Text Classification based on Spiking Neural Network with LSTM Conversion

Shaoheng Zhou[1,a,*]

[1]Qingdao Institute of Software College of Computer Science and Technology, China University of Petroleum (East China), Qingdao, Shandong Province, 266580, China
a. 964014846@qq.com
**corresponding author*

Abstract: In recent years, spiking neural network (SNN) have attracted much attention due to their low energy consumption and have achieved remarkable results in the fields of vision and information processing. However, the application of SNNs in the field of natural language processing (NLP) is still relatively small. Given that current popular large-scale language models rely on huge arithmetic and energy consumption, it is of great practical importance to explore SNN-based approaches to implement NLP tasks in a more energy-efficient way. This paper investigates the conversion method from LSTM network to SNN and compares the performance of the original LSTM network with the converted SNN in a text classification task. The paper experimentally compares the accuracy of different LSTM-based models using different methods on the same dataset. The experimental results show that the converted SNN is able to achieve similar performance to the original LSTM network with significantly lower power consumption in text classification tasks on multiple datasets.

Keywords: Spiking Neural Network, LSTM, NLP, Text Classification.

1. Introduction

Natural Language Processing (NLP), an important part of the intersection of Artificial Intelligence and Linguistics, has achieved remarkable successes in tasks such as text classification as technology continues to advance. However, these achievements are often built on top of computationally intensive deep learning models, which, despite their outstanding performance, are backed by equally impressive energy consumption and computational resource requirements. For example, the training of the GPT-3 model consumed about 1,287 MWh of energy, while OpenAI runs ChatGPT at about 564 MWh per day [1].

At a time when the amount of text data is proliferating, the cost of traditional deep learning models in terms of storage and processing has risen. Meanwhile, with the popularity of NLP applications, from data centres to personal devices, there is a growing demand for low-power operation. In this context, as an emerging neuromorphic computing model, the spiking neural network (SNN) shows potential for application in NLP tasks such as text classification due to its high performance and low power consumption characteristics in simulating the human brain in processing information.SNN provides new possibilities for achieving more resource-efficient text classification models by simulating the spiking activities of biological neurons for information processing. In addition, SNNs

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

based on conversion methods have been shown to have accuracy comparable to traditional deep neural networks on a variety of tasks, while performing better in terms of energy consumption [2, 3].

In fact, text classification tasks mainly rely on two types of deep learning models: recurrent neural networks (RNNs) and convolutional neural networks (CNNs.) RNNs have been widely used in text classification due to their advantages in processing sequential data. Long Short-Term Memory Network (LSTM) [4], an RNN architecture with long short-term memory units as hidden units, has been shown to have great potential in extracting advanced textual information.

Currently, the application and exploration of Spiked Neural Networks (SNNs) as an efficient low-energy computational model for text classification tasks is still in its infancy. The aim of this study is to explore the application of SNN based on LSTM conversion in text classification tasks and to evaluate whether it can maintain classification accuracy while operating at lower energy consumption using existing ANN to SNN conversion methods [5]. By comparing the performance of SNNs with that of traditional artificial neural networks, this study aims to demonstrate that SNNs based on LSTM conversion can not only significantly reduce energy consumption on text classification tasks, but also achieve comparable performance to traditional LSTMs, thus providing a more energy-efficient method for text classification in the field of NLP.

2. Research method

2.1. Data sources

In this study, we use five text classification datasets for model performance evaluation, including three English datasets and two Chinese datasets. The details are as follows:

- SST-5: A five-category version of the Stanford Sentiment Treebank containing 11,855 sentence samples designed for a sentiment classification task. The dataset covers five emotion categories: very negative, negative, neutral, positive and very positive.
- SST-2: As a dichotomised version of the SST-5, this dataset contains only two sentiment categories: positive and negative. Despite the reduced number of categories, the SST-2 is still of significant research value in the field of sentiment analysis.
- IMDb: This dataset consists of movie review texts divided into a training set and a test set containing 20,000 and 2,000 samples, respectively.The goal of the IMDb dataset is to determine the sentiment tendency of a review, i.e., a positive or a negative evaluation.
- ChnSenti: This is a Chinese dataset containing about 7,000 customer reviews of hotels in China, each of which is labelled with positive or negative sentiment.The ChnSenti dataset provides rich research material for Chinese sentiment analysis.
- TNEWS: The dataset is derived from the news section of today's headlines, covering 17 news categories, including travel, education, finance, military, and other fields. The size of the dataset is large, including 53,360 samples in the training set, 10,000 samples in the validation set, and 10,000 samples in the test set.

