An Improved the Performance of GRU Model based on Batch Normalization for Sentence Classification

Muhammad Zulqarnain[†] Rozaida Ghazali[†], Shihab Hamad Khaleefah[††], Ayesha Rehan[†], Muhammad Shehzad[†††], Yana Mazwin Mohmad Hassim[†]

[†]Faculty of Computer Science and IT, Universiti Tun Hussein Onn Malaysia (UTHM), Johor, Malaysia
[††]Faculty of Computer Science, Al Maarif University College, "31001, Anbar, Iraq
[†††]Department of Computer Science and IT, The Islamia University of Bahawalpur, Punjab, Pakistan

Summary
Sentiment classification is a very popular topic for identifying user opinions and has been extensively applied in Natural Language Processing (NLP) tasks. Gated Recurrent Unit (GRU) has been successfully implemented to NLP mechanism with comparable, outstanding results. GRUs network performs better on sequential learning tasks and overcomes the issues of vanishing and explosion of gradients in standard recurrent neural networks (RNNs). In this paper, we describe to improve the efficiency of the GRU framework based on batch normalization and replace traditional tanh activation function with Leaky ReLU (LReLU). Empirically, we present that our model, with slight hyperparameters, and tuning the statistic vectors, obtains excellent results on benchmark datasets for sentiment classification. The proposed BN-GRU model performs well as compared to various existing approaches in terms of accuracy and loss function. The experimental results has shown that the proposed model achieved better performance over several state-of-the-art approaches on two benchmark datasets, IMDB dataset with 82.4% accuracy, and SSTb dataset with 88.1% binary classification accuracy and 49.9% Fine-grained accuracy respectively. The proposed results are obtained to show the proposed model capable to minimize the loss function, and extract long-term dependencies with a compact architecture that obtained superior performance with significantly fewer parameters.

Key words:
RNN, GRU, Batch Normalization, Long-term dependencies, Sentence Classification.

1. Introduction

Natural Language Processing (NLP) is a massive field of artificial intelligence that is involved with the connection among computers and human language. With the exponential development of collected huge numbers of opinion-rich resources, sentence analysis [1] has been one the superior tasks in (NLP), which purposes to automatically categorize the sentiment separation of a provided texts as negative, positive or more fine-grained" classes. It can support big organization to develop and capture valuable information from large amounts of data which consist of outstanding business significance in brand observation, customer services, and social services. Language modeling is an essential task in machine learning and NLP. Recent, deep neural networks approaches have obtained outstanding performance in text classification [2], and computer vision [3]. Sentence classification performs as the main task in several NLP applications, such as social media analysis [4], documents classification, information retrieval, and medical applications [5], in which require to assigns predefined categories of the sequential texts. Machine learning approaches have been extensively applied in sentence classification, and several of them follows [6] by concentrating on efficient framework handcrafted features for developing an efficient sentiment classifier [7].

However, most of the performance of machine learning is strongly reliant on features representations [8]. Deep learning models, included deep belief network (DBN) [9] convolutional neural networks [10], and recurrent neural networks [11], have successfully achieved remarkable performance in sentence classifications. Between these models, gated recurrent neural networks (GRNNs) with long "short-term memory (LSTM) [12] and gated recurrent units (GRU) [13], is a most popular model because of their capability to manage variable-size of texts and extracts long-term dependency. GRU recently achieved very successful results in many sequences of data such as named entity recognition [14], audio detection and sentence classification [15]. Firstly, their approach performs sentence illustration using a type of recurrent model long-short term memory. Then, these sentences and their dependencies relation are appropriately combined using GRNNs. We investigate GRU in deep recurrent neural networks language model to implement for sentiment classification. Mostly, deep recurrent neural network architectures for natural language processing required several layers to handle variable length and capture long-term semantics [16]. However, the proposed model was inspired by the recently successful standard RNNs approaches in natural language processing applications and its reality that RNNs can extract long-term dependencies.

Manuscript received September 5, 2019
Manuscript revised September 20, 2019

In this research, we presented a GRU model that takes the local features captured by GRU for sentiment classification of sequential text. We proposed a new framework that exploits and apply the batch normalization into GRU architecture of pre-trained real word vector for sentiment classification. We exploit long short-term memory as substitutes in order to minimize the loss function, and extract long-term dependencies through the sequential inputs system. Our research work continues to improve these efforts by further revising standard GRUs. Based on previous work, our main objective is modifying the standard GRU structure to improve the performance of sentence classifications and minimize the information loss. In particularly, our main contributions of this research are summarized in threefold:

- We applied the batch normalization in each layer of the network to reduce the internal covariate shift and cover all informatics information. For this purpose, we used a technique to feed-forward connections only (i.e., W_z, W_r and $W_{\hat{h}}$), to acquire a more efficient architecture that is approximately similarly performance but significantly less computationally expensive.
- We proposed to replace hyperbolic tangent activation function (tanh) with Leaky Rectified Linear Unit (LReLU) activation. LReLU units have been demonstrated to be better performance than sigmoid non-linearities and *tanh* for feed-forward DNNs.
- The proposed BN-GRU architecture performs both tasks well to takes advantages of extracted encoded local features and captured the long-term dependencies between word sequence texts. Experimental results showed that proposed framework attains better results with fewer parameters.

2. Recurrent Neural Network

Recurrent Neural Networks (RNNs) are effective architecture for sequential datasets. The purpose of RNN is to utilize consistent information and output depends on the previous calculation. RNNs are deep learning network that use deep temporal dimension in sequential modeling through time and has presented excellent performance in several NLP tasks [17]. The purpose of RNNs for sentence embeddings are located a dense and illustration low-dimensional semantically by recurrently and consecutively handling each word in a sentence and mapping it through a low-dimensional word vectors. The inclusive contextual features of the entire texts will be in the semantic illustration of the last word in the sentence [18]. If we have accessible to the whole sequential input, we can utilize information not only from the previous time steps, but also from the future ones, enabling for bidirectional RNNs [19] as follows:

$$\overrightarrow{O_t} = \varphi(\overrightarrow{W_x} x_t + \overrightarrow{U_o}\,\overrightarrow{h_{t-1}}) \quad (1)$$

$$\overleftarrow{O_t} = \varphi(\overleftarrow{W_x} x_t + \overleftarrow{U_o}\overleftarrow{h_{t+1}}) \quad (2)$$

$$O_t = [\overrightarrow{O_t} : \overleftarrow{O_t}] \quad (3)$$

The activation function φ is commonly used as a sigmoid function such as the hyperbolic tangent in existing RNNs. However, training of the networks specifically complicated, because of vanishing and exploding gradients [18].

