Tightly-coupled Convolutional Neural Network with Spatial-temporal Memory for Text Classification

Shiyao Wang† and Zhidong Deng*
†*State Key Laboratory of Intelligent Technology and Systems*
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science, Tsinghua University, Beijing 100084, China
Email: sy-wang14@mails.tsinghua.edu.cn
* *Email: michael@tsinghua.edu.cn*

***Abstract*—Although several traditional models like bag of words (BOW), n-grams, and their variants of TFIDF, have exhibited high performance in the field of text classification, neural network methods such as LSTM, GRU, and convolutional neural network (CNN) are recently attracting increasing attention. Considering that CNN has surprising capabilities of extracting hierarchical features, combination of LSTM/GRU with CNN seems to be quite reasonable for semantic representation and sequence analysis. On the other hand, it is also a promising subject to enable CNN to have memory embeddings and/or recurrent pathway. In this paper, we propose a novel tightly-coupled convolutional neural network with spatial-temporal memory (TCNN-SM). It comprises feature-representation and memory functional columns. Feature-representation functional column in our TCNN-SM actually performs hierarchical feature extraction as regular CNN does while memory functional column retains memories of different granularity and fulfills selective memory for historical information. In order to validate effectiveness and efficiency of the proposed TCNN-SM, we conduct extensive experiments on AG's News public dataset. The experimental results show that our new TCNN-SM achieves 7.99% test error, which has the best performance among other existing deep learning methods and is very close to state of the art results yielded using classical n-grams algorithm.**

1. Introduction

Text classification in natural language processing (NLP) has been applied to a wide range of fields such as web search, document categorization, and information retrieval. It is expected to analyze text, extract key information and eventually assign it to pre-defined categories (e.g., sports, world). Building good representations for text classification seems to be vital to gain better performance. Most models begin with distributed representations of words or characters in a vector space. One of the earliest use of word representations dates back to 1986 due to Hinton et al., and then applied to statistical language modeling with considerable success. Many follow up works have extended the previous method both in representation quality and efficiency. [1] and [2] are the most popular models for word representations, and both of them released pre-trained vectors that can be subsequently used for further research.

After obtaining word representations, each document can be expressed in the form of word embedding vectors. The traditional text modeling uses human-designed features of text and feeds them to a classifier, such as SVMs. Some of linear models [3], [4]have shown impressive performance especially the training datasets are relatively small.

Recently, the models based on deep learning approaches, such as recurrent neural networks (RNNs) [5] and convolutional neural networks (CNNs) [6] have attracted increscent interest for text classification. The main-stream approach is to consider a text as a sequence of tokens and process them with RNN, especially a popular and successful RNN variant - Long Short-Term Memory neural network (LSTM) [7]. The most significant advantages of RNNs are the abilities to handle sequences of any length and capture long-term dependencies. On the other hand, many studies have found CNNs have surprising capabilities of extracting hierarchical features ranging from computer vision to speech recognition. Hierarchy is an effective and common way of representation information and other real-world textual data can be organized in this way. CNNs possess such natural hierarchical architecture and massive prior work has further improve its abilities [8]. It is regrettable that CNNs require fixed-length input and seem unable to remember contextual information like RNN. It must be a promising subject to enable CNN to have memory embeddings so as to learn both hierarchical and temporal representations.

In this paper, we propose a novel model that can naturally host both hierarchical and temporal information called tightly-coupled convolutional neural network with spatial-temporal memory (TCNN-SM)(see Figure 1). It comprises feature-representation and memory functional columns. Feature-representation functional column contains four hidden convolutional layers (HCL) and three pooling layers, which completely functions the same as regular CNN. In parallel, memory functional column consists of four convolutional spatial memory layers (CSML) with cycle. CSML retains memories of different granularity and receives three incoming streams from upper CSML, preceding

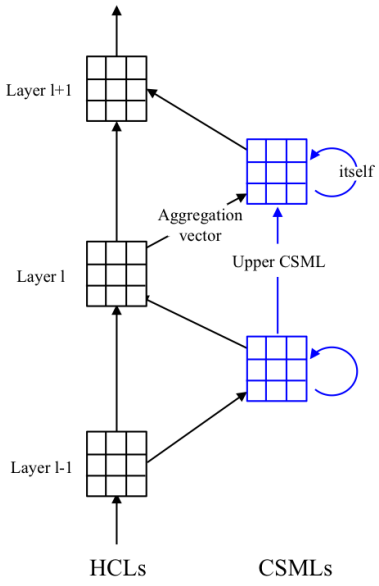

Figure 1. A simplified sample of TCNN-SM. Blocks in blue indicate the CSMLs that receive three incoming streams.

