Expert Systems With Applications 184 (2021) 115394

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Bengali text document categorization based on very deep convolution neural network

Md. Rajib Hossain [a,1], Mohammed Moshiul Hoque [a,*,2], Nazmul Siddique [b,3], Iqbal H. Sarker [a,4]

[a] Department of Computer Science & Engineering, Chittagong University of Engineering & Technology, Chittagong-4349, Bangladesh
[b] School of Computing, Engineering and Intelligent Systems, Ulster University, UK

ARTICLE INFO

Keywords:
Intelligent systems
Natural language processing
Low resource language
Semantic feature extraction
Document categorization
Deep convolution network

ABSTRACT

In recent years, the amount of digital text contents or documents in the Bengali language has increased enormously on online platforms due to the effortless access of the Internet via electronic gadgets. As a result, an enormous amount of unstructured data is created that demands much time and effort to organize, search or manipulate. To manage such a massive number of documents effectively, an *intelligent* text document classification system is proposed in this paper. *Intelligent* classification of text document in a resource-constrained language (like Bengali) is challenging due to unavailability of linguistic resources, intelligent NLP tools, and larger text corpora. Moreover, Bengali texts are available in two morphological variants (i.e., Sadhu-bhasha and Cholito-bhasha) making the classification task more complicated. The proposed *intelligent* text classification model comprises GloVe embedding and *Very Deep Convolution Neural Network (VDCNN)* classifier. Due to the unavailability of standard corpus, this work develops a large *Embedding Corpus (EC)* containing 969,000 unlabelled texts and *Bengali Text Classification Corpus (BDTC)* containing 156,207 labelled documents arranged into 13 categories. Moreover, this work proposes the *Embedding Parameters Identification (EPI)* Algorithm, which selects the best embedding parameters for low-resource languages (including Bengali). Evaluation of 165 embedding models with intrinsic evaluators (semantic & syntactic similarity measures) shows that the GloVe model is more suitable (regarding Spearman & Pearson correlation) than other embeddings (Word2Vec, FastText, m-BERT) in Bengali text. Experimental results on the test dataset confirm that the proposed GloVe + VDCNN model outperformed (achieving the highest 96.96% accuracy) the other classification models and existing methods to perform the Bengali text classification task.

1. Introduction

Automatic document classification is an emerging research area in the field of Natural Language Processing (NLP), where each document needs to be assigned to a predefined class or category. In recent years, categorization of Bengali text documents is treated as a significant research issue in the domain of Bengali Language Processing (BLP). Bengali is the 7^{th} most widely spoken language in the world (Hossain, Hoque, & Sarker, 2021). Two hundred forty-five million people in Bangladesh and some parts of India speak in this language (Hossain & Hoque, 2021). In recent years, the digital Bengali text contents have been growing readily on the Internet, news portals, blogs, websites, and so on due to the effortless usage of electronic gadgets. These content is creating an enormous amount of unstructured data. Therefore, it is a challenging task to organize, search or manipulate such a massive amount of unstructured data by human experts manually. This manual process consumes an immense amount of time incurring the cost of money. An *intelligent* document categorization system can handle a massive amount of Bengali text data in which documents can be sorted, manipulated and organized expeditiously and competently.

Categorizing Bengali text documents is one of the most challenging tasks due to the constitution of the language itself having well-off

* Corresponding author.
E-mail addresses: moshiul_240@cuet.ac.bd (M.M. Hoque), nh.siddique@ulster.ac.uk (N. Siddique), iqbal@cuet.ac.bd (I.H. Sarker).
[1] ORCID: 0000-0002-7941-9124
[2] ORCID: 0000-0001-8806-708X
[3] ORCID: 0000-0002-0642-2357
[4] ORCID: 0000-0003-1740-5517

https://doi.org/10.1016/j.eswa.2021.115394
Received 16 December 2020; Received in revised form 29 April 2021; Accepted 8 June 2021
Available online 2 July 2021

dialects and complex morphological structure. It also articulates the huge variations of subject-verb and person-tense-aspect agreements. Unavailability of standard Bengali text corpus with rich morphological variants (i.e., Sadhu-bhasha and Cholito-bhasha) and scarcity of resources are antecedents making the Bengali text categorization task more complicated. Moreover, there are no efficient tools available that have been developed till today for Bengali language processing (BLP). Therefore, there is an insistence of developing intelligent tools for BLP so that professionals, as well as the common people, can use these tools to their needs. However, developing an *intelligent* system that can categorize Bengali text documents in distinct categories is also an important research issue leading to the utmost use in real-world applications. An intelligent text categorization system can be used in information retrieval, text mining, and text analytic (Hashmi & Bansal, 2019; Akhter et al., 2020). The framework and techniques developed under the proposed research can be used as the baseline for developing many intelligent systems such as news portal, medical services or hospital management, library/archival management, vertical search engine, and toxicity or hostility detector, and semantic analysis tools where Bengali text processing is major concerned. The security agencies can use the proposed framework to discover aggressive or hostile web content or rumours in social media. Moreover, the proposed framework can be used by the hospital management to categorize patient's reports according to their diagnosis, an e-library to arrange books according to subject categories and news agencies to classify the news based on text contents. The considerable barriers of performing research on document categorization in BLP are due to the shortage of linguistic resources in digital form and inadequate corpora (Phani, Lahiri, & Biswas, 2017). Significant research has been carried out on Bengali document categorization (Hossain & Hoque, 2021; Kabir, Siddique, Kotwal, & Huda, 2015; Ahmad & Amin, 2016; Hossain & Hoque, 2019; Dhar et al., 2020). Most of these research used traditional machine learning techniques which do not provide reliable results on large datasets and can not be applied a higher number of classes.

The key difficulty of machine learning techniques is to select useful features from high dimensional feature spaces (Deng, Li, Weng, & Zhang, 2019). Moreover, a common Algorithm for selecting appropriate features is not available that can be applied to all kinds of classification task (Zia, Akhter, & Abbas, 2015). Very deep Convolutional Neural Network (CNN) technique has been used extensively to solve complex problems in many domains, including text document categorization and NLP. Surprisingly, the potential of the very deep learning techniques has not been investigated for Bengali document classification (Dhar et al., 2020). In comparison with other machine learning-based techniques, very deep learning techniques are more competent to discover complex features inherently, when the feature dimension is high. In addition to that, the very deep learning is faster than the other machine learning techniques due to its usage of Graphics Processing Units (GPUs) (Khan & Adnan, 2018; Khan, Jan, & Farman, 2019).

Bengali is a low-resource language, and no well-performed feature extractor and classification frameworks are available in Bengali. Moreover, there are huge structural variations between Bengali and English languages (or other high-resource languages). As a result, the embedding/classification models developed for one language cannot be directly applicable to another language due to their linguistic divergences. The hyperparameters of embedding and classification models must be tuned afresh based on the language itself before performing the classification task. The hyperparameters of embedding and classification models must be newly tuned based on the language itself before performing the classification task. Moreover, Bengali texts usually available in two morphological variants, such as Sadhu-bhasha and Cholito-bhasha. Thus, an intelligent system should also consider these morphological variations while performing text classification task in Bengali.

In order to achieve a reasonable accuracy, selecting the optimum hyperparameter values of the embedding and classification models is critical for the associated corpus. This is where the challenges lie in this research requiring us to carry out the embedding parameters identification (EPI) and optimization of the parameters. An algorithm has been developed (Algorithm 3), which generates a set of optimized embedding parameters from the embedding corpus. This algorithm is applicable to optimization of embedding parameters (such as embedding dimension, contextual window and the minimum number of the frequent word) in any low-resource language. Devising an objective function is the key to any embedding model hyperparameter optimization, In the EPI algorithm, we introduce a new intrinsic evaluation function that evaluates the semantic and syntactic accuracy of the embedding model. We evaluated 165 embedding models based on these optimized parameters using the intrinsic evaluation function. None of the work hitherto performed embedding model evaluation and text classification in Bengali concerning morphological variations to the best of our knowledge.

The major contributions of this research are:

- Develop two larger corpora: word embedding corpus (*EC*) and text classification corpus (*BDTC*) to perform Bengali text classification task considering two forms of Bengali text (i.e., Sadhu-bhasha and Cholito-bhasha). The *EC* containing 969,000 text documents (with 19,517,390 unique words) whereas *BDTCC* contained 156,207 text documents into 13 categories (Section 4).
- Develop an embedding parameters identification (EPI) Algorithm, which selects the best embedding parameters for low resource languages (including Bengali) concerning Word2Vec, GloVe and FastText embedding techniques. (Algorithms 3 & 4). This research also introduces a new objective function (e.g., intrinsic evaluation) in EPI Algorithm that aided to optimize the embedding parameters.
- Evaluate 165 embedding models using intrinsic evaluators (i.e., syntactic similarity and semantic similarity measures) to select the best embedding model in Bengali (Section 8.1). Among 165, select 12 best performing models to perform the downstream task (i.e., text classification). To the best of our knowledge, none of the previous work performed embedding model evaluation intrinsically in Bengali for text classification.
- Investigate the performance of 109 classification models developed by combining 12 best-performed embedding models and variants of four classification techniques (ML, CNN, sequential and BERT). For clarity, we presented 26 best classifiers performance (Section 8.3 & Table 8).
- Investigate the performance of the proposed GloVe + VDCNN based model for the Bengali text classification task by comparing its performance with the existing techniques (Sections 8.3.1 and 8.5). Perform a detailed error analysis of the proposed text classifier (Section 8.7).

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 introduces the problem statement, whereas Section 4 describes the development of the Bengali text document classification corpus. A detailed explanation of the various methodology used in work is described in Section 5. Section 6provides a detailed development of the proposed VDCNN-based Bengali text categorization with its major constituents. Section 7 highlights the experiment with a comprehensive explanation of hyperparameters optimization and evaluation measures. The performance analysis of the developed models concerning embedding model evaluation and text classification are reported in Section 8. Section 9 presents the analysis of the results, including the future implications of the work. Finally, the paper concludes with a comprehensive summary in Section 10.

2. Related work

Text document categorization is a long-studied research issue for well-resourced languages like Arabic, English and most European languages. Conneau et al. (Conneau, Schwenk, Barrault, & Lecun, 2017)

developed a very deep convolutional networks (VDCN) based text categorization system, which achieved 98.71% accuracy on the DBPedia ontology classification data set. Their system used 560 K data for training and 70*K* for testing in 14 categories. Liebeskind, Kotlerman, and Dagan (2015) developed an unsupervised multi-tag text categorization system for entertainment industries. This system used the unsupervised technique to categorize the movie titles on Internet movie databases in 100 categories but failed to achieve higher accuracy due to low feature extraction and classification techniques.

Johnson and Zhang (2017) have designed a Deep Pyramid Convolutional Neural Networks (DPCNNs) system for English text categorization. They have achieved 97.36% accuracy on Yelp dataset with 560*K* training documents, 38*K* testing documents and 2 document classes. The accuracy is decreased with increasing classes in DPCNNs system and hence not suitable for a higher number of document classes. Character-level feature extraction and convolution network-based classifier model have evolved since the first application (Zhang, Zhang, & LeCun, 2015). This work achieved 98.69% test accuracy on DBPedia ontology classification dataset with 14 distinct categories. However, memory exhaustion is the major shortcoming of the system despite the use of two costly Tesla *K*40 GPUs. (Lee & Dernoncourt, 2016) introduced a short text classification system based on recurrent and convolution neural networks. In this method classifier model is trained with 78 K, tested with 15 K and validated with 16 K text documents. This work used the Meeting Recorder Dialog Act (MRDA) Corpus (Shriberg, Dhillon, Bhagat, Ang, & Carvey, 2004) with 5 categories and achieved an accuracy of 84.60%. The recurrent networks demanded huge time during training and testing.

Xu, Feng, Huang, and Zhao (2015) proposed a semantic relation classification technique for document classification in which a feature extractor is developed by negative sampling with global vector representation, and the classifier model has emerged by CNNs. This system is trained with 8,000 text data and tested with 2,717 labelled data. This technique achieved 85.60% accuracy for 10 semantic relations. Tang, Qin, and Liu (2015) developed a sentiment analysis system in which feature extraction was done by using Word2Vec Algorithm (Mikolov, Chen, Corrado, & Dean, 2013). The sentiment classification is developed by using Gated Recurrent Neural Network (GRNN). This system achieved 67.10% accuracy on Yelp2014 datasets which consist of 1,125,457 training texts in 5 categories. Saad and Yang (2019) developed a text-based binary sentiment (i.e., positive and negative) classification system using a decision tree Algorithm which achieved 91.81% accuracy. This system is trained using 7,000 Twitter texts (positive/negative sentiments) and tested using 3,000 Twitter labelled texts. In these sentiment classification systems, the overall classification accuracy decreased with the increasing sentiment/document classes.

Kowsari et al. (2017) developed a Hierarchical Deep Learning (HDL) based document which extracted semantic features using Global Vectors for Word Representation (GloVe) (Pennington, Socher, & Manning, 2014) Algorithm. The classifier model was trained using 44,085 text documents and tested with 20,603 text documents for 7 categories. Overall, 90.93% accuracy was achieved. However, a combined approach of HDL and long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) with generalized regression neural network (GRNN) caused more training and testing time. Fuzzy clustering-based document categorization system is proposed by Mei, Wang, Chen, and Miao (2017). This technique did not determine the number of clusters accurately, which is required for unknown data sets. Xiao, Liu, Yin, and Hao (2017) developed an unsupervised clustering based system for adaptive document categorization. In this system, approximately 20,000 newsgroups of 20 different news from 20 Newsgroup data sets are used and accuracy of 74.50% was achieved.

Nikolentzos, Meladianos, Rousseau, Stavrakas, and Vazirgiannis (2017) developed a document classification system where Word2Vec Algorithm is used for feature extraction, and SVM is used for classification (Hearst, 1998). Their system was trained using TREC news data set (Li & Roth, 2002) consisting of 5,452 train and 500 test documents for six categories. This system achieved 98.20% accuracy. He, Wang, Liu, Feng, and Wu (2019) proposed document classification system where a document is represented by an embedding model. Recurrent Attention Learning (RAL) with CNNs Algorithm is used for a classification task. This system is trained using 33,388 text documents for 11 categories and achieved about 94.18% accuracy. Bahassine, Madani, and Mohamed (2017) developed a stemmer based feature selection and decision tree based classifier for Arabic text classification. This classifier was trained using 3,382 text documents for 6 categories, tested using 1,688 text documents and achieved 79.00% accuracy. The decision tree based classifier is unable to be trained with high dimensional data set. Alhawarat and Aseeri (2020) designed an Arabic text categorization system which used TF-IDF for feature extraction and CNNs for the classifier tasks. Their system used 15 Arabic open-access news data set and achieved 98.89% accuracy on Abuaiadah (V2) data set (Abuaiadah, Sana, & Abusalah, 2014) for nine categories. The TF-IDF based features extractor is not capable of extracting semantic features leading to higher misclassification rate.

Bengali is a low resource language due to unavailability of linguistic resources and inadequate corpora (Rahman & Dey, 2018; Phani et al., 2017). The research on automatic text document classification is in the preliminary stage until now in the realm of Bengali languages. Hossain and Hoque (2021) proposed a Bengali document categorization system in which the DCRNNs technique is used to extract features. The DCRNNs classifier is trained using 120,000 text documents and tested by using 36,207 text documents for twelve categories. Kabir et al. (2015) proposed a TF-IDF feature extraction and gradient descent classifier for Bengali document classification. They trained the classifier using 5,448 documents and tested using 3,679 news text documents for 9 categories. This system achieved 93.78% categorization accuracy. Another work also used stochastic gradient descent (SGD) based classifier for Bengali text categorization (Hossain & Hoque, 2018). This system obtained 93.33% test accuracy for 9 distinct categories with 10,000 training and 4,651 testing document samples. Both SGC based techniques worked with a limited number of texts categories and test samples. Moreover, the SGD based classifiers do not work well for large feature dimensions (Ruder, 2016).

Recent work used, Word2Vec based, feature extraction technique, a combination of K-Means clustering with SVM, for classification of Bengali texts (Ahmad & Amin, 2016). This system used 19,705 online news text documents for 7 text categories. The feature extraction technique can extract semantic features with a large number of words, but the K-Means Algorithm has failed to handle the outlier data. Hossain and Hoque (2019) introduced a Bengali document categorization system based on Word2Vec and Deep CNNs. This system used 86,199 training news text documents for 12 types of document categories. The system obtained, on average 94.96% document classification accuracy on 10,707 test samples. However, this system is unable to carry out the sentence level semantic/syntactic meanings. Dhar et al. (2020) proposed a graph-based feature extraction technique along with LSTM RNN based classifier for Bengali text categorization. This system considered only 9 document classes and achieved 99.21% classification accuracy on 14,371 text documents. In this system, more time is required to extracts features and failed to extracts semantic/syntactic features.

Most previous studies used SVM (Kabir et al., 2015), SGD (Hossain & Hoque, 2018), KNN (Mucherino et al., 2009), and Deep CNN (Hossain & Hoque, 2019) techniques to perform the text classification task in Bengali. These studies dealt with a limited number of text categories and classification corpora. Moreover, none of the work performed any embedding model evaluation so far in Bengali. Thus, it is unknown how the embedding model's parameters affected the Bengali text classification performance. This research introduces an EPI Algorithm for selecting the embedding model's parameters to overcome the shortcomings of previous statistical feature extraction-based methods concerning resource-constrained languages. The proposed GloVe + VDCNN

based Bengali text classification technique also overcome the weaknesses of existing text classification methods by improving the accuracy with a larger corpus and a more significant number of classes.