These datasets vary in size of the examples and length of the text. In the absence of a standard training-testing split, we randomly selected 10% of the examples from the entire dataset as the test set.

2.2. Word embedding and word encoding

In the field of Natural Language Processing (NLP), the conversion of raw textual data into numerical representations is a crucial step. This conversion is usually achieved by word embedding techniques, which map discrete text units (e.g., words, phrases) into a continuous n-dimensional space to form "word vectors". These word vectors capture rich semantic information, enabling deep learning models to effectively process and analyse linguistic data. Pre-trained word embedding models have shown significant effectiveness in a variety of NLP tasks, and in this study, they also play a key role in our Stimulus Neural Network (SNN) model, as shown in Figure 1.

In this study, we used the following publicly available pre-trained word embedding models to transform text data. For English text, we used the GloVe model pre-trained word vectors, which were trained based on a large-scale text corpus including the Wikipedia 2014 version and the Gigaword 5 dataset, with a dimension of 300 for each word vector [6]. For Chinese text, we used pre-trained word vectors from the Word2Vec model, which were obtained by training based on text datasets from multiple sources including Baidu encyclopaedia, Wikipedia Chinese pages, People's Daily, Sogou news, financial news, and microblogging, with the same dimensionality of 300 for each word vector [7].

When applying spiking neural networks (SNNs) in natural language processing tasks, the text data must be encoded as a sequence of spikes, but the values of the word embeddings are not always positive, and we use the following method [2] to convert these embeddings into non-negative vectors. The mean (μ) and standard deviation (σ) of the word embeddings were first calculated. Subsequently, the data was pruned to remove outliers and reduce the skewness of the data by restricting the embedding values to the mean plus or minus three times the standard deviation (μ-3σ , μ+3σ). The pruned data were standardized by subtracting the mean and dividing by six times the standard deviation, giving the data a zero mean, and reducing the scale of the data by dividing by a larger multiple. Finally, to ensure that all values were non-negative and suitable for SNN processing, we converted the standardized values to non-negative values by restricting the maximum value to one and ensuring that all values did not fall below zero. This series of conversion steps not only preserves the semantic information of the original data, but also provides an appropriate input format for SNN. The specific formula is as follows:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{1}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(X_i - \mu\right)^2} \tag{2}$$

$$X_i^{'} = \max(\min(X_i, \mu + 3\sigma), \mu - 3\sigma) \tag{3}$$

$$X_i^{''} = \frac{X_i^{'} - \mu}{6\sigma} \tag{4}$$

$$X_i^{'''} = \max(\min(X_i^{''}, 1), 0) \tag{5}$$

where N is the number of word embeddings and $X_i$is the ith embedding, and μ is the mean of word embeddings, and σ is the standard deviation of word embeddings.

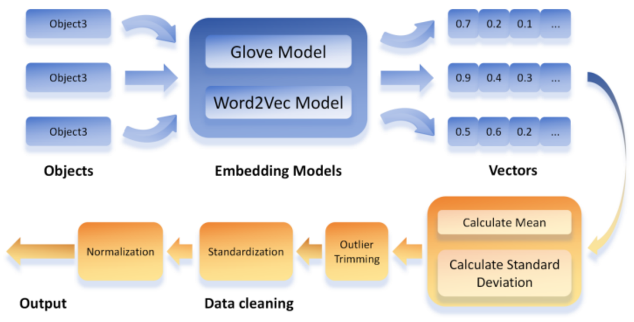

Figure 1: Flowchart of word embedding coding

2.3. Neural network model customisation and training

Despite the potential of spiking neural networks (SNNs) in modelling biological neural systems, their training process is limited by the available backpropagation algorithms and activation functions. To overcome this challenge, this study employs a conversion strategy. Specifically, we first train a custom artificial neural network (ANN) using a backpropagation algorithm, and then map the learned weights of this network into SNNs with the same topology. In constructing the customised ANN, we took the following steps to adapt to the characteristics of the SNN [5]. We adapted all the nonlinear activation functions and replaced them with modified linear units (ReLUs) except for the original activation function inside the LSTM layer.The ReLU function is defined as ReLU(x) = max(x, 0). In addition, to further simplify the model, we set all bias terms to zero. This strategy not only ensures a stronger correlation between the ReLU activation function in the ANN and the issuance rate of the IF neurons in the SNN, but also reduces the number of parameters of the model and improves the training efficiency.

