Major-Minor Long Short-Term Memory for Word-Level Language Model

Kai Shuang, Rui Li, Mengyu Gu, Jonathan Loo, and Sen Su

***Abstract*—Language model plays an important role in natural language processing (NLP) systems like machine translation, speech recognition, learning token embeddings, natural language generation and text classification. Recently, the multi-layer Long Short-Term Memory (LSTM) models have been demonstrated to achieve promising performance on word-level language modeling. For each LSTM layer, larger hidden size usually means more diverse semantic features, which enables the language model to perform better. However, we have observed that when a certain LSTM layer reaches a sufficiently large scale, the promotion of overall effect will slow down as its hidden size increases. In this paper, we analyze that an important factor leading to this phenomenon is the high correlation between the newly extended hidden states and original hidden states, which hinders diverse feature expression of the LSTM. As a result, when the scale is large enough, simply lengthening the LSTM hidden states will cost tremendous extra parameters but has little effect. We propose a simple yet effective improvement on each LSTM layer consisting of a large-scale Major LSTM and a small-scale Minor LSTM to break the high correlation between the two parts of hidden states, which we call Major-Minor LSTMs (MMLSTMs). In experiments, we demonstrate the language model with MMLSTMs surpasses the existing state-of-the-art model on Penn Treebank (PTB) and WikiText-2 (WT2) datasets, and outperforms the baseline by 3.3 points in perplexity on WikiText-103 dataset without increasing model parameter counts.**

***Index Terms*—language model, Long Short-Term Memory (LSTM), Natural Language Processing (NLP), shortcut connections.**

I. Introduction

LANGUAGE model (LM) is a foundational component of natural language processing (NLP) tasks, which estimates the probability distribution of a sequence of tokens $(w_0, ..., w_n)$ by modeling the probability of the next token (w_i) given preceding tokens $(w_0, ..., w_{i-1})$, i.e.

$$P(w_0, ..., w_n) = P(w_0) \prod_{i=1}^{n} P(w_i | w_0, ..., w_{i-1})$$

Language model plays an important role of systems for machine translation [1], speech recognition [2], learning token embeddings [3], [4], natural language generation [5], [6] and text classification [7].

This work was supported in part by the National Key Research and Development Program of China (No. 2017YFB1400603).

K. Shuang is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: shuangk@bupt.edu.cn).

R. Li, M. Gu and S. Su are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: lirui@bupt.edu.cn).

J. Loo is with the School of Computing and Engineering, University of West London, London W5 5RF, U.K. (e-mail: jonathan.loo@uwl.ac.uk).

Language models can operate at various granularities, with these tokens formed from either words (i.e. word-level language models [8], [9], sub-words (e.g. syllable-level language models [10]), or characters (i.e. character-level language models [11]). The main difference between the above three types of language models is the granularity of the prediction tokens (i.e., word/sub-word/character-level language models respectively predict the probability distribution of the next word/sub-word/character). In linguistics, a word is the smallest element that can be uttered in isolation with objective or practical meaning, so word-level language model, as a direct method to extract the dependency between each word and its contexts is more widely used in various downstream tasks [1], [3], [7], because they can extract semantic features directly from the original corpus. In this paper, we focus on word-level language modeling [1].

For word-level language model, the vanilla Long Short-Term Memory (LSTM) networks have been demonstrated to achieve state-of-the-art performance [6], [12]. As one of the most successful variants of Recurrent Neural Network (RNN), LSTM [13] can not only process word sequences of any length, but also effectively alleviate the issue of gradient vanishing. Since the probability distribution to be predicted is complex, existing LSTM language models often leverage LSTMs with high-dimensional hidden states to extract semantic features as comprehensive as possible. In [14]–[16], the LSTM language models with larger scale always exceeded the smaller ones under the same architecture, which means the scale of LSTMs ought to be large enough so that the language model has sufficient generalization capabilities. But in the other hand, we have observed in experiments that when a LSTM layer reaches a sufficiently large scale, the promotion of overall effect will slow down as its hidden sizes increase, even the changes within a certain interval will not cause any improvement in the final results. In other words, when the scale of LSTM is large enough, simply lengthening the hidden states will introduce tremendous extra parameters, but have little effect.

We analyze that one of the most important factors leading to this phenomenon is that LSTM as a recursive model, has a high correlation between its newly extended hidden states and original hidden states, which hinders the diverse semantic feature expression of it. Therefore, even with a larger hidden size, the vanilla LSTM is still hard to extract comprehensive semantic features from input. At the same time, a larger amount of useless parameters will be consumed.

[1]Unless otherwise specified, the language models mentioned later all refers to the word-level language models.

Although some existing regularization methods (e.g. recurrent dropout) can eliminate the negative impacts of the correlation between LSTM hidden state features to some extent, they cannot break the shackles of the recursive pattern of the single LSTM, and the effect is quite limited. In this paper, we propose a simple yet efficient improvement on each LSTM in the LSTM language model called Major-Minor LSTMs (MMLSTMs), which employs two LSTMs of different scales to generate the output features independently. Concretely, we carefully reduce the hidden size of each LSTM in original language model without impairing the overall performance significantly, and use the cut LSTM as the Major LSTM to extract main semantic features in each layer. At the same time, the saved parameters are used to build another small-scale Minor LSTM, whose hidden states can be seen as a set of auxiliary features of the Major LSTM hidden states. In this way, the structure of MMLSTMs can break the correlation between features in individual LSTM hidden state, so that the generated auxiliary features can better extract the diverse semantic features. In addition, MMLSTMs can effectively prevent the meaningless parameters introduced by simply lengthening a certain LSTM layer hidden state in original language model. We evaluate our MMLSTMs language model on three widely used language datasets Penn Treebank (PTB), WikiText-2 (WT2) and WikiText-103 (WT103). The experimental results show that the MMLSTM language model surpasses existing state-of-the-art language model on two small datasets PTB and WT2 and exceed the baseline 3.3 points in perplexity on the larger WT103.

The contributions and innovations of this work are summarized as follows:

1) We analyzed the operation of LSTM and revealed that there is a high correlation between different features in each hidden state, which can be a factor limiting the performance improvement of the large-scale LSTM language model;
2) We proposed a simple architecture called Major-Minor LSTMs (MMLSTMs), as the substitution of LSTM in vanilla LSTM language models, which is intuitively proven to effectively weaken the output feature correlations compared with the single LSTM by experiments;
3) We designed the Minor LSTM as a variant of the shortcut connections, and we demonstrate that the Minor LSTM as a new form of shortcut connections can significantly enhance the performance of the language model without sacrificing extra parameters;
4) Our experimental results showed that language model with MMLSTMs surpasses the existing state-of-the-art model on PTB and WT2 datasets, and outperforms the baseline by 3.3 points in perplexity on WT103 dataset without increasing model parameter counts.

II. Related Work

A. Word-level language model

In addition to the general word-level language model that directly using the word embeddings to represent each word, the recent works have proposed character-aware language model [17], [18] and subword-aware language model [3], [19] to incorporate the character and subword information into the word representations, thereby integrating the morphological properties of words. Like the general model, the subword-aware and the character-aware models are both word-level language models, but their model architectures add the part that using subword or character information to generate word representations. Therefore, the general model can be seen as the common part of different types of word-level language models, which is also the focus of the following discussion.

Word-level language modeling has gone through significant development from conventional N-gram language models [20], [21] to neural language models [22]–[24] in these decades. These classical N-gram models suffer from data sparsity, which makes it difficult to represent large contexts and thus, long-range dependencies. The LSTM-based language models effectively alleviate the problem of long-range dependency, so their performance greatly exceed other neural network language models. [14] applied dropout in the non-recurrent connections in LSTM language models with medium and large sizes, and achieved the best results with the large one at that time. [15] used the large LSTM language model with a dropout variation to surpass the result of [14]. [16] introduced a novel theoretical framework leads to tying together the input embedding and the output projection matrices, greatly reducing the number of trainable parameters. [6] proposed a weight-dropped LSTM, which used DropConnect on hidden-to-hidden weights as a form of recurrent regularization. Further, they introduced a variant of averaged stochastic gradient method named NT-AvSGD, which is a successful improvement in the language model optimization method. [25] proposed to train two identical copies of an RNN (that share parameters) with different dropout masks while minimizing the difference between their predictions. [12] identified the Softmax bottleneck by formulating language modeling as a matrix factorization problem, and proposed a method called Mixture of Softmax to address it.

For these large-scale LSTM language models, they mainly leveraged the regularization methods to handle the over-fitting of the models, so that the LSTM language model can achieve an ideal result on a large scale. In addition, the use of NT-AvSGD is also one of the key factors to greatly enhance the effect of the models [6], [12], [25]. However, the introduction of NT-AvSGD increases the training epochs of AWD-LSTM, so some sophisticated improvements on the AWD-LSTM will require a lot of extra training time and computational resources.

Different from these existing work, our goal is to overcomethis drawback caused by the correlation between the newly added hidden states and original hidden states of a certain single LSTM, which lead to a slower improvement of overall effect. Besides, in order to enable our improvements to work on models with NT-AvSGD optimization method, we design a simple but effective method: appropriately reduce the hidden size of each LSTM in original language model, then construct several Minor LSTMs with the saved parameters, and apply them in extracting a set of auxiliary features for each layer. Due to the scale of Minor LSTMs are small, our

method does not introduce extra parameters. Furthermore, the existing regularization methods can be combined with our method to further enhance the final effect. We will describe the motivation and the method in detail in the following Section III.

