Applied Intelligence
https://doi.org/10.1007/s10489-022-04221-9

# *TextConvoNet*: a convolutional neural network based architecture for text classification

**Sanskar Soni**[1] · **Satyendra Singh Chouhan**[1] · **Santosh Singh Rathore**[2]

Accepted: 28 September 2022


**Abstract**

This paper presents, *TextConvoNet*, a novel Convolutional Neural Network (CNN) based architecture for binary and multi-class text classification problems. Most of the existing CNN-based models use one-dimensional convolving filters, where each filter specializes in extracting *n-grams* features of a particular input word embeddings (Sentence Matrix). These features can be termed as intra-sentence *n-gram* features. To the best of our knowledge, all the existing CNN models for text classification are based on the aforementioned concept. The presented *TextConvoNet* not only extracts the intra-sentence *n-gram* features but also captures the inter-sentence *n-gram* features in input text data. It uses an alternative approach for input matrix representation and applies a two-dimensional multi-scale convolutional operation on the input. We perform an experimental study on five binary and multi-class classification datasets and evaluate the performance of the *TextConvoNet* for text classification. The results are evaluated using eight performance measures, accuracy, precision, recall, f1-score, specificity, gmean1, gmean2, and Mathews correlation coefficient (MCC). Furthermore, we extensively compared presented *TextConvoNet* with machine learning, deep learning, and attention-based models. The experimental results evidenced that the presented *TextConvoNet* outperformed and yielded better performance than the other used models for text classification purposes.

**Keywords** Text classification · Convolution neural network (CNN) · Multi-dimensional convolution · Deep learning

## 1 Introduction

Natural language processing (NLP) involves text processing and extracting the key patterns from the natural/human languages. It involves various tasks that rely on various statistics and data-driven computation techniques. One of the important tasks in NLP is text classification. It is a classical problem where the prime objective is to classify (assign labels or tags) the textual contents [1]. Textual contents can either be sentences, paragraphs, or queries [1, 2]. There are many real-world applications of text classification, such as sentiment analysis [3], news classification [4], intent classification [5], spam detection [6], and so on.

Text classification can be done by manual labeling of the textual data. However, with the exponential growth of text data in the industry and over the Internet, automated text categorization has become very important. Automated text classification approaches can be broadly classified into Rule-based, Data-Driven based (Machine Learning/Deep Learning-based approaches), and Hybrid approaches. Rule-based approaches classify text into different categories using a set of pre-defined rules. However, it requires complete domain knowledge [7, 8]. Alternatively, machine learning-based approaches have proven to be significantly effective in recent years. All the machine learning approaches work in two stages: first, they extract some handcrafted features from the text. Next, these features are fed into a machine learning model. A bag of words, *n-grams* based model, term frequency, and inverse document frequency (TF-IDF) and their extensions have been popularly used for extracting the handcrafted features.

✉ Satyendra Singh Chouhan
sschouhan.cse@mnit.ac.in

Sanskar Soni
2018ucp1265@mnit.ac.in

Santosh Singh Rathore
santoshs@iiitm.ac.in

[1] Department of Computer Science and Engineering, MNIT Jaipur, Jaipur, 302017, India

[2] Department of Computer Science and Engineering, ABV-IIITM Gwalior, Gwalior, 474015, India

Published online: 22 October 2022

For the second stage, many classical machine learning algorithms such as Support Vector Machine (SVM), Decision Tree (DT), Conditional Probability-based such as Naïve Bayes, and other Ensemble-based approaches have been used [9, 10].

Recently, some deep learning methods, specifically Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN), have shown significant results in text classification [11–16]. CNN-based models are trained to recognize patterns in text automatically, such as key phrases. Most CNN-based models utilize one-dimensional (1-D) convolution followed by a one-dimensional pooling operation (average or max) to extract a feature vector from the input word embeddings. This feature vector is fed into the classification layer as an input for classification purposes. Input word embedding is a word matrix where each row represents a word vector. Therefore, one-dimensional (1-D) convolution extracts *n-gram* based features by performing convolution operation on two or more two-word vectors at a time.

However, improving text classification results by utilizing the *n-gram* features in between different sentences using convolution operation remains an open research question for all researchers. Furthermore, the input matrix structure remains a point to ponder, which could be revamped to apply multidimensional convolution. This paper presents, *TextConvoNet*, a new CNN-based architecture for text classification. In contrast to the existing works, the proposed architecture uses a 2-dimensional convolutional filter to extract the intra-sentence and inter-sentence *n-gram* features from text data. First, it represents the text data as a paragraph-level (multi-sentence) embedding matrix, which helps apply 2-dimensional convolutional filters. After that, multiple convolutional filters are applied to the extracted features. The resultant features are concatenated and fed into the classification layer. To evaluate the performance of the presented *TextConvoNet*, we perform a thorough experimental analysis on five different benchmarked binary-class and multi-class classification datasets. Evaluation of the *TextConvoNet* is based on eight performance measures: accuracy, precision, recall, f1-score, specificity, G-means, and MCC. Furthermore, we compare the performance of the *TextConvoNet* with state-of-the-art classification models, including attention-based models, BERT, and deep-learning-based models.

### 1.1 Contributions

The main contributions of the presented work are as follows.

1. This work presents *TextConvoNet*, a CNN-based architecture to represent input text data as a multidimensional word embedding. The presented architecture extracts both intra-sentence and inter-sentence features of the text.
2. The presented architecture is comprehensively evaluated on five benchmarked text datasets, including binary-class and multi-class datasets. This analysis helps in generalizing the findings of the presented work.
3. An extensive comparison of the presented *TextConvoNet* with existing machine learning models, deep learning-based models, attention models, and state-of-the-art CNN-based models is performed to validate the performance of the presented *TextConvoNet*.

The rest of the article is organized as follows. Section 2 discusses the literature review. Section 3 presents information on text classification using CNN models and provides details of the presented *TextConvoNet* architecture. Section 4 provides the details of the experimental setup and analysis. It includes details of used datasets followed by performance measures and implementation details. Section 5 presents experimental results and comparison results of the *TextConvoNet* with the state-of-the-art text classification models. Section 6 discusses the main findings and limitations of the presented work. Section 7 concludes the paper along with the details of the future directions of the presented work.

## 2 Literature review

Text classification is one of the important tasks in Natural Language Processing (NLP). There have been many data-driven-based approaches suggested for text classification. Recently, deep learning-based approaches have emerged and performed significantly well in text classification. This section discusses some of the relevant deep learning-based models suggested for text classification.

Two neural networks have been prevalent in NLP problems: Long Short Term Memory (LSTM) and Convolutional Neural Network (CNN). LSTM can extract current information and remember the past data points in sequence [13, 14, 17, 18]. However, LSTM-based models have very high training time because it is characterized as each token per step. Recently, attention-based networks have proven effective for Natural Langauge Processing [19, 20]. However, LSTM with attention networks introduces an additional computation burden. It is because of the exponential function and normalized alignment score computation of all the words in the text [21].

One of the early CNN-based models was Dynamic CNN (DCNN) for text classification [22]. It used dynamic max-pooling, where the first layer of DCNN makes a sentence matrix using word embeddings. Then it has a convolutional architecture that uses repeated convolutional layers with dynamic k-max-pooling to extract feature maps on the sentence. These feature maps are capable enough of capturing the short and long-range relationship between the words. Later, Kim [23] gave the simplest CNN-based model for text classification, which has become the benchmark architecture for many recent models for text classification. Kim's model adapted a single layer of convolution on top of the input word vectors obtained from an unsupervised neural language model (word2vec). It has been evident to improve upon the state-of-the-art binary and multi-class text classification problems. Recently, some attempts have been made to enhance the architectures of CNN-based models. Liu et al. [15, 24–26]. Instead of using pre-trained low-dimensional word vectors as an input to CNN, authors [26] directly applied CNNs to high-dimensional text data to learn the embeddings of small text regions for the classification. Most existing works have shown that convolutions of sizes 2, 3, 4, or 5 give significant results in text classification.

Some of the research explored the impact of model performance while using word embeddings and CNN architectures. Encouraged by VGG [27] and ResNets [28], Conneau et al. [29] proposed a Very Deep CNN (VDCNN) model for text processing. It applies CNN directly at the character level and uses small convolutions and pooling functions. The research exhibited that the performance of VDCNN improves with the increase in depth. In another work, Le et al. [30] showed that deep architectures could outperform shallow architectures where text data is represented as a sequence of characters. Later on, in another research, Squeezed-VDCNN suggested [15] which improved VDCNN to work on mobile platforms. However, a basic shallow-and-wide network outperforms deep models, such as DenseNet [31], with word embeddings as inputs. There were some application-specific models have been proposed for text classification. Some researchers used deep learning-based models for short text classification [32, 33]. In some research, language-specific (for example, Arabic and Urdu) deep learning models have been proposed [34, 35]. Moreover, the author used combination of CNN and bi-LSTM for medical text classification [36]. In another work, authors used CNN with a hierarchical encoder for defect text classification [37].

To the best of our knowledge, above discussed, all the CNN-based networks extract the *n-gram* based feature using varied sizes of kernels/filters. In light of the above works, we present a novel CNN-based architecture that extracts intra-sentence n-gram features and captures the inter-sentence n-gram features.

## 3 Proposed *TextConvoNet* architecture

This section first discusses the existing CNN-based approach of text classification with an example (Section 3.1). Next, the section presents details of the proposed *TextConvoNet* architecture (Section 3.2). The mathematical symbols used in this section are given in Table 1.

### 3.1 Text classification using existing CNN models (*background*)

Text classification problems can be formally defined as follows [38].

**Definition 1** Given a text dataset $\mathcal{T}$ consisting of labelled text articles. Depending on a particular NLP task, text articles have a particular label/class $l \in L$. In case of binary-class classification, there are two labels for the text dataset. A text article $te \in \mathcal{T}$ consists of sentences and words. Let us say, text article $te_i$ contain $m$ sentences $s_1, \ldots, s_m$ and sentence $s_j$ $(0 \leq j \leq m)$ contain $n$ words.

The objective of text classification is to learn a model $M$ that can correctly classify any new text articles $t_{new}$ into label $l \in L$.

Kim et al. [23] presented a simple and effective architecture for text classification. From now on, we call

**Table 1** Symbols used in proposed framework

| Symbols | Description |
|---|---|
| $\mathcal{T}$ | Text dataset |
| $l$ | A label/class such that $l \in L$ (set of classes) |
| $te \in \mathcal{T}$ | A text data (instance) in the dateset |
| $s_j$ | A $j^{th}$ sentence in a paragraph |
| $we$ | Word embedding (vector) of size $z$ |
| $n$ | Number of words in a sentence |
| $x$ | Window size of $x$ words |
| $E_w$ | Embedding matrix |
| $c_p$ | A value in feature map (1-D vector) that obtained after convolution operation at location $p$ |
| $C$ | 1-D feature map vector |
| $b \in \mathbb{R}$ | Bias value |
| $k$ | One dimensional kernel size |
| $f(.)$ | Any activation function |
| $H$ | Two dimensional kernel |
| $r_{i,j}$ | A value in feature map (2-D matrix) at location $(i, j)$ |
| $\sigma$ | Sigmoid activation fuction |
| $\hat{y}$ | Predicted class |
| BCE | Binary cross entropy |
| CCE | Categorical cross entropy |

Kim's CNN model throughout the paper for simplicity. This presented architecture served as a guiding light and basis for many CNN-based architectures for text classification. Many recent architectures internally use this model [39–42]. In Kim's CNN model, sentences are matched to the embedding vectors made available as an input matrix to the model. It uses only a single layer of convolution with word vectors on top obtained from an existing pre-trained model, with kernel sizes 3, 4, and 5. The resultant feature maps are further processed using a max-pooling layer to distill or summarize the extracted features, subsequently sent to a fully connected layer. Figure 1 shows a simple example of text classification using CNN-based Kim's model.

As shown in Fig. 1, the input to the model is a sentence represented as a matrix. Each row of the matrix is a vector that represents a word. 1D convolution is performed onto the matrix with the kernel size being 3 along with 4 and 5. Max-Pooling is performed upon the filter maps, which are further concatenated and sent to the last fully connected layer for classification purpose. Formally, the sentence modeling is as follows.