aggregation vector, and cycle itself (called temporal memory), respectively. In the same layer of TCNN-SM, outputs of HCL and CSML are concatenated into an aggregation vector, which is further employed as input of next layer. Feature-representation functional column in our TCNN-SM actually performs hierarchical feature extraction as regular CNN does, while memory functional column is exploited to fulfill selective memory for historical information through error back-propagation. Except for the above, we also utilize L1-norm as regularization in cost function, which is very beneficial to highlight of key words of text due to sparsity strengthening. Interestingly, we find out that repeated stimulus of input samples is more useful for a variable-length of sentences in text classification task compared to all-zero paddings.

We evaluate our models on AG's News public dataset. Experiment results show significant and consistent improvements compared to state-of-the-art deep learning models and is very close to state of the art results yielded using classical n-grams algorithm.

2. Related Work

2.1. Traditional NLP Method

Traditional models for text classification generally have extraction of a rich set of hand-designed features that are then fed to standard classification algorithm, where the bag of words (BOW) model is commonly employed in feature engineering, including unigrams, bigrams, n-grams or some other patterns. In the feature selection, [9], [10] aim at selecting more useful and discriminative features. More complex features can be found in [11], [12]. Classification algorithms often include logistic regression (LR), naive Bayes (NB), and support vector machine (SVM). However, the use of handcrafted input features that are required to be carefully optimized for each task seems inefficiency and these traditional models may get involved in data sparsity problem.

2.2. The Deep Learning Models

Recently, deep neural networks (Hinton and Salakhutdinov 2006) have achieved great success in many NLP tasks, including distributed representation learning [2], [13], machine translation [14], [15], and text classification [16]. Among them, recurrent neural network (RNN) and convolutional neural network (CNN) are believed to have remarkable performance.

RNN is a neural network model that is capable of dealing with variable-length input sequences and learning long and short-term dependencies. LSTM [7] is undoubtedly one of the most successful and popular RNNs, aiming to facilitate training of RNNs through solving diminishing and exploding gradient problems in deep or long structures [17]. There are a lot of literatures that exhibit LSTM ability to model long-range dependencies in NLP applications [18], [19], [20], [21]. But LSTM is a regular structure and same cell is exploited in all applications, which may be not efficiently applicable to particular task. Morever, the unrolling of sequence over time leads to the fact that recent tokens could be better memorized than older ones [22].

On the other hand, CNNs have task-specific structures, which are optimized through not only network structure improvements but also several valuable strategies such as Batch Normalization, PReLU, and Maxout [23], [24] and task-specific algorithms [25]. In text classification, some of previous works have achieved considerable results. In [26], Collobert et al. use convolutional kernels to successive windows for a sequence. Kim et al. [27] utilizes multiple filters with varying window sizes to extract multiple features. The max-over-time pooling is also employed to handle variable-length problems. In addition, Kalchbrenner et al. propose a dynamic k-max pooling mechanism in [28].that has capabilities of capturing local correlation along with extraction of high-level features. All the above approaches are based on word-level representation. [16] and [22] present modeling of text from character-level representation and achieve excellent results. In particular, [22] adopts very deep convolutional neural network like ResNet [8]. In this paper, we will take it as our strong baseline.

