MODE-LSTM: A Parameter-efficient Recurrent Network with Multi-Scale for Sentence Classification

Qianli Ma, Zhenxi Lin, Jiangyue Yan, Zipeng Chen, Liuhong Yu
School of Computer Science and Engineering,
South China University of Technology, Guangzhou, China
qianlima@scut.edu.cn
zhenxi_lin@foxmail.com

Abstract

The central problem of sentence classification is to extract multi-scale n-gram features for understanding the semantic meaning of sentences. Most existing models tackle this problem by stacking CNN and RNN models, which easily leads to feature redundancy and overfitting because of relatively limited datasets. In this paper, we propose a simple yet effective model called **M**ulti-scale **O**rthogonal in**D**epend**E**nt LSTM (MODE-LSTM), which not only has effective parameters and good generalization ability, but also considers multi-scale n-gram features. We disentangle the hidden state of the LSTM into several independently updated small hidden states and apply an orthogonal constraint on their recurrent matrices. We then equip this structure with sliding windows of different sizes for extracting multi-scale n-gram features. Extensive experiments demonstrate that our model achieves better or competitive performance against state-of-the-art baselines on eight benchmark datasets. We also combine our model with BERT to further boost the generalization performance.

1 Introduction

Sentence classification (SC) is a fundamental and traditional task in natural language processing (NLP), which is widely used in many subareas, such as sentiment analysis (Wang et al., 2016a, 2018) and question classification (Shi et al., 2016). The central problem of SC is to understand the semantic meaning of a sentence by some key-phrases located at different positions (Wang et al., 2015).

CNNs excel at extracting n-gram features of sentences through a convolution operation followed by non-linear and pooling layers and have achieved impressive results in sentence classification (Kalchbrenner et al., 2014; Kim, 2014). However, the convolution operation itself is linear, which may not be sufficient to model the non-consecutive dependency of the phrase (Lei et al., 2015) and may lose the sequential information (Madasu and Anvesh Rao, 2019). As shown in Figure 1, the weighted sum of the phrase "***not*** *almost as* ***bad***" does not capture the non-consecutive dependency of "*not bad*" very well and ignores the sequential information.

I think this movie is *a bit boring* but ***not*** *almost as* ***bad***.

Figure 1: An example with variable-size phrases.

On the other hand, LSTMs (Hochreiter and Schmidhuber, 1997) are suitable for encoding structure-dependent semantics by storing previous word representations and preserving sequential information. However, LSTMs are still biased toward later words and ignoring the earlier words (Yin et al., 2017), so some current methods (Lai et al., 2015; Wang et al., 2016b; Zhang et al., 2016a; Song et al., 2018) combine the CNN and LSTM by stacking. However, merely stacking multiple layers can easily lead to feature redundancy and overfitting, because only relatively small training sets are available for SC tasks (Yin and Schütze, 2015; Guo et al., 2019). Hence, some researchers (Zhao et al., 2018a; Zhou et al., 2018; Madasu and Anvesh Rao, 2019) additionally attach an over-parameterized attention mechanism to enhance salient features and remove redundancy, but overfitting still occurs due to the increase in parameters for limited datasets.

A flexible combination method is to model nonlinear mapping and non-consecutive dependency by replacing the convolution operation with a tensor product (Lei et al., 2015) or RNN unit (Shi et al., 2016; Wang, 2018). However, these methods only consider fixed-size n-gram features. This has apparent drawbacks in that there may be variable-size phrases (n-grams) in a sentence, as shown in Figure 1, we need to extract variable-size n-gram features to form a better sentence representation.

The above observation motivates us to explore a better structure for sentence classification, balancing the capability and complexity. In this paper, we propose a lightweight model called **M**ulti-scale **O**rthogonal in**D**epend**E**nt LSTM (MODE-LSTM), which has minimal effective parameters, good generalization performance, and considers n-gram features of different scales. First, inspired by (Kuchaiev and Ginsburg, 2017), we disentangle the hidden state of LSTM into several independently updated small hidden states, which reduces the number of parameters. Furthermore, an orthogonal constraint is applied to the recurrent transition matrices of the small hidden states to improve the diversity of features. We call this structure **O**rthogonal In**D**epend**E**nt LSTM (ODE-LSTM). Then we use ODE-LSTM within a local window for extracting n-gram features instead of simply using a weighted sum as in convolution. Specifically, we introduce a Triple-S (Slide-Split-Stack) operation that splits a sentence into multiple sub-sentences by a sliding window and stacks them together. These sub-sentences are regarded as a mini-batch, which can be processed in parallel by a shared ODE-LSTM. We take the last hidden state of ODE-LSTM as the n-gram features for each sub-sentence. Furthermore, in order to capture the variable-size phrases in sentences, we use different scale windows with different initialized ODE-LSTMs to extract features of multiple scale phrases. We refer to this structure as a multi-scale ODE-LSTM (MODE-LSTM).

MODE-LSTM can extract multi-scale n-gram features like a CNN, while retaining the non-linear ability and long-term dependency of LSTMs, so it has stronger modeling ability but with fewer parameters than other methods. MODE-LSTM is analogous to a 1D CNN using multiple filters with different window sizes, but it uses recurrent transitions instead of the convolution operation. We conduct experiments on eight sentence classification datasets. The experimental results show that our proposed model achieves comparable or better results on these datasets with fewer parameters than other models. In addition, we further improve our model's generalization performance by integrating the BERT representation of the sentence.