**Sentence modelling** In each sentence, $we_p \in \mathbb{R}_z$ denotes the word embedding (a vector) for the $p^{th}$ word in the sentence, where $z$ is the word embedding dimension. Suppose that a sentence has $n$ words, the sentence can now be represented as an embedding matrix $E_{we} \in \mathbb{R}^{n \times z}$. So we can refer to it as a word matrix where every row denotes the vectors for a particular word of the sentence. Let $we_{p:p+q}$ represents the concatenation of vectors $we_p, we_{p+1}, \ldots, we_q$. The convolution operation is performed on this input embedding layer. It involves a filter $k \in \mathbb{R}^{x,z}$ that applies to a window of $x$ words to produce a new feature. For example, a feature $c_p$ is generated using the window of words $we_{p:p+x-1}$ by (1).

$$c_p = f(we_{(p:p+x-1)}.k + b) \tag{1}$$

Here, $b \in \mathbb{R}$ and $f$ denotes the bias and non-linear activation function respectively. The filter (kernel) $k$ applies to all possible windows using the same weights to create the feature map (1-D vector).

$$C = [c_1, c_2, \ldots, c_{n-x+1}] \tag{2}$$

### 3.2 Proposed *TextConvoNet* architecture

The proposed *TextConvoNet* architecture finds *n-gram* features between words of the different sentences and the intra-sentence *n-gram* feature. It is because, in the text data, having multiple sentences may have useful *n-gram* features. This could only be possible by using the paragraph matrix instead of the sentence matrix and applying 2-D filters. Thus, the motivation and the research question are to explore *"if combining n-gram-based inter-sentence characteristics with n-gram-based intra-sentence features is beneficial or not"*. In real-world scenarios, the paragraphs are stringed together in a very complex manner, making it very difficult for any model to come up with correct labeling, whether it be any sentiment or a news category. Therefore, there may be instances when the model cannot extract the inter-sentence features and hence fails to come up with a suitable result. Taking inspiration from the above shortcoming, we present an alternative input structure for the model and propose a novel CNN model using the alternative input structure and employing 2-D Convolution [43].

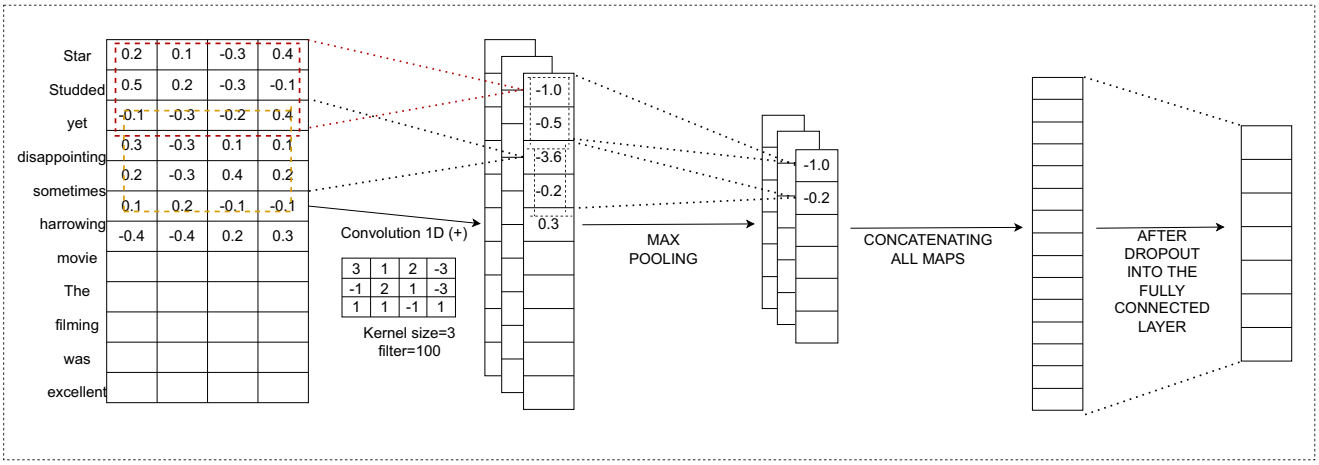


**Fig. 1** Example: Text classification using CNN [23]

### 3.2.1 Input representation

We propose a new input representation for text data. In existing works, each sentence is represented as two-dimensional matrix where each row represents an embedding vector for a word. Whereas in our model, the input is represented as three-dimensional matrix. In this representation, each row depicts each sentence of a paragraph, with each cell as a single word and the $3^{rd}$ dimension as the embeddings or the word vectors. This representation may be termed as a sentence matrix. The formal description of our input structure is mentioned below. For each sentence in a paragraph, let $E_{w_i} \in \mathbb{R}^z$ represents the word embedding for the $i^{th}$ word in the sentence, where $z$ is the dimension of the word embedding. Given that a paragraph has $m$ sentences and $n$ words in each sentence, the paragraph can be represented as an embedding matrix $W$ of size $(m, n, z)$ such that $W \in \mathbb{R}^{m \times n \times z}$.

The overall architecture of our proposed model, *TextConvoNet*, is shown in Fig. 2. The presented *TextConvoNet* model uses an alternate input structure of the paragraph, using 2D convolution instead of 1D convolution and differing kernel sizes. *TextConvoNet* sends the input matrix into 4 parallel pathways of Convolution layers. The first two layers (intra-sentence layers) with 32 filters each and kernel sizes of $(1 \times 2$ and $1 \times 3)$, respectively, are concatenated and have the role of extricating the intra-sentence n-gram features. The other two layers (inter-sentence) with 32 filters each and kernel sizes of $(2 \times 1$ and $2 \times 2)$ concatenated together have the sole purpose of drawing out the inter-sentence n-gram features. These two intra-sentence and inter-sentence layers are further concatenated and fed into the fully connected layer consisting of 64 neurons and subsequently perform the relevant classification task. A detailed explanation of the architecture is given as follows.

### 3.2.2 Convolution layer

This layer applies filters to the input to create feature maps and condense out the input's detected features. It is a process where we take a small matrix of numbers (called kernel or filter) and pass it over the paragraph matrix and transform it based on the values from filter. Let $E_w(m, n)$ be an input paragraph matrix of size $m \times n$ and $H$ is a two dimensional with kernel size of $(2g + 1, 2d + 1)$, where $g$ and $d$

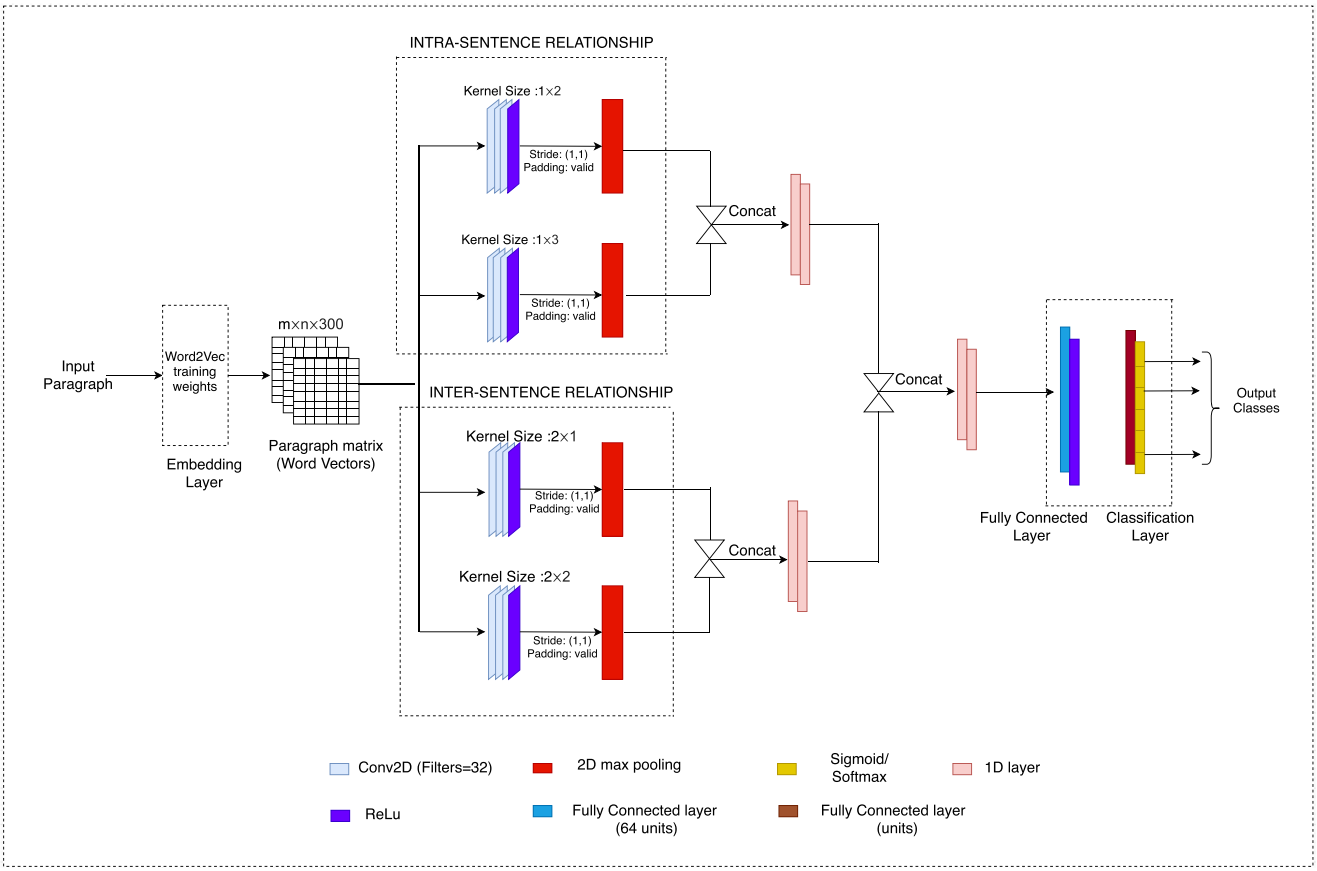


**Fig. 2** Proposed *TextConvoNet* Architecture

are constants. The outcome of the convolutional layer is represented by (3).

$$r_{i,j} = \sum_{u=-g}^{g} \left( \sum_{v=-d}^{d} H[u, v] F[i - u, j - v] \right) \quad (3)$$

Here, $r_{i,j}$ is the value at location $(i, j)$ in the feature map.

#### 3.2.3 ReLu activation layer

The purpose of the ReLu activation layer after each convolution layer is to normalize output. This layer also aids the model to learn something complex and complicated with a reduced possibility of vanishing gradient and cheap computation costs. The activation function for ReLu is given in (4). Here $r_{i,j}$ is the input to the ReLu function.

$$f(r_{i,j}) = max(0, r_{i,j}) \quad (4)$$

#### 3.2.4 Classification

The feature maps generated by using different kernel sizes are concatenated and fed into the fully connected layer. The fully connected layer is a multilayer perceptron connected to all the activations from the previous layers. The activation of these neurons is calculated by matrix multiplication of its weights added by an offset value. A dropout layer is also used that randomly activates or deactivates (makes them 0) the outgoing edges of hidden units at each update of the training phase, which helps to reduce overfitting. In the end, the classification layer performs classification based on the attributes extricated by the previous layers. It is a traditional ANN layer with softmax or sigmoid as the activation function.

#### 3.2.5 Loss function

For binary-class text classification task, *TextConvoNet* is trained by minimizing the binary-cross entropy (5) over a sigmoid activation function. For the task of multi-class classification, *TextConvoNet* is trained by minimizing the categorical-cross entropy (6) over a softmax activation function. The above loss functions can be formulated as

$$BCE = -\frac{1}{m} \sum_{i}^{m} \sum_{j}^{c} y_{ij} log(\sigma(\hat{y}_{ij}))) - (1 - y_{ij}) log(1 - \sigma(\hat{y}_{ij}))) \quad (5)$$

$$CCE = -\frac{1}{m} \sum_{i}^{m} \sum_{j}^{c} y_{ij} log \left( \frac{e^{\hat{y}_{ij}}}{\sum_{r=1}^{c} e^{\hat{y}_{ij}}} \right) \quad (6)$$

Here $i$ is the index of a training instance, $j$ is the index of a label (class), $\hat{y}_{ij}$ is output of the final fully connected layer, and $y_{ij}$ is the ground truth (actual value) of $i^{th}$ training sample of the $j^{th}$ class.

