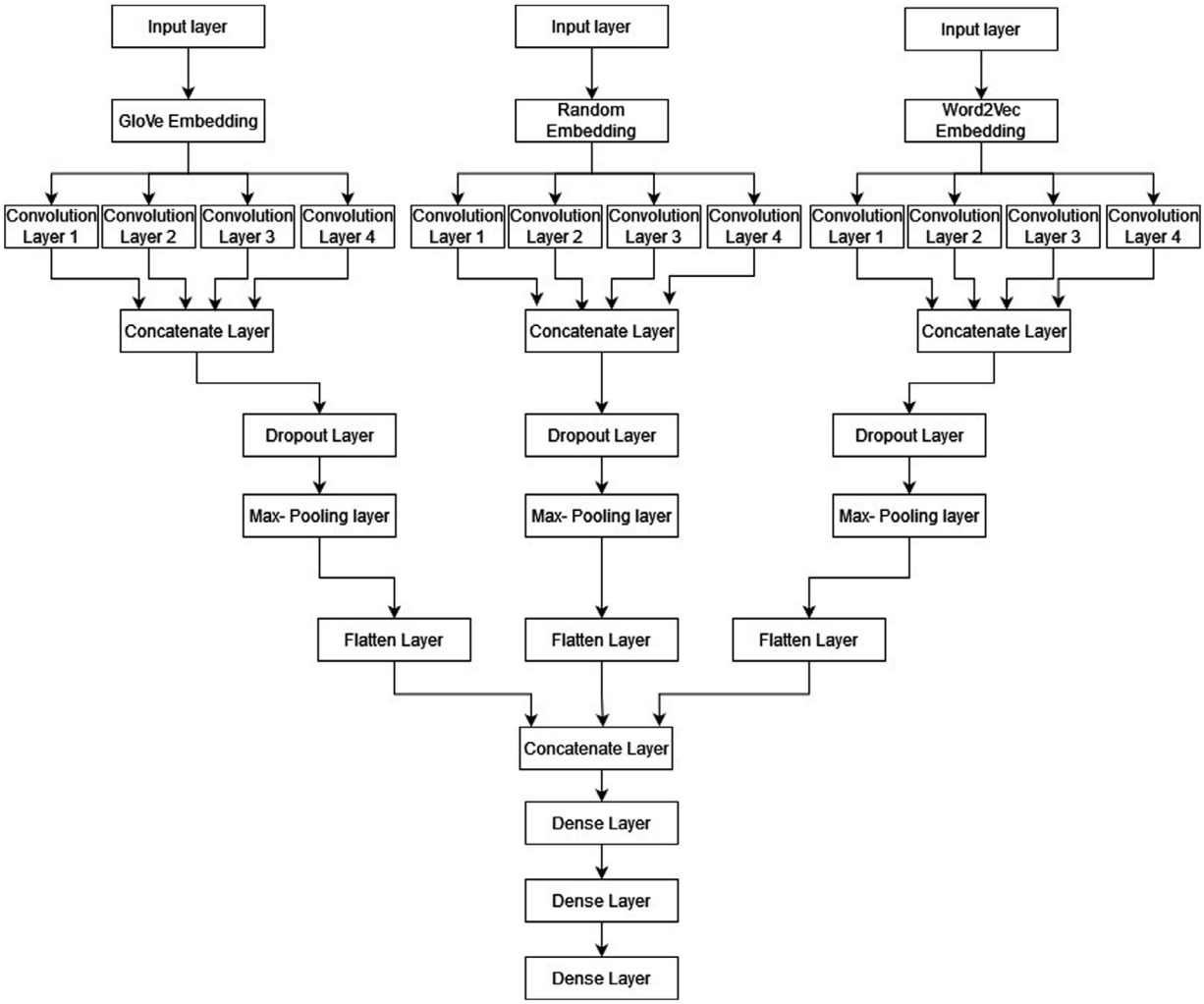

Fig. 3. CNN Model Architecture.

3.3.5. 1D MaxPooling Layer

This layer helps to reduce the dimensionality of the activation maps and hence reduces the number of parameters and computations required. 1D MaxPooling Layer takes in the concatenated output of the dropout layer and instead of performing Global Maxpooling, as depicted in Kim (2014) where 1 maximum element from the entire activation was chosen, here a window of size 2 with stride 2 is used for Maxpooling which effectively changes the shape of each $A_{i,j,z}$ in the concatenated output C_i from $(N,1)$ to $(\frac{N-2}{2}+1,\ 1)$ and this is repeated for all the activations giving the final shape of the output as shown in (7).

$$\left(\frac{N-2}{2}+1, \sum_{j=1}^{4} f_{i,j}\right) \tag{7}$$

3.3.6. Flatten and Dense Layers

The flatten layer takes the output matrix of the pooling layer and converts it into a vector $D_i \in \mathbb{R}^{(\frac{N-2}{2}+1)\cdot\sum_{j=1}^{4} f_{i,j}}$. Following this, the model concatenates all the flattened layers from the 3 channels on top of one another as shown in (8).

$$\mathbf{D} = D_1 \oplus D_2 \oplus D_3 \tag{8}$$

where $\oplus$ is the concatenation operator. Following this D is passed through a fully connected feed-forward network containing two dense layers which perform the operation depicted in (9).

$$Y = g(W_2 \cdot g(W_1 \cdot D + b_1) + b_2) \tag{9}$$

where W_1, b_1 and W_2, b_2 are weights and biases corresponding to the 1st and 2nd dense layer with 512 and 128 neurons respectively. Finally, the output layer has 1 neuron with a sigmoid activation function which outputs the probability of the message being spam, as shown in (10).

$$O = Sigmoid(Y) = \frac{1}{1+e^{-Y}} \tag{10}$$

where, $O \in \mathbb{R} \mid 0 \leq O \leq 1$ and the final label l assigned to the input message is determined as shown in (11).

$$l = \begin{cases} 0 & O \leq 0.5 \\ 1 & O > 0.5 \end{cases} \tag{11}$$

where $l = 0$ indicates message is non-spam while $l = 1$ indicates message is spam.

4. Experimental Setup

The model described in the previous section is trained using binary cross-entropy as the loss function, 512 as the batch size, and Adam optimizer which makes use of both first and second-order moments for faster optimization. Table 2 gives the hyper-parameters which are used for training corresponding to the convolution layers in different channels. Other model hyper-parameters are tabulated in Table 3. During

Table 2
Convolution Layer Hyper-Parameters.

