Statistics and Its Interface Volume 16 (2023) 397–407

Compressing recurrent neural network models through principal component analysis

Haobo Qi, Jingxuan Cao, Shichong Chen, and Jing Zhou*

Currently, deep learning-based neural network models, such as recurrent neural networks (RNNs) and long short-term memory (LSTM) architecture, are considered state-of-the-art solutions to most of the problems associated with the effective execution of tasks in the field of natural language processing (NLP). However, a large number of parameters and significantly high memory complexity are required to ensure the effective application of such models, thereby increasing the difficulty of deploying such models in embedded systems, such as those used in mobile devices and tablets. In this study, we propose a technique for compressing RNN-based models through principal component analysis. Our proposed compression approach begins with the embedding layer, after which it progresses to the final output layer. For each target layer, we propose a principal component analysis approach for reducing the dimensions in the two-dimensional (2D) estimated weight matrix. Through this approach, we develop a reduced model structure with fewer parameters than those of the benchmark model. Additionally, our proposed approach ensures improved prediction accuracy compared to that of the benchmark model. Moreover, we propose a novel parameter-initialization method based on the score matrix of the principal component. We evaluate the effectiveness of our proposed method by conducting experiments on various NLP-related tasks, such as text classification and language translation, and datasets. The results of our experiments are significantly encouraging, as they pertain to the compression of RNN models through principal component analysis.

AMS 2000 subject classifications: Primary 68T50; secondary 68U15, 68W25.
Keywords and phrases: Model compression, Principal component analysis, RNN compression.

1. INTRODUCTION

Over the past few decades, natural language processing (NLP) has become a popular field of research. Text sequence generation [27, 35], machine translation [4, 6], and text sentiment analysis [16] are some of the popular practical approaches associated with NLP. The field of NLP mainly involves the investigation and application of various theories and methods that can be used to achieve effective communication between computers and human beings through natural language. Unlike structured data, natural languages are typical representations of unstructured textual data with specific challenges when it comes to modeling. Fortunately, current state-of-the-art deep learning-based models have become solutions for most of the challenges associated with the effective execution of tasks pertaining NLP. Various researchers have proposed various algorithms and models for addressing such problems. For instance, [17, 21] proposed N-gram-based models, [18, 37] proposed convolutional neural network (CNN)-based models, [6, 15] proposed recurrent neural network (RNN)-based models, [4] proposed an attention-based model, and [11] proposed a transformer-based model.

Among all the models mentioned above, RNN models, such as those based on long short-term memory (LSTM) architecture and those based on autoencoder frameworks, are the most studied and most popularly used models in NLP. RNN-based models demonstrate superior performance in the execution of tasks related to NLP because they consider the long-term and short-term dependencies between words. However, the effective application of such RNN-based models usually requires a large number of parameters and a significant level of memory complexity, thereby increasing the difficulty of deploying such models in embedded systems, such as those used in mobile devices and tablets, whose effectiveness is limited by memory, computing power, and battery life. Therefore, the development of methods for compressing RNN-based models could result in memory and computational cost savings. This makes RNN-based model compression a problem of great practical importance.

Consider, for example, a standard RNN model with one embedding layer, L recurrent hidden layers, and one fully connected layer. Let $x_t \in \mathbb{R}^{V_1}$ be the input text sequence, where V_1 represents the vocabulary size. Correspondingly, let $y_t \in \mathbb{R}^{V_2}$ be the output text sequence, and V_2 be the related vocabulary size. Next, define $h_t^l \in \mathbb{R}^{k_l}$ as the output, i.e., the activation, of the l-th recurrent layer at time t, where $t = 1, 2, ..., T$ and $l = 1, 2, ..., L$. Specifically, denote the output of the embedding layer as $h_t^0 \in \mathbb{R}^{k_0}$. A standard RNN model can then be formulated as follows:

$$\begin{aligned} & h_t^0 = E x_t \\ (1)\qquad & h_t^l = \sigma(W_x^l h_t^{l-1} + W_h^l h_{t-1}^l + b^l) \quad \text{for } l = 1, 2, ..., L \\ & y_t = \text{softmax}(W^{L+1} h_t^L + b^{L+1}) \end{aligned}$$

*Corresponding author.

where $\sigma(\cdot)$ is a non-linear activation function, $b^l \in \mathbb{R}^{k_l}$ and $b^{L+1} \in \mathbb{R}^{V_2}$ denote the bias vector, $E \in \mathbb{R}^{k_0 \times V_1}$ is the embedding weight matrix, $W^{L+1} \in \mathbb{R}^{V_2 \times k_L}$ is the weight matrix of the fully connected layer, $W_x^l \in \mathbb{R}^{k_l \times k_{l-1}}$ is the weight matrix for the current state, and $W_h^l \in \mathbb{R}^{k_l \times k_l}$ is the weight matrix for the historical state.

For a specific RNN model, the number of parameters depends on the embedding weight matrix E and various W matrices. Suppose that $k_i = k$ for $i = 0, 1, ..., L$; then, the total number of parameters (omit bias) in an RNN model is approximately $V_1k + V_2k + 2Lk^2$. In practice, the vocabulary size of V_1 and V_2 can be very large. Therefore, W^0 and W^{L+1} contains a great quantity of parameters, i.e., V_1k and V_2k, respectively. Moreover, to ensure satisfactory performance, the dimensions of the recurrent hidden layer k_i cannot be too small. This results in another non-ignorable cost of parameters because the number of parameters in the hidden layers is quadratic in k, i.e., $2Lk^2$. To demonstrate the issue of huge parameters in a more intuitive manner, we provide a simple example. Consider the WMT2017 news dataset [5] for Chinese–English translation. In the translation of English to Chinese, we have a vocabulary size of $V_1 = 27{,}669$ and $V_2 = 29{,}941$. Consider a simple RNN model with only one hidden layer, i.e., $L = 1$, of $k_1 = k = 1{,}024$. For the embedding layer, let the dimension of k_0 be equal to k. Therefore, the total number of parameters (omit bias) in this simple RNN model is approximately 60 million. Furthermore, such a simple RNN model requires a total of 856 MB of storage space, which is unbearable for mobile devices.