B. Shortcut connection

Formally, the Minor LSTM we proposed (in Section IV) can be seen as a variation of Shortcut connection [26], [27]. An early practice of training multi-layer perceptrons (MLPs) was to add a linear layer connected from the network input to the output [26]. In [28], a few intermediate layers were directly connected to auxiliary classifiers for addressing vanishing/exploding gradients. [29] proposed methods for centering gradients and propagated errors, implemented by shortcut connections. In recent years, many works on shortcut connections have achieved great success in fields of computer vision and natural language processing. [27] constructed a deep Residual Network with identity shortcut connections, which won the 1st place on the ILSVRC 2015 classification task. [30] designed a Highway Network and presented shortcut connections with gating function. [31] introduced the shortcut connections into convolutional network and named it Dense Convolutional Network (DenseNet). In task of language modeling, [30] proposed a Recurrent Highway Network, which extended the LSTM architecture to allow step-to-step transition depths larger than one by a set of shortcut connection between hidden layers. [32] compared the multi-layer LSTM and NAS with shortcut connections, and demonstrated the LSTM language model with skip connections can achieve the state-of-the-art performance.

Different from existing shortcut connections in deep networks, our proposed Minor LSTM is applied in shallow LSTM language models (no more than 5 layers), and used to extract auxiliary features for the Major LSTM, not to facilitate model optimization. In sum, our Minor LSTM is essentially different from the existing shortcut connections, and more detailed comparisons are in Section III and Section V.

III. The Vanilla LSTM Language Model

In this section, we firstly introduce the propagation processes in forward and backward of the vanilla LSTM language model shown in Fig. 1. Then we explore the impact of different LSTM hidden sizes on the overall model performance, the experimental results demonstrate that when the scale of a certain LSTM is large enough, simply lengthening the hidden states will introduce tremendous extra parameters, but has little effect.

A. Overview of the vanilla LSTM language model

A vanilla K-layer LSTM language model can be simply divided into three components: the input of word embeddings, the model of LSTMs, and the Softmax layer. We will describe the overall model from its propagation processes in the forward and backward respectively.

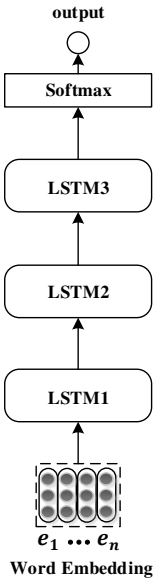


Fig. 1. The architectures of a K-layer vanilla LSTM language model (K=3).

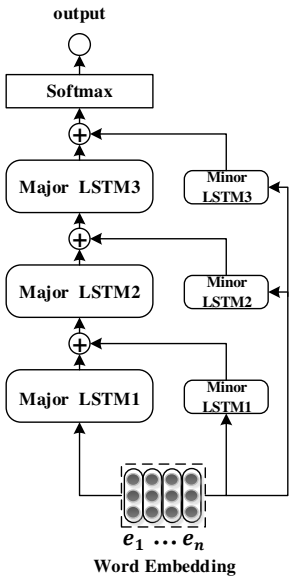
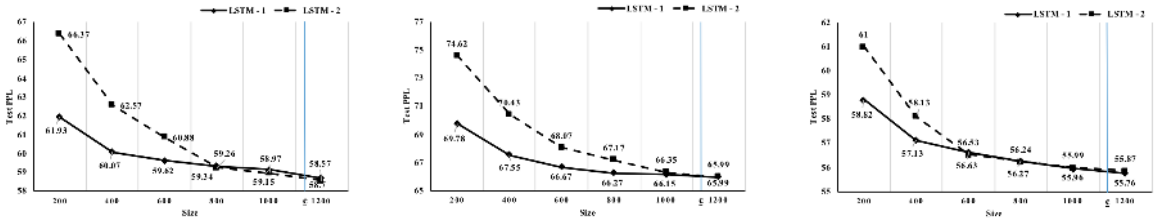


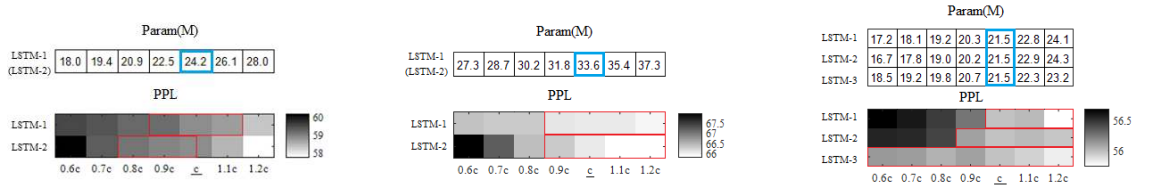
Fig. 2. The proposed architecture of a K-layer MMLSTMs language model (K=3). This architecture is an improvement from the vanilla LSTM language model shown in Fig. 1 where each LSTM layer is replaced by our proposed MMLSTM layer.

In the forward propagation, assume that $S = [w_1..., w_i..., w_n]$ denotes a sentence sequence consisting of n words. The words in S are randomly initialized to a set of real-valued embeddings $E_s = [e_1..., e_i..., e_n]$ at the beginning of model training. Then the E_s is fed into the K-layer LSTMs. There are three gates (input gate i, forget gate f, and output gate o) and a memory cell c in each



(a) AWD-LSTM on PTB (b) AWD-LSTM on WT2 (c) AWD-LSTM-MoS on PTB

Fig. 3. The model perplexities for the specific LSTM layer with different hidden sizes (from 200 to 1200 at interval of 200). (a) and (b) are AWD-LSTM on PTB and WT2, (c) is AWD-LSTM-MoS on PTB. In each subfigure, LSTM-i $(i = 1, 2)$ represents that the i_{th} LSTM layer is the specific LSTM with a variable hidden size, the other LSTM hidden sizes are fixed as in [6] and [12]. c is the standard hidden size for both LSTM-1 and LSTM-2 in (a), (b), (c).



(a) AWD-LSTM on PTB (b) AWD-LSTM on WT2 (c) AWD-LSTM-MoS on PTB

Fig. 4. The parameter counts (Param) and perplexity bands (PPL) of the language model for the specific LSTM layer with hidden sizes from $0.6c$ to $1.2c$ at interval of $0.1c$ (≈ 100). For each model in different corpus, c is the standard hidden size for the specific LSTM set in [6] or [12]. LSTM-i $(i = 1, 2, 3)$ represents that the i_{th} LSTM layer is the specific LSTM with a variable hidden size. It is worth noting that for both LSTM-1 and LSTM-2 in AWD-LSTM ((a) and (b)), the parameter counts are always the same, so we present the common parameter count at each hidden size. And the blue boxes mark the parameter counts corresponding to the standard hidden sizes of LSTM-i $(i = 1, 2, 3)$. In the perplexity band for each specific LSTM layer, the part in which the perplexity differ from the standard perplexity (in [6] or [12]) within the range of $[-0.25, 0.25]$ are marked with a red box.

LSTM. Given input $V_s^l = \left[v_1^l ..., v_i^l ..., v_n^l\right]$, the j_{th} hidden state of the l_{th} LSTM layer h_j^l is updated as follows:

$$i_j^l = \sigma\left(W_i^l v_j^l + U_i^l h_{j-1}^l + b_i^l\right) \tag{1}$$

$$g_j^l = \tanh\left(W_r^l v_j^l + U_r^l h_{j-1}^l + b_r^l\right) \tag{2}$$

$$f_j^l = \sigma\left(W_f^l v_j^l + U_f^l h_{j-1}^l + b_f^l\right) \tag{3}$$

$$o_j^l = \sigma\left(W_o^l v_j^l + U_o^l h_{j-1}^l + b_o^l\right) \tag{4}$$

$$c_j^l = i_j^l \odot g_j^l + f_j^l \odot c_{j-1}^l \tag{5}$$

$$h_j^l = o_j^l \odot \tanh\left(c_j^l\right) \tag{6}$$

where v_j^l is the j_{th} input vector (e_j if it is the first LSTM, or the output of the previous LSTM layer h_j^{l-1}), h_j^l is the current exposed hidden state, c_j^l is the memory cell state, and $\odot$ is element-wise multiplication. $\left[W_i^l, W_r^l, W_f^l, W_o^l\right]$ $\left[b_i^l, b_r^l, b_f^l, b_o^l\right]$ are parameters of weight matrices and bias vectors, which are generated by model training. Then the c_j^l participates in the processing of the next input vector v_{j+1}^l. And h_j^l is fed into the next layer. The output of the last LSTM layer h_j^K is used in predicting the probabilities distribution of the next word. Assume there are $|D|$ different words in corpus that make up a vocabulary list D. Given the preceding j words, the Softmax layer calculates the probability distribution of $(j+1)_{th}$ word in the word sequence as follow:

$$P(w_{j+1}|w_1, ..., w_j) = \frac{e^{W_s \cdot h_j^K}}{1^T \cdot e^{W_s \cdot h_j^K}} \tag{7}$$

where W_s is weight matrix of the Softmax layer, which has $|D|$ rows. h_j^K is the output of the last LSTM layer corresponding to j_{th} input, $1^T = (1, ..., 1) \in R^{|D|}$, and $P(w_{j+1}|w_1, ..., w_j) \in R^{|D|}$ contains the predicted probabilities corresponding to each word in vocabulary list D.

