International Journal of Artificial Intelligence and Machine Learning
Volume 12 • Issue 1

Boosting Convolutional Neural Networks Using a Bidirectional Fast Gated Recurrent Unit for Text Categorization

Assia Belherazem, SIMPA Laboratory, Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf, Algeria*
https://orcid.org/0000-0003-3058-4062

Redouane Tlemsani, Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf, Algeria

ABSTRACT

This paper proposes a hybrid text classification model that combines 1D CNN with a single bidirectional fast GRU (BiFaGRU) termed CNN-BiFaGRU. Single convolution layer captures features through a kernel by applying 128 filters which are slid over these embeds to find convolutions and drop the entire 1D feature maps by using spatial dropout and combined vectors using max-pooling layer. Then, the bidirectional CUDNNGRU block is used to extract temporal features. The results of this layer are normalize by the batch normalization layer and transmitted to the fully connected layer. The output layer produces the final classification results. Precision/loss score was used as the main criterion on five different datasets (WebKb, R8, R52, AG-News, and 20 NG) to assess the performance of the proposed model. The results indicate that the precision score of the classifier on WebKb, R8, and R52 data sets significantly improved from 90% up to 97% compared to the best result achieved by other methods such as LSTM and Bi-LSTM. Thus, the proposed model shows higher precision and lower loss scores than other methods.

KEYWORDS

Convolutional Neural Networks, Deep Neural Network, Gated Recurrent Unit, Natural Language Processing, Text Classification

INTRODUCTION

This paper proposes an automatic approach to categorizing text using deep learning. Text categorization is the associating documents process to predefined classes (categories or labels) written in natural language using natural language processing (NLP). Many researchers have used text classification with deep learning architectures that assure high precision with less need for engineering features. The key aspect of deep learning is that the resultant layers of features are not designed by human engineers, but, rather, are learned from data using a general-purpose learning procedure.

In particular, the recurrent neural network (RNN*)* is a very powerful dynamic system and an important implementation mechanism of deep learning. The RNN method can find the dependencies

DOI: 10.4018/IJAIML.308815 *Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

relationship of time series that provide more effective ways for time memory to operate. Loop memory can extract valuable information from the history data through memory cell execution and other control mechanisms. The long short-term memory (LSTM) and gated recurrent units (GRUs) are two kinds of special memory cells of RNN that employ different memory cell mechanisms. LSTM and GRU networks use special hidden units whose natural function is remembering inputs for a long time (Hochreiter & Schmidhuber, 1997). However, regarding a load of power data with obvious time series and cycles characteristics, load forecasting can take advantage of history information via the LSTM and the GRU cell(Zhang,Wu et al., 2018).

This paper examines the multiclass automatic classification applying a hybrid approach by integrating convolution neural network (CNN) and bidirectional fast gated recurrent unit (BiFaGRU*)* termed as CNN-BiFaGRU. CNN-BiFaGRU is a supervised text classification testing on different textual databases (Reuters8, Reuters52, WebKB, 20NewsGroup, and AG NEWS) using the GloVe word embedding proposed by Pennington et al. (2014). The model is evaluated using different metrics such as accuracy, precision, recall, F1-score, and the confusion matrix. The obtained results are detailed in the section of results and discussion.

The main contributions of this paper are summarized as follows:

- The implementation of a new model using a 1D CNN followed by a single bidirectional CuDNNGRU performs the classification.
- The use of both CNN and Bi-CUDNNGRU maximizes the potential of the text representation with the capability to generate complex content sequences with minimal storage requirements.
- Experiments on five commonly used datasets demonstrate that the proposed model yields remarkable computing time and precision performance with a low loss against state-of-the-art methods.

The rest of the paper is organized as follows: The second section represents the related work; the third section defines the proposed model, how it works, how the authors implemented the hybrid model and the implementation of other four models to compare them with their best model; The fourth section focuses on the problem statement, description of the datasets, and the exploratory data analysis and settings used to solve the problem; in addition, the fourth section explains in detail how the data were prepared and represented, which word emblems were used, how the dataset was divided for training and testing, and which evaluation criteria were used to assess the performance of the proposed model; The fifth section presents the results using various words embedding; the sixth section discusses the results obtained; finally, the seventh section concludes the paper.

STATE OF THE ART

Many approaches have been proposed in the past few years. Johnson and Zhang's (2015b)ConvNets model or char-level CNN applies only to a character. It can work in different languages (Johnson & Zhang, 2015a). The Kim's (2014)TextCNN model uses the Word2vec (Mikolov et al., 2013). This architecture is a variant Collobert et al.'s (2011)CNN architecture; it is training only on labeled data.

Yang et al.(2016) explored LSTM encoders for text classification tasks, the researchers proposed a hierarchical attention network for document classification, also Liu et al. (2016) proposed LSTM, and Bi-LSTM, the LSTM uses the last hidden state to represent the text and a bidirectional LSTM, commonly used in text classification with pretrained word embeddings. LSTMs are similar to BiLSTMs in their recurrent bidirectional message flow between words, but different in the design of state transition. Dai and Le (2015) proposed two approaches: LM-LSTM is to predict what comes next in a sequence, and SA-LSTM used a sequence auto-encoder, which reads the input sequence into a vector and predicts the input sequence again.

Guo et al.'s (2018) CRAN model joined the CNN and RNN effectively with the help of the attentive mechanism (NVIDIA Developer. 2022). Zhang,Li et al.(2018)proposed another hybrid model,where the LSTM can preserve the historical information characteristics in long text sequences and extract local features of text by using CNN. The TextGCN (Yao et al., 2019) builds a single text graph for a corpus-based on word cooccurrence and document word relations using the Bi-LSTM-CNN method. In the same context, Song et al. (2022) constructed two different graphs based on contextual information, called sentence graphs and corpus graphs, respectively. Yang et al. (2016) employed a significant comprehensive expression to express semantics accurately. Zulqarnain et al. (2019) proposed a unified structure that was implemented to investigate the effects of word embedding and GRU for text. Dai and Le's (2015) proposed GRU model can effectively learn the word usage in the context of texts provided training.

Ren et al. (2021)introduced the BG-TCA model, which uses the bidirectional temporal convolution network (TCN) to extract bidirectional temporal features in text data. A gating mechanism similar to the LSTM is added between the convolution layers. In the feature aggregation stage, the model uses the attention mechanism to replace the max-pooling method. In the same context, Liu et al. (2022) proposed a novel short text classification approach, CRFA, combining context-relevant features with a multistage attention model based on temporal TCN and CNN. Li et al. (2020) filled the gap by reviewing state-of-the-art approaches from 1961 to 2020, focusing on shallow to deep learning models. Moreover, Malekzadeh et al. (2021) reviewed the most recent state-of-the-art graph-based text classification, datasets, and performance evaluations vs. baseline models.

PROPOSED MODEL

This work presents a CNN-BiFaGRU model, mainly a combination of 1D CNN with a single BiFaGRU. The overall model consists of five parts (Figure 3). The details of each component are described in the following subsections.

Input Layer

The input layer transforms the text into an embedding matrix (Figure 3). The sentence matrix consists of 250 words, represented as s for the maximum sequence length. If the text is not long enough, the authors will use 0 as padding. Each word is represented as a d-dimension vector pretrained by Word2vec embedding (Mikolov et al., 2013), Glove embedding (Pennington et al., 2014) or One-Hot Encoding (Brownlee, 2017), where $d = \{100,200,300\}$. Therefore, the sentence can be represented as a feature map of dimension $d \times s$.

