# Dynamically Hierarchy Revolution: DirNet for Compressing Recurrent Neural Network on Mobile Devices

**Jie Zhang**[1*], **Xiaolong Wang**[2†], **Dawei Li**[2], **Yalin Wang**[1]
[1]Arizona State University, Tempe, AZ, USA
[2]Samsung Research America, Mountain View, CA, USA
{jiezhang.joena, ylwang}@asu.edu, {xiaolong.w, dawei.l}@samsung.com

## Abstract

Recurrent neural networks (RNNs) achieve cutting-edge performance on a variety of problems. However, due to their high computational and memory demands, deploying RNNs on resource constrained mobile devices is a challenging task. To guarantee minimum accuracy loss with higher compression rate and driven by the mobile resource requirement, we introduce a novel model compression approach DirNet based on an optimized fast dictionary learning algorithm, which 1) dynamically mines the dictionary atoms of the projection dictionary matrix within layer to adjust the compression rate 2) adaptively changes the sparsity of sparse codes cross the hierarchical layers. Experimental results on language model and an ASR model trained with a 1000h speech dataset demonstrate that our method significantly outperforms prior approaches. Evaluated on off-the-shelf mobile devices, we are able to reduce the size of original model by eight times with real-time model inference and negligible accuracy loss.

## 1 Introduction

Deep learning technique is becoming the dominant force of recent breakthroughs in artificial intelligence area [Geng *et al.*, 2017]. Recently, Recurrent Neural Networks (RNNs) have gained wide attentions with dramatic performance improvements in sequential data modeling, e.g., automatic speech recognition (ASR) [Sak *et al.*, 2014], language modeling [Mikolov *et al.*, 2011], image captioning [Vinyals *et al.*, 2017], neural machine translation [Cho *et al.*, 2014], etc. Recently, due to the success of RNN-ASR [Lei *et al.*, 2013], personal assistant system (e.g., Amazon's Alexa, Apple's Siri, Google Now, Samsung's Bixby, Microsoft's Cortana) has become a standard system configuration of smartphones.

Generally, the trained deep learning models are deployed on the cloud which requires strict Internet connection and also may compromise user privacy. Therefore, there is a high demand to deploy such RNNs with millions of parameters on mobile devices. Deep model compression techniques have been proposed to solve the mobile deployment problem. Specifically for RNNs model compression, a pruning based method [Narang *et al.*, 2017] was introduced which progressively prunes away small parameters using a monotonically increasing threshold during training. Another representative method is to apply singular value decomposition (SVD) low rank approach for decomposing RNNs weight matrices [Prabhavalkar *et al.*, 2016]. Although some good results for compressing RNNs have been reported, these works fail to consider the hierarchical changes of weight matrices. In addition, none of the existing methods can dynamically adjust the compression rate according to the requirement of the deployed mobile devices.

In this work, we propose a dynamically hierarchy revolution (DirNet) to address the problems in existing RNNs model compression methods (Fig. 1). To further compress the model with considering different degrees of redundancies among layers, we exploit a novel way of dynamically mining dictionary atoms from original network structures without manual setting the compression rate for different layers. Our approach achieves significant improvement in model compression compared to previous approaches. It also has much higher re-training's convergence rate and lower accuracy loss compared to the state-of-the-art. In addition, considering different functionalities among hierarchical hidden layers, we come up with the idea of adaptively adjusting sparsity of different hidden layers in RNNs. This can improve the precise structure approximation to the original network with minimizing the performance drop.

Our most significant contributions can be summarized into threefold: Firstly, this is the first work dynamically exploring dictionary learning in weight matrix of both RNN and LSTM that jointly compresses both hidden layer and inter layer to reconstruct original weight matrices. Secondly, our approach is a dynamic model compression algorithm. It can dynamically mine the dictionary atoms of the projection dictionary matrix within the layer to better learn a common codebook representation across inter-layer and recurrent layer. This can find the optimal numbers of neurons among layers to better control the degree of compression with negligible accuracy loss. Thirdly, given a hierarchical RNN structure, our approach can adaptively set various sparsities of sparse codes

*Work was done during the first author's internship at Samsung Research America.
†Corresponding author.

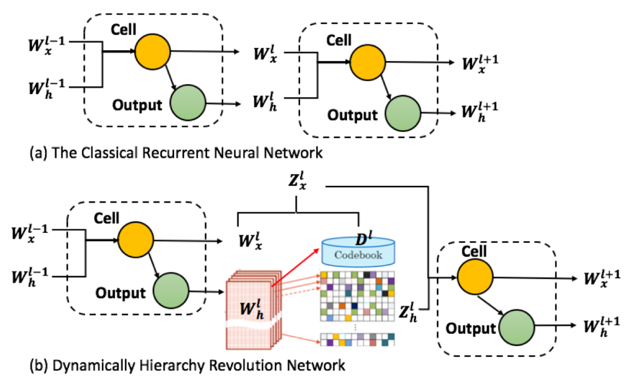

Cell
Output
(a) The Classical Recurrent Neural Network
$Z_x^l$
$D^l$
Codebook
$Z_h^l$
(b) Dynamically Hierarchy Revolution Network

Figure 1: (a) The initial model is compressed by (b) jointly dynamically adjust sparsity of recurrent and inter-layer matrices, using a shared projection dictionary.

for each layer to minimize the performance drop for the compressed network. The experimental results demonstrate that DirNet achieves significant improvement in terms of compression scale and speedup rate compared to the state-of-the-art approaches.