3. Problem statement

The proposed framework's main purpose is to develop a text document categorization (*TDC*) system that can categorize a text document into an expected class or category. For this purpose, it contains a classification corpus $\mathscr{CC}$ and $\mathscr{CC} \in \{TD^s \cup TD^u\}$, TD^s denotes the seen corpus whereas $TD^s \in \{(td_i^s, cl_j^s)\}_{i,j=0}^{n,CL}$. The (td_i^s, cl_j^s) represents i^{th} sample seen text document and label. As well as TD^u indicates the unseen corpus and $TD^u \in (\{td_i^u, cl_j^u)\}_{i,j=0}^{m,CL}$. The (td_i^u, cl_j^u) indicates i^{th} example unseen text document and label. Where, n, m represent the total number of training and testing document in $\mathscr{CC}$ and CL denotes the total number of class in $\mathscr{CC}$. Now, the goal is to train a high-quality document classification model ($\mathscr{Z}$) using the seen corpus TD^s and assigning category label $cl_j^u \in CL$ to each text document $td_i^u \in TD^u$. Before train the Bengali text classification model, td_i^s convert to a $(V_d \times F)$ feature representation using an embedding model. V_d indicated the number of words and F denotes the feature dimension. The embedding techniques take the embedding corpus EC and $EC = \{et_i\}_{i=0}^{D}$, where D denotes the total number of embedding corpus and output is an embedding model. For example, the i^{th} unseen text document td_i^u is converted to $(V_d \times F)$ features representation and corresponding classification label is predicted using Eq. (1).

$$cl_i^u = max_{c=1}^{c=N}(\mathscr{Z}_c[V_d \times F]^i) \tag{1}$$

where, $\{\mathscr{Z}_1, \mathscr{Z}_2, \mathscr{Z}_3, \ldots, \mathscr{Z}_N\}$ denotes the classification models sets and cl_i^u represents the i^{th} unseen predicted label and N indicates the total number of classification models in the proposed system. The *max* is a symbolic function that takes the input as an unknown features representation $(V_d \times F)$ and maximum expected value calculated using Eqs. (9)–(10). The *max* function was operated through the classifier models and obtained a maximum accuracy with a category.

A deep learning-based text classification system consists of two key components: embedding and classification. The embedding technique generates an embedding model using unlabelled texts, whereas the classification technique produces a classification model based on labelled texts. Many state-of-the-art embedding model generation techniques (such as GloVe, FastText, Word2Vec, BERT) and embedding models have available for English Language (Pennington et al., 2014; Bojanowski, Grave, Joulin, & Mikolov, 2017; Mikolov et al., 2013; Devlin, Chang, Lee, & Toutanova, 2019). These models evolved based on optimized hyperparameters that are worked for the English language only. However, the model developed for one language can not be helpful for another language, i.e., English language's models are not working efficiently for low resource language like Bengali, Italian and others (Hossain & Hoque, 2020; Gambino & Pirrone, 2019). As a result, available embedding techniques in other languages can not be appropriate for the Bengali texts. In recent time, the multilingual embedding model (Grave, Bojanowski, Gupta, Joulin, & Mikolov, 2018; Devlin et al., 2019), has investigated for the resource-constrained language (e.g., Bengali), but the performance of their downstream tasks are not significant due to the shortage of vocabularies, lack of handling morphological variations (e.g., Sadhu-bhasha and Cholito-bhasha) and unavailability of optimized embedding hyperparameters. In addition to that, most classification techniques devised and tested for high-resource languages (i.e., English) and the model's hyperparameters profoundly optimized for their own benchmarks datasets. The classifier model and corresponding hyperparameters can not be replicated to develop a model in a resource-constrained language to perform downstream tasks due to inferior performance (Grießhaber, Vu, & Maucher, 2020). Bengali is considered one of the most resource-scarce languages globally due to its lack of large corpora or manually crafted linguistic resources adequate for developing NLP applications (Hossain & Hoque, 2019). Now, to overcome the previous problems, this research introduces a VDCNN classifier model based on developed corpus and developing a new technique to identifying optimizing parameters for embedding model generation techniques (Algorithms 3 & 4).

4. Bengali text corpora

The big challenge to research in Bengali language processing is the unavailability of the standard corpus. Thus, to develop a Bengali text corpus. This work adopted the same approach as described by Dash and Ramamoorthy (2019) to develop the corpus, which consists of five main phases: data crawling, preprocessing, crowd-sourcing or manual labelling, verification and Kappa measure. The details of the corpus development process illustrated in the following subsections.

4.1. Data crawling

There is no well known linguistic resource repository in Bengali. Thus, data is accumulated from different web source using a Python crawler.[5]. Initially, we have selected an authentic Bengali data source such as online newspapers, blogs and e-book has been selected. The crawler crawled text data from the selected sources regarding dates, subjects, popularity, geographical areas and trends. Total of $1,020,000$ Bengali text files have been collected, and each of the text is encoded as UTF-8 and stored in $*.txt$ format. The crawling system extracted the data about ten years duration (January 2010 to December 2019).

4.2. Data preprocessing

The corpus contained noisy data due to the freestyle of writing on the web. Thus, a program is developed to remove noisy or ill-formatted data from the corpus. Thus, each collected text file requires the following preprocessing:

- The foreign alphabets (e.g., non-Bengali) and digits are replaced with a NULL value.
- All the regular expression and symbols are eliminated by a single white space.
- A whitespace substitutes a HTML tags, hashtags, URLs and punctuation.
- Multiple new lines are eliminated by a single newline.
- All the duplicate texts are removed.

If a text contains less than three words and its size is greater than 100 KB, then this text is eliminated from the corpus. After preprocessing, there are $969,000$ texts in the clean corpus and the preprocessing task is reduced to $51,000$ texts (e.g., 5.00%). Therefore, $969,000$ texts have been considered as embedding corpus (*EC*). Table 1 illustrates the key characteristics of the developed Bengali text corpus which consists of $200,081,093$ words in total and $19,517,390$ unique words.

4.3. Crowd-sourcing/manual labelling

A total of $969,000$ preprocess texts are available for human annotation. However, due to the shortage of annotators, only 35.00%, e.g., $339,150$ texts are randomly selected for the crowd-sourcing purpose. The crowd-sourcing task assigns the class label in the hand-crafted text data and is based on maximum voting. Seven Computer Science and Engineering graduate students worked to annotate the data by hand. All of them are Bengali native speakers, and the data are stored in *.txt* file

[5] https://github.com/mrhossain/BD-Scraper-Cleaner

Table 1
Characteristics of the embedding corpus *EC*.

Corpus Attributes	Attributes Value
Number of text documents	969,000
Number of sentence	1,744,200
Maximum number of sentence	170
Minimum number of sentence	1
Average number of sentence per document	7
Number of words	200,081,093
Number of unique words	19,517,390
Estimated required memory	29.7 GB

format into one of the 13 predefined categories. These categories are sports, lifestyle, art, science & technology, education, environment, entertainment, economics, opinion, health, politics, crime, and accident. If a text is not semantically similar to the predefined categories, then it is discarded from the corpus. A total of 175,000 hand-crafted labelled texts acquired in the crowd-sourcing process in 13 categories. There are 164,150 texts (e.g., 48.40%) that are discarded from the crowd-sourcing step due to a shortage of predefined semantic tag similarities. The overall data labelling procedure is summarized in Algorithms 1 and 2. These Algorithms take the unlabelled inputs corpus and return a labelled corpus.

Algorithmic 1 Automatic text label determination.

```
Input : vector < string > unlabelText
Output : vector < string > BDTC
total←unlabelText.size()
annotators←7
vector < int > crowdSourcingLabel
vector < string > crowdSourcingText
procedure CrowdSourcingannotators, unlabelText
    for i←1→total do
        tempLabel←Voting(annotators, unlabelText[i]))
        if tempLabel > 0 then
            crowdSourcingLabel.push_back(tempLabel)
            crowdSourcingText.push_back(unlabelText[i])
        end if
    end for
end procedure
experts←10
procedure Verificationexperts, crowdSourcingText
    for i←1→crowdSourcingText.size()
        tempLabel←Voting(annotators, crowdSourcingText[i])
        if temoLabel = = crowdSourcingLabel[i]
            BDTC.push_back(crowdSourcingText[i])
        end if
    end for
    return BDTC
end procedure
```

Algorithmic 2 Text category selection based on annotators or experts votes

```
Input : Number of annotators and text document.
Output : Label of the given text.
map < int, int > voteCast
voteCast.clear()
totatlClass←13
procedure Votingannotators, text
    TempLabel←0
    for i←1→annotators
        TempLabel← labeling(text)
        if TempLabel > 0
            VoteCast[TempLabel]←VoteCast[TempLabel] + 1
        end if
    end for
    finalLabel←0
    for i←1→totalClass
        finalLabel←max(finalLabel, voteCast[i])
    end for
    return (finalLabel)
end procedure
```

4.4. Verification

Ten experts were recruited to verify the manual data label. Among them, 3 are postgraduate, and 7 are graduate students of Bengali literature. Each of the text documents is labelled by the ten experts. The expert's caste the individual's vote (e.g. category name) for each text document. A document class or category is grouped into one of the 13 categories, which achieved the highest votes. The category labelled by the experts and the crowd-sourcing process both are cross-checked. If the experts and crowd-sourced process category assignment are matched for a document, its category is confirmed and included in the Bengali text classification corpus (*BDTC*). Otherwise, it is rejected, and no class is assigned. This process is explained in Algorithms 1 and 2. A total of 156,207 documents are matched among 175,000, and the corpus includes all matched category documents. A total of 18,793 documents (i.e., 10.74%) is ignored due to the mismatch, which lacks the quality and extractions. Fig. 1 shows the category-wise data distribution of the corpus.

Developed corpus was partitioned into three sets where 106,899 documents are used in training, 27,831 documents in validation and 21,477 documents in testing sets respectively. Table 2 shows the categorical summary. The highest number of text set belongs to politics, whereas the lowest number of documents belongs to the environment category. A random selection of 68.43% documents was allocated for the training set, 31.57% for the validation set and 13.75% for the test set, respectively.

4.5. Annotation quality

The difference in experiences, the annotation task itself, and focus on the annotators contribute to a disagreement between the annotators (Rebecca, 2006). Thus, it is required to find how much the annotators agree in assigning text classes by using Cohen's kappa (Cohen, 1960). Cohen's kappa measures the agreement among annotators and this determines how well one annotator agrees with another annotator. Thus, to investigate the standard inter-annotator agreement, a pairwise kappa coefficient is calculated using Eq. (2).

$$K = \frac{P_o - P_e}{1 - P_e} \tag{2}$$

where P_0 denotes relative observed agreement and P_e determines the hypothetical probability of chance agreement. The developed corpus (*BDTC*) achieved a Kappa score (K) is 67.23%, which indicates a

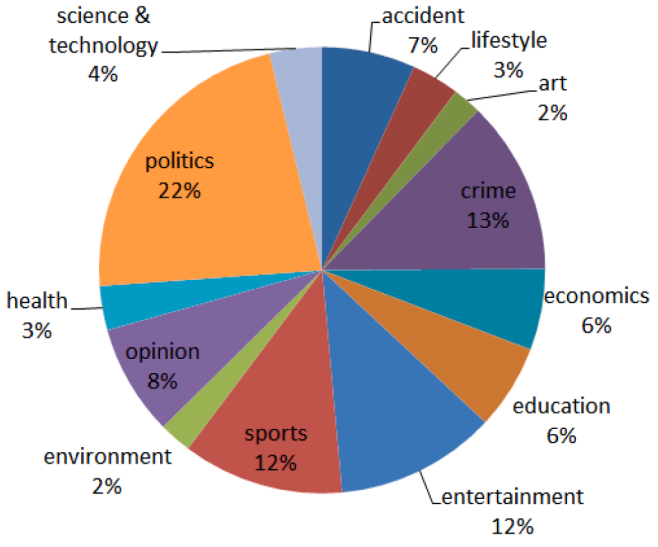

Fig. 1. Class-wise distribution of corpus (BDTC).

Table 2
Categorical distribution of corpus (*BDTC*).

Document category	No. of Training documents	No. of Validation documents	No. of Testing documents
Sports	13,499	2,761	1,958
Lifestyle	2,615	750	2,017
Art	2,320	720	236
Science & technology	3,554	1,200	1,183
Education	6,590	2,300	970
Environment	2,265	900	534
Entertainment	11,194	3,520	3,439
Economics	6,100	2,150	953
Health	3,500	650	859
Opinion	8,754	1,780	1,985
Politics	25,490	5,600	3,713
Crime	13,577	3,200	2,814
Accident	7,441	2,300	816
Total	106,899	27,831	21,477

reasonable agreement among annotators.

5. Methodology

The primary objective of this research is to classify the Bengali text documents into 13 predefined categories. To accomplish this objective, a computational model is developed by investigating various machine learning, deep learning and transformer-based methods. This section briefly explains the methods and techniques employed to address the objective. Fig. 2 illustrates the overview of the proposed system. Parameters selection and architectures of different approaches will be discussed in the subsequent subsections.

5.1. Text preprocessing and feature extraction

Raw input texts may contain noises such as punctuation, digits, unwanted symbols and characters written in other languages. All of these were removed during the preprocessing step (Section 4.2). Three kinds of feature extraction techniques are used in this study for extracting text relevant features. For example, traditional feature extraction (e.g., TF-IDF), contextual feature extraction (e.g., m-BERT) and local contextual feature extraction (e.g., GloVe, FastText & Word2Vec). The local contextual feature extraction can be further divided into frequency-based (e.g., GloVe) and prediction-based methods (e.g., FastText & Word2Vec). The frequency-based method extracts features based on local context, global word frequency and word-word co-occurrences over the whole corpus. In contrast, the prediction based method extracts text features based on the neural embedding and local context whether words occur in allied contexts.

The term-frequency with inverse document frequency (TF-IDF) is the most popular as the traditional feature extractor technique used in classification tasks (Kumari, Jain, & Bhatia, 2016). This feature extractor focus on the most frequent word and n-gram in the whole document. In this work, a maximum of $2,048$ words considered for the feature extraction. The contextual feature extracts the word feature based on word context and semantics (Dang & Palmer, 2002). The pre-trained Multilingual-BERT (m-BERT) (Devlin et al., 2019) is used with 12 hidden layers, 12 multi attention head, 512 position embedding, GELU activation and 768 feature dimension. Three local context or local window-based feature extraction techniques are considered in this study, such as Word2Vec (Mikolov et al., 2013), FastText (Bojanowski et al., 2017) and GloVe (Pennington et al., 2014). The details of the local contextual feature extraction techniques are described in Section 6.1.

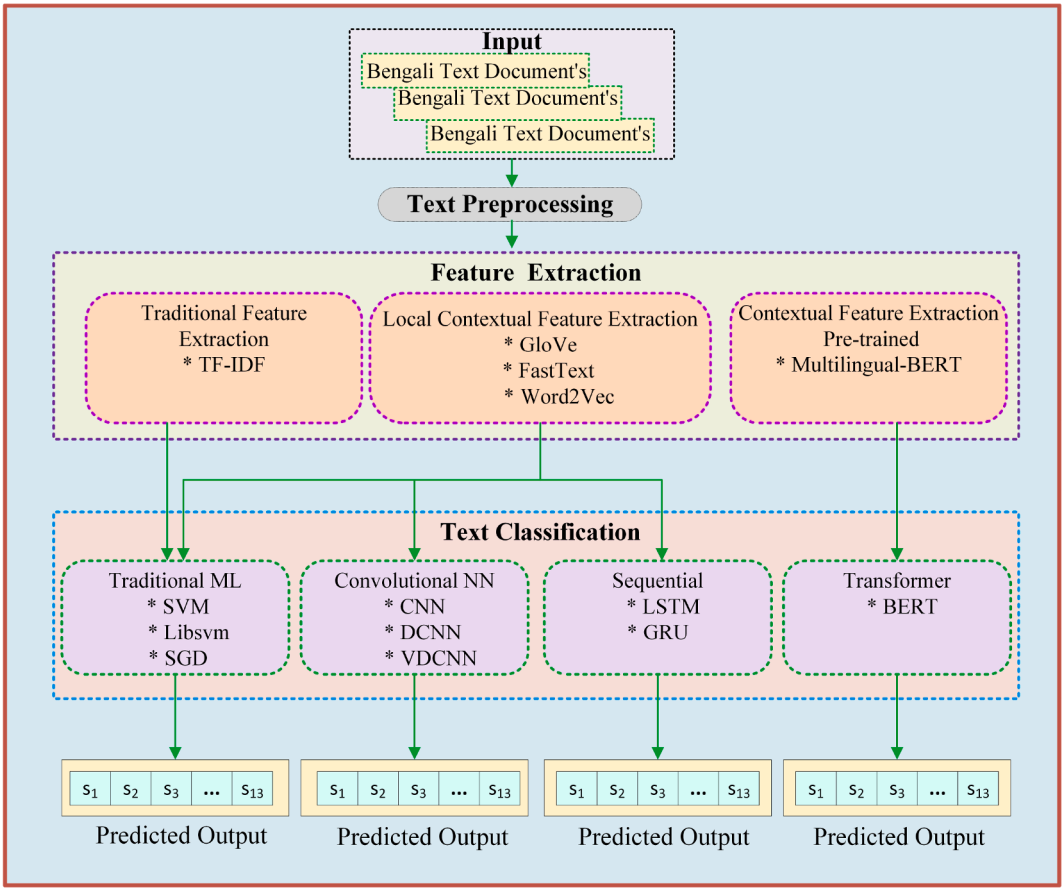

Fig. 2. Abstract view of the document categorization system.

5.2. Text classification

To investigate the performance of text document categorization task in Bengali, four different classification techniques are employed on the developed corpus (*BDTC*). The techniques are: traditional machine learning (ML)-based (Mirończuk & Protasiewicz, 2018), convolutional neural network(CNN)-based (Zhou, 2020), sequential-based (Sakalle, Tomar, Bhardwaj, Acharya, & Bhardwaj, 2021) and transformer-based (Ambalavanan & Devarakonda, 2020). Each of the technique is described in the following subsections in details.