Convolution Layer

A convolution is a linear operation that involves the multiplication of a set of weights with the input. The multiplication is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel. The proposed model performs convolution and pooling operations. The single convolution layer 1D captures features through a convolution kernel with a window size 3×3. The convolution kernel is applied to each possible word window in the sentence applying 128 filters producing a 128-dimensional vector for each document. Then, the authors slide the filter /kernel over these embeds to find convolutions. To drop entire 1D feature maps instead of individual elements, the authors used Li et al.'s (2020)SpatialDropout 1D. The vectors are combined using the Max-Pooling 1D Layer, which takes the maximum value over the window, using ReLu activation.

Bidirectional Fast GRU Layer

GRU Layer

The RNN is a powerful dynamic system and an important implementation mechanism of deep learning. It is an extension of a traditional neural network, which can handle a variable-length sequence input. The variable-length sequence is solved by a recurrent hidden layer whose activation at each time in RNN. A GRU is a modified version of the general RNN and a simple variation of LSTMs with fewer parameters (Figure 1). The GRU unit has two gates, as follows: Update z_t and reset r_t. The update gate determines if the hidden state will be updated with a new hidden state or not. If yes, it will be computed by Eq. (1), while the reset gate, which decides if the previous hidden state will be ignored, is calculated by Eq. (2).The hidden layer h_t is computed by Eq. (4) using H_t, which is calculated by Eq. (3).

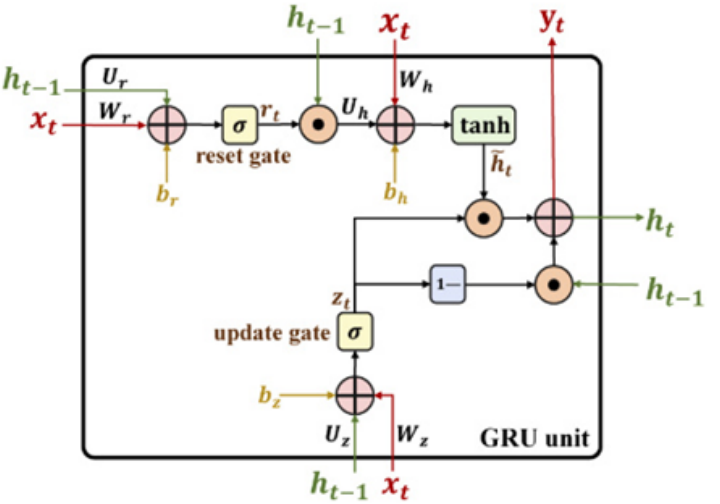
Given that $x = \{x_1, x_2, \ldots, x_t\}$ and $y = \{y_1, y_2, \ldots, y_t\}$ are the input and output layers of sequence, h_t is a hidden layer. In the GRU, the authors use model parameters, where, x_t is the input at a time, and weight matrices are denoted by W_z, W_r, W_H, U_z, U_r, U_H and their outputs are z_t and r_t, respectively. b_r, b_z, b_h are the synthesis of bias vectors for input *xt* and previous states h_{t-1}; σ is the logistic sigmoid function, tanh is the hyperbolic tangent activation function, $\odot$ denotes the Hadamard product. The detailed operations of the GRU unit are illustrated in Eqs. (1)-(4):

$$\mathcal{Z} = \sigma(\mathcal{W}_z x_t + \mathcal{U}_z h_{t-1} + b_z) \tag{1}$$

$$r_t = \sigma(w_r x_t + \mathcal{U}_r h_{t-1} + b_r) \tag{2}$$

$$\mathcal{H}_t = \tanh(\mathcal{W}_{\mathcal{H}} x_t + \mathcal{U}_{\mathcal{H}} \left(r_t h_{t-1}\right) + b_{\mathcal{H}}) \tag{3}$$

Figure 1. A detail structure of gated recurrent unit



$$h_t = (1 - z_t)h_{t-1} + z_t \mathcal{H}_t \tag{4}$$

z_t represents the update gate operation. Using a sigmoid function, the authors decide which previous information to pass through. H_t denotes the reset gate operation. The authors multiply the connection value of the previous and current time steps by r_t. h_t is the new memory. This will produce the values the authors want to discard from the previous step.

Bidirectional GRU

Bidirectional GRU is a bidirectional RNN with only the input and forgets gates. It allows using information from previous and later steps to make predictions about the current state(Silwimba, 2018). Figure 2 presents the structure of the Bi-GRU model diagram, and it is defined as follows:

$$\overrightarrow{h_t} = GRU_{forward}\left(x_t, \overrightarrow{h_{t-1}}\right) \tag{5}$$

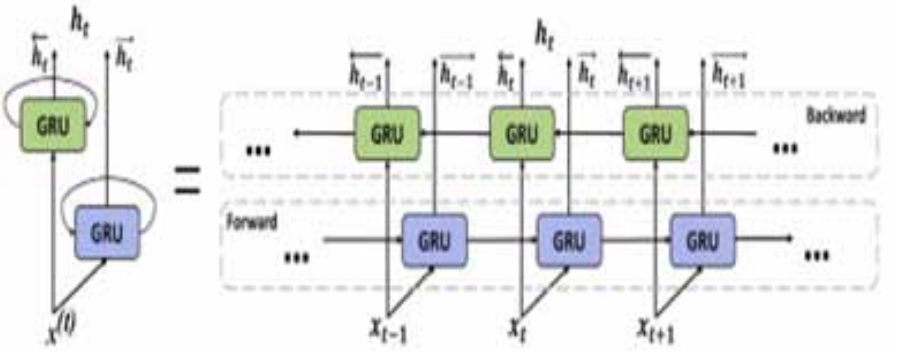
$$\overleftarrow{h_t} = GRU_{backward}\left(x_t, \overleftarrow{h_{t-1}}\right) \tag{6}$$

$$h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t} \tag{7}$$

where $\overrightarrow{h_t}$ is the state of the forward GRU, $\overleftarrow{h_t}$ is the state of the backward GRU, and $\oplus$ indicates the operation of concatenating two vectors.

Figure 2 illustrates two GRU cells. One GRU that moves forward is the normal input sequence beginning from the start of the data sequence, while the other GRU that moves backward is the same input sequence in reverse order, beginning from the end of the data sequence. To maximize the performance, the authors choose the CuDNN implementation. The original GRU implementation is compatible with CuDNNGRU (GPU only) and allows inference on the CPU. The CuDNN requirements are: The activation function is Tanh, Sigmoid for recurrent_activation and recurrent_dropout equal to

Figure 2. Structure of bidirectional GRU



zero, the unroll is False, use_bias is True (the layer uses a bias vector), and reset_after is True (apply reset gate after matrix multiplication) (Keras, 2021).

Batch Normalization Layer

Batch normalization standardizes the inputs to a layer for each mini batch. This has the effect of stabilizing the learning process and reducing the number of training epochs required to train deep networks(Brownlee,2019).This layer applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1to normalize its input (Keras, 2021b).

