



Muhammad Ahsan Tariq¹

¹MS Project Management, Bahria University Lahore Campus.

KEYWORDS	ABSTRACT
Text classification; graph convolutional network; Text GCN; Leaky ReLU; activation function; training efficiency.	Text GCNs create a graph atop the entire corpus and assign classification to each document as a collaborative node classification process. Text GCNs work well on long document benchmarks, but the corpus graph is slow to train for computers without a graphics processing unit (GPU). Training time for a process is decreased if the rectified linear unit (ReLU) in the hidden layer of a Text GCN is replaced with a Leaky ReLU, but it is not known if the classification performance is also reduced. The Graph constructions, Architectures and Training schedules of Text GCNs were kept the same, and hidden layers were changed to Leaky ReLUs, while everything else was kept constant. The experiments were run on an 8th generation Intel Core i5 CPU with 16 GB of RAM, and the hidden layer activation was Leaky ReLU for all experiments. In general, Test accuracy for all corpora shows the Leaky ReLU was within a percentage point of the ReLU baseline, and macro-averaged F1 changed by no more than 1.8 points. Total training time for the 8 corpora decreased by about 20% overall, with 20 Newsgroups showing a 20% decrease, R52 a 26% decrease, and R8 a 32% decrease. Since the two activations cost almost the same per epoch, the most likely reason for the decrease is validation-based early stopping. All results come from single training runs, therefore the time saving should be noted as a practical observation that still needs repeated runs in order to confirm early stopping.
ARTICLE HISTORY Date of Submission: 05-06-2026 Date of Acceptance: 02-07-2026 Date of Publication: 07-07-2026	 2026 Journal of Social Research Development
Correspondence	Muhammad Ahsan Tariq
Email:	mohammadahsantariq@gmail.com
DOI	https://doi.org/10.59075/jsrd.v7i5.526
Volume/Issue	7(5), 2026, 46-62

Introduction

It is used for news filtering, spam detection, document sorting, sentiment analysis, and abusive language detection (Kowsari et al., 2019; Minaee et al., 2021). Typically, classification does not work on unstructured data, so the first step in the design is to think of a way to represent text in a numerical way. Some of the first systems relied on sparse features that were designed by hand and included the number of words in a bag, n -grams, or TF-IDF weights (Salton & Buckley, 1988; Wang & Manning, 2012). Later, much of this work was done by dense word embeddings that were learned

from large corpora, such as word2vec, GloVe, and fastText (Mikolov et al., 2013; Pennington et al., 2014; Joulin et al., 2017), and pre-trained transformers such as BERT have since dominated many benchmarks (Devlin et al., 2019).

Convolutional and recurrent networks extract features from sequence of words (Kim, 2014; Hochreiter & Schmidhuber, 1997). The design is focused on locality and word order (Battaglia et al., 2018). As a result, they may overlook long-distance relationships that span the whole corpus. Such relations are modeled in graph neural networks. Yao et al. (2019) created a different heterogeneous graph for each corpus and placed every document and every vocabulary word as a node. They trained a two-layer graph convolutional network (GCN; Kipf & Welling, 2017) on this graph. Their model, Text GCN, performs document classification in a semi-supervised node classification framework and requires no pre-trained embeddings.

This design is sensitive to price due to the size of the graphs. In one of their examples, Yao et al. (2019) use the 20 Newsgroups' graph containing 61,603 nodes. Training involves multiplying a large, sparse matrix by a dense vector for each hidden unit. For each iteration of this process, one training pass takes many minutes for a machine without a GPU. For researchers who work on a laptop or node on a shared CPU server, these costs constrain how many experiments can be performed.

The design of the activation function is an inexpensive, simple, area to intervene. Commonly used activation functions such as ReLU (Nair & Hinton, 2010; Glorot et al., 2011) do not back propagate negative values. Therefore, a unit whose pre-activation remains negative ceases learning and forever stays at that negative value (Lu et al., 2020). Leaky ReLU does not back propagate negative values, but instead a negative input will have a small positive slope (Xu et al., 2015; Dubey et al., 2022). Leaky ReLU has matched or outperformed ReLU activation in many of the empirical studies. To the best of our knowledge, this has not been studied in the context of Text GCN.

This paper evaluates a controlled change to Text GCN. All that changes is the activation function at the hidden layer. We assess accuracy, precision, recall, and F1, as well as wall-time for model training on eight publicly available datasets from different domains on a "CPU only" machine. Three novel contributions are made by this paper. First, the authors evaluate ReLU and Leaky ReLU in the context of Text GCN on eight datasets that include TREC, WebKB and a multilingual web-video dataset that were not considered in the original analysis of Text GCN. Second, the authors find a trade-off resulting in a drop of training time by around one fifth with Leaky ReLU at the cost of maintaining baseline classification performance. Third, the authors describe the potential time-saving mechanism and explain what else would be needed to consider this a robust finding.