### 3.3 Analysis of *TextConvoNet*

Almost all real-life conversations, reviews, and remarks are generally very long and complex and thus convey a different perspective in each line. However, only a single deep-rooted sentiment is attached to the whole paragraph. To uphold the semantics, the paragraph is converted into paragraph-level sentence embedding without any preprocessing of text. The embedding matrix is then sent into 4 lateral pathways subdivided into the intra-sentence (kernel sizes $1 \times 2$, $1 \times 3$) layer and the inter-sentence layer (kernel sizes $2 \times 1$, $2 \times 2$) with 32 filters in every layer. These hyperparameters were selected through the *GridSearchCV* method from the plethora of other suitable hyperparameter choices. The results also complemented our thinking/approach of selecting small window sizes to capture every minute detail. Similarly, using the *GridSearchCV* method, the learning rate was chosen to be 0.01 and the number of neurons to be 64 for the final fully connected layer. The convolutional layers were limited to four only, as increased layers led to overfitting.

### 3.4 Variants of *TextConvoNet*

We have created various variants of the proposed model to develop an effective text classification framework. Figure 2 shows the baseline model. However, it might be interesting to see whether increasing the number of *n-gram* based kernels, and inter-sentence kernels will improve the model's efficacy. Thus, we have extended the baseline to create two versions of *TextConvoNet*: *TextConvoNet_4* and *TextConvoNet_6*.

- *TextConvoNet_4*: The base/parent model with 4 convolution layers (with different kernel sizes), 2 for extracting out the intra-sentence *n-gram* features, and the other 2 for extricating the *n-gram* based inter-sentence attributes.
- *TextConvoNet_6*: It is the same framework as mentioned above but extending the convolutional pathways to 6, 3 for extracting out intra-sentence *n-gram* features and other 3 for extracting inter-sentential *n-gram* features.

We have also performed modifications on various parameters of *TextConvoNet*: number of filters, dropout rate, kernel sizes, number of nodes in fully connected layer and optimizers, etc. However, the effectiveness of these modifications is experimentally validated in Section 5.

## 4 Experimental setup and analysis

This section first describes the used datasets. The model building and evaluation using the presented *TextConvoNet*

is conducted *via* an experimental analysis on various binary-class and multi-class text classification datasets. This section also discusses the used performance measures and baseline machine-learning and deep-learning-based models used for comparison.

### 4.1 Used datasets

We have performed experiments on various publicly available binary-class and multi-class text datasets. Only a subset of instances from the datasets has been included for training and testing. The details of the used datasets are given in Table 2. No additional changes have been made to the datasets, and no preprocessing has been applied to the text. For the experiments, we have used two binary-class and three multi-class datasets. The binary-class datasets are the famous SST-2 and Amazon Review dataset. The multi-class datasets consisted of Ohsumed (R8), Twitter Airline Sentiment, and the Coronavirus Tagged datasets. All the datasets are publicly available and are sourced from Kaggle. The details of the datasets are mentioned below.

- DATASET-1[1]: **Binary SST−2**: This dataset is similar to the Stanford Sentiment Treebank dataset with only positive and negative reviews. We have removed all neutral reviews from the dataset.
- DATASET-2[2]: **Amazon Review for Sentiment Analysis Dataset**: This dataset contains a few million customer reviews and the star ratings.
- DATASET-3[3]: **R8 Dataset**: This is a subset of Reuters-21578 dataset containing 8 categories for multiclass classification.
- DATASET-4[4]: **Twitter User Airline Sentiment**: The dataset contains the tweets of six different airlines as positive, negative, or neutral.
- DATASET-5[5]: **Covid Tweets**: The tweets have been pulled from Twitter followed by manual tagging as Extremely Negative, Negative, Neutral, Positive and Extremely Positive.

### 4.2 Performance evaluation measures

In the experimental evaluation, we used eight different performance measures. They are- accuracy, precision, recall, f1-score, specificity, g-means (gmean 1 and gmean 2), and MCC (Mathews Correlation Coefficient). Various previous works related to text classification tasks have used these measures to evaluate the performance of the prediction models. Therefore, we have chosen these measures due to their broad applicability. A detailed description of these performance measures is given in Appendix B, Table 9.

To assess the statistical significance of the presented *TextConvoNet_4* and *TextConvoNet_6* with other considered machine learning and deep learning techniques, we have performed the Wilcoxon Signed-Rank paired sample test. It is a non-parametric test that does not assume the normality of within-pair differences. It tests the hypothesis of whether the median difference is zero between the tested pair or not. We have used a significance level of 95% (i.e., $\alpha$=0.05) for all the tests. The framed Null Hypothesis ($H_0$) and Alternative Hypothesis ($H_a$) are as follow.

$H_0$: *No statistically significant difference is there between the paired group for value* $\alpha$*=0.05.*

$H_a$: *A statistically significant difference is present between the paired group for value of* $\alpha$*=0.05.*

The null hypothesis can be rejected when the experimental p-value has come out to be lesser than the $\alpha$ value, and it can be concluded that there is a significant difference between the paired group. If this is not the case, then automatically accept the null hypothesis.

Further, we have performed an effect size analysis using Pearson Effect $r$ measure. The effect size shows the magnitude of performance difference among the groups—the more significant the effect of size, the stronger the relationship between the two variables. It is defined by (7).

$$r = \frac{z}{\sqrt{2n}} \tag{7}$$

Where 2n = number of observations, including the cases where the difference is 0 and $z$ is the z-score value defined by (8).

$$z = \frac{|U - \mu| - 0.5}{\sigma} \tag{8}$$

According to Cohen [44], the effect size is: Low, if r≈0.1; Medium, if r≈0.3; and Large, if r≈0.5.

### 4.3 Machine learning and deep learning models used for comparison

For a comprehensive performance evaluation of the proposed *TextConvoNet*, we have used seven different machine learning techniques namely, Multinomial Naive Bayes [45], Decision Tree (DT) [46], Random Forest (RF) [47], Support Vector Classifier (SVC) [48], Gradient Boosting classifier [49], K-Nearest Neighbour (KNN) [50], and XGBoost [51]. An evaluation of the proposed *TextConvoNet* using these techniques helps establish the usability of the

[1] https://www.kaggle.com/jgggjkmf/binary-sst2-dataset

[2] https://www.kaggle.com/bittlingmayer/amazonreviews

[3] https://www.kaggle.com/weipengfei/ohr8r52

[4] https://www.kaggle.com/crowdflower/twitter-airline-sentiment

[5] https://www.kaggle.com/datatattle/covid-19-nlp-text-classification

**Table 2** Details of the used experimental datasets

| Dataset | Type of data | #Classes | Exceeding ratio | Vocabulary size | #Training set | #Testing set |
|---|---|---|---|---|---|---|
| DATASET-1 | BINARY | 2 | 0.471 | 16449 | 6911 | 1821 |
| DATASET-2 | BINARY | 2 | 0.495 | 39515 | 15000 | 5000 |
| DATASET-3 | MULTICLASS | 8 | 0.297 | 13737 | 4937 | 2175 |
| DATASET-4 | MULTICLASS | 3 | 0.563 | 16230 | 10000 | 3000 |
| DATASET-5 | MULTICLASS | 5 | 0.553 | 24742 | 6000 | 1500 |

Note: Exceeding ratio = number of samples with a length greater than the average length/number of all training samples

*TextConvoNet* and increases the generalization of the results. Since *TextConvoNet* is a convolutional neural network-based deep learning architecture, we have included some deep learning-based approaches for performance comparison. Specifically, we have implemented Kim's CNN model [23], Long Short Term Memory (LSTM) [13, 52] and VDCNN [27] based model proposed for text classification (Table 4) and compared our model with these models. We have also compared our model with other recent attention and/or transformer-based deep learning models such as BERT [53], Attention-based BiLSTM [17], Hierarchical attention networks (HAN) [18], and hybrid models such as BerConvoNet [38] and CNN-BiLSTM [54, 55]. The description and implementation details of these techniques are given as follows. All implementation has been carried out using Python libraries.

#### 4.3.1 Kim's CNN model [23]

A detailed description of Kim's model has been provided in Section 3.1. The implementation details of this model are as follows. In Kim's model, sentences are matched to the embedding vectors that are made available as an input matrix to the model. It uses 3 parallel layers of convolution with word vectors on top obtained from the existing pre-trained model, with 100 filters of kernel sizes 3, 4, and 5. It is followed by a dense layer of 64 neurons and a classification layer.

#### 4.3.2 Long short term memory (LSTM) [13, 52]

Long Short-Term Memory Networks (LSTMs) are a special form of recurrent neural network (RNN) that can handle long-term dependencies. Also, LSTMs have a chain-like RNN structure, but the repeated module has a different structure. Rather than having a single layer of the neural network, there are four, which communicate in a unique way. Some of the works used the LSTM model for different text classification tasks [13, 14, 17, 18]. For the comparison with our model, we used a single LSTM layer with 32 memory cells followed by the classification layer.

#### 4.3.3 Very deep convolutional neural networks (VDCNN) [27]

Unlike *TextConvoNet*, which is a shallow network, VDCNN uses multiple layered convolution and max-pooling operations. Therefore, inspired by VDCNN, we implemented its version based on word embedding. This model uses four different pooling processes, each of which reduces the resolution by half, resulting in four different feature map tiers: 64, 128, 256, and 512, followed by a max-pooling layer. After the end of 4 convolution pair operations, the $512 \times k$ resulting features are transformed into a single vector which is the input to a three-layer fully connected classifier (4096,2048,2048) with ReLU hidden units and softmax outputs. Depending on the classification task, the number of output neurons varies.

#### 4.3.4 Attention+BiLSTM model [17]

It starts with an input layer that tokenizes input sentences and indexed lists, followed by an embedding layer. There exist bidirectional LSTM cells (100 hidden units) which can be concatenated to get a representation of each token. Attention weights are taken from linear projection and non-linear activation. Final sentence representation is a weighted sum of all token representations. The final classification output is derived from a simple dense and softmax layer.

#### 4.3.5 BERT model [53]

The pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks, and we have also used the same strategy to compare *TextConvoNet* with BERT based model. We used pre-trained bert-base-uncased embeddings to encode, followed by a dense layer of 712 neurons ended by a classification layer.

#### 4.3.6 Hierarchical attention networks (HAN) model [18]

In our experiment, we set the word embedding dimension to 300 and the GRU dimension to 50. A combination

of forward and backward GRU provides us with 100 dimensions for word/sentence annotation in this scenario. The word/sentence context vectors have a dimension of 100 and are randomly initialized. We utilize a 32-piece mini-batch size for training.

#### 4.3.7 BERT+CNN [38]

Some recent works used BERT as a text embedder and CNN as a classifier [38, 56]. In our experiments, the BERT+CNN model uses kernel sizes of 2, 3, & 4 and the number of kernels (filters) were set to 100. The word vector size was 768. The model is trained on a batch size of 100 with a learning rate of 0.001. Adam optimizer is used with the Binary Cross-Entropy loss function (BCE).

#### 4.3.8 CNN+BiLSTM [54, 55]

Some recent works used a hybrid model based on CNN and BiLSTM for text classification. In our experiments, we use a densely connected BiLSTM layer after the two convolution layers of kernel size 3 and one max pooling layer of size 3. The embedding dimension was set to 300 and the learning rate to 0.001.