## 2 Related Work

In general, we can summarize existing RNN compression methods into three categories: Pruning, low-rank approximation, and knowledge distillation. The first pruning work was proposed by Han et al. [2015b] where a hard threshold was applied as the pruning criterion to compress the deep neural network. Li et al. [2017] discussed the optimization of non-tensor layers such as pooling and normalization without tensor-like trainable parameters for deep model compression. Xue et al. [2013] used low-rank approximation to reduce parameter dimensions in order to save storage and reduce time complexity. Denile et al. [2013] showed that the given neural network can be represented by a small number of parameters. Sainath et al. [2013] reduced the number of model parameters by applying the low-rank matrix factorization to simplify the neural network architecture.

Hinton et al [2015] learned a small student network to mimic the original teacher network by minimizing the loss between the student and teacher output. Kim et al. [2016] integrated knowledge distillation to approximately match the sequence-level distribution of the teacher network. Chen et al. [2017] introduced a knowledge of cross sample similarities for model compression and acceleration.

In addition, there are some other compression methods. Low-precision quantization [Xu *et al.*, 2018] is a scalar-level compression method without considering the relation among learned parameters in layers. Compared to these works, our approach dynamically adjusts the compression rate across layers and explores the sparsity of weight matrices to compress RNN. It achieves better compression rate with less accuracy loss compared with above methods.

Recently, Han et al. [2017] used a pruning method to compress the LSTM on speech recognition problem and it reduced the parameters of the weight matrix to 10% and achieved 5X compression rate (each non-zero element is represented by its value and index in the matrix). However, DirNet considers the sparsity of various matrices and dynamically adjusts the sparsity based on the feature of models and gets a better compression rate. Our approach can compress the model by around 8X with negligible performance loss.

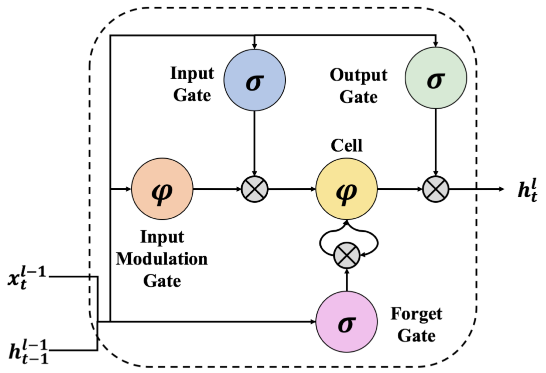


Figure 2: A graphical representation of LSTM memory cells used in this work.

## 3 Methods

In this section, we first introduce the background of RNNs. Then, we present the basic sparse coding based RNN compression method without considering the mobile requirement and the hierarchical network structure. Further, we present the proposed approach DirNet.

### 3.1 RNN Background

Let $t = 0, 1, \cdots, T$ denotes time steps and $l = 1, 2, \cdots, L$ denotes the hidden layers in RNN. At time $t$, $h_t^l \in \mathbb{R}^{N^l}$ denotes the activations of the $l$-th hidden layer with $N^l$ node. Therefore, the inputs to this layer at time $t$ are donated by $h_t^{l-1} \in \mathbb{R}^{N^{l-1}}$. Then, we can define the output activations of the $l$-th and $(l+1)$-th layers in a classical RNN:

$$h_t^l = \sigma(W_x^{l-1} h_t^{l-1} + W_h^l h_{t-1}^l + b^l), \tag{1}$$

$$h_t^{l+1} = \sigma(W_x^l h_t^l + W_h^{l+1} h_{t-1}^{l+1} + b^{l+1}), \tag{2}$$

where $\sigma(\cdot)$ denotes a non-linear activation function and $b^l \in \mathbb{R}^{N^l}$ and $b^{l+1} \in \mathbb{R}^{N^{l+1}}$ represent bias vectors. $W_x^l \in \mathbb{R}^{N^{l+1} \times N^l}$ and $W_h^l \in \mathbb{R}^{N^l \times N^l}$ denote inter-layer and recurrent weight matrices, respectively.

Different from traditional RNNs, LSTM contains specialized cells called *memory cell* in the recurrent hidden layer. Empirically, LSTM has a memory cell for storing information for long periods of time, we use $c_t^l \in \mathbb{R}$ to denote the long-term memory. LSTM can decide to overwrite the memory cell, retrieve it, or keep it for the next time step [Zaremba *et al.*, 2014]. In this paper, our LSTM architecture is the same as [Graves *et al.*, 2013]. The general structure is illustrated in Figure 2.

$$\begin{pmatrix} i \\ o \\ f \\ g \end{pmatrix} = \begin{pmatrix} sigmod \\ sigmod \\ sigmod \\ \tanh \end{pmatrix} (T_{n,4n}, T_{n,4n}) \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^{l} \end{pmatrix}, \quad (3)$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g, h_t^l = o \odot \tanh(c_t^l),$$

where $i, o, f$ and $g$ are input gate, output gate, forget gate and input modulation gate, respectively. And $c$ denotes the memory cells and all gates are the same size as the memory cells. $\odot$ is the element-wise product of the vectors. $n$ and $4n$ denote the dimension of one gate and four gate, respectively.