Dense Layer

The dense layer is fully connected. All the layers' neurons are connected to those in the next layer. The authors use this layer to control the size and shape of the output layer. The number of units in the fully connected output layer will equal the number of classes, with a Softmax activation function used to create a distribution over classes.

Output Layer

Finally, the output layer produces the result. Since the authors have a classification task, they use the Softmax activation. Figure 3 illustrates the proposed CNN-BiFaGRU model diagram. Table 1 lists the structural parameters of the authors' CNN-BiFaGRU model using 300 as input size. The dropout and spatial dropout have a dropout probability of 0.1, sequence length fixed to 250, and the number of filters (channels) to 128.

The network is trained using categorical cross-entropy as the loss function with different pretrained embeddings: Glove, Word2Vec embeddings, and One-Hot Encoding. Padding same (output size is the same as input size by padding evenly left and right) and stride equal to one.

DATASETS AND EXPERIMENTAL SETUP

Many datasets are actively used for research in text categorization. The authorstested the proposed model on various benchmarks.

Datasets

The authors ran their experiments on five widely used benchmark corpora, including 20-Newsgroups (20NG), AG-News (AG), R8 and R52 of Reuters 21578, and WebKb.

20-NewsGroups

The 20NG dataset contains 18,846 documents, evenly categorized into 20 different categories. The training set contains 11,314 documents, and the test set (Cardoso-Cachopo, 2007) includes 7,532.

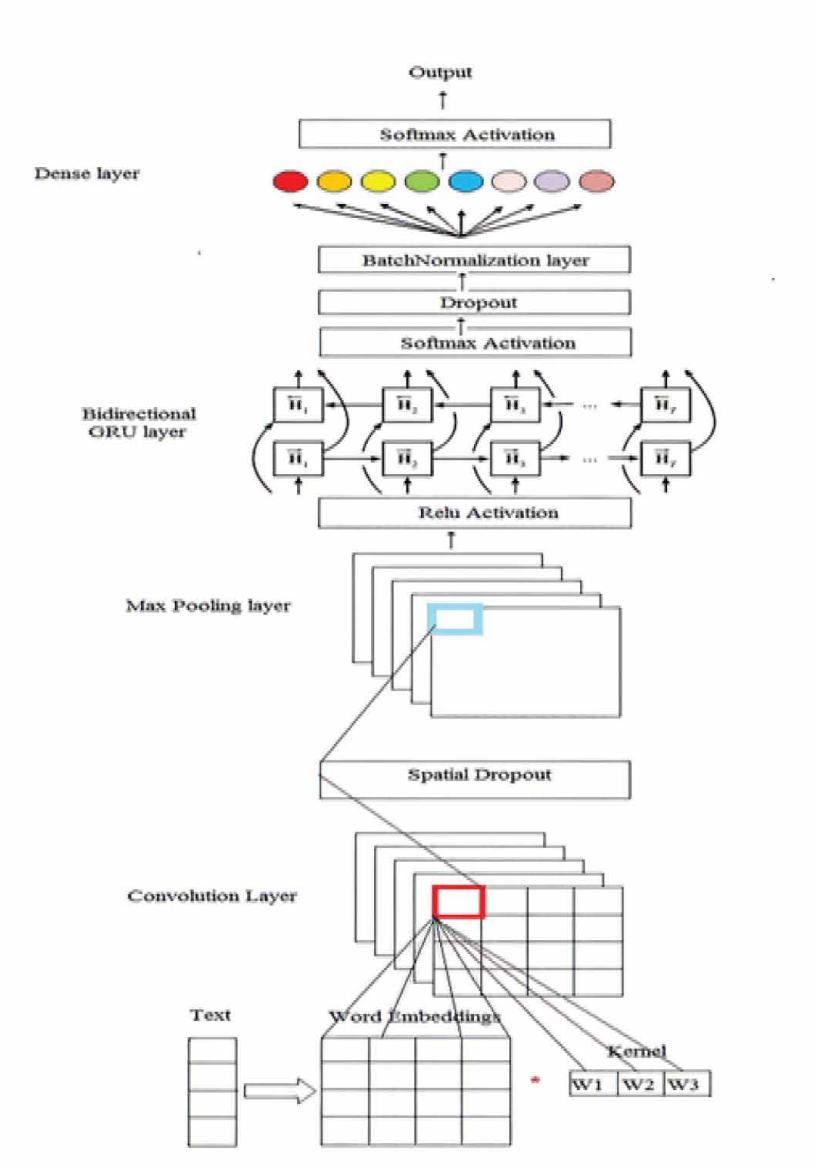
AG-News

AG-News is a topical classification dataset from over 2000 sources. Xiang Zhang built it in its third version, and the last update was on September 9, 2015. The AG has four categories and was split into 1,200,000 documents in the training set and 7,600 documents in the test set(Zhang et al., 2015).

R52 and R8

R52 and R8are two subsets of the Reuters 21,578 datasets. R8 has eight categories and was split into 5,485 documents in training and 2,189 test documents. R52 has 52 categories and was split into 6,532 documents in training and 2,568 test documents(Zhang et al., 2015).

Figure 3. Full diagram of CNN-BiFaGRU model



WebKb

WebKb documents are Web pages compiled by the World Wide Base project of the text learning group CMU. They were collected from the IT departments of various universities in 1997 and manually classified into four different classes. WebKb was split into 4,199 documents in training and 2,803 test documents(Zhang et al., 2015).

Table3 summarizes the general information about these five datasets. Firstly, the authors cleaned their datasets and tokenized text and sentences into words. They removed stop words defined in natural language toolkit (NLTK) and low-frequency words appearing less than 10 times for AG. Each word

Table 1. Structural parameters of CNN-BiFaGRU model

Layer	Number of Filters	Padding	Activation Shape	Activation Size
Input	-	-	(300,300,1)	90000
Embedding	-	-	(300,300,1)	90000
Conv1D (f=3,s=1)	128	Same	(300,300,128)	11520000
SpatialDropout1D	-	Same	(300,300,128)	11520000
MaxPool (f=2,s=1)	-		(150,150,128)	11520000
BiFaGRU	-	-	(64,1)	64
Softmax	-	-	(64,1)	64
Dropout	-	-	(64,1)	64
Batch normalization	-	-	(64,1)	64
Dense	-	-	(m,1)	8
Softmax	-	-	(m,1)	8

Table 2. Proposed models parameters

Models	Input Size	Optimization	Mini Batch
CNN-BiFaGRU GloVe300	300x300	Adaptive moment estimation (Adam)	32
CNN-BiFaGRU Word2Vec200	200x200	Adam	32
CNN-BiFaGRU One-Hot	100x100	Adam	32

Table 3.Summary statistics of AG-NEWS,20NG, R52, R8, and WebKB datasets

Dataset	Class	TotalDoc	Train Size	Test Size
AG-NEWS	4	127600	120000	7600
20NG	20	18821	11293	7528
R52	52	9100	6532	2568
R8	8	7674	5485	2189
WEBKB	4	4199	2803	1396

has a One-Hot encoded vector; padding uses sequences with the same length and then passes the padded sequences as input to the embedding layer in the case of One-Hot encoding.

Text Representation

The text data must be converted into numbers to be used in machine learning, including neural networks.The embedding layer is mainly used in NLP problems. It is possible to use pretrained word embeddings such as GloVe and Word2Vec or train the embeddings using the Keras embedding layer.