Related Work

Text representation and deep text classifiers

The focus of traditional text classification models has been on feature selection, engineering and classification. Along with the classical bag-of-words model, researchers have proposed n-gram based models (Wang & Manning, 2012), entity based models that are derived from ontologies (Chenthamarakshan et al., 2011) and models that are based on text graphs (Rousseau et al., 2015; Luo et al., 2017). Methods that are based on embeddings either combine unsupervised pre-trained word or paragraph vectors to form document vectors (Le & Mikolov, 2014; Joulin et al., 2017) or jointly train word, document and label representations (Tang et al., 2015; Wang et al., 2018). Shen et al. (2018) provided strong baselines for text classification by simply averaging word embeddings. Deep learning models that operate on text sequences include word-level CNNs (Kim, 2014; Johnson & Zhang, 2015), character-level CNNs (Zhang et al., 2015; Conneau et al., 2017) and LSTM-based modelling (Tai et al., 2015; Liu et al., 2016).

Graph neural networks for text

Defferrard et al. (2016) made spectral graph convolution efficient with Chebyshev polynomial filters. Kipf and Welling (2017) simplified it to the first-order layer used in this study. Wu et al. (2021) examined later developments. Peng et al. (2018) applied graph CNNs to word graphs for large-scale hierarchical classification. Text GCN (Yao et al., 2019) was the first to jointly learn word and document embeddings by representing an entire corpus as a word-document graph. Later work combined a graph for each document so that unclassified documents could be classified (Huang et al., 2019), combined several graphs of the same corpus (Liu et al., 2020) and coupled Text GCN with BERT (Lin et al., 2021).

Two results from the GCN literature have direct implications for this study. Li et al. (2018) illustrated how, in most cases, the convergence behavior of a simple two-layer GCN is superior to that of a deep stack. Wu et al. (2019) eliminated the nonlinear activation between GCN layers and achieved similar accuracies to the state of the art on several benchmarking tasks. From the above results, we can draw two important observations: in a shallow GCN, selection of a nonlinear activation function may not improve accuracy as much as optimization.

Rectifier activations

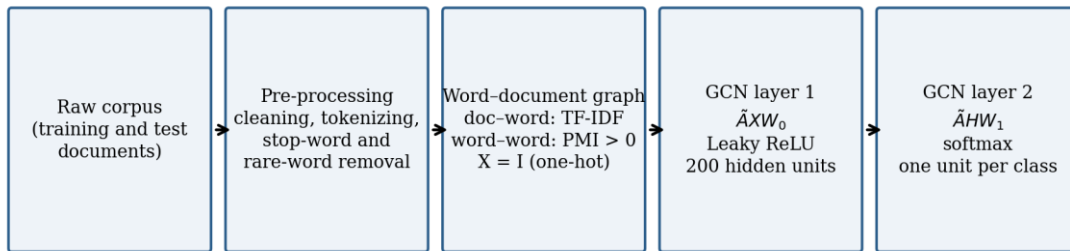
ReLU makes deep networks easier to train than sigmoid or tanh units because its gradient does not saturate for positive inputs. It is also easier to train (Nair & Hinton, 2010; Glorot et al., 2011). For negative inputs, the gradient is zero, and this leaves units permanently inactive. Lu et al. (2020) proved that a deep ReLU network dies with probability approaching one as the number of layers increases. The zero gradient of ReLU inspired several unit designs. Leaky ReLU (Maas et al., 2013) replaces the zero gradient with a small positive slope. Parametric ReLU learns the slope from the data (He et al., 2015), and the exponential linear unit uses a negative branch (Clevert et al., 2016).

Leaky variants of ReLU outperform ReLU in numerous classification tasks on images (Xu et al., 2015). However, a recent benchmark studied the best activation for various networks and several types of data (Dubey et al., 2022) and found that the performance of ReLU, Leaky ReLU, and other activations depended on the network and the data. Recently, in the Leaky ReLU absolute unit, Mastromichalakis (2020) reported positive results on image, text, and tabular benchmarks.

The study of Leaky ReLU networks is also expanding. First, Frei et al. (2023) studied the implicit bias of gradient descent on two-layer Leaky ReLU networks on almost orthogonal vectors in high-dimensional space, and later Liu et al. (2023) presented the inexact augmented Lagrangian method for training Leaky ReLU networks with group sparsity. Text GCN uses one-hot node features that are exactly orthogonal before propagation. This gives some relevance to the first of these studies. However, the features that propagate to hidden units to be activated are not orthogonal.

Methodology

The model is designed in accordance with Text GCN (Yao et al., 2019) except in the case of the hidden layer's activation function. Figure 1 illustrates the steps of the model.



loss: cross-entropy on labelled document nodes

Figure 1 Text GCN pipeline used in this study. The corpus graph contains all training and test documents; only the labels of training documents enter the loss. The single modification examined here is the activation after the first graph convolution (ReLU in the baseline, Leaky ReLU in the modified model).