5.2.1. Traditional ML

This work investigates the performance of three ML-based techniques: GPU based support vector machine (SVM) (Catanzaro, Sundaram, & Keutzer, 2008), GPU based Libsvm (Wen, Shi, Li, He, & Chen, 2018) and stochastic gradient descent (SGD). Various parameters are tuned on the developed corpus to generate ML-based classifier models to perform the Bengali text classification task. The SVM and Libsvm parameters are: kernel = sigmoid, tol = 0.00001, decision function shape = over and remaining parameters are used as default value. Libsvm and SVM classification parameters are almost similar, but Libsvm is faster than SVM. The parameter loss = modified_huber and alpha = 0.001 are used for the SGD classifier, and the rest of the parameters are used as default value. The ML classifiers are not suitable for improving accuracy for a higher dimension and a large number of samples. These techniques also failed to capture local and global text semantics.

5.2.2. Convolutional NN

Three variations of CNN-based techniques have been investigated in this study, such as very deep convolutional neural networks (VDCNN) (Conneau et al., 2017), deep convolution neural networks (DCNN) (Hossain & Hoque, 2019) and convolutional neural network (CNN) (Kim, 2014). The interior of each convolutional are NN techniques are explained in the following.

- **CNN:** A single layer multi-kernels CNN architecture (Kim, 2014) with three embedding models (FastText, GloVe & Word2Vec) are used to investigate the Bengali text classification task performance. The distinguished kernels are 3, 4 and 5, whereas the number of filters is 128, 128 and 256. The convolution layer follows a 1D max-pool layer and activation layers. Finally, all of the polled features are appended one after another and applied to a dropout operation with a threshold value of 0.045. The softmax layer or output layer are responsible for classifying the document using 512 flatten features.
- **DCNN:** The DCNN architecture contains three convolution layers with different filters (e.g., 16, 32 and 64). Each of the convolution layer is followed by activation and max-pooling layers. There are two fully connected (FC) layers, and one softmax layer is in the frameworks. Each of the FC layers are densely connected by 512 neurons. The softmax layer uses sigmoid activation function and generates the predicted category name using the FC layer feature values.
- **VDCNN:** This work proposes VDCNN architecture to perform the Bengali text document categorization task. The layered architecture and interior design are inspired by VDCNN (Conneau et al., 2017), ResNet (Kaiming, Xiangyu, Shaoqing, & Jian, 2015) and VGG (He, Zhang, Ren, & Sun, 2016a) philosophy. However, VDCNN feature extraction work on character-level embedding is not suitable for achieving good performance of Bengali classification tasks (Khatun, Rahman, Islam, & Marium-E-Jannat, 2019). The original VDCNN takes more training time and suffers from model overfitting problem due to the shortage of corpus and deep convolution operation. The VDCNN architecture combined with different embedding techniques is capable of reducing few convolution operations to improve Bengali text classification performance (Section 6.1). The detailed architecture and preparation of VDCNN architecture are described in Section 6.

5.2.3. Sequential

There are two sequential classification techniques used in this research: long short term memory (LSTM) (Behera, Jena, Rath, & Misra, 2021) and gated recurrent units (GRU) (Chung, Gülçehre, Cho, & Bengio, 2014). A combination of CNN with LSTM and GRU has also been implemented with the developed corpus (*BDTC*). A detailed description of each technique is presented in the following.

LSTM: A two-layers LSTM is employed in this research. The sequential layers contain the following parameters: max sequence = 512, hidden dimension = {128, 256}, batch size = 16, dropout = {0.49, 0.38}, loss = categorical_crossentropy, optimizer = adam, activation = softmax. A maximum epoch of 50 is used on the developed corpus. Although a higher number of sequences degrades the classification performance. A max sequence length (or a number of words) of 1024 and 2048 is also tried in this work.

GRU: The two-layers GRU contains the following parameters: hidden state = {256, 256}, max sequence = 512, batch size = 32, epoch = 80, dropout = {0.50, 0.47}, loss = categorical_crossentropy, optimizer = adam and activation function = {tanh, softmax}. The last GRU layer follows a 1D max-pool layer. Finally, 512 feature values for softmax layer are concatenated and the softmax layer generates the prediction of the expected category.

CNN + LSTM: A single layer multi-kernel CNN (Kim, 2014) and a single layer LSTM are combined (CNN + LSTM) and used in this work. There are three 1D kernels (e.g., 3, 4 & 5) and 128 filters used in the convolution layer. The convolution layer follows an activation and 1D max-pool (e.g., 3, 4 & 5) layer. The pooled feature concatenating the pooled feature are fed into LSTM cells. The LSTM cell applied a dropout (e.g., 0.49) operation, and its output feeds to the softmax layer. Finally, the softmax layer produces the prediction values.

CNN + GRU: A Single layer GRU and multi-kernel CNN (Kim, 2014) are communally (CNN + GRU) used in this study. A single layer with three 1D kernels (e.g., 2, 3 & 4) and 256 filters conduct the convolution operation. A tanh activation and 1D max-polling operation are applied after the convolution operation. The pooled feature feeds to the GRU cell, and the GRU imposes a dropout operation for the model generalization. Finally, the softmax layer takes input from GRU output and generate the category-wise prediction.

5.2.4. Transformer

A transformer-based pre-trained m-BERT model is used for feature extraction purpose, whereas the model is trained with the developed corpus (*BDTC*) using the BERT technique. The classification feature dimension of 768, and the number of multi-head attention of 12 with 12 hidden layers. The batch size of 8 and 30 epoch helped converging the model. There is a vocabulary list for multilingual language containing 105,879 vocabularies for 104 languages (Devlin et al., 2019).

6. Proposed text document categorization system

The proposed *TDC* system architecture comprises five main modules: Embedding model generation, two text to feature extraction modules, VDCNN training module, and VDCNN testing module. The proposed document categorization system shown in Fig. 3. The embedding model generation phase generates the embedding model which is inserted into the training and testing text representation modules to generate features. VDCNN training module creates the classifier model using features of training samples, whereas the testing module produces the document category of the test document based on the classifier model and features of the unlabeled test sample. Fig. 3 illustrates the constituent-wise details architecture of the proposed VDCNN based document categorization system.

6.1. Embedding model generation

There are many embedding techniques (e.g., GloVe, Word2Vec,

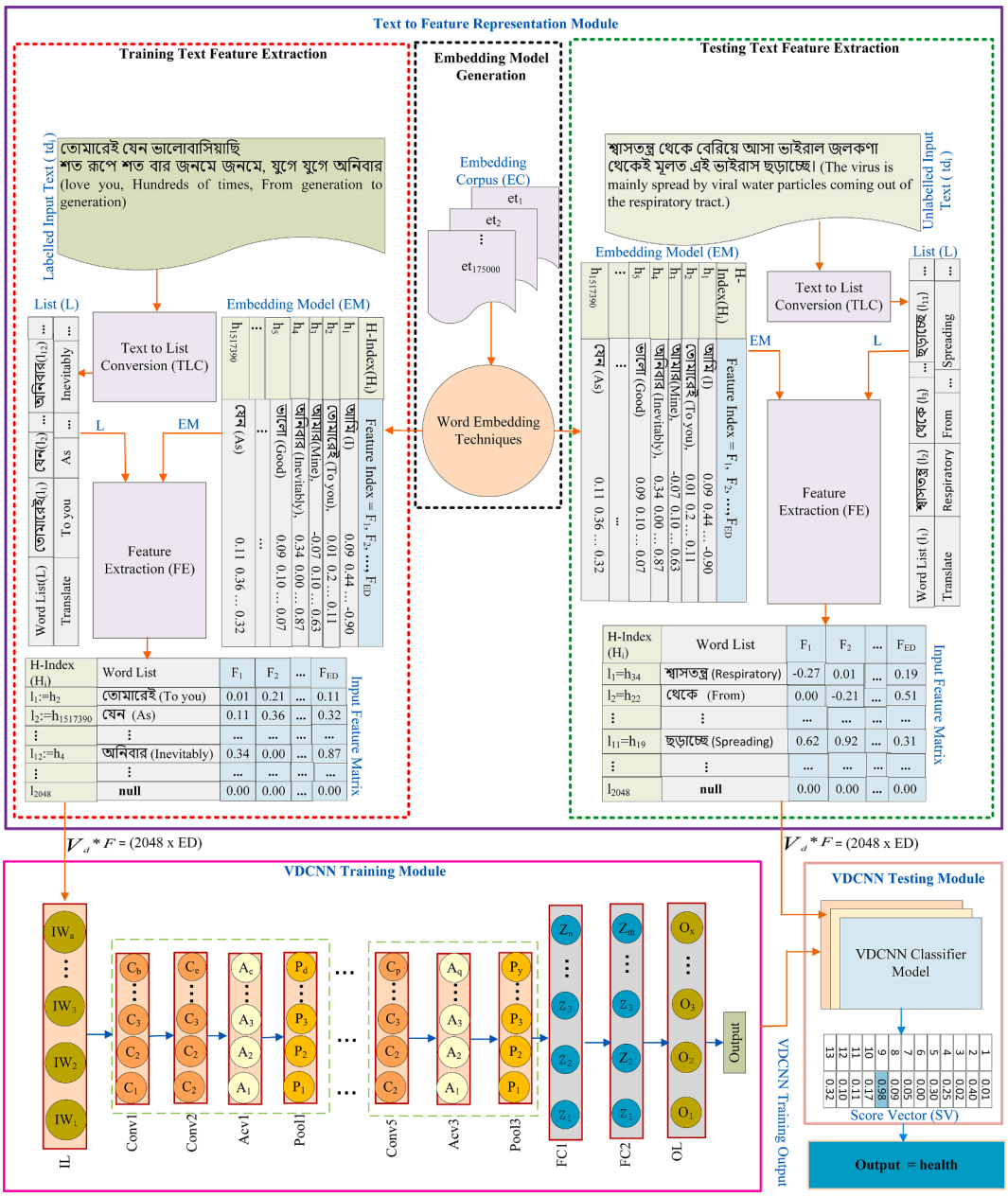

Fig. 3. Proposed very deep convolution neural network based text categorization.

FastText & m-BERT) and pre-trained embedding models available for English text language (Pennington et al., 2014; Mikolov et al., 2013; Bojanowski et al., 2017; Devlin et al., 2019), and their performances are outstanding for downstream tasks (Moirangthem & Lee, 2021a; Enríquez, Troyano, & López-Solaz, 2016). Nevertheless, a limited number of embedding models that provides a set of optimized parameters are available for low resource languages including Bengali. Most of the embedding techniques have shared some standard parameters such as embedding dimension (ED), contextual window (CW), word frequency, learning rate, epoch number and context type. Among these parameters, ED, CW and minimum word frequency (min_count) are the most influential for semantic and syntactic features representation. The overall impact of three parameters (e.g., ED, CW & min_count) on classification accuracy is interrelated, and their actual combinations vary from corpus to corpus. An exponential combination of these parameters is possible and it is nearly impossible to generate these huge amount of combinations manually. Thus, this study proposed an embedding parameters identification (EPI) Algorithms 3 to select a set of optimized parameters from the embedding corpus. The Algorithm 4 is part of Algorithm 3 which use an objective function to maximize the semantic and syntactic similarity accuracy based on the given corpus. In particular, the proposed EPI Algorithm 3 is considered 89 ED, 24 CW and 96 minimum word-frequencies. Thus, a total of 205,056 [e.g., ED (89) x CW (24) x min_count (96)] parameters are generated, which is quite impossible to optimize by the traditional trial-and-error method. Out of 205,056, the proposed Algorithm 3 selected the best 33 parameters (ED: 11, CW: 3 & min_count: 1) for the downstream tasks (text classification), which aided to reduce the overall classification model generation time.

The embedding model generation process initializes with an embedding corpus (*EC*) which is a collection of unlabeled embedding text. The *EC* can be represented as $EC : EC = \{et_1, et_2, et_3, \ldots, et_D\}$, where D denotes the total number of embedding text in *EC*. The total number of embedding text (D) is equal to 969, 000 in this research. There are three word embedding techniques are used in the proposed system e.g., GloVe (Pennington et al., 2014), FastText (Bojanowski et al., 2017) and Word2Vec (Mikolov et al., 2013).

Embedding model generation techniques are described in the next subsections.

Algorithmic 3 Automatic embedding parameters identification (EPI) based on intrinsic accuracy (e.g., semantic & syntactic)

```
Input : Embedding corpus (EC), semantic word similarity datasets (SE),
syntactic word similarity datasets (ST)
Output : Optimize hyperparameters
ED←{}    ▷ Optimize embedding dimensions
CW←{}    ▷ Optimize contextual window
min_count←0    ▷ Optimize minimum word frequency
X_MAX←95    ▷ Maximum word frequency
gSE←0, gSY←0    ▷ Global semantic and syntactic accuracy
procedure EPIEC    ▷ Embedding Parameters Identification (EPI)
  SE←0, SY←0    ▷ local semantic and syntactic accuracy
  i, j, k←0    ▷ Global initialization
  for i←10→450 do    ▷ Increment with 5 (i+=5)
    SeAcc←0, SyAcc←0    ▷ Local semantic and syntactic accuracy
    for j←2→25 do    ▷ Increment linearly j+=1
      SeAcc1←0, SyAcc1←0    ▷ Local semantic, syntactic accuracy
      for k←min_count→X_MAX do    ▷ Increment linearly k+=1
        model←EmbeddingTechnique(EC, i, j, k)
        SticAc, SyticAc←intrinsicEvaluation(model, SE, ST)
        SeAcc1, SyAcc1←max(SeAcc1, SyAcc1, SticAc, SyticAc)
      end for
      SeAcc, SyAcc←max(SeAcc1, SyAcc1, SeAcc, SyAcc, 1)
      ED, CW, min_count←Update(SeAcc, SyAcc, i, j, k)
    end for
    SE, SY←max(SeAcc, SyAcc, SE, SY)
    ED, CW, min_count←Update(SE, SY, i, j, k, 2)
  end for
end procedure
```

Algorithmic 4 Update the embedding dimension, contextual window and min frequency words based on global semantic and syntactic accuracy

```
procedure UpdatesemacticAc, syntacticAc, i, j, k, Flag
  if Flag == 1 then    ▷ Picked optimized the contextual window
    if fabs(gSE − semacticAc) < 1e − 3 then
      if fabs(gSY − syntacticAc) < 1e − 3 then
        gSE←semanticAc, gSY←syntacticAc
        CW.append(j)
      end if
    end if
  end if
  ifFlag == 2 then    ▷ Picked optimized embedding dimension (ED) and min
  frequency count
    if fabs(gSE − semacticAc) < 1e − 3 then
      if fabs(gSY − syntacticAc) < 1e − 3 then
        gSE←semanticAc, gSY←syntacticAc
        ED.append(i)
        min_count←k
      end if
    end if
  end if
  return ED, CW, min_count    ▷Optimized parameters sets
end procedure
```

6.1.1. GloVe

The GloVe technique generates a total of 33 embedding models using embedding corpus *EC* with the combinations of embedding dimension (ED) and context window (CW) where ED ∈ {25, 50, 100, 150, 200, 250, 275, 300, 325, 350, 400} and CW ∈ {6, 9, 12}. The remaining hyperparameters are set as default settings e.g., *min_count* : 2, *X_MAX* : 95, *epochs* : 30, *binary* : *true*, *memory* : *true* and *threads* : 8 respectively. GloVe generates the feature vector based on global word count, local word count, word-word co-occurrence and local context with the centre word. These characteristics of GloVe can extract better semantic and syntactic features. However, it consumes more time due to matrix factorization.

6.1.2. FastText (Skip-gram & CBOW)

FastText is a prediction based embedding technique and carries the sub-word information. This technique takes *EC* as an input and generates an embedding model as the output uses gensim library (Řehůřek & Sojka, 2010). FastText is a modified version of Word2Vec where FastText works with sub-word information. Two variants of FastText, such as the skip-gram with negative sampling (*negative* : 25) and CBOW with hierarchical loss, were used. A 66 embedding model (e.g., 33 for skip-gram & 33 for CBOW) has been generated using hyperparameters *ED* and *CW* where ED ∈ {25, 50, 100, 150, 200, 250, 275, 300, 325, 350, 400} and CW ∈ {6, 9, 12}. The rest of the hyperparameters are used as default settings e.g., *min_count* = {2}, *epochs* = 30, learning rate (*lr*) : 0.098, minimum an-gram 2 and maximum n-gram 6 respectively. Minimum two hours required for *CW* : 6 and *ED* : 25 where the maximum thirty-one hours has been spend for *CW* : 12 and *ED* : 300.

6.1.3. Word2Vec (Skip-gram & CBOW):

Word2Vec is a prediction based embedding technique and produces the embedding vector from the centre word to the context word or vice versa. This technique takes *EC* as input and produces the embedding model as the output using the gensim library. There can be 66 embedding models (e.g., 33 for Skip-gram & 33 for CBOW) produced from the combinations of hyperparameters ED and CW with ED ∈ {25, 50, 100, 150, 200, 250, 275, 300, 325, 350, 400} and CW ∈ {6, 9, 12}. The negative sampling *negative* : 25 and *epoch* : 30 are used for the skip-gram model and hierarchical loss value for the CBOW model. The rest of the hyperparameters are set as default settings (e.g., *min_count*, lr & threads).

Every embedding techniques consists of its own pros and cons. The significant advantage of the GloVe embedding is that it takes the whole corpus statistics during word feature extraction, but Word2Vec explores the local window-based statistics only. The GloVe considers local and global word frequency and word-to-word co-occurrences, whereas the FastText considers only local word frequency. GloVe embedding consumes less training time compared to the m-BERT. The primary cons of the Glove embedding are the consumption of massive memory during matrix factorization time than the Wor2Vec. A higher frequency of stop words may change the actual feature values in GloVe embedding, which is not affected by FastText embedding. The GloVe embedding considers the local context of a word, whereas m-BERT assesses the full context of a word in the corpus. The three embedding techniques can generate 165 embedding models (e.g., 33 for GloVe, 66 for FastText & 66 for Word2Vec). All of these embedding models are evaluated by the intrinsic evaluators (Section 7.1.1). Among these models, we selected the best performing 12 embedding models for document classification purpose (Section 8.1). The default hyperparameters of GloVe, FastText and Word2Vec Algorithms are not suitable for all types of text processing. That is why the model should be tuned with the best hyperparameters before training.