Keras Embedding Layer

The embedding layer converts each word into a fixed-length vector of defined size. The resultant vector is dense with real values, instead of just 0 and 1. The fixed length of word vectors helps to represent words in a better way along with reduced dimensions. There are three parameters to the embedding: layer.input_dim is the vocabulary size, output_dim is the length of the vector for each word, and the input_length is the maximum length of a sequence (Saxena, 2020).While working with text data, it is necessary to train the embedding layer by creating a One-Hot encoded vector for each word to get

the correct word embeddings. One-Hot encoding is a representation of categorical variables as binary vectors. The categorical values are mapped to integer values. Each integer value is represented as a binary vector with zero values,except the integer index, which is marked with a 1 (Brownlee, 2017).

GlobalVectors for Word Representation (GloVe)

GloVe is an unsupervised learning algorithm for obtaining vector representations for words, developed by the Stanford NLP Group. The GloVe model is trained in the nonzero entries of a global word-word cooccurrence matrix, which tabulates how frequently words cooccur in a present corpus. Populating this matrix requires a single pass through the entire corpus to collect the statistics (Pennington et al., 2014).This embedding technique is based on factorizing a matrix of word cooccurrence statistics. The authors experimented on all dimensions of glove vectors (i.e., 100,200, and 300) and decided to work with 300-dimensional GloVe vectors as they provided the best results for their task.

Word2Vec

The authorsused the publicly available word2vec vectors trained on 100 billion words from GoogleNews. The vectors have dimensionality of 50,100, and 200.Word2Vec is a method to construct such an embedding. It can be obtained using the continuous bag-of-words (CBOW) architecture (Mikolov et al., 2013).The input layer receives words *Wn* as arguments, where the projection layer corresponds to an array of multidimensional vectors and stores the sum of several vectors. The output layer corresponds to the layer that outputs the results of the vectors from the projection layer.

The basic principle of CBOW involves predicting when a certain word appears via analyzing neighboring words. The CBOW projection layer projects all words to the same position. Thus, the vectors of all words maintain an average and share the positions of all words. The structure of CBOW exhibits the advantage of uniformly organizing the information distributed in the dataset. The authors trained the Word2vec model using the Gensim library with their dataset. A vector represents each word. Vectors are nothing but neurons' weights. Those weights are the authors' word embeddings or simply dense vector.

To get dense vectors, the authors used a few parameters as *sentences*(vocabulary), *min_count* (words that are infrequent will be ignored; default value is 5),*workers* (are the number of CPU Threads to use at once for faster training, size of Word Embeddings),*window*(maximum distance between the current and predicted word within a sentence is equal to 3 in the authors' case), and *iter* (the number of iterations or epochs)(Padawe, 2019).

Model Variation

CNN-BiFaGRU With GloVe Embedding

The authors used the pretrained vectors of Glove Embedding in different dimensions:Glove100, Glove-200, and Glove-300.The authors downloaded the zip file called glove.6B.zip from Pennington et al.'s (2014) study . Next, they built an embedding matrix that they loaded into an embedding layer. It must be a matrix of shape (max_words, embedding_dim), where each entry *i* contains the embedding_dim-dimensional vector for the word of index *i* in the reference word index (built during tokenization). Words not found in the embedding index will be all zeros. Then, the authors put their block CNN and then the bidirectional CuDNNGRU layer with 32 neurons.

CNN-BiFaGRU With Word2Vec (CBoW) Embedding

The authors used the pretrained vectors of Word2Vec Embedding as input with different dimensions, namely Word2Vec-50, Word2Vec-100, and Word2Vec-200 with Gensim library(Řehůřek, 2022) initiate word2vec model with author's vocabulary, size of word embedding is 50, the window is 3, 32 workers, and min-count equal to 1.A vector represents each word. Vectors are neurons' weights;

those weights are the authors' word embeddings and are the inputs of their CNN block and the bidirectional CuDNNGRU layer.

CNN-BiFaGRU With One-Hot Representation

The input of the model is Keras's (2021a) One-Hot representation embedding layer of. Each integer value is represented as a binary vector with zero values, except the integer index, which is marked with a 1. The CNN block is put on top, followed by the bidirectional CuDNNGRU layer.

Other Models

In order to compare the performance, the authors evaluated the following popular models:

- **1D-CNN*:** Convolutional neural network representing a stacked neural network with a one-dimensional convolution and pooling layers.
- **LSTM*:** Long Short-Term Memory is an artificial neural network that stores the input for a short period, usually a one-time step.
- **BiLSTM*:** The basic idea of bidirectional long short-term memory is to connect two LSTM hidden layers of opposite directions to the same output.
- **GRU*:** A gated recurrent unit is an LSTM without an output gate, which therefore fully writes the contents from its memory cell to the larger net at each time step. Table 4 lists the hyper-parameters of each model.

Table 4 indicates that every model begins with an input size equal to 250, and the embedding layer with a size of 100 uses the One-Hot representation. The authors used a filter with a size equal to 300 and a Kernel size equal to 3, and a simple dropout of 0.3 using ReLu and Softmax as activation functions for the CNN* . The authors used no dropout in the GRU* model, and the activation function is Sigmoid like in LSTM*.With the LSTM* model, three kinds of dropout are used: The simple one equal to 0.25, the spatial and recurrent dropout equal to 0.2 for both. The BiLSTM* model uses just two kinds of dropout: The simple and recurrent one equal to 0.2 for both, but the activation function is ReLu.

Performance Metrics

The researchers defined several metrics to evaluate the efficiency of the proposed model (accuracy, classification error, precision, recall, F1-score) using the confusion matrix (Table5).

The numbers along the major diagonal represent the correct decisions made, and the diagonal numbers represent the errors (i.e., the confusion between the various classes). The confusion matrix

Table 4. The hyper-parameters of the 1D-CNN*, LSTM*,BiLSTM*, and GRU* models

Model	Input Size	Embedding Size	Filter Size	Kernel Size	Activation Function	Units Number	Dropout
CNN*	250	100	300	3	ReLu,Softmax	100	0.3
LSTM*	250	100	-	-	Sigmoid	100	Spatial dropout: 0.2 dropout:0.25 recurrent dropout:0.2
BiLSTM*	250	100	-	-	ReLu	100	Recurrent dropout:0.2 Dropout=0.2
GRU*	250	100	-	-	Sigmoid	100	-

Table 5. The confusion matrix of the performance evaluation for binary classification

	Predicted Class	
Actual Class	**Categorized Positive**	**Categorized Negative**
Actual positive	TP	FN
Actual negative	FP	TN

uses the four kinds of results the authors discussed above: True-positive(TP),true-negative(TN),false-positive (FP)or false-negative(FN). Then, precision, recall, and F1-score(Fawcett, 2006) are calculated from the confusion matrix using the following formulas:

$$\text{Precision}= \frac{TP}{TP + FP} \tag{8}$$

$$\text{Recall}= \frac{TP}{TP + FN} \tag{9}$$

$$\text{Accuracy}= \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$\text{F1-Score}= \frac{2 * precision * recall}{precision + recall} \tag{11}$$

The categorical cross-entropy loss measures the dissimilarity between the true label distribution y and the predicted label distribution $\hat{y}$, and is defined as cross-entropy:

$$\text{Categorical\_CrossEntropy } (y, \hat{y}) = - \sum_{i}^{outputsize} y_i \log\left(\hat{y}_i\right) \tag{12}$$

where, $\hat{y}_i$ is the i-th scalar value in the model output, y_i is the corresponding target value, and the output size is the number of scalar values in the model output. The categorical cross-entropy is well suited to classification tasks, since one example can be considered to belong to a specific category with probability 1 and other categories with probability 0 (Koidl, 2013).