Text graph construction

Let the corpus have N documents and a vocabulary of M words after the application of the pre-processing steps. The graph contains $n = N + M$ nodes. The edges link documents with the words contained in them. There are also edges linking pairs of words that co-occur. The weighted adjacency matrix A is defined (Yao et al. 2019), as follows:

$$\begin{aligned}
 A_{ij} &= PMI(i, j) \text{ if } i \text{ and } j \text{ are words and } PMI(i, j) > 0 \\
 A_{ij} &= TF - IDF_{ij} \text{ if } i \text{ is a document and } j \text{ is a word} \\
 A_{ij} &= 1 \text{ if } i = j; \quad A_{ij} = 0 \text{ otherwise}
 \end{aligned} \tag{1}$$

The document-word pairs are linked by weights which are computed as TF-IDF. In this case, the frequency of the word is counted as the total number of times the word appears in the document and the document frequency is counted as the logarithm of the total number of documents which did not contain the word (Salton & Buckley, 1988). The weights linking the word pairs are computed as point-wise mutual information (PMI; Church & Hanks, 1990), with a fixed-size sliding window that moves over the documents:

$$PMI(i, j) = \log \frac{p(i, j)}{p(i) p(j)}, \quad p(i, j) = \frac{\#W(i, j)}{\#W}, \quad p(i) = \frac{\#W(i)}{\#W} \tag{2}$$

Let $\#W$ be the total number of sliding windows, $\#W(i)$ be the number of windows that contain the word i , and $\#W(i, j)$ be the number of windows that contain the co-occurring word pair (i, j) . A positive PMI indicates that the word pair co-occurs more than expected. Hence, edges are assigned only to word pairs with POS PMI. Node features are the identity matrix, $X = I$, which means every word and document is a one-hot vector.

Two-layer graph convolutional network

The adjacency matrix is symmetrically normalized with its degree matrix D (Kipf & Welling, 2017):

$$\tilde{A} = D^{-1/2} A D^{-1/2}, \quad D_{ii} = \sum_j A_{ij} \tag{3}$$

Two graph convolution layers produce class probabilities for every node:

$$Z = \text{softmax}(\tilde{A} \varphi(\tilde{A} X W_0) W_1) \tag{4}$$

where W_0 associates the one-hot encoded inputs with a 200-dimensional hidden layer, W_1 associates the hidden layer with one output per class, and φ denotes the hidden activation. The first layer enables nodes to exchange information with direct neighbors. The second layer extends this to nodes that are two hops away. Hence, even in the absence of direct edges between document and document nodes, information can be transmitted between two documents through the words they have in common (Yao et al., 2019).

Hidden-layer activation

The baseline uses ReLU and the modified model uses Leaky ReLU with negative slope α :

$$\text{ReLU}(x) = \max(0, x), \quad \text{LeakyReLU}_\alpha(x) = \max(\alpha x, x), \quad 0 < \alpha < 1 \quad (5)$$

For positive inputs, both functions yield x with a derivative of 1. For negative inputs, since ReLU returns 0 with a derivative of 0, Leaky ReLU returns αx with a derivative of α (Figure 2). When using ReLU, the hidden unit, whose pre-activation is negative, does not contribute to the gradient for that row of W_0 , while it does contribute a scaled gradient with Leaky ReLU. We set $\alpha = 0.01$, which is a standard choice for Leaky ReLU (Maas et al., 2013).

This modification will not have a significant effect on the cost of an epoch. Both activations perform the same operation on an $n \times 200$ matrix which is comparatively smaller than the two sparse-dense matrix multiplications in Equation (4) which will have a cost that is related to the number of edges in the graph. As such, any difference in total time spent on training must be driven by the number of epochs that were conducted before training was halted.

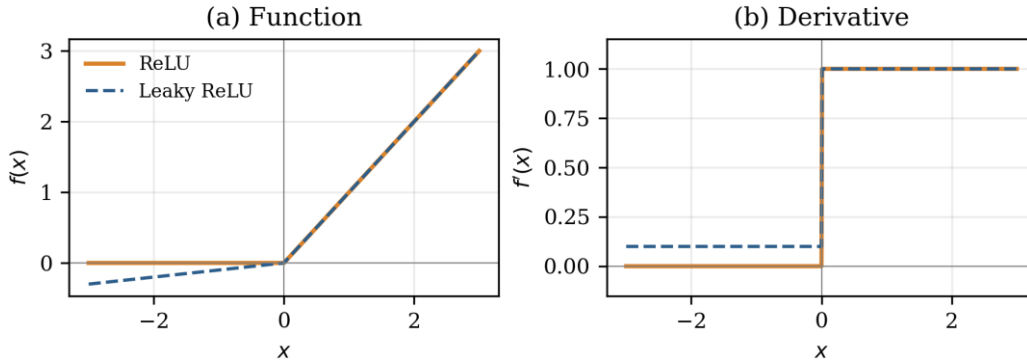


Figure 2. ReLU and Leaky ReLU (a) and their derivatives (b). The negative slope is drawn as $\alpha = 0.1$ so that it is visible; the experiments used a smaller value.

Training objective

The loss is the cross-entropy over labelled document nodes only:

$$\mathcal{L} = - \sum_{d \in \mathcal{Y}_D} \sum_{f=1}^F Y_{df} \ln Z_{df} \quad (6)$$

where $\mathcal{Y}_D$ is the set of labeled training documents, F is the number of classes, and Y is the one-hot label matrix. Test documents are included in the graph, but their labels are not used. Therefore, the model is transductive.