## 3.2 Sparse Coding based Model Compression

As indicated above, one direct way to compress RNN is to sparsify weight matrix $W_x^l$ and $W_h^l$. Given a weight matrix $W \in \mathbb{R}^{p \times t}$, each $w_i \in \mathbb{R}^p$. Sparse coding aims to learn a dictionary $D$ where $D \in \mathbb{R}^{p \times m}$ and a sparse code matrix $Z \in \mathbb{R}^{m \times t}$. The original matrix $W$ is modeled by a sparse linear combination of $D$ and $Z$ as $W \approx DZ$. We can formulate the following optimization problem:

$$\min_{D \in \Psi, z_i \in \mathbb{R}^l} f(D, Z) = \frac{1}{2}||W - DZ||_F^2 + \lambda||Z||_1, \quad (4)$$

where $\Psi = \{D \in \mathbb{R}^{p \times m} : \forall j \in 1, \cdots, m, ||d_j||_2 \leq 1\}$, $d_j$ denotes the $j$th column of $D$ and $Z = (z_1, \cdots, z_t)$. $\lambda$ is the positive regularization parameter and $||W||_F$ denotes the Frobenius norm of the matrix.

For RNNs, we can compress simultaneously the weight matrices of both recurrent layer and inter-layer (from Eq. (1) or Eq. (2)). Such joint compression can be achieved by a projection matrix [Sak *et al.*, 2014], denoted by $D^l \in \mathbb{R}^{p^l \times N^l}$ ($p^l < N^l$), such that, $W_h^l = D^l Z_h^l$ and $W_x^l = D^l Z_x^l$. Therefore, we can incorporate this idea into Eq. (1) and Eq. (2), then we get the followings:

$$\min_{D^l, Z_h^l} ||W_h^l - D^l Z_h^l|| + \lambda_1 |Z_h^l|_1, \quad (5)$$

$$h_t^l = \sigma(W_x^{l-1} h_t^{l-1} + D^l Z_h^l h_{t-1}^l + b^l), \quad (6)$$

$$\min_{Z_x^l} ||D^l Z_x^l - W_x^l||_2^2 + \lambda_2 ||Z_x^l||_1, \quad (7)$$

$$h_t^{l+1} = \sigma(D^l Z_x^l h_t^l + W_h^{l+1} h_{t-1}^{l+1} + b^{l+1}), \quad (8)$$

where $Z_h^l \in \mathbb{R}^{p^l \times N^l}$ and $Z_x^l \in \mathbb{R}^{p^l \times N^{l+1}}$. Therefore, we can calculate the compression rate by $\frac{N^l \times N^l + N^{l+1} \times N^l}{p^l \times N^l + p^l \times N^l + p^l \times N^{l+1}}$.

However, a problem exists in this deep compression model is that the compression rate and the sparsity are manually set which limits the compression space. It would be much more efficient and promising to automatically set the compression rate and sparsity based on the understanding of deep neural network's structure. Also we should consider minimizing the training time during the model compression which may take weeks or even months, especially for RNN based sequential models. In this paper, we propose DirNet, which can dynamically adjust the sparsity of Eq. (5) and Eq. (7), to meet this requirement.

**Algorithm 1:** DirNet

```
input  : Inter-layer matrix $W_h$ and recurrent matrix $W_x$,
         shift operators $\triangle$, $K, M, \lambda_1 = 0.1, \lambda_2$
output: Compressed weight matrices $Z_h, Z_x$ and
         projection matrix $D$
1  begin
2      for $l = 1 \to L$ do
3          Initialize $D^l$ and $Z_h^l$
4          repeat
5              Update sparse code $z_i^k \leftarrow CD(d_i^k, z_i^{k-1}, w_i)$
6              Update dictionary
                 $d_k \leftarrow \arg\min_{\{d_i\}_{i=1}^K} \sum_{j=1}^M \frac{1}{2}||x_j - \sum_{i=1}^K z_{ij}\delta_{ij}(d_i)||_2^2, s.t. ||d_k||_2 = 1$
7          until convergence;
8          $h_t^l = \sigma(W_x^{l-1} h_t^{l-1} + D^l Z_h^l h_{t-1}^l + b^l)$
9          while model is unempty do
10             $t = 0$
11             repeat
12                 $t = t + 1$
13                 Fix $\Theta_t^l$ update $z_t^l$ according to Eq. 14
14                 Fix $z_t^l$ update $\Theta_t^l$ according to Eq. 15
15             until Convergence;
16             $\Theta_0^{\tau+1} = \Theta_t^\tau$
17             $\tau = \tau + 1$
18         $h_t^{l+1} = \sigma(D^l Z_x^l h_t^l + W_h^{l+1} h_{t-1}^{l+1} + b^{l+1})$
19     return $Z_h, Z_x, D$;
```

DirNet is mainly composed of two steps and depicted graphically in Fig. 1: In Step 1, a dynamically adaptive dictionary learning (Sec. 3.3) is proposed to adapt the atoms in shared dictionary among inter-layer and recurrent layer to adjust the sparsity (compression rate and accuracy) within the layer (Eq. (5)). In Step 2, we propose an adaptive sparsity learning approach with considering the network's hierarchical structure (Sec. 3.4) to optimize the $Z$ and sparsity parameter $\Theta$ hierarchically with an appropriate sparsity degree. The sparsity (compression rate and accuracy) achieved by different layers is depending on the number of neurons and the architecture of the specific layer. In general, the non-zero values in $Z$ are ten to twenty percent of the original weight matrices $W$ in our work. From this, we can find the proposed approach adaptively set various sparsity among the hierarchical RNN architecture on Eq. 7 instead of using a fixed fraction function of explained variance as advocated in Prabhavalkar et al. [2016]. We summarize the key steps of DirNet in Algorithm 1.