2 Related Work

CNN-based models Kalchbrenner et al. (2014) propose a deep CNN model with a dynamic k-max pooling operation for the semantic modeling of sentences. However, a simple one-layer CNN with fine-tuned word embeddings also achieves remarkable results (Kim, 2014). Some researchers also use multiple word embeddings as inputs to further improve performance (Yin and Schütze, 2015; Zhang et al., 2016b). Xiao et al. (2018) propose a transformable CNN that can adaptively adjust the scope of the convolution filters. Although the above CNN-based methods perform excellently in extracting local semantic features, linear convolution operation limits the ability of modeling non-consecutive dependency and sequential information.

RNN-based models RNNs are suitable for processing text sequences and modeling long-term dependencies, so it is also used for sentence modeling. Recently, some work incorporate residual connections (Wang and Tian, 2016) or dense connections (Ding et al., 2018) into recurrent structures to avoid vanishing gradients. Dangovski et al. (2019) introduce a rotational unit of memory into RNNs for recalling long-distance information. Zhang et al. (2018) propose an HS-LSTM that can automatically discover structured representation in a sentence via reinforcement learning. However, these RNN-based models still display the bias problem where later words are more dominant than earlier words (Yin et al., 2017).

Hybrid models A natural strategy is to combine the advantages of CNNs and RNNs by stacking. Lai et al. (2015) equip an RNN with max-pooling to tackle the bias problem of RNNs. Zhou et al. (2015) use 1D convolutions to extract phrase features followed by an LSTM to obtain the sentence representation, and some subsequent work (Wang et al., 2016a,b; Lee and Dernoncourt, 2016) are similar. Alternatively, Zhang et al. (2016a) first model long-term dependencies using an LSTM and then apply a CNN to extract task-specific features. However, these methods simply stack multiple layers, resulting in feature redundancy and overfitting because of limited datasets (Yin and Schütze, 2015; Guo et al., 2019). Some researchers have introduced attention mechanisms (Er et al., 2016; Lin et al., 2017; Zhao et al., 2018a; Zhou et al., 2018) to enhance salient features, but this leads to a large number of parameters that overfit for small-scale datasets. A more flexible way is to combine them by replacing the convolution operation with a tensor product (Lei et al., 2015) or RNN unit (Shi et al., 2016; Wang et al., 2018), which can capture the non-linear n-gram features directly. Nevertheless, these methods currently only consider fixed-scale n-gram features.

Other models Some work (Tai et al., 2015; Liu et al., 2017; Wang et al., 2019) has used tree-LSTMs based on parse trees for sentiment analysis, but the performance depends heavily on the quality of the parser, and the parsing process itself is time-consuming. Others (Gong et al., 2018; Zhao et al., 2018b; Zheng et al., 2019) have tried using capsule networks with dynamic routing for encoding text representations.

The most relevant work to our approach is the DRNN (Wang, 2018), which also uses RNNs locally to learn semantic features. The differences between their approach and ours are: (1) The DRNN uses GRUs as the recurrent unit while we use the ODE-LSTM, which has better generalization performance. (2) We introduce the Triple-S operation to execute all sub-sentences in parallel instead of in sequence, which is faster than the DRNN. (3) We consider multi-scale n-gram features in sentences, while DRNN only considers a fixed scale.

3 Proposed Method

In the following, we start with our most straightforward model, which is a parameter-efficient structure of LSTMs to avoid over-fitting and achieve better generalization performance. This structure is then equipped with local sliding windows to learn key phrase features of the sentence, which is the central problem for understanding sentence semantics (Wang et al., 2015). Finally, we further easily extended our method to capture multi-scale features of the sentence via using different sized windows in parallel.

3.1 Orthogonal InDependEnt LSTM (ODE-LSTM)

Given a sentence of T input vectors $\{\mathbf{x}_1, \cdots, \mathbf{x}_T\}$, where $\mathbf{x}_t \in \mathbb{R}^{d_0}$, and d_0 is the dimension of input embeddings. The hidden state $\mathbf{h}_t \in \mathbb{R}^d$ of LSTM cell can be expressed as follows:

$$\begin{pmatrix}\mathbf{f}_t\\ \mathbf{i}_t\\ \mathbf{o}_t\\ \mathbf{g}_t\end{pmatrix} = \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}, \tag{1}$$

$$\mathbf{c}_t = \sigma(\mathbf{f}_t) \odot \mathbf{c}_{t-1} + \sigma(\mathbf{i}_t) \odot tanh(\mathbf{g}_t), \tag{2}$$

$$\mathbf{h}_t = \sigma(\mathbf{o}_t) \odot tanh(\mathbf{c}_t), \tag{3}$$

where $\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t$ are the forget, input and output gates respectively, and $\mathbf{g}_t$ is the candidate cell state. $\mathbf{W} \in \mathbb{R}^{4d\times d}$, $\mathbf{U} \in \mathbb{R}^{4d\times d_0}$, and $\mathbf{b} \in \mathbb{R}^{4d}$ are the learnable parameters. σ denotes the sigmoid function, and $\odot$ denotes element-wise multiplication. The number of distinct parameters in the LSTM are $4d(d_0 + d + 1)$ which are $\mathcal{O}(d^2)$. This easily leads to over-fitting for the sentence classification tasks where there are relatively limited data.