#### 4.3.9 Graph neural network based models [57, 58]

In [57], a Graph Neural Network (GNN) based model, TLGNN, is proposed. It generates a text-level graph for each input text. It builds graphs by assuming smaller windows in the text for extracting more local features. The details of the methodology are given in [57]. For comparison purposes, except for *max length parameter*, we consider the exact implementation details and hyperparameters settings provided in the paper. The max length parameter was set to 100. In another work, the Sequential GNN model is proposed for short text classfication [58]. It builds individual graphs for each document based on word co-occurrence and uses a bidirectional long short-term memory network (Bi-LSTM) to extract the sequential features. After that, it uses a Graph Convolutional Network (GCN) for learning word representations. For comparison purposes, we used the default settings of SeqGNN given in the original paper.

### 4.4 Implementation details

All the experiments to examine the model performance of *TextConvoNet_4* and *TextConvoNet_6* are carried out on a system having with Dual-Core Intel Core i5 processor and 8 GB RAM, running Macintosh operating system, with 64- bit processor and access to NVidia K80 GPU kernel. All experiments were performed in Python 3.0. The models are trained over a mini-batch size of 32 using Adam as an optimizer. The learning rate is chosen to be 0.1, and the models are trained over 10 epochs with early stopping to avoid overfitting. All these hyperparameters are chosen using a hyperparameter optimization technique called *GridSearchCV*. We use GloVe[6], a pre-trained word embedding model for generating word vectors from sentences.

## 5 Results and analysis

This section presents the results of the *TextConvoNet* on five datasets (Section 4.1) on different performance measures. Additionally, the comparison results of the *TextConvoNet* with other machine learning and deep learning models are reported in this section.

First, the performance comparison of *TextConvoNet* with other baseline models is presented in the results and analysis. A statistical test is conducted to assess whether *TextConvoNet* performed significantly different from other used baseline models. Additionally, the performance analysis of the presented model is performed by varying the number of sentences in a paragraph. After that, the experimental results to dataset size are discussed, i.e., how does the presented *TextConvoNet* performs with minimal data (few-shot learning) and in challenging scenarios.

### 5.1 Results of *TextConvoNet* architecture

Table 3 shows the results of the presented *TextConvoNet* architecture (*TextConvoNet_6* and *TextConvoNet_4*) compared to different machine learning and deep learning-based models. The results are reported in terms of the used performance measures, accuracy, precision, recall, f1-score, specificity, Gmean1, and Gmean2. Table 3, reports results for binary classification datasets (dataset 1 and dataset 2) and multi-class classification datasets (dataset 3, dataset 4, and dataset 5). The following inferences can be drawn from the results.

It has been found that the presented *TextConvoNet* produced significant result values for the considered performance measures for different used datasets. For accuracy, precision, and recall measures, the average values of *TextConvoNet_6* are 0.889, 0.829, and 0.807, respectively. These values are higher than the other used machine learning and deep learning-based models. The *TextConvoNet* produced significant results for the f1-score measure. The highest f1-score value is 0.969, with an

[6] https://nlp.stanford.edu/projects/glove/

**Table 3** Classification results of the proposed *TextConvoNet* and other baseline models for different performance measures, (**MNB: Multinomial Naive Bayes, GBC= Gradient Boosting Classifier*)

| Models | Accuracy | Precision | Recall | F1-score | Specificity | Gmean1 | Gmean2 | MCC |
|---|---|---|---|---|---|---|---|---|
| DATASET-1 | | | | | | | | |
| TextConvoNet_6 | **0.822** | **0.848** | 0.783 | 0.814 | **0.860** | 0.815 | **0.821** | **0.645** |
| TextConvoNet_4 | 0.819 | 0.833 | 0.798 | 0.815 | 0.841 | 0.815 | 0.819 | 0.639 |
| Kim's model | 0.798 | 0.753 | **0.886** | 0.814 | 0.711 | 0.817 | 0.793 | 0.605 |
| LSTM | 0.792 | 0.747 | 0.883 | 0.809 | 0.702 | 0.812 | 0.787 | 0.595 |
| VDCNN | 0.815 | 0.828 | 0.794 | 0.811 | 0.836 | 0.811 | 0.815 | 0.63 |
| MNBs | 0.819 | 0.804 | 0.844 | **0.823** | 0.795 | **0.824** | 0.819 | 0.639 |
| Random forest | 0.741 | 0.718 | 0.792 | 0.753 | 0.69 | 0.754 | 0.739 | 0.484 |
| Decision tree | 0.649 | 0.646 | 0.656 | 0.651 | 0.641 | 0.651 | 0.649 | 0.297 |
| SVC | 0.775 | 0.756 | 0.811 | 0.782 | 0.739 | 0.783 | 0.774 | 0.551 |
| GBC | 0.689 | 0.646 | 0.832 | 0.727 | 0.546 | 0.733 | 0.674 | 0.394 |
| KNN | 0.577 | 0.599 | 0.462 | 0.522 | 0.692 | 0.526 | 0.565 | 0.158 |
| XGBoost | 0.736 | 0.711 | 0.794 | 0.75 | 0.678 | 0.751 | 0.734 | 0.475 |
| DATASET-2 | | | | | | | | |
| TextConvoNet_6 | **0.909** | 0.909 | 0.862 | **0.890** | 0.939 | 0.876 | **0.901** | **0.807** |
| TextConvoNet_4 | 0.871 | 0.829 | **0.905** | 0.855 | 0.859 | 0.861 | 0.879 | 0.741 |
| Kim's model | 0.885 | 0.867 | 0.859 | 0.861 | 0.888 | 0.843 | 0.879 | 0.75 |
| LSTM | 0.877 | 0.849 | 0.83 | 0.845 | 0.904 | 0.852 | 0.87 | 0.75 |
| VDCNN | 0.77 | **0.968** | 0.467 | 0.61 | **0.983** | 0.694 | 0.686 | 0.565 |
| MNB | 0.824 | 0.831 | 0.921 | 0.812 | 0.556 | **0.881** | 0.715 | 0.532 |
| Random forest | 0.75 | 0.764 | 0.976 | 0.849 | 0.259 | 0.859 | 0.476 | 0.334 |
| Decision tree | 0.689 | 0.774 | 0.784 | 0.779 | 0.451 | 0.781 | 0.582 | 0.229 |
| SVC | 0.781 | 0.788 | 0.967 | 0.868 | 0.361 | 0.867 | 0.58 | 0.428 |
| GBC | 0.786 | 0.799 | 0.928 | 0.858 | 0.442 | 0.857 | 0.651 | 0.428 |
| KNN | 0.723 | 0.737 | 0.956 | 0.813 | 0.178 | 0.839 | 0.43 | 0.2 |
| XGBoost | 0.809 | 0.838 | 0.908 | 0.845 | 0.576 | 0.868 | 0.711 | 0.51 |
| DATASET-3 | | | | | | | | |
| TextConvoNet_6 | 0.992 | 0.968 | 0.968 | 0.968 | 0.995 | 0.968 | 0.981 | 0.963 |
| TextConvoNet_4 | **0.992** | **0.969** | **0.969** | **0.969** | **0.996** | **0.969** | **0.982** | **0.965** |
| Kim's model | 0.986 | 0.945 | 0.945 | 0.945 | 0.992 | 0.945 | 0.968 | 0.937 |
| LSTM | 0.874 | 0.495 | 0.495 | 0.495 | 0.928 | 0.495 | 0.678 | 0.423 |
| VDCNN | 0.965 | 0.862 | 0.862 | 0.862 | 0.98 | 0.862 | 0.919 | 0.842 |
| MNB | 0.99 | 0.958 | 0.958 | 0.958 | 0.994 | 0.958 | 0.976 | 0.952 |
| Random forest | 0.983 | 0.931 | 0.931 | 0.931 | 0.99 | 0.931 | 0.96 | 0.921 |
| Decision tree | 0.977 | 0.91 | 0.91 | 0.91 | 0.987 | 0.91 | 0.948 | 0.897 |
| SVC | 0.978 | 0.914 | 0.914 | 0.914 | 0.988 | 0.914 | 0.95 | 0.901 |
| GBC | 0.986 | 0.944 | 0.944 | 0.944 | 0.992 | 0.944 | 0.968 | 0.936 |
| KNN | 0.972 | 0.887 | 0.887 | 0.887 | 0.984 | 0.887 | 0.934 | 0.871 |
| XGBoost | 0.99 | 0.96 | 0.96 | 0.96 | 0.994 | 0.96 | 0.977 | 0.955 |
| DATASET-4 | | | | | | | | |
| TextConvoNet_6 | **0.884** | **0.826** | **0.826** | **0.826** | **0.913** | **0.826** | **0.869** | **0.739** |
| TextConvoNet_4 | 0.873 | 0.809 | 0.809 | 0.809 | 0.904 | 0.809 | 0.855 | 0.713 |
| Kim's model | 0.869 | 0.803 | 0.803 | 0.803 | 0.901 | 0.803 | 0.851 | 0.704 |
| LSTM | 0.876 | 0.813 | 0.813 | 0.813 | 0.907 | 0.813 | 0.859 | 0.72 |
| VDCNN | 0.871 | 0.806 | 0.806 | 0.806 | 0.903 | 0.806 | 0.853 | 0.709 |
| MNB | 0.87 | 0.804 | 0.804 | 0.804 | 0.902 | 0.804 | 0.852 | 0.706 |
| Random forest | 0.872 | 0.809 | 0.809 | 0.809 | 0.904 | 0.809 | 0.855 | 0.713 |
| Decision tree | 0.807 | 0.71 | 0.71 | 0.71 | 0.855 | 0.71 | 0.779 | 0.565 |
| SVC | 0.881 | 0.821 | 0.821 | 0.821 | 0.911 | 0.821 | 0.865 | 0.732 |
| GBC | 0.866 | 0.8 | 0.8 | 0.8 | 0.9 | 0.8 | 0.848 | 0.699 |

**Table 3** (continued)

| Models | Accuracy | Precision | Recall | F1-score | Specificity | Gmean1 | Gmean2 | MCC |
|---|---|---|---|---|---|---|---|---|
| KNN | 0.662 | 0.493 | 0.493 | 0.493 | 0.746 | 0.493 | 0.606 | 0.239 |
| XGBoost | 0.877 | 0.815 | 0.815 | 0.815 | 0.908 | 0.815 | 0.86 | 0.723 |
| DATASET-5 | | | | | | | | |
| TextConvoNet_6 | **0.839** | **0.597** | **0.597** | **0.597** | **0.899** | **0.597** | **0.732** | **0.496** |
| TextConvoNet_4 | 0.828 | 0.571 | 0.571 | 0.571 | 0.893 | 0.571 | 0.714 | 0.463 |
| Kim's model | 0.804 | 0.51 | 0.51 | 0.51 | 0.878 | 0.51 | 0.669 | 0.388 |
| LSTM | 0.702 | 0.255 | 0.255 | 0.255 | 0.814 | 0.255 | 0.456 | 0.069 |
| VDCNN | 0.762 | 0.404 | 0.404 | 0.404 | 0.851 | 0.404 | 0.586 | 0.255 |
| MNB | 0.754 | 0.385 | 0.385 | 0.385 | 0.846 | 0.385 | 0.571 | 0.232 |
| Random forest | 0.75 | 0.375 | 0.375 | 0.375 | 0.844 | 0.375 | 0.562 | 0.218 |
| Decision tree | 0.729 | 0.321 | 0.321 | 0.321 | 0.83 | 0.321 | 0.517 | 0.152 |
| SVC | 0.76 | 0.401 | 0.401 | 0.401 | 0.85 | 0.401 | 0.584 | 0.251 |
| GBC | 0.78 | 0.449 | 0.449 | 0.449 | 0.862 | 0.449 | 0.622 | 0.312 |
| KNN | 0.69 | 0.224 | 0.224 | 0.224 | 0.806 | 0.224 | 0.425 | 0.03 |
| XGBoost | 0.785 | 0.461 | 0.461 | 0.461 | 0.865 | 0.461 | 0.632 | 0.327 |
| Average | | | | | | | | |
| TextConvoNet_6 | 0.8892 | 0.8296 | 0.8072 | 0.819 | 0.9212 | 0.8164 | 0.8608 | 0.73 |
| TextConvoNet_4 | 0.8766 | 0.8022 | 0.8104 | 0.8038 | 0.8986 | 0.805 | 0.8498 | 0.7042 |
| Kim's model | 0.8684 | 0.7756 | 0.77925 | 0.7866 | 0.874 | 0.7836 | 0.832 | 0.6768 |
| LSTM | 0.8242 | 0.6318 | 0.6552 | 0.6434 | 0.851 | 0.6454 | 0.73 | 0.5114 |
| VDCNN | 0.8366 | 0.7736 | 0.6666 | 0.6986 | 0.9106 | 0.7154 | 0.7718 | 0.6002 |
| MNB | 0.8514 | 0.7564 | 0.7824 | 0.7564 | 0.8186 | 0.7704 | 0.7866 | 0.6122 |
| Random forest | 0.8192 | 0.7194 | 0.7766 | 0.7434 | 0.7374 | 0.7456 | 0.7184 | 0.534 |
| Decision tree | 0.7702 | 0.6722 | 0.6762 | 0.6742 | 0.7528 | 0.6746 | 0.695 | 0.428 |
| SVC | 0.835 | 0.736 | 0.7828 | 0.7572 | 0.7698 | 0.7572 | 0.7506 | 0.5726 |
| GBC | 0.8214 | 0.7276 | 0.7906 | 0.7556 | 0.7484 | 0.7566 | 0.7526 | 0.5538 |
| KNN | 0.7248 | 0.588 | 0.6044 | 0.5878 | 0.6812 | 0.5938 | 0.592 | 0.2996 |
| XGBoost | 0.8394 | 0.757 | 0.7876 | 0.7662 | 0.8042 | 0.771 | 0.7828 | 0.598 |