The discussion presented above suggests that it would be of great importance to develop a model compression method for RNN-based models. Specifically, in this study, we propose a compression technique based on principal component analysis (PCA). We start with the embedding layer, and the compression procedure of the subsequent layers is consistent. For the embedding layer with k_0 hidden neurons, the column vectors of weight matrix E are usually highly correlated with each other because of the relatively large k_0. Therefore, the weight matrix E can be well approximated through the linear combinations of a few basis weights. Accordingly, a novel PCA can be conducted on weight matrix E. The resulting eigenvalues are then obtained. In most cases, only a few top eigenvalues can be used to explain a significant portion of total tensor weight variability. Specifically, let $\widetilde{k}_0$ be the minimal number of top eigenvalues, such that their explained total variability exceeds a pre-specified value, e.g., 90%. E is then reduced to $\widetilde{E}$ of shape $V_1 \times \widetilde{k}_0$. Because k_0 is reduced to $\widetilde{k}_0$, the weight matrix W_x^1 also requires dimension reduction to match the output shape of the embedding layer and the hidden recurrent layer. This reduces the dimension of W_x^1 into $\widetilde{k}_0 \times k_1$. We then apply this PCA technique to the possible subsequent hidden layers till the output layer. As a result, the entire model structure can be substantially compressed, and both the number of parameters and the inference costs can be substantially reduced. Meanwhile, the prediction accuracy remains comparable compared to that of the baseline model.

Compared to previous model compression methods, the proposed method has the following contributions. First, unlike some previous methods [14], the proposed PCA method does not increase or decrease the number of layers in the RNN model. The reduced dimension is selected based on a variance contribution rate criterion. Second, the proposed method is a more general method that can be applied to both the embedding layer and each hidden layer, instead of treating them independently. Finally, we propose a new parameter initialization method. It takes the principal component score matrix as the parameter initialization value for the weight matrix. We found that compared to random initialization, the new parameter initialization strategy can make the model converge faster.

The remainder of this paper is organized as follows. In Section 2, we review the classical and most recent literature associated with RNN model compression. In Section 3, we develop a PCA-related compression method for both the text classification and language translation models. In Section 4, we present extensive experiments to evaluate the proposed method, and the results are summarized in Section 5. Section 6 concludes this paper with a brief discussion on directions for future research.

2. RELATED WORKS

Recently, various scholars have devoted significant effort to the field of model compression. According to the summary presented by [9, 12], the current model compression methods related to RNN-based models can be categorized as follows: compact model, tensor decomposition, data quantization, and weight pruning. Our proposed new method is associated with tensor decomposition. Next, we shall focus on the literature review of tensor decomposition, after which we shall briefly review the other three categories.

Tensor decomposition refers to exploiting matrix decomposition on weight tensors, such as the weight matrix of hidden layers and word embedding layers. By decomposing the weight matrix into several matrices, one actually decomposes a target layer into multiple layers, and each layer corresponds to a decomposed matrix. In the scenario of RNN-based models, researchers explored various tensor decomposition methods, such as full-rank decomposition [23, 26], singular value decomposition (SVD) [20, 22, 34], QR decomposition [2, 36], and CUR decomposition [13, 28]. Among those methods, the SVD-based method empirically results in a relatively low accuracy loss, and sometimes, it improves the accuracy. For example, [25] adopted SVD to independently compress each recurrent hidden layer into three different weight matrices. Acharya [1] applied SVD to the embedding matrix only and replaced that matrix with two low-rank matrices. Both the abovementioned methods require an intermediate layer to be inserted between layers, and they only consider either the embedding matrix or the recurrent hidden layers. It seems that none of the existing methods have considered compressing both the embedding layer and hidden layers.

Except for tensor decomposition, the other three methods are also frequently used in model compression. The compact model refers to merging or removing unimportant network modules in the original model structure and designing a new simplified model. Various improved RNN-based models belong to this category, such as the GRU model [7], the S-LSTM proposed by [32], and JANET proposed by [30]. The parameter pruning method is generally aimed at pruning the gradients of each part in the model [8, 38]. It requires a large amount of time and computational resources to iteratively find the appropriate pruning thresholds. The data quantization method has relatively poor results on RNN-based models. Both [31, 33] pointed out that data quantization could result in a significant loss of model accuracy, although it can achieve a great compression rate.

Our proposed method maintains the idea of tensor decomposition and compresses the weight matrix from the perspective of principal component analysis. In summary, we determine that the existing methods for model compression can be further improved from the following perspectives. First, the method should not increase the number of layers in the intermediate process of compression. Second, we require a more general method that can be applied to both the embedding layer and each hidden layer. Finally, there should be a new initialization strategy for the compressed model.

3. METHODOLOGY

3.1 Model compression via PCA

Consider a 2D matrix A of shape $p \times q$, such as the embedding weight matrix or weight matrices in hidden layers. The goal is to reduce the dimension of the row, i.e., reducing p to $\tilde{p}$ with $\tilde{p} \ll p$, while maintaining most of its variability. Let $A = (A_1, ..., A_q)$ with A_i denote the i-th column vector of A and $\bar{A} = (\bar{A}_1, \cdots, \bar{A}_q)^\top \in \mathbb{R}^q$ denote the column mean vector, where $\bar{A}_i = p^{-1}\sum_{j=1}^{p} A_{ij}$. Therefore, the covariance matrix of A can be defined as follows: $\Sigma = (A - \bar{A}\mathbf{1}_q^\top)(A - \bar{A}\mathbf{1}_q^\top)^\top / q \in \mathbb{R}^{p\times p}$, where $\mathbf{1}_q = (1, 1, ..., 1)^\top \in \mathbb{R}^q$ represents the vector with all its values equal to one. Σ then has the following spectral decomposition

$$\Sigma = U\Lambda U^\top = \sum_{j=1}^{p} \lambda_j u_j u_j^\top,$$

where $U = (u_1, u_2, ..., u_p)$ is an orthonormal matrix and $\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, ..., \lambda_p)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ is a diagonal matrix. Here, λ_js are the eigenvalues of Σ, and u_js are their corresponding eigenvectors. In the scenario of PCA, the accumulative variability contribution is typically used to determine the number of principal components. Specifically, for any positive integer $1 \leq n \leq p$, the accumulative variability contribution is defined as follows:

$$\eta_n = \frac{\sum_{j=1}^{n} \lambda_j}{\sum_{j=1}^{p} \lambda_j} \tag{2}$$

According to (2), η_n is a number between 0 and 1. It increases with respect to n. For a prespecified threshold η (e.g., 85% or 90%), we choose the smallest integer, i.e., $\tilde{p}$, that satisfies $\eta_{\tilde{p}} \geq \eta$ as the reduced dimension. For A with extremely large p, the distribution of $\lambda_j, j = 1, 2, ..., p$ is usually tailed. In other words, only a few top eigenvalues are very large, whereas the rest are close to zero. As a result, $\tilde{p}$ is expected to be much smaller than p. Once the reduced dimension is determined, we must calculate the reduced matrix $\widetilde{A}$. Let $U_{\tilde{p}} = (u_1, u_2, ..., u_{\tilde{p}}) \in \mathbb{R}^{p\times\tilde{p}}$ be the truncated matrix of U with its first $\tilde{p}$ column vectors. The reduced matrix $\widetilde{A}$ can then be calculated as follows: $\widetilde{A} = U_{\tilde{p}}^\top A \in \mathbb{R}^{\tilde{p}\times q}$. This completes the proposed PCA compression method.