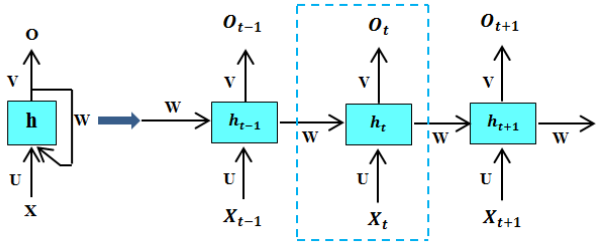

Fig. 1 Traditional structure of RNN

In Fig. 1, Xt is the input unit of timestep t, which shows the word vector of the t th word in the text; ht is the hidden state of timestep t; Ot shows the output of step t, the output is chosen regarding to the require of the network; U, W and V are the weights matrix of the networks that are require to be trained in the model.

Given an sequential input of word vectors (X1,...,XT), they generate a sequential hidden states (h1,...,hT), which are calculated at time step t, thus we can calculate the output as follows in a RNNs:

$$O_t = \varphi(W_x x_t + U_o h_{t-1}) \quad (4)$$

$$H_t^l = \varphi(W_x h_{t-1}^l + U_H h_{t-1}^l) \quad (5)$$

where U_o, U_H is the recurrent weights matrix, W_x is the input to-hidden weights matrix, and φ is an arbitrary activation function. The equations transmit to shows hidden layer activity h_t with its previous hidden layer activity h_{t-1}. This dependence is non-linear due to the usage of logistic activation function $\varphi(\cdot)$.

3. GRU-Based Model

The GRU was recently introduced by [20]. GRU is one of the latest type of the traditional gated RNNs which are applied to addresses the common issues of vanishing and exploding gradients in standard RNNs when capturing long-term dependencies. The architecture of GRU is

adaptively reset or updates its memory contents and is a slightly more simplified variation of the LSTM.
It is combination of the input and forget gates into a single "update gate" and has an additional "reset gate". The GRU model is simpler and fewer parameters as compare traditional LSTM models and are gain progressively popularity. However, distinct the LSTM, the GRU completely exposed it's memory contents each time step and balance among the previous memory contents and the new memory contents particularly use leaky combination, although with its adaptive time continuously control by update gate. The activation h_t^i of the GRU at timestep t is a linear interpolating among the previous activation h_{t-1}^i and the candidate activation $\hat{h}_t^i$:

$$h_t^i = (1 - z_t^i)h_{t-1}^i + z_t^i \hat{h}_t^i \quad (6)$$

The update gate z_t^i help the model to decides how much the unit updates its activation

$$z_t^i = \text{sigm}(W_z x_t + U_z h_{t-1})^i \quad (7)$$

The candidate activation $\hat{h}_t^i$ is calculated same to the update gate:

$$\hat{h}_t^i = \tanh(W x_t + U(r_t * h_{t-1}))^i \quad (8)$$

where r_t^i is referred to as reset gate and $*$ denoted by element-wise multiplication. When the reset gate is off $(r_t^i == 0)$ it permits the unit to forget the past. This is similar to allow the unit learning the first symbol of a sequential input. The reset gate is evaluates the following equation:

$$r_t^i = \text{sigm}(W_r x_t + U_r h_{t-1})^i \quad (9)$$

Update gate z decides how much the previous state should matter now. Units of reset gate r help to capture short-term dependencies with the time-step *t*. Units of update gate z help to capture long-term dependencies with the timestep *t*.

4. Limitations and Motivation

Recurrent Neural Networks (RNNs) is a suitable architecture for modeling units in sequential learning processing [21]. In this way, most of the cases the elements of a sequence are not independent. Means, in common, the transmission of the specific output could depend on the wrapping elements or even on the full-history. But in standard RNN architecture, the vanishing gradient and exploding problem have properly addressed. An ordinary approach dependent on the RNNs, which is a fundamental concept to presents gating techniques for superior handling the flow of the data over the several timesteps. Based on this architecture, vanishing gradient issues are mitigated by generating effective "shortcuts", in which the gradients can bypass many temporal steps. While overcome this issue the most popular gated RNNs are GRU. The main goal of using the GRU network is for which can train to extracts higher-level translation and capture the invariants features from the sequential inputs by composing multiple layers. Regarding this advantage, we found that most of the traditional deep learning architectures need many recurrent layers to extract long-term dependencies but in every recurrence are maximum chances to lose informatics local features. This issue becomes much crucial of RNNs as the range of the sequential inputs increases.
Therefore, we substitute try to use a GRU model is captures long-term dependencies more effectively and also to adjust the number of suitable parameters in the structure. According to the literature reviews and base on existing observations, we concentrated on proposing an efficient model that concentrates on loss reduction from the experimental design, while also extracting long-term depen-dencies more effectively in the term of accuracy and reduce the maximum loss function by using batch normalization (BN) technique. Furthermore, we trained the model by using pre-trained word embedding method and also fed the features maps to the recurrent layer to extract long-term semantic for more powerful classification as presented in the proposed architecture.

5. The Proposed BN-GRU Architecture

In this work, we demonstrate the particular description of the proposed framework architecture that contains gated recurrent unit with batch normalization. The proposed architecture apply word embedding as inputs and learns to extract high-level context words features through time steps, whose outputs are then given by GRU language model, and finally, followed by a softmax classifier.