RESULTS

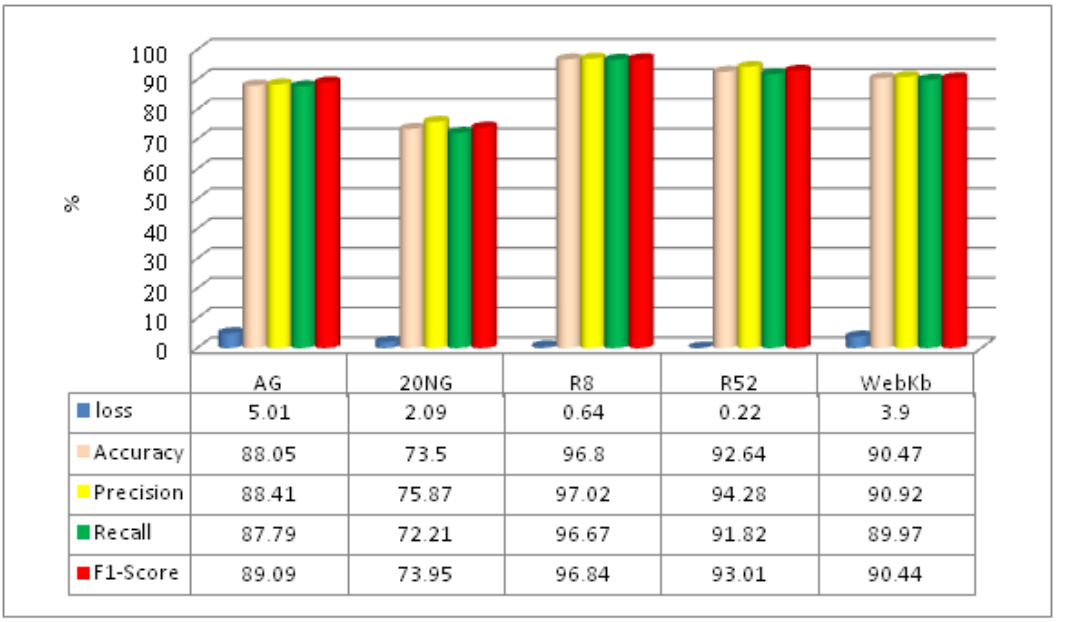
To assess the impact of each contribution, the authors performed a series of analyses. They ran experiments on five corpora, 20NG, R8, R52, WebKb, and AG, representing different tasks and sizes. For all experiments, they fine-tuned the classifier for 100 epochs. They repeated the training process for all models using an early stopping where the model would stop training before it overfits the training

data. The confusion matrix calculates the evaluation performances using Table 5 and Eq.s(8)-(12). Table 6 lists the categorical test losses for the authors' deep learning model CNN+BiFaGRU.The word embeddings used to convert the features to vector are One-Hot encoding, GloVe embedding, and Word2Vec embedding. The inputs for these models were the training data. Figure 4 represents a histogram of the performance model using GloVe-300 only. Table 7 shows the performance comparison

Table 6. Test categorical loss for CNN-BiFaGRU with word embedding and without embedding

Pretraining	AG	20NG	R8	R52	WEBKB
One-Hot	5.30	3.10	1.27	0.41	6.03
Glove-100	4.82	2.46	0.85	0.31	7.44
Glove -200	4.97	2.12	0.81	0.23	5.31
Glove-300	4.80	2.05	0.67	0.20	4.41
Word2Vec -50	4.93	2.38	0.86	0.31	4.52
Word2Vec -100	5.00	2.17	0.63	0.28	5.15
Word2Vec -200	4.24	2.09	1.08	0.30	4.58

Figure 4. Evaluation metrics of CNN+BiFaGRU using GloVe-300 embedding on the five datasets



	AG	20NG	R8	R52	WebKb
loss	5.01	2.09	0.64	0.22	3.9
Accuracy	88.05	73.5	96.8	92.64	90.47
Precision	88.41	75.87	97.02	94.28	90.92
Recall	87.79	72.21	96.67	91.82	89.97
F1-Score	89.09	73.95	96.84	93.01	90.44

Table 7. Test error (%) on two text classification datasets used in state-of-the-art

	Model	Test		Model	Test
AG	Char-level CNN	9.51	20 NG	One-Hot LSTM	13.32
	CNN	6.57		SA- LSTM	15.60
	DPCNN	6.87		LSTM	18.0
	CNN-BiFaGRU (w2v-200)	4.24		CNN-BiFaGRU (GloVe-300)	2.05

Table 8. Precision test rates on text classification datasets used in state-of-art

Model	20NG	R8	R52	WEBKB	AG
Text GCN	0.8634	0.9707	0.9356	--	--
LSTM	0.6571	0.9368	0.8554	--	--
Bi-LSTM	0.7318	0.9631	0.9054	--	--
CNN-BiFaGRU (GloVe300)	0.7662	0.9688	0.9510	0.8900	0.899

of the proposed CNN+BiFaGRU method with Glove-300 against existing state-of-the-art methods for text classification.

The authors made a comparison between the CNN-BiFaGRU model with the Word2Vec-100, Glove-300, and other embeddings in terms of performance evaluation (Figure 5 and Table 13) and a percentage error of precision and time consumed (Table 9). For each model in Table9, the precision is in the first row and the time consumed in the second row. Figure 6 illustrates the loss and precision for CNN-BiFaGRU using Word2vec—100 and GloVe-300 with the R8 dataset in (a) and (b) individually based on the results shown in Table 12. Considering the details of each class in R8, Tables10 and 11 summarize the precision, recall, and F1-score results (percentage) with the Word2Vec-100 and Glove-300 embeds. Table 12 lists the evaluation performances of the authors' CNN-BiFaGRU model with and without the word embedding for the five datasets. Table13 indicates the evaluation performances of the other models.

DISCUSSION

BoW vs. Word Embedding

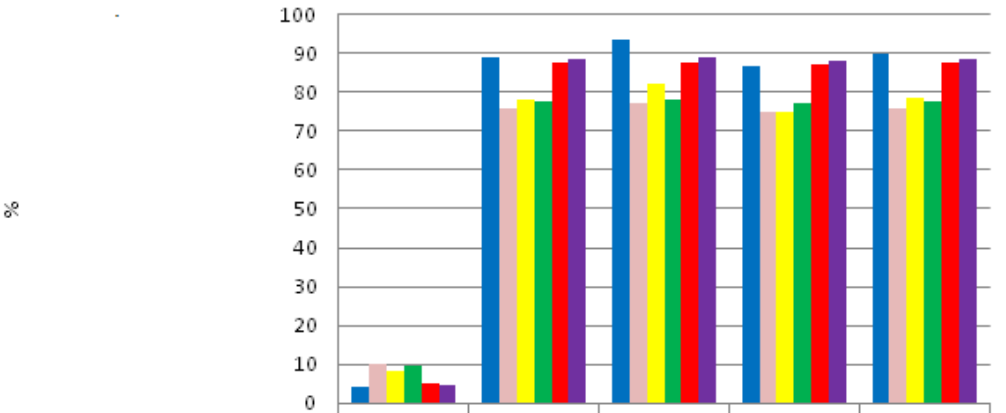
The model uses pretrained word embedding called Word2Vec vectors and GloVe vectors. The vectors are kept static during training. The models with One-Hot in this experiment cannot do much, despite having so many hyper-parameter tuning. The large numbers of text data will make the vocabulary of One-Hot extensive. Hence, the input features will be in sparse form, presenting a bit of information over many zeros. This text representation makes the model harder to train to achieve a better result.