Bold entries show the significant values

average value of 0.819. Similarly, the specificity measure's highest value is 0.996, and the average value is 0.921. For other measures, g-means (Gmean1 and Gmean2) and MCC, the *TextConvoNet* produced average values of 0.816, 0.86, and 0.73, respectively. On average, for accuracy, the maximum performance improvement achieved by *TextConvoNet* compared to other models is 16%. Similarly, on average, for precision, recall, and f1-score, the maximum average improvement achieved by *TextConvoNet* is 24%, 21%, and 23.9%, respectively. For the other measures, on average, the improvement achieved by *TextConvoNet* is 24% in terms of Specificity, 22% in terms of Gmean1, 27% in terms of Gmean2, and 44% in terms of MCC measure. Overall, in multi-class datasets (3, 4, and 5), variants of *TextConvoNet* perform better than all the other models in terms of all the performance measures.

## 5.2 Comparison of the *TextConvoNet* with recent attention-based models, BERT model and graph based models

This section compares the performance of the presented *TextConvoNet* with two attention-based models: BiLSTM followed by attention (Attention+BiLSTM), Hierarchical Attention Network (HAN) model, one transformer-based BERT model, and CNN-based hybrid text classification models: BERT-CNN (BerConvoNet) and CNN-BiLSTM. Additionally, it is also compared with graph-based models: Text Level Graph Neural Network (TLGNN) and Sequential GNN (SeqGNN). Section 4.3 provides details of these models. Table 4 shows the results of *TextConvoNet* and other considered models, BiLSTM +Attention, BERT, HAN, BerConvoNet, CNN-BiLSTM, TLGNN, and

**Table 4** Result comparison of *TextConvoNet* with attention and/or transformer-based deep learning models

| | Accuracy | Precision | Recall | F1-score | Specificity | Gmean 1 | Gmean 2 | MCC |
|---|---|---|---|---|---|---|---|---|
| DATASET-1 | | | | | | | | |
| BiLSTM + Attention | 0.806 | 0.816 | 0.790 | 0.803 | 0.822 | 0.803 | 0.806 | 0.613 |
| BERT | 0.776 | 0.736 | 0.859 | 0.793 | 0.694 | 0.795 | 0.772 | 0.560 |
| HAN | 0.802 | 0.791 | 0.821 | 0.806 | 0.783 | 0.806 | 0.802 | 0.606 |
| BerConvoNet | **0.831** | 0.802 | 0.797 | 0.794 | 0.697 | 0.799 | 0.814 | 0.640 |
| CNN-BiLSTM | 0.820 | 0.796 | 0.859 | 0.826 | 0.781 | 0.827 | 0.819 | 0.642 |
| TLGNN | 0.710 | 0.725 | 0.675 | 0.699 | 0.745 | 0.700 | 0.709 | 0.422 |
| SeqGNN | 0.821 | 0.787 | **0.882** | **0.828** | 0.763 | 0.829 | 0.820 | 0.649 |
| *TextConvoNet_6* | 0.822 | **0.848** | 0.783 | 0.814 | **0.860** | **0.815** | **0.821** | **0.645** |
| *TextConvoNet_4* | 0.819 | 0.833 | 0.798 | 0.815 | 0.841 | 0.815 | 0.819 | 0.639 |
| DATASET-2 | | | | | | | | |
| BiLSTM + Attention | 0.886 | 0.883 | 0.846 | 0.864 | 0.915 | 0.864 | 0.880 | 0.766 |
| BERT | 0.772 | 0.685 | 0.867 | 0.765 | 0.700 | 0.771 | 0.779 | 0.564 |
| HAN | 0.863 | 0.880 | 0.788 | 0.831 | 0.919 | 0.833 | 0.851 | 0.720 |
| BerConvoNet | 0.883 | 0.821 | 0.911 | 0.866 | 0.859 | 0.849 | 0.852 | 0.755 |
| CNN-BiLSTM | 0.872 | 0.901 | 0.861 | 0.881 | 0.886 | 0.881 | 0.873 | 0.745 |
| TLGNN | 0.732 | 0.653 | 0.800 | 0.719 | 0.680 | 0.723 | 0.737 | 0.476 |
| SeqGNN | 0.821 | 0.780 | 0.882 | 0.828 | 0.763 | 0.829 | 0.820 | 0.649 |
| *TextConvoNet_6* | **0.904** | **0.905** | 0.867 | **0.886** | **0.932** | **0.886** | **0.899** | **0.804** |
| *TextConvoNet_4* | 0.872 | 0.819 | **0.902** | 0.858 | 0.849 | 0.859 | 0.875 | 0.745 |
| DATASET-3 | | | | | | | | |
| BiLSTM + Attention | 0.990 | 0.960 | 0.960 | 0.960 | 0.994 | 0.960 | 0.977 | 0.954 |
| BERT | 0.956 | 0.825 | 0.825 | 0.825 | 0.975 | 0.825 | 0.897 | 0.801 |
| HAN | 0.890 | 0.560 | 0.560 | 0.560 | 0.937 | 0.560 | 0.724 | 0.497 |
| BerConvoNet | 0.992 | 0.968 | 0.968 | 0.968 | 0.995 | 0.966 | 0.982 | 0.966 |
| CNN-BiLSTM | 0.989 | 0.959 | 0.959 | 0.959 | 0.994 | 0.959 | 0.976 | 0.954 |
| TLGNN | 0.944 | 0.777 | 0.777 | 0.777 | 0.968 | 0.777 | 0.867 | 0.745 |
| SeqGNN | 0.967 | 0.962 | 0.819 | 0.885 | 0.994 | 0.887 | 0.902 | 0.870 |
| *TextConvoNet_6* | 0.992 | 0.968 | 0.968 | 0.968 | 0.995 | 0.968 | 0.981 | 0.963 |
| *TextConvoNet_4* | **0.992** | **0.969** | **0.969** | **0.969** | **0.996** | **0.969** | **0.982** | **0.965** |
| DATASET-4 | | | | | | | | |
| BiLSTM + Attention | 0.861 | 0.792 | 0.792 | 0.792 | 0.896 | 0.792 | 0.843 | 0.689 |
| BERT | 0.808 | 0.712 | 0.712 | 0.712 | 0.856 | 0.712 | 0.7806 | 0.568 |
| HAN | 0.795 | 0.693 | 0.693 | 0.693 | 0.846 | 0.693 | 0.766 | 0.540 |
| BerConvoNet | 0.882 | 0.824 | 0.824 | 0.824 | 0.921 | 0.826 | 0.868 | 0.738 |
| CNN-BiLSTM | 0.841 | 0.762 | 0.762 | 0.762 | 0.881 | 0.762 | 0.819 | 0.643 |
| TLGNN | 0.800 | 0.700 | 0.706 | 0.706 | 0.850 | 0.706 | 0.771 | 0.551 |
| SeqGNN | 0.875 | 0.861 | 0.747 | 0.8 | **0.940** | 0.802 | 0.838 | 0.714 |
| *TextConvoNet_6* | **0.884** | **0.826** | **0.826** | **0.826** | 0.913 | **0.826** | **0.869** | **0.739** |
| *TextConvoNet_4* | 0.873 | 0.809 | 0.809 | 0.809 | 0.904 | 0.809 | 0.855 | 0.713 |
| DATASET-5 | | | | | | | | |
| BiLSTM + Attention | 0.799 | 0.493 | 0.493 | 0.493 | 0.874 | 0.499 | 0.660 | 0.374 |
| BERT | 0.711 | 0.278 | 0.278 | 0.278 | 0.819 | 0.278 | 0.477 | 0.098 |
| HAN | 0.700 | 0.250 | 0.250 | 0.250 | 0.812 | 0.250 | 0.450 | 0.062 |
| BerConvoNet | 0.828 | 0.572 | 0.572 | 0.572 | 0.894 | 0.572 | 0.721 | 0.474 |
| CNN-BiLSTM | 0.830 | 0.576 | 0.576 | 0.576 | 0.89 | 0.576 | 0.718 | 0.470 |

**Table 4** (continued)

| | Accuracy | Precision | Recall | F1-score | Specificity | Gmean 1 | Gmean 2 | MCC |
|---|---|---|---|---|---|---|---|---|
| TLGNN | 0.729 | 0.322 | 0.322 | 0.322 | 0.830 | 0.322 | 0.517 | 0.153 |
| SeqGNN | 0.737 | 0.285 | 0.209 | 0.241 | 0.869 | 0.244 | 0.426 | 0.0885 |
| *TextConvoNet_6* | **0.839** | **0.597** | **0.597** | **0.597** | **0.899** | **0.597** | **0.732** | **0.496** |
| *TextConvoNet_4* | 0.828 | 0.571 | 0.571 | 0.571 | 0.893 | 0.571 | 0.714 | 0.463 |

Bold entries show the significant values

SeqGNN, on different performance measures. From the table, it can be observed that the BERT and HAN models have produced relatively poor results on all the datasets compared to other models. On datasets 1 and 3, *TextConvoNet_4* has produced the best results in comparison with others. On datasets 2, 4, and 5, *TextConvoNet_6* has yielded better performance than other others. Overall, it has been found that *TextConvoNet* performed better in comparison to other models. The performance of the BerConvoNet is comparable to the presented *TextConvoNet* models. The presented models produced better performance than the graph-based models for all the datasets, except for Dataset-1. Where the SeqGNN produced better recall and f1-score values than the presented models. Furthermore, it has been observed from Table 3 that the non-attention-based competing models have produced different values for the used performance measures themselves. However, for these models, the precision, recall, and F1-score values are the same for datasets 3, 4, and 5. These three are multiclass datasets. We have adopted the micro-averaging technique of calculating the values of different metrics for multiclass classification. Micro-averaging is a well-suited technique as it doesn't discriminate between classes based on their population. Hence, micro-averaging in a multiclass setting with