6.2. Training/testing text feature extraction

There are two texts to feature extraction modules: one is for the labelled input text, and the other is for the unlabelled input text. Each module comprises Labelled/Unlabelled input text, text to list conversion (TLC), List (*L*), Embedding Model (EM) and Feature Extraction (FE). The labelled input text is used for the VDCNN training module, and the unlabelled input text is used for the VDCNN testing module. The training or testing text feature representation takes the labelled or unlabelled text document as the input and produces a 2D feature matrix as the output which goes into the VDCNN training module or into the VDCNN testing module.

The training/testing text feature extraction is performed in three phases: text to list conversion (TLC), GloVe model generation, and feature extraction (FE). The TLC is initialized with the labelled input text document (td_i), and it converts the input text as a word list vector (*L*). The list vector is a collection of words defined as $L = [l_1, l_2, l_3, \ldots, l_N]$, where l_i denotes the i^{th} word for $i = 1, 2, 3, \ldots, N$. N is the maximum

length of L and $N = 2,048$. If an input text contains more than $2,048$ words, then it is truncated to the first $2,048$ words. It uses zero padding if the number of words is smaller than $2,048$.

The FE process considers the list vector (L) an embedding model (EM) as inputs. The list L is a 1D vector containing a total of $2,048$ words. The EM is a 2D vector containing a total of $1,517,390$ unique words each assigned an individual h-index and *ED* corresponding feature values whereas ED $\in \{25,50,100,150,200,250,275,300,325,350,400\}$. For every word in L, a set of (H_i) is generated using FE. FE process extracts the features from the EM by mapping (H_i) to EM. If the L of (H_i) is found in EM, then FE returns the corresponding (H_i) feature values. Otherwise, it returns a null vector for *ED* features. For example, the first word (l_1) is mapped to the index h_2 of EM. The FE phase generates an output of feature matrix (V_d*F), where V_d denotes the number of words and F denotes the number of features. The i^{th} word l_i extracts a vector of *ED* features ($F_1,F_2,F_3,\ldots,F_{ED}$) from the EM. Each word of a text document is arranged in rows and corresponding features in columns. As a result, each document is represented by an input feature matrix of dimension ($2048 \times ED$) where each row of the feature matrix is an H-Index (H_i),H_i : $H_i = \{h_1,h_2,\ldots,h_{2048}\}$ and each column is a feature $F_j : F_j = \{F_1,F_2,F_3,\ldots, F_{ED}\}$. The testing text feature extraction takes the unlabelled text document as the input and produces a 2D feature matrix as the output using the GloVe model generation and FE processes. The process of the 2D feature matrix generation is the same as the training text feature extraction technique. The upper right part of Fig. 3 illustrates the process of the testing text feature extraction.

6.3. VDCNN training module

The training module generates a document classifier model with the Bengali corpora. The input of the VDCNN is the 2D feature matrix (V_d*F), which propagates through several layers such as the input layer, convolution layer, pooling layer, activation layer, and fully connected layer. Fig. 4 shows the layers of VDCNN training module.

6.3.1. Input layer

The Input Layer (*IL*) is a set of tensor nodes $IL = \{IW_1,IW_2,IW_3,\ldots, IW_a\}$, where a represents the tensor length of 2048. This layer brings the initial feature vector from the word embedding model and transmits this vector to subsequent VDCNN layers for further processing. The size of input layer tensor is *Tensor* (2048, *ED*), where ED indicates the embedding dimension. In the next subsection, the tensor is presented as *Tensor*(*heights*, *width*, *filter*), where height and width are substituted by the number of words and features length. The filter is substituted by the number of filters.

6.3.2. Convolution layer

It is a local feature extractor layer where the kernels are well trained for weight adjustment using the back-propagation technique (Rumelhart, Hinton, & Williams, 1986). In the proposed architecture, there are five convolution layers used at different depths. The convolution layers are *Conv*1, *Conv*2, …, *Conv*5. The layer Conv1 comprises tensor nodes ($C_1,C_2,C_3,\ldots,C_b$), where C denotes the tensor nodes and the subscript (b) denotes the tensor height and $b = 2048$ at Conv1 and gradually decreases in the following layers. The i^{th} layer input propagates through to the output defined by Eq. (3).

$$\tau^i = \tau^{i-1}. \tag{3}$$

where τ^i represents i^{th} layer kernel/weight metric. During the convolution, each kernel is multiplied by the input matrix element-wise.

The coordinate position such as $input(s,r)$ is updated according to Eq. (4) where s and r denotes a coordinate position in the 2D input matrix.

$$input(s,r)^i = \sum_{p=0}^{k_h^i} \sum_{q=0}^{k_w^i} input(p+s, q+r) \times K(p,q) \tag{4}$$

where, k_h^i, and k_w^i denotes the i^{th}-layer kernel height and width. p, and q represent the kernel coordinate. The kernel K(p,q) is sliding from the left to right and top to bottom in each position. The kernel sliding or convolution operation extracts the local (e.g., sentence level) and global (e.g., document level) features based on the document semantics. The small kernel size extracts the local details, and the large size kernel extracts the abstract document view.

In each convolution layer, the number of outputs is equal to the number of the kernels. All parameters are broadcasting from the output of this layer to the next layers. The tensor size of the five convolution layers (*Conv*1 to *Conv*5) are *Tensor*$(2048, ED, 8)$, *Tensor*$(2046, ED_1, 16)$, *Tensor*$(1021, ED_2, 32)$, *Tensor* $(1017, ED_3, 64)$ and *Tensor* $(337, ED_5, 128)$ respectively. For example, tensor attributes of *Conv*1 are 2048,*ED*, and 8 meaning the number of words, embedding dimension and number of filters, respectively. The embedding dimension ED to ED_5 values are changed according to layer operations.

6.3.3. Activation layer

In this layer, each node determines whether the value of the current node is delivered or not to the next layer depending on the activation function. A non-linear activation function ReLU is used in the element-wise operation in the input tensor matrix. ReLU is an aide to optimization function that speeds up the training process (Agarap, 2018). The i^{th} layer tensor metric value is affected by the ReLU operation (Eq. (5)).

$$input(s,r)^i = max(input(s,r)^{i-1}, 0) \tag{5}$$

where, $input(s,r)$ denotes (s,r) index value of i^{th} layer. All non-zero nodes fire for the next layer operation in the activation layer. There are three activation layers (*Acv*1, *Acv*2, and *Acv*3) and each of the layer tensor shape are *Tensor*$(2042, ED_1, 16)$, *Tensor*$(1011, ED_4, 64)$ and *Tensor*$(336, 1)$, *Tensor* $(335, 1)$, …, *Tensor*$((332,1),128)$ respectively. The *Acv*1 exposes as $(A_1,A_2,A_3,\ldots,A_c)$, where A indicates the activation tensor nodes, and c represents the tensor height.

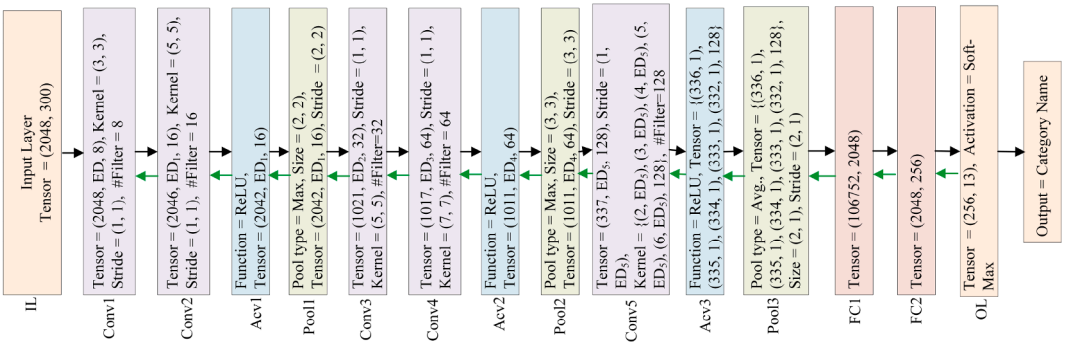

Fig. 4. Proposed VDCNN classifier architecture. Left green arrow indicates the backpropagation and right black arrow denotes the forward propagation.

6.3.4. Pooling layer

The pooling operation generalizes the input feature map and reduces the input dimension. There are three Pooling layers (*Pool1*, *Pool2*, and *Pool3*). The Pooling tensors are denoted as $P_1, P_2, P_3, \ldots, P_d$, where P denotes tensor nodes and d is the tensor height. The first two-layer tensor shapes are $Tensor(2042, ED_1, 16)$, $Tensor(1011, ED_4, 64)$ and $Tensor((336, 1), (335, 1), \ldots, (332, 1), 128)$ respectively. Pooling summarizes the output over a whole adjacent element. This operation substitutes the output of a tensor at an index by the final statistic of the adjacent outputs. The max-pool function is used in this proposed VDCNN.

6.3.5. Fully connected layer

Two fully connected layers (FC_1 and FC_2) are used in the proposed architecture. The internal tensor nodes are represented by $Z_1, Z_2, Z_3, \ldots, Z_{(n \vee m)}$, where the subscript $(n \vee m)$ denotes the lengths of node (FC_1 or FC_2). The convolution layer extracts high-level features which flatten in the fully connected layer. The first fully connected layer contains 106, 752 neurons and the neurons are connected with 2, 048 neuron in the next layer. The fully connected layers propagate feature values with a linear model defined in Eq. (6).

$$W^i = W^{i-1} \times \omega^i \tag{6}$$

where, W^i denotes the i^{th} layer weight and ω^i indicates the flattened feature values. A dropout operation is applied to control the model overfitting. The dropout operation generalizes the high-level features. The training process generated is a hierarchical structure and is a well-trained weighted kernel.

6.3.6. Output layer

In this layer, input tensor takes an 1D feature map ($Tensor(256, 1)$) and outputs an expected category name. The internal nodes of the Output Layer (OL) represent $O_1, O_2, O_3, \ldots, O_x$, where the subscript (x) denotes the length of the flattened feature map. The expected score is calculated using Eq. (7).

$$E_i = \frac{\sum_{j=0}^{g} W(j,i) \times X(j)^i}{\sum_{k=1}^{CL} \sum_{j=0}^{g} W(k,j) \times X(j)^i} \tag{7}$$

where, E_i represents the i^{th} class expectation and $X(j)$ denotes the feature vector. g is the feature-length, which is 256. The accuracy of the classifier depends on the values of g. In this implementation, we fixed the value of g to 256 to optimize the classifier performance. The class value is normalized by the Soft-Max function. The maximum value with the corresponding index is the expected class name.

The expected value of a class is subtracted from the actual value to adjust the classification error. Error (i.e. the subtracted value) is back-propagated to updated the weights of *FC* and *Conv* layers. The backward and forward propagation processes are continued until the model is converged or error is minimized according to the predefined value.

In this study, the aim is to investigate the classification performance of a pre-train multi-lingual transformer based BERT (Devlin et al., 2019), DCRNNs (Hossain & Hoque, 2021), LibSVM (Chang & Lin, 2011), SGD (Kabir et al., 2015), DCNN (Hossain & Hoque, 2019), CNN (Kim, 2014), GRU (Moirangthem & Lee, 2021b) and CNN-LSTM (Behera et al., 2021) model with the same datasets. The classification performance are summarised in Table 8.

6.4. VDCNN testing module

The purpose of the testing module is to assign an expected class from a set of predefined classes to an unlabelled text document (td_i). This module takes the result of VDCNN training model and the output of the testing text feature extraction module (i.e., an input feature matrix, V_d*F). The VDCNN testing layers load the trained metafiles and assign the kernel weights. The feature matrix $(2048 \times ED)$ is passed through the VDCNN classifier model to produce the output of a 2D score vector (SV). The SV is a set of classification scores, $\{s_1, s_2, s_3, \ldots, s_{CL}\}$, where CL is the number of classes. SV is computed using the Eq. (8).

$$s_i = \frac{e^{s_i}}{\sum_{j=1}^{CL} e^{s_j}} \tag{8}$$

here, e^{s_i} denotes the i^{th} score exponential (e) value and CL is the total number of classes. The maximum expected value of an index represents the corresponding class and is the final output class. For example, in the case of the unlabeled input text in Fig. 1, VDCNN classifier model produced the maximum expected value of 0.98 at the index 9. The classifier generates category or class corresponding to this index is *health*.

7. Experiments

The system is implemented on a multi-core processor with NVIDIA GTX 1070 GPU. The physical memory of the processor is 32*GB* and GPU internal memory is 8*GB*. The VDCNN architecture is deployed in the Tensor-Flow framework of Python.

7.1. Evaluation measures

The proposed VDCNN based document categorization system is evaluated in four phases: embedding model phase, training phase, validation phase and testing phase.

7.1.1. Embedding model evaluation phase

Embedding model evaluation is referred to the characteristic assessment of feature vectors which is an essential task for low-resource languages (Hossain & Hoque, 2020). Intrinsic and extrinsic evaluations are used for evaluating the embedding model. Intrinsic evaluators evaluate the semantic, syntactic and relatedness quality whereas extrinsic evaluators evaluate the downstream tasks. The Spearman ($\widehat{\rho}$) and Pearson ($\widehat{r}$) correlations are used for intrinsic evaluation such as semantic word similarity ($S_{s\widehat{\rho}}/S_{s\widehat{r}}$) and syntactic word similarity ($S_{y\widehat{\rho}}/S_{y\widehat{r}}$).

7.1.2. Training/validation phase evaluation

The evaluation of the training and validation phases is performed in terms of loss and accuracy. In the case of large data size, batch-wise training and validation are most common in any deep learning framework. In a single iteration, a fixed number of batch size data is processed for training. The i^{th} batch with j^{th} example expectation $E(i,j)$ is calculated by Eq. (9).

$$E(i,j) = \sum_{k=0}^{256} W(j,k) \times X(j,k) \tag{9}$$

here, $W(j,k)$ denotes the weight and $X(j,k)$ indicates the feature value of j^{th} sample with k^{th} index. The probability $P(i,j)$ of i^{th} batch with j^{th} sample is calculated by Eq. (10).

$$P(i,j) = \frac{e^{(E(i,j))}}{\sum_{y=0}^{CL} e^{(\sum_{k=0}^{256} W(y,k) \times X(y,k))}} \tag{10}$$

where, $e^{(E(i,j)}$ denotes the normalized expectation and CL indicates the number of classes. The batch-wise loss and accuracy are computed according to Eqs. (11)–(12).

$$L^i = -\sum_{l=0}^{SP} \sum_{y=0}^{CL} R(l,y) \times \log_2 P(l,y) \tag{11}$$

$$A^i = \frac{H^i}{SP} \tag{12}$$

where, L^i, A^i, and H^i represent the logarithmic-loss, accuracy and number of correctly predicted class in i^{th} batch respectively. Moreover, SP indicates the total data sample in i^{th} batch whereas $R(l, y)$ denotes the number of correct prediction in i^{th} batch. The total loss and accuracy are computed by averaging all batch-wise loss and accuracy values overall training/validation samples.

7.1.3. Testing phase evaluation

The evaluation of the testing phase is performed in terms of accuracy and error. In addition to that several statistical measures such as precision (p), recall (r), micro f_1 score, confusion matrix and accuracy (a) are also used to evaluate the performance of the proposed document categorization system (Wu, Zhang, Shen, Huang, & Gu, 2020; Alhaj et al., 2019).

7.2. Hyperparameters optimization

In order to optimize the hyper-parameters, the proposed VDCNN classifier model is initialized with various parameter values which are shown in Table 3. The proposed system is implemented on Intel Core i7-7700K 4.5 GHz processor, 32 GB of RAM, NVIDIA GeForce GTX 1070 graphics card, NVIDIA GeForce with 1,920 CUDA Cores, Ubuntu 16.04, Python 3.6, CUDNN *v*6.0, and Tensorflow-GPU 1.4.0 with CUDA 8.0 toolkit.

In order to examine the effect of various hyper-parameters on the accuracy, the developed corpus is divided into three sets: (i) Small Data Set (SDS), (ii) Medium Data Set (MDS), and (iii) Large Data Set (LDS) or *BDTC* which contains whole data of the developed corpus. Table 4 shows the summary of data sets. It can be observed that *BDTC* is the voluminous dataset with the highest number of documents, words and size (in MB).

A vocabulary is produced by aligning each word to an integer index. Word embedding is the most widely used technique to illustrate the vocabulary of a textual document (Akhter et al., 2020). Words in a document can be illustrated as a vector of real-values and the model performance depends on the dimension of the vector. Therefore, the embedding dimension (*ED*) can be selected as a hyper-parameter. Fig. 5 shows the effect of *ED* on the classifier performance.

Form Fig. 6, it is evident that the accuracy depends on the embedding dimension and the size of the dataset. SDS, MDS, and *BDTC* obtained the highest accuracies for the embedding dimensions of 325, 325, and 300 respectively.

Batch size indicates the number of samples assigned at a time to the classifier for extracting in a particular iteration (Akhter et al., 2020). The batch-wise operation increases the GPU performance and has a significant impact on classifier performance that is observed in Fig. 6.

Huge batch size uses up greater memory of GPU and slows down the training process. In other words, the processing of different batch size strongly depends on the physical memory (RAM) and GPU memory. It is observed from Fig. 6 that a batch size of 64 achieved about 98.2% accuracy for proposed developed data set (*BDTC*). Due to the GPU memory limitations, it is limited to use a batch size of 64. Due to the reasonable accuracy achieved, batch size is fixed to 64 for all the evaluations. The huge batch size demands substantial memory, requires higher time to process data and reduce the overall accuracy while the small batch size improves the results (Keskar, Mudigere, Nocedal, Smelyanskiy, & Tang, 2016).

