HIGHLY EFFICIENT NEURAL NETWORK LANGUAGE MODEL COMPRESSION USING SOFT BINARIZATION TRAINING

Rao Ma, Qi Liu, Kai Yu

SpeechLab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence
Shanghai Jiao Tong University, Shanghai, China
{rm1031, liuq901, kai.yu}@sjtu.edu.cn

ABSTRACT

The long short-term memory language model (LSTM LM) has been widely investigated in large vocabulary continuous speech recognition (LVCSR) task. Despite the excellent performance of LSTM LM, its usage in resource-constrained environments, such as portable devices, is limited due to the high consumption of memory. Binarized language model has been proposed to achieve significant memory reduction at the cost of performance degradation at high compression ratio. In this paper, we propose a soft binarization approach to recover the performance of binarized LSTM LM. Experiments show that the proposed method can achieve a high compression rate of 30× with almost no performance loss in both language modeling and speech recognition tasks.

Index Terms— language model, speech recognition, binarized neural network, knowledge distillation, model compression

1. INTRODUCTION

Language model measures the fluency and reasonable degree of a sentence, which plays an important role in machine translation [1], handwriting recognition [2], automatic speech recognition [3] and other tasks. N-gram language model and its variance have been widely used in LVCSR system for a long time [4]. However, n-gram fails to model long context sequences as it uses very limited history information. Neural network language models (NN LMs) have received much attention since proposed. Feedforward neural network (FNN) language model [5] solves the problem of data sparseness. Later on, Recurrent neural network (RNN) and long short-term memory (LSTM) networks are proposed to handle the problem of long-term dependency, which are successfully applied in language modeling [6, 7].

This work has been supported by the National Key Research and Development Program of China (Grant No.2017YFB1002102) and the China NSFC projects (No.61573241). Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

In practical applications, such as ASR or machine translation tasks, NN LM often builds on high dimensional input/output/state vectors, e.g., large vocabulary or deep recurrent layers, prohibiting it from deploying in portable devices with limited resources. In addition, the significant inference latency caused by the recursive nature of RNN is unfavorable in real-time applications.

To alleviate such problems, several language model compression methods have been proposed, i.e. low-rank decomposition [8, 9], structured embedding [10, 11], and vector quantization [12]. In this work, we mainly force on the binarization method, which could achieve a particularly high compression ratio of 32×. However, existing binarized language models always show a large performance gap compared with the full precision networks. Despite the success of BinaryConnect [13] and BWN [14] in binarizing CNN weights, they fail to binarize the parameters of RNNs [15]. In [16], all fully binarized LSTM LMs are subject to performance degradation to varying degrees, up to 7.5% degradation in PPL can be observed. [17] only binarizes the hidden layers of the language model, resulting in a small overall compression ratio of 1.1×. [18] targets the problem of multi-bit quantization, which increases the model size more than two times compared to the binarized models. Therefore, there is no binarized language model that can match the performance of the full precision model in literature.

The performance of highly compact binarized language model degrades sharply mainly for two reasons. First, the representation ability of the model is weakened due to the destructive property of binary quantization. Second, the mere use of training data with the given label to train binarized language models may not provide sufficient information when quantization is employed in the training phase, which suggests that it might be useful to incorporate some kind of 'dark knowledge' [19, 20] to help fill the information gap smoothly.

In this paper, we adopt a novel soft binarization method to deal with the above shortcomings of the binarization method. The term 'soft' of our approach has two aspects. First, a small number of scalars are used to reduce the quantization

error in the binarized language models. Second, we modify the training criterion of NN LM. We apply knowledge distillation technique to transfer knowledge from a trained full precision language model. Previous works on binarized language models only train the compressed model on the original data, which is insufficient for the compressed model to learn the optimal parameters. Experimental results show that by incorporating the soft labels generated by full precision LSTM LM into the training phase of the binarized language model, comparable or even better results can be achieved w.r.t to the full precision counterpart.

This paper is organized as follows. In section 2, we give a detailed review of LSTM LM and existing binarized neural network frameworks. In section 3, the proposed soft binarization method is introduced. Section 4 shows experimental results and analysis. Finally, we conclude this paper and discuss future work in section 5.