3.2 PCA compression for text classification models

In this subsection, we apply the PCA compression technique presented above to a text classification model, which is one of the most basic and important NLP tasks. Numerous real applications, such as sentiment analysis, public opinions analysis, and spam filtering, belong to this field. Consider, for example, a simple text classification model with one embedding layer, one LSTM layer, and one fully connected layer. Let $v_t \in \mathbb{R}^{V_1}$ and $y_t \in \mathbb{R}^{V_2}$ be the input and output text sequence, respectively, where V_1 and V_2 are their corresponding vocabulary size. Denote the output of the embedding layer as follows: $x_t \in \mathbb{R}^{k_0}$. Next, define $f_t, i_t, o_t \in \mathbb{R}^{k_1}$ as the forget gate, input gate, and output gate, respectively, in the LSTM layer at time t. Furthermore, define $\tilde{c}_t, c_t, h_t \in \mathbb{R}^{k_1}$ as the current state, cell state, and output of the LSTM layer at time t, respectively. The model can then be formulated as follows:

$$\begin{aligned}
x_t &= Ev_t \\
f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \\
i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \\
o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \\
\tilde{c}_t &= \tanh(W_{ch}h_{t-1} + W_{cx}x_t + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t) \\
y_t &= \operatorname{softmax}(Wh_t + b)
\end{aligned} \tag{3}$$

where $\sigma(\cdot)$ and $\tanh(\cdot)$ denote the sigmoid and hyperbolic tangent activation functions, respectively. The sign $\odot$ indicates element-wise multiplication. To this end, we obtain the embedding weight matrix as follows: $E \in \mathbb{R}^{k_0\times V_1}$. However,

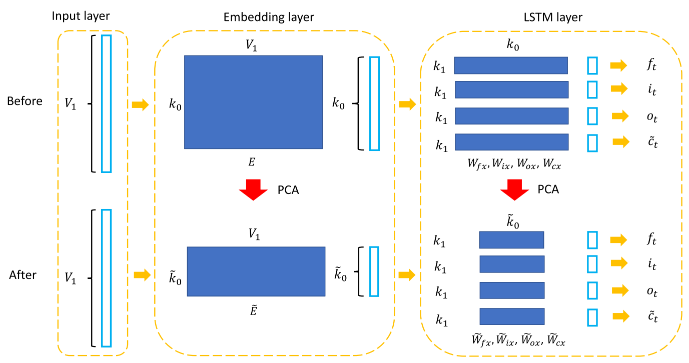

Figure 1. Illustration of PCA compression method for model (3). For simplicity, we only show the layers that corresponded with dimension reduction.

the weight matrices for the subsequent layers are somewhat complicated. Considering the forget gate f_t as an example, denote the weight matrices as follows: $[W_{fh}, W_{fx}, b_f]$, where $W_{fh} \in \mathbb{R}^{k_1 \times k_1}$ represents the historical output, $W_{fx} \in \mathbb{R}^{k_1 \times k_0}$ represents the current input, and $b_f \in \mathbb{R}^{k_1}$ represents the bias. Similarly, we obtain $[W_{ih}, W_{ix}, b_i]$ through input gate i_t, $[W_{oh}, W_{ox}, b_o]$ through input gate o_t, and $[W_{ch}, W_{cx}, b_c]$ through the current state $\tilde{c}_t$. Their dimensions are correspondingly equal to $[W_{fh}, W_{fx}, b_f]$. Finally, $W \in \mathbb{R}^{V_2 \times k_1}$ and $b \in \mathbb{R}^{V_2}$ represent the weight matrices and the bias of the fully connected layer.

For such a text classification model, we begin from the embedding layer. By applying the proposed PCA compression technique, the embedding weight matrix $E \in \mathbb{R}^{k_0 \times V_1}$ is reduced to $\widetilde{E} \in \mathbb{R}^{\tilde{k}_0 \times V_1}$. Once k_0 is reduced to $\tilde{k}_0$, we must further change the dimensions of the subsequent weight matrices in the LSTM layer because the compression in the embedding layer has a progressive effect on the next layer. Specifically, the columns of W_{fx}, W_{ix}, W_{ox} and W_{cx} should be reduced from k_0 to $\tilde{k}_0$ via PCA compression so that the dimensions of the weight matrices and input vectors match with each other. Consequently, we should obtain $\widetilde{W}_{fx}, \widetilde{W}_{ix}, \widetilde{W}_{ox}, \widetilde{W}_{cx} \in \mathbb{R}^{k_1 \times \tilde{k}_0}$. Here, $\tilde{k}_0$ is selected through a pre-specified accumulative variability contribution threshold η. The procedure explained above is summarized in Figure 1. Finally, we substitute $E, W_{fx}, W_{ix}, W_{ox}$ and W_{cx} with their corresponding reduced matrices in (3), after which we retrain the model.

3.3 PCA compression for language translation models

In the previous subsection, the simple LSTM model is adopted for text classification tasks, and the proposed PCA technique is conducted for compression. However, such a simple model may not work in language translation tasks, such as the translation of Chinese to English. This is because such tasks usually involve dealing with complex grammar and word orders. Thereafter, the RNN models for translation tasks often have significantly complex structures for exploring deeper textual information. Specifically, we adhere to the encoder–decoder framework proposed by [6], and we consider a simplified model, which uses one LSTM layer as both the encoder block and the decoder block. Let $u_t \in \mathbb{R}^{V_1}$ and $v_t \in \mathbb{R}^{V_2}$ represent the input and output text sequences, respectively, of languages A, i.e., Chinese, and B, i.e., English, where V_1 is the vocabulary size for language A and V_2 is the vocabulary size for language B. Furthermore, let the length of the input text sequence be T and the output text sequence be T'. First, we consider the encoder block, which is exactly similar to that of (3), except for the last fully connected layer. For $t = 1, 2, ..., T$, we formulate it as follows:

$$
\begin{aligned}
x_t &= E_1 u_t \\
f_t &= \sigma(W_{fh} h_{t-1} + W_{fx} x_t + b_f) \\
i_t &= \sigma(W_{ih} h_{t-1} + W_{ix} x_t + b_i) \\
o_t &= \sigma(W_{oh} h_{t-1} + W_{ox} x_t + b_o) \\
\tilde{c}_t &= \tanh(W_{ch} h_{t-1} + W_{cx} x_t + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t),
\end{aligned}
\tag{4}
$$