**Table 5** Different versions of *TextConvoNet*

| | # Filters | Dropout rate | Optimizers | Dense layer (units) | Kernel size (intra-sentence layers) | Kernel sizes (inter-sentence layers) |
|---|---|---|---|---|---|---|
| V1.1 | 32 | 0.4 | adam | 64 | (1X2,1X3,1X4) | (2X1,2X2,2X3) |
| V1.2 | 32 | 0.4 | RMSProp | 64 | (1X2,1X3,1X4) | (2X1,2X2,2X3) |
| V1.3 | 48 | 0.4 | adam | 64 | (1X2,1X3,1X4) | (2X1,2X2,2X3) |
| V1.4 | 48 | 0.5 | adam | 64 | (1X2,1X3,1X4) | (2X1,2X2,2X3) |
| V1.5 | 32 | 0.4 | adam | 96 | (1X2,1X3,1X4) | (2X1,2X2,2X3) |
| V1.6 | 32 | 0.5 | adam | 96 | (1X2,1X3,1X4) | (2X1,2X2,2X3) |
| V2.1 | 32 | 0.4 | adam | 64 | (1X3,1X4,1X5) | (2X2,2X3,2X4) |
| V2.2 | 32 | 0.4 | RMSProp | 64 | (1X3,1X4,1X5) | (2X2,2X3,2X4) |
| V2.3 | 48 | 0.4 | adam | 64 | (1X3,1X4,1X5) | (2X2,2X3,2X4) |
| V2.4 | 48 | 0.5 | adam | 64 | (1X3,1X4,1X5) | (2X2,2X3,2X4) |
| V2.5 | 32 | 0.4 | adam | 96 | (1X3,1X4,1X5) | (2X2,2X3,2X4) |
| V2.6 | 32 | 0.5 | adam | 96 | (1X3,1X4,1X5) | (2X2,2X3,2X4) |
| V3.1 | 32 | 0.4 | adam | 64 | (1X2,1X3,1X4) | (3X1,3X2,3X3) |
| V3.2 | 32 | 0.4 | RMSProp | 64 | (1X2,1X3,1X4) | (3X1,3X2,3X3) |
| V3.3 | 48 | 0.4 | adam | 64 | (1X2,1X3,1X4) | (3X1,3X2,3X3) |
| V3.4 | 48 | 0.5 | adam | 64 | (1X2,1X3,1X4) | (3X1,3X2,3X3) |
| V3.5 | 32 | 0.4 | adam | 96 | (1X2,1X3,1X4) | (3X1,3X2,3X3) |
| V3.6 | 32 | 0.5 | adam | 96 | (1X2,1X3,1X4) | (3X1,3X2,3X3) |
| V4.1 | 32 | 0.4 | adam | 64 | (1X3,1X4,1X5) | (3X2,3X3,3X4) |
| V4.2 | 32 | 0.4 | RMSProp | 64 | (1X3,1X4,1X5) | (3X2,3X3,3X4) |
| V4.3 | 48 | 0.4 | adam | 64 | (1X3,1X4,1X5) | (3X2,3X3,3X4) |
| V4.4 | 48 | 0.5 | adam | 64 | (1X3,1X4,1X5) | (3X2,3X3,3X4) |
| V4.5 | 32 | 0.4 | adam | 96 | (1X3,1X4,1X5) | (3X2,3X3,3X4) |
| V4.6 | 32 | 0.5 | adam | 96 | (1X3,1X4,1X5) | (3X2,3X3,3X4) |

all labels included, produces equal precision and recall (thus F1-score) for datasets 3, 4, and 5 (multiclass datasets).

Overall, it has been found that the presented *TextConvoNet* outperformed all the other attention and non-attention models for multiclass datasets (Datasets -3, 4, and 5) for all performance measures. For binary class datasets, *TexConvoNet* produced a relatively lower performance for only 2-3 cases compared to other models. The results of the presented *TextConvoNet* model are comparable or improved than the other used attention models and the BERT model.

### 5.3 Effect of different parameter values on the performance of the *TextConvoNet*

The presented *TextConvoNet* has been evaluated over a variety of parameters given in Table 5 to analyze their effect on the performance of the *TextConvoNet* for various measures. Between any two versions, there is a change in the kernel size. Within a version, between any two sub-versions, there are changes in the number of filters, dropout rate before the classification layer, optimizer, and the units in the fully connected layer. Generally, performance measures can change by adhering to the needs, application, and model type. In practice, each NLP application is unique. Therefore, a unique approach/model is needed for every application of NLP. Hence, this analysis has been performed on all five datasets for accuracy, precision, and recall measures, and results are recorded. Table 6 shows the results of this analysis. The following observations have been drawn from the results of various versions of *TextConvoNet*.

- It has been found that Adam is the best optimizer for the presented model. All other used optimizers took a large amount of time to train or did not give good results, as in the case of *RMSProp* (V1.2, V2.2, V3.2, V4.2).
- Dropout Rate of 0.4 was found to be optimum as the value 0.5 model was slightly overfitting.

**Table 6** Results of accuracy, precision, and recall evaluation measures on different datasets

| | Accuracy | | | | | Precision | | | | | Recall | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Dataset 1 | Dataset 2 | Dataset 3 | Dataset 4 | Dataset 5 | Dataset 1 | Dataset 2 | Dataset 3 | Dataset 4 | Dataset 5 | Dataset 1 | Dataset 2 | Dataset 3 | Dataset 4 | Dataset 5 |
| V1.1 | 0.822 | **0.904** | 0.992 | 0.884 | **0.839** | **0.848** | 0.905 | 0.968 | 0.826 | **0.597** | 0.783 | 0.867 | 0.968 | 0.826 | **0.597** |
| V1.2 | 0.825 | 0.880 | 0.991 | 0.867 | 0.799 | 0.811 | 0.891 | 0.963 | 0.801 | 0.497 | 0.848 | 0.821 | 0.963 | 0.801 | 0.497 |
| V1.3 | 0.820 | 0.890 | 0.992 | 0.874 | 0.835 | 0.822 | 0.900 | 0.968 | 0.811 | 0.587 | 0.816 | 0.837 | 0.968 | 0.811 | 0.587 |
| V1.4 | 0.821 | 0.893 | 0.992 | 0.883 | 0.836 | 0.806 | 0.919 | 0.970 | 0.825 | 0.589 | 0.845 | 0.823 | 0.970 | 0.825 | 0.589 |
| V1.5 | 0.823 | 0.890 | 0.992 | 0.880 | 0.824 | 0.802 | 0.864 | 0.968 | 0.821 | 0.561 | 0.856 | 0.884 | 0.968 | 0.821 | 0.561 |
| V1.6 | 0.805 | 0.896 | 0.992 | 0.868 | 0.825 | 0.762 | 0.916 | 0.968 | 0.802 | 0.563 | 0.886 | 0.835 | 0.968 | 0.802 | 0.563 |
| V2.1 | 0.823 | 0.880 | 0.992 | 0.880 | 0.818 | 0.806 | 0.846 | 0.969 | 0.820 | 0.545 | 0.850 | 0.881 | 0.969 | 0.820 | 0.545 |
| V2.2 | 0.807 | 0.842 | 0.992 | 0.876 | 0.806 | 0.775 | **0.942** | 0.970 | 0.815 | 0.515 | 0.865 | 0.674 | 0.970 | 0.815 | 0.515 |
| V2.3 | 0.825 | 0.891 | 0.991 | 0.879 | 0.829 | 0.801 | 0.853 | 0.964 | 0.819 | 0.573 | 0.864 | 0.902 | 0.964 | 0.819 | 0.573 |
| V2.4 | 0.825 | 0.896 | 0.991 | 0.886 | 0.831 | 0.821 | 0.916 | 0.963 | 0.830 | 0.579 | 0.829 | 0.835 | 0.963 | 0.830 | 0.579 |
| V2.5 | 0.826 | 0.890 | 0.991 | 0.875 | 0.827 | 0.805 | 0.883 | 0.966 | 0.812 | 0.567 | 0.860 | 0.858 | 0.966 | 0.812 | 0.567 |
| V2.6 | 0.811 | 0.892 | **0.994** | **0.892** | 0.828 | 0.764 | 0.874 | **0.978** | **0.838** | 0.569 | 0.900 | 0.874 | **0.978** | **0.838** | 0.569 |
| V3.1 | 0.826 | 0.884 | 0.992 | 0.879 | 0.824 | 0.825 | 0.881 | 0.968 | 0.819 | 0.561 | 0.826 | 0.844 | 0.968 | 0.819 | 0.561 |
| V3.2 | 0.794 | 0.874 | 0.992 | 0.854 | 0.806 | 0.732 | 0.817 | 0.966 | 0.782 | 0.516 | **0.925** | 0.912 | 0.966 | 0.782 | 0.516 |
| V3.3 | 0.826 | 0.894 | 0.994 | 0.882 | 0.832 | 0.805 | 0.870 | 0.976 | 0.823 | 0.580 | 0.861 | 0.886 | 0.976 | 0.823 | 0.580 |
| V3.4 | 0.825 | 0.878 | 0.994 | 0.880 | 0.827 | 0.826 | 0.822 | 0.974 | 0.820 | 0.567 | 0.822 | **0.914** | 0.974 | 0.820 | 0.567 |
| V3.5 | 0.820 | 0.894 | 0.992 | 0.884 | 0.822 | 0.799 | 0.911 | 0.967 | 0.826 | 0.554 | 0.855 | 0.835 | 0.967 | 0.826 | 0.554 |
| V3.6 | 0.818 | 0.886 | 0.992 | 0.869 | 0.824 | 0.785 | 0.907 | 0.968 | 0.804 | 0.560 | 0.875 | 0.819 | 0.968 | 0.804 | 0.560 |
| 0.825 | 0.808 | 0.917 | 0.968 | 0.817 | 0.563 | 0.861 | 0.851 | 0.968 | 0.817 | 0.563 | | | | | |
| V4.2 | 0.800 | 0.851 | 0.991 | 0.871 | 0.790 | 0.885 | 0.932 | 0.962 | 0.806 | 0.476 | 0.688 | 0.705 | 0.962 | 0.806 | 0.476 |
| V4.3 | 0.812 | 0.897 | 0.992 | 0.878 | 0.826 | 0.783 | 0.885 | 0.968 | 0.817 | 0.564 | 0.862 | 0.874 | 0.968 | 0.817 | 0.564 |
| V4.4 | 0.822 | 0.889 | 0.990 | 0.885 | 0.827 | 0.804 | 0.867 | 0.962 | 0.828 | 0.567 | 0.850 | 0.877 | 0.962 | 0.828 | 0.567 |
| V4.5 | 0.825 | 0.896 | 0.993 | 0.884 | 0.815 | 0.828 | 0.890 | 0.971 | 0.827 | 0.537 | 0.821 | 0.865 | 0.971 | 0.827 | 0.537 |
| V4.6 | 0.823 | 0.884 | 0.992 | 0.875 | 0.823 | 0.795 | 0.862 | 0.968 | 0.812 | 0.557 | 0.868 | 0.870 | 0.968 | 0.812 | 0.557 |

Bold entries show the significant values

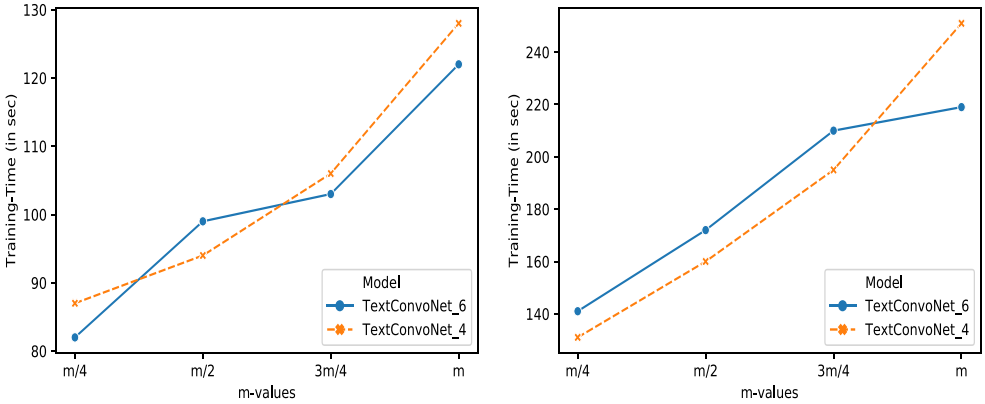


**Fig. 3** Training time with different values of $m$

(a) $m$ on Dataset 2 (b) $m$ on Dataset 5

– In datasets with longer texts (such as Dataset-2 and Dataset-5), versions 3 and 4 give slightly better results when compared to versions 1 and 2. The large kernel sizes in versions 3 and 4 might be the possible reason behind the inference. In contrast, datasets (Datasets 1, 3, and 4) with comparatively smaller paragraphs do well in versions 1 and 2.