5.1 The Embedding Layer

In the sentence classification process the initial step is "pre-processing the inputs sentence and sentiment context words. We used the pre-trained GloVe [22] method in the words embedding layer in sequence to transmit each word in the sentences to a real value vector. The embedding layer of the model changes words context into real-valued features vectors that captured semantic and syntactic information. Let L $\in$ RV*d be the embedding query table produced by Glove, where d is the dimension of words and V is the vocabulary size. Assume that the sequential input contains of n words and the sentiment resource contains m words. The input sentence retrieves the word vectors from L and obtains a list of vectors [W1, W2 …, Wn] where

$W_i \in R^d$ is the word vector of the i^{th} word. Consequently, the sentiment resource sequence can recover the word vectors from and form a list of vectors $[W_1^s, W_2^s, \ldots, W_m^s]$. In this way, we can get the matrix $W^c = [W_1, W_2, \ldots, W_n] \in R^{n*d}$ for context words and the matrix $W^s = [W_1^s, W_2^s, \ldots, W_m^s] \in R^{m*d}$ for sentiment resource words. Consequently, we describe to structure the word-level relationship among the sentiment words and the context words to the format of the correlations matrix presented in Fig.2 This process is simply a concatenation of all words embedding in V.

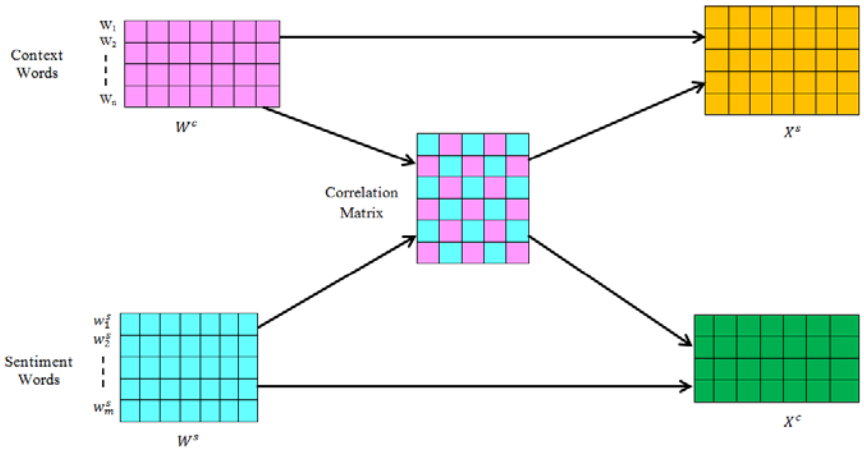

Fig. 2 Sentiment-context word correlation

5.2 The Recurrent Layer

GRU is latest kind of traditional RNNs which are particularly have to use for sequential modeling. At each timestep t, a recurrent layer takes the input vector xt ∈ Rn and hidden state ht by implementing the recurrent procedure:

$$h_t = f(W_{xt} + Uh_{t-1} + b) \quad (10)$$

The major modification of the standard GRU design involves the batch normalization and Leaky ReLU activations as summarized in the next sub-sections. Where $W \in R^{m*n}$, $b \in R^{m*m}$, $b \in R^m$, weights matrix, and f is an element-wise nonlinearity. Training the long-term dependencies with a RNNs is very complicated because the problem of vanishing and exploding gradient [23]. However, gated recurrent unit GRU overcomes such deficiencies of existing RNNs by adding the RNN with a update and reset gates that takes as an input x_t, h_{t-1}, and generates h_t," by the following:

$$z_t = \sigma(W_x^{(z)} x_t + U^{(z)} h_{t-1} + b_z) \quad (11)$$

$$r_t = \sigma(W_x^{(r)} x_t + U^{(r)} h_{t-1} + b_r) \quad (12)$$

$$\hat{h}_t = \tanh(W_x^{(\hat{h})} x_t + r_t * U^{(\hat{h})} h_{t-1} + b_{\hat{h}}) \quad (13)$$

$$h_t = z_t * h_{t-1} + (1 - z_t) * \hat{h}_t \quad (14)$$

where σ is the logistic sigmoid activation function and tanh is the element-wise hyperbolic tangent function and zt, rt, ĥt are referred to as update gate, reset gate, and candidate state. At t=1, h_0 are started to zero vector. * is the element-wise product operator, W, U are weights matrix and vectors. At currently, for RNNs such as GRU, become a common training method include back propagation trough time (BPTT) and real time recurrent learning (RTRL).

5.3 Back Propagation Through Time

Back propagation through time ("BPTT) is the basic procedure which produces the learning algorithm of deep neural networks by computationally adjustable, and in this way of calculating gradients expression over the recursive technique of the chain rule. The essential problem is provided to approximately function $f(x)$ where inputs vectors is x, and we are absorbed in calculating the gradient of f at x(i.e ▽f (x)). BPTT is a simple variation of the backpropagation algorithm for RNNs, with BPTT the error is broadcasted through recurrent connection back in time for particular time-steps. Consequently, the networks absorb and remember information for many time-steps in the hidden layers when it is trained by BPTT." Further detail about the implementation description can be find in [24].

5.4 Leaky Relu Activations (LRelu)

In deep neural network, it's an attempts to solve the prevents dying ReLU issue. The leaky ReLU is a variation of the ReLU has small positive values in the negative area, so it does allow back-propagation, even for negative input values. In this research, we performed second modification to replacing the standard hyperbolic tangent tanh with Leaky ReLU activation. In particularly, we change the candidate state $\hat{h}_t$ (Eq. (13)), by computation as follow:

$$\hat{h}_t = LRelu(W_h x_t + r_t * U_h h_{t-1} + b_h) \quad (15)$$

Standard tanh activations function is less utilized in feed-forward networks because they do not perform better as piecewise-linear activations when training deep network [48]. The selection of ReLU-based neurons, which have demonstrated the efficient to enhancing such constraints, was not so general in the previous for RNNs. However, the activation function connection through batch normalization turns out to be effective for getting benefit of LReLU neurons excluding statistical problems.