where the symbols and parameters are defined similarly to those in (3), and the only difference is the dimensions of different weight matrices. Specifically, the embedding weight matrix is $E_1 \in \mathbb{R}^{k_0 \times V_1}$, and the output of the embedding layer in the encoder block is $x_t \in \mathbb{R}^{k_0}$. The three gates in the LSTM are $f_t, i_t, o_t \in \mathbb{R}^{k_1}$, and the current state, cell state, and hidden state are $\tilde{c}_t, c_t, h_t \in \mathbb{R}^{k_1}$. Finally, the weight matrices of the input layers are $W_{fx}, W_{ix}, W_{ox}, W_{cx} \in \mathbb{R}^{k_1 \times k_0}$, and the weight matrices of the hidden state layers are $W_{fh}, W_{ih}, W_{oh}, W_{ch} \in \mathbb{R}^{k_1 \times k_1}$. Next, to connect the encoder block with the decoder block, Cho [6] suggested using latent state vectors. In the proposed model, the latent state vectors are defined as h_T and c_T, where h_T represents the output of the hidden state at time T, and c_T represents the current state at time T. Finally, we consider the decoder block. Because the dimensions of state units in the encoder and decoder blocks could be different, we use two weight matrices H and C to transform the dimensions of h_T and c_T. Supposing that the dimension of the hidden state in the decoder block is k_3, we obtain $H \in \mathbb{R}^{k_3 \times k_1}$ and $C \in \mathbb{R}^{k_3 \times k_1}$. To this end, we formulate the decoder block for $t = 1, 2, ..., T'$ as follows:

$$
\begin{aligned}
c_0' &= Cc_T, h_0' = Hh_T \\
y_t &= E_2 v_t \\
f_t' &= \sigma(W_{fh}' h_{t-1}' + W_{fx}' y_{t-1} + b_f') \\
i_t' &= \sigma(W_{ih}' h_{t-1}' + W_{ix}' y_{t-1} + b_i') \\
o_t' &= \sigma(W_{oh}' h_{t-1}' + W_{ox}' y_{t-1} + b_o') \\
\tilde{c}_t' &= \tanh(W_{ch}' h_{t-1}' + W_{cx}' y_{t-1} + b_c') \\
c_t' &= f_t' \odot c_{t-1}' + i_t' \odot \tilde{c}_t' \\
h_t' &= o_t' \odot \tanh(c_t'), \\
v_t &= \mathrm{softmax}(W h_t' + b)
\end{aligned} \tag{5}
$$

The series of equations (5) suggests that the decoder block also contains an LSTM layer, whose initial states of h_0' and c_0' are inherited from the hidden states of h_T and c_T in the encoder block. In the decoder block, the embedding weight matrix is $E_2 \in \mathbb{R}^{k_2 \times V_2}$, and the output of the embedding layer in the encoder block is $x_t \in \mathbb{R}^{k_2}$. The three gates in the LSTM layer are $f_t', i_t', o_t' \in \mathbb{R}^{k_3}$, and the current state, cell state, and hidden state are $\tilde{c}_t', c_t', h_t' \in \mathbb{R}^{k_3}$. The weight matrices of the input layers are $W_{fx}', W_{ix}', W_{ox}', W_{cx}' \in \mathbb{R}^{k_3 \times k_2}$, and the weight matrices of the hidden states are $W_{fh}', W_{ih}', W_{oh}', W_{ch}' \in \mathbb{R}^{k_3 \times k_3}$. Finally, the weight matrix of the fully connected layer is $W \in \mathbb{R}^{k_3 \times V_2}$.

We next explain how to conduct the proposed PCA compression method on models (4) and (5). In the previous text classification task, we did not consider compressing the matrices of $W_{fh}, W_{ih}, W_{ch}, W_{oh}$. This is because the dimensions of these hidden states are not that large. However, in the translation task, the dimensions of $W_{fh}, W_{ih}, W_{ch}, W_{oh}$ could be very large in the encoder–decoder framework, which will result in a significantly complex compression procedure. Specifically, for the encoder block, we first compress the row of the embedding weight matrix of E_1. By applying the PCA technique, E_1 is reduced to $\widetilde{E}_1 \in \mathbb{R}^{\tilde{k}_0 \times V_1}$, where $\tilde{k}_0$ is selected through a pre-specified accumulative variability contribution threshold η.

Next, we compress the rows and columns of the weight matrices $W_{fh}, W_{ih}, W_{oh}, W_{ch}$ simultaneously. Specifically, the rows of the four matrices are reduced from k_1 to $\tilde{k}_1$, where $\tilde{k}_1$ is also selected through the pre-specified threshold η. The columns of these matrices are also reduced from k_1 to $\tilde{k}_1$ in the same manner. As a result, we obtain the reduced hidden state weight matrices as follows: $\widetilde{W}_{fh}, \widetilde{W}_{ih}, \widetilde{W}_{oh}, \widetilde{W}_{ch} \in \mathbb{R}^{\tilde{k}_1 \times \tilde{k}_1}$. Therefore, to match the dimensions of the current input layer and the hidden states, we must adjust the dimension of $W_{fx}, W_{ix}, W_{ox}, W_{cx}$ into a reduced matrix of the shape $\tilde{k}_1 \times \tilde{k}_0$. As for the decoder block, a similar compression procedure can be used. For example, the dimension of the embedding weight matrix E_2 is reduced into $\widetilde{E}_2 \in \mathbb{R}^{\tilde{k}_2 \times V_2}$, the weight matrices of hidden states $W_{fh}', W_{ih}', W_{oh}', W_{ch}'$ are reduced into $\widetilde{W}_{fh}', \widetilde{W}_{ih}', \widetilde{W}_{oh}', \widetilde{W}_{ch}' \in \mathbb{R}^{\tilde{k}_3 \times \tilde{k}_3}$, and the weight matrices of the current input layers $W_{fx}', W_{ix}', W_{ox}', W_{cx}'$ are reduced into $\widetilde{W}_{fx}', \widetilde{W}_{ix}', \widetilde{W}_{ox}', \widetilde{W}_{cx}' \in \mathbb{R}^{\tilde{k}_3 \times \tilde{k}_2}$, respectively. Finally, the matrices of H and C are reduced into $\widetilde{H} \in \mathbb{R}^{\tilde{k}_3 \times \tilde{k}_1}$ and $\widetilde{C} \in \mathbb{R}^{\tilde{k}_3 \times \tilde{k}_1}$, respectively, so that the dimensions of the hidden states in the encoder and decoder blocks match with each other. The entire procedure is summarized in Figure 2.