**Optimum value of number of sentences in a paragraph (*m*)** To evaluate our models' run-time performance, we tracked the training time by varying the value of *(m)*. The rationale behind it is that varying the value of $m$ ranging from ($\frac{m}{4}$ to $m$) leads to an increase in the computation time. The results for the run-time performance of *TextConvoNet* are shown in Fig. 3. The value of $m$ has been varied from $\frac{m}{4}$ to $m$, and the performance metrics for each of the above m-values were calculated. The results are mentioned in Table 7. The observations obtained from the results are as follows.

– It has been observed that *TextConvoNet_6* improves marginally for all the performance measures when the value is varied from $\frac{m}{4}$ to $m$ on datasets having a maximum length of sentences in a paragraph considerably small, as shown for (Dataset-4).

**Table 7** Optimum value of $m$

| | m/4 | | m/2 | | 3m/4 | | m | |
|---|---|---|---|---|---|---|---|---|
| | TextConvo Net_6 | TextConvo Net_4 | TextConvo Net_6 | TextConvo Net_4 | TextConvo Net_6 | TextConvo Net_4 | TextConvo Net_6 | TextConvo Net_4 |
| DATASET-2 | | | | | | | | |
| Accuracy | 0.890 | 0.886 | 0.901 | 0.887 | 0.889 | 0.888 | 0.901 | 0.894 |
| Precision | 0.894 | 0.864 | 0.903 | 0.901 | 0.888 | 0.875 | 0.899 | **0.911** |
| Recall | 0.844 | **0.872** | 0.863 | 0.828 | 0.849 | 0.863 | 0.867 | 0.835 |
| F1_score | 0.868 | 0.868 | 0.882 | 0.863 | 0.868 | 0.869 | **0.883** | 0.871 |
| Specificity | 0.925 | 0.896 | **0.930** | 0.932 | 0.919 | 0.907 | 0.926 | 0.939 |
| Gmean1 | 0.869 | 0.868 | **0.883** | 0.864 | 0.868 | 0.869 | 0.883 | 0.872 |
| Gmean2 | 0.883 | 0.884 | **0.896** | 0.878 | 0.883 | 0.885 | 0.896 | 0.885 |
| MCC | 0.775 | 0.768 | **0.798** | 0.769 | 0.773 | 0.771 | 0.798 | 0.784 |
| [6pt] DATASET-4 | | | | | | | | |
| Accuracy | 0.826 | 0.820 | 0.827 | 0.828 | 0.829 | 0.829 | **0.829** | 0.824 |
| Precision | 0.565 | 0.551 | 0.569 | 0.570 | **0.573** | 0.572 | 0.572 | 0.559 |
| Recall | 0.565 | 0.551 | 0.569 | 0.570 | **0.573** | 0.572 | 0.572 | 0.559 |
| F1_score | 0.565 | 0.551 | 0.569 | 0.570 | **0.573** | 0.572 | 0.572 | 0.559 |
| Specificity | 0.891 | 0.888 | 0.892 | 0.893 | 0.893 | 0.893 | **0.893** | 0.890 |
| Gmean1 | 0.565 | 0.551 | 0.569 | 0.570 | **0.573** | 0.572 | 0.572 | 0.559 |
| Gmean2 | 0.710 | 0.699 | 0.712 | 0.713 | 0.715 | 0.715 | **0.715** | 0.705 |
| MCC | 0.457 | 0.438 | 0.461 | 0.463 | **0.466** | 0.465 | 0.465 | 0.449 |

Bold entries show the significant values

– For the datasets having maximum length of sentenca es in a paragraph considerably larger, *TextConvoNet_6* performs well for lower values of $m$ $\left(\frac{m}{4}, \frac{m}{2}, \frac{3m}{4}\right)$.

The possible reason for the following observations could be that *TextConvoNet_6* finds it difficult to extract features from smaller paragraphs due to fewer data and hence requiring a high number of sentences to work on, as seen in Dataset-4. On the other hand, *TextConvoNet_6* drops a portion of sentences, such as in the case of Dataset-2, having larger paragraphs due to the ample amount of textual data already present. Therefore, the training time reduces for the *TextConvoNet_6*.

### 5.4 Performance evaluation of *TextConvoNet* for fewshot learning

In minimal data and challenging scenarios, it becomes rather important that the model can train well on a minimalist dataset (i.e., a dataset with a fewer training instances) and perform reasonably well on the test set [59]. Few-shot learning for text classification is a scenario in which a small amount of labeled data for each category is available. The goal of the prediction model is to generalize new unseen examples in the same categories quickly and effectively. In the experiment, the test dataset's size remains constant, as mentioned in Table 2, for all the training percentages and is plotted against the test error rate at those training percentages. We evaluate *TextConvoNet_6*, *TextConvoNet_4*, Kim's CNN model, LSTM, and VDCNN on one binary dataset and one multi-class dataset with varying proportions of training examples. The results are shown in Figs. 4 and 5.

It is observed that the *TextConvoNet* model performs better than all the other baseline models with lower test error rates at an even lower proportion of training examples. Furthermore, the *TextConvoNet* achieved lower test error rates without any change in its parameter space. *TextConvoNet* extracts not just the n-gram based

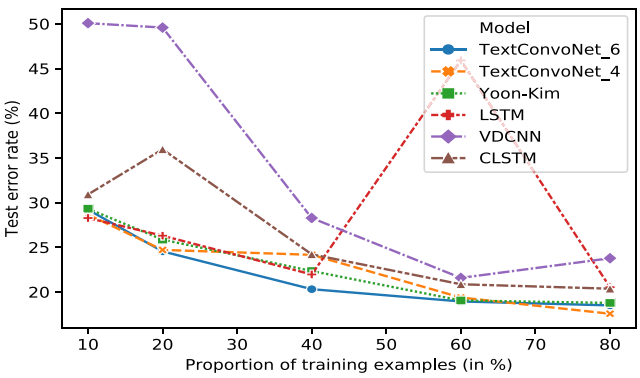


**Fig. 4** Test error rates on Dataset-1 (Binary-class)

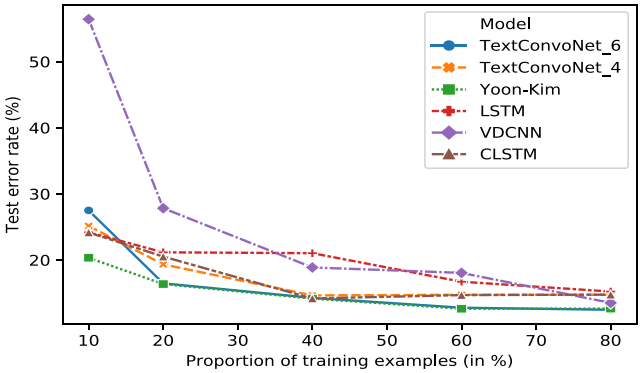


**Fig. 5** Test error rates on Dataset-4 (Multi-class)

characteristics between the words of the same sentence as 1-D CNN does but also the inter-sentence n-gram based features. As a result, *TextConvonet* will be able to extract additional features that 1-D CNN models will not be able to. It strengthens our claim that the proposed *TextConvoNet* performs reasonably well even with fewer training examples.

### 5.5 Statistical test results

Table 8 repors the Wilcoxon signed-rank test results in terms of p-values and effect $r$ for the *TextConvoNet_6* and *TextConvoNet_4* with other used techniques, respectively. If the p-value is less than 0.05, the performance difference is statistically significant and marked with the asterisk (*); thus, the null hypothesis is rejected. The effect $r$ value of the test shows the magnitude of the performance difference between the comparison groups. From Table 8, it is observed that there is a statistically significant difference between the presented *TextConvoNet_6* and all other considered techniques. The experimental p-values are less than the significance level of 0.05 in all groups. Further, effect $r$ values are higher than 0.45 in all the groups, showing a large magnitude of performance diffen the *TextConvoNet_6* and other techniques. Similarly, Table 8 shows that the performance difference between the presented *TextConvoNet_4* and other techniques is also statistically significant at the given significance level for all cases. A significant difference can be seen in all the groups as the p-values are below 0.05. Further, effect $r$ values are higher than 0.40 for all the groups showing a large magnitude of the performance difference.

## 6 Discussion

This work has presented *TextConvoNet*, a CNN-based architecture for text classification. The essence of the text

**Table 8** Results of Wilcoxon signed-rank paired sampled test between the presented *TextConvoNet_6*, *TextConvoNet_4*, and other techniques/models (* showing the groups with the statistically significant difference)

| | Comparison Group | P-value | Effect *r* | Comparison Group | P-value | Effect *r* |
|---|---|---|---|---|---|---|
| TextConvoNet_6 (Two-tailed) *Vs.* | TextConvoNet_4 | 3.36E-05* | 0.463 | TextConvoNet_4 (Two-tailed) *Vs.* | Yoonkim | 0.00017* | 0.419 |
| | Yoonkim | 1.63E-07* | 0.585 | | LSTM | 3.36E-05* | 0.463 |
| | LSTM | 4.66E-08* | 0.61 | | VDCNN | 6.75E-07* | 0.55 |
| | VDCNN | 2.77E-07* | 0.574 | | CLSTM | 1.83E-05* | 0.479 |
| | CLSTM | 5.47E-08* | 0.607 | | MNB | 0.00039* | 0.395 |
| | MNB | 1.72E-06* | 0.534 | | RF | 4.66E-08* | 0.61 |
| | RF | 6.40E-08* | 0.604 | | DT | 6.50E-09* | 0.648 |
| | DT | 6.50E-09* | 0.648 | | SVC | 1.22E-05* | 0.489 |
| | SVC | 1.76E-07 | 0.83 | | GBC | 1.29E-07* | 0.59 |
| | GBC | 1.76E-07* | 0.583 | | KNN | 9.89E-09* | 0.64 |
| | KNN | 1.07E-08* | 0.639 | | XGBoost | 5.82E-05* | 0.449 |
| | XGBoost | 1.20E-07 | 0.591 | | BiLSTM + Attention | 8.79E-05* | 0.438 |
| | BiLSTM + Attention | 1.29E-08* | 0.635 | | BERT | 1.37E-08* | 0.634 |
| | BERT | 1.37E-08 | 0.634 | | HAN | 1.50E-07* | 0.587 |
| | HAN | 1.16E-08* | 0.638 | | BerConvoNet | 0.055 | 0.178 |
| | BerConvoNet | 0.00011* | 0.412 | | CNN-BiLSTM | 0.070 | 0.164 |
| | CNN-BiLSTM | 1.05E-06* | 0.53 | | TLGNN | 4.66E-09* | 0.64 |
| | TLGNN | 4.66E-09* | 0.642 | | SeqGNN | 1.73E-05* | 0.462 |
| | SeqGNN | 3.8E-06* | 0.50 | | | | |

**MNB= Multinomial Naïve Bayes, RF= Random Forest, DT= Decision Tree, SVC= Support Vector Classifier, GBC= Gradient Boosting Classifier*

classification model is to extract the key phrases from the text to assign them an appropriate label. Most existing models utilized one-dimensional (1-D) convolution followed by a one-dimensional pooling operation to extract a feature vector from the input word embeddings. The existing 1-D convolution only extracts n-gram-based features from two or more than two-word in a sentence. However, these models did not extract the *n-gram* features between different sentences. The presented *TextConvoNet* architecture uses a 2-dimensional convolutional filter and extracts text data's intra-sentence and inter-sentence n-gram features. Therefore, it results in a rich feature set for better text classification. A comprehensive evaluation of the presented *TextConvoNet* using five different datasets and eight performance measures showed that *TextConvoNet* produced a state-of-the-art performance for text classification. We summarize the main findings of the presented work as follows.