## 3.3 Dynamically Adaptive Dictionary Learning

It is an essential step to dynamically adjust the dimension of the projection matrices $D^l$ to get the optimal compression rate. We introduce a shift operation $\triangle$ on atoms to better adjust the compression rate. Given a set of shift operations $\triangle$ which contains only small shifts relative to the size of the (time window), for every $j$ there exist coefficients $z_{ij} \in \mathbb{R}$

and shift operators $\delta_{ij} \in \triangle$ [Hitziger *et al.*, 2013], such that $w_j = \sum_{i=1}^{K} z_{ij}\delta_{ij}(d_i)$.

Now, we can formulate the dynamically adaptive dictionary learning problem as follows:

$$\min_{d_i, z_{ij}, \delta_{ij}} \sum_{j=1}^{M} (\frac{1}{2}||W_h^l - \sum_{i=1}^{K} z_{ij}\delta_{ij}(d_i)||_2^2 + \lambda_1 ||z_j||_1), \quad (9)$$

$$s.t. \quad ||d_i||_2 = 1, \delta_{ij} \in \triangle, K = N^l, M = p^l.$$

The problem becomes Eq. 4 when $\triangle = \{I\}$, thus we use alternate minimization to solve it.

**Sparse Codes Update** It is known that updating the sparse code is the most time consuming part [Mairal *et al.*, 2009]. One of the state-of-the-art methods for solving such lasso problem is Coordinate descent (CD) [Friedman *et al.*, 2007]. Given an input vector $w_i$, CD initializes $z_i^0 = 0$ and then updates the sparse codes many times via matrix-vector multiplication and thresholding. However, the iteration takes thousands of steps to converge. Lin et al. [2014] observed that the support (non-zero element) of the coordinates after less than ten steps of CD is very accurate. Besides, the support of the sparse code is usually more important than the exact value of the sparse code. Therefore, we update the sparse code $z_i$ by using a few steps of CD operation because the original sparse coding is a non-convex problem and do not need to run CD to the final convergence. For the $k$-th epoch, we denote the updated sparse code as $z_i^k$. It will be used as an initial sparse code for the $k+1$-th epoch.

Update $z_i^k$ via one or a few steps of coordinate descent:

$$z_i^k \leftarrow CD(D_i^k, z_i^{k-1}, x_i).$$

Specifically, for $j$ from 1 to $m$, we update the $j$th coordinate $z_{i,j}^{k-1}$ of $z_i^{k-1}$ cyclicly as follows:

$$b_j \leftarrow (d_{i,j}^k)^T (w_i - d_i^k z_i^{k-1}) + z_{i,j}^{k-1},$$

$$z_{i,j}^{k-1} \leftarrow s_\lambda(b_j),$$

where $s$ is the soft thresholding shrinkage function [Combettes and Wajs, 2005]. We call above updating cycle as one step of CD. The updated sparse code is then denoted by $z_i^k$.

**Dictionary Update** For updating the dictionary, we use block coordinate descent [Tseng, 2001] for updating each atom $d_k$.

$$d_k = \arg\min_{d_k} \sum_{j=1}^{M} \frac{1}{2}||w_j^l - \sum_{i=1}^{K} z_{ij}\delta_{ij}(d_i)||_2^2, s.t. ||d_k||_2 = 1.$$

This can be solved in two steps, the solution of the unconstrained problem by differentiation followed by normalization and can be summarized by

$$\tilde{d}_k = \sum_{j=1}^{M} z_{kj}\delta_{kj}^{-1}(w_j^l - \sum_{i \neq k} z_{ij}\delta_{ij}(d_i)), \quad (10)$$

$$d_k = \frac{\tilde{d}_k}{||\tilde{d}_k||_2}.$$

As in [Mairal *et al.*, 2009], we found that one update loop through all of the atoms was enough to ensure fast convergence of Algorithm 1. The only difference of this update compared to common dictionary learning is the shift operator $\delta_{ij}$. In Eq. (10), $\delta\delta^t = I$. If the shift operator is a non-circular operators, the inverse $\delta_{kj}^{-1}$ needs to be replaced by the adjoint $\delta_{kj}^t$ and the rescaling function $\phi = (\sum_{j=1}^{M} z_{kj}^2 \delta_{kj}\delta_{kj}^t)^{-1}$ needs to be applied to the update term.

Besides, we used a random selection method (randomly select $\hat{l}$ samples from matrices $W_h^l$ to construct initial dictionaries $D^l$) to initialize the dictionaries for different layers in DirNet. Then, we set all the sparse codes $Z_h^l$ to be zero in the beginning and $k = 10$ epoch.

## 3.4 Adaptively Hierarchical Network

Considering different functionalities among hierarchical hidden layers, we propose the method of adaptively changing sparsity in different hidden layers. We use an initial perturbations as [Zhang *et al.*, 2016] to let all features can be selected by competing with each other. Then we gradually shrank the network by using stronger $l_1$-penalties and fewer features remaining in the progressive shrinking. Therefore, DirNet will go through the self-adjusting sequential stages before reaching the final optimal.