Table 3
VDCNN initial hyperparameters.

Parameters	Parameters Value
Batch size	32
GloVe embedding dimension	200
Kernel size	(1, 1), (3, 3), (5, 5), (7, 7) (2, 45), (3, 45) 2048, 256
Number of filter	4, 8, 16, 32, 64
Pooling size	(2, 2), (3, 3), (2, 1)
Learning rate	0.1
Dropout rate	0.50
Activation	ReLU, SoftMax
Pooling type	Max, Avg.

Table 4
Summary of the three datasets.

Properties	SDS	MDS	*BDTC*
No. of documents.	85,000	10,500	156,207
Size	369.44 MB	456.61 MB	678.98 MB
No. of sentences.	0.9345 M	1.155 M	1.7175 M
No. of words	21.37609 M	26.419886 M	39.286713 M
Max. documents size	21 KB	26 KB	38 KB
Min. documents size	230 B	600 B	1 KB
No. of categories	13	13	13

Dropout is a simple mathematical operation that is used to anticipate the classifier model from overfitting and noisy features. The classifier may fall in the difficulty of overfitting due to the training with the limited dataset. This difficulty can be solved by increasing the size of the dataset or decreasing the number of hidden units. Dropout eliminates or disables the disengaged units in the hidden layer which do not take part in the computation on subsequent iterations. Fig. 7 illustrates the performance of the classifier model on various dropout values. The result indicates that the classifier model achieved the highest accuracy on 0.56 dropout on developed dataset (*BDTC*).

The learning rate determines the optimum termination time requirements for a classifier model. A large value for learning rate is responsible for quick termination while a small value for learning rate requires a very long time to terminate the model. The effects of the various learning rates on classifier performance are shown in Fig. 8.

The accuracy is increased gradually up to 97.25% at a learning rate of 0.01, which is the maximum. After the learning rate of 0.01, the accuracy is decreased slowly, and the trend continues due to model overfitting.

In order to compute the feature maps, convolution operation utilizes one or more filters with distinct or same kernel size. The number of filters and the size of the kernel can influence network performance. Larger kernel size and greater numbers of filters obtained better accuracy but slowed down the training/testing process. On the other hand, small kernel size and fewer filters degrade the performance by missing distinguishing features (Akhter et al., 2020). There are various possible combinations of kernel size and number of filters. Table 5 shows the few such combinations that affect the performance.

In the initial experiment, the kernel starts from the size $(1, 1)$ and ends at $(5, 21)$ with a number of filter combinations $(4, 8, 16, 32, 64)$ in each convolutional layer, which achieved 92.64% accuracy. The kernel size and number of filter combinations are varied on trial and error basis. A combination of large and small kernel size and variable size filters in different layers perform better (highest accuracy 96.75% for the combination 7).

The epoch number is one of the important linear hyperparameters of VDCNN classifier model. The whole documents are passed through the model once in a forward pass and a backward pass in a single epoch. A careful selection of a number of epochs is necessary because quite a number of epochs can induce overfitting which in turn affect the classifier performance. Fig. 9 shows the effects of a number of epochs on classifier performance. For this developed *BDTC* dataset, the classifier model achieved the maximum accuracy (97.42%) on around 700 epochs.

Max-average pooling is used in this proposed model. Softmax layer is used to acquire a confidence score that helps to determine the level of the categorization. After running several experiments on the develop

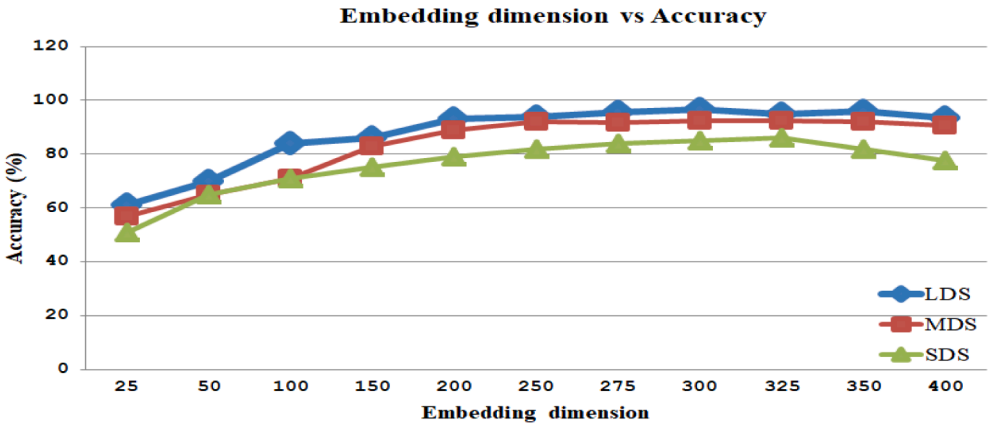

Fig. 5. Impact of ED on network performance.

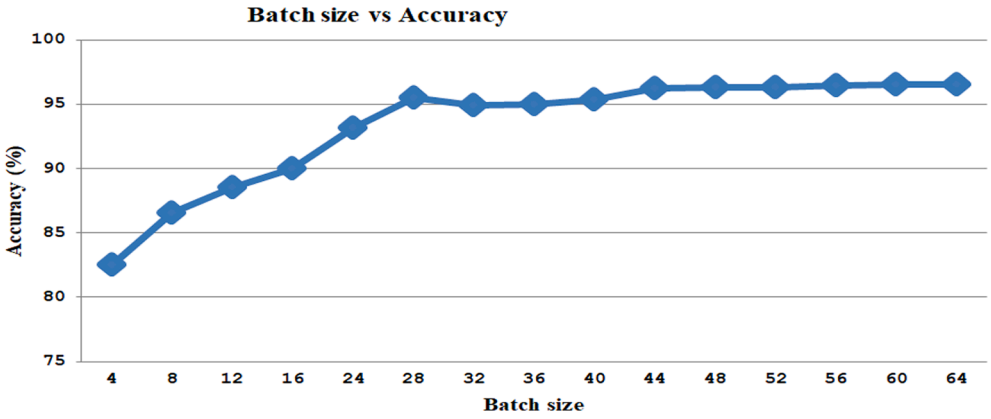

Fig. 6. Effect of batch size on classifier performance.

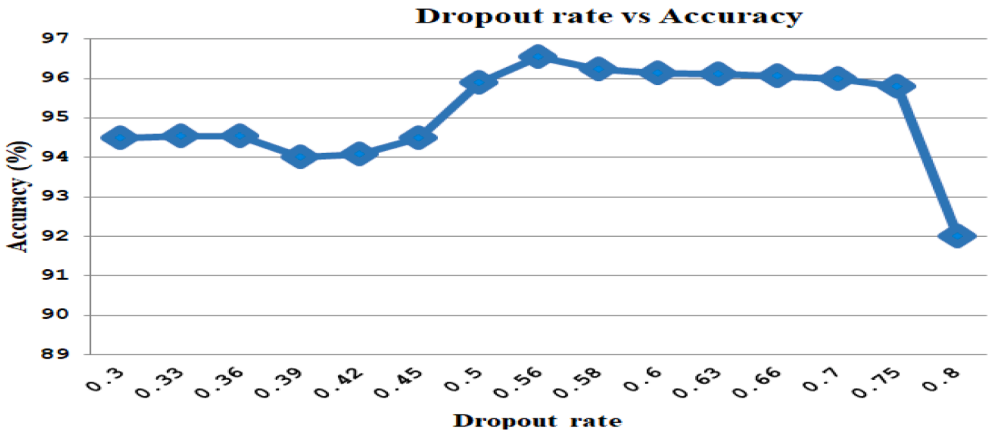

Fig. 7. Accuracy of classifier on distinct dropout values.

corpus, we choose the optimized hyperparameter values are chosen (Table 6).

8. Results

After optimizing all the hyperparameters, the performance of the proposed VDCNN based Bengali document categorization is evaluated in various ways which are described in the following subsections.

8.1. Embedding model evaluation

In this work, various combinations of hyperparameters three embedding techniques (e.g., GloVe, FastText and Word2Vec) have generated 165 local contextual embedding models (33 for GloVe, 66 for FastText (Skip-gram & CBOW), 66 for Word2Vec (Skip-gram & CBOW)). Intrinsic evaluators evaluate a total of 166 models (165 for local contextual embedding and one for pre-trained m-BERT) using syntactic and semantic similarity measures (Chiu, Korhonen, & Pyysalo, 2016). Based on the intrinsic evaluation performance, a total of 13 top-performing models are selected to perform the downstream task (i.e.,

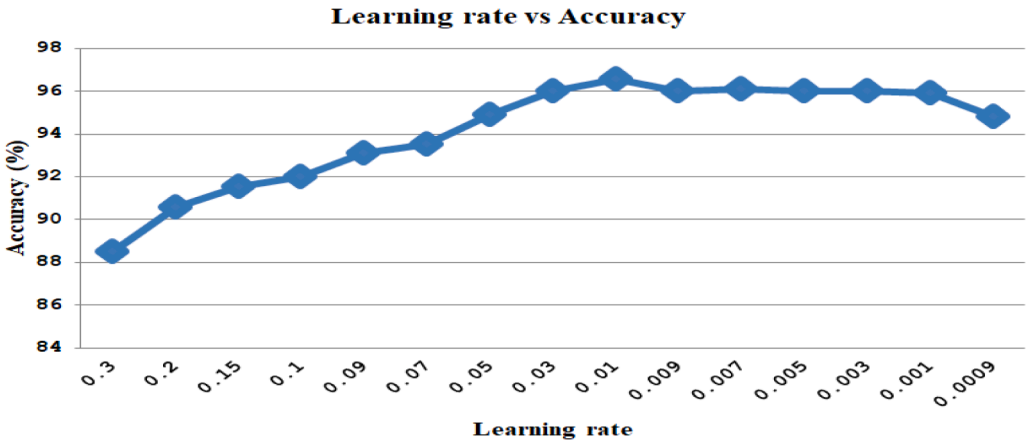

Fig. 8. Effects of learning rate on classifier performance.

Table 5
Effect of various kernel size and number of filters.

Combinations	Kernel Size	Number of Filter	Accuracy (%)
1	C1=(1, 1) C2=(3, 3) C3=(5, 5) C4=(7, 7) C5=(2, 17) (3, 17) (5, 17)	F1 = 4 F2 = 8 F3 = 16 F4 = 32 F5 = 64	92.67%
2	C1=(3, 3) C2=(5, 5) C3=(7, 7) C4=(9, 9) C5=(2, 21) (3, 21) (5, 21)	F1 = 10 F2 = 15 F3 = 20 F4 = 25 F5 = 30	90.82%
3	C1=(3, 3) C2=(5, 5) C3=(5, 5) C4=(7, 7) C5=(2, 45) (3, 45) (4, 45) (5, 45)	F1 = 32 F2 = 64 F3 = 128 F4 = 256 F5 = 256	95.31%
4	C1=(3, 3) C2=(5, 5) C3=(5, 5) C4=(7, 7) C5=(2, 45) (3, 45) (4, 45) (5, 45) (6, 45)	F1 = 8 F2 = 16 F3 = 32 F4 = 64 F5 = 128	96.75%

text classification). Table 7 exhibits the intrinsic evaluation results of the 13 best performing models for 1000 semantic and syntactic word pairs.

The result shows that the GloVe model with 300 embedding dimension (ED) and 9 contextual window (CW) achieved the highest Spearman correlation ($S_{s\hat{\rho}}$) of 63.12% for measuring the semantic word similarity task. The GloVe model also achieved the highest Pearson correlation ($S_{s\hat{r}}$) score of 64.05% with the same ED and CW. In measuring the syntactic word similarity task, the GloVe model achieved the highest Spearman correlation ($S_{y\hat{\rho}}$) score of 71.75% with 200 ED and 12 CW, whereas the highest of 73.04% Pearson correlation ($S_{y\hat{r}}$) is achieved with the 300 ED and 9 CW. The contextual pre-trained m-BERT model achieved lower scores than the GloVe embedding due to a shortage of distinct vocabulary and unavailability of morphological variations (such as Sadhu-bhasha and Cholito-bhasha). As a result, this model faces the out-of-vocabulary (OOV) problems (Bojanowski et al., 2017), affecting the semantic and syntactic evaluation performance. The FastText (Skip-gram) and Word2Vec (Skip-gram) cannot carry out reasonable accuracy due to the lack of global word-word co-occurrence information(Pennington et al., 2014). The CW with 6 does not perform well, and the CBOW version of FastText and Word2Vec embedding techniques are excluded from the analysis due to their lower performance.

As regards to the semantic similarity, the GloVe embedding achieved a maximum Spearman correlation of 63.12% and Pearson correlation of 64.05%, whereas the Word2Vec achieved a maximum of 45.57% (Spearman correlation) and 46.09% (Pearson correlation). Concerning

Table 6
Optimized hyperparameters for VDCNN.

Hyperparameters	Value
Batch size	64
GloVe embedding dimension	300
Kernel size	(3, 3), (5, 5), (5, 5), (7, 7), (2, 45), (3, 45), (4, 45), (5, 45), (6, 45), 2048, 256
Number of filter	8, 16, 32, 64, 128
Pooling size	(2, 2), (3, 3), (2, 1)
Number of epoch	700
Learning rate	0.01
Dropout rate	0.56
Activation	ReLU, SoftMax
Pooling type	Max-Avg.

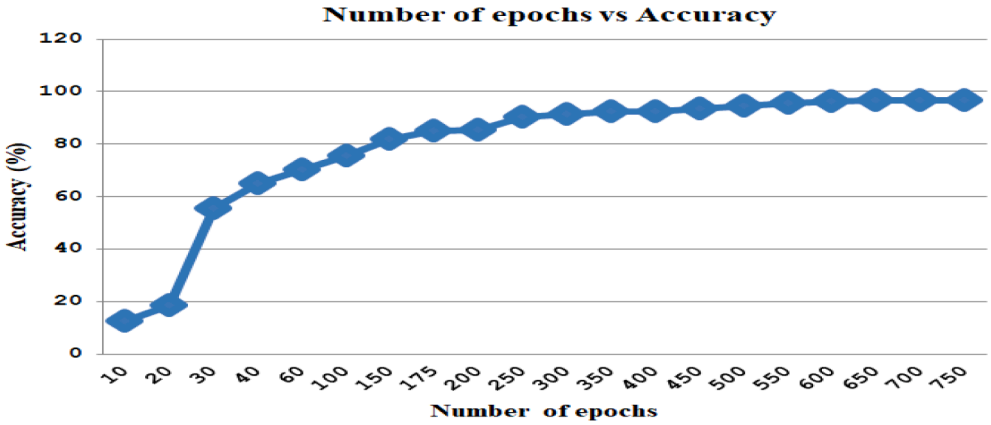

Fig. 9. Effect of epochs on classifier performance.

Table 7
Intrinsic evaluation performance of 13 top-performed models concerning semantic and syntactic word similarities.

Model	ED	CW	Semantic (%)		Syntactic (%)	
			$S_{s\hat{\rho}}$	$S_{s\hat{r}}$	$S_{y\hat{\rho}}$	$S_{y\hat{r}}$
GloVe	**300**	**9**	**63.12**	**64.05**	68.79	**73.04**
	250	9	62.02	62.98	67.78	71.92
	200	**12**	55.68	56.97	**71.75**	72.43
	100	12	53.32	54.96	66.88	70.06
FastText (Skip-gram)	300	9	58.01	57.98	45.00	45.87
	250	12	56.74	57.89	44.79	45.98
	200	12	54.72	54.12	40.23	39.43
	100	12	53.12	52.98	39.01	37.11
Word2Vec (Skip-gram)	300	9	45.57	46.09	32.66	31.29
	250	9	44.00	44.17	30.97	29.77
	200	12	43.53	43.98	30.00	28.90
	100	12	42.20	42.14	29.01	28.97
Pre-trained m-BERT	768	–	60.17	61.02	65.27	68.92

Table 8
Bengali text classification task performance of26 best-performed models.

Techniques	Models	ED	CW	f_1-score	A(%)
Traditional ML	TF-IDF + SVM	–	–	77.24	78.17
	Word2Vec (Skip-gram)+ Libsvm	300	9	81.40	82.06
	GloVe + Libsvm	300	9	82.10	83.15
	FastText (Skip-gram)+ Libsvm	300	9	81.89	82.72
	TF-IDF + SGD	300	9	79.31	80.41
	Word2Vec (Skip-gram)+ SGD	300	9	84.15	85.31
	GloVe + SGD	300	9	**85.08**	**86.20**
	FastText (Skip-gram)+ SGD	300	9	84.78	85.98
Convolutional NN	GloVe + CNN	200	12	89.41	90.82
	Word2Vec (Skip-gram)+ CNN	300	9	89.97	91.23
	FastText (Skip-gram)+ CNN	300	9	90.06	91.79
	Word2Vec (Skip-gram)+ DCNN	300	9	92.18	93.23
	GloVe + DCNN	300	9	93.50	94.86
	FastText (Skip-gram)+ DCNN	300	9	92.69	93.85
	Proposed(GloVe + VDCNN)	300	9	**97.00**	**96.96**
	GloVe + DCNN	200	12	92.50	93.16
Sequential	Word2Vec (Skip-gram)+ LSTM	300	9	91.98	92.15
	GloVe + LSTM	200	12	92.30	92.80
	GloVe + CNN-GRU	200	12	89.47	90.07
	GloVe + LSTM	300	9	94.14	94.76
	GloVe + CNN-LSTM	300	9	**95.14**	**95.20**
	FastText (Skip-gram)+ LSTM	300	9	93.40	93.97
	FastText (Skip-gram)+ CNN-LSTM	300	9	94.15	94.88
	Word2Vec (Skip-gram)+ CNN-LSTM	300	9	92.73	93.00
	GloVe + CNN-LSTM	200	12	92.95	93.12
Transformer	**m-BERT**	768	-	-	**92.45**

the syntactic similarity, a maximum of 71.75% (Spearman correlation) and 73.04% (Pearson correlation) are achieved for the GloVe embedding. On the other hand, the Word2Vec obtained a maximum of 32.66% (Spearman correlation) and 31.29% (Pearson correlation). Word2Vec embedding achieves whether words occur in related contexts based on the local window only, whereas the GloVe embedding extracts word features based on global word frequency and word-to-word co-occurrence over the entire dataset. Thus, the GolVe embedding outperformed Word2Vec due to its better feature representation by grasping meanings, semantic relationships between words, and words similarities. The evaluation results showed that the GloVe embedding achieved the best result due to its better semantic and syntactic feature representations. Although GloVe embedding performed better in intrinsic evaluation, other methods may perform better in the downstream task evaluation (i.e., text classification). Thus, among 165 models, 13 best-performed models have been chosen for the classification task.