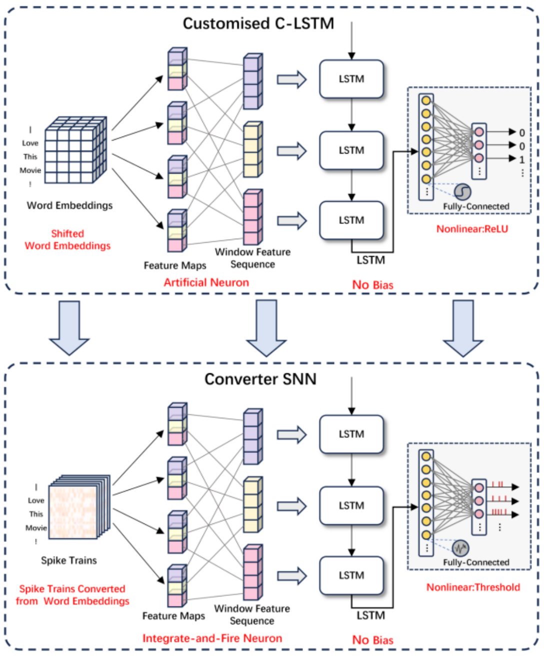

Figure 2: Architecture of customised neural network and converted spiking neural network

We customise LSTM-based artificial neural networks as specified. As an example, the C-LSTM model [8], which is suitable for text classification tasks, combines the strengths of Convolutional Neural Networks (CNNs) and Long Short-Term Memory Networks (LSTMs.) CNNs are responsible for extracting local phrasal features in the text, whereas LSTMs further process these features to obtain sentence-level semantic representations. This combination enables C-LSTM to effectively capture both local features and global semantic information of the text. In the original C-LSTM model, the pooling layer after the CNN layer is omitted because the pooling operation may destroy the sequentiality of the features, which is detrimental to the subsequent LSTM layer. Instead, the output of the CNN layer (i.e., feature mapping) is directly serialised to maintain its original order and passed as input to the LSTM layer. In this study, we improved the C-LSTM model, including replacing the original nonlinear activation function with the ReLU activation function and removing bias terms in all convolutional and fully connected layers, as shown in Figure 2.

In our study, we use positive word vector embeddings as inputs to the artificial neural network, the hidden state of the LSTM at the last time step as the representation vector of the document, and add a softmax layer at the end for classification. We train the whole model by minimising the cross-entropy error through gradient descent algorithm. The loss function is defined as follows:

$$\mathrm{L} = -\frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{c}=1}^{\mathrm{C}}\mathrm{y}_{\mathrm{ic}}\log(P_{ic}) \tag{6}$$

where N is the total number of training samples and y_{ic} is an indicator variable, if sample i belongs to category c, then$\mathrm{y}_{\mathrm{ic}} = 1$, otherwise$\mathrm{y}_{\mathrm{ic}} =0$. P_{ic} is the probability that the model predicts that sample i belongs to category c.

In this study, all network models containing a Long Short-Term Memory (LSTM) layer were configured with a uniform hidden layer dimension in order to maintain consistent representation capabilities across model variants. Specifically, the hidden layer dimension of the LSTM layer was set to 150.The convolutional-Long Short-Term Memory network (C-LSTM) model architecture used in this study contains one convolutional layer with one LSTM layer. Two different filter configuration schemes were explored for the configuration of the convolutional layer:

a) Single Convolutional Layer Configuration: in this setup, we choose a filter of length 3 and the number of filters are all set to 150.

b) Multiple Convolutional Layers in Parallel Configuration: in addition, we used convolutional layers with different filter lengths, specifically including filters of lengths 3, 4, and 5, with the number of filters of each length set to 150.

For the training of spiking neural networks (SNNs), we used the SnnTorch framework. In both the customised artificial neural network (ANN) and the converted SNN, we used an integrated prediction strategy [9]. Specifically, each category was assigned 10 neurons, and the outputs of these neurons were integrated to obtain the final classification results.

During the training of the custom network, we set the following hyperparameters: dropout rate of 0.5, batch size of 32, and learning rate of 1 × 10-4. In the training and fine-tuning phase of the SNN, we adjusted the following parameters: time steps was set to 50, membrane potential threshold (Uthr) was set to 1, membrane potential decay rate (β) was set to 1, batch size was adjusted to 50, and the learning rate was refined to 5×10-5.