Considering that CNN has surprising capabilities of extracting hierarchical features, combination of RNN with CNN seems to be quite reasonable for semantic representation and sequence analysis. [29] utilizes CNN to have extraction of a sequence of high-level phrase representation and fed it into LSTM, in order to acquire representation for sequence or temporal data. [30] propose a novel multi-scale RNN model, which can learn hierarchical multi-scale structure from temporal data without explicit boundary information. To the best of our knowledge, there is no existing

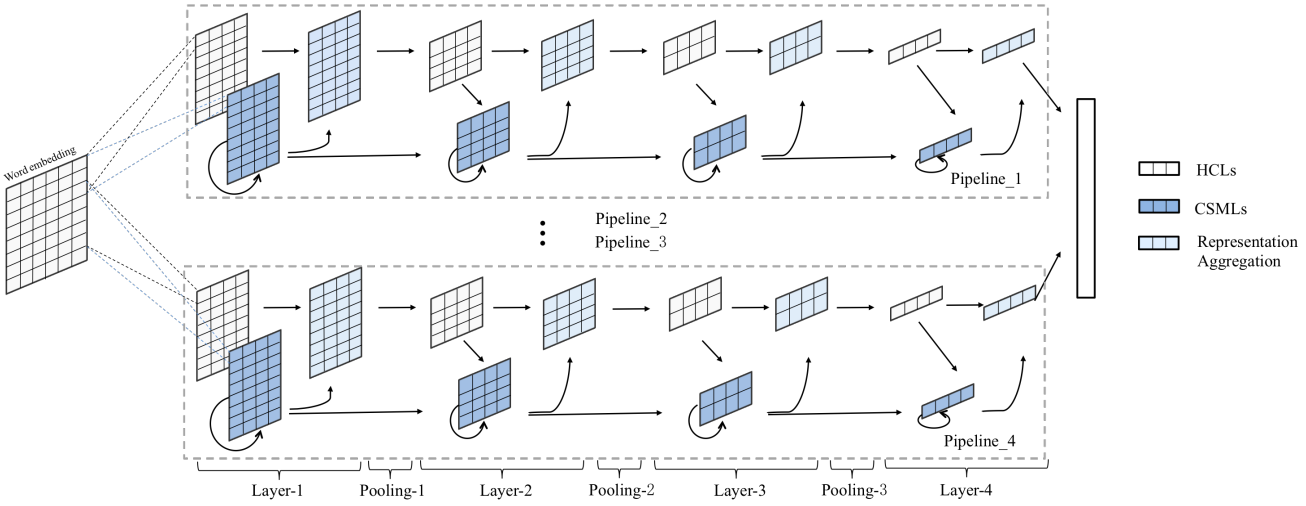

Figure 2. The proposed network for text classification. The white blocks lined up indicates the HCLs, the dark blue blocks are the CSMLs and the light blue blocks are the representation aggregation vector consist of a pair of HCL and CSML.

methods that enable regular convolutional neural network to have tightly-coupled layers with spatial-temporal memory.

3. Tightly-coupled Convolutional Neural Network with Spatial-temporal Memory

In this section, the framework of our tightly-coupled convolutional neural network with spatial-temporal memory (TCNN-SM) for text classification is at first described. Second, we focus on hidden convolutional layer (HCL) and convolutional spatial memory layer (CSML) in TCNN-SM. Finally, we introduce policies of repeated stimulus of input samples and L1-norm regularization in training process so as to further improve classification performance of the proposed TCNN-SM.

3.1. Framework of TCNN-SM for Text Classification

In this paper, we propose a novel TCNN-SM model for text classification. As illustrated in Figure 2, our TCNN-SM model takes embeddings of words in text as inputs, which are yielded through pre-trained representation vectors based on unsupervised method proposed in [1]). Inspired by parallel CNN layers in [31], our TCNN-SM model comprises four convolutional pipelines that have the identical architecture and the same parameter settings in parallel, and are eventually concatenated and fed them into fully-connected layer. This is originated from an idea to learn different local regions to enhance final accuracy. We find out that the way of parallel convolutional pipelines is usually more effective than that of simply increasing the number of filters.

Within each pipeline, our core module is composed of two functional columns: feature-representation (colored with white feature maps in Figure 2) and memory functional columns (with blue feature maps in Figure 2). The light blue maps indicate intermediate output of one layer. The feature-representation functional column contains four HCLs that functions the same as regular CNN, while the memory functional column consists of four CSMLs with cycle in parallel. A pair of HCL and CSML is regarded as a basic layer in our TCNN-SM network. The outputs of HCL and CSML are concatenated into an representation aggregation vector or intermediate output, which is further employed as input of next layer. Note that a pooling layer is located between every two basic layers in order to jointly accomplish robust and hierarchical representations.