To reduce the number of parameters, inspired by (Kuchaiev and Ginsburg, 2017), we disentangle the hidden state $\mathbf{h}_t$ of the LSTM into K independently updated small hidden states. Specifically, the hidden state at time step t is composed by K small hidden states as $\widetilde{\mathbf{h}}_t = [\widetilde{\mathbf{h}}_t^1, \cdots, \widetilde{\mathbf{h}}_t^K]^\top$, where $\widetilde{\mathbf{h}}_t \in \mathbb{R}^{K\times p}, p = d/K$. The corresponding recurrent matrix is defined as $\widetilde{\mathbf{W}} = [\widetilde{\mathbf{W}}^1, \cdots, \widetilde{\mathbf{W}}^K]$, where $\widetilde{\mathbf{W}} \in \mathbb{R}^{K\times 4p\times p}$ and $\widetilde{\mathbf{W}}^k \in \mathbb{R}^{4p\times p}$. Each small hidden state $\widetilde{\mathbf{h}}_t^k$ is independently updated by an individual recurrent matrix $\widetilde{\mathbf{W}}^k$ and then merged via concatenation to constitute the hidden state $\widetilde{\mathbf{h}}_t$ at time step t. The update equation of hidden state $\widetilde{\mathbf{h}}_t$ is defined as :

$$\begin{pmatrix}\widetilde{\mathbf{f}}_t\\ \widetilde{\mathbf{i}}_t\\ \widetilde{\mathbf{o}}_t\\ \widetilde{\mathbf{g}}_t\end{pmatrix} = \widetilde{\mathbf{W}} \circledast \widetilde{\mathbf{h}}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}, \tag{4}$$

$$\widetilde{\mathbf{c}}_t = \sigma(\widetilde{\mathbf{f}}_t) \odot \widetilde{\mathbf{c}}_{t-1} + \sigma(\widetilde{\mathbf{i}}_t) \odot tanh(\widetilde{\mathbf{g}}_t), \tag{5}$$

$$\widetilde{\mathbf{h}}_t = \sigma(\widetilde{\mathbf{o}}_t) \odot tanh(\widetilde{\mathbf{c}}_t), \tag{6}$$

where $\circledast$ is the tensor-dot operation which denotes the product of two tensors along the K-axis, e.g., $\widetilde{\mathbf{W}} \circledast \widetilde{\mathbf{h}}_{t-1} = [\widetilde{\mathbf{W}}^1\widetilde{\mathbf{h}}_{t-1}^1, \cdots, \widetilde{\mathbf{W}}^K\widetilde{\mathbf{h}}_{t-1}^K]^\top$ where $\widetilde{\mathbf{W}}^k\widetilde{\mathbf{h}}_{t-1}^k \in \mathbb{R}^{4p}$. Note that standard LSTM is a special case of ODE-LSTM when $K = 1$.

The updated hidden state $\widetilde{\mathbf{h}}_t$ may be redundant if all hidden states provide similar features. To avoid this, we introduce a penalization loss that orthogonally constrains $\widetilde{\mathbf{W}}$ to explicitly encourage diversity among hidden states, inspired by (Lin et al., 2017).

$$\mathcal{L}_P = \sum_{i=1}^{K}\sum_{j=1}^{K} \|\widetilde{\mathbf{W}}\widetilde{\mathbf{W}}^\top - \mathbf{I}\|_2^2. \tag{7}$$

With the same size d of hidden states as the LSTM, ODE-LSTM reduces the number of parameters by $4d(d-p)$. The smaller p is, the more parameter reduction. Because of the disentanglement of hidden states, each small hidden state can

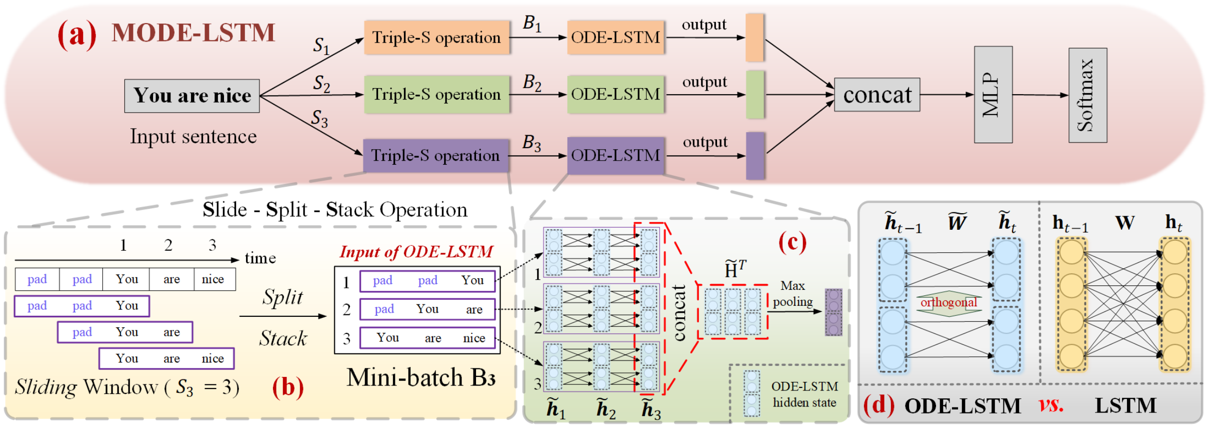

Figure 2: **(a)** The diagram of MODE-LSTM with three different scale windows $[S_1, S_2, S_3]$. The input sentence is converted into three mini-batches $[B_1, B_2, B_3]$ by the Triple-S operation. These mini-batches are respectively fed into different initialized ODE-LSTMs to extract n-gram features for each scale. **(b)** The detail of Triple-S operation. **(c)** The process of performing mini-batch B_3 for an ODE-LSTM. **(d)** The comparison of ODE-LSTM and LSTM. Here, ODE-LSTM disentangles the hidden state into two small hidden states. An orthogonal constraint is apply on the recurrent matrix $\widetilde{\mathbf{W}}$ to improve the diversity of features.

focus on a different aspect of semantics, with better generalization performance. Figure 2(d) shows the comparison between ODE-LSTM and LSTM.