On the other hand, the models perform better when using word embedding. The results inTable6 and Table14 prove the importance of word embedding as a default feature extractor. When comparing the proposed model with and without pretraining embedding in Table 6, the categorical test loss

Table 9. Precision test (%) and time-consuming on text classification datasets (modified state-of-the-art models proposed by the authors)

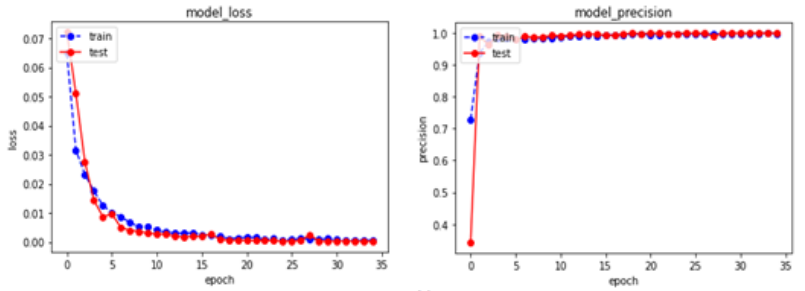
Time Consuming	Models Datasets	1D-CNN*	LSTM*	Bi-LSTM* Precision	GRU*	BiFaGRU W2Vec-100	BiFaGRU Glove-300
	AG-News	88.89	90.11	89.56	90.12	88.86	88.97
		1 h	41 m	2 h	42 m	1 h	1 h
	Webkb	93.57	77.06	82.44	78.18	87.88	89.00
		46 m	37 m	5 m	41 m	4 m	3 m
	R8	94.99	93.49	93.46	96.53	97.11	96.88
		1 h	1 h	1 h	1 h	6 m	6 m
	20NG	75.23	77.53	78.57	67.29	74.95	76.62
		3 h	2 h	8 m	1 h 30 m	15 m	24 m
	R52	93.20	89.18	93.74	88.62	93.58	95.10
		2 h	34 m	23 m	1 h 30 m	10m	16 m

Figure 5. Comparison of performance for different models with the WebKb dataset

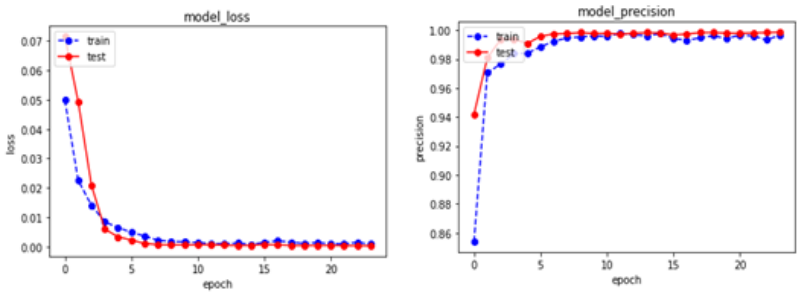


	loss	Accuracy	Precision	Recall	F1-Score
1D-CNN *	3.89	89.26	93.57	86.8	90
LSTM *	10	76	77.06	75.14	76.07
Bi-Lstm *	8.28	78	82.44	75.11	78.53
GRU *	9.42	77.94	78.18	77.49	77.8
CNN+BiFaGRU (Word2Vec-100)	5.15	87.54	87.88	87.27	87.57
CNN+BiFaGRU(Glove-300)	4.41	88.61	89	88.3	88.64

Figure 6. (a) From left to right,loss and precision for CNN+BiFaGRU–using Word2Vec-100 with R8 dataset; (b) from left to right,loss and precision for CNN+BiFaGRU–using GloVe-300 with R8 dataset



(a)



(b)

Table10. Precision, recall, and F1-score results (%) for every class on R8 with Word2Vec-100 embedding

R8 classes	Precision	Recall	F1-Score
0:acq	98	98	98
1:crude	96	95	95
2:earn	99	99	99
3:grain	53	80	64
4:interest	95	85	90
5:money-fx	82	89	85
6:ship	86	69	77
7:trade	90	97	94

Table 11. Precision, recall, and F1-score results (%) for every class on R8 with GloVe-300 embedding

R8 classes	Precision	Recall	F1-Score
0:acq	98	98	98
1:crude	93	96	94
2:earn	98	99	99
3:grain	100	80	89
4:interest	93	85	89
5:money-fx	88	80	84
6:ship	93	69	79
7:trade	87	97	92

using One-Hot is 5.30%. Still, it is just 4.24% using Word2vec-200 in the AG-News dataset and the R52 dataset (One-Hot: 0.41% and Glove-300 it is just 0.20%). As Table 12 shows, models can have a steep increase in precision up to 5%. For example, both the 20NG dataset and R52 dataset jump from 59.98, 84.81 to 74.39 and 93.61%, respectively. The word embedding using pretrained Word2Vec and GloVe always performs better. Thus, in practice, the proposed model CNN-BiFaGRU with Glove-300 dimensional vectors and Word2Vec-100 dimensional vectors should be used.

Word2Vec-100 vs. GloVe-300

Importantly, the results the authors reported in Figure 6 are similar for CNN+BiFaGRU with Word2vec and Glove-300.The bar chart in Figure 4 demonstrates the different evaluation metrics on datasets AG, 20NG, R8, R52, and WebKb. The best accuracy, precision, recall, and F1-score are 96.94%, 97.11%, 96.76%, and 96.93%, respectively, with an R8 dataset with 0.63% loss using Word2vec-100. The authors chose R8 dataset in Figure 6 and Tables 10 and 11 because it is an imbalanced dataset with1083 samples just for the fourth class "earn," and it has a sufficient number of classes to make the results seem clear. It offers the best results, compared to other datasets. According to the precision curves represented in Figure 6, the authors notice that the precision curves with Word2Vec-100 and GloVe-300 are very close, and the same remark is valid for the loss curves.