Within each basic layer, CSML retains memories of different granularity and receives three incoming streams from upper CSML, preceding representation aggregation vector, and cycle itself (called temporal memory), respectively. These three incoming streams come with different goals. One stream from upper CSML aims to retains memories of different spatial granularity. Features in bottom layers always hold more detailed information along with some noises, whereas top layers has more robust features but loss some details. The combination of these different spatial granularity can be complementary to each other. Another stream comes from the preceding representation aggregation vector indicates the current input of this layer. The third stream contains the information from itself at the previous time step. Actually, when a document is assigned to the pre-defined categories, it is unnecessary for a model to remember all the sequential information in the content. A more sensible model is to extract key words or information in a local part of text, e.g. a few sentences. And then gather all the crucial information over time. Hence, it motivates us to divide each document into several parts, and each part is fed to the network in order. First, they will get though the HCLs, and it is expected to extract crucial information because CNN has surprising capabilities of extracting local information layer by layer and finally form a hierarchical

representation. In the mean while, they are passing through the CSMLs which retain the historical information from both the bottom layer and previous parts. And these CSMLs are intended to learn a combination of historical information and current information. Finally, the output of CSMLs and HCLs will be concatenated into a representation aggregation vector for the next layer.

3.2. Basic Layer: HCL and CSML

Let us make description of basic layer in our TCNN-SM model, which is composed of both HCL and CSML. A local part of Figure 2 can be shown as Figure 3(a). The left column represents the HCLs and the right column are CSMLs. We use h_t^l and m_t^l to denote a HCL and a CSML at the time step t of layer l, respectively. The data are passed from the bottom to the top.

Consider the TCNN-SM model of L layers (l = 1,...,L), it performs the following update at time step t and at each layer l:

$$h_t^l, m_t^l = f_{TCNN-SM}^l(h_t^{l-1}, m_t^{l-1}, m_{t-1}^l). \quad (1)$$

The function $f_{TCNN-SM}^l$ is implemented as follows. First, we will update the memory functional column m_t^l. It receives three incoming streams: preceding representation aggregation vector, upper CSML and cycle itself, which are corresponded to $Z_{h_t^{l-1}}, Z_{m_t^{l-1}} and Z_{m_{t-1}^l}$ respectively. And they will be concatenated into a memory aggregation vector m_t^l:

$$\begin{aligned} Z_{h_t^{l-1}} &= sigm(W_{zh^{l-1}}^{sigm} h_t^{l-1}) \odot tanh(W_{zh^{l-1}}^{tanh} h_t^{l-1}) \\ Z_{m_t^{l-1}} &= sigm(W_{zm^{l-1}}^{sigm} m_t^{l-1}) \odot tanh(W_{zm^{l-1}}^{tanh} m_t^{l-1}) \\ Z_{m_{t-1}^l} &= sigm(W_{zm^l}^{sigm} m_t^l) \odot tanh(W_{zm^l}^{tanh} m_t^l) \\ m_t^l &= [Z_{h_t^{l-1}}, Z_{m_t^{l-1}}, Z_{m_{t-1}^l}] \end{aligned} \quad (2)$$

Here, we use W_j^i to denote learnable filters and the term of bias is left out in all formulea. $\odot$ denotes the element-wise multiplication. $[Z_{h_t^{l-1}}, Z_{m_t^{l-1}}, Z_{m_{t-1}^l}]$ indicates the concatenation of $Z_{h_t^{l-1}}, Z_{m_t^{l-1}}$ and $Z_{m_{t-1}^l}$. $sigm$ is the logistic sigmoid function that has an output in $[0, 1]$, while $tanh$ denotes the hyperbolic tangent function that has an output in $[-1, 1]$. Above three incoming streams form into a memory aggregation vector . And then the batch normalization and ReLU activation will be adopted like the regular CNN.

$$m_t^l = ReLU(BN(m_t^l)) \quad (3)$$

The feature-representation functional column is defined as

$$\begin{aligned} R_{h_t^{l-1}} &= sigm(W_{rh^{l-1}}^{sigm} h_t^{l-1}) \odot tanh(W_{rh^{l-1}}^{tanh} h_t^{l-1}) \\ R_{m_t^l} &= sigm(W_{rm^l}^{sigm} m_t^l) \odot tanh(W_{rm^l}^{tanh} m_t^l) \\ h_t^l &= [R_{h_t^{l-1}}, R_{m_t^l}] \end{aligned} \quad (4)$$