2. RELATED WORKS

2.1. LSTM Language Model

The goal of language model is to calculate the joint probability of a given sentence $(w_1, w_2, \cdots, w_N)$, which could be decomposed as

$$p(w_1, \cdots, w_N) = \prod_{i=1}^{N} p(w_i | w_1 \cdots w_{i-1}) \tag{1}$$

NN LM typically consists of three parts to calculate the conditional probability: input embedding, hidden layers, and output embedding. The input words are encoded by 1-of-K coding $\bar{\boldsymbol{x}}_t$ where K is the vocabulary size. The input embedding $\boldsymbol{W}_{in}$ is a lookup table which maps each word to an embedding vector $\boldsymbol{x}_t$. The output embedding $\boldsymbol{W}_{out}$ is a linear layer followed by a softmax operation, which projects hidden state $\boldsymbol{h}_t$ to a word probability distribution $\boldsymbol{p}_t$. The detailed formula is given as:

$$\begin{aligned} \boldsymbol{x}_t &= \boldsymbol{W}_{in}\bar{\boldsymbol{x}}_t \\ \boldsymbol{h}_t &= \text{Hidden}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \\ \boldsymbol{p}_t &= \text{softmax}(\boldsymbol{W}_{out}\boldsymbol{h}_t + \boldsymbol{b}) \end{aligned} \tag{2}$$

Hidden denotes the hidden layer transformation function. RNN is a type of cyclic neural network which is efficient in modeling sequential data. At each time step t, RNN reads in $\boldsymbol{x}_t$ and generates output as

$$\boldsymbol{h}_t = g(\boldsymbol{W}_x\boldsymbol{x}_t + \boldsymbol{W}_h\boldsymbol{h}_{t-1} + \boldsymbol{b}_r) \tag{3}$$

where $\boldsymbol{W}_x$, $\boldsymbol{W}_h$, $\boldsymbol{b}_r$ are model parameters, g is the activation.

Due to the vanishing and exploding gradients problem of RNN [21], LSTM as a unit structured RNN, has been proposed for more stable behavior. Given $\boldsymbol{x}_t$, $\boldsymbol{h}_{t-1}$, $\boldsymbol{c}_{t-1}$ as input, LSTM produces $\boldsymbol{h}_t$ and $\boldsymbol{c}_t$ at each time step. The LSTM formula is shown below:

$$\begin{aligned} \boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \\ \boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \\ \boldsymbol{g}_t &= \tanh(\boldsymbol{W}_{xg}\boldsymbol{x}_t + \boldsymbol{W}_{hg}\boldsymbol{h}_{t-1} + \boldsymbol{b}_g) \\ \boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \\ \boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \\ \boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \end{aligned} \tag{4}$$

where $\boldsymbol{c}_t$ is the cell state to remember history information. $\boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t$ are called gates which protect and control the cell state through sigmoid function. All the $\boldsymbol{W}_{**}$ are the weight matrix parameters, $\boldsymbol{b}_{**}$ are the bias. Back propagation through time (BPTT) [22] algorithm is commonly used to train LSTM language model.

2.2. Binarized Neural Network

Several methods attempt to binarize the weights in deep neural networks. In BinaryConnect [13], weights are constrained to either $+1$ or -1 during propagations. For a real-valued matrix $\boldsymbol{W}$ with entries w_{ij}, the binarized weight element w_{ij}^{b} is obtained by the sign function,

$$w_{ij}^{\text{b}} = \text{sign}(w_{ij}) = \begin{cases} +1 & \text{if } w_{ij} \geq 0, \\ -1 & \text{otherwise.} \end{cases} \tag{5}$$

[15] shows that BinaryConnect yields poor performance on binarized language models due to the problem of exploding gradients. Binary-weight-network (BWN) [14] extends BinaryConnect to estimate the full precision weight $\boldsymbol{W}$ by $\boldsymbol{W} \approx \alpha\boldsymbol{W}^{\text{b}}$ where scaling factor α is the average of absolute weight values, i.e. $\alpha = \frac{1}{n}\|\boldsymbol{W}\|_{l1}$. BWN outperforms BinaryConncet by a large margin on the ImageNet classification task. However, there is still noticeable performance loss for language models compressed with BWN [15].