Experimental Setup

Datasets

Descriptions of the eight corpora used are in Table 1. 20 Newsgroups (20NG) is the "bydate" version of Lang's newsgroup collection (1995). R8 and R52 are the single-label R8 and R52 subsets of Reuters-21578. Ohsumed consists of single-label MEDLINE cardiovascular-disease abstracts (Hersh et al., 1994). Movie Review (MR) is a binary, sentence-level, sentiment corpus (Pang & Lee, 2005). The statistics in Table 1 are the same as those provided by Yao et al. (2019), and for the first five corpora, we used the train and test splits, and the pre-processed files, provided by the authors of Text GCN. TREC is a six-coarse-class question classification corpus (Li & Roth, 2002), and in WebKB, university web pages are categorized by page type (Craven et al., 1998). MIX consists of eight categories and has a mix of English, French, and Spanish. It is the web video corpus of Afzal et al. (2016), and consists of text that is attached to web videos, including titles and descriptions.

Table 1. Summary statistics of the corpora after pre-processing. Nodes = documents + words.

Dataset	Documents	Training	Test	Words	Nodes	Classes	Avg. length
20NG	18,846	11,314	7,532	42,757	61,603	20	221.26
R8	7,674	5,485	2,189	7,688	15,362	8	65.72
R52	9,100	6,532	2,568	8,892	17,992	52	69.82
Ohsumed	7,400	3,357	4,043	14,157	21,557	23	135.82
MR	10,662	7,108	3,554	18,764	29,426	2	20.39
TREC	5,952	5,452	500	–	–	6	–
WebKB	–	–	–	–	–	–	–
MIX	4,482	3,314	1,168	6,602	11,084	8	36.70

TREC counts follow the standard split of Li and Roth (2002). –: not reported.

Pre-processing

Following Kim (2014), text was cleaned, tokenized, and set to lowercase. R8, 20NG, R52, and Ohsumed lost all words with less than five occurrences. MR, per Yao et al. (2019), lost no words, as its documents are single sentences. TREC, WebKB, and MIX used the same unspecified approach. There were no translation, language ID, or language specific stemming.

Implementation and training

The word and its translated version are treated as separate nodes. Yao et al. (2019), Text GCN, and the default settings were used. These include a learning rate of .02 and a sliding window of 20 words, the Adam optimizer with Srivastava et al. (2014) dropout of .5, no L2 weight regularization, 200 maximum epochs, and 10% of the training documents set aside as validation data. If the validation loss did not improve in 10 epochs, training was stopped (Yao et al. (2019)). The ReLU and Leaky ReLU models differed only in the value of ϕ in Equation (4). All of the experiments were

conducted on a desktop without a dedicated graphics card. Each model was trained once for each corpus. A measure of training time was taken for each individual instance.

Evaluation metrics

Test accuracy, along with precision, recall, and F1 scores, was also evaluated. Metrics were evaluated in both their macro- and weighted- sense. These metrics were evaluated using scikit-learn (Pedregosa et al., 2011). Evaluating metrics in a macro-sense affects small classes the most. As a result, these evaluations often show the poor performance on small classes that accuracy conceals. For the single-label multi-class scenario we are evaluating, we found that weighted recall was the same as accuracy. This relationship helped us verify the values we report.

Results

Classification performance

Test accuracy, as reported in Table 2, shows that Leaky ReLU and ReLU were mostly the same across different systems. The largest accuracy difference was observed to be 0.9 for Ohsumed and 0.7 for MIX. The average difference across the 8 systems was 0.3. Leaky ReLU showed lower accuracy than ReLU for MR and WebKB by 0.1 and was the same for TREC. All these values are based on a single execution. As a result, we cannot consider any of these accuracy differences as improvements (see Section 6.3).

Table 2: Test accuracy (%) of Text GCN with ReLU and Leaky ReLU hidden activations. Δ = Leaky ReLU – ReLU in percentage points. The last column gives the mean $\pm$ standard deviation over 10 runs reported by Yao et al. (2019) for the original ReLU model.

Dataset	ReLU	Leaky ReLU	Δ	Yao et al. (2019)
20NG	84.9	85.1	+0.2	86.34 $\pm$ 0.09
R8	95.2	95.5	+0.3	97.07 $\pm$ 0.10
R52	92.1	92.5	+0.4	93.56 $\pm$ 0.18
Ohsumed	67.8	68.7	+0.9	68.36 $\pm$ 0.56
MR	75.9	75.8	−0.1	76.74 $\pm$ 0.20
TREC	90.7	90.7	0.0	—
WebKB	85.7	85.6	−0.1	—
MIX	57.3	58.0	+0.7	—

Weights for macro-averaged precision, recall, and F1 scores were also reported, as shown in Table 3. These scores followed the same pattern as accuracy. The largest changes were measured for macro-averaged recall and F1 scores for R52 (+2.4 and +1.8 points, respectively) and Ohsumed (+1.1 and +1.4 points, respectively). These two datasets also contained a large imbalance of class sizes. For these datasets, Leaky ReLU was able to retrieve a greater number of documents that were in the small classes. The size of the difference we observed was also within the range of value variability

we expect from one execution to the next. Accuracy and macro-F1 were plotted in Figure 3 for visual evaluation.

