Supplementary file

This file contains the following information. The first part contains details of the machine learning and deep learning techniques with the values of the control parameters. Next, we present the additional results and discussion that are not reported in the paper.

1. Machine learning and deep learning models

1.1. The results of machine learning models were calculated using the following parameters.

Table 1: Parameter values set for different machine learning techniques

Technique	**Parameter values**
K-Nearest Neighbour (KNN)	n_neighbors = 10, weights = ['uniform'], metric = ['minkowski']
Multinomial Naive Bayes	With default parameter in scikit learn
Decision Tree (DT)	criterion:[’Gini’] ,splitter:[‘best’]
Random Forest (RF)	n_estimators=100, min_sample_split=2 ,max_features=’auto’
SVC	kernel=[‘rbf’], degree:3, gamma=[‘scale’]
Gradient Boosting	learning_rate=0.1,max_depth=5,max_features=0.5
XGBoost	booster=[‘gbtree’], learning_rate: 0.3

1.2. The internal implementation details of the deep learning models used are as follows

I. **KIM’ CNN model**: In the Kim's model, sentences are matched to the embedding vectors that are made available as an input matrix to the model. It makes use of 3 parallel layers of convolution with word vectors on top obtained from existing pretrained models, with 100 filters of 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, which are subsequently sent to a fully connected layer of 64 neurons and then to the classification layer.

II. **LSTM**: The most standard model for text classification purposes. For the comparison with our model we used a single LSTM layer with 32 memory cells in it followed by the classification layer.

III. **CLSTM**: In this model, we stack the LSTM network on the top of a continuous CNN network. We use 3 consecutive layers of CNN with kernel sizes of 10, 20, 30 and filters to be 64. The memory dimension of LSTM is also set to be 64. The LSTM layer was followed by a dense layer of 128 neurons which in turn is succeeded by the classification layer.

IV. **VDCNN**: This model checks if increasing the depth of Convolution networks improves performance or not. The 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. There is a possibility to use a shortcut connection between the convolutional blocks, but we do not use it because the results show no benefit in evaluation. We utilize a max pooling method to downsample to a fixed dimension after a pair of convolution blocks. 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.

V. **BiLSTM +ATTENTION:** It starts with an input layer which Tokenizes input sentences and indexed lists followed by an embedding layer. There exists bidirectional LSTM cells (100 hidden units) which can be concatenated to get representation of each token. Attention weights are taken from linear projection and non-linear activation. Final sentence representation is the weighted sum of all token representations. Final classification output is derived from a simple dense and softmax layer.

VI. **BERT:** We used pretrained bert-base-uncased embeddings to encode followed by a dense layer of 712 neurons ended by a classification layer.

VII. **HAN:** We set the word embedding dimension to 300 and the GRU dimension to 50 in our experiment. 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.

2. Additional results

2.1. Comparison of *TextConvonet* with the results of BERT, BiLSTM+Attention, and HAN. Additional Results on extra dataset.

DATASET-1 (sst2)									
	Accuracy	**Precision**	**Recall**	**F1_score**	**Specificity**	**Sensitivity**	**Gmean 1**	**Gmean 2**	**MCC**
BiLSTM+Attention	0.8066	0.816	0.7909	0.803	0.8223	0.7909	0.803	0.806	0.6136
BERT	0.7764	0.7367	0.859	0.7932	0.694	0.859	0.7956	0.772	0.56
HAN	0.8028	0.7913	0.8217	0.80626	0.7839	0.821	0.806	0.802	0.606
TextConvoNet_6	0.822	0.848	0.783	0.814	0.86	0.783	0.815	0.821	0.645
TextConvoNet_4	0.819	0.833	0.798	0.815	0.841	0.798	0.815	0.819	0.639
DATASET-2 (AMAZON REVIEWS)									
BiLSTM+Attention	0.886	0.8834	0.8465	0.864	0.9157	0.8465	0.8648	0.8804	0.7667
BERT	0.772	0.6856	0.8674	0.76591	0.7	0.8674	0.7712	0.7792	0.564
HAN	0.863	0.8805	0.7883	0.8319	0.9192	0.7883	0.8331	0.8513	0.72
TextConvoNet_6	0.904	0.905	0.867	0.886	0.932	0.867	0.886	0.899	0.804
TextConvoNet_4	0.872	0.819	0.902	0.858	0.849	0.902	0.859	0.875	0.745
DATASET-3(OHsumed)									
BiLSTM+Attention	0.99	0.96	0.9602	0.9602	0.99432	0.96025	0.9602	0.9771	0.9545
BERT	0.956487	0.8259	0.82594	0.82594	0.97513	0.82594	0.82594	0.8974	0.801
HAN	0.8901325	0.5605	0.5605	0.56052	0.9372	0.5605	0.5605	0.7248	0.4977
TextConvoNet_6	0.992	0.968	0.968	0.968	0.995	0.968	0.968	0.981	0.963
TextConvoNet_4	0.992	0.969	0.969	0.969	0.996	0.969	0.969	0.982	0.965
DATASET-4 (AIRLINE TWITTER SENTIMENT)									
BiLSTM+Attention	0.8619253	0.7928	0.7928	0.79288	0.896	0.7928	0.7928	0.843	0.68933
BERT	0.808	0.712	0.712	0.712	0.856	0.712	0.712	0.7806	0.568
HAN	0.7955556	0.69333	0.6933	0.69333	0.84666	0.6933	0.6933	0.7661	0.54
TextConvoNet_6	0.884	0.826	0.826	0.826	0.913	0.826	0.826	0.869	0.739
TextConvoNet_4	0.873	0.809	0.809	0.809	0.904	0.809	0.0.809	0.855	0.713
DATASET-5(NLP-COVID TEXT CLASSIFICATION)									
BiLSTM+Attention	0.7114667	0.2786	0.2786	0.27866	0.81966	0.27866	0.2786	0.4779	0.0983
BERT	0.7114667	0.2786	0.27866	0.27866	0.81966	0.27866	0.27866	0.47792	0.0983
HAN	0.7	0.25	0.25	0.25	0.8125	0.25	0.25	0.45069	0.0625
TextConvoNet_6	0.839	0.597	0.597	0.597	0.899	0.597	0.597	0.732	0.496
TextConvoNet_4	0.828	0.571	0.571	0.571	0.893	0.571	0.571	0.714	0.463
DATASET-6 (YELP REVIEWS) BINARY									
TextConvoNet_6	0.92	0.90286	0.91376	0.90828	0.920347	0.913762	0.90829	0.91704	0.8332
TextConvoNet_4	0.91	0.89538	0.91018	0.90272	0.91383	0.91018	0.90275	0.912	0.8228