Channel Number	Conv Layer, Filter count, Width
1	(1, 32, 2) (2, 32, 3) (3, 32, 5) (4, 32, 7)
2	(1, 128, 2) (2, 128, 3) (3, 64, 5) (4, 32, 7)
3	(1, 32, 2) (2, 32, 3) (3, 32, 5) (4, 32, 7)

Table 3
Other Model Hyper-Parameters.

Hyper-Parameter	Value
Word embedding size for GloVe, Random and Word2Vec	100,100 and 300
Dropout probability	0.5
MaxPooling Size	2
Early stopping patience	15
Dense 1 and Dense 2 neurons	512 and 128

training, the dropout regularization value was set to 0.5 and in addition to dropout, early stopping was also used as another regularization technique to prevent overfitting and obtain better validation accuracy. Early stopping halts the training when some fixed number of successive epochs do not provide an improvement on a validation set and uses the last best parameters which were stored during training.

Since the model has been trained on a dataset having self-collected messages it is difficult to compare the model's performance against other models trained on different datasets. So to put things into perspective, after training and testing the proposed model, other variants of the CNN model along with some common machine learning models, namely logistic regression, SVM, decision trees, and naive Bayes, have been trained on the same dataset and their performance is compared against the proposed model. For the sake of comparison, we also train and test some of the pre-existing state-of-the-art models using the new dataset.

The variant models are created by changing the number of channels and connection of convolution layers in series instead of parallel. When convolution layers are stacked on top of one another in form of a series the first convolution layer acts as the n-gram detector while the subsequent layers detect complex character-level features from the feature map of the first layer (Zhang, Zhao, & LeCun, 2015). This type of text convolution is generally helpful when analyzing long sequences of text but in the case of SMSs where the messages are usually short stacking convolution layers should not make much difference in performance, therefore we hypothesize those model variants where the number of channels is the same but the orientation of convolution layers is different will have comparable accuracies. In addition to this, we feel that, since the use of multiple embeddings helps to provide a more diverse representation of the words leading to better disambiguation, models with multiple embeddings will outperform the models with a lower number of embeddings. To verify these notions, we have trained and compared the performance of all these variant models in the next section.

To simplify the referencing of these models we have used the notation described in the Table 4. According to the convention decided, our model which is described in the earlier section with three channels and convolutions in parallel has the notation M3P. Similarly, some other models with 2 channels and convolutions in series will have the notation M2S. In the case of a model with 1 and 2 channels, for each model 3 sub-variants are possible based on the combination of the embeddings used. For M4S and M4P, we use 2 random embeddings along with GloVe and Word2Vec embeddings.

For comparing the models we have used accuracy, recall, and F1 score as the performance metrics. Accuracy is calculated using (12).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{12}$$

It indicates the fraction of the time the classifier is correct, that is when the classifier predicts an SMS is or is not spam, what is the probability that it's correct. Recall is calculated using (13).

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

Recall indicates how many spam messages the model can identify out of all the spam SMSs present in the test set. F1 score is calculated using (14).

$$F1 = \frac{2TP}{FP + 2TP + FN} \tag{14}$$

It is the harmonic mean of the precision and recall which helps us to compare multiple models having a different recall and precision values.

We compare the performance metrics of our proposed model to that of the machine learning models like logistic regression, decision tree, SVM, and Naive Bayes by training and testing these models on the dataset prepared by us. On the same dataset, we train and test state-of-the-art models like the BERT-base-cased architecture proposed in Yaseen et al. (2021) and LSTM, whose neural network architecture is the same as that proposed in Jain et al. (2019). However, it is important to note that the same N-vector representation of the text (derived by us after pre-processing) is fed to all the variant models, BERT-base-cased model, and LSTM. We also record the training and testing time associated with each of the Multi-channel CNN variants as well as the state-of-the-art techniques used for spam detection.

Though we have theoretically justified the need for including additional self-collected data in the dataset in Section 3, we now aim to verify our claim experimentally by means of a small experiment. To experimentally justify the need of using self-collected data as a means to achieve generalization (i.e. to justify that there is indeed a need for having a richer dataset comprising of a greater variety of messages), we first train and test our proposed model solely on the popular pre-existing dataset (Almeida, Hidalgo, & Yamakami, 2011, 2012). Next, we take a random sample of 250 messages each from both classes (so 500 samples in total) from the set of self-collected messages (which are manually labeled) and let the proposed model make its predictions to check how accurate it really is when confronted with data that is not part of the popular dataset. The idea behind this experiment is that if the pre-existing dataset (Almeida, Hidalgo, & Yamakami, 2011, 2012) is indeed ideal, then there would not be a very significant difference between the accuracy obtained on the test set (sourced from the popular dataset Almeida, Hidalgo, & Yamakami, 2011, 2012) and the accuracy obtained when confronted with our randomly sampled data sourced from the self-collected messages.

Table 4
Variant Model Notations.

	Single Channel	2 Channels	3 Channels	4 Channels
Convolution Layers arranged in Parallel	M1P	M2P	M3P	M4P
Convolution Layers arranged in Series	M1S	M2S	M3S	M4S

Table 5
Performance Metric of ML Models.

		Logistic Regression	Decision Tree	SVM	Naive Bayes
Train	Accuracy	0.6757	1.0000	0.8373	0.3607
Test	Accuracy	0.6941	0.7477	0.7728	0.3575
	Recall	0.6941	0.6447	0.7664	0.9770
	F1	0.5315	0.6395	0.7008	0.5147

Table 6
Performance Metric of Models with 1 Channel G:GloVe; R:Random; W:Word2Vec.

		M1S			M1P		
		G	R	W	G	R	W
Train	Accuracy	0.9630	0.9943	0.9389	0.9636	0.9945	0.9340
Test	Accuracy	0.9053	0.9509	0.9224	0.9189	0.9566	0.9167
	Recall	0.9111	0.8922	0.9017	0.9179	0.8938	0.8973
	F1	0.8557	0.9327	0.8867	0.8786	0.9410	0.8777

Table 7
Performance Metric of Models with 2 Channels G:GloVe; R:Random; W:Word2Vec.

		M2S			M2P		
		G+W	G+R	R+W	G+W	G+R	R+W
Train	Accuracy	0.9670	0.9947	0.9935	0.9722	0.9949	0.9942
Test	Accuracy	0.9121	0.9269	0.9532	0.9098	0.9532	0.9463
	Recall	0.8851	0.8320	0.9000	0.8576	0.8860	0.8750
	F1	0.8719	0.9042	0.9354	0.8728	0.9366	0.9276

Table 8
Performance Metric of Models with 3 Channels G:GloVe; R:Random; W:Word2Vec.