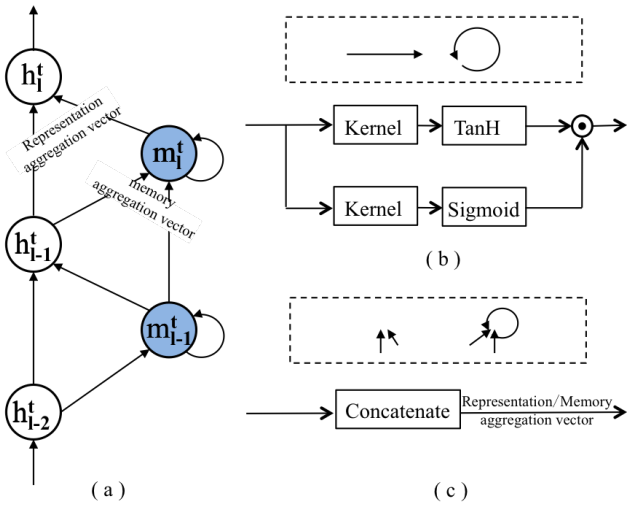

Figure 3. The detail operation in HCLs and CSMLs. Figure 3(a) is the simplified structure of Figure 2. The implementation of each straight line or arched line with an arrow is shown in Figure 3(b). And the node receives multi-stream information will be concatenated into a representation aggregation vector or a memory aggregation vector. $\odot$ denotes the element-wise multiplication.

Smiliarly, h_t^l is followed by a temporal batch norm layer and activation.

$$h_t^l = ReLU(BN(h_t^l)) \quad (5)$$

From above formulae, we can find that before each concatenation, the input information will go though a standard module, shown as Figure 3(b). The hyperbolic tangent function $tanh$ aims to normalize the different incoming information into $[-1, 1]$. We use $sigmoid$ with the intent of forgetting some of useless information. Since different local parts pass through the model over time, each operation of concatenation is meant to accumulation of information. We use a $sigmoid$ gate to drop unnecessary information in order to learn better. We achieve better performance by using this module.The comparison experiments will be exhibited in the experiment section.

3.3. Repeated Stimulus and L_1-norm regularization

Since CNNs require fixed-length input whereas the number of words vary from each document, we provide a simple but useful way to handle the variable lengths of text. In the previous work, [27] use over-time max pooling scheme to deal with variable sentence lengths. And [29] pad each sentence that has a length less than $maxlen$, with special symbols at the end that indicate the unknown words. The $maxlen$ is pre-defined empirically. It is similar to the all-zero padding vectors in [32]. In our paper, we still pad the text to a fix length. However, differ from this padding method without the semantic information, we prefer to pad each text by repeated stimulus. In other words, if

the length of the text less than $maxlen$, we can repeat the sentence over again. If a document consists of a few words, the repeated stimulus may promote our model to extract useful information. Furthermore, this padding vector is friendly to the following operation, e.g. bias terms or batch normalization without any extra gate, which keeps away the artifacts from padding in all layers. We conduct a comparison between our repeated stimulus and all-zero padding, and prove our simple method works better.

For regularization, most existed classification tasks employ L2 weight regularization [27], [29], [33], but we prefer to use L1 regularization which is more effective in text classification. The cost function with L1-norm and L2-norm weight decay can be denoted as:

$$J(w) = arg\min_{w} \sum_{i} L(y_i, f(x_i; w)) + ||w||^1 \quad (6)$$

$$J(w) = arg\min_{w} \sum_{i} L(y_i, f(x_i; w)) + \frac{\lambda}{2}||w||^2 \quad (7)$$

Where $J(w)$ denotes the cost function, w is a set of model parameters, including weights and biases. The Equ.(6) denotes the L1-norm and Equ.(7) denotes the L2-norm. The L1-norm has an ability of sparsity strengthening. We can find there many stop words in each document, e.g. "the", "a" , which may influence the feature extraction. Due to the task of text classification, models should pay attention to key words in order to extract more crucial information and suppress the expression of irrelevant words. [10] has indicated that L1 regularization of the parameters matches the best known bounds for feature selection, and L1 regularized logistic regression can be effective even if there are exponentially many irrelevant features as there are training examples. Therefore, we also conduct related experiments to prove the efficiency of L1-norm.