KIM's CNN MODEL	0.91	0.86963	0.94191	0.90433	0.88559	0.941912	0.90505	0.91331	0.8233
VDCNN	0.9034	0.8994	0.88293	0.89109	0.919985	0.882931	0.89113	0.9012	0.8044
HAN	0.8938	0.87189	0.8941	0.88285	0.89355	0.8941	0.88292	0.89382	0.7859

The detailed description of Datasets 1-5 have been provided in the main paper [Table II]. We further calculated results of some models on Dataset 6 as well. The description of Dataset 6 is as follows:

DATASET 6 [YELP REVIEWS DATASET]
{Link: https://www.kaggle.com/ilhamfp31/yelp-review-dataset }
We included random 20000 training instances and 5000 testing instances. The Yelp reviews dataset consists of reviews from Yelp extracted from Yelp Dataset Challenge 2015. The Yelp reviews polarity dataset is constructed by considering stars 1 and 2 negative, and 3 and 4 positive.

2.2. Selection of performance metrics

The reason for choosing Accuracy, Precision, Recall, F1_Score, Specificity, GMean1, GMean2, MCC as the performance metrics due to the wide variety of applications of NLP in the current scenario. There are many cases where precision is given more preference over accuracy and similar for other performance metrics. So, we evaluated our model *TextConvoNet* on a multitude of performance metrics in order to get a wholesome analysis of the model and check its viability for each and every NLP application. For instance, in detecting a lethal disease through symptoms False-negative is a crime. So here Recall is much better than precision.

Accuracy: accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost the same. Therefore, you have to look at other parameters to evaluate the performance of your model.

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false-positive rate. Precision is a good measure to determine when the costs of False Positive is high. For instance, email spam detection. In email spam detection, a false positive means that an email that is non-spam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall: Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative. For instance, in sick patient detection. If a sick patient (Actual Positive) goes through the test and is predicted as not sick (Predicted Negative). The cost associated with False Negative will be extremely high if the sickness is contagious.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

F1_Score: F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is

usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Specificity: **Specificity is used if you want to cover all true negatives**, meaning you don't want any false alarms, you don't want any false positives. For example, you're running a drug test in which all people who test positive will immediately go to jail, you don't want anyone drug-free going to jail. False positives here are intolerable.

$$\text{Specificity} = \frac{\text{True negative}}{\text{True negative + false positive}}$$

MCC: The Matthews correlation coefficient (MCC), instead, is a more reliable statistical rate that produces a high score only if the prediction obtained good results in all of the four confusion matrix categories (true positives, false negatives, true negatives, and false positives), proportionally both to the size of positive elements and the size of negative elements in the dataset. MCC takes into account all four values in the confusion matrix, and *a high value (close to 1) means that both classes are predicted well*, even if one class is disproportionately under- (or over-) represented.

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

2.3 Performance gain of *TextConvoNet* over other compared model w.r.t performance matrices.

	DATASET-1		DATASET-2		DATASET-3		DATASET-4		DATASET-5	
	Min (%)	**Max (%)**	**Min (%)**	**Max (%)**	**Min (%)**	**Max (%)**	**Min (%)**	**Max (%)**	**Min (%)**	**Max (%)**
Accuracy	0.3	42.4	2.7	32.1	0.2	13.5	0.3	33.5	4.3	33.5
Precision	2.4	24.9	-	-	0.9	95.7	0.6	67.5	17	166
Recall	-	-	-	-	0.9	95.7	0.6	67.5	17	166
F1 Score	-	-	1.1	13.7	0.9	95.7	0.6	67.5	17	166
Specificity	2.8	57.5	-	-	0.4	1.9	0.2	22.3	2.3	11.5
Gmean1			1	13.7	0.9	95.5	0.6	67.5	17	166
Gmean2	0.1	45	2.5	123	0.6	44.8	1	43.3	9.4	72.2
MCC	0.9	48.7	6.4	325	1	128	0.9	209	27.8	870

Few-Shot Learning :

Few-shot learning, as the name implies, is the process of feeding a learning model with a limited quantity of training data, as opposed to the more common method of using a large number of data. Deep learning techniques for few-shot learning are founded on the idea that trustworthy algorithms may be constructed to make predictions from little datasets and that is what TextConvoNet is capable of doing.