3.2 Equipping ODE-LSTM with Sliding Window

The core of SC task is to understand the semantics of the sentence, which are determined by key words and variable-size phrases. Although a CNN can capture n-grams, the linear convolution operation is insufficient to model sequential information and non-consecutive dependency of sentences. Our ODE-LSTM can maintain word order, and control information preserving or forgetting through gates for modeling non-consecutive dependency. Taking the phrase “***not** almost as **bad***” as an example, the gates can selectively retain the representation of “not” and “bad” while decaying the representation of “almost” and “as”, allowing it to perceive the relation “not bad”.

Hence, we equip ODE-LSTM with a sliding window for extracting n-gram features, which means that the recurrent transition of ODE-LSTM is only performed in a local window with size S sliding along the sentence, as illustrated in the left of Figure 2(b). S is a hyperparameter. For each target position t, ODE-LSTM will sequentially process S consecutive words in the range $(t - S + 1, t)$ of the sentence and generate relevant hidden states. The last hidden state $\widetilde{\mathbf{h}}_t$ output by ODE-LSTM is used as the n-gram feature of the target position:

$$\widetilde{\mathbf{h}}_t = \text{ODE-LSTM}(\mathbf{x}_{t-S+1}, \cdots, \mathbf{x}_t). \quad (8)$$

For convenience, we reshape $\widetilde{\mathbf{h}}_t \in \mathbb{R}^{K\times p}$ to a vector of d dimension. Meanwhile, we pad $(S - 1)$ zeros before the start position of the sentence to maintain consistent window size at all positions. This kind of local way is analogous to DRNN (Wang, 2018), but they process all windows sequentially, equivalent to processing a sentence of length $S \times T$ in order, which is highly time-consuming. However, we observe that all windows are independent of each other, so they can be processed in parallel by a GPU, which greatly improves the computational efficiency.

Correspondingly, we introduce a Triple-S (Slide-Split-Stack) operation to compose all the windows, as shown in Figure 2(b). First we split a sentence into multiple sub-sentences by a sliding window with size S, and then stack them together to form a mini-batch $B \in \mathbb{R}^{T\times S\times d_0}$. The mini-batch B is fed into an ODE-LSTM, obtaining the n-gram feature matrix $\widetilde{\mathbf{H}} \in \mathbb{R}^{T\times d}$, as shown in Figure 2(c):

$$\widetilde{\mathbf{H}} = [\widetilde{\mathbf{h}}_1, \cdots, \widetilde{\mathbf{h}}_T]^\top, \quad (9)$$

where $\widetilde{\mathbf{h}}_t$ is calculated by equation (8), corresponding to the n-gram feature at t-th position. In this way, the recurrent steps of ODE-LSTM are determined by S rather than the sentence length T, so the time complexity is much lower than DRNN.

3.3 Multi-Scale ODE-LSTM (MODE-LSTM)

Sentence phrases have multiple granularities, i.e., n-gram features at different scales. Nevertheless, what we consider above uses a fixed window size S. A natural idea is to use multiple scale windows in parallel with ODE-LSTM to extract n-gram features of different scales. The Multi-scale ODE-LSTM (MODE-LSTM) model is illustrated in Figure 2(a). According to the Triple-S operation described in Section 3.2, the sentence is converted into multiple mini-batches $[B_1, \cdots, B_M]$ based on different scale window sizes $[S_1, \cdots, S_M]$, where M is the number of scales. Then, the mini-batches are fed into different ODE-LSTMs to obtain the n-gram feature matrix :

$$\widetilde{\mathbf{H}}_m = [\widetilde{\mathbf{h}}_{m,1}, \cdots, \widetilde{\mathbf{h}}_{m,T}]^\top, \quad (10)$$

$$\widetilde{\mathbf{h}}_{m,t} = \text{ODE-LSTM}_m(\mathbf{x}_{t-S_m+1}, \cdots, \mathbf{x}_t), \quad (11)$$

where $\widetilde{\mathbf{H}}_m \in \mathbb{R}^{T\times d}$ denotes the n-gram feature matrix of scale S_m, $m = 1, \cdots, M$. $\widetilde{\mathbf{h}}_{m,t} \in \mathbb{R}^d$ denotes the t-th n-gram feature of scale S_m. Subsequently, we apply max pooling (MP) along the T-axis over each n-gram feature matrix to extract salient features for each scale, and then concatenate them to constitute the multi-scale feature representation $\mathbf{F} \in \mathbb{R}^{M\times d}$:

$$\mathbf{F} = [MP(\widetilde{\mathbf{H}}_1), \cdots, MP(\widetilde{\mathbf{H}}_M)]^\top. \quad (12)$$

Afterward, the feature representation $\mathbf{F}$ is reshaped to a vector and fed into an MLP layer with rectified linear unit (ReLU) activation function and a softmax layer for the final classification.

3.4 Objective Function

The overall objective function includes a cross-entropy category loss and the penalization loss for all ODE-LSTMs. So it's defined as:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}_{cross}(y_n, \hat{y}_n) + \lambda \sum_{m=1}^{M} \mathcal{L}_{P_m}, \quad (13)$$

where N is the number of samples, y_n and $\hat{y}_n$ are the ground-truth label and softmax output respectively, $\mathcal{L}_{P_m}$ is penalization term for the m-th ODE-LSTM, and λ is a hyperparameter for balancing the strength of the orthogonality constraint. We minimize the above function by BPTT.