Table 3. Macro-averaged precision (P), recall (R) and F1, and weighted F1 (%), for ReLU (Re) and Leaky ReLU (LR).

Dataset	Macro-P		Macro-R		Macro-F1		Weighted-F1	
	Re	LR	Re	LR	Re	LR	Re	LR
20NG	84.9	85.1	84.5	84.7	84.5	84.7	84.9	85.1
R8	93.2	93.4	88.3	89.1	89.7	90.5	95.4	95.4
R52	74.5	74.9	67.4	69.8	68.4	70.2	91.8	92.2
Ohsumed	69.1	69.9	59.0	60.1	61.1	62.5	67.2	68.2
MR	76.0	75.8	75.9	75.9	75.9	75.7	75.9	75.7
TREC	89.6	89.6	91.0	91.0	90.1	90.1	90.8	90.8
WebKB	55.7	55.7	55.2	55.0	55.4	55.4	85.3	85.3
MIX	54.7	55.3	53.2	53.8	53.4	54.2	57.8	58.6

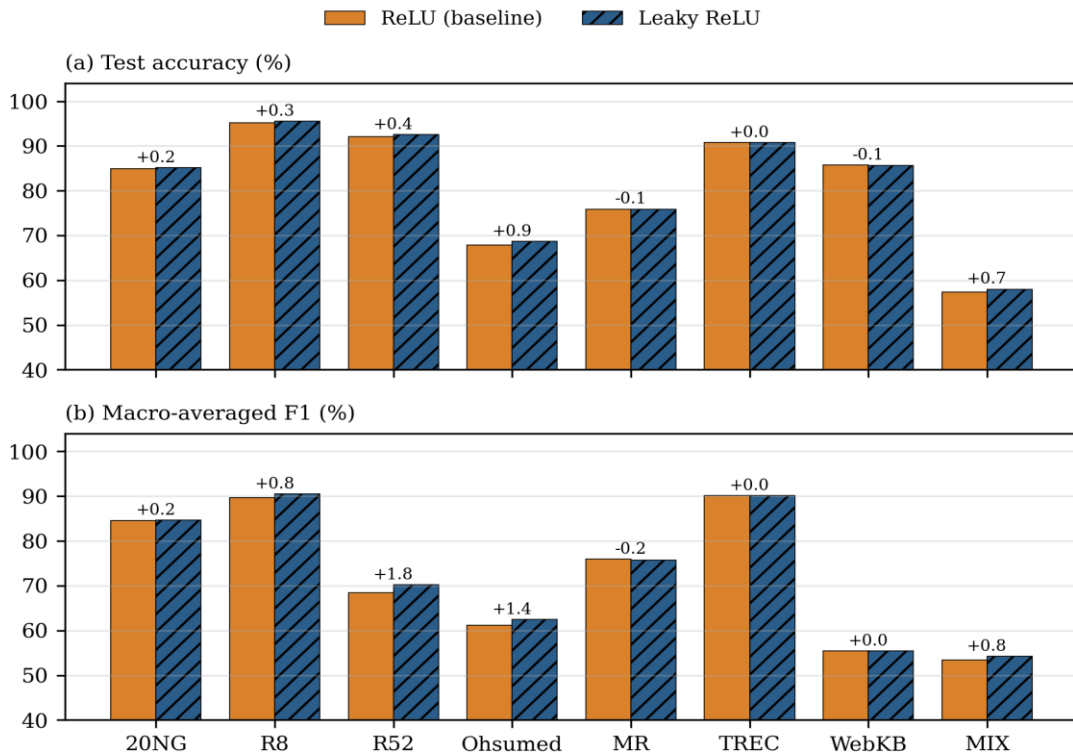


Figure 3. Test accuracy (a) and macro-averaged F1 (b) for ReLU and Leaky ReLU. Numbers above the bars give the difference (Leaky ReLU – ReLU) in percentage points.

Training time

Wall clock training time is documented in Table 4 and Figure 4. Training took a total of 79.8 min using ReLU and 63.8 min using Leaky ReLU. This is a saving of 20.1%. Taking the savings in relative and absolute terms, the largest savings were on the 20NG graph with 9.2 min (20%), and largest savings in relative terms on R8 (32%) and R52 (26%). On Ohsumed and WebKB the savings were 8–11%. MR, MIX and TREC trained in 2 mins or less with either activation, and savings were negligible with single-run time for those cases.

Table 4. Wall-clock training time (min) on the CPU-only machine.