We note that sharing $D^l$ across the inter-layer and recurrent-layer matrices allows more efficient parameterization for the weight matrices. Besides, this does not result in a significant loss of performance. By adjusting the dimensions of the projection matrices ($p^l$) in each of the layers of the network, the compression rate of the model will be determined. Therefore, after fixing the dictionary to $D^l$, solving Eq. 7 is equivalent to solve a LASSO problem [Tibshirani, 1996]. The objective function of solving $Z_x^l$ as follow:

$$\min_{Z_x^l} ||W_h^{l+1} - D^l Z_x^l||_2^2 + \lambda_2 |Z_x^l|_1. \quad (11)$$

After we get dictionary $D^l$ from Eq. 5, we can determine $Z_x^l$ as the solution to the following LASSO problem with adaptive weights to regularize the model coefficients along with different features:

$$\min_{Z_h^{l+1}} ||W_h^{l+1} - D^l Z_h^{l+1}||_2^2 + \lambda_2 |\Theta \odot Z_h^{l+1}|_1, \quad (12)$$

where $\lambda_2$ is different across layers and we selected the best value from $10^-3$ to $10^3$ in this work. $\Theta$ denotes regularization weight, we will receive different penalized matrices $Z_h^{l+1}$ instead of controlling the sparsity by $\lambda_2$ as in Eq. (11). In addition, $|\Theta \odot Z_h^{l+1}|_1 = \sum_i \theta_i |z_i^{l+1}|$ and $\Theta = |Z_{ols}|^{-\gamma}$, where $Z_{ols}$ is the ordinal least-square solution.

Therefore, we can consider the following linear regression problem with the inter weight matrix $W_x^l \in \mathbb{R}^{N^{l+1} \times N^l}$, where $N^{l+1}$ is the sample size and $N^l$ is the dimension of the target response vector. We use $n$ to represent $N^{l+1}$. Then, we use an adaptive weight vector $\Theta = [\theta_1, \theta_2, \cdots, \theta_n]^T \in \mathbb{R}^n$ to regularize over different covariates, as

$$\min_{\Theta, Z_x} ||W_x^l - D^l Z_x^l||_2^2 + \lambda_2 ||\Theta^{-\gamma} \odot Z_x^l||_1, \quad (13)$$

$$s.t. \sum_i \theta_i = \vartheta, \theta_i \geq 0,$$

where $||\Theta^{-\gamma} \odot Z_x^l||_1 = \sum_{i=1}^n \theta_i^{-\gamma} \cdot |Z_x^l|$ and we alternatively optimize $\Theta$ and $Z_x^l$ in the learning process.

Suppose we initialize $\Theta^l$, with $|\Theta^l| = \Theta_0^l$, $l$ denotes $l$-th layer. Then, we alternatively update $\Theta^l$ and $Z_x^l$ in Eq. (13) under this equality norm constraint until convergence. We will start the second stage of iterations with an updated norm constraint $|\Theta^l| = \Theta_1^l$ after the initial stage, which imposes a stronger $l_1$ penalty. Then we alternatively update $\Theta^l$ and $Z_x^l$ until the second stage ends. We keep strengthening the global $l_1$-norm regularization stage by stage during the updating procedure. We use $\tau$ to denote the index of each stage and $\Theta^l = \Theta_\tau^l$ is the compose of iteration.

To solve Eq. (13), we first fix $\Theta$ and solve $Z_x^l$, which can be computationally converted to a LASSO problem:

$$\min_{Z_x^l} ||W_x^l - D^l Z_x^l||_2^2 + \lambda_2 ||\Theta^{-\gamma} \odot Z_x^l||_1. \tag{14}$$

Then, when we fix $Z_x^l$ update $\Theta$, the problem becomes the following constrained optimization problem:

$$\min_\Theta \sum_i z_i \cdot \theta_i^{-\gamma}, \tag{15}$$

$$s.t. \sum_i \theta_i = \vartheta, \theta_i \geq 0.$$

We used the Lagrangian of Eq. 15, and drop the non-negativity constraint. Let $c_i = |z_i|_1$. Then the Lagrangian can be written as

$$J = \sum_i \theta_i c_i^{-\gamma} + \beta(\sum_i c_i - \vartheta).$$

By setting $\partial J / \partial \theta_i = 0$, we have

$$\beta = \frac{c_{i\gamma}}{\theta_i^{1+\gamma}}. \tag{16}$$

Plugging the above relation in the constraint $\sum_i \theta_i = \vartheta$, then we have

$$\theta_i^{1+\gamma} = \frac{c_{i\gamma}}{\beta} = \frac{c_{i\gamma} \cdot \vartheta^{1+\gamma}}{(\sum_i (c_{i\gamma})^{\frac{1}{1+\gamma}})^{1+\gamma}}.$$

Finally, we plug the above equation into Eq. 16 and get

$$\theta_i = (\frac{c_i^{\frac{1}{1+\gamma}}}{\sum_{i=1}^n c_i^{\frac{1}{1+\gamma}}})\vartheta. \tag{17}$$

Since $c_i = \sum_j |z_j^i| \geq 0$ and $\vartheta \geq 0$, the solution will satisfy the non-negative constraints automatically. After we trained the original RNN model and retrieved the weight matrices of both inter-layer and recurrent layer, the proposed algorithm is applied to learn the new matrices $Z_h$, $Z_x$ and $D$ of each layer, which are used to initialize parameters of neural network layers in the new network structure (Fig.1 (b)). After the initialization, we fine-tune the neural network as Han et al. [2015a] and the sparsity is preserved by only updating non-zero elements in the sparse matrix. Fine-tune step is also required for other compression works, e.g., [Prabhavalkar *et al.*, 2016].

### 3.5 Extended Model Compression

We extend the Sec. 3.3 - Sec. 3.4 from standard RNNs to LSTM. In LSTM, the recurrent-weight matrix $W_h^l$ and the inter-layer matrix $W_x^l$ are both concatenation of four gates, which are input gate, output gate, forget gate and input modulation gate. We stack them vertically and denote as $(i, o, f, g)^T$.