		M3S G+R+W	M3P G+R+W
Train	Accuracy	0.9924	0.9957
Test	Accuracy	0.9566	0.9612
	Recall	0.8938	0.9045
	F1	0.9410	0.9469

Further, to investigate the role and significance of data augmentation as a means to achieve generalization, we train our proposed model on the non-augmented training data (keeping the rest of the data preprocessing unchanged), so that the accuracy obtained can be contrasted with that which we obtain in the experiments which are conducted using augmented training data.

5. Results

In this section, we will compare the results obtained after training and testing the proposed model, the variant models, the machine learning models, and the state-of-the-art techniques as mentioned in the previous section. In addition to this, we will discuss the effect of change in the number of channels on the performance of the model and visualize the activations of the convolution layers of these variant models, by plotting the feature maps, to gain better insights about these models. Finally, we will discuss the results concerning the experiments conducted to show why there exists a dire requirement for using additional data as a means to achieve generalization and will also compare the performance of the proposed model trained using non-augmented data to that of the same model trained on augmented data.

The performance metrics obtained by training and testing Machine learning, 3-channeled, 2-channeled, and 1-channeled models are tabulated in the Tables 5,6,7,8 respectively. It can be seen that ML algorithms give a rather poor performance on the current dataset. The highest accuracy obtained for the ML algorithms is 77.28% with an F1 score of 0.7008 for SVM and the worst performance among ML algorithms is given by naive Bayes with accuracy and F1 score of 36.07% and 0.5147. It can be noticed that in the case of naive Bayes, the recall is high, that is, it can identify a large number of spam messages despite this its accuracy and F1 score is very low indicating that the trained model has a high bias and is almost every time predicting 1 irrespective of the message and therefore is not able to identify non-spam messages. Another observation that can be made is in the case of the Decision tree algorithm although the training accuracy of 100% is received the accuracy and F1 score on the test set is merely 74.77% and 0.6395, indicating the model is highly over-fitting the training examples and is not able to generalize well to unseen examples. Accuracies and F1-scores of the ML models can be visualised and compared in Fig. 6. From the Tables 6,7,8 it can be observed that these deep learning models give far better results than the machine learning models and the limitations of high bias and overfitting which were seen in the previous case seem to have been mitigated by these models. Comparing these three tables we can see that our proposed model M3P with an accuracy of 96.12% and F1 score of 0.9469 outperforms all other variant models, albeit, in some models variants like M1P with random embedding, M2P with GloVe and random, M2P with Word2Vec and random and M3S the difference in performance is not much. Accuracies and F1-scores of the variant models can be visualised and compared in Fig. 7.

It can also be seen that in the case of single-channeled models without random embedding some models have recall greater than our proposed M3P model although their accuracy and F1 score is lower than our model. This can be explained by the fact that single-channeled models with static embeddings provide a restricted word representation when compared to models with a higher number of channels leading to some bias and more frequent prediction of 1*s* than 0*s* causing higher recall but lower accuracy and F1 score. It can be observed that single-channeled models M1P and M1S have the least overall testing accuracy, M2P and M2S models' accuracy is slightly more than single-channeled models but less than the 3-channeled models. There are a few observations that bring out some important points regarding these variant models which helps us to explain why the M3P model outperforms the rest.

Firstly, we can see that in the case of single-channel models, using non-static random embedding gives better performance than using static