Tables 10 and 11 indicate the precision, recall, and F1-score of every class on R8 with Word2Vec-100 and Glove-300 embeddings. The aim is to distinguish which is the best between the two word embeddings Word2Vec-100 and GloVe-300.The first-class "acq" results are the same for the two-word embeddings. Recall for the two-word embeddings Word2Vec-100 and GloVe-300 are similar for all classes except class "crude" and class "money-fx," where Word2Vec-100 exceeds GloVe-300.Then, the difference is in the precision and F1-score. Best results for the classes "crude,""earn,""interest," and "trade" are with Word2Vec, and the remaining classes " grain,""money-fx," and "ship" are best

Table 12. Evaluation performance (%) of CNN+BiFaGRU with word-embedding and without on R8, R52, 20NG, AG, and WebKb dataset

Dataset	Model	Loss	Accuracy	Precision	Recall	F1-Score
R8	One-Hot	1.27	94.06	94.33	93.80	94.06
	Glove-100	0.85	95.98	96.14	95.81	95.97
	Glove-200	0.81	96.21	96.30	96.04	96.17
	Glove-300	0.67	96.76	96.88	96.68	96.49
	Word2vec-50	0.86	95.89	95.99	95.81	95.90
	Word2vec-100	0.63	96.94	97.11	96.76	96.93
	Word2vec-200	1.08	94.56	94.95	94.19	94.55
R52	One-Hot	0.41	84.81	92.25	82.91	87.15
	Glove-100	0.31	89.56	92.21	88.62	90.33
	Glove-200	0.23	92.37	94.45	91.20	92.77
	Glove-300	0.20	93.61	95.10	92.86	93.95
	Word2vec-50	0.31	89.37	91.93	88.27	89.99
	Word2vec-100	0.28	90.54	93.58	89.78	91.59
	Word2vec-200	0.30	90.03	93.05	88.73	90.79
20NG	One-Hot	3.10	59.98	63.64	57.56	60.38
	Glove-100	2.46	68.85	71.62	67.52	69.45
	Glove-200	2.12	73.82	75.94	72.59	74.19
	Glove-300	2.05	74.39	76.62	72.80	74.63
	Word2vec-50	2.38	71.27	72.62	70.50	71.52
	Word2vec-100	2.17	73.02	74.95	71.99	73.41
	Word2vec-200	2.09	74.24	75.81	73.27	74.48
AG-News	One-Hot	5.30	88.28	88.75	87.85	88.29
	Glove-100	4.82	88.14	88.80	87.53	88.14
	Glove-200	4.97	88.04	88.77	87.58	88.16
	Glove-300	4.80	88.57	88.97	88.27	88.62
	Word2vec-50	4.93	88.72	88.93	88.67	88.80
	Word2vec-100	5.00	88.68	88.86	88.62	88.74
	Word2vec-200	4.24	89.51	89.98	89.15	89.56
WEBKB	One-Hot	6.03	85.32	86.28	84.42	85.32
	Glove-100	7.44	81.73	82.50	81.41	81.94
	Glove-200	5.31	86.53	87.22	85.98	86.59
	Glove-300	4.41	88.61	89.00	88.30	88.64
	Word2vec-50	4.52	88.97	89.27	88.65	88.95
	Word2vec-100	5.15	87.54	87.88	87.27	87.57
	Word2vec-200	4.58	88.54	89.31	88.05	88.67

with Glove-300.For example, the “crude” class precision in Word2Vec-100 is 96%, but GloVe-300 is 93%.For the “grain” class, precision in Word2vec-100 is 53% only. However, it is 100% in Glove-300. Although most of the classes in the R8 dataset offer the best results with Word2Vec-100 compared to GloVe-300, the authors cannot say that Word2Vec-100 is better than GloVe-300, since the results are too close.

CNN-BiFaGRU vs.RNN Models

The authors made a double comparison between the proposed model and state–of–the–art RNN models and the second between the proposed model and the most famous RNN models with the modifications they made.

CNN-BiFaGRU vs. State-of-the-Art Models

The tests in Table7and Table8 reveal that the proposed model surpasses the expectations of many existing models. For example, for the AG-News dataset, the proposed model’s loss of CNN-BiFaGRU

Table 13. Evaluation performance (%) of modified models * on R8, R52, 20NG, AG, and WebKb datasets

Dataset	Model	Loss	Accuracy	Precision	Recall	F1-Score
R8	1D-CNN*	1.32	93.79	94.99	92.03	93.45
	LSTM*	1.52	92.42	93.49	92.08	92.77
	Bi-LSTM*	1.81	91.05	93.46	89.59	91.42
	GRU*	0.92	95.71	96.53	94.61	95.54
R52	1D-CNN*	0.2	92.7	93.20	92.50	92.80
	LSTM*	0.43	85.28	89.18	84.41	86.65
	Bi-LSTM*	0.51	79.71	93.74	75.58	83.06
	GRU*	0.51	82.20	88.62	79.90	83.86
20NG	1D-CNN*	2.29	73.42	75.23	72.70	73.90
	LSTM*	2.18	72.78	77.53	68.37	72.39
	Bi-LSTM*	2.32	68.20	78.57	58.00	66.26
	GRU*	2.99	62.37	67.29	58.53	62.37
AG-News	1D-CNN*	4.89	88.08	88.89	87.50	88.18
	LSTM*	4.10	89.59	90.11	89.00	89.54
	Bi-LSTM*	4.50	89.00	89.56	88.73	89.14
	GRU*	4.21	89.59	90.12	89.17	89.63
WEBKB	1D-CNN*	3.89	89.26	93.57	86.80	90.00
	LSTM*	10.00	76.00	77.06	75.14	76.07
	Bi-LSTM*	8.28	78.80	82.44	75.11	78.53
	GRU*	9.42	77.94	78.18	77.49	77.80

is much lower than the state-of-the-art model (4.24%) against Char level CNN (9.51%), CNN (6.57%), and DPCNN (6.87%).Furthermore, with the 20NG dataset, the values are just 2.05% loss and 76.62% precision for the proposed model CNN-BiFaGRU, against 18% loss and 65.71% precision for LSTM, 13.32% loss for One-Hot LSTM, and finally 15.60% loss for SA-LSTM. Using word embedding, CNN-BiFaGRU is more effective than RNN-based models such as LSTM or BiLSTM. In three datasets (i.e., 20NG,R8, and R52),the proposed model outclasses all the RNN architectures, with an acceptable precision margin. Compared with TextGCN, the CNN-BiFaGRU precision is still high and close to the highest.

CNN-BiFaGRU vs. Modified Models

The authors chose the WebKb database (Figure 5) to prove that time is an essential factor in the comparison. The best results are marked with 1D-CNN*. However, in terms of time, it can be noted that it takes several minutes, comparing it with other models. For example, CNN-BiFaGRU had a precision of 89% in just 3 m against 93.57% precision for 1D-CNN* in 46 m. The bad results were marked with the LSTM* model with 10% loss and just 77.06%precision.The 1D-CNN* model offers better results, but it takes a long time. However, the proposed model has the best results in a short time. Table 12 summarizes the final comparison for each model performance. Table 13 lists the modified models: 1D-CNN*, LSTM*, BiLSTM*, and GRU*.

Time-Consuming

CNN-BiFaGRU proposed to reduce the cost and time. To improve this, the authors compared it with the four modified models they proposed previously. Table9 shows that the best precision for AG-News is 90.12%, with the GRU* model in 41 m, and the worst results regarding time-consuming are the 1D-Cnn* and BiLSTM *(1h and 2h, respectively). The best precision for the 20NG dataset is 78.57%, with the Bi-LSTM* model in 8 m. The other modified models require more than 1 h (1 h 30mn for GRU*, 2h for LSTM*, and 3h for 1D-CNN*).For the R8 dataset, the best precision is 97.11% with Word2Vec-100 in just 6 m, and the four modified models require 1 h for each to complete the execution. The best precision observed for the WebKb dataset is 93.57%, but it takes 46 m to

complete execution. However, with the GloVe-300, the precision is 89%, but the time required is just 3 m, so the authors consider that the best result for the WebKb dataset is with Glove-300.Finally, the R52dataset with Glove-300 takes just 16m with a precision of 95.10% compared with the modified models. They require from 23m to 2h.The CNN-BiFaGRU model is faster and less time-consuming than the modified models.