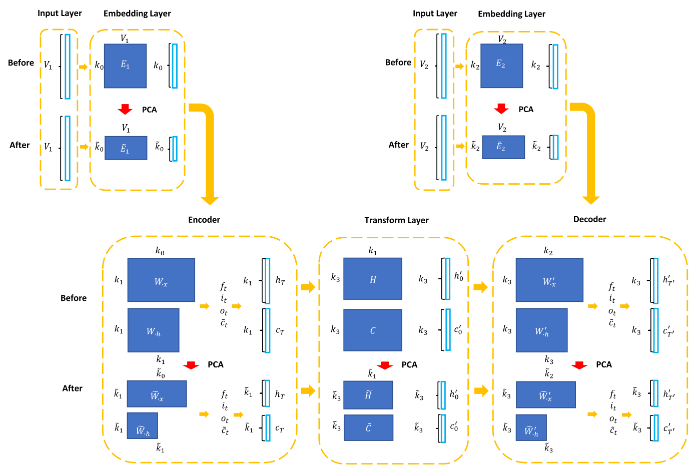

Figure 2. Illustration of PCA compression method for models (4) and (5). For convenience, we use $W_{\cdot x}$ and $W_{\cdot h}$ to represent the four weight matrices of the input and hidden states in the encoder block. Similarly, $[W_{\cdot x}', W_{\cdot h}']$, $[\widetilde{W}_{\cdot x}, \widetilde{W}_{\cdot h}]$ and $[\widetilde{W}_{\cdot x}', \widetilde{W}_{\cdot h}']$ are defined accordingly.

4. EXPERIMENTS

To empirically demonstrate the proposed compression method, we conducted various experiments on two classical tasks: text classification and language translation. In the text classification task, the proposed compression method is evaluated on both the RNN and LSTM models using the DBpedia dataset [3], whereas in the task of language translation, a model based on the encoder–decoder framework is evaluated using the WMT2017 dataset [5].

4.1 Datasets

We first introduce the datasets used in the two classical NLP tasks. They are the DBpedia dataset [3] used for text classification and the WMT2017 dataset [5] used for language translation. The DBpedia dataset is a classical textual dataset derived from 240,798 Wikipedia entries. Each of the entries is marked using multiple tags. An initial data preprocessing task is conducted for the raw dataset to delete the error words. We then select the top-five frequently appearing tags for this particular setting, which are agent, place, species, work, and event. This results in a total of 231,998 Wikipedia entries remaining. Because 81% of the entries are shorter than 1,000, we set the maximum input length as

Table 1. Parameter Information of Text Classification Model

Layer	Number of Nodes	Output Shape	Number of Parameters
Input	526,786	(256, 1000)	0
Embedding	128	(256,1000,128)	67,428,608
LSTM	128	(256,128)	131,584
Dense	64	(256,64)	8,256
Dropout	64	(256,64)	0
Dense	14	(256,14)	910
Dense(Softmax)	5	(256,5)	75

Table 2. Parameter Information of Language Translation Model

Layer	Number of Nodes	Output Shape	Number of Parameters
Input (English)	27,669	(256, 40)	0
Embedding (English)	512	(256,40,512)	14,166,528
LSTM (Encoder)	1024	(256,1024)	6,295,552
Dense	1024	(256,1024)	1,049,600
Dense	1024	(256,1024)	1,049,600
Input (Chinese)	29,941	(256, 39)	0
Embedding (Chinese)	512	(256,39,512)	15,329,792
LSTM (Decoder)	1024	(256,39,1024)	6,295,552
Dense (Softmax)	29,941	(256,39,29941)	30,689,525

1,000 and the vocabulary size as 526,786. The other dataset is WMT2017, which is a famous machine translation benchmark dataset. It contains approximately 250,000 sentences for both Chinese and English comments for news. A similar data preprocessing task is conducted, and the comments longer than 40 are deleted. This results in a total of 87,072 pairs of comments left. The vocabulary sizes for Chinese and English are 29,941 and 27,669, respectively.

4.2 RNN-based models

In the task of text classification, we follow the settings presented in the work of [16], and we adopt the RNN and LSTM architecture. Specifically, the model contains seven layers, including one input layer with a dimension of 1,000, one embedding layer with 128 hidden nodes, one simple RNN layer or LSTM layer with 128 hidden nodes, two fully connected layers with dimensions of 64 and 14, respectively, one dropout layer, and one softmax layer. The detailed parameter information is listed in Table 1. As shown in Table 1, the dimension of the embedding weight matrix is extremely high. It contains 67,428,608 parameters, which explains 99.79% of the parameters in the entire model. This suggests that the proposed compression strategy for text classification is reasonable.

Next, for the translation task, we follow the settings presented in the work of [6], and we adopt an encoder–decoder model structure. The model contains two blocks and a total of nine layers. Specifically, the encoder block includes one input layer for English sequences with an input length of 40, one embedding layer with 512 hidden nodes, and one LSTM layer with 1,024 hidden states. The decoder block contains one input layer for Chinese sequences with an input length of 39, one embedding layer with 512 hidden nodes, and one LSTM layer with 1,024 hidden states. Two fully connected layers are set between the two blocks mentioned above. This approach aims to transform the dimensions of hidden states in the LSTM encoder layer to the dimensions of hidden states in the LSTM decoder layer. Finally, a fully connected layer with softmax activation is used after the decoder layers. The parameter information of the entire model is summarized in Table 2.

As shown in Table 2, the number of parameters in the embedding layer is still significantly large. However, it only accounts for 18.92% of all the model parameters, which is smaller than the proportion in the classification task. This is because the number of parameters in other layers, such as the encoder and decoder layers, is also significantly large. This indicates that the compression method for the translation task will be more complicated than that for the classification task.

4.3 Performance measures

Following existing literature obtained from the works of [10, 24], we consider three metrics for evaluating compression performance. Specifically, we calculate (i) the parameter reduction ratio (Prr), (ii) the FLOPs reduction ratio (Frr), and we monitor the (iii) out-of-sample prediction accuracy (Acc).

4.4 Implementation details

Although the proposed PCA compression method is conceptually simple, its practical implementation is not trivial.

It involves two important tuning parameters: the variability threshold η and the parameter initialization value. The variability threshold η is used to determine the reduced dimensions of weight matrices in neural networks. Specifically, the smaller the η value, the higher the compression rate that can be obtained, e.g., a slimmer model structure. In our experiments, we set $\eta = 0.4, 0.6, 0.85, 0.9, 0.95$. Another tuning parameter that could influence the results of the training process is the choice of initialization for the reduced model. Generally speaking, dimension reduction techniques pay a cost of prediction accuracy, which can be partly recovered by retraining the reduced model. Therefore, initialization methods might result in different final prediction accuracies. In our experiments, we consider the he-normal initialization method and the reduced PCA matrices as the initialization parameters, respectively. This results in a total of 10 different combinations for the classification and translation tasks.