[23] instead tries to approximate $\boldsymbol{W}$ with k multiple binary weight matrices: $\boldsymbol{W} \approx \sum_{j=1}^{k} \alpha_j \boldsymbol{W}_j^{\text{b}}$ where $\alpha_j > 0$ and $\boldsymbol{W}_j^{\text{b}}$ is a binarized matrix. This problem is also defined as multi-bit quantization. [24] and [25] test the effectiveness of multi-bit quantized RNNs in language modeling on the PTB dataset. The compressed model fails to reach the performance of the full precision model until using 4-bits quantization. [18] uses alternating minimization to improve the optimization algorithm and match the performance of the baseline language model when $k = 3$. In general, multi-bit quantization outperforms binarization based methods at the cost of a smaller compression ratio of $32/k$.

[17] targets the problem of RNN binarization. They propose the use of batch normalization to learn binarized recurrent weights. Nevertheless, binarization of embedding layers is not studied in their paper, so the memory cost of the compressed model remains large. There is another limitation of this paper that a large batch size has to be used since a

small batch size results in a high variance when estimating the statistics of the unnormalized vector [17]. [16] trains language model with binarized embeddings and binarized recurrent weights. However, the performance gap between the full precision model and the proposed binarized model is huge. Moreover, we claim that their method gives poor performance in capturing long-range dependencies in documents, which will be shown in our experiments.

In the training phase of binarized neural networks (BNN), instead of directly training on binarized weights, real-valued weights are employed which accumulate real-valued gradients. Then in the forward propagation, weights are constrained to $\{+1, -1\}$ based on their sign. Since the binarization functions used in the above works are not differentiable, the derivative of the loss C w.r.t the real-valued matrix $\boldsymbol{W}$ is approximated by a straight-through estimator [26],

$$\frac{\partial C}{\partial \boldsymbol{W}} \approx \frac{\partial C}{\partial \boldsymbol{W}^{\mathrm{b}}} \tag{6}$$

where $\boldsymbol{W}^{\mathrm{b}}$ denotes the binarized weights.

Apart from the benefit of substantial memory savings, most multiply-accumulate operations of BNN could be replaced by simple accumulations, which might lead to a substantial increase of power-efficiency [27]. Training BNN could also be seen as a variant of dropout, in which instead of randomly substituting half of the activations with zeros, weights are binarized when computing the parameter gradients. The drawback of binarized models, however, is degraded accuracy. In this work, we aim at training a binarized language model that achieves comparable prediction accuracy to its full precision counterpart.

3. NN LM WITH SOFT BINARIZATION

3.1. Binarized Embedding Language Model

In NN LM, word embedding layers represent the linguistic information of words in the form of continuous vectors, which account for more than 90% of the total parameters in practice. For a language model with 200K vocabulary, embedding layers occupy 800MB space when each embedding has 500 dimensions. If the embedding vectors are binarized, the memory consumption can be reduced to only 25MB.

In this work, we propose a novel binarized embedding language model (BELM) framework based on [16]. The main goal of our method is to train binary weights for embedding layers. Following [16], for a full precision weight matrix $\boldsymbol{W}$, the binarization function for each entry w_{ij} is defined as

$$\text{binarize}(w_{ij}) = \begin{cases} +\frac{1}{\sqrt{H}} & \text{if } w_{ij} \geq 0, \\ -\frac{1}{\sqrt{H}} & \text{otherwise.} \end{cases} \tag{7}$$

where H is the hidden dimension. We find in experiments that instead of constraining weights to $\{+1, -1\}$, binarization with this function helps the model converge faster and render slightly better performance.