Dataset	ReLU	Leaky ReLU	Saved	Reduction (%)
20NG	46.8	37.6	9.2	19.7
R8	5.6	3.8	1.8	32.1
R52	12.6	9.3	3.3	26.2
Ohsumed	7.6	6.8	0.8	10.5
MR	1.5	1.4	0.1	6.7
TREC	< 0.5	< 0.5	—	—
WebKB	3.6	3.3	0.3	8.3
MIX	2.0	1.5	0.5	25.0
Total	79.8	63.8	16.0	20.1

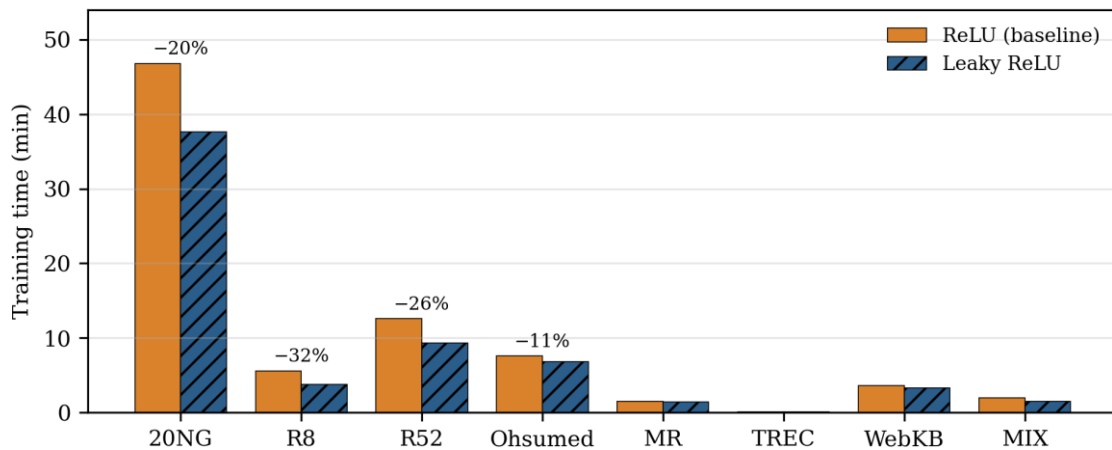


Figure 4. Training time for ReLU and Leaky ReLU. Percentages show the relative reduction for corpora whose baseline training time exceeded 4 min.

Discussion

Why classification performance did not change

A lack of an accuracy impact supports existing information of shallow GCNs. Text GCN has a single hidden nonlinearity. The depth-driven deactivation of ReLU units described by Lu et al. (2020) has little space to occur. Dropout on the hidden layer alters which units are active from one epoch to another, which likely prevents a unit from being inactive throughout the entire training process. Most of the work is done by the graph convolution as it smoothes features through neighborhoods (Li et al., 2018), and Wu et al. (2019) achieved competitive accuracy even in the absence of hidden nonlinearity. A slope with a small negative value that is on the order of 0.01 has a slight impact on reshaping the function, so similar decision boundaries at convergence are to be expected.

Where the time saving comes from

Section 3.3 showed that the activation has almost no effect on the cost of one epoch. The reduction in total time must therefore come from fewer epochs. Text GCN has an early stopping criterion which terminates training after 10 epochs of no improvement in validation loss. Design changes resulting in improvements in the validation loss (and therefore the validation loss curve) will have an impact on the termination criteria. The use of Leaky ReLU provides a gradient path to the hidden units if the pre-activation values are negative. This may allow the validation loss to reach a plateau sooner. This cannot be confirmed with the current data since the stopping epoch was not recorded. Two possible explanations still remain. The stopping epoch was determined by the random initialization of the model, and the wall-clock time in the CPU may have varied based on the other processes running concurrently. Each of these explanations is plausible for the observed differences in a single run. The stopping epoch should be recorded for multiple random initializations to obtain the mean time per epoch in order to differentiate the possible causes of the observations.

The practical effect is that the savings occur in the most important parts. The three largest graphs (20NG, R52 and R8) brought most of the savings at 14.3 of 16.0 minutes. Those tuning Text GCN on a CPU can implement a single line of code, save 1/5 of the total training time, and keep the same accuracy, which allows for more variations of the model to be run in a single trial.

Comparison with published Text GCN results

Our ReLU baseline outperformed the ten-run means in Yao et al. (2019) by 1.4 on 20NG, 1.9 on R8, 1.5 on R52, 0.6 on Ohsumed, and 0.8 on MR. These differences may be due to using a single run, as opposed to a ten-run average, software version differences, hardware differences, and differences in numerical libraries. The standard deviations provided help evaluate the activation differences. On 20NG, R8, and R52 the standard deviations are 0.1–0.2 points and thus a 0.2–0.4-point difference could be a real, but an extremely small effect. On Ohsumed (standard deviation 0.56) and MR

(0.20), the reported differences are within about 2 standard deviations. A repeated measures paired test is necessary to substantiate any claims of accuracy improvements.

Multilingual and imbalanced corpora

MIX achieved the worst results, around 58% accuracy. The average number of tokens in a MIX text is short (36.7) and texts are multilingual. Thus, words in English, French, and Spanish that are synonymous exist in separate nodes and the co-occurrence edges that Text GCN relies on are distributed in three distinct vocabularies. Afzal et al. (2016) were the first to create this corpus and were able to categorize the videos by integrating contextual text features and the concept-level semantics pulled from the visual content, so their findings are not comparable to the text-only findings reported here.