The training of the language model is implemented in backward propagation. In the backward propagation process, the parameters of the model are updated by optimizing the loss function. We take the word sequence S as an example, whose loss function is negative log-likelihood (NLL) of the sequence:

$$NLL\left(w_1, ..., w_n\right) = -\frac{1}{n}\sum_{j=1}^{n} 1_{w_{j+1}}^T \cdot \log P\left(w_{j+1}|w_1, ..., w_j\right) \tag{8}$$

where $P(w_{j+1}|w_1, ..., w_j)$ is calculated in (7), $1_{w_j}^T = (0, ..., 1, ..., 0) \in R^{|D|}$, the 1 corresponding to the index of w_j in vocabulary list D. We set θ as the collection of all parameters in the language model, which consists of word embeddings in vocabulary list D, parameters in the K-layer LSTMs, and the weight matrix in the Softmax layer. The goal of backward propagation is to minimize the loss function, and find the optimal value of θ:

$$\underset{\theta}{\operatorname{argmin}}\, \mathrm{NLL}\left(w_1, \ldots, w_n\right) \tag{9}$$

The final language model is established by updating parameters in θ layer-by-layer.

B. The Drawbacks of LSTM in the vanilla LSTM language model

Although the vanilla LSTM model has become one of the most promising language models in these years, there are still some potential drawbacks on it. As we discussed before, when a certain LSTM layer reaches a sufficient large scale, simply lengthening its hidden states has little improvement on the model effect, but introduces a lot of extra parameters. In order to reflect this drawback more intuitively, we investigate the influence of changes in different LSTM hidden size on the vanilla LSTM language model by experiments. We use AWD-LSTM[2] [6] and AWD-LSTM-MoS [12] as experimental objects, and apply perplexity as the evaluation metrics. Given a word sequence $[w_1, ..., w_n]$, whose perplexity is calculated as:

$$PPL\left(w_1, ..., w_{\mathrm{n}}\right) = \exp\left(\frac{\mathrm{NLL}\left(w_1, ..., w_{\mathrm{n}}\right)}{n}\right) \tag{10}$$

In our experiments, we choose a specific LSTM layer for each language model, and change its hidden size with the other hyper-parameters unchanged, then record the model perplexities on corpora of PTB and WT2[3]. We take the hidden size of each LSTM layer in [6] as the standard size for the models of AWD-LSTM, and the hidden size in [12] as the standard size for the models of AWD-LSTM-MoS. And we uniformly denote these standard sizes as c.

In order to obtain the overall trend of perplexities with changes in the specific LSTM hidden size, we firstly use three line graphs[4] to observe in a large hidden size range from 200 to 1200 at intervals of 200 in Fig. 3. From these line graphs we can clearly see that as the hidden sizes increase, the perplexities keep decreasing, but the rate of decline is getting slower and slower. This means that the contribution of newly added parameters to the improvement of model performance decrease. Besides that, Fig. 3 also shows that compared to the first layer, the model perplexities drop faster as the hidden size of the second LSTM layer increases. So, we can conclude that the second LSTM layer is more sensitive to the dimensional change than the first LSTM.

From Fig. 3, we can see the trend of the overall model effect when the specific LSTM hidden size changes at a large interval (200). For observing the influence of the specific LSTM hidden size changes in a more gradual way, we conduct experiments presented in Fig. 4. Fig. 4 shows the parameter counts (Param) and perplexity bands (PPL) of the model for the specific LSTM layer with hidden sizes from $0.6c$ to $1.2c$ at interval of $0.1c$. From the change in parameter count, the small changes in the specific LSTM hidden size can lead to obvious changes in overall parameter count. However, the final effects are not improved distinctively. In the perplexity band for each specific LSTM layer, the part in which the perplexity differ from the standard perplexity (in [6] and [12]) within the range of $[-0.25, 0.25]$ are marked with a red box. We found the hidden sizes of $0.9c$ in most layers are included in a red box, and the red box of the third LSTM layer in AWD-LSTM-MoS includes the hidden size of $0.6c$.

Combining the results in Fig. 3 and Fig. 4, we can clearly conclude that appropriately reducing the hidden size of a LSTM layer in the language model can reduce the useless parameters under the premise of maintaining the model performance. For the main reason of the phenomenon in Fig. 3 and Fig. 4, we exclude the following two common points: 1) Insufficient training: each model is trained as same as in [6] and [12], the number of training epochs achieves 500. 2) Overfitting: in Fig. 3 the regularization methods we use are same as the standard in [6] and [12], it can be found that the hidden size of almost all points for both LSTM-1 and LSTM-2 in each subfigure is less than the standard hidden size c, with only a few exceptions, but they do not cause serious overfitting. The similar situation occurs in Fig. 4. Therefore, we believe that the phenomenon is related to the structure of LSTM itself, which we will analyze in the next section.

IV. The Improvement for LSTM in the vanilla LSTM Language Model: Major-Minor LSTMs

In the previous sections, we have introduced the details of the vanilla LSTM language model and pointed out its drawback: in a nutshell, we found that when the scale of a certain LSTM is large enough, simply increasing its hidden size not only has little effect, but also introduces a large number of extra parameters.

In this section, we provide a further theoretical analysis, by firstly converting the operation of the LSTM to a general RNN equivalent form, in which we aim to reveal the existence of high correlation between features in the LSTM hidden states that is one of the main causes of the afore discussed drawback. At last, we propose a simple yet effective improvement on LSTMs in the language model called Major-Minor LSTMs to alleviate the issue.

A. Derivation of converting LSTM to RNN

In order to facilitate the following analysis of LSTM, we simplify the operation of each LSTM in the language model to a general RNN. For the sake of clarity, we omit the symbol superscripts of the layer number in Section III.

Assume $v = (v_1, ..., v_n)$, $v_j \in R^{d_0}$ are the input sequence of the LSTM, the operation of LSTM can be written as:

$$y_j = f\left(Wv_j + Uy_{j-1} + b\right) \tag{11}$$

$$W = \left[\tilde{W}; \mathbf{0}_1^T\right]^T \in R^{5d_1 \times d_0}, \tilde{W} = \left[W_i^T; W_r^T; W_f^T; W_o^T\right]^T \tag{12}$$

$$U = \begin{bmatrix} \tilde{U} & \mathbf{0}_2 \\ \mathbf{0}_3 & I \end{bmatrix} \in R^{5d_1 \times 2d_1}, \tilde{U} = \left[U_i^T; U_r^T; U_f^T; U_o^T\right]^T \tag{13}$$

[2]The third LSTM layer of AWD-LSTM is connected to the Softmax layer, whose input size is fixed after tying the word vectors and word classifier. So we only experiment over the first two LSTM layers.

[3]Detailed information about PTB and WT2 datasets will be introduced in Section V.

[4]Since the hidden size of the third LSTM of AWD-LSTM-MoS is much smaller than the first two layers, it is not suitable to draw their hidden size changes in the same coordinate system, so we only plot the hidden size changes of the first two LSTM layers in the line graph.

$$b = \left[\tilde{b}; \mathbf{0}_4^T\right]^T \in R^{5d_1}, \tilde{b} = \left[b_i^T; b_r^T; b_f^T; b_o^T\right]^T \tag{14}$$

$$f(x) = f_{d_1}(x) = f_{d_1}^{(3)} \circ f_{d_1}^{(2)} \circ f_{d_1}^{(1)}(x) \tag{15}$$

$$\begin{aligned} f_{d_1}^{(1)}(x) = &(\sigma(x_{1:d_1}), \tanh(x_{d_1+1:2d_1}), \\ &\sigma(x_{2d_1+1:4d_1}), x_{4d_1+1:5d_1}) \end{aligned} \tag{16}$$

$$\begin{aligned} f_{d_1}^{(2)}(x) = &(x_{1:d_1} \odot x_{d_1+1:2d_1} + x_{2d_1+1:3d_1} \\ &\odot x_{4d_1+1:5d_1}, x_{3d_1+1:4d_1}) \end{aligned} \tag{17}$$

$$f_{d_1}^{(3)}(x) = (\tanh(x_{1:d_1}) \odot x_{d_1+1:2d_1}, x_{1:d_1}) \tag{18}$$

Where $y_j = (h_j, c_j) \in R^{2d_1}$, is the concatenation of the hidden state and the cell state corresponding to (5) and (6) of the LSTM; $\left[\tilde{W}, \tilde{U}, \tilde{b}\right]$ are the non-recurrent, recurrent parameter matrices and bias parameter vectors in (1), (2), (3), (4); $\mathbf{0}_1 \in R^{d_1 \times d_0}$, $\mathbf{0}_2 \in R^{d_1 \times d_1}$, $\mathbf{0}_3 \in R^{d_1 \times 4d_1}$ and $\mathbf{0}_4 \in R^{d_1}$ are fixed zero matrices; $I \in R^{d_1 \times d_1}$ is the fixed identity matrix. $x_{p:q}$ denotes the fragment of vector x from the p_{th} to the q_{th} elements.

In fact, the result of the affine transformation $Wx_j + Uy_{j-1} + b$ is the stacking of results of (1), (2), (3), (4) before activation function and c_{j-1}; $f_{d_1}^{(1)}(Wv_j + Uy_{j-1} + b)$ is the stacking of i_j^l, g_j^l, f_j^l, o_j^l in (1), (2), (3), (4) and c_{j-1}; $f_{d_1}^{(2)} \circ f_{d_1}^{(1)}(Wv_j + Uy_{j-1} + b) \in R^{2d_1}$ is concatenation of the current cell state c_j and the output gate o_j^l. At last, the output of $f_{d_1}^{(3)} \circ f_{d_1}^{(2)} \circ f_{d_1}^{(1)}(Wv_j + Uy_{j-1} + b) = (h_j, c_j) \in R^{2d_1}$, which makes the (11) established.

B. The High Correlation between Features in LSTM Hidden States

Through the above derivation process, the complex operation of LSTM is transformed into a simple recursive process of RNN. In various task of NLP, this recursive operation mode enable the LSTM to process the input texts of any length and capture the long-term dependencies. However, we find that this recursiveness can also lead to a high correlation between different features in each LSTM hidden state, which impedes the enhancement of model effect.