The binarized embeddings are calculated as $\boldsymbol{W}_{in}^{\mathrm{b}} = \text{binarize}(\boldsymbol{W}_{in})$ and $\boldsymbol{W}_{out}^{\mathrm{b}} = \text{binarize}(\boldsymbol{W}_{out})$ and the gradients are propagated according to equation 6. To compensate for the quantization error, additional real-valued shift vectors $\boldsymbol{\gamma}_{\text{in}}$, $\boldsymbol{\gamma}_{\text{out}}$ and a real-valued projection layer $\boldsymbol{W}_y$ are used in our model. Previous works have shown that the output layer greatly affects the performance of the language model [28]. In order to enhance the binarized output embedding, we insert a full precision projection layer $\boldsymbol{W}_y$ with no activation function after the hidden layers. [29] indicates that words with higher norms carry more semantic information. In their work, the vocabulary of a text classification model is pruned by selecting words with the highest norms, which achieves minor accuracy loss. However, the $L2$-norm for each word in the binarized output embedding would be the same. So we adopt a real-valued vector $\boldsymbol{\gamma}_{out}$ to readjust the norm of each word embedding. A real-valued shift vector $\boldsymbol{\gamma}_{in}$ is also used to rescale the binarized input embeddings. The computations for BELM network are defined as

$$\begin{aligned} \boldsymbol{x}_t &= \boldsymbol{W}_{in}^{\mathrm{b}} \bar{\boldsymbol{x}}_t \odot e^{\boldsymbol{\gamma}_{in}} \\ \boldsymbol{h}_t &= \text{LSTM}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \\ \boldsymbol{s}_t &= \boldsymbol{W}_y \boldsymbol{h}_t + \boldsymbol{b}_y \\ \boldsymbol{p}_t &= \text{softmax}(\boldsymbol{W}_{out}^{\mathrm{b}} \boldsymbol{s}_t \odot e^{\boldsymbol{\gamma}_{out}} + \boldsymbol{b}) \end{aligned} \tag{8}$$

3.2. Fully Binarized Language Model

We define fully binarized language model (FBLM) as a LSTM LM containing only binary weights. As shown in subsection 2.1, one LSTM layer contains eight parameter matrices, which are typically overparameterized. For each binarized matrix, a real-valued scaling vector $\boldsymbol{\gamma}$ is used to enhance the model performance. The formula for LSTM with binarized weights is

$$\begin{aligned} \boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{xi}^{\mathrm{b}} \boldsymbol{x}_t \odot e^{\boldsymbol{\gamma}_{xi}} + \boldsymbol{W}_{hi}^{\mathrm{b}} \boldsymbol{h}_{t-1} \odot e^{\boldsymbol{\gamma}_{hi}} + \boldsymbol{b}_i) \\ \boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{xf}^{\mathrm{b}} \boldsymbol{x}_t \odot e^{\boldsymbol{\gamma}_{xf}} + \boldsymbol{W}_{hf}^{\mathrm{b}} \boldsymbol{h}_{t-1} \odot e^{\boldsymbol{\gamma}_{hf}} + \boldsymbol{b}_f) \\ \boldsymbol{g}_t &= \tanh(\boldsymbol{W}_{xg}^{\mathrm{b}} \boldsymbol{x}_t \odot e^{\boldsymbol{\gamma}_{xg}} + \boldsymbol{W}_{hg}^{\mathrm{b}} \boldsymbol{h}_{t-1} \odot e^{\boldsymbol{\gamma}_{hg}} + \boldsymbol{b}_g) \\ \boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{xo}^{\mathrm{b}} x_t \odot e^{\boldsymbol{\gamma}_{xo}} + \boldsymbol{W}_{ho}^{\mathrm{b}} \boldsymbol{h}_{t-1} \odot e^{\boldsymbol{\gamma}_{ho}} + \boldsymbol{b}_o) \\ \boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \\ \boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \end{aligned} \tag{9}$$

where $\boldsymbol{W}_{**}^{\mathrm{b}}$ denote binarized matrices obtained with the *binarize* function. We also apply binarization on the additional projection layer inserted in BELM. The third row of BELM formula is replaced by

$$\boldsymbol{s}_t = \boldsymbol{W}_y^{\mathrm{b}} \boldsymbol{h}_t \odot e^{\boldsymbol{\gamma}_y} + \boldsymbol{b}_y \tag{10}$$

3.3. Memory Consumption Comparison

Denote the embedding dimension, hidden dimension as E and H. Suppose the vocabulary size is V. In table 1, we calculate the memory consumption of LSTM LM with one hidden layer and the corresponding BELM and FBLM. Here we assume $E = H$ for simplicity.