4. Experimental Results

In this section we evaluate effectiveness and efficiency of our TCNN-SM model based on experimental results on AG's News public dataset [16]. The AG's corpus of news article contains 496,835 categorized news articles from more than 2,000 news sources. Using title and description field, 4 largest classes are labeled,. The number of training and test samples for each class is 30,000 and 1,900, respectively, which means that totally the training dataset has 120,000 samples and the test dataset comprises 7,600 ones. The average word length in AG's News dataset is 45.

TABLE 1. WORD LENGTH OF THE DATASET AFTER PREPROCESSING.

	Minmum Length	Maxmum Length	Average Length
Training Set	4	177	37
Test Set	9	136	37

4.1. Experimental Setup

4.1.1. Data. To experiment with our TCNN-SM model, we adopt the pre-trained word representations by unsupervised method in [1] and convert the text into word embeddings. Since the pre-trained representations can not hold all the words of the dataset, we ignored the words which do not have corresponding representations. After above preprocessing, the words length of dataset is shown in Table 1. We use repeated-padding method to convert the input to a fix length - $maxlen$. In all the following experiments, we choose $maxlen = 100$.

4.1.2. Architecture. There are four CSMLs and four HCLs in a TCNN-SM shown in Figure 2. All the layers have 32 feature maps. In all experiments, our models are trained using stochastic gradient descent (SGD) algorithm with a mini-batch of 256. The learning rate is initialized to 0.01 and repeatedly decreased 2 times, until it arrives at 1e-4. We run our net through the whole training data about 20 epochs. A momentum of 0.9 is used in the entire training process to make SGD stable and fast.

As illustrated in method section, each document will be divided into n local parts. We choose $n = 4$, which means that every 25 words will be fed to our model at once in oder to extract key information among the local parts. The individual local parts are lined up by the recurrent path in the CSMLs according to time sequence. CSMLs learn selective memory of historical information and enrich the representations with HCLs.

TABLE 2. COMPARISON OF DIFFERENT CONFIGURATIONS.

Configuration	Error (%)
L1-norm, repeated-padding, single pipeline	8.43
L2-norm, repeated-padding, single pipeline	9.66
L1-norm, 0-padding, single pipeline	8.70
L2-norm, 0-padding, single pipeline	9.89
L1-norm, repeated-padding, four-pipeline	**7.99**
L1-norm, repeated-padding, wide single pipeline	8.77

4.1.3. Comparison of Different Settings. In Table 2, we show the error rate results of different configurations, which consist of three kinds of comparisons. L1-norm and L2-norm denotes the different regularization. 0-padding is putting all-zero padding vectors. Four-pipeline model is the combination of four single pipeline network. We concatenate the output until the penultimate inner product layer. And wide single pipeline model changes the former 32 feature maps to 64 feature maps. From the comparisons in Table 2, L1-norm performs better than L2-norm and repeated-padding is superior than 0-padding. Intuitively, increasing the number of feature maps will promote the model, however, it seems to be overfitted to the training set even we have raised the dropout rate. On the other hand, we use four-pipeline model, it consistently improve the performance of single pipeline model. It proves that the way of parallel convolutional pipelines is more effective than roughly increase the number of filters.

TABLE 3. COMPARISON TO PUBLIC MODELS.

Related NN models	Error (%)	Reported In
LSTM	13.94	[16]
Best w2v-Conv.	9.92	[16]
Best Lk-Conv.	8.55	[16]
Best Char-Conv.	8.67	[22]
TCNN-SM	**7.99**	**Our Model**

4.1.4. The Improvement of $sigm \odot tanh$ Gate. As an initial version, CSML in TCNN-SM is designed to directly concatenate different kinds of historical information. It is expected to further learn suitable combination of incoming streams. But it causes training process to be unstable and yields unsatisfactory results, as shown in Table 4. We find out that this CSML seems to just accumulate the past information and is lack of forgetting mechanism. It motives us to add a normalization function $tanh$ and a forgetting factor $sigm$ for output of convolutional kernel, both of which are placed before each concatenation. In Figure 3(b), outputs of $tanh$ and $sigm$ are combined through element-wise multiplication. Interestingly, it leads to stable training process, as shown in Table 4.