For both the DBpedia and WMT2017 datasets, 80% of the data is used as the training set, and the remaining 20% is used as the test set. The models used for the classification task are trained using the RMSprop [29] algorithm with an initial learning rate of $\alpha = 10^{-4}$ and a decay rate of $\rho = 0.9$. For the translation task, an Adam [19] algorithm with an initial learning rate of $\alpha = 10^{-3}$ and an exponential decay rate of $\beta_1 = 0.9$, $\beta_2 = 0.98$ is adopted. The other settings are similar. First, the batch size is set to 256 for all the working models. Second, the weight decay rate is set to 10^{-5} with an l_2-norm regularizer for all models. Finally, a total of 50 epochs are conducted for each working model, and we record the best prediction accuracy on the validation dataset as the Acc performance.

5. RESULTS

5.1 Compression results for the text classification task

In this subsection, we investigate the impact of PCA compression on the two proposed text classification models. Three measures are used to evaluate the finite sample performance: prediction accuracy (Acc), the parameter reduction ratio (Prr), and the FLOP reduction ratio (Frr). We list the detailed compression results in Table 3, and we present the training performance in Figures 3-4.

For illustration purposes, the simple RNN and LSTM models described in Section 4.2 are considered the baseline models. Next, we investigate a total of five different values of the variability threshold $\eta = (0.4, 0.6, 0.85, 0.9, 0.95)$. For each η, we consider two initialization strategies: the he-normal and the reduced PCA matrices. This results in a total of ten combinations for the RNN and LSTM models, respectively. We then obtain the following conclusions from Table 3. First, we find that as long as the variability threshold value η is set to be larger than 0.85, insignificant prediction accuracy would be sacrificed. Instead, the best prediction accuracy can be even higher than that of the baseline model. However, we can observe a significant drop in accuracy if η is set too low. For example, the accuracy drops by approximately 3.5% for the RNN model with $\eta = 0.4$. Similar results can be observed in the LSTM model. Second, we establish that different initialization strategies have an insignificant effect on prediction accuracy. The prediction accuracy achieved through the proposed PCA initialization approach is slightly better than that of the he-normal initialization approach when the value of η is set higher than 0.85.

We then present the detailed training performance in Figures 3-4. For each figure, the left panel shows the accuracy curves with different variability threshold values of η (including the baseline model) under the PCA initialization strategy, and the right panel shows those of the curves under the he-normal initialization approach. We establish that when $\eta \geq 0.85$, the reduced models using the PCA-based initialization strategy converge faster than the models using the he-normal initialization approach. This suggests that the PCA-based initialization strategy offers a satisfactory starting point during the retraining process. It not only reduces the training cost but also avoids sacrificing the prediction accuracy.

5.2 Compression results for the language translation task

In this subsection, we investigate the impact of PCA-based compression on the translation task using the encoder–decoder model. The three measurements, Acc, Frr, and Prr, are summarized in Table 4. The detailed training process is displayed in Figure 5. As shown in Table 4, as long as the variability threshold value η is set to be larger than 0.85, the Acc values of the reduced models decrease by less than 1%. Compared to the results of the classification task, the accuracy drops significantly under similar values of η. Specifically, the prediction accuracy decreases by approximately $1\%, 8\%$, and 12% for $\eta = 0.85, 0.6$ and 0.4. Additionally, we establish that different initialization settings have a significant effect on prediction accuracy. The prediction accuracy achieved using the proposed PCA-based initialization strategy is better than that achieved through the he-normal initialization approach.

Figure 5 shows the prediction accuracy curves of the encoder–decoder framework with different settings of η and initialization strategies. The left panel shows the accuracy curves with different variability thresholds η (including the baseline model) using the PCA-based initialization strategy, and the right panel shows the accuracy curves under the he-normal initialization approach. We establish that the reduced models achieved through the PCA-based initialization strategy generally converge faster than the models using the he-normal initialization approach. This suggests that the PCA-based initialization strategy offers a satisfactory starting point during the retraining process. It not only reduces the training cost but also avoids sacrificing prediction accuracy.

Table 3. Compression Results of Text Classification Models

Model	η_0	Initialization	Acc	Frr(%)	Prr(%)
RNN	baseline		0.9866		
	0.4	PCA	0.9405	99.20	99.18
		He normal	0.9530	99.20	99.18
	0.6	PCA	0.9831	98.41	98.40
		He normal	0.9838	98.41	98.40
	0.85	PCA	0.9873	95.29	95.28
		He normal	0.9836	95.29	95.28
	0.9	PCA	0.9863	91.38	91.37
		He normal	0.9868	91.38	91.37
	0.95	PCA	0.9865	79.67	79.66
		He normal	0.9846	79.67	79.66
LSTM	baseline		0.9863		
	0.4	PCA	0.9529	99.15	99.11
		He normal	0.9562	99.15	99.11
	0.6	PCA	0.9845	98.37	98.33
		He normal	0.9841	98.37	98.33
	0.85	PCA	0.9872	97.59	97.55
		He normal	0.9834	97.59	97.55
	0.9	PCA	0.9873	96.80	96.77
		He normal	0.9855	96.80	96.77
	0.95	PCA	0.9874	95.24	95.21
		He normal	0.9874	95.24	95.21

Table 4. Compression Results of the Language Translation Model

η_0	Initialization	Acc	Frr(%)	Prr(%)
baseline		0.3236		
0.4	PCA	0.2164	98.70	98.66
	He normal	0.1905	98.70	98.66
0.6	PCA	0.2501	95.57	95.45
	He normal	0.2363	95.57	95.45
0.85	PCA	0.3205	69.67	69.07
	He normal	0.3118	69.67	69.07
0.9	PCA	0.3294	56.90	56.16
	He normal	0.3225	56.90	56.16
0.95	PCA	0.3316	38.10	37.36
	He normal	0.3265	38.10	37.36