Model	Memory (byte)
LSTM	$8VH + 32H^2 + 16H + 4V$
BELM	$0.25VH + 36H^2 + 24H + 8V$
FBLM	$0.25VH + 1.125H^2 + 60H + 8V$

Table 1. Model size for LSTM LM, BELM and FBLM.

In language modeling, vocabulary size V is usually significantly larger than the hidden size H or embedding size E. Therefore the embedding layers dominate the parameter size of the language model. The table shows that with binarization on embedding layers, the coefficient of the VH item drops from 8 to 0.25, resulting in a substantial reduction in model size. While embedding layers are binarized, parameters of the hidden layers become crucial to the size of NN LM. As shown in the table, FBLM yields a particularly high compression ratio approximates to $32\times$.

3.4. Knowledge Distillation

Knowledge distillation is also referred to as teacher-student learning, which is widely used in model compression. In [30], knowledge is successfully transferred from full precision ResNets into networks with ternary or 4-bit precision. The application of knowledge distillation in language modeling is investigated in [31, 32]. The previous work shows that LSTM LM performance can be transferred into FNN LM with a compression ratio of 1.6. The latter work reduces the parameter size to 18.5% with similar WER performance. In our work, we aim at reducing the language model size to 3% with no performance degradation. We propose to use the full precision language model as the teacher model to guide the training of binarized language models. The schematic of the knowledge distillation setup is shown in figure 1.

Denote sample from training data as (x, y) and suppose the vocabulary size is V. The original training criterion is to minimize NLL loss for each sample from the training data,

$$\mathcal{L}_{\text{NLL}} = \sum_{t=1}^{T}\sum_{k=1}^{|V|} -\mathbb{1}\{y=k\}\log p(y=k|h_t;\theta) \qquad (11)$$

In the knowledge distillation process, the binarized model is also trained to mimic the behavior of the full precision model. Denote the output distribution for teacher model and student model as $q(y|x;\theta_T)$ and $p(y|x;\theta)$. $q(y|x;\theta_T)$ is also called 'soft label' which could provide extra context information. The Kullback-Leibler divergence between them is calculated as

$$\begin{aligned} KL(\theta_T|\theta) &= \sum_{t=1}^{T}\sum_{k=1}^{|V|} q(y=k|h_t;\theta_T)\log\frac{q(y=k|h_t;\theta_T)}{p(y=k|h_t;\theta)} \\ &= \mathcal{L}_{\text{KD}}(\theta;\theta_T) - \mathcal{L}(\theta_T) \\ \mathcal{L}(\theta_T) &= \sum_{t=1}^{T}\sum_{k=1}^{|V|} -q(y=k|h_t;\theta_T)\log q(y=k|h_t;\theta_T) \end{aligned} \qquad (12)$$

Here, $\mathcal{L}(\theta_T)$ is the entropy of the teacher distribution, which is irrelevant to θ and thus has no influence on the training of the student model. Our goal is equivalent to minimizing the cross entropy

$$\mathcal{L}_{\text{KD}} = \sum_{t=1}^{T}\sum_{k=1}^{|V|} -q(y=k|h_t;\theta_T)\log p(y=k|h_t;\theta) \qquad (13)$$

It's a common practice to interpolate the loss generated from the original data and the teacher output. The final objective function is therefore

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{\text{NLL}} + \alpha\mathcal{L}_{\text{KD}} \qquad (14)$$

where α is the interpolation weight to combine the one-hot distribution and teacher distribution.