5.5 Batch Normalization

Batch normalization is a method to developing with the purpose of directly solves the issue of internal covariate shift. Its known that for deep learning approaches, an internal covariate shift is an ordinary problem where the features are presenting to network changes in distribution during the training process [25]. When using a GRU that resembles very deep feed-forward networks to process sequence data, this internal covariate shift may play an especially important role.

In order to addressed internal covariates shift issue and reduce them by standardizing the intermediate representations of each layer using the statistics for each existing training mini-batch. Batch normalization includes standardizing the activations going into each layer, enforcing their means μ and variances "σ^2 to be invariant to changes in the parameters of the underlying layers, to accelerate the training. Many previously research efforts have already presented that this method is efficient to increase the system performance and to speed-up the learning process [25]. In recurrent neural networks, the batch normalization can be implemented in various ways. In [26], the author proposed to applied normalization step to feed-forward connections only, however in [27] the normalization steps is extended to recurrent connections, utilizing distinct statistics for each timestep.

In this research work, we have to try both techniques, but we did not perceive significant benefit when extension the batch normalization to recurrent settings (i.e., U_z, U_r and U_h). Based on this observation, we proposed this method to feed-forward connections only (i.e., W_z, W_r and W_h), achieving a more efficient architecture that is approximately similarly performance but significantly less computationally expensive. When batch standardization is particularly bounded to feed-forward connection," obviously completely of the relating computations becomes separated by each time-step and they can be executed in equivalent. Batch normalization, adjust the neurons pre-activation, essentially bounding the values of the Leaky (LReLU) neurons. Consequently, the proposed architecture simultaneously takes advantages of both common beneficial impact of LReLU activation and batch normalization. In experimental research, we discovered the recent method which supports against statistical problem also when it is restricted to feed-forward connections only.

Conventionally, in our research, we have applied Leaky ReLU activation with the replacement of standard tanh activation function, and implemented batch normalization. In particular, this change leads to the subsequently modification of Eq. (16, 17, 18), now conducts to the following equations:

The update gate z_t:

$$z_t = \sigma(BN(W_{zx} x_t) + U_{zh} h_{t-1} + b_z) \quad (16)$$

The reset gate r_t:

$$r_t = \sigma(BN(W_{rx} x_t) + U_{rh} h_{t-1} + b_r) \quad (17)$$

Candidate hidden state $\hat{h}_t$:

$$\hat{h}_t = LReLU(BN(W_{\hat{h}} x_t) + r_t * U_{\hat{h}} h_{t-1} + b_{\hat{h}}) \quad (18)$$

Current (output) state h_t:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h}_t) \quad (19)$$

The output state h_t utilizes the update gate z_t to update the past hidden state h_{t-1} and the candidate hidden state $\hat{h}_t$. If the update gate z_t is approximately 1, the previous hidden state will be held and passed to the current moment. When provided a sequential inputs X = [x1, x2 ... xt ... xT] of length T, GRU passes the last hidden state hT through a nonlinear transformation as the output. Furthermore, GRU network constrained with batch normalization concentrate significantly faster and improve generalize. The batch normalizing transform is as follows:

$$BN(c_i, \lambda, \beta) = \lambda * \frac{c_i - \mu_\beta}{\sqrt{\sigma_B^2 + \varepsilon}} + \beta \quad (20)$$

where $c_i \in R^d$ are normalized vectors, $\lambda \in R^d$ and $\beta \in R^d$ are architecture parameters that define the mean and standard deviation of the normalized activation and $\varepsilon \in R^d$ is a regularization hyperparameter. The $*$ denotes the Hademard product (element-wise multiplication).

According to Reference [27] we set β and ε equal 0. At training time, we apply the mini-batch training approach, which splits all the training samples into several mini-batches, and each mini-batch holds out a parameter update. Consequently, the input of batch normalization is the current mini-batch consisting k samples, which can be express as B = $\{c_i \ldots k\}$. $\mu_B \leftarrow \frac{1}{k}\sum_{i=1}^{k} c_i$ is the sample mean and $\sigma_B^2 \leftarrow \frac{1}{k}\sum_{i=1}^{k}(c_i - \mu_B)^2$ is the sample variance. We called this architecture Batch Normalization GRU (BN-GRU), to focus the modification procedure performed on a traditional GRU.

5.6 The Classification Layer

In neural networks, for text sentiment classification, softmax regression is frequently implemented as a final layer for binary and multiclass classification. Its computes fast and provide results with a probabilistic description. Assume that the final sentence representation S∗ of the input text S is fed into a softmax layer to predictive the probabilities distribution of sentences sentiment label over C ("number of sentiment category labels"), and the sentiment label with the maximum possibility is chosen as the final sentiment classification to which the sentence relates. The function is presented as follows:

$$\text{ϒ} = \frac{\exp(W_o^T S^* + b_o)}{\sum_{i=1}^{C} \exp(W_o^T S^* + b_o)} \quad (21)$$

$$y_{pre} = argmaz\,\text{ϒ} \quad (22)$$

Where ϒ the predicted sentiment distribution of the sentence is, y_{pre} is the selected sentiment label, W_o^T and b_o are the parameters of the softmax regression model to be learned