6. CONCLUSIONS

In this study, we propose a novel PCA-based method for compressing RNN-based models. The proposed method focuses on weight matrices of both the embedding layer and the hidden layers. The entire compression procedure is conducted sequentially. Although the proposed PCA-based method is associated with the tensor decomposition method, it admits several distinctive characteristics. First, compared to the current tensor decomposition methods, our proposed method does not introduce additional intermediate linear layers. Second, our proposed method is a more general approach that can be applied to both the embedding layer and each hidden layer, instead of approaching them independently. Finally, numerical experiments show that using PCA-based score matrices as the parameter initialization strategy yields improved results. This approach accelerates the retraining process with an insignificant loss of accuracy. In conclusion, in this study, we present various interesting topics for future research. First, our proposed PCA-based method only considers the weight matrix, e.g., W, whereas the layer response is not used effectively. This attribute makes our proposed method less sensitive to the data. This should be an interesting topic for future research. Second, many state-of-the-art methods have been developed in the field of RNN model compression. Future studies should investigate approaches for combining such methods to further improve empirical performance. Finally, our proposed PCA-based approach involves the tuning parameter, i.e., η, to be learned for different models and datasets. This is a time-consuming task. It would be of great interest to develop an algorithm that can ensure the detection of the best tun-

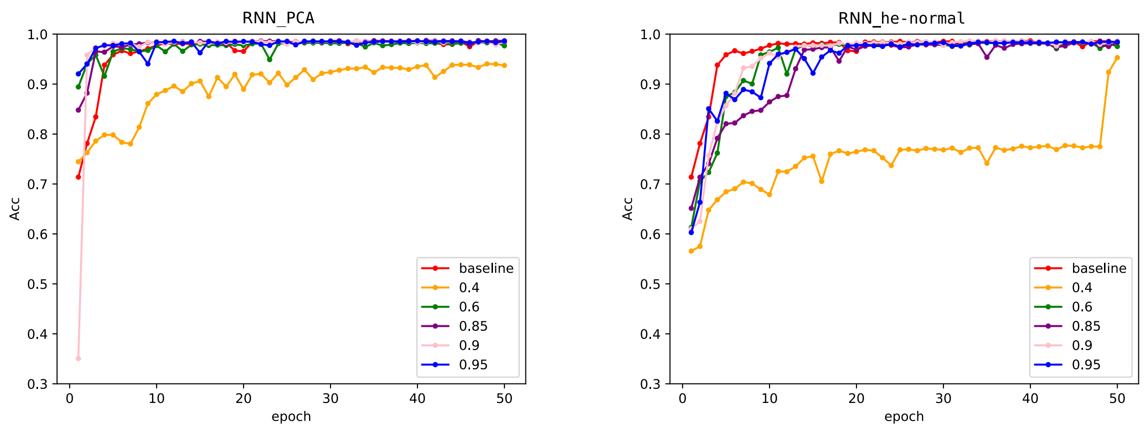

Figure 3. Left panel: prediction accuracy curves of RNN models using reduced PCA-based matrix initialization. Right panel: prediction accuracy curves of RNN models using the he-normal initialization approach.

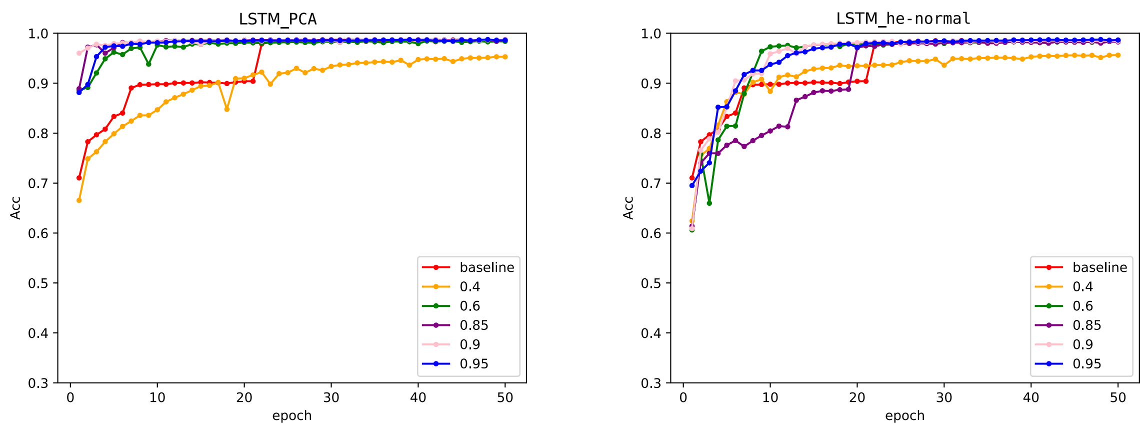

Figure 4. Left panel: prediction accuracy curves of LSTM models using reduced PCA-based matrix initialization. Right panel: prediction accuracy curves of LSTM models using the he-normal initialization approach.

ing parameter. This is also a crucial direction for future research.

ACKNOWLEDGEMENTS

This research was supported by the Fundamental Research Funds for the Central Universities,and the Research Funds of Renmin University of China, No.21XNA027.

Received 24 November 2021

REFERENCES

[1] Acharya, A., Goel, R., Metallinou, A. and Dhillon, I. (2019). Online embedding compression for text classification using low rank matrix factorization. *Proceedings of the AAAI Conference on Artificial Intelligence* **33** 6196–6203.

[2] Aizenberg, I., Luchetta, A. and Manetti, S. (2012). A modified learning algorithm for the multilayer neural network with multi-valued neurons based on the complex qr decomposition. *Soft Computing* **16** 563–575. MR3024805

[3] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *The Semantic Web* 722–735. Springer.

[4] Bahdanau, D., Cho, K. and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In *Computing Research Repository (CoRR)*.

[5] Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia, L. and Turchi, M. (2017). Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation* **2** 169–214. Association for Computational Linguistics.

[6] Cho, K., Bart van Merrienboer, Gülçehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014). Learning phrase representations using RNN Encoder–Decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* 1724–1734.

[7] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[8] Dai, X., Yin, H. and Jha, N. K. (2019). Grow and prune compact, fast, and accurate lstms. *IEEE Transactions on Computers* **69**