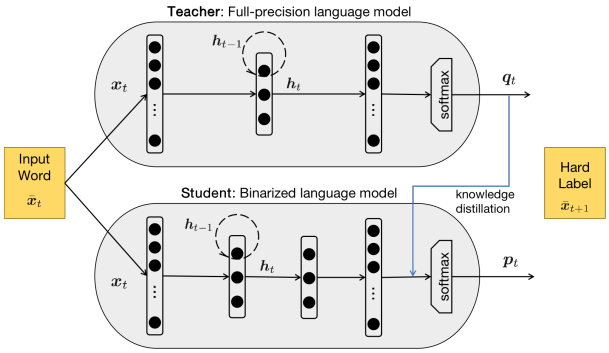

Fig. 1. Schematic of the knowledge distillation setup. The teacher network is a full precision language model and the student network is a binarized language model.

4. EXPERIMENTS

4.1. Data Description

We design experiments on the English Penn TreeBank (PTB), Chinese short message (SMS) corpus and SWB-Fisher (SWB) spoken language dataset to evaluate the performance of compressed language models. The PTB corpus is a famous dataset with a vocabulary size of 10K words. The training set contains 42K sentences with 887K words. The Chinese

SMS corpus has a vocabulary size of about 40K words. The training set contains 380K sentences with 1,931K words. The SWB-Fisher corpus contains 2.5M English sentences with 24.9M words, and the vocabulary size is 31K.

4.2. Experiments on Language Modeling

We use the same experiment settings as in [33] to train language models on PTB corpus. [1] We train LSTM language models with two different sizes, both unrolled for 35 steps. The small model has one LSTM layer with 300 units, which could be fairly compared with [17] and [18]. We also examine the prediction accuracy of our method on a large model, which contains two LSTM layers of size 650.

Hidden size	Model	PPL		Size (byte)
		NLL	+KD	
300×1	LSTM	86.7	83.0	25.7M
	BELM	92.8	85.2	3.9M
	FBLM	100.3	88.1	0.9M
650×2	LSTM	81.8	81.2	75.4M
	BELM	86.0	75.0	29.0M
	FBLM	92.3	77.5	2.5M

Table 2. Test perplexity results for LSTM, BELM and FBLM on PTB dataset. +KD indicates knowledge distillation from LSTM LM.

Experimental results for NN LM with soft binarization are listed in table 2. On the third column, BELM and FBLM are trained without using any help from a teacher network, which suffer from severe performance degradation. The fourth column shows that with the modified training criterion, the performance of the compressed models improves significantly. In fact, the accuracy of the full precision LSTM LM also improves slightly when learned from a trained model. Combination weight α is set to 0.5 and 0.8 accordingly. With knowledge distillation, small BELM which has binary embeddings surpasses the baseline LSTM on PPL measure and yields 6.6× compression. And the distilled large BELM has a relative PPL improvement of 8.3% with 2.6× savings w.r.t. full precision model. By distilling knowledge from the teacher LSTM LM, small FBLM achieves perplexity of 88.1 with the model size of only 0.9MB. Distilled FBLM with large size outperforms the teacher model but with only 3% of memory consumption. Experiment results indicate that binarizing weights in NN LM scarcely hurt model performance when paired with knowledge distillation. Indeed, binarization provides a certain degree of regularization to help the model generalize better [25].

[1] We follow the example codes on `https://github.com/pytorch/examples/tree/master/word_language_model`

4.3. Experiments on ASR Rescoring

We also test our soft binarization method on the n-bset rescoring task. Our baseline language model contains one layer LSTM with 600 units. The dropout rate is set to 0.2 on the SMS corpus. For the SWB corpus, we do not apply the dropout technique. The batch size is set to 64. Combination weight α is set to 0.5 in the knowledge distillation progress.

Table 3 shows the perplexity results and word error rate (WER) results for binarized models on the SMS and SWB corpus. Reported perplexities on the evaluation set are calculated without using context across sentence boundaries, which is consistent with the speech recognition setup. With distillation based technique, both BELM and FBLM perform better than models only trained with the original data by a large margin. On the SMS corpus, distilled models outperform LSTM LM on both PPL and WER measure with compression ratios of 10.7× and 30.4×. On the SWB dataset, distilled BELM and FBLM perform better than baseline LSTM in terms of perplexity and get identical word error rates, with 9.0× and 30.0× memory savings. The results show that our proposed language model compression method works well on ASR rescoring tasks with significant memory reduction.