WebKB and R52 establish significant discrepancies between accuracy and macro-averaged scores (WebKB: Accuracy: ~86%, Macro-F1: ~55%; R52: Accuracy: ~92%, Macro-F1: ~70%). Sizable classes dominate the datasets. The model performs well for those classes, but numerous smaller classes tend to be misclassified. For these types of datasets, accuracy is an overestimation of performance. Macro-averaged metrics need to be included.

Limitations

Each configuration was only trained on one occasion, limiting the ability to estimate variance and test for significance. Due to this, the timing results in particular must be recast with an arbitrary number of random seeds on a different machine (more or less idle). The number of training epochs was not logged, leaving the proposed mechanism for the time savings untested. Only one instance of the negative slope was examined. The comparison was only made with ReLU and Leaky ReLU; other layers including parametric ReLU, ELU, and a linear (SGC-style) hidden layer were not explored but would be logical additions. All other experiments were conducted on a CPU, and as with other studies, the relative gain may differ on a GPU. Finally, Text GCN is transductive. Test documents must be included at the time the document graph is created, which constrains work with a continuous stream of new documents (Yao et al., 2019; Huang et al., 2019).

Conclusion

Using Leaky ReLU in place of ReLU in the hidden layer of Text GCN kept test accuracy on eight corporas within a percentage point of the baseline and reduced the total CPU training time by about 20%. Most of the savings were on the biggest graphs. Since this change is literally cost free and benefits anyone that trains Text GCN without a GPU, it's clearly worth adopting. Only three things are needed to make the observation more concrete: repeated-seed experiments with the stopping epoch and time-per-epoch recorded, a sweep over the negative slope, and testing on GPU. The poor results from the multilingual MIX corpus indicate that the construction of the graph (versus the activation function) is the limiting factor and will, likely, be the new focus for low-resource and mixed-language text.

Declarations

1. *Data availability.* 20NG, R8, R52, Ohsumed and MR are available with the Text GCN code of Yao et al. (2019). TREC and WebKB are public benchmark corpora (Li & Roth, 2002; Craven et al., 1998). The MIX corpus was obtained from Afzal et al. (2016).
2. *Code availability.* The experiments build on the public Text GCN code (https://github.com/yao8839836/text_gcn). The modified code is available from the corresponding author on reasonable request.
3. *Funding.* This research received no external funding.
4. *Competing interests.* The authors declare no competing interests.
5. *Use of AI-assisted tools.* Generative AI tools were used to help prepare and edit the manuscript text. The authors reviewed all content and take full responsibility for it.

References

- Afzal, M., Wu, X., Chen, H., Jiang, Y.-G., & Peng, Q. (2016). Web video categorization using category-predictive classifiers and category-specific concept classifiers. *Neurocomputing*, 214, 175–190. <https://doi.org/10.1016/j.neucom.2016.06.004>
- Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*. <https://arxiv.org/abs/1806.01261>
- Chenthamarakshan, V., Melville, P., Sindhwani, V., & Lawrence, R. D. (2011). Concept labeling: Building text classifiers with minimal supervision. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011)*.
- Church, K. W., & Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1), 22–29. <https://aclanthology.org/J90-1003>
- Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (ELUs). In *International Conference on Learning Representations (ICLR 2016)*. <https://arxiv.org/abs/1511.07289>
- Conneau, A., Schwenk, H., Barrault, L., & LeCun, Y. (2017). Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)* (pp. 1107–1116). <https://doi.org/10.18653/v1/E17-1104>
- Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (1998). Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)* (pp. 509–516).
- Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29* (pp. 3844–3852). <https://arxiv.org/abs/1606.09375>

- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*(pp. 4171–4186). <https://doi.org/10.18653/v1/N19-1423>
- Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503, 92–108. <https://doi.org/10.1016/j.neucom.2022.06.111>
- Frei, S., Vardi, G., Bartlett, P. L., Srebro, N., & Hu, W. (2023). Implicit bias in leaky ReLU networks trained on high-dimensional data. In *International Conference on Learning Representations (ICLR 2023)*.<https://arxiv.org/abs/2210.07082>
- Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, PMLR 15 (pp. 315–323). <https://proceedings.mlr.press/v15/glorot11a.html>
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp. 1026–1034). <https://doi.org/10.1109/ICCV.2015.123>
- Hersh, W., Buckley, C., Leone, T. J., & Hickam, D. (1994). OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of SIGIR '94*(pp. 192–201). https://doi.org/10.1007/978-1-4471-2099-5_20
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Huang, L., Ma, D., Li, S., Zhang, X., & Wang, H. (2019). Text level graph neural network for text classification. In *Proceedings of EMNLP-IJCNLP 2019* (pp. 3444–3450). <https://doi.org/10.18653/v1/D19-1345>
- Johnson, R., & Zhang, T. (2015). Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of NAACL-HLT 2015* (pp. 103–112). <https://doi.org/10.3115/v1/N15-1011>
- Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Volume 2* (pp. 427–431). <https://doi.org/10.18653/v1/E17-2068>
- Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of EMNLP 2014*(pp.1746–1751). <https://doi.org/10.3115/v1/D14-1181>
- Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR 2015)*.<https://arxiv.org/abs/1412.6980>
- Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR 2017)*. <https://arxiv.org/abs/1609.02907>
- Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4), 150. <https://doi.org/10.3390/info10040150>