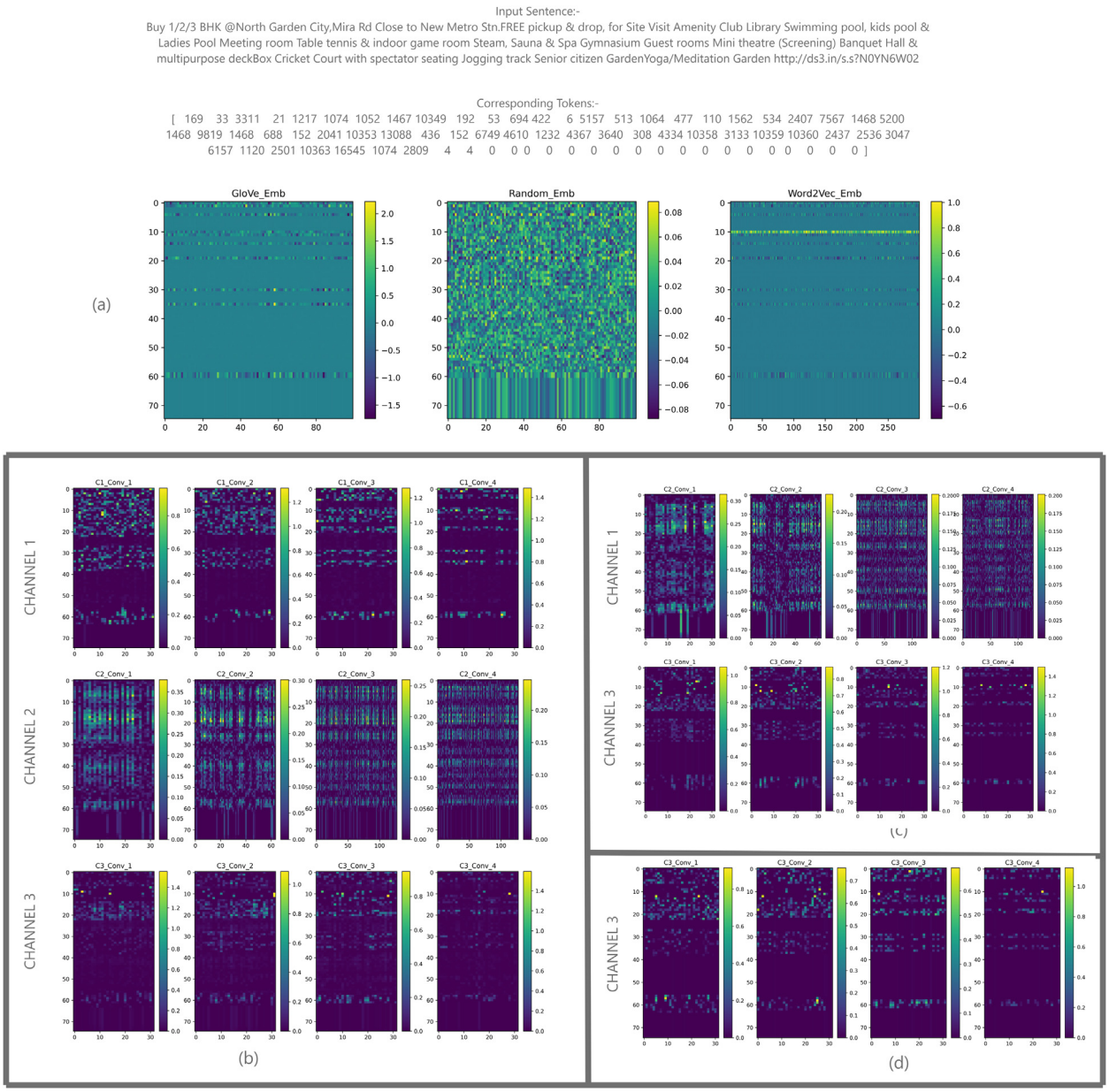

Fig. 4. Activation of Convolution layers is arranged in parallel; (a) Word embeddings corresponding to the GloVe, Random, and Word2Vec; (b) Activations of M3P Model (c) Activations of M2P Model (d) Activations of M1P Model.

pre-trained embeddings. A similar observation can be made for a model with two channels where a combination of random and pre-trained embeddings gave better accuracy than using pre-trained embedding in both channels. Generally, it is seen that using pre-trained embeddings allows efficient learning by capturing the semantic and syntactic meaning of a word and hence facilitating the improved performance of NLP models (Goldberg, 2017). Therefore, it is a natural instinct to think that pre-trained embeddings might outperform models with random embeddings, but here this is not the case. This observation can be, according to us, primarily explained as a consequence of three factors. Firstly, the pre-trained embeddings which are available, that is GloVe and Word2Vec, have been trained on Wikipedia and Google news data which is likely to be of a different nature compared to the data that is being used in the classification task at hand. For example, certain datasets may have an eclectic variety of abbreviations and slang and hence the semantic meaning and synthetic structure represented by the pre-trained embeddings might possibly be slightly different than that required for the task at hand. Secondly, these pre-trained embeddings were kept static to avoid overfitting and therefore are not fine-tuned for the task at hand. Finally, as evident in Fig. 4(a), in the case of pre-trained embeddings, most of the vectors are initialized to zero and these vectors correspond to those words which are not present in the vocabulary. To put things into perspective, for our dataset out of the 21,885 unique tokens present in the data set, Word2Vec embeddings were available only for 10,797 tokens, and the rest 11,088 tokens were initialized to null vectors. Thus we feel that the combination of the aforementioned factors is responsible for the poor performance of the pre-trained embeddings in a stand-alone manner. However, when the same pre-trained embeddings are used in combination with random embeddings, the random embeddings provide some representation for missing words that are non-static and are fine-tuned for the specific task to be performed leading to improved performance.

The next observation which can be made is that as the number of channels increases from 1 to 3, irrespective of the orientation of the convolution layers- the accuracy improves. This, according to us, is the effect of obtaining multiple representations of single words which rein-

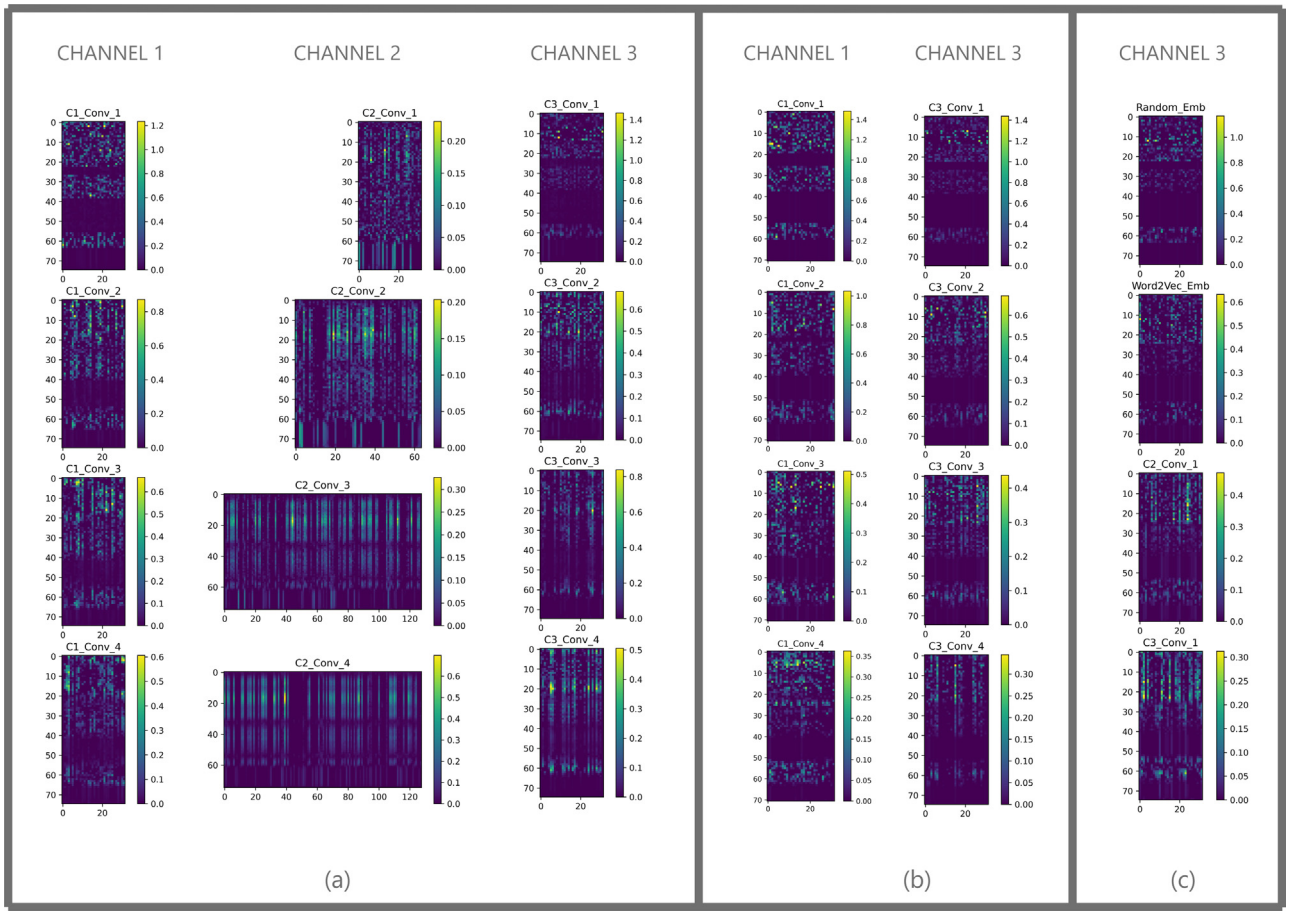

Fig. 5. Activation of Convolution layers is arranged in Series; (a) Activations of M3S Model (b) Activations of M2S Model (c) Activations of M1S Model.