4 Experiments

4.1 Experimental Setup

Datasets To evaluate the effectiveness of our model, we conduct experiments on eight widely-studied datasets (Kim, 2014; Liu et al., 2017) for sentence classification. Statistics of these datasets are listed in Table 1. These datasets come from different topics, such as sentiment analysis, movie reviews (MR, SST2, SST5), customer reviews (CR), and idioms (IE); question type (TREC) classification; opinion (MPQA) or subjectivity (SUBJ) classification.

Dataset	c	l	ml	$Train$	Dev	$Test$
MR	2	19	53	10662	–	CV
CR	2	19	100	3775	–	CV
SUBJ	2	23	108	10000	–	CV
MPQA	2	3	34	10606	–	CV
TREC	6	10	33	5452	–	500
IE	3	16	75	2221	–	300
SST2	2	19	53	6920	872	1821
SST5	5	18	53	8544	1101	2210

Table 1: Statistics of eight datasets for sentence classification. c: Number of target classes. l: Average sentence length. ml: Maximum sentence length. $Train/Dev/Test$: Size of train/development/test set (CV means 10-fold cross validation is used).

Implementation Details We initialize the word embeddings with 300D pre-trained GloVe vectors (Pennington et al., 2014) and incorporate 50D character embeddings constructed by a convolution layer with a max pooling layer to avoid the Out-Of-Vocabulary (OOV) problem (Zhang et al., 2019). These two embeddings are then concatenated as the input embeddings and fine-tuned along with model parameters during training. We use three scale windows, [5, 10, 15], to initialize various ODE-LSTMs. K is set to 2 and the size p of each small hidden state is set to 50 for each scale. This configuration results in a 300D multi-scale feature representation for classification. For regularization, we employ dropout with a rate of 0.2 and 0.5 for input embeddings and the single MLP hidden layer, respectively. L2 regularization, with a factor of 0.001, is applied to the weights of the softmax layer. The hyperparameter λ is set to 0.01, and the batch size is set to 50. Our model is optimized by Adam with a learning rate of 1e-3. Similar to (Kim, 2014), these hyperparameters are

Type	Model	#Params	MR	CR	SUBJ	TREC	MPQA	SST2	SST5	IE	Average
Others	Tree-LSTM†	–	80.7	83.2	91.3	91.8	–	85.7	50.1	–	–
	DC-treeLSTM†	–	81.7	–	93.7	93.8	–	87.8	–	60.2	–
	capsuleB†	–	82.3	85.1	93.8	92.8	–	86.8	–	–	–
	HAC†	–	**83.3**	86.4	**95.1**	95.0	89.8	88.2	49.1	–	–
CNN/RNN -based	HM-LSTM	–	82.1	–	93.7	–	–	–	49.8	–	–
	LSTM*†	827K	81.2	84.6	93.7	94.2	89.7	86.6	46.6	62.3	79.86
	TextCNN*†	466K	81.7	85.2	94.3	93.6	89.9	87.5	47.8	62.0	80.25
Hybrid	DARLM	7.9M	83.2	–	94.1	96.0	–	–	48.8	–	–
	DLSTM*†	827K	82.4	86.5	94.2	94.2	90.4	87.8	49.2	62.3	80.88
	C-LSTM*†	1.1M	80.7	84.0	94.0	94.6	89.5	87.8	48.7	63.0	80.29
	Self-Attentive*†	42M	82.0	85.9	94.4	93.8	90.0	86.8	49.7	61.3	80.49
Ours	ODE-LSTM†	527K	82.2	85.1	94.2	93.4	90.0	88.1	48.8	62.7	80.56
	MODE-LSTM	527K	**83.3**	**86.8**	94.8	**96.1**	**90.6**	**89.2**	**51.2**	**63.3**	**81.91**
Combine with Pre-trained Sentence Representations											
pre-training	InferSent‡	–	81.1	86.3	92.4	88.2	90.2	84.6	–	–	–
	BOW + ELMo‡	–	79.7	85.1	94.3	93.4	89.6	86.3	48.7	–	–
	USE‡	–	81.2	87.5	93.6	**98.1**	87.3	86.7	–	–	–
	HAC + ELMo‡	–	85.0	88.9	95.9	96.8	91.2	89.4	49.7	–	–
	BERT$_{base}$‡	110M	86.8	90.3	96.8	96.8	90.8	93.5	53.3	69.0	84.66
Ours	MODE-LSTM + BERT$_{base}$	111M	**87.3**	**91.5**	**97.0**	97.2	**91.3**	**93.8**	**54.6**	**73.3**	**85.75**

Table 2: Experimental accuracy comparison of our model and baselines on eight sentence classification benchmarks. "#Params" represents the approximate number of parameters except input embedddings for models. The results of models marked with * are obtained by our implementation. The input embeddings used in these baselines are the same as our models. Other parameter settings of models are consistent with their references. The remaining results are collected from the corresponding papers. The model marked with † (‡) means MODE-LSTM (with BERT$_{base}$) is significantly superior to compared model by paired t-test (Wilcoxon, 1945) at $p < 0.05$ level.

determined by a grid search on the MR dataset and are applied to the other datasets[§].

.