2.4. Conversion of the model

The ANN to SNN conversion strategy used in this study centres on reinterpreting the activation outputs of artificial neural networks as spike issuance frequencies, achieving a correspondence between the outputs of the deep learning model and the pulse frequencies of the neurons in the spiking neural network. This mapping mechanism not only ensures the continuity of information transfer, but also improves the computational efficiency of the model on neuromorphic hardware.

As shown in Figure 2, the conversion operation consists of the following key steps: first, a spike generation mechanism is introduced. This mechanism is implemented through an integrated spike generation module, which is responsible for generating Poisson-distributed spike trains based on the learnt word vectors. This generation mechanism is the basis for converting the output of the ANN into SNN pulse sequences. Secondly, the neuron model was updated. The neurons in the original network were replaced by Leaky Integrate-and-Fire (LIF) neurons with leakage properties. Then, the conversion of synaptic weights was performed. The weights of the convolutional and fully connected layers in the original network were directly mapped to the synaptic weights of the converted SNN. Finally, the activation mechanism was simplified. In the converted SNN, the ReLU activation function was omitted, and instead, the activation function was implicitly implemented by adjusting the threshold of the LIF neurons.

2.5. Model Post-Processing

In order to enhance the performance of models after conversion from artificial neural networks (ANN) to spiking neural networks (SNN), researchers have explored a variety of optimisation strategies. Diehl et al [10] provide an in-depth analysis of the conversion process, pointing out the importance of the parameter selection of the pulsed neurons during the conversion process, and propose a series of optimisation techniques to minimise the performance loss. Among them, two new weight normalisation methods were introduced to regulate the discharge rate, which are capable of ensuring low-latency classification of the converted SNN right after the first output pulse. They proposed a model-based normalisation approach which enhances robustness to high input rates by ensuring that the same neuron does not fire multiple pulses in a single time step. This strategy helps to maintain the stability of the SNN and improves its adaptability to input variations. In addition, they proposed a data-based normalisation method which uses a training set to estimate typical activation values and normalises the weights accordingly, with the aim of ensuring that a single pulse is generated at each time step. Such a treatment not only improves the classification accuracy of SNNs, but also their overall efficiency. Both normalisation methods can improve the performance of SNNs without increasing the training time compared to previous SNN methods.

In addition, Lv et al [2] proposed a novel fine-tuning method which uses an alternative gradient method on the transformed SNN for further fine-tuning on the same dataset. Specifically, they employed the Backpropagation Through Time (BPTT) algorithm and trained the impulse neural network with fast sigmoid as a proxy gradient function. This technique improves the applicability and performance of SNNs in text classification tasks.

3. Results and analysis

The experimental results are detailed in Table 1, which shows the classification accuracies obtained by different models using different methods on the five datasets. We denote the customised original artificial neural network as "ANN", the SNN trained directly using alternative gradients as "SNN", the converted unprocessed SNN as "Conv SNN ", the model obtained by applying model-based normalisation to the transformed SNN is denoted as "Conv SNN+MN", the model obtained by data-based normalisation is denoted as "Conv SNN+DN" and the fine-tuned model is denoted as "Conv SNN+FT".

Table 1 shows the performance comparison of SNNs trained using the conversion method with the original ANNs and directly trained SNNs on five text classification datasets. On the English dataset, compared with the original ANN, the total average accuracy of SNNs decreases by 3.87%, in which the average accuracy of "Conv SNN" decreases by 4.32%, while the average accuracy of "Conv SNN+FT" only decreases by 1.97%. A similar trend is observed in the Chinese dataset, where the total average accuracy decreases by 3.03%, with "Conv SNN+FT" showing the smallest decrease of 1.42%. This indicates that the performance of SNN trained by the conversion method is very close to that of ANN.

It is worth noting that compared with the directly trained SNNs, the SNNs trained using the conversion method not only greatly reduce the training time and difficulty, but also show significant performance improvement on both the English and Chinese datasets. The overall average accuracy on the English dataset is improved by 13.39%, and the "Conv SNN+FT" shows the most significant improvement of 15.29%. The Chinese dataset shows a similar improvement, with an overall average accuracy improvement of 11.81%, with "Conv SNN+FT" showing an average accuracy improvement of 13.42%.

Although the accuracy of the SNN model based on LSTM conversion on various datasets fails to reach the level of the current state-of-the-art model, the core of this study lies in the comparative analysis with the original model. We observed that the accuracy of the SNN models after using various conversion methods did not show a significant decrease compared to the original model on different datasets, and all of them were maintained within an acceptable range. In addition, we find that the performance of the models can be further improved by applying suitable post-processing to them.