In this work, we do not consider the peephole weights because it already narrows and will not compress a lot of parameters. Thus, we rewrite the Eq. (1) and Eq. (2) as follows:

$$\begin{pmatrix} i \\ o \\ f \\ g \end{pmatrix} = \begin{pmatrix} sigmod \\ sigmod \\ sigmod \\ \tanh \end{pmatrix} (T_{n,4n}, D_{n,4n}^l Z_{h(n,4n)}^l) \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$\begin{pmatrix} i \\ o \\ f \\ g \end{pmatrix} = \begin{pmatrix} sigmod \\ sigmod \\ sigmod \\ \tanh \end{pmatrix} (D_{n,4n}^l Z_{x(n,4n)}^{l+1}, T_{n,4n}) \begin{pmatrix} h_t^l \\ h_{t-1}^{l+1} \end{pmatrix}$$

Besides, compressing a single-layer LSTM model is a special case for the proposed DirNet: (1) we can still learn the shared $D^l$ using $W_x^l$ and $W_h^l$ of the single LSTM/RNN layer as in Fig.1 (b) to achieve a high compression rate on a single-layer LSTM model; (2) compressing multiple LSTM/RNN layers, which can adaptively change the sparsity of the sparse codes relying on cross-layer information, will achieve higher compression rate than a single-layer LSTM/RNN model.

## 4 Experimental Results and Discussion

The proposed approach is a general algorithm for compressing recurrent neural networks including vanilla RNN, LSTM and GRU, etc, and thus can be directly used in many problems, e.g., language modeling (LM), speech recognition, machine translation and image captioning. In this paper, we select two popular domains: LM and speech recognition. We compare DirNet with (1) *LSTM-SVD* and *LSTM-ODL*. *LSTM-SVD* is the method proposed by Prabhavakar et al. [2016] which compresses RNNs by low-rank SVD. *LSTM-ODL* is the method which compresses RNNs by online dictionary learning [Mairal *et al.*, 2009]. To train DirNet, it takes 10 hours on PTB with 1 K80 GPU and around 120 hours on LibriSpeech using 16 Nvidia K80 GPUs.

### 4.1 Language Modeling

We conduct word-level prediction on the Penn Tree Bank (PTB) dataset [Marcus *et al.*, 1993], which consists of 929k training words, 73k validation words and 82k test words. We first train a two-layer unrolled LSTM model with 650 units per layer, which is using the same network setting of LSTM-medium in [Zaremba *et al.*, 2014]. For comparison, the same model is compressed using LSTM-SVD, LSTM-ODL as well as DirNet. We report the result of testing data in Table 1. We observe that DirNet achieves superior performances regarding compression rates and speedup rates while maintaining the accuracy. The reason is that dynamically adjusting the projection matrices $D$ can achieve better compression rate and adapt different sparsities across layers can receive negligible accuracy loss.

| Network | # Params | R | T | PER |
|---|---|---|---|---|
| LSTM | 4.7M | 1x | 1x | 81.3 |
| LSTM-SVD | 2.4M | 2.0x | 1.9x | 81.4 |
| LSTM-ODL | 1.6M | 2.9x | 2.8x | 81.5 |
| DirNet | 0.7M | 6.7x | 6.4x | 81.5 |
| LSTM-SVD | 0.7M | 6.7x | 6.4x | 87.4 |
| LSTM-ODL | 0.7M | 6.7x | 6.4x | 83.2 |
| DirNet | 0.7M | 6.7x | 6.4x | 81.5 |

Table 1: Comparison of the number of parameters (Params), compression rates (R), speedup on mobile CPU speed (T) and Perplexity (PER) on PTB dataset.

| Network | # Params | R | T | WER |
|---|---|---|---|---|
| LSTM | 10.3M | 1x | 1x | 12.7 |
| LSTM-SVD | 6.9M | 1.5x | 1.4x | 12.7 |
| LSTM-ODL | 3.7M | 2.8x | 2.6x | 12.7 |
| DirNet | 2.4M | 4.3x | 4.2x | 12.7 |
| LSTM-SVD | 1.3M | 7.9x | 7.6x | 16.1 |
| LSTM-ODL | 1.3M | 7.9x | 7.6x | 14.3 |
| DirNet | 1.3M | 7.9x | 7.6x | 12.9 |

Table 2: Comparison of the number of parameters (Params), compression rates (R), speedup on mobile CPU speed (T) and Word error rates (%) (WER) on LibriSpeech dataset.

## 4.2 Speech Recognition

**Dataset:** The LibriSpeech corpus is a large (1000 hour) corpus of English read speech derived from audiobooks in the LibriVox project, sampled at 16kHz. The accents are various and not marked, but the majority are US English. It is publicly available at http://www.openslr.org/12/ [Panayotov *et al.*, 2015]. LibriSpeech comes with its own train, validation and test sets and we use all the available data for training and validating our models. Moreover, we use the 100-hour "test clean" set as the testing set. Mel-frequency cepstrum (MFCC) [Davis and Mermelstein, 1990] features are computed with 26 coefficients, a 25 ms sliding window and 10 ms stride. We use nine time slices before and nine after, for a total of 19 time points per window. As a result, with 26 cepstral coefficients, there are 494 data points per 25 ms observation.