In general, if the hidden size of a LSTM is extended, its capability of extracting features will be enhanced. However, since LSTM is a recursive model, there is a high correlation between its newly extended hidden states and original hidden states, which may limit the diverse semantic features expression of the expanded hidden states. We randomly choose a LSTM in the multi-layer LSTM language model as the example, as in (11), the calculation of the LSTM can be written as: $y_j = f(Wv_j + Uy_{j-1} + b)$, $v_j \in R^{d_0}$, $y_j = (h_j, c_j) \in R^{2d_1}$. After increasing the hidden size from d_1 to $d_1 + d_2$, the extended output is $y_j^{new} = \left(\left(h_j, h_j^{'}\right), \left(c_j, c_j^{'}\right)\right) \in R^{2(d_1+d_2)}$ can be split into two parts $y_j = (h_j, c_j) \in R^{2d_1}$ and $y_j^{'} = \left(h_j^{'}, c_j^{'}\right) \in R^{2d_2}$ (here we exchange the order of c_j and $h_j^{'}$ in y_j^{new} for the following description), where y_j is calculated as (11) - (18) and the $y_j^{'}$ can be obtained by an operation similar to y_j:

$$y_j = f_{d_1}\left(Wv_j + Uy_{j-1}^{new} + b\right) \tag{19}$$

$$y_j^{'} = f_{d_2}\left(W^{'}v_j + U^{'}y_{j-1}^{new} + b^{'}\right) \tag{20}$$

Now if we assume that there is no term of y_{j-1}^{new} in (19) and (20), the calculations of (19) and (20) are no longer recursive procedures. The hidden state y_j and $y_j^{'}$ are generated by parameters in $[W, b]$ and $\left[W^{'}, b^{'}\right]$, respectively[5]. That is to say, when the y_j is extended to y_j^{new}, the newly added parameters in $\left[W^{'}, b^{'}\right]$ can be used to generate more diverse semantic features for the input v_j independently of the original parameters in $[W, b]$.

But in fact, the two sets of parameters cannot transform the input v_j independently due to the existence of terms of y_{j-1}^{new}. The two sets of parameters in $[W, U, b]$ and $\left[W^{'}, U^{'}, b^{'}\right]$ must cooperate with each other to generate features in LSTM hidden states, which results in the entire hidden state affected by the change of any parameters in the two parameter sets. So y_j and $y_j^{'}$ are highly correlated, which is adverse for diversity and comprehensive feature expressions of the layer outputs h_j, $h_j^{'}$ within y_j and $y_j^{'}$. Because of this, simply lengthening the LSTM hidden states cannot improve the final model effect continuously. In Section V-B, we will further explore the feature correlation of LSTM hidden states in the language model by experiments.

C. Major-Minor LSTMs for Language Modeling

A simple and straightforward way to break the correlation between the newly added hidden states and the original hidden states of the LSTM is to leverage multiple LSTMs to extract semantic features of inputs, which also facilitates reducing model parameters[6]. On the other hand, it is necessary to keep a LSTM containing most of parameters so that it has sufficient generalization capability to extract complex semantic features from input, which will be verified in Section V-D2. To this end, we construct a large-scale Major LSTM to extract the main semantic features from the output of the previous layer and apply a small-scale Minor LSTM in generating a set of auxiliary features to cooperate Major LSTM, we named this architecture to Major-Minor LSTMs (MMLSTMs). In this way, y_j and $y_j^{'}$ in (19) and (20) are

[5]As in (12) - (14), there are also a part of fixed values in $[W, U, b]$ and $\left[W^{'}, U^{'}, b^{'}\right]$, but we only focus on the parts of parameters that can be updated in training.

[6]With a fixed output size n, using multiple small-scale LSTMs requires less parameters than using a single large-scale LSTM, because the $n \times n$-dimensional matrix of recurrent connection in the large LSTM is replaced by a set of $n_i \times n_i$–dimensional matrix distributed in recurrent connections in these small LSTMs and satisfy $\sum n_i = n$. So we can obtain $\sum (n_i)^2 \leq n^2$.

substituted by $y_j^{major} = \left(h_j^{major}, c_j^{major}\right) \in R^{2d_{ma}}$ and $y_j^{minor} = \left(h_j^{minor}, c_j^{minor}\right) \in R^{2d_{mi}}$:

$$y_j^{major} = f_{d_{ma}}\left(W^{major} v_j + U^{major} y_{j-1}^{major} + b^{major}\right) \tag{21}$$

$$y_j^{minor} = f_{d_{mi}}\left(W^{minor} v_j + U^{minor} y_{j-1}^{minor} + b^{minor}\right) \tag{22}$$

where $f_{d_{ma}}(x)$ and $f_{d_{mi}}(x)$ are non-linear functions defined as (15) - (18), W^{major}, W^{minor} correspond to (12) containing non-recurrent parameter matrices of two LSTMs. As (13), the form of U^{major}, U^{minor} are block diagonal matrices, which involve the recurrent parameter matrices. b^{major} and b^{minor} are biases as in (14). Obviously, the parameters of Major LSTM and Minor LSTM generate the two parts of hidden states h_j^{major} and h_j^{minor} independently.

The core idea of MMLSTMs is to divide the output features into two parts and generate them independently with two LSTMs. [33] also splits the output of the LSTM into several groups and utilizes the LSTM with matrix factorization to obtain the output of each group by separated input features. In contrast, our MMLSTMs always maintain the integrity of the input features instead of splitting them into separate parts, so that the complete input information can be taken into account during generating the outputs of both Major and Minor LSTMs. In addition, MMLSTMs leverage the vanilla LSTM rather than the LSTM with matrix factorization to ensure that the model expressiveness does not degrade due to the sharp drop in the amount of parameters.

For the MMLSTMs language model, there are two branches in each layer: Major LSTM and Minor LSTM. Naturally, according to above discussion, the inputs of Major and Minor LSTMs are both the output of the previous layer, and we highlight the relationship of "major" and "minor" by setting the ratio of their hidden sizes. For further improvement, in our proposed model, we use the original word embeddings as the input of Minor LSTM instead of the output of the previous layer. Fig. 2 shows the language model build by our MMLSTMs, which substitutes a MMLSTMs layer for each LSTM in the vanilla language model.

The reasons why we utilized the original word embeddings as the input of Minor LSTM are three-folds. First, according to our MMLSTMs design, we appropriately reduce the hidden size of original LSTM in LSTM language model as the Major LSTM, which will not impair its generalization ability significantly. Therefore, the Major LSTM has enough capability to process the output of the previous layer, and there is no need to process it with the Minor LSTM. Second, if we consider the layers before the current layer as a whole, the output of the previous layer can be written as the function of the original sequence embedding. The output of the previous layer is just an intermediate result for generating the auxiliary features of the current layer. The original embeddings, by contrast, can generate the auxiliary features more directly. Third, as mentioned by [6], the size of embedding should be smaller than the hidden size, so that the overfitting of the word embeddings can be eased. Therefore, the size of original word embeddings is always smaller than the output of LSTM layer, using the original embedding as the input of Minor LSTM requires fewer parameters. We will compare these two types input in detail in Section V.

It is worth noting that the Minor LSTM can be seen as a shortcut connection in form. But in fact, The Minor LSTM is essentially different from existing shortcut connections [27], [30], [31], [34], [35]. Firstly, the existing shortcut connection is an indispensable part of many deep networks, where its main role is to alleviate the issues encountered in training process (e.g. gradient vanishing/exploding). But in our scenario, we apply the Minor LSTM in shallow neural networks (no more than 5 layers), which does not suffer the same problem as those in deep networks in training process. So the motivation of Minor LSTM and existing shortcut connection are completely different. Secondly, in the architecture of MMLSTMs, we have defined the primary and secondary status of the Major and Minor LSTMs at the beginning, the Minor LSTM is just used for generating the auxiliary features. But for the existing neural networks with shortcut connection, the importance of the two branches (the main branch and the branch of shortcut connection) cannot be determined in advance. In the experimentation discussed in Section V, we will demonstrate that the primary-secondary structure of our MMLSTMs can effectively improve the performance of LSTM language model.

As for how the auxiliary features actually work, we do the following analysis:

When the output of the current MMLSTMs layer $\left(h_j^{major}, h_j^{minor}\right)$ is fed into the next layer, which is a combination of different components in forms of nonlinear affine transformation, the calculation of each component can be denoted as:

$$z_j = g\left(Ah_j^{major} + Cz_{j-1} + d + Bh_j^{minor}\right) \tag{23}$$

where $z_j \in R^{d_3}$ is the j_{th} output of the component, $\left[A \in R^{d_3 \times d_1}, B \in R^{d_3 \times d_2}, C \in R^{d_3 \times d_3}, d \in R^{d_3}\right]$ are parameters. g is the non-linear activation function of the component. If the component is not recursive, the C is set to 0. From (11) we can infer that the term of Bh_j^{minor} can work as an adaptive bias, which adjust the calculation of h_j^{major} dynamically. Different from the fixed bias d, the term Bh_j^{minor} will change with the various input. But if we use the $\left(h_j, h_j^{'}\right)$ generated by (19), (20) as the input of the component, the correlation between h_j and $h_j^{'}$ will constrain $h_j^{'}$, so that it cannot adjust the calculation of h_j freely.

V. Experiments

In this section, we conduct a series of experiments to verify the effectiveness of MMLSTMs in language model and explore the characteristics of MMLSTMs language model.