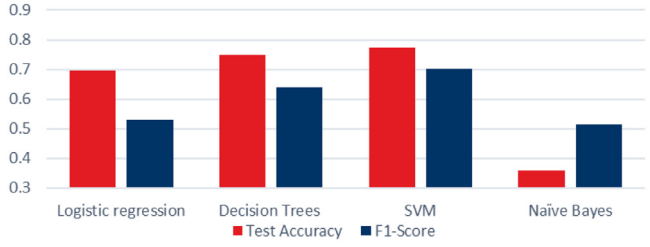

Fig. 6. Accuracies and F1 Score of ML Models.

Table 9
Performance Metric of Models with 4 Channels G:GloVe; R:Random; W:Word2Vec.

		M4S G + 2R + W	M4P G + 2R + W
Train	Accuracy	0.9985	0.9866
Test	Accuracy	0.9593	0.9586
	Recall	0.9823	0.953
	F1	0.9586	0.9443

forces the n-gram detection capabilities of the convolution filters at the various layers. This can be proved by observing the Activation of the CNN layer in Figs. 4 and 5. Figure 4(c) shows the activations of convolution layers placed in parallel for the M2P model with channel 1 and channel 3 and Fig. 4(d) shows the activations of parallel convolution layers for the M1P model with channel 3. From these figures, it can be seen that when channel 3 is used alone, its convolution layers can detect some n-gram patterns with some particular activation levels which are less than or equal to the level of activation for the same layers when used in combination with channel 1 in Fig. 4(c) and channel 1 and 2 in Fig. 4(b). A similar observation can be made in Fig. 5 This indicates that multiple channels or embeddings reinforce the activations which help in making more confident predictions. Now since the M3P and M3S models both have random embedding and multiple channels their convolution layers can detect text patterns more confidently leading to better accuracy.

Another important thing that can be observed is as hypothesized earlier, the performance of models with convolution layers in series and convolution in parallel is similar with only slight variation in accuracy occurring due to n-gram level and character level features extracted. Finally, we can see that in the case of single-channel models the difference between the training and testing accuracy is relatively large suggesting over-fitting of the model due to the usage of a single representation. But as the number of channels increases to two and three, it is seen that this difference becomes smaller and smaller leading to better testing accuracy and generalization.

The M3P model uses 3 channels with static pre-trained and non-static random embeddings which helps in the creation of an enhanced representation of the input, in turn aiding better learning, and reinforcing text pattern detecting filters. As a consequence, it becomes capable of providing better generalization to reduce overfitting and obtains the best accuracy out of the variant models.

Moving to M4S and M4P models, we can observe in Table 9 that accuracies comparable to M3S and M3P models are obtained but Tables 10 and 11 indicate 40.37% and 85.02% increase in training time of M4S and M4P models when compared to M3S and M3P models. Further, Table 12 indicates that the M3P model is significantly better than state-

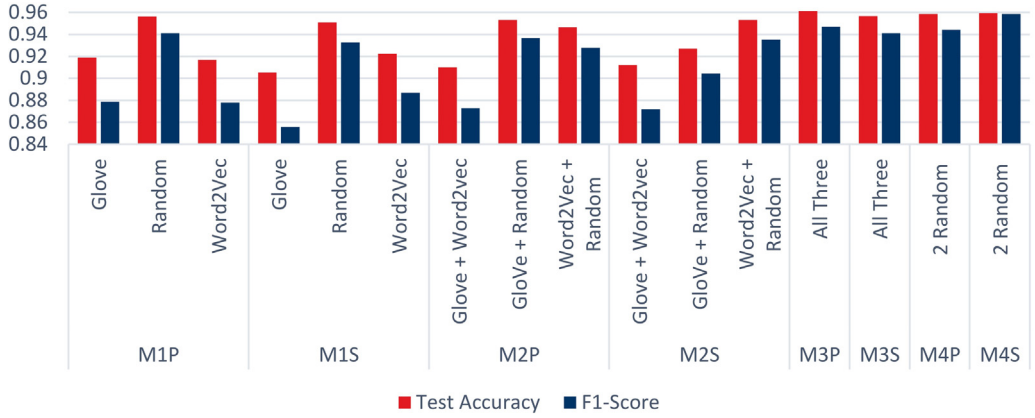

Fig. 7. Accuracies and F1 Score of Variant Models.

Table 10
Time Taken by Models (Series) on GPU for Training and Testing.

Model	Channels	Total Training Time (s)	Total Testing Time (s)
M1S	W	10.9481	0.0884
	R	16.6039	0.1094
	G	12.0196	0.0972
M2S	G+W	16.8921	0.1037
	G+R	13.6799	0.1133
	R+W	24.9511	0.1173
M3S	G+R + W	17.2144	0.1850
M4S	G+2R + W	24.1646	0.4335

Table 11
Time Taken by Models (Parallel) on GPU for Training and Testing.

Model	Channels	Total Training Time (s)	Total Testing Time (s)
M1P	W	15.2921	0.0991
	R	13.3467	0.1037
	G	24.4662	1.3247
M2P	G+W	31.4373	0.1765
	G+R	38.3124	0.2083
	R+W	27.4057	19.3056
M3P	G+R + W	30.8133	0.2212
M4P	G+2R + W	57.0113	0.4325

of-the-art models in terms of training and testing computational times, with BERT having training and testing time 22.63 and 20.12 times more than that of the M3P model and LSTM having training and testing time 1.0844 and 2.015 times more than that of M3P model. In general, the training process for BERT is slow because there are a huge number of model parameters (weights) that need to be updated during the training process.

The number of trainable parameters associated with the models helps us understand their computational resource requirements. A model with a larger number of trainable parameters needs more computational resources. It is observed in Table 12 that BERT has 4.7 times more trainable parameters than M3P, thus making the M3P model computationally lighter when compared to BERT. The M3P model has approximately 18 times more trainable parameters than LSTM, making LSTM computationally the lightest among the three models. Despite this, the drawback of the LSTM model is that it is not as quick as the M3P model when it comes to making decisions.