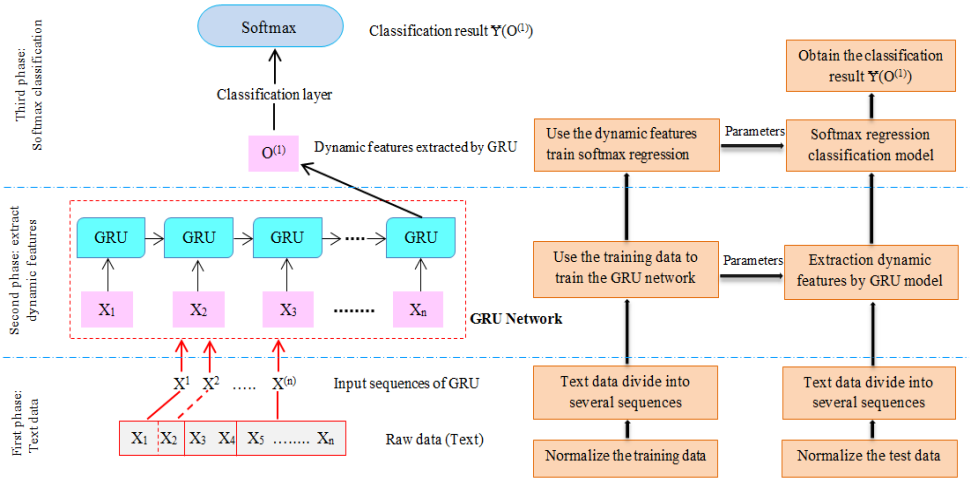

Fig. 3 The proposed flowchart of sentence classification method based on GRU

6. Loss Function and Optimizer

The main concept of the forward pass is performed to calculate the loss functions at the output of GRU (mean squared error for sentiment classification) calculate the values of the weight appropriates gradients, and back propagate them. Therefore, at the level of text sentiment classification, the parameters updates not only rely on the sentence classification cost function but also on the sentiment classification loss. Based on the experimental results, the entire BN-GRU model is trained by minimize the cost function K(Θ) and also computes the classification loss as follows:

$$K(\Theta) := \frac{1}{d}\left[\sum_{i=1}^{d}\sum_{k=1}^{N} 1\{¥^{(i)} = k\} \log(p(¥^{(i)} = k | Ä^{(i)}; \Theta))\right] \quad (23)$$

$$\partial_{SC} \leftarrow \partial_{SC} - lr * (\lambda g_{SC}) \quad (24)$$

where Θ = {Wzx, Wĥx, Uzh, Uĥh, Wo, bz, bh, bo, ¥,θ} is the set of training parameters consisting of all the parameters above. As discussed earlier, this paper uses the mini-batch training scheme, so d here can be understood as a mini-batch. K is the number of classes, $1\{¥^{(i)} = k\}$ is the indicative function indicating that if the class of the ith sample is k, then $1\{¥^{(i)} = k\} = 1$, otherwise $1\{¥^{(i)} = k\} = 0$. In our research, we used Adam algorithm [28] to optimize the loss function. Adam is the first-order optimizer algorithm that can replace the standard stochastic gradient descent procedure. It can iteratively update deep neural networks parameters base on training data. The learning rate maintains by stochastic gradient descent to update all

parameters, in fact, the learning rate does not change during the training process.

Adam optimizer evaluates independently adaptively learning rates for various parameters by computing the first-moment calculation and second-moment calculation of the gradients. The inputs were fixed by through the idea have applied in [29] to suitable an input to a recurrent unit and implemented Adam optimization function [28] with β1 = 0.9, β2 = 0.999 and initial learning rate α = 0.001. The softmax classifier calculates the loss function by using the dense layer. In Eq.24 ∂_{sc} are the parameters of the sentiment classification BN-GRU, g_{sc} is the gradient of such parameters calculated from the sentiment classification cost function ("mean squared error"). Finally, λ is a hyper-parameter for weighting g_{sc} and lr is the learning rate.

The pseudocode of the Adam algorithm for updating Θ is shown in Fig 4. For more details regarding Adam, please refer to [28].

1: Required α = 0.001,
2: Required $\beta_1 = 0.9, \beta_2 = 0.999,$
3: Required $\eta = 10^{-8}$
4: **GRU initialization**
5: **while** *i* in mini-batches **do**
6: $d_0 \leftarrow 0$ (Initialize 1st moment)
7: $z_0 \leftarrow 0$ (Initialize 2st moment)
8: $i_0 \leftarrow 0$ (Initialize timestep)
9: **Forward Pass:**
10: Initiating from the input layer do a forward pass
11: ("with batch normalization") over the networks
12: (Θ_i not converged)
13: **Compute SC Cost Function:**
14: $MSE_i = \frac{1}{d}\sum_{k=1}^{d}(¥_{act.}^{i} - ¥_{pred}^{i})^2$
15: **Backward pass:**
16: Calculate the grad. g_{SC}^{i} of $MSE_i$and back-progate it.
17: **Parameters Updates:**
18: $i \leftarrow i + 1$
19: $\partial_{SC}^{i} \leftarrow \partial_{SC}^{i} - lr * (\lambda \partial_{SC}^{i})$
20: $g_i \leftarrow \partial_{\Theta} f_i(\Theta_{i-1})$ (Get gradient at step *i*)
21 $d_i \leftarrow \beta_1 . d_{i-1} + (1-\beta_1).g_i$ (Update biased 1st moment estimation)
22: $z_i \leftarrow \beta_2 . z_{i-1} + (1-\beta_2).g_i^2$ (Update biased 2nd raw moment estimation)
23: $\hat{d}_i \leftarrow d_i/(1-\beta_1^i)$ (Computer bias-corrected 1st moment-estimation)
24: $\hat{z}_i \leftarrow z_i/(1-\beta_2^i)$ (Computer bias-corrected 2nd raw moment estimation)
25: $\Theta_i \leftarrow \Theta_{i-1} - \alpha . \hat{d}_i/(\sqrt{\hat{z}_t} + \eta)$ (Update parameters)
26: **end while**
27: **return** Θ_i (resulting parameters)

Fig. 4 Pseudocode of the Adam algorithm

7. Experiment Setup

In this section, we explain the experimental settings and empirical results of the proposed model.

7.1 Sentiment Analysis Datasets

In this section, we manage the experimental activity was conducted to achieved results on two publically accessible datasets. To typically evaluated state-of-the-art performance of the proposed BN-GRU model on two experimental sentiment classification datasets: the Stanford Large Movie Review dataset (IMDB) and the Stanford Sentiment Treebank dataset (SSTb).

7.1.1 IMDB Dataset

The IMDB dataset was initially introduced by [23] as a benchmark for sentence classification. It contains the labeled dataset of 50,000 binary IMDB movie reviews; especially chosen for sentiment classification. The IMDB reviews are dividing into 50:50 training and testing sets. The sentiment classification contains determine positive and negative reviews. One basic prospect of this data set is that particular review has many sentences.