Baseline Methods We compare MODE-LSTM with three types of strong baselines: 1) CNN/RNN-based model: TextCNN (Kim, 2014), LSTM (Tai et al., 2015) and HM-LSTM (Zhang et al., 2018). 2) Hybrid models: C-LSTM (Zhou et al., 2015) which directly stacks CNN and LSTM, while DARLM (Zhou et al., 2018) and Self-attentive (Lin et al., 2017) additionally includes an attention mechanism for distilling important information. Relatively, DRNN (Wang, 2018) incorporates position-invariance into RNN. For a fair comparison, we use the LSTM as the basic unit of DRNN, called DLSTM. 3) Other models: tree-LSTM (Tai et al., 2015) and DC-treeLSTM (Liu et al., 2017) based on parse trees; capsuleB (Zhao et al., 2018b) and HAC (Zheng et al., 2019) based on capsule networks. In addition to the above models, we use ODE-LSTM as a baseline. We set K to 6 and the size p of small hidden states to 50 to make the number of parameters consistent with MODE-LSTM.

[§]The source code is publicly available at `https://github.com/qianlima-lab/MODE-LSTM`.

4.2 Experimental Results

Table 2 reports the performance of our approaches against other methods. With fewer parameters, MODE-LSTM significantly outperforms the compared models and is superior to DLSTM with an average accuracy gain over 1.0% because ours disentangles the RNN hidden states and considers multi-scale features in sentences. Meanwhile, our model achieves better or similar performance with recent state-of-the-art model HAC. HAC is a complex model that uses deep dilated convolutional layers and a capsule module at each layer. However, our model is simple yet effective, like the one-layer TextCNN. Specifically, although the parameters of TextCNN are less than ours, its parameters increase with the size of the filter window, whereas the parameters of our model are independent of the window size. ODE-LSTM also outperforms LSTM with an average accuracy gain 0.7%, which verifies

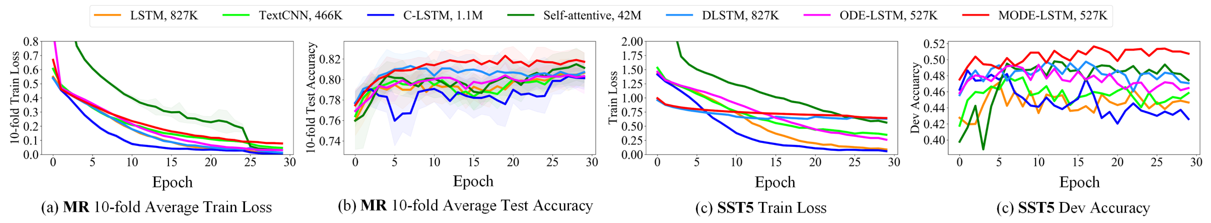

Figure 3: The convergence analysis on MR and SST5 datasets. (a)(b) are the average train loss and average test accuracy of 10-fold cross-validation on MR dataset, where the shaded area is the standard deviation. (c)(d) are the train loss and development accuracy on SST5 dataset.

the effectiveness of disentangling the hidden states.

To investigate how our model makes a difference with others, we visualize the convergence trends in Figure 3. We observe that the direct-stacked C-LSTM (dark blue line) converges quickly on the training set but has poor performance on development or testing sets. Although the Self-Attentive (dark green line) can alleviate feature redundancy by employing the attention mechanism, overfitting still occurs due to a large number of parameters. MODE-LSTM (red line) achieves better generalization performance on development or testing sets than other models.

4.3 Combining MODE-LSTM with BERT

Recently, the pre-trained language model BERT (Devlin et al., 2018) is more effective than conventional word embeddings when fine-tuned on downstream tasks. Compared with word embeddings, BERT can learn context-dependent sentence representations. Nevertheless, recent work (Yang et al., 2018, 2019; Xu et al., 2019) has indicated that the self-attention used in BERT disperses the attention distribution and thus overlooks the essential neighboring elements and phrasal patterns. MODE-LSTM can explicitly extract multi-scale local features, which is complementary to BERT representation. Hence, we try to combine MODE-LSTM with BERT to improve the generalization performance of our model further. Concretely, the sentence is fed into BERT$_{base}$ model, and the hidden representation of the last layer of BERT$_{base}$ is used as the input embeddings of MODE-LSTM rather than GloVe and character embeddings. BERT provides contextualized sentence-level representations, which help MODE-LSTM understand sentence semantics more accurately. The detailed diagram and the hyper-parameter settings of this configuration can be found in the appendix.

We compare MODE-LSTM equipped with BERT (MODE-LSTM + BERT) with some recent strong baselines that also combine with pre-trained sentence representations, including InferSent (Conneau et al., 2017), combining ELMo with bag-of-words (BOW + ELMo) (Perone et al., 2018) or HAC (HAC + ELMo) (Zheng et al., 2019), universal sentence encoder (USE) (Cer et al., 2018), and BERT. The results are shown in the bottom row of Table 2. Using the BERT representation, MODE-LSTM can further boost the generalization performance. Although BERT already provides strong performance on almost all datasets, it may tend to ignore the local phrasal information due to the self-attention mechanism. Therefore, the combination of MODE-LSTM and BERT can further improve the prediction power, which indicates that our model can better understand the semantic meaning. Notably, our model without BERT has surpassed some pre-trained models, such as InferSent and BOW + ELMo, and is comparable to USE, verifying its effectiveness and generalization.

Scales	Pena.	Char.	MR	SUBJ	SST5
5,10,15	✓	✓	**83.3**	**94.8**	**51.2**
5,5,5	✓	✓	83.1	93.9	50.7
10,10,10	✓	✓	83.0	94.3	49.2
15,15,15	✓	✓	82.9	94.3	50.0
5,10,15	×	✓	83.0	94.5	51.0
5,10,15	✓	×	82.7	94.6	51.0

Table 3: Ablation study on some datasets. “Pena.” denotes penalization loss. “Char.” denotes character embeddings.