Table 13
Performance Metrics of M3P Model Trained Using UCI dataset.

		M3P Model
Train	Accuracy	0.9998
Test	Accuracy	0.9838
	Recall	0.9474
	F1	0.9231

Coming to the experiment associated with understanding the need for self-collected data as a means to achieve generalization; the results obtained when the M3P model is trained and tested on data sourced from the popularly used dataset alone (without including the self-collected samples) are tabulated in Table 13. It is seen that the testing accuracy obtained is 98.38% when the testing data is sourced from the popular pre-existing dataset. However, quite surprisingly, it is found that when the same model is confronted with 500 of our self-collected messages (250 each from the 2 classes), the model classifies only 40% of the total samples correctly. The model finds it easy to classify samples sourced from the same dataset but fails terribly when confronted with self-collected samples; indicating that the model is not being able to generalize well to data gathered from sources that are not a part of the dataset it was trained on. This can be explained by the fact that the dataset (Almeida, Hidalgo, & Yamakami, 2012) lacks diversity- it is composed of very few spam samples and the non-spam samples are mostly sourced from informal conversations. One of the possible reasons for the lack of diversity in the publicly available dataset could be that the contributors may have been hesitant to share certain text messages containing personally identifiable and sensitive information in order to preserve their privacy, unknowingly making the dataset prone to selection bias. This experiment justifies the dire need for using additional self-collected data so that both- the training and testing of the models can happen in an environment that is a good simulation of the real world and consequently build models which are likely to perform better when deployed in production.

Table 12
Comparison With Other Deep Learning Techniques.

Model	Total Training Time on GPU(s)	Total Testing Time on GPU(s)	Training Accuracy	Testing Accuracy	No. of Trainable Parameters
M3P	30.8133	0.2212	0.9957	0.9612	23,000,161
BERT	697.4268	4.4521	0.9916	0.9614	108,311,813
LSTM	33.4169	0.4459	0.9944	0.9590	1,278,002

Table 14
Performance Metrics of M3P Model Trained Using Augmented Data and Non-Augmented Data.

		M3P Model (Trained on Non-Augmented Data)	M3P Model (Trained on Augmented Data)
Train	Accuracy	0.9982	0.9957
Test	Accuracy	0.9429	0.9612
	Recall	0.9122	0.9045
	F1	0.9153	0.9469

We now come to our experiment concerned with understanding how significant data augmentation is, as a means to achieve generalization. Table 14 indicates that training the proposed M3P model on augmented data leads to better generalization. A significant increase in testing accuracy (from 94.29% to 96.12%) is observed when the model is trained using augmented data and it is, therefore, fair to conclude that data augmentation has indeed helped us build better models. Considering the fact that our dataset is not very large, the addition of synthetic data by means of data augmentation helps increase the number (and variety) of samples used for training the model. As a result, the model learns from a larger variety of data and makes better decisions when confronted with unseen data.

6. Discussion

6.1. Contributions to Literature

Electronic communication has been ingrained into our daily life, and its governance is a growing cause of concern. Though we are more digitally connected than ever, the propagation of unsolicited content is on a steep rise. The widespread popularity of messaging services has instigated spammers to use them as a means of propagating spam. With tremendous advances in artificial intelligence and deep learning over the last decade, businesses can leverage these techniques to improve their services and safeguard users and customers.

Text mining, in particular, has been extremely useful for the automated governance of electronic communication. The application of text mining in the field of management has been explored quite extensively in various business domains (Kumar, Kar, & Ilavarasan, 2021) and the authors assert that there is further scope for research on the applications of text mining in fake detection. Hyperpartisan news articles (articles that are politically biased) play a big role in manipulating public opinion and perception, and researchers have proposed deep learning techniques for their detection (Naredla & Adedoyin, 2022). More recently, researchers have also explored approaches that do not rely on textual content for the detection of such unethical practices. For example, authors in Michail, Kanakaris, & Varlamis (2022) have used graph convolutional networks to identify fake news campaigns without using the textual content of the news itself. Machine learning and deep learning techniques have also been leveraged for detecting hate speech and offensive content posted on social media (Khanday, Rabani, Khan, & Malik, 2022; Wadud et al., 2022).

Apart from the spread of fake news and offensive content, unsolicited content (like promotional spam, phishing spam, etc.) passed through various communication channels is another menace that needs to be dealt with. Spam reaches the public through various sources- social media, text messages, and emails being the most common. As discussed in Section 2.2, spam detection finds significant applications in the field of information management, especially in a world where almost every person is digitally connected and has access to tons of information via various communication channels. Researchers have explored different techniques and approaches for detecting spam in textual content (mainly using data from emails and text messages). For example, bio-inspired techniques have recently been explored for email spam detection (Batra, Jain, Tikkiwal, & Chakraborty, 2021). As discussed in Section 2.1, there has been a significant amount of work associated with the detection of SMS spam and deep learning techniques have proved to be very effective. Our work involves the application of deep learning for detecting SMS spam and extends the literature by proposing a deep learning based Multi-channel CNN model for SMS spam classification.

The proposed model takes advantage of the fact that the combination of multiple word embeddings results in a better representation of the text and eventually leads to better feature extraction, as supported by the results obtained in Yin & Schtze (2016). It overcomes some of the disadvantages associated with state-of-the-art models like BERT, which are relatively slow to train and require a relatively larger amount of time for making decisions. It is experimentally confirmed that the proposed model overcomes these limitations without any significant compromise in accuracy, which makes it suitable for deployment in real-time applications requiring low latency. Additionally, it is computationally lighter when compared to the state-of-the-art BERT model proposed in Yaseen et al. (2021). Moreover, as discussed in Section 3.1 and confirmed during our experiments, the popular dataset used for spam classification in the existing literature is prone to selection bias, and the present research addresses this by adding a significant number (and variety) of self-collected messages to the dataset. Finally, to the best of our knowledge, the present study is the first to study the effect of using different orientations of convolution layers in Multi-channel CNN for SMS spam classification.

6.2. Implications to Practice

The present paper deals with the usage of artificial intelligence for the governance of messaging services. Models used for spam detection can be integrated into mobile applications to automatically detect spam messages in real-time, without the explicit involvement of the user. Instant messaging applications may be considered the main beneficiaries of such models. Apart from detecting promotional spam, the models can be made capable of detecting possibly malicious spam messages as well, provided care is taken to ensure sufficient such messages are used for training and testing.