A. Results on Two Small Corpus: PTB and WT2

We firstly evaluate our methods on two small corpus Penn Treebank (PTB) [39] and the WikiText-2 (WT2) [36], and their

TABLE I
STATISTICAL RESULTS PENN TREEBANK (PTB), WIKITEXT-2 (WTB) AND WIKITEXT-103 (WT103) DATASETS.

	Penn Treebank			WikiText-2			WikiText-103		
	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
Arcitles	-	-	-	600	60	60	28,475	60	60
Tokens	887,521	70,390	78,669	2,088,628	217,646	245,569	103,227,021	217,646	245,569
Vocab		10,000			33,278			267,735	
OoV		4.8%			2.6%			0.4%	

TABLE II
SINGLE MODEL PERPLEXITY ON VALIDATION AND TEST SETS ON PENN TREEBANK. BASELINE RESULTS ARE OBTAINED FROM [6], [12]. * INDICATES USING DYNAMIC EVALUATION.

Model	#Param	Validation	Test
[10] – RNN-LDA + KN-5 + cache	9M	-	92.0
[14] - LSTM (large)	66M	82.2	78.4
[15] – Variational LSTM (MC)	66M	-	73.4
[17] - CharCNN	19M	-	78.9
[36] – Pointer Sentinel-LSTM	21M	72.4	70.9
[37] - LSTM + continuous cache pointer	-	-	72.1
[16] – Tied Variational LSTM + augmented loss	51M	71.1	68.5
[30]– Variational RHN	23M	67.9	65.4
[35] – NAS Cell	54M	-	62.4
[32] – 2-layer skip connection LSTM	24M	60.9	58.3
[36] – AWD-LSTM	24M	60.7	58.8
AWD-LSTM-MoS$^-$	18.8M	59.74	57.97
[12] – AWD-LSTM-MoS	21.5M	58.08	55.97
Ours-AWD-MMLSTMs-MoS	**21.3M**	**56.52**	**54.51**
[6] – AWD-LSTM + finetune	24M	60.0	57.3
AWD-LSTM-MoS$^-$ + finetune	18.8M	58.01	55.89
[12] – AWD-LSTM-MoS + finetune	21.5M	56.54	54.44
Ours-AWD-MMLSTMs-MoS + finetune	**21.3M**	**55.42**	**53.46**
[6] – AWD-LSTM + finetune + continuous cache pointer*	24M	53.9	52.8
[38] – AWD-LSTM + finetune + dynamic evaluation*	24M	51.6	51.1
AWD-LSTM-MoS$^-$ + finetune + dynamic evaluation*	18.8M	49.72	48.92
[12] – AWD-LSTM-MoS + finetune + dynamic evaluation*	21.5M	48.33	47.69
Ours-AWD-MMLSTMs-MoS + finetune + dynamic evaluation*	**21.3M**	**47.31**	**46.81**

TABLE III
SINGLE MODEL PERPLEXITY OVER WIKITEXT-2. BASELINE RESULTS ARE OBTAINED FROM [6] AND [12]. * INDICATES USING DYNAMIC EVALUATION.

Model	#Param	Validation	Test
[16] - Variational LSTM (tied) + augmented loss	28M	91.5	87.0
[37] - LSTM + continuous cache pointer	-	-	68.9
[32] - 2-layer skip connection LSTM (tied)	24M	69.1	65.9
[6] – AWD-LSTM	33M	69.1	66.0
AWD-LSTM-MoS$^-$	29.4M	67.33	64.64
[12] – AWD-LSTM-MoS	35M	66.01	63.33
Ours-AWD-MMLSTM-MoS	**32.3M**	**64.3**	**61.77**
[6] – AWD-LSTM + finetune	33M	68.6	65.8
AWD-LSTM-MoS$^-$ + finetune	29.4M	65.11	63.02
[12] – AWD-LSTM-MoS + finetune	35M	63.88	61.45
Ours-AWD-MMLSTM-MoS + finetune	**32.3M**	**63.11**	**60.51**
[6] – AWD-LSTM + finetune + continuous cache pointer*	33M	53.8	52.0
[38] – AWD-LSTM + finetune + dynamic evaluation*	33M	46.4	44.3
AWD-LSTM-MoS$^-$ + finetune + dynamical evaluation*	29.4M	44.23	42.19
[12] – AWD-LSTM-MoS + finetune + dynamical evaluation*	35M	42.41	40.68
Ours-AWD-MMLSTMs-MoS + finetune + dynamical evaluation*	**32.3M**	**41.91**	**40.15**

statistical results are shown in Table I[7]. We used perplexity as the primary metric, the results of both our models (AWD-MMLSTMs-MoS) and other competitive models on PTB and WT2 are reported in Table II and Table III respectively. Following is the descriptions of PTB and WT2 datasets.

PTB: The Penn Treebank dataset has long been a central dataset for experimenting with language modeling. The dataset has been preprocessed and does not contain capital letters, numbers, or punctuation. The vocabulary is capped at 10,000 unique words, quite small in comparison to most modern

[7]The statistical results are excerpted from: https://einstein.ai/research/blog/the-wikitext-long- term-dependency-language-modeling-dataset

TABLE IV
HYPER-PARAMETERS USED FOR AWD-MMLSTMS-MOS. V-DROPOUT ABBREVIATES VARIATIONAL DROPOUT [15]. SEE [6] FOR MORE DETAILED DESCRIPTIONS.

Hyper-parameter	PTB	WT2
Learning rate	20	20
MMLSTMs layer size	[1080, 1080, 620]	[1200, 1200, 650]
Proportions of Major LSTM	[90%, 90%, 60%]	[90%, 90%, 60%]
Embedding V-dropout	0.5	0.55
Hidden state V-dropout	0.25	0.2
Non-monotone interval	10	10

TABLE V
HYPER-PARAMETERS USED FOR DYNAMIC EVALUATION OF AWD-MMLSTMS-MOS. SEE [38] FOR MORE DETAILED DESCRIPTIONS.

Hyper-parameter	PTB	WT2
Batch size	150	130
Learning rate(η)	0.0024	0.00198
ε	0.0025	0.0023
λ	0.07	0.0245
bptt	7	7

datasets, which results in a large number of out of vocabulary (OoV) tokens.

WT2: WikiText-2 is sourced from curated Wikipedia articles and is approximately twice the size of the PTB dataset. The text is tokenized and processed using the Moses tokenizer [40], frequently used for machine translation, and features a vocabulary of over 30,000 words. Capitalization, punctuation, and numbers are retained in this dataset.

We applied our proposed MMLSTMs in AWD-LSTM–MoS architecture proposed by [12], and called the new model AWD-MMLSTMs-MoS. For fair comparison, we only adjust the hyper-parameters associated with our improvement, and set other hyper-parameters to be the same as in [12]. Table IV lists the hyper-parameters we adjust. For facilitating the adjustment of the regularization methods on Major and Minor LSTMs, we tied up their embedding and hidden state variational dropout values. We divided the dimension of each MMLSTMs layer output into 10 parts, and heuristically and manually search various combinations of Major and Minor LSTMs hidden size proportions. It was found that the hidden sizes of the Major LSTMs on three layers accounted for $[90\%, 90\%, 60\%]$ respectively can achieve the best effect on both datasets. The improvement of MMLSTM on the single LSTM can be divided into two steps: reducing the hidden size of the original LSTM, construct the Minor LSTM. For exploring the influence of smaller LSTM hidden sizes on the model performance, we built a AWD-LSTM-Mos$^-$ with the same LSTM hidden sizes as the Major LSTMs in our AWD-MMLSTMs-MoS, which can better compare the performance of the language model before and after using Minor LSTMs.

We also applied dynamic evaluation [38] to further enhance our model performance, whose hyper-parameters are listed in Table V. Dynamic evaluation [41]–[43] adapts models to recent sequences using gradient descent based mechanisms. Different from the traditional static evaluation, the dynamic evaluation will dynamically adjust the trained model parameters as the test progresses. For traditional static evaluation, the model can only use the training set to update the parameters, while dynamic evaluation also allows model to utilize the samples in test set that have been tested. Therefore the model effects can been improve significantly with the dynamic evaluation.

The evaluation process can be divided into three stages: training, fine-tuning, and dynamic estimation. After the training, we can obtain a preliminary model, which will be tuned to achieve better results in the further stages namely the fine-tuning and dynamic estimation. The Table II and III are divided into three parts, which corresponding to the three stages. It can be seen that AWD-MMLSTMs-MoS exceeds other baselines in all stages on both PTB and WT2, which demonstrates that MMLSTMs can effectively enhance the model performance without sacrificing the extra parameters. Moreover, we guaranteed that the hyper-parameters are consistent with the baseline model (AWD-LSTM-MoS) except for the part of MMLSTM, so it excludes the influence of other factors (e.g. regularization, optimizer) on the final results.

B. Results on a medium corpus: WT103

To further validate the effects of MMLSTMs language model on real life corpus, we conducted an experiment on a widely-used medium-scale dataset WikiText-103 (WT103). The WT103 is also derived from Wikipedia, whose training set is 103M and vocabulary contains 267K words. Compared to the preprocessed version of WT2, WT103 is over 55 times larger. Therefore, WT103 is a representative language model dataset, Table I shows the details.

In this experiment, in order to eliminate the impact of some advanced regularization and optimization techniques on evaluate results, we directly improve from a simple 2-layer LSTM language model, and only add some basic regularization techniques to prevent overfitting, then leverage Adam [45] optimizer with learning rate of 0.001 to speed up the training. The details are shown in Table VI.