CONCLUSION

This paper introduced a novel and efficient classifier using CNN and bidirectional CuDNNGRU named CNN-BiFaGRU. The authors conducted a comprehensive experiment on building deep learning models using two different feature extractions on five text classification datasets. Based on the results, they could note that using a pretrained word embedding such as Word2Vec or GloVe can increase the model accuracy and precision with a high margin. CNN-BiFaGRU is a good recurrent architecture and is effective in classifying text data. It is the best performing algorithm the authors obtained in this series of text classification tasks. The best results were obtained using GloVe-300 and Word2Vec-100 embedding.

The results reveal that the proposed model yields remarkable computing time and precision performance with a low loss against state-of-the-art methods and modified models. For future experiments, the authors suggest extending the kernel sizes between 1 to 10 with more or fewer filters in CNN to see how it affects the model performance. The deeper version of CNN and BI-GRU should be explored to see how it affects the existing performance and also to investigate other options for embedding preentered words such as FastText with static and dynamic modes and compare the result to Word2Vec and GloVe.

ACKNOWLEDGMENT

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.The authors have no conflicts of interest to declare that are relevant to the content of this article.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., . . . Zheng, X. (2016). *TensorFlow: Large-scale machine learning on heterogeneous distributed systems*. http://tensorflow.org

Brownlee, J. (2017). How to one hot encode sequence data in python. *Machine Learning Mastery*, *12*. https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/

Brownlee, J. (2019). A gentle introduction to batch normalization for deep neural networks. *Machine Learning Master.* https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

Cardoso-Cachopo, A. (2007). *Improving methods for single-label text categorization* [Unpublished doctoral dissertation]. https://ana.cachopo.org/datasets-for-single-label-text-categorization

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., & Shelhamer, E. (2014). *CUDNN: Efficient primitives for deep learning*. CoRR,abs/1410.0759.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*(12), 2493–2537.

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. Advances in Neural Information Processing Systems, 28, 3079–3087.

Fawcett, T. (2006). Introduction to receiver operator curves. Pattern Recognition Letters, 27(8), 861–874. doi:10.1016/j.patrec.2005.10.010

Guo, L., Zhang, D., Wang, L., Wang, H., & Cui, B. (2018, October). CRAN: A hybrid CNN-RNN attention-based model for text classification. In *Proceedings of the International Conference on Conceptual Modeling* (vol. 11157, pp. 571-585). Springer. doi:10.1007/978-3-030-00847-5_42

Hochreiter, S.,&Schmidhuber, J. (1997).Long short-term memory. *Neural Computation, 9*(8), 1735-1780. 10.1162/neco.1997.9.8.1735

JohnsonR.ZhangT. (2015a). Effective use of word order for text categorization with convolutional neural networks. 10.3115/v1/N15-1011

Johnson, R., & Zhang, T. (2015b). Semi-supervised convolutional neural networks for text categorization via region embedding. *Advances in Neural Information Processing Systems*, *28*, 919–927. PMID:27087766

Keras. (2021a). *Keras documentation: GRU layer*. https://keras.io/api/layers/recurrent_layers/gru/

Keras. (2021b). *Keras documentation: Batch Normalization layer*. https://keras.io/api/layers/normalization_layers/batch_normalization/

Kim, Y. (2014) Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1746-1751. doi:10.3115/v1/D14-1181

Koidl, K. (2013). *Loss functions in classification tasks.* School of Computer Science and Statistic Trinity College, Dublin. https://www.scss.tcd.ie/~koidlk/cs4062/Loss-Functions.pdf

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P,S., & He, L. (2020). *A survey on text classification: From shallow to deep learning*. arXiv 2020, arXiv:2008.00364.

LiuP.QiuX.HuangX. (2016). Recurrent neural network for text classification with multi-task learning. https://arxiv.org/pdf/1605.05101

Liu, Y., Li, P., & Hu, X. (2022). Combining context-relevant features with multi-stage attention network for short text classification. *Computer Speech & Language*, *71*(C), 101268. doi:10.1016/j.csl.2021.101268

Malekzadeh, M., Hajibabaee, P., Heidari, M., Zad, S., Uzuner, O., & Jones, J. H. (2021). Review of graph neural network in text classification. In *Proceedings of the2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)* (pp. 0084-0091). IEEE. doi:10.1109/UEMCON53757.2021.9666633

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, *26*, 3111–3119.

NVIDIA Developer. (2022). *Recurrent neural network*. https://developer.nvidia.com/discover/recurrent-neural-network

Padawe, G. (2019, October 27). *Word2Vector using Gensim*. https://medium.com/analytics-vidhya/word2vector-using-gensim-e055d35f1cb4

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, *14*, 1532-1543. doi:10.3115/v1/D14-1162

Řehůřek, R. (2022, May 6). *Gensim: Topic modelling for humans*. https://radimrehurek.com/gensim/

Ren, J., Wu, W., Liu, G., Chen, Z., & Wang, R. (2021). Bidirectional gated temporal convolution with attention for text classification. *Neurocomputing, 455*(C), 265-273. 10.1016/j.neucom.2021.05.072

Saxena, S. (2020, October 3). *Understanding embedding layer in Keras*. https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce

Silwimba, F. (2018, October 17). *Bidirectional GRU for Text classification by relevance to SDG#3 indicators.* https://medium.com/@felixs_76053/bidirectional-gru-for-text-classification-by-relevance-to-sdg-3-indicators-2e5fd99cc341

Song, R., Giunchiglia, F., Zhao, K., Tian, M., & Xu, H. (2022). Graph topology enhancement for text classification. *Applied Intelligence*, 1-14.10.1007/s10489-021-03113-8

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016, June). 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). Association for Computational Linguistics.

Yao, L., Mao, C., & Luo, Y. (2019). Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence* (*vol. 33*, pp. 7370-7377). Open Journal Systems. doi:10.1609/aaai.v33i01.33017370

Zhang, B., Wu, J. L., & Chang, P. C. (2018). A multiple time series-based recurrent neural network for short-term load forecasting. *Soft Computing*, *22*(12), 4099–4112. doi:10.1007/s00500-017-2624-5

Zhang, J., Li, Y., Tian, J., & Li, T. (2018). LSTM-CNN hybrid model for text classification. In *Proceedings of the2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (pp. 1675-1680). IEEE. doi:10.1109/IAEAC.2018.8577620

Zhang, X., Zhao, J., & LeCun, Y. (2015). *Character-level convolutional networks for text classification*. 10.48550/arXiv.1509.01626

Zulqarnain, M., Ghazali, R., Ghouse, M. G., & Mushtaq, M. F. (2019). Efficient processing of GRU based on word embedding for text classification. *International Journal on Informatics Visualization*, *3*(4), 377–383. doi:10.30630/joiv.3.4.289