7.1.2 SSTb Dataset

The SSTb-1 (Fine-grained) dataset was first introduced by [30] and extended by [31] as a benchmark for sentiment classification. It contains approximately 11,857 reviews adopted from the movie review site Rotten Tomatoes, with one sentence for each review. The SSTb-1 was divided into three sets: 8544 sentences for training, 2210 sentences for testing, and 1101 sentences for validation (or development"") which purposes to categorize a review with fine-grained labels ("very negative, negative, neutral, positive, and very positive"). In Table 1, we provide further detail about the two benchmark datasets. SSTb-2 (Binary): is similar to SSTb-1 (fine-grained) dataset but remove the neutral reviews from it and the binary labels (positive, negative) are implemented.

Table 1: The summary statistics of the two data sets. "#Classes: number of target classes, Ave.length: average sentence length, Max.length: maximum sentence length, train/dev/test: train/development/test set size, V-size: vocabulary size, V-sizepre: number of words show in the set of pre-trained word embeddings, Cross.V: 10-fold" cross validation.

Details	**(SSTb-1) Fine-grained**	**(SSTb-2) Binary**	**IMDB**
Train	8544	6920	10,662
Dev	1101	872	-

Test	2210	1821	Cross.V
Max.length	51	48	59
Ave.length	18	19	21
V-size	17,836	16,185	20,191
V-size$_{pre}$	14,378	13,985	16,746
#Classes	5	2	2

7.2 Implementation Detail

In order to improve the performance of the proposed model, that first step is to improve the quality of the dataset, we enhance the quality of text dataset by preprocessing technique such as eliminating stop words from the input sequence (e.g., “and”, “are” “of”, “the”, “to”) and punctuations. Through the training of sentence embeddings, no stemming is implemented, since, in this way, we will maintain the all information. Then, all words embedding from the text data were initialized by “300-dimensional GloVe word vectors pre-trained by Penington et al. [22]. Some researchers adopted the fine-tuned training strategies for word vectors to enhance the performance for sentence sentiment classification tasks [32]. In a variation, with the aim of well-reflected generalization capability of the model, we prefer to apply the common embeddings for all datasets. For the deep learning networks, the hidden states of the GRU unit in each layer were set to 200. Through the training process, we optimized the proposed model with the AdaDelta algorithm [28] by following the learning rate of 0.001 and the mini-batch size of 32. To alleviate the overfitting problem, we applied the dropout strategy [33], with a dropout rate of 0.5 for the Bi-GRU layer and 10−5 for the coefficient λr of L2 regularization. To evaluate state-of-the-art performance, we have used accuracy and error rate as metrics for the sentiment classification task.

7.3 Hyperparameters and Software

Data preprocessing and manipulate have performed in Python 3.6 and anaconda, required libraries that are used in this research work such as TensorFlow, Sklearn, Numpy, Scipy, Pandas and Keras packages. Deep learning GRU networks and traditional DNNs are executed with TensorFlow, an open-source software library for numerically computations using data flows graph. Performance of all methods was based on pre-defined assessment measures. Completely simulations were implemented on Intel Core i7-3770CPU @3.40 GHz,” and 4GB of RAM machine.

Table 2: Hyperparameters used in proposed BN-GRU model

Hyperparameters	BN-GRU
Mini-batch Size	32
Cell Size	128
Dropout Rate	0.5
Epochs	20-30
Learning Rate	0.001
No.of classes	2,5

8. Empirical Results and Analysis

8.1 Effects of Batch Normalization

In deep neural networks, such as gated recurrent unit, as the deep network, there will be issues with the covariate shift, which will minimize the learning efficiency of the GRU network. The recently proposed batch normalization algorithm that helps to addresses these issues and solves it efficiently. We can see the impact of batch normalization from the convergence speed and extent of the loss function during the training process in Figure 5.

In addition, Table 3 compares the proposed BN-based GRU with standard GRU in various details and presents that the BN-based GRU is better, both on the basis of speed and accuracy. The decision of architecture hyperparameters and the utilization of BN techniques are conceptually based.

The deep neural networks models have too many parameters (weights and biases), thus it has poor generalization capability and is easily over-fitted when handling with high-dimensional data, that is, the “curse of dimensionality”. Therefore, the GRU model is adopted in this research paper. The GRU model is comparatively sparse, so it has benefits in processing texts data. The research experimental results also present to reduce the sentence classification loss function, when applied the batch normalization with GRU architecture. Furthermore, in order to avoid over-fitting, the BN algorithm was cited here to increase the GRU performance, and shows that the introduction of BN was efficient.

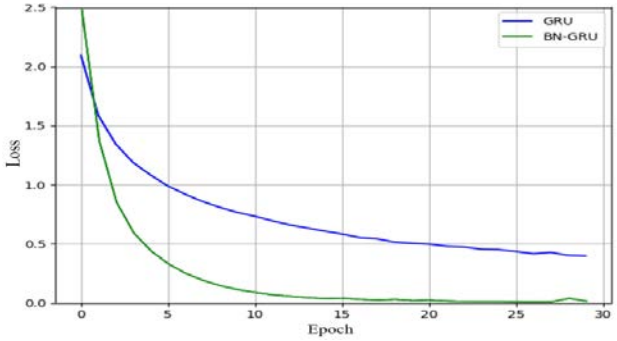

Fig. 5 Loss function of GRU and BN-GRU during the training process.

Table 3: The comparison between GRU and BN-GRU

Details	GRU	BN-GRU
Training accuracy	0.8835	0.9164
Testing accuracy	0.7940	0.8231
Training loss	0.3865	0.0016
Testing loss	0.7216	0.4029
Epochs at convergence	30	25

8.2 Optimization

We executed the implementation of our proposed BN-GRU model on two "SSTb, IMDB benchmark datasets with particular parameters. Model training was completed over the stochastic gradient descent through shuffled mini-batches. For training and validation, we randomly divided the complete training examples. The limit of the validation set is similar to the respective test limit and is balanced in each class. We trained the proposed model by reducing the negative log-likelihood or cross-entropy loss. In addition, a common direction was identified the number of dimensions in words embedding and result in the performance of gated units is superior. But the improvement significantly reduces as they are increases beyond a particular limit [34]. We also observed that maintaining the embedding dimensions equivalent to the number of gated units performed well than networks containing units much more than words embedding dimensions.