- Lang, K. (1995). NewsWeeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML 1995)* (pp. 331–339). <https://doi.org/10.1016/B978-1-55860-377-6.50048-7>
- Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, PMLR 32(2) (pp. 1188–1196). <https://proceedings.mlr.press/v32/le14.html>
- Li, Q., Han, Z., & Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1). <https://doi.org/10.1609/aaai.v32i1.11604>
- Li, X., & Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*. <https://doi.org/10.3115/1072228.1072378>
- Lin, Y., Meng, Y., Sun, X., Han, Q., Kuang, K., Li, J., & Wu, F. (2021). BertGCN: Transductive text classification by combining GNN and BERT. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021* (pp. 1456–1462). <https://doi.org/10.18653/v1/2021.findings-acl.126>
- Liu, P., Qiu, X., & Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*(pp. 2873–2879). <https://arxiv.org/abs/1605.05101>
- Liu, W., Liu, X., & Chen, X. (2023). An inexact augmented Lagrangian algorithm for training leaky ReLU neural network with group sparsity. *Journal of Machine Learning Research*, 24(212), 1–43. <https://jmlr.org/papers/v24/liu22-0491.html>
- Liu, X., You, X., Zhang, X., Wu, J., & Lv, P. (2020). Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05), 8409–8416. <https://doi.org/10.1609/aaai.v34i05.6359>
- Lu, L., Shin, Y., Su, Y., & Karniadakis, G. E. (2020). Dying ReLU and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28(5), 1671–1706. <https://doi.org/10.4208/cicp.OA-2020-0165>
- Luo, Y., Uzuner, Ö., & Szolovits, P. (2017). Bridging semantics and syntax with graph algorithms: State-of-the-art of extracting biomedical relations. *Briefings in Bioinformatics*, 18(1), 160–178. <https://doi.org/10.1093/bib/bbw001>
- Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *ICML 2013 Workshop on Deep Learning for Audio, Speech and Language Processing*.
- Mastromichalakis, S. (2020). ALReLU: A different approach on Leaky ReLU activation function to improve neural networks performance. *arXiv preprint arXiv:2012.07564*. <https://arxiv.org/abs/2012.07564>
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*(pp. 3111–3119). <https://arxiv.org/abs/1310.4546>

- Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2021). Deep learning-based text classification: A comprehensive review. *ACM Computing Surveys*, 54(3), Article 62. <https://doi.org/10.1145/3439726>
- Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)* (pp. 807–814).
- Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the ACL* (pp. 115–124). <https://doi.org/10.3115/1219840.1219855>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. <https://jmlr.org/papers/v12/pedregosa11a.html>
- Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., & Yang, Q. (2018). Large-scale hierarchical text classification with recursively regularized deep graph-CNN. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)* (pp. 1063–1072). <https://doi.org/10.1145/3178876.3186005>
- Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of EMNLP 2014* (pp. 1532–1543). <https://doi.org/10.3115/v1/D14-1162>
- Rousseau, F., Kiagias, E., & Vazirgiannis, M. (2015). Text categorization as a graph classification problem. In *Proceedings of ACL-IJCNLP 2015, Volume 1* (pp. 1702–1712). <https://doi.org/10.3115/v1/P15-1164>
- Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513–523. [https://doi.org/10.1016/0306-4573\(88\)90021-0](https://doi.org/10.1016/0306-4573(88)90021-0)
- Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., Li, C., Henao, R., & Carin, L. (2018). Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of ACL 2018, Volume 1* (pp. 440–450). <https://doi.org/10.18653/v1/P18-1041>
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56), 1929–1958. <https://jmlr.org/papers/v15/srivastava14a.html>
- Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*. <https://arxiv.org/abs/1503.00075>
- Tang, J., Qu, M., & Mei, Q. (2015). PTE: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1165–1174). <https://doi.org/10.1145/2783258.2783307>

- Wang, G., Li, C., Wang, W., Zhang, Y., Shen, D., Zhang, X., Henao, R., & Carin, L. (2018). Joint embedding of words and labels for text classification. In *Proceedings of ACL 2018, Volume 1* (pp. 2321–2331). <https://doi.org/10.18653/v1/P18-1216>
- Wang, S., & Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of ACL 2012, Volume 2* (pp. 90–94). <https://aclanthology.org/P12-2018>
- Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019). Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, PMLR 97 (pp. 6861–6871). <https://proceedings.mlr.press/v97/wu19e.html>
- Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4–24. <https://doi.org/10.1109/TNNLS.2020.2978386>
- Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*. <https://arxiv.org/abs/1505.00853>
- Yao, L., Mao, C., & Luo, Y. (2019). Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01), 7370–7377. <https://doi.org/10.1609/aaai.v33i01.33017370>
- Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28* (pp. 649–657). <https://arxiv.org/abs/1509.01626>