Table 1: Classification accuracies obtained by different models using different methods on 5 datasets

MODEL	**METHOD**	**ENGLISH DATASET**			**CHINESE DATASET**	
		SST-2	SST-5	imdb	TNEWS	ChnSenti
LSTM	ANN	82.15	41.62	70.05	56.40	84.59
	SNN	50.25	23.08	51.75	32.17	68.90
	Conv SNN	76.00	38.51	65.65	53.01	82.92
	Conv SNN + MN	75.62	37.15	66.10	52.03	81.93
	Conv SNN + DN	76.51	39.05	65.72	54.21	82.88
	Conv SNN + FT	79.90	38.73	68.80	54.97	84.29
BI-LSTM	ANN	82.59	41.86	70.20	57.42	85.10
	SNN	75.29	28.64	50.74	34.25	74.84
	Conv SNN	76.99	38.64	59.65	52.90	81.94
	Conv SNN + MN	77.05	37.01	59.38	52.72	81.46
	Conv SNN + DN	77.10	39.59	59.28	54.58	82.23
	Conv SNN + FT	81.11	38.24	67.08	54.57	84.76
C-LSTM (A)	ANN	78.86	41.81	69.50	54.88	82.16
	SNN	69.96	31.18	51.26	48.79	68.92
	Conv SNN	76.54	37.74	65.81	50.31	80.13
	Conv SNN + MN	74.52	35.84	63.95	50.12	80.09
	Conv SNN + DN	77.38	40.72	66.09	51.51	80.35
	Conv SNN + FT	77.65	40.54	68.28	52.51	81.37
C-LSTM (B)	ANN	79.90	41.13	69.47	54.82	83.32
	SNN	50.08	28.64	51.26	43.17	68.93
	Conv SNN	77.54	37.69	66.51	45.24	81.93
	Conv SNN + MN	72.65	37.51	65.83	44.92	81.90
	Conv SNN + DN	79.29	38.51	66.32	49.33	81.97
	Conv SNN + FT	77.87	38.46	68.89	52.78	82.06

4. Conclusion

This study provides a comprehensive comparison of the performance differences between the original LSTM networks used in a text classification task and the converted SNNs. The experimental results reveal an important finding: a variety of neural networks containing LSTM structures do not significantly degrade in performance after being converted to SNNs, but rather approach the performance level of the original LSTM networks on multiple datasets. In addition, we further observed significant performance improvements through the introduction of model post-processing techniques. It is worth noting that these models are achieved with significantly reduced energy consumption. This finding not only confirms the effectiveness of utilising a conversion approach when applying LSTM neural networks to text classification tasks, but also demonstrates the potential of this approach to maintain model performance stability across different data environments. This result has important implications for resource-constrained environments and energy-sensitive application scenarios. Given the positive results of this study, future work will explore the possibility of converting more complex LSTM networks to SNNs and whether such conversion can further reduce energy consumption while maintaining performance.

References

[1] de Vries, A. (2023). The growing energy footprint of artificial intelligence. Joule, 7(10), 2191-2194.

[2] Lv, C., Xu, J., & Zheng, X. (2023). Spiking Convolutional Neural Networks for Text Classification. In Proceedings of the International Conference on Learning Representations (ICLR). Retrieved from https://arxiv.org/html/2406.19230v1

[3] Hu, Y., Zheng, Q., Li, G., Tang, H., & Pan, G. (2024). Toward Large-scale Spiking Neural Networks: A Comprehensive Survey and Future Directions. arXiv preprint arXiv:2409.02111.

[4] Hochreiter, S. (1997). Long Short-term Memory. Neural Computation MIT-Press.

[5] Cao, Y., Chen, Y., & Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. International Journal of Computer Vision, 113, 54-66.

[6] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[7] Li, S., Zhao, Z., Hu, R., Li, W., Liu, T., & Du, X. (2018). Analogical Reasoning on Chinese Morphological and Semantic Relations. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Association for Computational Linguistics.

[8] Zhou, C., Sun, C., Liu, Z., & Lau, F. (2015). A C-LSTM neural network for text classification. arXiv preprint arXiv:1511.08630.

[9] Eshraghian, J. K., Ward, M., Neftci, E. O., Wang, X., Lenz, G., Dwivedi, G., ... & Lu, W. D. (2023). Training spiking neural networks using lessons from deep learning. Proceedings of the IEEE.

[10] Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S. C., & Pfeiffer, M. (2015, July). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In 2015 International joint conference on neural networks (IJCNN) (pp. 1-8). ieee.