In our work, we employed the pre-trained GloVe [22] method in the word-level embedding layer in sequence to transfer each word in the sentences for computing a real value vector representation of a word. We conducted the experiment on SSTb and IMDB datasets, SSTb consists of two sentiment categories, (Fine-grained and binary) consider variance in the number of parameters. We evaluated the classification performance of the proposed BN-GRU model with traditional RNNs models, CNN and SVM. Our model BN-GRU achieved much better performance in both datasets in the term of accuracy. Table 4 is shown evaluation results (accuracy %) on IMDB and SSTb datasets. The best result of each dataset shows in bold. Results marked with * are obtained either by our own implementation or with the same codes shared by the original authors. While those without * were re-printed from the references (i.e.," [[35], [36], [31], [37], [38], [39]]).

Table 4: Comparison accuracy results of proposed model with existing models

Models	(SSTb-1) Fine-grained	(SSTb-2) Binary	IMDB
CNN-non-static [35]	48.0	87.2	81.5
DRNN [36]	49.8	86.6	78.8*
RNTN [31]	45.7	85.4	76.1*
MV-RNN [37],[31]	44.4	82.9	79.0
LSTM [38]	46.4	85.8*	77.4
Bi-LSTM [39]	49.1	87.5	80.2*
GRU	49.5*	87.4*	80.0*
SVM	40.8*	79.4*	76.6*
BN-GRU (Proposed)	**49.9**	**88.1**	**82.4**

8.3 Comparison an Error rate with traditional RNNs

In this section, we compare and analysis an error rate of our proposed BN-GRU model with two traditional deep learning RNNs models GRU and LSTM [20]. We fixed both the word embedding dimensions and the number of units in a hidden layer to 128 and execute the model for 26 epochs. We found that BN-GRU model converged faster than GRU and LSTM to achieved lower error rate even after many epochs. To make these models comparable, we implement these models with the identical structural design shown in Fig 6. The traditional RNNs models were run for 30 to 35 epochs to achsieve the shown accuracy while BN-GRU was trained in just 24 epochs. Furthermore, we identified that the proposed BN-GRU model outperformed all of the other RNNs models by an important margin. We evaluate our model BN-GRU with traditional RNNs models on two SSTB, and IMDB datasets. Fig 6 shows the results that proposed BN-GRU performs better than standard GRU and LSTM. Our model BN-GRU achieves much better performance in the term of the error rate than GRU and LSTM.

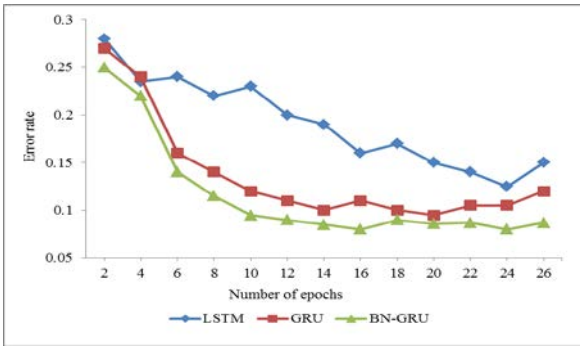

Fig. 6 Comparison an error rate (%) with traditional RNNs models

9. Conclusion

Sentiment classification remains common and significant area of natural language processing. In this paper, we proposed and improved gated recurrent unit for sentiment classification. The main idea of our proposed model is utilize to replace the standard hyperbolic tangent activation function (tanh) with Leaky Rectified Linear Unit (LReLU)

activation with final softmax output layer to the sentence classification. Furthermore, we add the batch normalization technique only feed forward connections (i.e., W_r, W_z, and $W_{\hat{h}}$) to cover the all informatics information to minimize the loss function and attaining a more compact model that is approximately similar performance but significantly less computationally expensive. In this way, data can be well trained using the BN-GRU neural network. Moreover, we trained the GRU model and optimized it with the batch normalization method to minimize the influence of the covariate displacement that exists in the deep neural networks. The proposed approach of using the GRU-based model was relatively simple and efficient, and being a recurrent network its guarantee to both efficiency and significant accuracy when capturing the useful information from a massive array of sequential text data. The proposed model performed better on two experimental datasets included (SSTb, IMDB) and obtained competitive classification accuracy while outperforming some other traditional RNNs models. Furthermore, it will be remarkable to observe future work on implementing proposed model for further applications such as information retrieval or machine translation.

Acknowledgments

The authors wish to thanks the Ministry of Education Malaysia and University Tun Hussein Onn Malaysia for conducting these research activities under vote no.1641.

References

[1] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou, “Sentiment Embeddings with Applications to Sentiment Analysis,” IEEE Trans. Knowl. Data Eng., vol. 28, no. October, pp. 496–509, 2016.

[2] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, “HDLTex: Hierarchical Deep Learning for Text Classification,” 2017 16th IEEE Int. Conf. Mach. Learn. Appl., no. Octuber, pp. 364–371, 2017.

[3] Q. Abbas, “MADeep-Automatic Microaneurysms Detection on Retinal Fundus by using region growing and deep neural networks,” IJCSNS Int. J. Comput. Sci. Netw. Secur., vol. 19, no. 1, pp. 161–166, 2019.

[4] V. Sornlertlamvanich and P. Jotikabukkana, Effectiveness of Social Media Text Classification by Utilizing the Online News Category, no. August. 2015.

[5] M. Hughes, I. Li, S. Kotoulas, and T. Suzumura, “Medical Text Classification using Convolutional Neural Networks,” Stud Heal. Technol Inf., vol. 235, no. May, pp. 246–250, 2017.

[6] B. Pang, L. Lee, H. Rd, and S. Jose, “Thumbs up ? Sentiment Classification using Machine Learning Techniques,” proceeding to EMNLP, no. July, pp. 79–86, 2002.