The experimental results on WT103 are reported in Table VII. From Table VII, we can see that the LSTM language model implemented by us surpasses the LSTM language model in [37]. Comparing our LSTM language model and MMLSTMs language model, it can be found that the perplexity of the MMLSTMs language model is about 3.3 points lower than the baseline LSTM language model we construct. In addition, many regularization techniques in AWD-LSTM are excluded, which fully validates the effectiveness of the MMLSTMs on real life dataset.

C. Evaluation on output feature correlation

Combining our previous analysis in Section IV-B, for the single LSTM in each layer of existing LSTM language model, the generation of a specific output feature requires the joint implementation of all parameters during training, which also affects the other output features. This to some extent limits the free expression of various semantic features and increase

TABLE VI
THE DETAILS OF BASELINE LSTM LANGUAGE MODEL AND MMLSTMS LANGUAGE MODEL ON WT103. BPTT ABBREVIATES BACKPROPAGATION THROUGH TIME. RECURRENT WEIGHT DROPOUT REPRESENTS THE DROPOUT CONNECTION ON LSTM RECURRENT MATRIX, SEE [6] FOR MORE DETAILS. EXCEPT FOR THE HIDDEN SIZES OF LSTM (MMLSTM) LAYERS, ALL OTHER HYPER-PARAMETERS ARE THE SAME.

Model	LSTM language model	MMLSTMs language model
Embedding size	256	256
LSTM(MMLSTM) size	1024	1100
Proportions of Major LSTM	$[100\%, 100\%, 100\%]$	$[90\%, 90\%, 90\%]$
LSTM (MMLSTM) layer	2	2
BPTT	70	70
Batch size	128	128
Number of epoch	50	50
Weight tying [16]	True	True
Dropout of LSTM (MMLSTM) output	0.1	0.1
Recurrent weight dropout [44]	0.15	0.15

TABLE VII
SINGLE MODEL PERPLEXITY OVER WIKITEXT-103.

Model	#Param	Validation	Test
LSTM language model [37]	-	-	48.7
LSTM language model(our implement)	75.4M	47.01	47.95
MMLSTMs language model	75.0M	43.45	44.69

the correlation of different features. For the model with high-correlated output features, a natural and direct manifestation is that a part of its multiple output features need to work together closely in the subsequent operations, and once some of the multiple high-correlated output features are missing, the subsequent operations will be influenced significantly and the final results will get worse obviously. Thus, in order to visually verify that the correlation between our MMLSTM output features is weaker than the general LSTM, we designed the following experiment on the PTB corpus.

During test, we arbitrarily masked a part of the features (set the value to 0) within different output vectors of a LSTM layer in the trained language model, and refer to this layer as masked layer. The process is same as the forward propagation of the model with dropout in training process. One of the effects of the dropout is to prevent complex co-adaptations in which a feature dector is only helpful in the context of several other specific feature detectors [46]. In our scenario, we only need to compare the degree of correlation between the output features of general LSTM and MMLSTM, which does not require the backward propagation as in dropout.

In the experiments, we constructed five language models denoted as LM, LM$^+$, LM$^\#$, LM$^{\#\#}$, MMLM. LM is the original AWD-LSTM-MoS in [6]; for LM$^+$, we slightly amplified the dropout rate of LSTM hidden states, recurrent weights in AWD-LSTM-MoS (increase by 0.025 per value) to study whether increasing the intensity of regularizations can effectively reduce the correlation of LSTM output features. For fair comparison, we set three language models of LM$^\#$, LM$^{\#\#}$ and MMLM: LM$^\#$ adjusts the hidden size of each LSTM layer in LM to be the same as our AWD-MMLSTM-MoS in Section V-A, and keep the other hyper-parameters unchanged; LM$^{\#\#}$ denotes the AWD-LSTM-MoS with all hyper-parameters to be the same as AWD-MMLSTM-MoS (including all hidden sizes and dropout rates); MMLM is the AWD-MMLSTM-MoS in Section V-A.

Similar to dropout, we leveraged a Bernoulli distribution with probability P to decide which features within test layer output are masked, the range of P is from 0 (no mask) to 0.5 (mask a half features) at interval of 0.05. Figure 5 (a), plots the line charts of perplexities of the five language models with all three LSTM layers as masked layers. Figure 5 (b), Figure 5 (c), Figure 5 (d) separately set the masked layer as the first to the third LSTM layer in each model. In all the sub-figures, the perplexity of every model is the average of five runs for each value of P.

From the four sub-figures of Figure 5, we can observe that regardless of whether the test layer is all LSTMs in language model or a certain LSTM layer, the effect of MMLM is consistently better than the other three language models, which fully reflects the superiority of MMLSTM. Comparing the LM$^{\#\#}$ with MMLM, whose hyper-parameters are the same except for the structure of each LSTM layer, we can conclude that the significant weakening of the feature correlation is absolutely due to the MMLSTMs.

For the two object pairs: LM and LM$^+$, LM$^\#$ and LM$^{\#\#}$, LSTM hidden size and other hyper-parameters are the same in language models of each pair except for the regularization in each LSTM, but the LSTM feature correlations is not distinctly weakened by increasing the regularization intensity of LSTM, especially for LM$^+$. Although the dropout rates of different parts of the LSTM in LM$^+$ are amplified by 0.025, the LSTM feature correlation is almost unchanged. In contrast, using MMLSTM can weaken the LSTM feature correlation more simply and effectively than fine-tuning of some existing regularization methods. From this phenomenon we can also conclude that due to the existing regularization can not break the shackles of the recursive pattern of the single LSTM, the degree of output feature correlation can not been weaken once it reaches a certain level. But our MMLSTM can further reduce

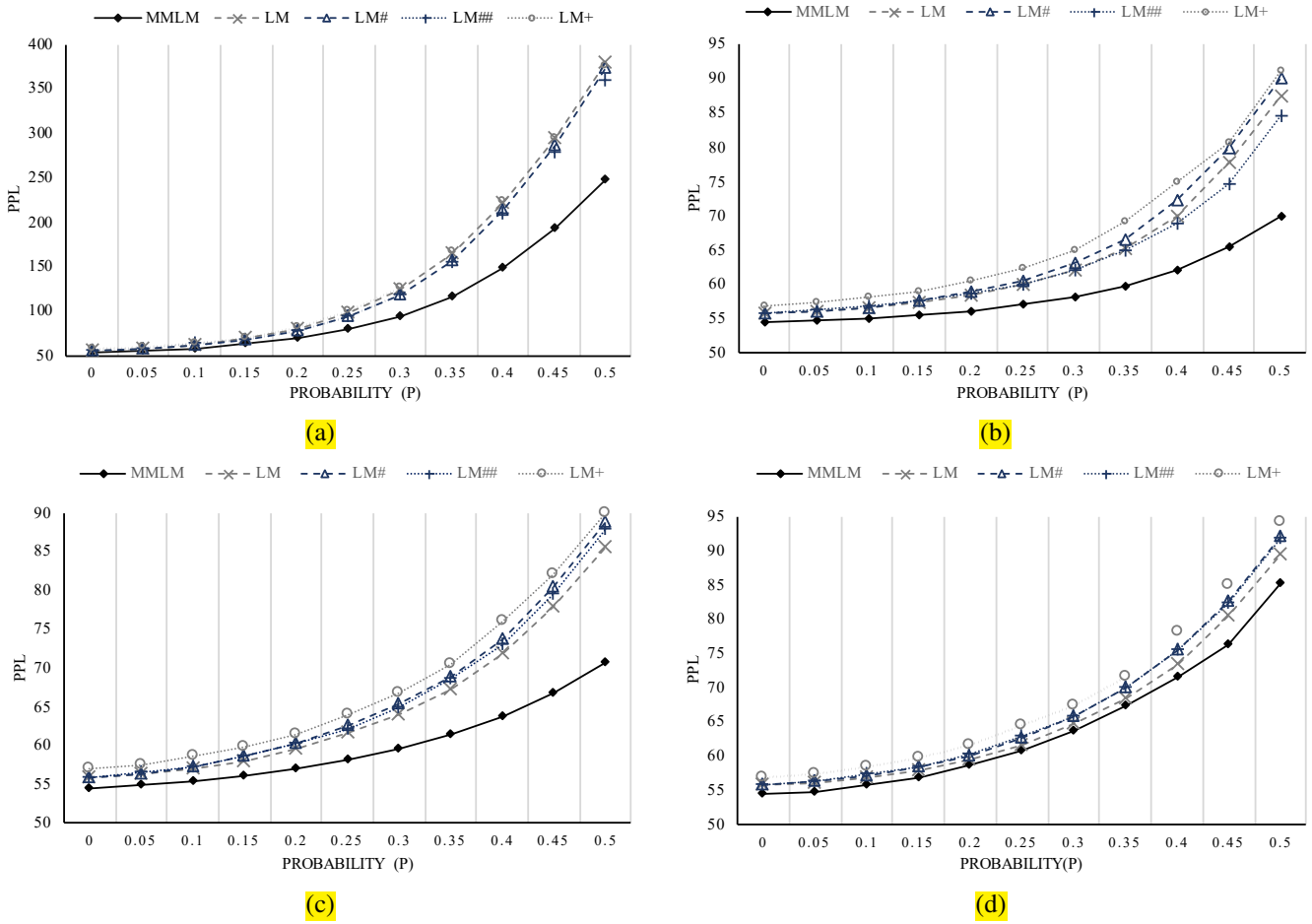


Fig. 5. The test perplexities (from P=0 to 0.5) of MMLM, LM, LM$^{\#}$, LM$^{\#\#}$, LM^{+} with different masked layers on PTB corpus. (a) Masked layer is all LSTMs in corresponding language model. For MMLM, each LSTM layer involves both Major and Minor LSTMs (the same for (b), (c), (d)). (b) Masked layer is the first LSTM layer. (c) Masked layer is the second LSTM layer. (d) Masked layer is the third (last) LSTM layer. In all sub-figures, the perplexity of every model is the average of five runs for each value of P.

the feature correlation of the layer output.