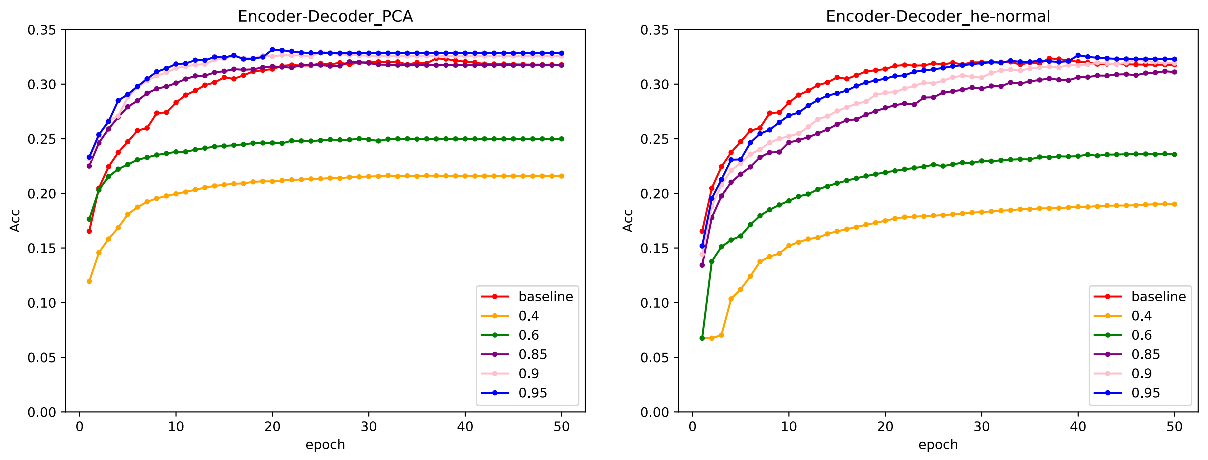

Figure 5. Left panel: prediction accuracy curves of encoder–decoder-based models using reduced PCA-based matrix initialization. Right panel: prediction accuracy curves of encoder–decoder-based models models using the he-normal initialization approach.

441–452. MR4086683

[9] Deng, L., Li, G., Han, S., Shi, L. and Xie, Y. (2020). Model compression and hardware acceleration for neural networks: A comprehensive survey. In *Proceedings of the IEEE* **108** 485–532.

[10] Denil, M., Shakibi, B., Dinh, L., Ranzato, M. A. and Freitas, N. D. (2013). Predicting parameters in deep learning. *arXiv preprint arXiv:1306.0543*.

[11] Devlin, J., Chang, M. W., Lee, K. and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019* 4171–4186.

[12] Geng, L. and Niu, B. (2020). Survey of deep neural networks model compression. *Journal of Frontiers of Computer Science and Technology* **14** 1441–1455. MR3753802

[13] Gittens, A. and Mahoney, M. W. (2016). Revisiting the nyström method for improved large-scale machine learning. *The Journal of Machine Learning Research* **17** 3977–4041. MR3543523

[14] Grachev, A. M., Ignatov, D. I. and Savchenko, A. V. (2019). Compression of recurrent neural networks for efficient language modeling. *Applied Soft Computing* **79** 354–362.

[15] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation* **9** 1735–1780.

[16] Hossain, E., Sharif, O., Hoque, M. and Sarker, I. (2020). SentiLSTM: A deep learning approach for sentiment analysis of restaurant reviews. In *Proceedings of 20th International Conference on Hybrid Intelligent Systems.*

[17] Jelinek, F. and Mercer, R. I. (1980). Interpolated estimation of markov source parameters from sparse data. In *Proceedings, Workshop on Pattern Recognition in Practice.*

[18] Yoon Kim (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* 1746–1751.

[19] Kingma, D. P. and Ba, J. L. (2015). Adam: a method for stochastic optimization. In *International Conference on Learning Representations* 1–13.

[20] Virginia Klema and Alan Laub (1980). The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control* **25** 164–176. MR0567374

[21] Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing* **1** 181–184.

[22] Li, C. H. and Park, S. C. (2007). Neural network for text classification based on singular value decomposition. In *7th IEEE International Conference on Computer and Information Technology (CIT 2007)* 47–52. MR2321027

[23] Liu, Y., Yang, S., Wu, P., Li, C. and Yang, M. (2015). l_1-norm low-rank matrix decomposition by neural networks and mollifiers. *IEEE transactions on neural networks and learning systems* **27** 273–283. MR3463943

[24] Molchanov, P., Tyree, S., Karras, T., Aila, T. and Kautz, J. (2015). Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

[25] Prabhavalkar, R., Alsharif, O., Bruguier, A. and McGraw, L. (2016). On the compression of recurrent neural networks with an application to lvcsr acoustic modeling for embedded speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 5970–5974.

[26] Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E. and Ramabhadran, B. (2013). Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing* 6655–6659.

[27] Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. and Pineau, J. (2015). Building end-to-end dialogue systems using generative hierarchical neural network models. *AAAI Press.*

[28] Thurau, C., Kersting, K. and Bauckhage, C. (2012). Deterministic cur for improved large-scale data analysis: An empirical study. In *Proceedings of the 2012 SIAM International Conference on Data Mining* 684–695.

[29] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning* 26–31.

[30] Jos Van Der Westhuizen and Joan Lasenby (2018). The unreasonable effectiveness of the forget gate. *arXiv preprint arXiv:1804.04849*.

[31] Wang, P., Xie, X., Deng, L., Li, G., Wang, D. and Xie, Y. (2018). Hitnet: Hybrid ternary recurrent neural network. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* 602–612.

[32] Wu, Z. and King, S. (2016). Investigating gated recurrent networks for speech synthesis. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 5140–5144.

[33] Xu, C., Yao, J., Lin, Z., Ou, W., Cao, Y., Wang, Z. and Zha, H. (2018). Alternating multi-bit quantization for recurrent neural

networks. *arXiv preprint arXiv:1802.00150*.

[34] Xue, J., Li, J., Yu, D., Seltzer, M. and Gong, Y. (2018). Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 6359–6363.

[35] Yang, P., Sun, X., Li, W., Ma, S., Wu, W. and Wang, H. (2018). SGM: sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING* 3915–3926.

[36] Yu, M., Lin, Z., Narra, K., Li, S., Li, Y., Kim, N. S., Schwing, A., Annavaram, M. and Avestimehr, S. (2018). Gradiveq: Vector quantization for bandwidth-efficient gradient aggregation in distributed CNN training. *arXiv preprint arXiv:1811.03617*.

[37] Zhao, H., Lu, Z. and Poupart, P. (2015). Self-adaptive hierarchical sentence model. In *Proceedings of International Joint Conferences on Artificial Intelligence*.

[38] Zhu, M., Clemons, J., Pool, J., Rhu, M., Keckler, S. W. and Xie, Y. (2018). Structurally sparsified backward propagation for faster long short-term memory training. *arXiv preprint arXiv:1806.00512*.

Haobo Qi
Guanghua School of Management
Peking University
China
E-mail address: qihaobo_gsm@pku.edu.cn

Jingxuan Cao
Center for Applied Statistics
School of Statistics
Renmin University of China
China
E-mail address: caojx@ruc.edu.cn

Shichong Chen
Center for Applied Statistics
School of Statistics
Renmin University of China
China
E-mail address: csc789654123@163.com

Jing Zhou
Center for Applied Statistics
School of Statistics
Renmin University of China
China
E-mail address: jing.zhou@ruc.edu.cn