Spam propagated with criminal intentions may have the presence of malicious URLs leading to deceptive and/or malicious sites. To deal with such cases, URLs in the text can be analysed separately and classified into various categories like- ‘Safe’, ‘Spam’, ‘Malware’, or ‘Phishing’, using ensemble learning techniques (Manyumwa, Chapita, Wu, & Ji, 2020). Therefore, spam detection, in general, may possibly be combated by using a two-pronged strategy. Firstly, by detecting spam in the textual content itself (which happens to be the aim of the present paper), and secondly, by classifying the URLs embedded in the text. Instant messaging services can leverage this strategy to safeguard their users. Public chat groups are often considered hotbeds for spammers and automatic spam detection can be a useful tool for administrators or moderators for helping them in the governance of such chat groups. Similarly- posts, comments, and replies on social media can automatically be flagged as suspicious using this two-pronged strategy. Social media users can be warned of suspicious posts and can be protected from scams run through social media. Consequently, accounts that post spam regularly can be identified by data analysis and suitable actions can be taken against such accounts.

From a broader perspective; as far as applications of text mining in information management are concerned, often there exists a requirement of building models which are quick at making decisions. Social media servers can possibly have an influx of thousands of posts and comments every second. Similarly, servers of e-commerce platforms could be dealing with thousands of incoming product reviews every second. The massive influx of content needs to be handled in a way such that there is a minimum delay in the response to the actions taken by the user. Though software architects and developers do their best to design and develop software that can handle such a large influx of content efficiently, deploying machine learning or deep learning models to the back-end software for the purpose of automation can affect the software's overall latency quite significantly. This is primarily because these models often perform a number of calculations before making their decisions. Deployed models could be serving an important purpose for the business and hence cannot be done away with. On the other hand, providing slow software services can annoy and drive away users. One of the ways to speed up computations is to buy and use additional computational resources, but this can sometimes be expensive. Data scientists are therefore required to explore and build models which are quicker at making decisions, preferably using minimal computational resources, with little to no compromise in the overall accuracy. One of the advantages associated with our proposed model is that it is quicker at making decisions when compared to some of the state-of-the-art deep learning models which strongly leads us to believe that the proposed model, subject to further research, could possibly prove to be a very useful alternative to other deep learning models in applications of text mining in information management where there are low latency requirements. Future research work, therefore, could include leveraging similar architectures of Multi-channel CNN for the detection of offensive content or spam posts/comments on social media, and for the detection of fake content/reviews.

7. Conclusion

In this paper, we have proposed a Multi-Channel CNN architecture with static and non-static embeddings for spam classification. This model was trained on a dataset containing self-collected messages and messages from UCI's spam dataset and we compared it against different machine learning models, variants of the proposed model, and state-of-the-art deep learning techniques mentioned in the literature. This experimentation and comparison provided the following interesting observations. First, using the proposed M3P model and its other variants not only gave better performance than the machine learning algorithms in terms of accuracy, recall, and F1 score but also alleviated the problem of high bias and overfitting. The model's accuracy is comparable to that of state-of-the-art deep learning techniques present in the existing literature while having significantly lesser training and testing time, especially when compared to BERT. It is also seen that it is computationally lighter when compared to BERT. Second, the overall increased performance of the variant models and reduction in overfitting was concomitant with the increase in the number of channels till the number reached three. Visualizing the activations of the convolution layer of different models indicated that the activation strength and hence the confidence in classification increased with an increase in the number of channels. Third, for the current task, model variants that used non-static random embeddings either alone or in combination with another static pre-trained embedding provided better representation and performance than using only static pre-trained embeddings. Fourth, the orientation of the layers in parallel or series gave more or less similar results indicating performance was independent of orientation in this particular case. Finally, to conclude, the high performance of our proposed model M3P can be attributed to the fact that it was composed of both multiple channels and random embeddings which provided a varied and better representation of the input leading to stronger reinforced activations and better results.

References

Almeida, T. A., Hidalgo, J. M. G., & Yamakami, A. (2011). Contributions to the study of SMS spam filtering: New collection and results. *Doceng '11*.

Almeida, T. A., Hidalgo, J. M. G., & Yamakami, A. (2012). SMS spam collection data set. *UCI Machine Learning Repository*. https://archive.ics.uci.edu/ml/datasets/sms+spam+collection.

Ansar, W., & Goswami, S. (2021). Combating the menace: A survey on characterization and detection of fake news from a data science perspective. *International Journal of Information Management Data Insights, 1*(2), 100052. 10.1016/j.jjimei.2021.100052.

Ansari, S., & Gupta, S. (2021). Customer perception of the deceptiveness of online product reviews: A speech act theory perspective. *International Journal of Information Management, 57*, 102286. 10.1016/j.ijinfomgt.2020.102286.

Aswani, R., Kar, A. K., & Ilavarasan, P. V. (2019). Experience: Managing misinformation in social media–insights for policymakers from twitter analytics. *Journal of Data and Information Quality, 12*(1). 10.1145/3341107.

Aswani, R., Kar, A. K., & Vigneswara Ilavarasan, P. (2018). Detection of spammers in twitter marketing: A hybrid approach using social media analytics and bio inspired computing. *Information Systems Frontiers, 20*(3), 515–530. 10.1007/s10796-017-9805-8.

Batra, J., Jain, R., Tikkiwal, V. A., & Chakraborty, A. (2021). A comprehensive study of spam detection in e-mails using bio-inspired optimization techniques. *International Journal of Information Management Data Insights, 1*(1), 100006. 10.1016/j.jjimei.2020.100006.

Chauhan, T., & Palivela, H. (2021). Optimization and improvement of fake news detection using deep learning approaches for societal benefit. *International Journal of Information Management Data Insights, 1*(2), 100051. 10.1016/j.jjimei.2021.100051.

Fattahi, J., & Mejri, M. (2021). SpaML: A bimodal ensemble learning spam detector based on NLP techniques. In *2021 IEEE 5th international conference on cryptography, security and privacy (CSP)* (pp. 107–112). IEEE.

Goldberg, Y. (2017). *Pre-trained word representation* p. 128). Morgan & Claypool.

Jacovi, A., Shalom, O. S., & Goldberg, Y. (2020). Understanding convolutional neural networks for text classification.

Jain, A. K., & Gupta, B. B. (2019). Feature based approach for detection of smishing messages in the mobile environment. *Journal of Information Technology Research (JITR), 12*(2), 17–35.