Another noteworthy finding is that the performance of LM* is not obviously distinct from LM, but the parameter count has increased from 21.5M to 24.9M, which again confirms that: when the hidden sizes of LSTMs in existing LSTM language model are large enough, continuing to lengthen the hidden states not only wastes tremendous parameters but also has little effect.

D. Further Investigation

1) Different inputs for Minor LSTM: In the Section IV-C, we explained the reason why we used the original embeddings as the input of the Minor-LSTMs instead of the output of the previous layer. To compare the influence of the two types inputs on the final results, we conducted experiments on PTB and WT2 datasets, while limiting the amount of parameters. We build two AWD-MMLSTMs-MoS with the two types input for the Minor LSTMs, which are denoted as MMLM-1 (original embedding as input) and MMLM-2 (previous layer output as input). The MMLM-1 is the same as AWD-MMLSTM-MoS in Section V-A. For the MMLM-2, its hidden size in the first two LSTM layers are set to 1150, so that the parameters of the two models are roughly equal. Besides, the other hyper-parameters of MMLM-1 and MMLM-2 are the same as in Table IV. The results in Table VIII showed that the MMLM-1 steadily surpasses MMLM-2 on both datasets, which demonstrates the shortcut connection from input word embedding to each Minor LSTM is helpful for our AWD-MMLSTM-MoS under the hyper-parameters in Table IV.

2) Different ratios for Major and Minor LSTM hidden sizes: In the Section IV-C, we elaborated the difference between the Minor LSTM in our MMLSTMs and the shortcut connections in existing deep networks. A typical difference is that the MMLSTMs assigns the importance of Major and Minor LSTMs by setting their relative dimensional size at the beginning of the design. The Minor LSTM is only for generating a set of auxiliary features for the output of the Major LSTM; Whereas in existing deep networks with shortcut connection, there is no clear importance division between the main branch and the branch of shortcut connection. In most cases, the two braches interact with each other by element-wise addition operation [27], [30].

In this subsection, we verified the validity of the primary and secondary structure of our MMLSTMs through experiments,

TABLE VIII
THE PERPLEXITIES OF MMLM-1 AND MMLM-2 ON PTB AND WT2 DATASETS.

Model	**PTB**			**WT2**		
	#Param	Val	Test	#Param	Val	Test
MMLM-1	**21.3M**	**56.52**	**54.51**	**32.3M**	**64.30**	**61.77**
MMLM-2	21.7M	57.95	55.93	32.5M	65.43	62.76

TABLE IX
THE HIDDEN SIZES OF LSTM LAYER IN MODELS WITH DIFFERENT RATIOS.

Ratio of Major-Minor	**Proportions of Major LSTM**	**MMLSTMs (LSTM) Size**		**#Param (M)**	
		PTB	**WT2**	**PTB**	**WT2**
5:5	[50%, 50%, 50%]	1560	1550	24.3	33.4
6:4	[60%, 60%, 60%]	1500	1500	24.2	33.5
7:3	[70%, 70%, 70%]	1420	1440	24.1	33.8
8:2	[80%, 80%, 80%]	1330	1330	24.1	33.5
9:1	[90%, 90%, 90%]	1240	1230	24.2	33.3
10:0 [6]	[100%, 100%, 100%]	1150	1150	24.2	33.6

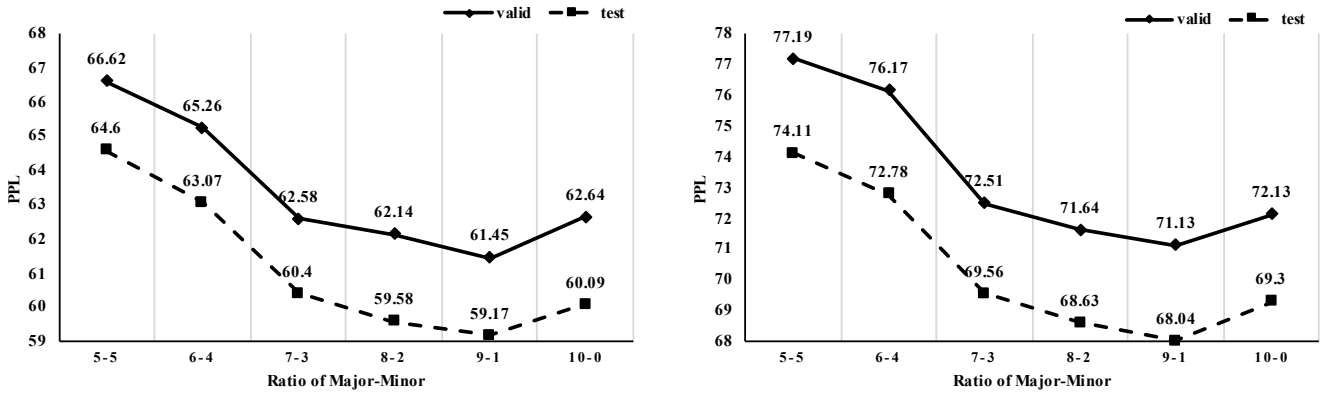


Fig. 6. The valid and test perplexities of AWD-MMLSTMs (training for 300 epochs) under different Major and Minor dimension ratios on PTB (a) and WT2 (b).

TABLE X
ABLATION STUDY ON PTB AND WT2 DATASETS. ADJUSTED DIM MEANS ADJUSTED DIMENSION OF THE LAYER REMOVED THE MINOR LSTM.

Model	**PTB**				**WT2**			
	Adjusted Dim	#Param	Val	Test	Adjusted Dim	#Param	Val	Test
MMLM-w/o-Minor-1	1030	21.5M	57.3	55.10	1150	32.5M	65.34	62.73
MMLM-w/o-Minor-2	1030	21.5M	57.14	54.93	1130	32.6M	65.07	62.43
MMLM-w/o-Minor-3	520	21.5M	57.84	55.74	550	32.6M	65.66	63.02
MMLM (in Section IV-B)	-	21.3M	56.52	54.51	-	32.3M	64.3	61.77

TABLE XI
THE MAIN HYPER-PARAMETERS OF NMT MODELS.

Model	**Baseline (we implement)**	**MMLSTM-NMT-model**
Source(target) embedding size	1000	1000
Encoder (Decoder) LSTM layer	4	4
LSTM (MMLSTM) sizes	[1000, 1000, 1000, 1000]	[1070, 1070, 1070, 1070]
Proportions of Major LSTMs	[100%, 100%, 100%, 100%]	[70%, 70%, 80%, 90%]
Initial learning rate	1.0	1.0
Dropout rate	1.0	1.0
Training step	340000	340000

TABLE XII
THE WMT 15 AND 16 GERMAN-ENGLISH RESULTS OF SINGLE NMT MODELS.

Model		Baseline [47]	Baseline + BPE (We implement)	MMLSTM-NMT model + BPE
# Param		170.7M	170.7M	172.8M
WMT	PPL	9.7	6.83	**6.15**
2015	BLEU	24.9	29.36	**30.83**
WMT	PPL	-	5.39	**4.88**
2016	BLEU	-	33.38	**35.48**

thus demonstrating the difference between our MMLSTMs and existing shortcut connections. We utilized the AWD-LSTM as the basic model, and replaced its LSTM layers with the MMLSTMs. All hyper-parameters are consistent with [6], except for the overall output size of each MMLSTMs and the ratio of the Major and the Minor LSTM hidden sizes. We adjusted the dimensional ratios of the Major LSTM and Minor LSTM from $5:5$ to $9:1$, and set different overall output sizes to ensure that the parameter counts of each model roughly equal. The experiments are conducted on PTB and WT2 datasets, and the output dimension of the hidden layers in different model are listed in Table IX.

Fig. 6 showed the perplexities (training for 300 epoch) of the models under different Major and Minor dimension ratios on PTB and WT2 datasets. We can clearly see that as the ratio of the Major and the Minor LSTMs decreases, the perplexities on valid and test sets continue to decline from the ratios of $5:5$ to $9:1$, but when the ratio reaches 10:0, that is, there is no Minor LSTM at all, the perplexities rebound. This phenomenon indicates that the Major LSTM requires sufficient parameters to extract the main semantic features, and the Minor LSTM is helpful when it is used to generate the auxiliary features in a secondary position. This experiments fully reflects the difference between our Minor LSTM and existing shortcut connections.

3) Ablation Study for Minor LSTMs in Different LSTM layers: Through our experiments, we can reach a conclusion that, using MMLSTMs layers can control the amount of parameters and improve the model performance. To further study the contribution of the Minor LSTM in different MMLSTMs layers, we conducted an ablation study for AWD-MMLSTMs-MoS (in the Section IV-B, denoted as MMLM) on both PTB and WT2. We removed the Minor LSTM from each MMLSTMs layer by layer, and denoted the model with the removed layer as MMLM-w/o-Minor-i $(i = 1, 2, 3)$. The dimension of the layer removed the Minor LSTM is adjusted to keep total parameters consistent, and the other hyper-parameters are the same to ensure a single variable contrast experiment. We also excluded the fine-tuning and dynamic evaluation steps, and the results are reported in Table X.

Comparing with MMLM (in the Section IV-B), we can conclude that the Minor LSTM of the third layer has the greatest effect on the final performance of the model, followed by the first Minor LSTM and finally the second.