TABLE 4. ILLUSTRATION OF THE $sigm \odot tanh$ GATE.

	Single Pipeline	Four Pipeline
No-Gate	8.99	8.41
$Sigm \odot tanh$	8.43	7.99

4.2. Compaison with Baseline Models

After the effectiveness of our new model is validated, we conduct comparison of TCNN-SM model with baseline models, as listed in Table 3. We give another four significant neural networks: one LSTM and three CNNs. The first three are reported in [16] and the Best Char-Conv. is reported in [22]. Among the first three CNNs, the major difference is the form of input. Best w2v-Conv. uses the pre-trained word2vec embeddings like our model. The input of best Lk-Conv. is in the form of lookup table in [26]. Best Char-Conv. uses up to 29 convolutional layers and creates a vectorial representation of each character.

In Table 3, our model outperforms all the related neural networks. The Best Char-Conv. uses 29 convolutional layers while our model holds only four layers. It seems that it is more difficult for models to learn from character-level, because internal structure of words by which words are formed are obscure.

However, the best linear model (ngrams TFIDF) has achieved a test error rate of 7.64% reported in [16]. Even our model has outperformed the other neural networks, the linear model is still superior to ours. A TF-IDF variant contains more priori knowledge and pay more attention to some of key words. It is efficiency for category classification, especially the dataset is relatively small. We can learn from this traditional model and further improve our model.

5. Conclusion

In this paper, we propose a novel TCNN-SM architecture that is capable of capturing both spatial and temporal memory of text. It comprises feature-representation and memory functional columns. Basically, HCLs in feature-representation functional column carry out hierarchical feature extraction while CSMLs in memory functional column accomplish selective memory of different granularity for historical information. To highlight key words of text due to sparsity strengthening, we also utilize L1-norm as regularization in cost function. Furthermore, we find out that repeated paddings of input samples are well suited to variable-length of sentences in text classification task. The experimental results achieved on AG's News public dataset demonstrate that our new TCNN-SM achieves 7.99% test error, which has the best performance among other competitive baseline models and is very close to state of the art results yielded using classical n-grams algorithm.

Acknowledgments

The authors would like to be grateful to the anonymous reviewers for their valuable comments that considerably contributed to improve this paper.

References

[1] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, vol. 14, 2014, pp. 1532–43.

[2] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, 2013.

[3] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*. Springer, 1998, pp. 137–142.

[4] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 2012, pp. 90–94.

[5] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[9] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[10] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 78.

[11] Y. S. Chan and D. Roth, "Exploiting syntactico-semantic structures for relation extraction," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 551–560.

[12] M. Post and S. Bergsma, "Explicit and implicit syntactic features for text classification." in *ACL (2)*, 2013, pp. 866–872.

[13] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents." in *ICML*, vol. 14, 2014, pp. 1188–1196.

[14] H. Schwenk, D. Dchelotte, and J.-L. Gauvain, "Continuous space language models for statistical machine translation," in *Proceedings of the COLING/ACL on Main conference poster sessions*. Association for Computational Linguistics, 2006, pp. 723–730.

[15] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. M. Schwartz, and J. Makhoul, "Fast and robust neural network joint models for statistical machine translation." in *ACL (1)*. Citeseer, 2014, pp. 1370–1380.

[16] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

[17] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[18] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling." in *Interspeech*, 2012, pp. 194–197.

[19] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[20] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso, "Finding function in form: Compositional character models for open vocabulary word representation," *arXiv preprint arXiv:1508.02096*, 2015.

[21] M. Ballesteros, C. Dyer, and N. A. Smith, "Improved transition-based parsing by modeling characters instead of words with lstms," *arXiv preprint arXiv:1508.00657*, 2015.

[22] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for natural language processing," *arXiv preprint arXiv:1606.01781*, 2016.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[26] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[27] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[28] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[29] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A c-lstm neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015.

[30] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," *arXiv preprint arXiv:1609.01704*, 2016.

[31] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *arXiv preprint arXiv:1412.1058*, 2014.

[32] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," in *Advances in Neural Information Processing Systems*, 2014, pp. 2042–2050.

[33] V. Landeiro and A. Culotta, "Robust text classification in the presence of confounding bias," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.