8.2. Training/validation accuracy and loss

Fig. 10 shows the training and validation accuracy in terms of epochs. At the beginning, the training accuracy is lower than the validation. The training accuracy increases gradually from epoch number 100, whereas the validation accuracy remained the same (i.e., the blue line in Fig. 10). The training and validation accuracy both are stable in between 300 −600 epoch, where the classifier model gained the maximum accuracy. This improvement is due to the normalized expectation value employed in Eq. (12). Fig. 11 illustrates the training and validation losses. The loss value is about 40 at the beginning, whereas the validation loss is about 2.5. The proposed system converges completely at epoch 100. The loss values almost remained constant and linear in between 150 to 600 epochs. This faster convergence is due to the normalized expectation used in Eq. (10).

8.3. Text classification task evaluation

To investigate the Bengali text classification task performance, 109 models are implemented using the combinations of embedding models and classification techniques (ML, CNN, sequential, BERT). In particular, 108 models are generated from three classification techniques where three ML-based techniques contributed to 36 models [12 (embedding model) × 3 (SVM, Libsvm & SGD)], three CNN-based techniques contributed to 36 models [12(embedding model) × 3 (CNN, DCNN, VDCNN)] and three sequential-based technique generated 36 models [12(embedding model) × 3 (LSTM, GRU, GRU + LSTM)]. One model is generated from the BERT-based technique (m-BERT). Among 109 models, Table 8 illustrates the result of 26 top-performing models, which are selected based on different classification techniques (8 for ML, 8 for CNN, 9 for sequential, 1 for transformer-based). Among 8 ML models, GloVe + SGD gained the maximum accuracy of 86.20% (f_1-score = 85.08%) to perform the Bengali text classification task. The SVM and Libsvm classification techniques cannot generate the separable decision boundary in the developed training corpus (*BDTC*) due to the lack of generalized text features and word inflexion rate. Thus, the performance of SVM and Libsvm-based models showed lower accuracy. Out of 36 CNN-based models, the results of *eight* best-performed models

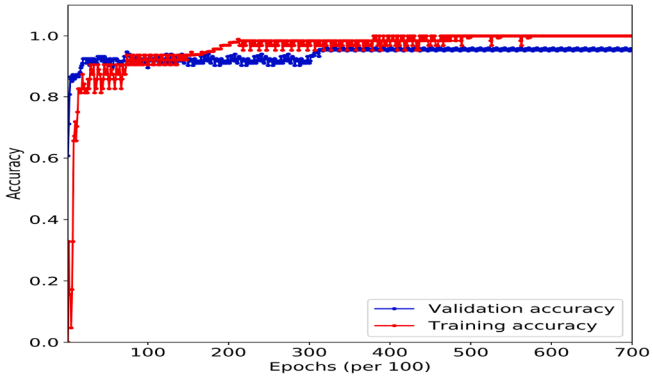

Fig. 10. Training and validation accuracy.

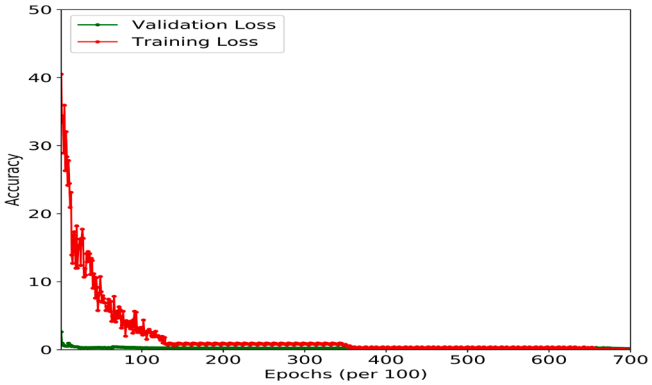

Fig. 11. Training and validation losses.

are shown in Table 8. The results showed that the GloVe + VDCNN achieved the highest accuracy of 96.96% (with an average f_1-score of 97.00%) than other CNN-based combinations. The CNN is implemented using a single layer multi-kernel (Kim, 2014) and this technique suffer from capturing adequate semantic and syntactic features (at the sentence and document levels). As a result, the CNN-based models showed a quite lower accuracy compared to DCNN and VDCNN-based models. Among 10 best performing sequential-based models, the GloVe + CNN-LSTM gained the maximum accuracy of 95.69% (f_1-score = 95.14%). The GRU contains fewer trainable parameters concerning LSTM. These fewer parameters cannot extract all relevant features from the large text datasets (e.g., *BDTC*). As a result, the GRU based model did not perform well using the developed corpus (the highest accuracy of 90.07% is obtained for CNN-GRU). This work investigates the Bengali text classification task performance using a transformer-based technique (m-BERT) on the developed corpus. The pre-trained m-BERT model achieved 92.45% accuracy. The pre-trained m-BERT model is well-performed for the English language, but the performance of this pre-trained model is not satisfactory for the low resource language (e.g., Italian (Gambino & Pirrone, 2019)), including the Bengali.

The results revealed that the proposed model (GloVe + VDCNN) obtained the maximum accuracy of 96.96% among all models. The traditional feature extractors (TF, IDF & TF-IDF) are not able to capture the semantic and syntactic features, whereas the GloVe, Word2Vec (Skip-gram), FastText (Skip-gram) and m-BERT can capture these type of features. Also, ML classifiers cannot capture the local (e.g., word & sentence level) and global (e.g., document level) semantics, whereas the CNN, DCNN, VDCNN, LSTM, GRU, and m-BERT can carry the local and global semantics. The Bengali language has morphological variations in textual form: Sadhu-bhasha and Cholito-bhasha. The morphological variations like Sadhu-bhasha and Cholito-bhasha are not considered in the pre-train model. The m-BERT model pre-trained the Wikipedia Bengali dataset, which suffers from handling the variations of Cholito-bhasha and Cholito-bhasha of Bengali. However, the proposed GloVe model coverage these morphological variations with the most extensive vocabulary sets of the developed corpus. Thus, the GloVe + VDCNN model obtained better performance than pre-trained m-BERT. On the other hand, Word2Vec (Skip-gram) and FastText (Skip-gram) only considered the contextual windows, and these models cannot cover the whole corpus and word pair statistics. However, the GloVe model has taken into consideration of the entire corpus and word pair statistics. The GloVe model reduces the feature vector values for high-frequency words, but FastText (Skip-gram) and Word2Vec are not considering these characteristics. In summary, the experimental analysis revealed that the proposed method (GloVe + VDCNN) achieved the maximum accuracy (96.96%) than all other models due to its better feature extraction capability and hyperparameters selection strategy. A recent study on embedding has also shown that the FastText (Skip-gram), Word2Vec (Skip-gram), and pre-trained m-BERT cannot be performed well for the Italian language (a low-resource language) (Gambino & Pirrone, 2019).

8.3.1. Statistical evaluation

Table 9 presents the result of the several statistical measures (i.e., precision, recall, f_1-score measure) of the Bengali document classifier. The testing data set contains 21,477 Bengali text documents which are exclusive of the train and validation data sets. Here the term support represents the number of documents.

Two categories *lifestyle* and *sports* achieved the highest f_1 score which are close to 0.99. Only one category (*art*) achieved the lowest f_1-score and precision values (f_1-score = 0.87, and p = 0.78) due to the failure of classification of few large documents. Other categories provided fairly good classification scores.

8.4. Receiver Operator Characteristic (ROC)

ROC is a measure for the performance of a classification model in graphical representation. Fig. 12 shows the ROC curve of each category with its Area Under the Curve (AUC) value. The x-axis represents the false positive rate, and the y-axis represents the true positive rate. The AUC values are exposing the probability value of the respective category.

The diagonal curve represents the random selection of class probability. The maximum AUC value is 99.00%, and the values are covered by eight categories (health, lifestyle, accident, crime, education, entertainment, politics, and sports, respectively). The minimum AUC value is 96.00%, that value covered by opinion and art categories. An intra-class separability threshold is 0.96 to 0.99, which revealed that the proposed VDCNN based document categorization provided the better separability value.

8.5. Performance comparison

In recent years, few research activities have been carried out on text classification in Bengali. Due to the unavailability of other authors' dataset and benchmark corpus, the contemporary techniques are implemented on the developed corpus (BDTC) including existing methods (Hossain & Hoque, 2021; Ahmad & Amin, 2016; Kabir et al., 2015; Hossain & Hoque, 2018; Hossain & Hoque, 2019; Dhar et al., 2020). The performance of the proposed system is compared with the previous approaches in terms of corpus size and accuracy. Table 10 presents the number of training documents, testing documents, and classes used in the existing approaches and the proposed method.

Table 10 indicates that the proposed approach is developed on the largest training and testing datasets (106,899 and 21,477) with a higher number of classes (13 classes) than the existing approaches of Bengali document categorization.

Table 9
Statistical evaluation of document classifier.

Document Category	Precision	Recall	f_1-score	Support
Sports	0.99	0.98	0.99	1,958
Lifestyle	0.98	0.99	0.99	2,017
Art	0.78	0.98	0.87	236
Science & technology	0.97	0.92	0.95	1,183
Education	0.96	0.98	0.97	970
Environment	0.95	0.92	0.93	534
Entertainment	0.98	0.98	0.98	3,439
Economics	0.94	0.96	0.95	953
Health	0.99	0.97	0.98	859
Opinion	0.96	0.93	0.94	1,985
Politics	0.97	0.97	0.97	3,713
Crime	0.97	0.98	0.98	2,814
Accident	0.98	0.98	0.98	816
Avg/total	0.97	0.97	0.97	21,477

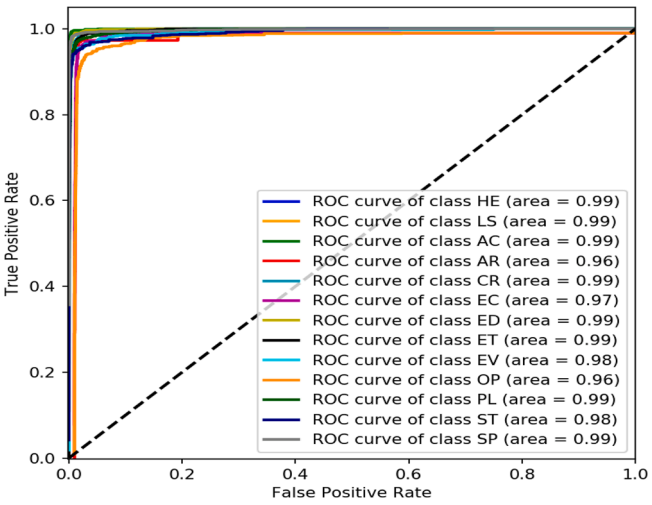

Fig. 12. ROC in terms of true positive and false positive rates.

Table 10
Summary of previous Bengali document classification techniques concerning training size, testing size and classes.

Approaches	#Training Data	#Testing Data	#Class
GloVe + DCRNNs (Hossain & Hoque, 2021)	120,000	36,207	12
Word2Vec + K-NN + SVM (Ahmad & Amin, 2016)	19,750	4,713	7
TF-IDF + SGD (Kabir et al., 2015)	5,448	3,679	9
Word2Vec + SGD (Hossain & Hoque, 2018)	10,000	4,651	9
Word2Vec + DCNN (Hossain & Hoque, 2019)	86,199	10,707	12
Graph-based + LSTM-RNN (Dhar et al., 2020)	-	14,373	9
Proposed (GloVe + VDCNN)	**106,899**	**21,477**	**13**

Table 11 summarizes the comparison of text document categorization systems in Bengali.

Results indicated that the accuracy of Libsvm based classifiers gained accuracy between 78.17% and 83.51% on developed *BDTC* dataset. The accuracy of the SGD based classifications varied from 80.41% to 86.20%. DCNN based classifier achieved the maximum accuracy of up to 94.86%. This research is comparing all the previous approaches and other methods (such as Word2Vec + Libsvm, Glove + Libsvm, Glove + SGD, and Glove + DCNN), which leads to the observation that the proposed VDCNN based technique achieved significantly higher accuracy of 96.96% outperforming all the approaches.

Usually, the performance of the classification model heavily depends on the number of training samples and classes (Hashemi, Yang, Mirzamomen, & Kangavari, 2009; Alhawarat & Aseeri, 2020). When the number of the classes increases, the classification technique faces inseparability problems (Hossain & Hoque, 2019). Thus, the classification model starts losing its uniqueness and degrades its accuracy. Moreover, the classification model relies on better data distributions while training with more samples. These helped the classification model to improve accuracy with various distributions of the large corpus. Although further investigations with different corpus size and classes are deemed necessary, the preliminary results in Table 11 revealed that the proposed model's performance (i.e., accuracy) increased with the lower number of document classes and with a higher number of training samples.

Table 11
Summary of the performance comparison.

Approaches	Accuracy (%)
GloVe + DCRNNs(Hossain & Hoque, 2021)	95.89
Word2Vec + Libsvm (Chang & Lin, 2011)	82.06
GloVe + Libsvm	83.51
TF-IDF + SGD (Kabir et al., 2015)	80.41
Word2Vec + SGD (Hossain & Hoque, 2018)	85.31
GloVe + SGD	86.20
Word2Vec + DCNN (Hossain & Hoque, 2019)	93.23
GloVe + DCNN	94.86
Proposed (GloVe + VDCNN)	**96.96**

8.6. Human experts vs. proposed technique

The performance of the proposed system of Bengali document categorization is compared with the human experts (HEs) in terms of accuracy. Three experts who are postgraduates in Bengali linguistic are asked to manually classify the 21,477 text documents into one of 13 categories. A single document is read by three experts individually and classify it accordingly. A document which achieved the highest number of category count is selected as the final category. The accuracy of the human experts is calculated by the ratio of the total number of documents that predicted the class correctly and the total number of test documents. The accuracy of the proposed system is determined from the confusion matrix by the ratio of correctly classified count and total documents. For example, in case of the health category, the system correctly predicts 833 documents out of 859 documents which produced the accuracy of 96.97% (833/859) (See in Table 13). Table 12 shows a summary of the comparison.

ANOVA (Analysis of Variance) analysis shows that there are no significant differences on accuracy achieved by the human experts while assigning the categories of text documents ($F(2,24) = 0.959, p = 0.39, \eta^2 = 0.050$). Moreover, the average accuracy of the human experts 1, 2 and 3 are 99.5%, 99.7% and 97.3% respectively. The analysis of comparison shows that human experts are better in identifying documents than the proposed method (99.49% vs. 96.96%). Although the proposed system shows almost similar performance as human experts in identifying lifestyle and art categories, there are significant differences for other categories (such as crime, environment, opinion, and science & technology respectively). ANOVA analysis also demonstrated that there is a substantial statistical differences on accuracy measures between human experts and VDCNN methods ($F(1,12) = 16.07, p = 0.0017, \eta^2 = 0.39$). Fig. 13 also illustrates this result.

The performance of the proposed system can be improved by adding more distinct data distribution in the lower support categories and extracting more common semantic and syntactic features.

Table 12
Performance comparison between human experts and proposed System.

Category Name	Human-Experts Accuracy (%) (correctly predicted)	VDCNN Accuracy(%) (correctly predicted)	Support
Sports	100.00 (1,958)	98.26 (1,924)	1,958
lifestyle	100.00 (2,017)	98.86 (1,994)	2,017
Art	97.46 (230)	98.31 (232)	236
Science & technology	100.00 (1,183)	92.10 (1,089)	1,183
Education	100.00 (970)	98.15 (952)	970
Environment	100.00 (534)	91.76 (490)	534
Entertainment	100.00 (3,439)	98.10 (3,373)	3,439
Economics	100.00 (953)	96.43 (919)	953
Health	98.95 (850)	96.97 (833)	859
Opinion	98.18 (1,949)	93.00 (1,846)	1,985
Politics	98.79 (3,668)	97.15 (3,607)	3,713
Crime	100.00 (2,814)	98.44 (2,770)	2,814
Accident	100.00 (816)	97.55 (796)	816
avg/total	99.55 (21,381)	96.96 (20,825)	21,477

Table 13
Proposed system confusion matrix.