4) The MMLSTM on Machine Translation: For verifying the effectiveness of MMLSTM in the sequence to sequence tasks, we applied it in the neural machine translation (NMT) model of [47], which is a classical NMT model and takes advantage of multi-layer LSTMs in both encoder and decoder. The original NMT model, as our baseline, involves three global attention mechanisms in the process of generating the context vector, and we choose the general attention. The models are trained on the WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs, and sentence were encoded using Byte-pair encoding [48] (short for BPE). We also use SGD as optimizer. The other training details are the same as in [47].

During transforming the baseline model, we noticed that: 1) Each layer of encoder LSTM and decoder LSTM corresponds to each other, the last cell state of the encoder LSTM are fed into the corresponding decoder LSTM, so the hidden size of each encoder LSTM should be consistent with corresponding decoder. 2) In [47], an input-feeding approach is proposed to make alignment decision jointly take into account past alignment information. To this end, the input of the first LSTM layer in decoder is the concatenation of the current target word embedding and the last attentional vector.

In order to meet the above two points, we make the following improvements: 1) As in Section V-A, we replaced each LSTM in both encoder and decoder to MMLSTM, and divide the dimension of each MMLSTMs layer output into 10 parts, then heuristically and manually search various combinations of Major and Minor LSTMs hidden size proportions. For the encoder MMLSTM and decoder MMLSTM in the same layer, we keep the hidden size of the encoder Major (Minor) LSTM to be the same with the decoder Major (Minor) LSTM. And the cell states are delivered from the encoder Major (Minor) LSTM to the corresponding decoder Major (Minor) LSTM. 2) In order to be compatible with input-feeding approach, the high-layer Minor LSTMs still adapt skip connection and process the input as the same as the first MMLSTM. Concretely, the input of each encoder Minor LSTM is set to the embedding of the input source sequences, the input of the decoder Minor LSTM is both the attentional vector of the last time step and the current input target word embedding. The evaluation measures are perplexity (PPL) and BLEU [49]. BLEU is a precision-based measure, which measures how well a candidate translation matches a set of reference translations by counting the percentage of n-grams in the candidate translation overlapping with the references, please see [49] for more details. The main hyper-parameters are shown in Table XI, and the results of German $\rightarrow$ English are present in Table XII.

In Table XII, the baseline model implemented by us significantly surpasses the baseline model in [47] with the help of BPE. Compare MMLSTM-NMT model with baseline model,

we can find that the former consistently outperforms the latter in both PPL and BLEU, but the parameter count of MMLSTM-NMT model only increases 1.2%, which fully verifies the effectiveness of MMLSTM on machine translation.

VI. Conclusions

From our research work, we observed that when a certain LSTM reaches a sufficiently large scale, the benefits of increasing its hidden size are very limited, which even leads to a lot of useless parameters. In this paper, we have studied the phenomenon systematically, and revealed a high correlation between the newly extended hidden states and the original hidden states, which hinders the LSTM from extracting diverse and comprehensive semantic features. We have then proposed to appropriately reduce the hidden size of the Major LSTM in the original LSTM layer, and constructed a small-scale Minor LSTM to extract a set of auxiliary features directly from the original sequence of words, so that the layer has capability to extract more diverse and comprehensive semantic features. In experiments, we verified that the MMLSTMs language model established a new state-of-the-art on Penn Treebank and Wikitext-2 datasets with perplexity as the evaluation metric. Therefore, like the existing regularization and optimization methods, the MMLSTMs can also be used as an effective technique to improve the effect of LSTM language model while controlling the parameter counts.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China (No. 2017YFB1400603).

References

[1] P. Koehn, *Statistical Machine Translation*, 1st ed. New York, NY, USA: Cambridge University Press, 2010.

[2] D. Yu and L. Deng, *Automatic Speech Recognition: A Deep Learning Approach*. Springer, 2014.

[3] J. Botha and P. Blunsom, "Compositional morphology for word representations and language modelling," in *International Conference on Machine Learning*, 2014, pp. 1899–1907.

[4] O. Press and L. Wolf, "Using the output embedding to improve language models," *arXiv preprint arXiv:1608.05859*, 2016.

[5] A. Radford, R. Jozefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *arXiv preprint arXiv:1704.01444*, 2017.

[6] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," *arXiv preprint arXiv:1708.02182*, 2017.

[7] J. Howard and S. Ruder, "Fine-tuned language models for text classification," *CoRR*, vol. abs/1801.06146, 2018. [Online]. Available: http://arxiv.org/abs/1801.06146

[8] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 933–941.

[9] N.-Q. Pham, G. Kruszewski, and G. Boleda, "Convolutional neural network language models," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1153–1162.

[10] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model." *SLT*, vol. 12, no. 234-239, p. 8, 2012.

[11] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, "Character-level language modeling with deeper self-attention," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3159–3166.

[12] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen, "Breaking the softmax bottleneck: A high-rank rnn language model," *arXiv preprint arXiv:1711.03953*, 2017.

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

[14] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[15] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[16] H. Inan, K. Khosravi, and R. Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," *arXiv preprint arXiv:1611.01462*, 2016.

[17] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models." in *AAAI*, 2016, pp. 2741–2749.

[18] K. Kawakami, C. Dyer, and P. Blunsom, "Learning to create and reuse words in open-vocabulary neural language modeling," *arXiv preprint arXiv:1704.06986*, 2017.

[19] Z. Assylbekov, R. Takhanov, B. Myrzakhmetov, and J. N. Washington, "Syllable-aware neural language models: A failure to beat character-aware ones," *arXiv preprint arXiv:1707.06480*, 2017.

[20] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *icassp*, vol. 1, 1995, p. 181e4.

[21] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech & Language*, vol. 13, no. 4, pp. 359–394, 1999.

[22] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[23] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[24] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[25] K. Zolna, D. Arpit, D. Suhubdy, and Y. Bengio, "Fraternal dropout," *arXiv preprint arXiv:1711.00066*, 2017.

[26] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[29] N. N. Schraudolph, "Centering neural network gradient factors," in *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 207–226.

[30] J. G. Zilly, R. K. Srivastava, J. Koutník, and J. Schmidhuber, "Recurrent highway networks," *arXiv preprint arXiv:1607.03474*, 2016.

[31] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[32] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," *arXiv preprint arXiv:1707.05589*, 2017.

[33] O. Kuchaiev and B. Ginsburg, "Factorization tricks for lstm networks," *arXiv preprint arXiv:1703.10722*, 2017.

[34] A. E. Orhan and X. Pitkow, "Skip connections eliminate singularities," *arXiv preprint arXiv:1701.09175*, 2017.

[35] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[36] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.

[37] E. Grave, A. Joulin, and N. Usunier, "Improving neural language models with a continuous cache," *arXiv preprint arXiv:1612.04426*, 2016.

[38] B. Krause, E. Kahembwe, I. Murray, and S. Renals, "Dynamic evaluation of neural sequence models," *arXiv preprint arXiv:1709.07432*, 2017.

[39] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[40] P. Koehn, *Statistical machine translation*. Cambridge University Press, 2009.

[41] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[42] A. G. Ororbia II, T. Mikolov, and D. Reitter, "Learning simpler language models with the differential state framework," *Neural computation*, vol. 29, no. 12, pp. 3327–3352, 2017.
[43] M. Fortunato, C. Blundell, and O. Vinyals, "Bayesian recurrent neural networks," *arXiv preprint arXiv:1704.02798*, 2017.
[44] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International Conference on Machine Learning*, 2013, pp. 1058–1066.
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[46] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.
[47] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.
[48] D. Britz, A. Goldie, M. Luong, and Q. V. Le, "Massive exploration of neural machine translation architectures," *CoRR*, vol. abs/1703.03906, 2017.
[49] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

Kai Shuang received the master's and Ph.D. degrees from the Beijing University of Posts and Telecommunications (BUPT) in 2003 and 2006, respectively. He is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology, BUPT. His research interests include deep learning, natural language processing, image processing, cloud computing, and big data technology.

Rui Li received the bachelor's degree in information and computing science from the Dalian University of Technology. He is currently a research assistant and a Doctor Degree Candidate in Beijing University of Posts and Telecommunications. His research interests include deep learning, natural language processing, language modeling and text classification.

Mengyu Gu received the bachelor's degree in telecommunications engineering from the Shanghai University. She is currently a research assistant and a Master Degree Candidate in Beijing University of Posts and Telecommunications. Her research interests include deep learning, natural language processing, language modeling and text classification.

Jonathan Loo received his M.Sc. degree in Electronics (with Distinction) and the Ph.D. degree in Electronics and Communications from the University of Hertfordshire, Hertfordshire, U.K., in 1998 and 2003, respectively. Between 2003 and 2010, he was a Lecturer in Multimedia Communications with the School of Engineering and Design, Brunel University, Uxbridge, U.K. Between June 2010 and May 2017, he was an Associate Professor in Communication Networks at the School of Science and Technology, Middlesex University, London, U.K. From June 2017, he is a Chair Professor in Computing and Communication Engineering at the School of Computing and Engineering, University of West London, United Kingdom. His recent research interests include deep learning, natural language and image processing, cloud computing, wireless/mobile communication and networks, cyber security. He has successfully graduated 18 Ph.D. students as their principal supervisor, and has co-authored more than 250 journal and conference papers in the aforementioned specialized areas. Dr. Loo has been an Associate Editor for Wiley International Journal of Communication Systems since 2011. He was the Lead Editor of the book entitled Mobile Ad Hoc Networks: Current Status".

Sen Su is a professor in Beijing University of Posts and Telecommunications. His research interests include deep learning, natural language processing, computer vision, cloud computing and big data technology. Contact him at susen@bupt.edu.cn.