Jain, G., Sharma, M., & Agarwal, B. (2019). Optimizing semantic LSTM for spam detection. *International Journal of Information Technology, 11*(2), 239–250.

Kaddoura, S., Chandrasekaran, G., Popescu, D. E., & Duraisamy, J. H. (2022). A systematic literature review on spam content detection and classification. *PeerJ Computer Science, 8*, e830.

Khanday, A. M. U. D., Rabani, S. T., Khan, Q. R., & Malik, S. H. (2022). Detecting twitter hate speech in COVID-19 era using machine learning and ensemble learning techniques. *International Journal of Information Management Data Insights, 2*(2), 100120. 10.1016/j.jjimei.2022.100120.

Kim, Y. (2014). Convolutional neural networks for sentence classification.

Kumar, S., Kar, A. K., & Ilavarasan, P. V. (2021). Applications of text mining in services management: A systematic literature review. *International Journal of Information Management Data Insights, 1*(1), 100008. 10.1016/j.jjimei.2021.100008.

Lester, B., Pressel, D., Hemmeter, A., Choudhury, S. R., & Bangalore, S. (2020). Multiple word embeddings for increased diversity of representation. *CoRR*. https://arxiv.org/abs/2009.14394.

Luo, Q., Liu, B., Yan, J., & He, Z. (2011). Design and implement a rule-based spam filtering system using neural network. In *2011 International conference on computational and information sciences* (pp. 398–401). 10.1109/ICCIS.2011.125.

Manyumwa, T., Chapita, P. F., Wu, H., & Ji, S. (2020). Towards fighting cybercrime: Malicious URL attack type detection using multiclass classification. In *2020 IEEE international conference on big data (big data)* (pp. 1813–1822). 10.1109/BigData50022.2020.9378029.

Mewada, A., & Dewang, R. K. (2021). Research on false review detection methods: A state-of-the-art review. *Journal of King Saud University - Computer and Information Sciences*. 10.1016/j.jksuci.2021.07.021.

Michail, D., Kanakaris, N., & Varlamis, I. (2022). Detection of fake news campaigns using graph convolutional networks. *International Journal of Information Management Data Insights, 2*(2), 100104. 10.1016/j.jjimei.2022.100104.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2021). Deep learning based text classification: A comprehensive review.

Mishra, S., & Soni, D. (2020). Smishing detector: A security model to detect smishing through SMS content analysis and URL behavior analysis. *Future Generation Computer Systems, 108*, 803–815.

Naredla, N. R., & Adedoyin, F. F. (2022). Detection of hyperpartisan news articles using natural language processing technique. *International Journal of Information Management Data Insights, 2*(1), 100064. 10.1016/j.jjimei.2022.100064.

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), 100007. 10.1016/j.jjimei.2020.100007.

Parwez, M. A., Abulaish, M., & Jahiruddin (2019). Multi-label classification of microblogging texts using convolution neural network. *IEEE Access, 7*, 68678–68691. 10.1109/ACCESS.2019.2919494.

Popovac, M., Karanovic, M., Sladojevic, S., Arsenovic, M., & Anderla, A. (2018). Convolutional neural network based SMS spam detection. In *2018 26th Telecommunications forum (TELFOR)* (pp. 1–4). IEEE.

Roy, P. K., Singh, J. P., & Banerjee, S. (2020). Deep learning to filter SMS spam. *Future Generation Computer Systems, 102*, 524–533.

Saidani, N., Adi, K., & Allili, M. S. (2020). A semantic-based classification approach for an enhanced spam detection. *Computers & Security, 94*, 101716.

Shrivastava, J. N., & Bindu, M. H. (2014). E-mail spam filtering using adaptive genetic algorithm. *International Journal of Intelligent Systems and Applications, 6*(2), 54–60.

Sonowal, G., & Kuppusamy, K. (2018). SmiDCA: An anti-smishing model with machine learning approach. *The Computer Journal, 61*(8), 1143–1157.

Wadud, M. A. H., Kabir, M. M., Mridha, M., Ali, M. A., Hamid, M. A., & Monowar, M. M. (2022). How can we manage offensive text in social media - a text classification approach using LSTM-boost. *International Journal of Information Management Data Insights, 2*(2), 100095. 10.1016/j.jjimei.2022.100095.

Xue, H., Wang, Q., Luo, B., Seo, H., & Li, F. (2019). Content-aware trust propagation toward online review spam detection. *Journal of Data and Information Quality, 11*(3). 10.1145/3305258.

Yaseen, Q., et al., (2021). Spam email detection using deep learning techniques. *Procedia Computer Science, 184*, 853–858.

Yin, W., & Schtze, H. (2016). Multichannel variable-size convolution for sentence classification. 10.48550/ARXIV.1603.04513.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems, 28*.

Zhuang, X., Zhu, Y., Peng, Q., & Khurshid, F. (2021). Using deep belief network to demote web spam. *Future Generation Computer Systems, 118*, 94–106.

Gopalkrishna Ratilal Waja is currently a student at Northeastern University in Boston, where he is pursuing his Master's degree in Artificial Intelligence. He received his Bachelor's degree in Information Technology from K. J. Somaiya College of Engineering, Mumbai, in 2022 and graduated as the department topper. His areas of experience and interests include Artificial Intelligence, Machine Learning, Computer Vision and Cybersecurity and he has worked on several projects in these domains.

Gaurang Patil graduated from K. J. Somaiya College of Engineering, Mumbai, India, in 2022 and holds a B.Tech. degree in Information Technology. His research interests mainly include applications of machine learning and deep learning in various domains. His B.Tech. thesis was focused on applications of machine learning and deep learning techniques in issues pertaining to cybersecurity.

Charmee Jayesh Mehta is currently pursuing her Master's degree in Computer Engineering at New York University. She has completed her Bachelor's degree in Information Technology from K. J. Somaiya College of Engineering, Mumbai. Her research domains include Deep Learning, Machine Learning and Blockchain. She has worked on numerous projects in these domains.

Sonali Patil is a Professor at K. J. Somaiya College of Engineering, Somaiya Vidyavihar University, Mumbai (India). Presently, she holds the position of Associate Dean, Academics. She received her PhD, M.E. and B.E. degrees in Electronics Engineering from Shivaji University, Kolhapur (India). She is a recognized PG teacher and PhD supervisor at the University of Mumbai and SVU. Her doctoral research was in the area of Medical Image Processing and Analysis. She has published in peer-reviewed Journals, Book chapters and conferences in the areas of Information Security and Image Processing. She has supervised M.Tech. and B.Tech. projects in these areas.