CM	HE	LS	AC	AR	CR	EC	ED	ET	EV	OP	PL	ST	SP
HE	833	9	0	0	2	2	0	2	1	3	2	3	2
LS	3	1994	0	1	0	1	0	14	0	1	0	1	2
AC	0	0	796	0	17	2	0	0	1	0	0	0	0
AR	0	0	0	232	0	0	4	0	0	0	0	0	0
CR	0	0	6	0	2770	0	3	0	2	11	19	2	1
EC	0	0	0	0	3	919	0	0	4	12	4	10	1
ED	0	0	0	0	3	0	952	7	0	4	4	0	0
ET	0	7	2	27	4	1	7	3373	0	8	9	1	0
EV	2	1	0	0	9	1	0	3	490	9	16	1	2
OP	1	10	3	28	21	1	7	17	13	1846	30	7	1
PL	1	0	1	8	28	16	12	7	3	26	3607	1	3
ST	4	5	1	1	2	32	7	23	1	10	4	1089	4
SP	0	0	0	1	4	7	1	12	0	2	5	2	1924

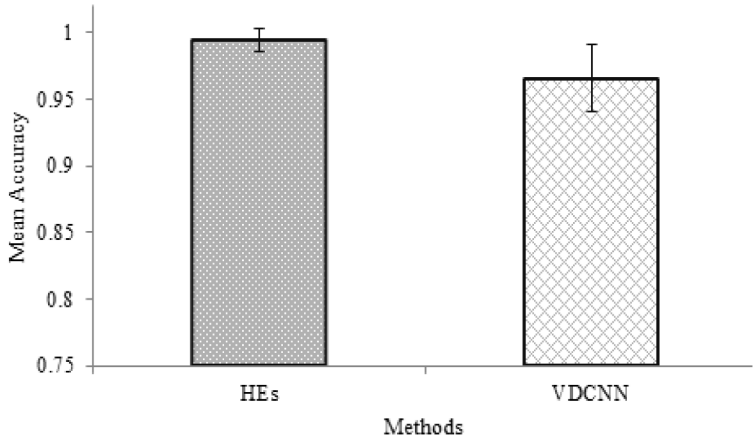

Fig. 13. Mean accuracy obtained by the human experts and VDCNN. Error bar indicates the standard deviation.

8.7. Error analysis

The error analysis deliberates the qualitative comparison of the proposed model with the other models, e.g., shown in Fig. 14. In contrast, the proposed model misclassified quantitative analysis represented in Table 8. Fig. 14, $Input_1$, taken from Blog[6] and it contains more than 500 words. The ground labelled is Science & Technology, whereas the proposed model, Pre-trained m-BERT and GloVe + CNN-LSTM model, is predicted correctly. But GloVe + SGD is incorrectly predicted. For $Input_2$, taken from Newspaper[7] that contains more than 300 words. The proposed model only predicts the correct label, but other models do not predict the correct labelled due to morphological (e.g., word level) variation. Sample $Input_3$, collected from online Newspaper[8] title, and that contains only four words. As a result, the proposed model and other model are not able to predict correctly. That means the proposed model is not capable of predicting concise text. For $Input_4$, taken from Newspaper[9] and all of the models are correctly predicted. Thus, Fig. 14 summarized that the proposed system is not working for concise text (e.g., 3 to 4 words), and other models cannot predict the morphological variant text, e.g., Sadhu-bhasha, Cholito-bhasha and Asamiya Bengali.

The confusion matrix (CM) is a table that is used to summarize the performance of a classifier in tabular form (Stehman, 1997). Table 13 shows the confusion matrix, which represents the summary of prediction results on the proposed Bengali text categorization system. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This matrix visualizes the classification error/confusion when it makes predictions. The diagonal blue cell represents the number of correct predictions of texts as the actual class and other cells represent the misclassification as other classes. For example, the HE class is evaluated for a total of 859 unlabeled texts. Among these, 833 texts are correctly predicted as HE class represented in the first diagonal cell of Table 13 texts are misclassified as other classes.

The confusion matrix illustrates that the minimum misclassification

[6] https://cadetcollegeblog.com/rabbi-ahmed/62535
[7] https://www.asomiyapratidin.in/congress-5-guarantees-of-brahmaputra-valley-went-to-barack-4-guarantees/
[8] https://www.prothomalo.com/lifestyle/fashion
[9] https://www.bbc.com/bengali/news-56669407

Input Text	Source	Ground Labelled	Predicted Labelled (SoftMax-Score)				
			Proposed (GloVe+VDCNN)	Pre-trained Multilingual-BERT	GloVe+CNN-LSTM	GloVe+SGD	Remarks
Input$_1$ = ডিজিটাল বাংলাদেশ' নামের এক শব্দবন্ধের সঙ্গে আমাদের তারুণ্যের বেড়ে ওঠার অনেক... **Transliteration**=Ḍijiṭāla bānlādēśa' nāmēra ēka śabdabandhēra saṅgē āmādēra tāruṇyēra bēṛē ōṭhāra anēka... **Translation** = With a phrase called 'Digital Bangladesh', our youth grows a lot ...	Blog	Science & Technology	Science & Technology (0.62)	Science & Technology (0.24)	Science & Technology (0.48)	Education (0.16)	Yes
Input$_2$ = ফাকিবাঁজ কংগ্রেছৰ ধূর্তালি : ব্রহ্মপুত্র উপত্যকাৰ ৫ গেৰান্টী বৰাকত হ'ল গৈ ৪,নিৰুদ্দিষ্ট 'কা'ৰ প্রসংগ... **Transliteration** = Phākibām̐ja kaṅgraēchara dhūrtāli: Brahmaputra upatyakāra 5 gēraāṇṭī baraākata ha'la gai 4,nirauddiṣṭa 'kā'ra prasaṅga... **Translation** = Fakibanj Congress scam: 5 guarantees in Brahmaputra Valley Barakat Gai 4, missing 'Ka' context ...	Newspaper	Politics	Politics (0.39)	Crime (0.19)	Opinion (0.27)	Art(0.09)	Yes
Input$_3$ = স্নিগ্ধ সৌন্দর্যের ট্রেন্ডি কালেকশন **Transliteration**=Snigdha saundaryēra ṭrēnḍi kālēkaśana **Translation** = Trendy collection of soft beauty	Newspaper	Lifestyle	Environment (0.51)	Entertainment (0.36)	Environment (0.44)	Opinion (0.27)	No
Input$_4$ = বাংলাদেশে আক্রান্তের ৮১ শতাংশই দক্ষিণ আফ্রিকান ভ্যারিয়ান্ট **Transliteration**=Bānlādēśē ākrāntēra 81 śatānśa'i dakṣiṇa āphrikāna bhyāriẏānṭa **Translation** = 81% of the victims in Bangladesh are South African variants	Newspaper	Health	Health (0.73)	Health (0.61)	Health (0.89)	Health (0.55)	Yes

Fig. 14. Comparison of ground labelled and predicted labelled with a different techniques.

rate occurred in the lifestyle category (1.00%), whereas the maximum misclassification rate achieved in the environment category (8.99%). It is complied with the confusion matrix that out of 21,477 documents, the proposed system correctly predicted 20,825 documents and incorrectly predicted 652 documents (Table 13). Several overlaps have occurred between categories in classifying documents. As an example, the health category correctly classified 833 documents out of 859. Out of the remaining 26 documents, nine are overlapped with LS category, and two overlapped with CR, EC, ET, PL, SP categories, three overlapped with OP, ST categories and one overlapped with EL category respectively. There was no overlap with AC, AR and ED categories. It is apparent from the confusion matrix that more documents should be included in some categories during training phase for improvement of performance (such as in LS category). In between frequent class overlap (e.g. CR and AC) can be reduced with the addition of more documents.

9. Discussion

The proposed system building theory is developed based a series of empirical investigations over a period of time (Sections 8.1 and 8.3) which led to the final development of the GloVe + VDCNN model for the text document categorization in Bengali. In the following, a number of key features and findings are highlighted:

- Availability of benchmark corpus is a crucial constituent to develop any *intelligent system*. Bengali is considered one of the resource-poor languages due to the unavailability of the benchmark dataset and other related resources. Thus, to develop an intelligent system for the Bengali text classification, this research had to develop two corpora: (i) an embedding corpus (*EC*) (ii) text classification corpus (*BDTC*) with 13 categories. Bengali text has two forms of written morphological variants, such as Sadhu-bhasha and Cholito-bhasha. Thus, to develop an *intelligent* text classification system in Bengali for real-world applications, these variants are to considered while building the embedding and training models. Multilingual-BERT (m-BERT) is a large model trained with 104 monolingual datasets, including Bengali Wikipedia. However, Wikipedia only one variant (i.e. Cholito) of Bengali text. The developed corpora considered both forms of text by crawling data from several sources. For example, Shadhu-bhasha texts were accumulated from the newspapers (e.g. Asomiya Pratidin) and Bengali literature. However, Cholito-bhasha texts have been collected from the newspapers (e.g., Prothomalo) and online portals (e.g., bdnews24.com). Thus, the research community and industries working on developing *intelligent* Bengali language technology tools can use this developed dataset.
- Embedding model evaluation is a crucial issue to develop a deep learning-based *intelligent* classification system. The performance of an intelligent classifier depends on the nature of the embedding

model and dataset. Thus, selecting an appropriate embedding model and its parameters is critical in the text classification. An Algorithm (i.e., EPI) is proposed to identify the suitable embedding parameters for low resource languages (including Bengali). In this Algorithm, a new objective function is devised in selecting the best embedding parameters. Although further investigations with other languages are needed for generalization, this Algorithm can be applied to any low-resource languages to develop an *intelligent* text classification model.

- Identification of appropriate embedding model on embedding dataset is another critical issue for developing an *intelligent* classification system based on deep learning. Thus, the embedding model should be evaluated using standard evaluators before performing the downstream task (e.g., machine translation, text classification, word sense disambiguation and so on). A total of 165 embedding models have been evaluated using intrinsic evaluators in finding the most suitable model. In particular, semantic similarity and syntactic similarity measures have been used to evaluate 165 embedding models. Among 165 models, 13 best performing models are selected for text classification. Spearman correlation and Pearson correlation scores are used determining the best models. The intrinsic evaluations showed that GloVe-based embedding outperformed other embeddings (Word2Vec, FastText, BERT) by achieving the highest accuracy on semantic similarity and syntactic similarity measuring tasks. The performance of the pre-trained m-BERT model is not good because of its limited unique words, out-of-vocabulary problem and incapability to handle Bengali morphological variants (such as Sadhu-bhasha and Cholito-bhasha). The FastText (Skip-gram) and Word2-Vec (Skip-gram) provided reduced performance due to the deficiency of global word-to-word co-occurrences. Moreover, the intrinsic evaluation results revealed that the GloVe model defeated the Word2Vec model by grasping better words semantics and word similarities. Thus, the intrinsic evaluation help finding the suitable embedding model for developing an *intelligent* system in Bengali that reduces the time during classification model generation and improves the classification accuracy.
- Although several text classification systems are available regarding high resourced languages, the model developed in one language cannot be applied to another language because of their linguistics divergences. Thus, several hyperparameters have been tuned on the training corpus to prepare the models for performing classification task. The proposed system is developed on manually optimized hyperparameters and obtained the highest classification accuracy on the test corpus. Intelligent hyperparameters optimization technique may be investigated in future to compare the manual selection of hyperparameters.
- A total of 109 models have been prepared with different parameters to perform the Bengali text classification task. Experimental results on the test dataset show that the proposed GloVe + VDCNN model outperformed other classification models and available Bengali text classification systems. The classification accuracy highly depends on the text semantics. The more semantics and prominent features can generate the more separable classification model. In this sense, the GloVe model is able to extract more semantics feature from the Bengali text. The proposed VDCNN classification frameworks have achieved better result due to their layer architecture and multi-level (e.g., word level, sentence level & document level) feature extraction constituent. The performance of the proposed (GloVe + VDCNN) model has investigated with several statistical measures. It has been observed that the accuracy of the proposed system strongly depends on the number of support. For example, sports, opinion, and lifestyle categories consist of larger support values than others, so these categories gained better classification accuracy. The overall performance of the Bengali text categorization model strongly depends on the number of classes. The semantic meaning of some categories is very close to each other, which reduces the accuracy. However, the proposed classifier model has overcome this ambiguity and achieved better performance. The proposed technique achieved superior performance compared to similar research conducted on the Bengali text document categorization in terms of a more extensive data set, better accuracy, and more significant categories.
- A detailed error analysis has been conducted using the confusion matrix to understand the model's insight. Analysis revealed that the proposed system could not correctly predict the very short text due to a shortage of text semantics. The m-BERT, GloVe + CNN-LSTM, and GloVE + SGD models have failed to predict the actual class for Sadhu-bhasha text, whereas the proposed model can classify this text correctly, which shows the module's functionality considering Bengali morphological variants.
- Feature dimension plays a key role in classification accuracy in semantic meaning-based embedding model. There is a greater chance to fall the classifier model into vanishing gradient problem (Hochreiter, 1998) with larger dimension which may creates model's converge problem and increase the validation sets errors. In that case, embedding hyper-parameters should be changed. The accuracy of the proposed classifier model is performing better in between 300–700 epochs (Fig. 9). The gap between the training/validation accuracy and losses learning curves are closer to each other, which implies that the proposed classifier can overcome the underfitting and overfitting problems (Tetko, Livingstone, & Luik, 1995). It is observed that the classifier performs better after 300 epoch, but it consumes more time. This problem can be overcome by increasing the training batch size and adding uniform data distribution in each category.
- The GloVe based features extraction with VDCNN classifier system performed better at fixed document-size rather than the variable size. When the number of words per document is larger than the predefined document length, then it is truncated according to the predefined size. As a result, there is a higher chance to loss of meaning of the actual document label. In that case, the accuracy of the VDCNN classifier has reduced. In addition to that if the document size is smaller than the predefined size, then it added zero paddings. In that situation, the VDCNN classifier failed to select the best category with a maximum expected value. Overlapping within classes and between classes is another issue in VDCNN model, which causes overall accuracy to a low value. The semantic meaning of the sentence which contained the common word in different categories are different but syntactically similar. As a result, the GloVe model fails to extract the distinguishing syntactic features in the related category. If the GloVe model is unable to extract both semantic and syntactic features accurately in word embedding, then VDCNN classifier model also fails to extract the distinguishing features at the sentence level. Therefore, class overlapping issues have occurred. Class overlap problem in between and within classes may be resolved using the Deep Residual Networks (ResNet) (He, Zhang, Ren, & Sun, 2016b).
- The technique developed in the proposed method can be applicable to Bengali news portals to select news based on class, Bengali search engine and textual sentiment analysis in Bengali. Other possible implications are hospital patient management system to classify patient's disease based on medical documents of disease, security agency to classify toxic/hostile texts, library management to classify books or journals based on the subject area. The research community and language technology-related industries that work on the Bengali language can utilize the dataset and model as a baseline to research this domain further.

In general, an expert system uses AI techniques to solve complex problems that ordinarily requires human expertise within a particular domain (Sarker, Furhad, & Nowrozy, 2021). Although the human expertise is used while annotating the datasets within the domain of NLP, this work mainly focuses on Bengali text document categorization

within the scope of intelligent systems by taking into account of deep learning techniques. The fundamental characteristic of any intelligent system is its ability to learn. The proposed work used machine learning-based classification techniques (very deep CNN). Thus, the proposed system can be considered as an intelligent system. In terms of intelligent systems, the framework and techniques developed under the proposed research can be used to develop many intelligent systems such as news portal, e-library/archival management, vertical search engine, toxic or hostile text detector where Bengali text processing is a central concern.

10. Conclusion

This research presents an *intelligent* deep learning-based text classification method using parameters optimization technique for word embedding (GloVe) and VDCNN to perform Bengali text document categorization. Due to the unavailability of the Bengali text classification benchmark corpus, this work developed two new corpora: embedding corpus and hand-crafted text classification corpus within 13 categories (*BTCC*). An *intelligent* Algorithm (i.e., EPI) is introduced to identify and selects the suitable embedding parameters for low resource languages (including Bengali). In this Algorithm, a new objective function (e.g., intrinsic evaluation) is used to optimize the embedding parameters. A total of 165 embedding models is evaluated using intrinsic evaluators (i.e., syntactic & semantic similarity measures). Intrinsic evaluation results with Spearman and Pearson correlation showed that GloVe embedding is more suitable than other embeddings (i.e., Word2Vec, FastText, m-BERT) for Bengali text embedding. Analysis of experimental results for 109 classification models revealed that the proposed GloVe + VDCNN model achieved the highest accuracy of 96.96% for Bengali text classification than other models. Moreover, the proposed model outperformed the available Bengali text classification techniques with *BTCC* concerning the highest accuracy, increased number of text classes and larger corpus.

Although the proposed technique showed reasonable outcomes, future improvements can enhance accuracy using code mixed datasets with more text document classes. Variants of transformer-based feature extractor can be investigated on the developed embedding corpus. Moreover, future research may investigate the effect of variable class size and corpus size on text classification task performance. As far as we know, this text document categorization model is the first tried-and-true endeavour in *Expert and Intelligent systems* concerning the Bengali language, especially since the previous studies had much-reduced corpus and categories.

CRediT authorship contribution statement

Md. Rajib Hossain: Conceptualization, Data curation, Methodology, Writing - original draft, Software. **Mohammed Moshiul Hoque:** Supervision, Conceptualization, Writing - review & editing. **Nazmul Siddique:** Writing - review & editing, Investigation. **Iqbal H. Sarker:** Software, Validation.

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.

Acknowledgements

This work was supported by the Establishment of CUET IT Business Incubator Project, BHTPA, ICT Division, Bangladesh for the research on “Automatic Bengali Document Categorization based on Summarization Techniques”.

References

Abuaiadah, D., Sana, J. E., & Abusalah, W. (2014). Article: On the impact of dataset characteristics on arabic document classification. *International Journal of Computer Applications, 101*, 31–38. https://doi.org/10.5120/17701-8680

Agarap, A. F. M. (2018). Deep learning using rectified linear units (relu). CoRR, abs/1803.08375. url:http://arxiv.org/abs/1803.08375. arXiv:1803.08375.