**Baseline Model:** Following [Prabhavalkar *et al.*, 2016], our baseline model for speech recognition is a 5-layer RNN model with the Connectionist Temporal Classification (CTC) loss function [Graves *et al.*, 2006], which predicts 41 context-independent (CI) phonemes [Sak *et al.*, 2015]. Each hidden RNN layer is composed of 500 LSTM units. We train all models for 100 epochs using Adam optimizer [Kingma and Ba, 2014] with an initial learning rate of 0.001. It takes around 35 and 55 epochs to converge of DirNet and the baseline model, respectively. The size of the batch is 32. The model is first trained to convergence for optimizing the CTC criterion, followed which are sequences discriminatively trained to optimize the state-level minimum Bayes risk (sMBR) criterion [Kingsbury, 2009]. We implement DirNet by Tensorflow [Abadi *et al.*, 2016] and use Samsung Galaxy S8 smartphone as the mobile platform to evaluate the performance.

**Results:** To give a comprehensive evaluation of the proposed approach, we list the word error rate along with parameter number and the execution time comparison between different approaches in Table 2. There are two sets of experiments: 1) compare the maximum compression rate obtained by different approaches without the accuracy drop 2) measure the word error rate among different approaches with the same compression rate.

As indicated in the first row of Table 2, the basic LSTM model can achieve 12.7% (WER) before any compression. It includes 10.3M parameters in total and takes 1.77s to recognize 1s audio. Our proposed approach DirNet can compress the model by 4.3 times without losing any accuracy. The speed up achieved by DirNet is 4.2 times faster compared to 1.4 and 2.6 times obtained by LSTM-SVD and LSTM-ODL. These results indicate the superiority of the proposed DirNet over current compressing algorithms. Moreover, such significant improvement shows that dynamically adjust sparsity across layers might help the compression models receive higher compression rates. From the results listed in the second row of Table 2, we can find that when compressing the model size by 7.9 times, LSTM-ODL and LSTM-SVD lose lots of accuracy, especially SVD based compression algorithm, which is 3.4% (16.1%-12.7%) compared to 0.2% achieved by DirNet.

## 4.3 Adaptively Shrinking Parameter Selection

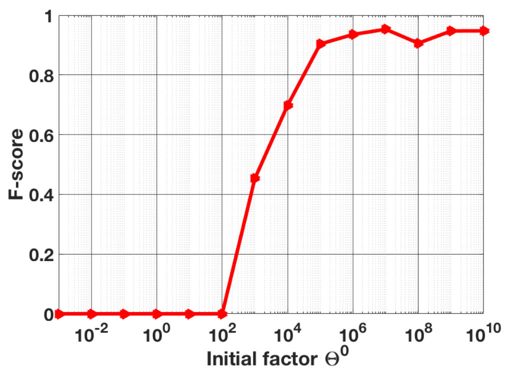
In this section, we study how the performance of our approach is affected by the following two parameters: $\Theta^0$ that controls the initial "sparsity" of the system and the shrinking factor $\gamma$ that controls the compression rate of the system. We use F-score on LibriSpeech dataset to measure the performance.

First, we examine the performance of choosing different $\Theta^0$ values. The range is from $10^{-3}$ to $10^{10}$, and the result is shown on the left side of the Fig. 3. We notice that when $\Theta^0$ is below $10^5$, the performance quickly drops and the lower initial sparsity even fails to start the whole system with sufficient energy. As a result the iterations could quickly stop at a local optimal. Therefore, we choose the best performance $\Theta^0 = 10^7$ in all experimental settings.

Second, we study the influence of varying shrinking factor $\gamma$. The explored value range is from $10^{-6}$ to $10^0$. We observe that the performance sharply increases first and becomes stable afterward. Besides, the system evolves slowly such that the shrinking stage is sufficient when the shrinking scheme $\gamma \rightarrow 1$. In practice, we choose $\gamma = 0.4$ to strike a balance between efficiency and the quality of shrinking procedure.

## 4.4 Adaptive Hierarchy of the Network

We also compare the performance of DirNet (O) adaptive cross-layers' compression capability with LSTM-SVD (G). All the results are listed in Table 3. We can find that DirNet achieves much lower WER compared to LSTM-SVD even with the same number of neurons. Also, these results further demonstrate the superiority of DirNet adaptive adjusting sparse features in deep model compression.



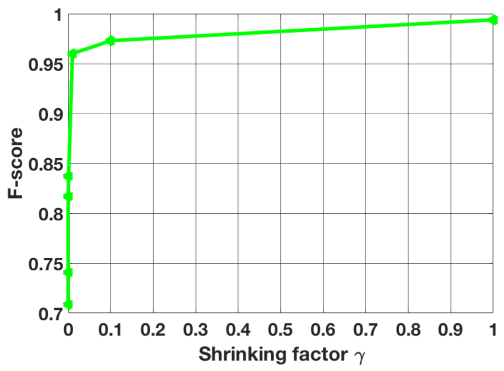


Figure 3: Performance of different parameter selections.

| # neurons of each layer | Params | WER (G) | WER (O) |
|---|---|---|---|
| 500, 500, 500, 500, 500 | 10.3M | 12.7 | 12.7 |
| 350, 375, 395, 405, 410 | 8.6M | 12.3 | 12.3 |
| 270, 305, 335, 345, 350 | 7.2M | 12.5 | 12.3 |
| 175, 215, 245, 260, 265 | 5.4M | 12.5 | 12.4 |
| 120, 150, 180, 195, 200 | 4.1M | 12.6 | 12.5 |
| 80, 105, 130, 145, 150 | 3.1M | 12.9 | 12.5 |
| 50, 70, 90, 100, 110 | 2.3M | 13.2 | 12.7 |
| 30, 45, 55, 65, 75 | 1.7M | 14.4 | 12.9 |
| 25, 35, 45, 50, 55 | 1.3M | 16.6 | 12.9 |

Table 3: The Word error rates (%) (WER) on the testing set as varying the dimensions of each projection matrices $D^l$ by adaptively Adjusting the hierarchical structure of RNN.