4.4 Ablation Study

In this section, we investigate to study the independent effect of each component in our proposed model. We explore the influence of the window scales, the penalization loss, and the character embeddings. The results are reported in Table 3. Com-

Examples	G.T.	TextCNN	DLSTM	Ours
1. While it 's***genuinely cool*** to hear characters talk about early ***rap records sugar*** hill gang etc ***the constant referencing*** of hip-hop arcana can ***alienate*** even the savviest audiences.	N	P	N	N
2. ***I admire it*** and yet ***cannot*** recommend it because it ***overstays its natural*** running time.	N	P	P	N

Table 4: Case study of our model compared to TextCNN and DLSTM. "G.T." is ground-truth. "N" and "P" represent Negative and Positive. Words with dotted lines, underlines, and wavy lines correspond to the important positions extracted by TextCNN, DLSTM, and MODE-LSTM respectively.

pared to using multiple windows with different scales (Row 1), using a single scale (Row 2-4) significantly reduces the accuracy. This demonstrates the necessity of integrating multi-scale windows to learn variable-size phrases in sentences. We can see that eliminating penalization loss (Row 5) or character embeddings (Row 6) also hurts the performance, which verifies that these components are beneficial to our model.

4.5 Case study

To explore why our model outperforms TextCNN and DLSTM, we display several most contributing positions in max-pooling by visualization techniques introduced in (Li et al., 2015). Table 4 shows two examples on the MR dataset. In the first example, CNN wrongly captures the key phrase *genuinely cool*. Thus the sentence is misclassified as Positive, while DLSTM and our model capture the non-consecutive dependency according to the key word *while*. Hence they attend to the second half of the sentence for correct classification. In the second sample, all the three models extract the key phrase *I admire it*, which suggests classifying the sentence as positive. Therefore, both TextCNN and DLSTM fail in this case. However, our model also extracts key phrases *cannot* and *overstays its natural* by learning multi-scale features so that it can obtain the correct answer.

4.6 Model Analysis

Impact of the value K To study the influence of the value K (the number of small hidden states), we conduct experiments on MR and SUBJ datasets. We fix the multi-scale feature representation output by MODE-LSTM to 300D and tune the value K. The larger K is, the smaller the size of the small hidden states. The results are reported in Figure 4(a). We found that $K = 2$ is a good trade-off between model accuracy and parameters. When K is too large, the hidden size is too small to provide enough features, which causes the overall performance to decrease.

Impact of the window size We then explore the effect of window size when using only one scale window. We found that the optimal window size may be different for different datasets, as shown in Figure 4(b). The optimal window size for MR is 5, while for SUBJ, it is 20. We speculate that the reason is that the length of SUBJ sentences are longer than MR, and so long-term dependencies may be more prominent.

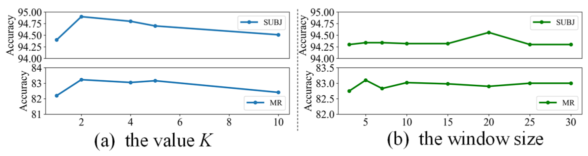

Figure 4: Impact of different K and window sizes.

Impact of training set size To further verify our model's generalization, we investigate the influence of different training set sizes. The results on MR are shown in Figure 5(a). MODE-LSTM outperforms others with an accuracy gain over 8% when only having 100 training samples. As the size continues to increase, the gain gradually decreases but our model is still superior to the others.

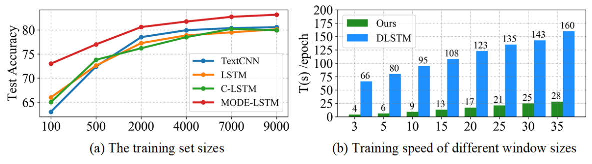

Figure 5: Effect on training set size and training time.

Training time comparison We assess the training time of our model and DLSTM on an NVIDIA GTX 1080ti GPU in Figure 5(b), testing on MR. In the case of using a single scale window, the training time for each model's epoch increases with the window size due to the recurrent structure. However, our model's training time marginally increases thanks to the ability to run in parallel by the Triple-S operation, which is $5 \sim 10\times$ faster than DLSTM performs in sequence. Since multiple window scales are independent and parallel, the training time for our multi-scale version mainly depends on the maximum window size. For example, with the same number of parameters and a maximum window size of 15, the training time for a multi-scale version is similar to that of the single-scale version on MR (15 vs. 13, T(s)/epoch).

5 Conclusion

This study presents a novel parameter-efficient model called MODE-LSTM that can capture multi-scale n-gram features in sentences. Instead of the tradition of exploiting complicated operations by stacking CNNs and RNNs, or attaching over-parameterized attention mechanisms, our work provides a lightweight method for improving the ability of neural models for sentence classification. Through disentangling the hidden states of the LSTM and equipping the structure with multiple sliding windows of different scales, MODE-LSTM outperforms popular CNN/RNN-based methods and hybrid methods on various benchmark datasets. In future work, we plan to validate its effectiveness for aspect-level sentiment classification.

Acknowledgments

We thank the anonymous reviewers for their helpful feedbacks. The work described in this paper was partially funded by the National Natural Science Foundation of China (Grant Nos. 61502174, 61872148), the Natural Science Foundation of Guangdong Province (Grant Nos. 2017A030313355, 2019A1515010768), the Guangzhou Science and Technology Planning Project (Grant Nos. 201704030051, 201902010020), and the Key RD Program of Guangdong Province (No. 2018B010107002).

References

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Rumen Dangovski, Li Jing, Preslav Nakov, Mićo Tatalović, and Marin Soljačić. 2019. Rotational unit of memory: A novel representation unit for rnns with scalable applications. *Transactions of the Association for Computational Linguistics*, 7:121–138.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zixiang Ding, Rui Xia, Jianfei Yu, Xiang Li, and Jian Yang. 2018. Densely connected bidirectional lstm with applications to sentence classification. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 278–287. Springer.