Ahmad, A., & Amin, M. R. (2016). Bengali word embeddings and it’s application in solving document classification problem. In *2016 19th international conference on computer and information technology (ICCIT)* (pp. 425–430). https://doi.org/10.1109/ICCITECHN.2016.7860236

Akhter, M. P., Jiangbin, Z., Naqvi, I. R., Abdelmajeed, M., Mehmood, A., & Sadiq, M. T. (2020). Document-level text classification using single-layer multisize filters convolutional neural network. *IEEE Access, 8*, 42689–42707. https://doi.org/10.1109/ACCESS.2020.2976744

Alhaj, Y. A., Xiang, J., Zhao, D., Al-Qaness, M. A. A., Elaziz, M. A., & Dahou, A. (2019). A study of the effects of stemming strategies on arabic document classification. *IEEE Access, 7*, 32664–32671. https://doi.org/10.1109/ACCESS.2019.2903331

Alhawarat, M., & Aseeri, A. O. (2020). A superior arabic text categorization deep model (satcdm). *IEEE Access, 8*, 24653–24661. https://doi.org/10.1109/ACCESS.2020.2970504

Ambalavanan, A. K., & Devarakonda, M. V. (2020). Using the contextual language model bert for multi-criteria classification of scientific articles. *Journal of Biomedical Informatics, 112*, Article 103578. url:https://www.sciencedirect.com/science/article/pii/S1532046420302069. 10.1016/j.jbi.2020.103578.

Bahassine, S., Madani, A., & Mohamed. (2017). Arabic text classification using new stemmer for feature selection and decision trees. *Journal of Engineering Science and Technology, 12*, 1475–1487.

Behera, R. K., Jena, M., Rath, S. K., & Misra, S. (2021). Co-lstm: Convolutional lstm model for sentiment analysis in social big data. *Information Processing and Management, 58*, Article 102435. https://doi.org/10.1016/j.ipm.2020.102435. url:https://www.sciencedirect.com/science/article/pii/S0306457320309286.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5*, 135–146. https://doi.org/10.1162/tacl_a_00051

Catanzaro, B., Sundaram, N., & Keutzer, K. (2008). Fast support vector machine training and classification on graphics processors. In Machine learning, proceedings of the twenty-fifth international conference (ICML 2008), Helsinki, Finland, June 5–9, 2008 (pp. 104–111). ACM volume 307 of ACM International Conference Proceeding Series. url:https://doi.org/10.1145/1390156.1390170. doi:10.1145/1390156.1390170.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology, 2*, 27:1–27:27. Software available at url:http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Chiu, B., Korhonen, A., & Pyysalo, S. (2016). Intrinsic evaluation of word vectors fails to predict extrinsic performance. In Proceedings of the 1st workshop on evaluating vector-space representations for NLP (pp. 1–6). Berlin, Germany: Association for Computational Linguistics. url:https://www.aclweb.org/anthology/W16-2501. doi:10.18653/v1/W16-2501.

Chung, J., Gülçehre, Ç., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555. url:http://arxiv.org/abs/1412.3555. arXiv:1412.3555.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*, 37–46. https://doi.org/10.1177/001316446002000104. url:https://doi.org/10.1177/001316446002000104.

Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2017). Very deep convolutional networks for text classification. In Proceedings of the 15th conference of the european chapter of the association for computational linguistics: Volume 1, Long Papers (pp. 1107–1116). Valencia, Spain: Association for Computational Linguistics. url:https://www.aclweb.org/anthology/E17-1104.

Dang, H.T., & Palmer, M. (2002). Combining contextual features for word sense disambiguation. In Proceedings of the ACL-02 workshop on word sense disambiguation: recent successes and future directions (pp. 88–94). Association for Computational Linguistics. url:https://www.aclweb.org/anthology/W02-0813. doi:10.3115/1118675.1118688.

Dash, N. S., & Ramamoorthy, L. (2019). Process of text corpus generation. In *Utility and application of language corpora* (pp. 17–34). Singapore: Springer. https://doi.org/10.1007/978-981-13-1801-6_2.

Deng, X., Li, Y., Weng, J., & Zhang, J. (2019). Feature selection for text classification: A review. *Multimedia Tools Applications, 78*, 3797–3816. https://doi.org/10.1007/s11042-018-6083-5. url:https://doi.org/10.1007/s11042-018-6083-5.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423. url:https://www.aclweb.org/anthology/N19-1423.

Dhar, A., Mukherjee, H., Obaidullah, S. M., Santosh, K. C., Dash, N.S., & Roy, K. (2020). Web text categorization: A lstm-rnn approach. In ICICC 2019: Intelligent computing and communication (pp. 281–290). Springer, Singapore. vol. 1034. doi:10.1007/978-981-15-1084-7_27.

Enrıquez, F., Troyano, J. A., & López-Solaz, T. (2016). An approach to the use of word embeddings in an opinion classification task. *Expert Systems with Applications, 66*, 1–6. https://doi.org/10.1016/j.eswa.2016.09.005. url:https://www.sciencedirect.com/science/article/pii/S0957417416304833.

Gambino, G., & Pirrone, R. (2019). Investigating embeddings for sentiment analysis in italian. In Proceedings of the 3rd workshop on natural language for artificial intelligence co-located with the 18th international conference of the italian association for artificial intelligence (AIIA 2019), Rende, Italy, November 19th-22nd, 2019. CEUR-WS.org volume 2521 of CEUR Workshop Proceedings. url:http://ceur-ws.org/Vol-2521/paper-03.pdf.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., & Mikolov, T. (2018). Learning word vectors for 157 languages. In Proceedings of the international conference on language resources and evaluation (LREC 2018). url:https://www.aclweb.org/anthology/L18-1550.

Grießhaber, D., Vu, N. T., & Maucher, J. (2020). Low-resource text classification using domain-adversarial learning. *Computer Speech & Language, 62*, Article 101056. https://doi.org/10.1016/j.csl.2019.101056. url:https://www.sciencedirect.com/science/article/pii/S0885230819303006.

Hashemi, S., Yang, Y., Mirzamomen, Z., & Kangavari, M. (2009). Adapted one-versus-all decision trees for data stream classification. *IEEE Transactions on Knowledge and Data Engineering, 21*, 624–637. https://doi.org/10.1109/TKDE.2008.181

Hashmi, S. U., & Bansal, A. (2019). Information extraction and visualization of unstructured textual data. In *2019 IEEE 13th international conference on semantic computing (ICSC)* (pp. 142–145). https://doi.org/10.1109/ICOSC.2019.8665534

Hearst, M. A. (1998). Support vector machines. *IEEE Intelligent Systems, 13*, 18–28. https://doi.org/10.1109/5254.708428. url:https://doi.org/10.1109/5254.708428.

He, J., Wang, L., Liu, L., Feng, J., & Wu, H. (2019). Long document classification from local word glimpses via recurrent attention learning. *IEEE Access, 7*, 40707–40718. https://doi.org/10.1109/ACCESS.2019.2907992

He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *nternational Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6*, 107–116. https://doi.org/10.1142/S0218488598000094. url:https://doi.org/10.1142/S0218488598000094.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735. url:https://doi.org/10.1162/neco.1997.9.8.1735 arXiv:https://doi.org/10.1162/neco.1997.9.8.1735.

Hossain, M. R., & Hoque, M. M. (2018). Automatic bengali document categorization based on word embedding and statistical learning approaches. In 2018 International conference on computer, communication, chemical, material and electronic engineering (IC4ME2) (pp. 1–6). doi:10.1109/IC4ME2.2018.8465632.

Hossain, M. R., & Hoque, M. M. (2020). Towards bengali word embedding: Corpus creation, intrinsic and extrinsic evaluations. In Proceedings of the 17th international conference on natural language processing (pp. 453–459). IIT Patna, India, 2020 NLP Association of India (NLPAI): Preprints 2020. url:https://www.preprints.org/manuscript/202012.0600/v1. doi:10.20944/preprints202012.0600.v1.

Hossain, M. R., & Hoque, M. M. (2021). Semantic meaning based bengali web text categorization using deep convolutional and recurrent neural networks (dcrnns). In Proc. ICIoTCT (pp. 494–505). India, IIT Patna. doi: 10.1007/978-3-030-76736-5_45.

Hossain, M. R., & Hoque, M. M. (2019). Automatic bengali document categorization based on deep convolution nets. In *Emerging Research in Computing, Information, Communication and Applications. Advances in Intelligent Systems and Computing* (vol. 882, pp. 513–525). Singapore: Springer. https://doi.org/10.1007/978-981-13-5953-8_43.

Hossain, M. R., Hoque, M. M., & Sarker, I. H. (2021). Text classification using convolution neural networks with fasttext embedding. In A. Abraham, T. Hanne, O. Castillo, N. Gandhi, T. Nogueira Rios, & T.-P. Hong (Eds.), *Hybrid intelligent systems* (pp. 103–113). Cham: Springer International Publishing.

Johnson, R., & Zhang, T. (2017). Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: long papers)* (pp. 562–570). Vancouver, Canada: Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1052. url:https://www.aclweb.org/anthology/P17-1052.

Kabir, F., Siddique, S., Kotwal, M. R. A., & Huda, M. N. (2015). Bangla text document categorization using stochastic gradient descent (sgd) classifier. In *2015 International conference on cognitive computing and information processing(CCIP)* (pp. 1–4). https://doi.org/10.1109/CCIP.2015.7100687

Kaiming, H., Xiangyu, Z., Shaoqing, R., & Jian, S. (2015). Deep residual learning for image recognition. CoRR, abs/1512.03385. url:http://arxiv.org/abs/1512.03385. arXiv:1512.03385.

Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P.T.P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. CoRR, abs/1609.04836. url:http://arxiv.org/abs/1609.04836. arXiv:1609.04836.

Khan, N. H., & Adnan, A. (2018). Urdu optical character recognition systems: Present contributions and future directions. *IEEE Access, 6*, 46019–46046. https://doi.org/10.1109/ACCESS.2018.2865532

Khan, M., Jan, B., & Farman, H. (2019). Deep learning: Convergence to big data analytics. In *SpringerBriefs in computer science* (pp. 31–42). Singapore: Springer. https://doi.org/10.1007/978-981-13-3459-7.

Khatun, A., Rahman, A., Islam, M. S., & Marium-E-Jannat. (2019). Authorship attribution in bangla literature using character-level cnn. In *2019 22nd International conference on computer and information technology (ICCIT)* (pp. 1–5). https://doi.org/10.1109/ICCIT48885.2019.9038560

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1746–1751). Doha, Qatar: Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1181. url:https://www.aclweb.org/anthology/D14-1181.

Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., Gerber, M. S., & Barnes, L. E. (2017). Hdltex: Hierarchical deep learning for text classification. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)* (pp. 364–371). https://doi.org/10.1109/ICMLA.2017.0-134

Kumari, M., Jain, A., & Bhatia, A. (2016). Synonyms based term weighting scheme: An extension to tf.idf. *Procedia Computer Science, 89*, 555–561. https://doi.org/10.1016/j.procs.2016.06.093. url:https://www.sciencedirect.com/science/article/pii/S1877050916311589.

Lee, J. Y., & Dernoncourt, F. (2016). Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies* (pp. 515–520). San Diego, California: Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1062. url:https://www.aclweb.org/anthology/N16-1062.

Liebeskind, C., Kotlerman, L., & Dagan, I. (2015). Text categorization from category name in an industry-motivated scenario. *Language Resources and Evaluation, 49*, 227–261. https://doi.org/10.1007/s10579-015-9298-3

Li, X., & Roth, D. (2002). Learning question classifiers. In *COLING 2002: The 19th international conference on computational linguistics*. url:https://www.aclweb.org/anthology/C02-1150.

Mei, J.-P., Wang, Y., Chen, L., & Miao, C. (2017). Large scale document categorization with fuzzy clustering. *IEEE Transactions on Fuzzy Systems, 25*, 1239–1251. https://doi.org/10.1109/TFUZZ.2016.2604009

Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013). Efficient estimation of word representations in vector space. CoRR, abs/1301.3781.

Mirończuk, M. M., & Protasiewicz, J. (2018). A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications, 106*, 36–54. https://doi.org/10.1016/j.eswa.2018.03.058. url:https://www.sciencedirect.com/science/article/pii/S095741741830215X.

Moirangthem, D. S., & Lee, M. (2021a). Hierarchical and lateral multiple timescales gated recurrent units with pre-trained encoder for long text classification. *Expert Systems with Applications, 165*, Article 113898. https://doi.org/10.1016/j.eswa.2020.113898. url:https://www.sciencedirect.com/science/article/pii/S095741742030693X.

Moirangthem, D. S., & Lee, M. (2021b). Hierarchical and lateral multiple timescales gated recurrent units with pre-trained encoder for long text classification. *Expert Systems with Applications, 165*, Article 113898. https://doi.org/10.1016/j.eswa.2020.113898. url:https://www.sciencedirect.com/science/article/pii/S095741742030693X.

Mucherino, A., authorPetraq J. Papajorgji, & Pardalos, P. M. (2009). k-nearest neighbor classification. In Data Mining in Agriculture (pp. 83–106). New York, NY: Springer, New York. url:https://doi.org/10.1007/978-0-387-88615-2_4. doi:10.1007/978-0-387-88615-2_4.

Nikolentzos, G., Meladianos, P., Rousseau, F., Stavrakas, Y., & Vazirgiannis, M. (2017). Multivariate gaussian document representation from word embeddings for text categorization. In Proceedings of the 15th conference of the european chapter of the association for computational linguistics: volume 2, short papers (pp. 450–455). Valencia, Spain: Association for Computational Linguistics. url:https://www.aclweb.org/anthology/E17-2072.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162. url:https://www.aclweb.org/anthology/D14-1162.

Phani, S., Lahiri, S., & Biswas, A. (2017). A supervised learning approach for authorship attribution of bengali literary texts. *ACM Transactions on Asian and Low-Resource Language Information Processing, 16*. https://doi.org/10.1145/3099473. url:https://doi.org/10.1145/3099473.

Rahman, M. A., & Dey, E. K. (2018). Datasets for aspect-based sentiment analysis in bangla and its baseline evaluation. *Data, 03*. https://doi.org/10.3390/data3020015

Rebecca, P. (2006). Measuring agreement on set-valued items (masi) for semantic and pragmatic annotation. In *Proceedings of the fifth international conference on language resources and evaluation (LREC'06)* (pp. 831–836). European Language Resources Association (ELRA). url:http://www.lrec-conf.org/proceedings/lrec2006/pdf/636_pdf.pdf.

Řeh◦uřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In Proceedings of LREC 2010 workshop new challenges for NLP frameworks (pp. 46–50). Valletta, Malta: University of Malta. url:http://is.muni.cz/publication/884893/en.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. CoRR, abs/1609.04747. url:http://arxiv.org/abs/1609.04747. arXiv:1609.04747.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536. https://doi.org/10.1038/323533a0

Saad, S. E., & Yang, J. (2019). Twitter sentiment analysis based on ordinal regression. *IEEE Access, 7*, 163677–163685. https://doi.org/10.1109/ACCESS.2019.2952127

Sakalle, A., Tomar, P., Bhardwaj, H., Acharya, D., & Bhardwaj, A. (2021). A lstm based deep learning network for recognizing emotions using wireless brainwave driven system. *Expert Systems with Applications, 173*, Article 114516. https://doi.org/10.1016/j.eswa.2020.114516. url:https://www.sciencedirect.com/science/article/pii/S095741742031160X.

Sarker, I. H., Furhad, M. H., & Nowrozy, R. (2021). Ai-driven cybersecurity: an overview, security intelligence modeling and research directions. *SN Computer Science, 2*, 1–18. https://doi.org/10.1007/s42979-021-00557-0

Shriberg, E., Dhillon, R., Bhagat, S., Ang, J., & Carvey, H. (2004). The ICSI meeting recorder dialog act (MRDA) corpus. In *Proceedings of the 5th SIGdial workshop on discourse and dialogue at HLT-NAACL 2004* (pp. 97–100). Cambridge, Massachusetts, USA: Association for Computational Linguistics. url:https://www.aclweb.org/anthology/W04-2319.

Stehman, V. S. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment, 62*, 77–89. https://doi.org/10.1016/S0034-4257(97)00083-7

Tang, D., Qin, B., & Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1422–1432). Lisbon, Portugal: Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1167. url:https://www.aclweb.org/anthology/D15-1167.

Tetko, I. V., Livingstone, D. J., & Luik, A. I. (1995). Neural network studies. 1. Comparison of overfitting and overtraining. *Journal of Chemical Information and Computer Sciences, 35*, 826–833. doi:110.1021/ci00027a006.

Wen, Z., Shi, J., Li, Q., He, B., & Chen, J. (2018). Thundersvm: A fast svm library on gpus and cpus. *Journal of Machine Learning Research, 19*, 1–5. url:http://jmlr.org/papers/v19/17-740.html.

Wu, D., Zhang, M., Shen, C., Huang, Z., & Gu, M. (2020). Btm and glove similarity linear fusion-based short text clustering algorithm for microblog hot topic discovery. *IEEE Access, 8*, 32215–32225. https://doi.org/10.1109/ACCESS.2020.2973430

Xiao, Y., Liu, B., Yin, J., & Hao, Z. (2017). A multiple-instance stream learning framework for adaptive document categorization. *A Knowledge-Based System, 120*, 198–210. url:https://doi.org/10.1016/j.knosys.2017.01.001.

Xu, K., Feng, Y., Huang, S., & Zhao, D. (2015). Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 536–540). Lisbon, Portugal: Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1062. url:https://www.aclweb.org/anthology/D15-1062.

Zhang, X., Zhang, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Proceedings of the 28th international conference on neural information processing systems (pp. 649–657). Cambridge, MA, USA: MIT Press Vol. 1. doi:10.5555/2969239.2969312.

Zhou, D.-X. (2020). Theory of deep convolutional neural networks: Downsampling. *Neural Networks, 124*, 319–327. https://doi.org/10.1016/j.neunet.2020.01.018. url: https://www.sciencedirect.com/science/article/pii/S0893608020300204.

Zia, T., Akhter, M. P., & Abbas, Q. (2015). Comparative study of feature selection approaches for urdu text categorization. Malaysian. *Journal of Computer Science, 28*, 93–109. url:https://ejournal.um.edu.my/index.php/MJCS/article/view/6857.