## 5 Conclusions

In this paper, we introduce DirNet that dynamically adjusts the compression rate of each layer in the network, and adaptively change the hierarchical structures among different layers on their weight matrices. Experimental results show that compared to other RNN compression methods, DirNet significantly improves the performance before retraining. In our ongoing work, we will integrate scalar level compression with our DirNet to further compress the deep model.

## References

[Abadi *et al.*, 2016] Martín Abadi, Paul Barham, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283, GA, 2016. USENIX Association.

[Chen *et al.*, 2017] Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. *arXiv preprint arXiv:1707.01220*, 2017.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. abs/1406.1078, 2014.

[Combettes and Wajs, 2005] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[Davis and Mermelstein, 1990] Steven B Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In *Readings in speech recognition*, pages 65–74. Elsevier, 1990.

[Denil *et al.*, 2013] Misha Denil, Babak Shakibi, et al. Predicting parameters in deep learning. In *NIPS*, pages 2148–2156, 2013.

[Friedman *et al.*, 2007] Jerome Friedman, Trevor Hastie, et al. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[Geng *et al.*, 2017] Yanyan Geng, Guohui Zhang, Weizhi Li, Yi Gu, Gaoyuan Liang, Jingbin Wang, Yanbin Wu, Nitin Patil, and Jing-Yan Wang. A novel image tag completion method based on convolutional neural network. In *International Conference on Artificial Neural Networks*, pages 539–546. Springer, 2017.

[Graves *et al.*, 2006] Alex Graves, Santiago Fernández, et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, pages 369–376. ACM, 2006.

[Graves *et al.*, 2013] Alex Graves, Abdel-rahman Mohamed, et al. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649. IEEE, 2013.

[Han *et al.*, 2015a] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[Han *et al.*, 2015b] Song Han, Jeff Pool, et al. Learning both weights and connections for efficient neural network. In *NIPS*, pages 1135–1143, 2015.

[Han *et al.*, 2017] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, et al. Ese: Efficient speech recognition engine with sparse lstm on fpga. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 75–84. ACM, 2017.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[Hitziger *et al.*, 2013] Sebastian Hitziger, Maureen Clerc, et al. Jitter-adaptive dictionary learning-application to multi-trial neuroelectric signals. *arXiv preprint arXiv:1301.3611*, 2013.

[Kim and Rush, 2016] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

[Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[Kingsbury, 2009] Brian Kingsbury. Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling. In *ICASSP*, pages 3761–3764, 2009.

[Lei *et al.*, 2013] Xin Lei, Andrew W Senior, et al. Accurate and compact large vocabulary speech recognition on mobile devices. In *INTERSPEECH*. Citeseer, 2013.

[Li *et al.*, 2017] Dawei Li, Xiaolong Wang, and Deguang Kong. Deeprebirth: Accelerating deep neural network execution on mobile devices. *arXiv preprint arXiv:1708.04728*, 2017.

[Lin *et al.*, 2014] Binbin Lin, Qingyang Li, et al. Stochastic coordinate coding and its application for drosophila gene expression pattern annotation. *arXiv:1407.8147*, 2014.

[Mairal *et al.*, 2009] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *ICML*, pages 689–696. ACM, 2009.

[Marcus *et al.*, 1993] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[Mikolov *et al.*, 2011] Tomàš Mikolov, S. Kombrink, et al. Extensions of recurrent neural network language model. In *ICASSP*, pages 5528–5531, May 2011.

[Narang *et al.*, 2017] Sharan Narang, Gregory Diamos, Shubho Sengupta, and Erich Elsen. Exploring sparsity in recurrent neural networks. *ICLR*, 2017.

[Panayotov *et al.*, 2015] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *ICASSP*, pages 5206–5210. IEEE, 2015.

[Prabhavalkar *et al.*, 2016] Rohit Prabhavalkar, Ouais Alsharif, et al. On the compression of recurrent neural networks with an application to lvcsr acoustic modeling for embedded speech recognition. In *ICASSP*, pages 5970–5974. IEEE, 2016.

[Sainath *et al.*, 2013] Tara N Sainath, Brian Kingsbury, et al. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *ICASSP*, pages 6655–6659, 2013.

[Sak *et al.*, 2014] Hasim Sak, Andrew Senior, et al. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, 2014.

[Sak *et al.*, 2015] Haşim Sak, Andrew Senior, et al. Learning acoustic frame labeling for speech recognition with recurrent neural networks. In *ICASSP*, pages 4280–4284. IEEE, 2015.

[Tibshirani, 1996] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[Tseng, 2001] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[Vinyals *et al.*, 2017] Oriol Vinyals, Alexander Toshev, et al. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):652–663, 2017.

[Xu *et al.*, 2018] Chen Xu, Jianqiang Yao, et al. Alternating multi-bit quantization for recurrent neural networks. In *ICLR*, 2018.

[Xue *et al.*, 2013] Jian Xue, Jinyu Li, and Yifan Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In *INTERSPEECH*, pages 2365–2369, 2013.

[Zaremba *et al.*, 2014] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[Zhang *et al.*, 2016] Kai Zhang, Shandian Zhe, et al. Annealed sparsity via adaptive and dynamic shrinking. In *KDD*, pages 1325–1334, 2016.