[7] Al-harbi, “A Comparative Study of Feature Selection Methods for Dialectal Arabic Sentiment Classification Using Support Vector Machine,” IJCSNS Int. J. Comput. Sci. Netw. Secur., vol. 19, no. 1, pp. 167–176, 2019.

[8] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” no. 1993, pp. 1–30, 2012.

[9] J. Song, S. Qin, and P. Zhang, “Chinese Text Categorization Based on Deep Belief Networks,” IEEE ICIS 2016, no. June, pp. 1–5, 2016.

[10] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A Convolutional Neural Network for Modelling Sentences,” Proc. 52nd Annu. Meet. Assoc. Comput. Linguist. (Volume 1 Long Pap., pp. 655–665, 2014.

[11] D. Tang, B. Qin, and T. Liu, “Document Modeling with Gated Recurrent Neural Network for Sentiment Classification,” Proc. 2015 Conf. Empir. Methods Nat. Lang. Process., no. September, pp. 1422–1432, 2015.

[12] S. Hochreiter, “Long Short Term Memory,” Neural Comput., vol. 9, no. 8, pp. 1–32, 1997.

[13] K. Cho et al., “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” arXiv, no. September, pp. 1–15, 2014.

[14] W. Khan, “Deep recurrent neural networks with word embeddings for Urdu named entity recognition,” ETRI J., vol. 0, no. February, pp. 1–11, 2019.

[15] Kumar, “Self-Attention Enhanced Recurrent Neural Networks for Sentence Classification,” 2018 IEEE Symp. Ser. Comput. Intell., pp. 905–911, 2018.

[16] T. Yang, T. Tseng, and C. Chen, “Recurrent Neural Network-based Language Models with Variation in Net Topology , Language , and Granularity,” 2016 Int. Conf. Asian Lang. Process., no. 3, pp. 71–74, 2016.

[17] C. Hansen, C. Hansen, S. Alstrup, and J. G. Simonsen, “Modelling Sequential Music Track Skips using a Multi-RNN Approach,” arXiv Prepr. arXiv1903.08408, no. April, pp. 1–4, 2019.

[18] R. Pascanu, D. Tour, T. Mikolov, and D. Tour, “On the difficulty of training recurrent neural networks,” Conf. ICLR, no. 2, pp. 1310–1318, 2013.

[19] K. K. P. Schuster, Mike, “Bidirectional recurrent neural networks,” IEEE Trans. SIGNAL Process., no. December 1997, pp. 2673–2681, 2016.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” pp. 1–9, 2014.

[21] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A Critical Review of Recurrent Neural Networks for Sequence Learning arXiv : 1506 . 00019v4 [cs . LG] 17 Oct 2015,” pp. 1–38, 2015.

[22] Pennington, R. Socher, and C. D. Manning, “GloVe : Global Vectors for Word Representation,” Proc. Conf. Empir. Methods Nat. Lang. Process., no. October, pp. 1532–1543, 2014.

[23] L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning Word Vectors for Sentiment Analysis,” Proc. 49th Annu. Meet. Assoc. Comput. Linguist. Hum. Lang. Technol., vol. 1, pp. 142–150, 2011.

[24] M. Bod, “A Guide to Recurrent Neural Networks and Backpropagation A guide to recurrent neural networks and

backpropagation," Dallas Proj. Dept. Comput. Sci., Univ, no. August, pp. 1–10, 2014.
[25] S. Ioffe and C. Szegedy, "Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift," Proc. Int. Conf. Mach. Learn. Lille, Fr., no. July, pp. 6–11, 2015.
[26] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Batch Normalized Joint Training for DNN-Based Distant Speech Recognition," 2016 IEEE Spok. Lang. Technol. Work., pp. 28–34, 2016.
[27] T. Cooijmans, N. Ballas, C. Laurent, and A. Courville, "Recurrent Batch Normalization," Proc. Int. Conf. Learn. Represent. Puerto Rico, San Juan, Puerto Rico, USA, no. Section 3, pp. 1–13, 2017.
[28] D. P. Kingma and J. L. Ba, "A method for stochastic optimization," arXiv, no. March, pp. 1–15, 2015.
[29] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM Fully Convolutional Networks for Time Series Classification," IEEE Access, vol. 6, pp. 1662–1669, 2018.
[30] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," inProc.43rdAnnu. Meet. Assoc. Comput. Linguist., vol. 3, no. 1, pp. 115–124, 2005.
[31] R. Socher, A. Perelygin, and J. Wu, "Recursive deep models for semantic compositionality over a sentiment treebank," Proc. …, no. October, pp. 1631–1642, 2013.
[32] H. Lee, "for Modeling Sentences and Documents," Proc. 15th Annu. Conf. North Am. Chapter Assoc. Comput., no. June, pp. 1512–1521, 2015.
[33] G. Hinton, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," J. Mach. Learn. Res. 2014, 15, vol. 15, pp. 1929–1958, 2014.
[34] S. Biswas, E. Chadda, and F. Ahmad, "Sentiment Analysis with Gated Recurrent Units," Adv. Comput. Sci. Inf. Technol., vol. 2, no. 11, pp. 59–63, 2015.
[35] Y. Kim, "Convolutional Neural Networks for Sentence Classification," Artif. Intell. Rev., no. 4, pp. 655–665, 2014.
[36] and C. Cardie, "Deep Recursive Neural Networks for Compositionality in Language," Adv. neural Inf. Process. Syst., pp. 2096–2104, 2014.
[37] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic Compositionality through Recursive Matrix-Vector Spaces," Proc. 2012 Jt. Conf. Empir. methods Nat. Lang. Process. Comput. Nat. Lang. Learn., no. July, pp. 1201–1211, 2012.
[38] Q. Qian and M. Huang, "Linguistically Regularized LSTM for Sentiment Classification," arXiv Prepr. arXiv, no. November, pp. 1–11, 2016.
[39] K. S. Tai, R. Socher, and C. D. Manning, "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks," arXiv Prepr. arXiv, no. May, pp. 1–11, 2015.