Meng Joo Er, Yong Zhang, Ning Wang, and Mahardhika Pratama. 2016. Attention pooling-based convolutional neural network for sentence modelling. *Information Sciences*, 373:388–403.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. *arXiv preprint arXiv:1806.01501*.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

HSepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Oleksii Kuchaiev and Boris Ginsburg. 2017. Factorization tricks for lstm networks. *arXiv preprint arXiv:1703.10722*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. *arXiv preprint arXiv:1508.04112*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Dynamic compositional neural networks over tree structure. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4054–4060.

Avinash Madasu and Vijjini Anvesh Rao. 2019. Sequential learning of convolutional features for effective text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5662–5671.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Yangyang Shi, Kaisheng Yao, Le Tian, and Daxin Jiang. 2016. Deep lstm based feature mapping for query classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1501–1511.

Xingyi Song, Johann Petrak, and Angus Roberts. 2018. A deep neural network sentence level classification method with context information. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 900–904.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Baoxin Wang. 2018. Disconnected recurrent neural networks for text categorization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2311–2320.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016a. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 225–230.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2019. Investigating dynamic routing in tree-structured lstm for sentiment analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3423–3428.

Peng Wang, Jiaming Xu, Bo Xu, Chenglin Liu, Heng Zhang, Fangyuan Wang, and Hongwei Hao. 2015. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 352–357.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 957–967.

Xingyou Wang, Weijie Jiang, and Zhiyong Luo. 2016b. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*, pages 2428–2437.

Yiren Wang and Fei Tian. 2016. Recurrent residual learning for sequence classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 938–943.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Transformable convolutional neural network for text classification. In *IJCAI*, pages 4496–4502.

Mingzhou Xu, Derek F. Wong, Baosong Yang, Yue Zhang, and Lidia S. Chao. 2019. Leveraging local and global patterns for self-attention networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3069–3075.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4449–4458.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019. Convolutional self-attention networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4040–4045.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Wenpeng Yin and Hinrich Schütze. 2015. Multichannel variable-size convolution for sentence classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 204–214.

Kun Zhang, Guangyi Lv, Linyuan Wang, Le Wu, Enhong Chen, Fangzhao Wu, and Xing Xie. 2019. Drr-net: Dynamic re-read network for sentence semantic matching. In *Thirty-three AAAI conference on artificial intelligence*.

Rui Zhang, Honglak Lee, and Dragomir R. Radev. 2016a. Dependency sensitive convolutional neural networks for modeling sentences and documents. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1512–1521.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ye Zhang, Stephen Roller, and Byron C. Wallace. 2016b. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1522–1527.

Jianyu Zhao, Zhiqiang Zhan, Qichuan Yang, Yang Zhang, Changjian Hu, Zhensheng Li, Liuxin Zhang, and Zhiqiang He. 2018a. Adaptive learning of local semantic and global structure representations for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2033–2043.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018b. Investigating capsule networks with dynamic routing for text classification. *arXiv preprint arXiv:1804.00538*.

Wanshan Zheng, Zibin Zheng, Hai Wan, and Chuan Chen. 2019. Dynamically route hierarchical structure representation to attentive capsule for text classification. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5464–5470.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Qianrong Zhou, Xiaojie Wang, and Xuan Dong. 2018. Differentiated attentive representation learning for sentence classification. In *IJCAI*, pages 4630–4636.

A The detailed diagram of combining MODE-LSTM with BERT

We use BERT$_{base}$ model, where the number of transformer layers is 12 and the hidden size is 784. In detail, the sentence is fed into BERT$_{base}$ model, and the hidden representation of the last layer of BERT$_{base}$ is used as the input embeddings of MODE-LSTM rather than GloVe and character embeddings. Then the BERT representation is fed into MODE-LSTM for extracting multi-scale feature representation.

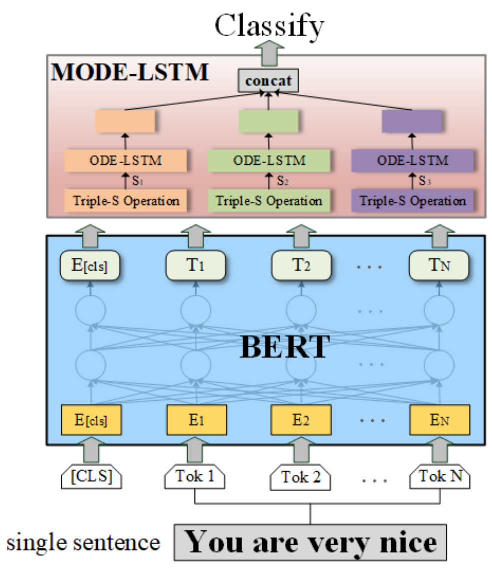

Figure 6: The diagram of combining MODE-LSTM with BERT

.

B The hyper-parameter settings of combining MODE-LSTM with BERT

The hyper-parameters of MODE-LSTM are the same as we mentioned in the experimental setup section of the main paper. That is we use three scale windows, $[5, 10, 15]$ with differently initialized ODE-LSTMs. The number of small hidden states K is set to 2 and the size p of each small hidden state is set to 50 for each scale. This configuration results in a 300D multi-scale feature representation for classification. Specifically, we tune learning rate in $\{1e-5, 3e-5, 5e-5\}$ and dropout rate in $\{0.8, 0.9\}$ for each dataset.