- We found that the presented *TextConvoNet* architecture produced a maximum average improvement in the performance of around 20% or more for different performance measures compared to the used machine learning models and deep learning-based models.
- When compared with the existing attention-based models and BERT model, we found that the presented *TextConvoNet* outperformed all the other attention and non-attention models for multiclass datasets (Datasets 3, 4, and 5). The performance of the *TexConvoNet* is approximately equal to or improved than the other models for binary class datasets.
- When the presented *TexConvoNet* is evaluated for the few-shot learning scenario, we found that the *TextConvoNet* produced lower test error rates than all the other baseline models with a lower proportion of training examples.

## 7 Conclusion and future work

This paper presented a convolutional neural network-based deep learning architecture for text classification. The important feature of the presented *TextConvoNet* is that it extracts the intra-sentence *n-gram* features from the text data and also extracts the inter-sentence *n-gram* features. We used the 2-D CNN model to provide an alternate input representation for text data (paragraph matrix). An extensive performance evaluation of the presented *TextConvoNet* architecture on five different text datasets has been done. The results showed that the presented *TextConvoNet* yielded better performance than the baseline machine learning models and state-of-the-art deep-learning-based models. The improved performance of *TextConvoNet* has been recorded for both binary-class and multi-class classification problems. The analysis showed that extracting the inter-sentence features along with the intra-sentence features improves the performance of the CNN model's text classification task. In future work, we will explore the idea of representing input in higher dimensions so that convolution operations can capture various features from the textual data.

## Appendix A

We have prepared a supplementary file, which includes the following details. The first part discusses the implementation of all the machine learning and deep learning models with the used values of the control parameters. This file is available in the GitHub repository[7].

## Appendix B: Used performance evaluation measures

**Table 9** Description of the performance evaluation metrics

| Measures | Formula | Description |
|---|---|---|
| Accuracy | $\frac{TP+TN}{TP+FP+TN+FN}$ | Accuracy refers to the amount of accurate assumptions the algorithm produces for forecasts of all sorts. |
| Precision | $\frac{TP}{TP+FP}$ | Precision is the percentage of successful cases that were reported correctly. |
| Recall | $\frac{TP}{TP+FN}$ | It is the number of right positive outcomes divided by the number of all related samples (including samples that were meant to be positive). |
| F1-score | $\frac{2\times P\times R}{P+R}$ | It is the harmonic mean of the precision and recall values. |
| MCC | $\frac{(TP*TN-FP*FN)}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}}$ | MCC is the correlation coefficient between the actual values of the class and the predicted values of the class. |
| Specificity | $\frac{TN}{TP+FN}$ | It is used to calculate the fraction of negative values correctly classified. |
| Gmean1 | $\sqrt{Precision \times Recall}$ | Gmean1 is computed as the square root of the product of precision and recall. |
| Gmean2 | $\sqrt{Specificity \times Recall}$ | Gmean2 is computed as the square root of the product of specificity and recall. |

[7] https://github.com/sonisanskar/TextConvoNet

## Declarations

**Conflict of Interests** The authors declare that they have no conflict of interest.

## References


1. Kowsari K, Meimandi KJ, Heidarysafa M, Mendu S, Barnes L, Brown D (2019) Text classification algorithms: a survey. Information 10(4):150
2. Minaee S, Kalchbrenner N, Cambria E, Nikzad N, Chenaghlu M, Gao J (2021) Deep learning–based text classification: a comprehensive review. ACM Comput Surveys (CSUR) 54(3):1–40
3. Wang SI, Manning CD (2012) Baselines and bigrams: simple, good sentiment and topic classification. In: Proceedings of the 50th annual meeting of the association for computational linguistics (vol 2: short papers), pp 90–94
4. Bozarth L, Budak C (2020) Toward a better performance evaluation framework for fake news classification. In: Proceedings of the international AAAI conference on web and social media, vol 14, pp 60–71
5. Parmar J, Soni S, Chouhan SS (2020) Owi: open-world intent identification framework for dialog based system. In: International conference on big data analytics, pp 329–343
6. Wang X, Qi K, An J, Zhou M (2019) Drifted twitter spam classification using multiscale detection test on kl divergence. IEEE Access 7:108384–108394
7. Scott S, Matwin S (1999) Feature engineering for text classification. In: ICML. Citeseer, vol 99, pp 379–388
8. Hadi W, Al-Radaideh QA, Alhawari S (2018) Integrating associative rule-based classification with naïve bayes for text classification. Appl Soft Comput 69:344–356
9. Ikonomakis M, Kotsiantis S, Tampakas V (2005) Text classification using machine learning techniques. WSEAS Trans Comput 4(8):966–974
10. HaCohen-Kerner Y, Miller D, Yigal Y (2020) The influence of preprocessing on text classification using a bag-of-words representation. Plos One 15(5):e0232525
11. Wang J, Wang Z, Zhang D, Yan J (2017) Combining knowledge with deep convolutional neural networks for short text classification. In: IJCAI, vol 350
12. Shi M, Wang K, Li C (2019) A c-lstm with word embedding model for news text classification. In: 2019 IEEE/ACIS 18th international conference on computer and information science (ICIS). IEEE, pp 253–257
13. Adhikari A, Ram A, Tang R, Lin J (2019) Rethinking complex neural network architectures for document classification. In: Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, pp 4046–4051
14. Yang P, Xu S, Li W, Ma S, Wu W, Wang H (2018) Sgm: sequence generation model for multi-label classification. In: Proceedings of the 27th international conference on computational linguistics, pp 3915–3926
15. Duque AB, Santos LJ, Macêdo D, Zanchettin C (2019) Squeezed very deep convolutional neural networks for text classification. In: International conference on artificial neural networks. Springer, pp 193–207
16. Minaee S, Kalchbrenner N, Cambria E, Nikzad N, Chenaghlu M, Gao J (2021) Deep learning–based text classification: a comprehensive review. ACM Comput Surveys (CSUR) 54(3):1–40
17. Zhou P, Shi W, Tian J, Qi Z, Li B, Hao H, Xu B (2016) Attention-based bidirectional long short-term memory networks for relation classification. In: Proceedings of the 54th annual meeting of the association for computational linguistics, pp 207–212
18. Yang Z, Yang D, Dyer C, He X, Smola A, Hovy E (2016) Hierarchical attention networks for document classification. In: Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies, pp 1480–1489
19. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. CoRR arXiv:1706.03762
20. Luong M-T, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation. CoRR arXiv:1508.04025
21. Wang Y, Huang M, Zhu X, Li Z (2016) Attention-based lstm for aspect-level sentiment classification. In: Proceedings of the 2016 conference on empirical method+s in natural language processing, pp 606–615
22. Kalchbrenner N, Grefenstette E, Blunsom P (2014) A convolutional neural network for modelling sentences. In: Proceedings of the 52nd annual meeting of the association for computational linguistics, pp 655–665
23. 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
24. Liu J, Chang W-C, Wu Y, Yang Y (2017) Deep learning for extreme multi-label text classification. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, pp 115–124
25. Effective use of word order for text categorization with convolutional neural networks (2015) In: Proceedings of the 2015 conference of the north american chapter of the association for computational linguistics: human language technologies, pp 103–112
26. 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 (vol 1: long papers), pp 562–570
27. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Proceedings of 3rd international conference on learning representations ICLR, pp 1–14
28. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
29. Conneau A, Schwenk H, Cun YL, Barrault L (2017) Very deep convolutional networks for text classification. In: Proceedings of the 15th conference of the european chapter of the association for computational linguistics, vol 1, long papers. Association for computational linguistics, pp 1107–1116
30. Le H, Cerisara C, Denis A (2018) Do convolutional networks need to be deep for text classification? In: Proceedings of the Workshops at the 32nd AAAI Conference on Artificial Intelligence, pp. 1–8.
31. Huang G, Liu Z, Maaten LVD, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
32. Liu Y, Li P, Hu X (2022) Combining context-relevant features with multi-stage attention network for short text classification. Comput Speech Language 71:101268
33. Liu J, Ma H, Xie X, Cheng J (2022) Short text classification for faults information of secondary equipment based on convolutional neural networks. Energies 15(7):2400

34. Akhter MP, Jiangbin Z, Naqvi IR, Abdelmajeed M, Fayyaz M (2022) Exploring deep learning approaches for urdu text classification in product manufacturing. Enterprise Inform Syst 16(2):223–248
35. Alsaleh D, Larabi-Marie-Sainte S (2021) Arabic text classification using convolutional neural network and genetic algorithms. IEEE Access 9:91670–91685
36. Ibrahim MA, Khan MUG, Mehmood F, Asim MN, Mahmood W (2021) Ghs-net a generic hybridized shallow neural network for multi-label biomedical text classification. J Biomed Inform 116:103699
37. Yang DU, Kim B, Lee SH, Ahn YH, Kim HY (2022) Autodefect defect text classification in residential buildings using a multi-task channel attention network. Sustainable Cities Soc, p 103803
38. Choudhary M, Chouhan SS, Pilli ES, Vipparthi SK (2021) Berconvonet: a deep learning framework for fake news classification. Appl Soft Comput 110:107614
39. Jacovi A, Shalom OS, Goldberg Y (2018) Understanding convolutional neural networks for text classification. In: Proceedings of the 2018 EMNLP workshop blackboxNLP: analyzing and interpreting neural networks for NLP, pp 56–65
40. Wang Y, Sohn S, Liu S, Shen F, Wang L, Atkinson EJ, Amin S, Liu H (2019) A clinical text classification paradigm using weak supervision and deep representation. BMC medical informatics and decision making 19(1):1
41. Kim H, Jeong Y-S (2019) Sentiment classification using convolutional neural networks. Appl Sci 9(11):2347
42. Akhter MP, Jiangbin Z, Naqvi IR, Abdelmajeed M, Mehmood A, Sadiq MT (2020) Document-level text classification using single-layer multisize filters convolutional neural network. IEEE Access 8:42689–42707
43. Merdivan E, Vafeiadis A, Kalatzis D, Hanke S, Kroph J, Votis K, Giakoumis D, Tzovaras D, Chen L, Hamzaoui R et al (2019) Image-based text classification using 2d convolutional neural networks. In: 2019 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computing, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation, pp 144–149
44. Diener MJ (2010) Cohen's d. The Corsini encyclopedia of psychology, 1–1. Wiley Online Library
45. Murphy KP et al (2006) Naive bayes classifiers. Univ British Columbia, vol 18(60)
46. Rasoul Safavian S, Landgrebe D (1991) A survey of decision tree classifier methodology. IEEE Trans Syst Man Cybern 21(3):660–674
47. Liaw A, Wiener M et al (2002) Classification and regression by randomforest. R news 2(3):18–22
48. Sathiya Keerthi S, Shevade SK, Bhattacharyya C, Murthy KRK (2000) A fast iterative nearest point algorithm for support vector machine classifier design. IEEE Trans Neural Netw 11(1):124–136
49. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Annals Stat, pp 1189–1232
50. Zhang S, Li X, Zong M, Zhu X, Cheng D (2017) Learning k for knn classification. ACM Trans Intell Syst Technol (TIST) 8(3):1–19
51. Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H et al (2015) Xgboost: extreme gradient boosting. R Package Ver 1(4):0.4–2
52. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Computat 9(8):1735–1780
53. Kenton JDM-WC, Toutanova LK (2019) Bert: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT, pp 4171–4186
54. Li E (2021) Densely connected bidirectional lstm with max-pooling of cnn network for text classification. In: Advanced data mining and applications: 16th international conference, ADMA 2020, Foshan, China, 12–14 November 2020, Proceedings. Springer Nature, vol 12447, p 98
55. Deng J, Cheng L, Wang Z (2021) Attention-based bilstm fused cnn with gating mechanism model for chinese long text classification. Comput Speech Language 68:101182
56. Wan C-X, Li B (2022) Financial causal sentence recognition based on bert-cnn text classification. J Supercomput 78(5):6503–6527
57. Huang L, Ma D, Li S, Zhang X, Wang H (2019) Text level graph neural network for text classification. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp 3444–3450
58. Ke Z, Huang L, Song R, Shen Q, Xu H (2021) A sequential graph neural network for short text classification. Algorithms 14(12):352
59. Yan L, Zheng Y, Cao J (2018) Few-shot learning for short text classification. Multimed Tools Appl 77(22):29799–29810