	Model	PPL		WER		Size (byte)
		NLL	+KD	NLL	+KD	
SMS	LSTM	89.6	88.9	10.3	10.3	197.5M
	BELM	92.9	84.8	10.3	10.1	18.5M
	FBLM	95.0	85.6	10.5	10.1	6.5M
SWB	LSTM	57.9	55.4	12.3	12.3	153.0M
	BELM	57.8	55.0	12.6	12.3	17.0M
	FBLM	60.0	57.3	12.5	12.3	5.1M

Table 3. Test perplexity and word error rate (%) results for LSTM, BELM and FBLM on SMS and SWB dataset.

4.4. Analysis on Binarized Language Models

To further verify the contribution of our proposed binarized language model framework, we conduct ablation experiments on the PTB dataset. All the ablation experiments exclude the usage of knowledge distillation to avoid distractive factors. Compared with the first row, each of the following experiments in the table only changes one factor.

Model	PPL	Size (byte)
Small BELM	92.8	3.9M
- Normalization	93.2	3.9M
- Output projection	102.7	3.5M
- Scaling vectors	99.2	3.8M
+ Input projection	95.1	4.2M

Table 4. Ablation analysis on PTB dataset. Each row changes only one factor compared to the first row.

Word	Model	Top ten similar words
monday	LSTM	thursday tuesday friday wednesday oct. yesterday late last dec. aug.
	BELM	thursday tuesday wednesday friday oct. dec. yesterday aug. july nov.
	BELM+KD	tuesday wednesday thursday friday oct. yesterday aug. sept. dec. nov.
more	LSTM	less better worse greater nearly roughly about almost faster easier
	BELM	less worse better greater larger closer different bigger substantial stronger
	BELM+KD	less better closer greater bigger worse easier smaller cheaper faster
first	LSTM	second latest third sixth sharpest biggest smallest worst fourth seventh
	BELM	third second latest fourth father fifth fiscal u.s. same october
	BELM+KD	second third latest most same worst second-quarter current main fifth

Table 5. Top ten semantically similar words to a given word for small LSTM, BELM and BELM+KD on PTB dataset.

In the second row, weights are quantized into $\{+1, -1\}$ instead of $\{+\frac{1}{\sqrt{H}}, -\frac{1}{\sqrt{H}}\}$. Although models trained with two methods produce similar perplexity, BELM converges faster when weights are quantized into a smaller scale. It takes about 40 epochs for BELM trained without normalization to converge and the PPL after one epoch training is 580. In contrast, the model on the first row takes 10 fewer epochs to reach convergence, and the PPL after one epoch training is only 231.

Next, we evaluate the importance of inserting a small number of real-valued scalars in the binarized language models. On the third and fourth row, we remove the additional real-valued projection layer and the real-valued vectors separately. Experimental results show that the performance of the resulting models degrades dramatically with little memory savings. This demonstrates the effectiveness of our binarized language model framework. The last row adds another real-valued linear layer after the binarized input embedding layer as in [16], which proves to be redundant.

On table 6 our method is compared with several existing language model binarization methods. All the models are LSTM LM of size 300. First we compare evaluation set perplexity for models with binarized embeddings. Our soft BELM has a substantial improvement over other compressed models. Since BinaryConncet and BWN already show poor performance on binarizing embedding layers, we do not apply them on further binarizing the recurrent weights. We reproduce the BELM and BLLM as described in [16]. It is notable that in their original paper hidden state is not passed between subsequent sentences. Their model degrades sharply in our settings where successive mini-batches are traversed sequentially. This proves that our approach better captures long-term dependency than their method. Our soft FBLM also outperforms [18] using 2-bit or 3-bit quantization in terms of perplexity and the memory size. Meanwhile, the compression ratio for [17] is fairly low since they only apply binarization on the hidden layers.

Training binarized embeddings is meaningful since calculating the cosine similarity between two binarized vectors is more convenient. On table 5 we investigate the quality of binarized embeddings. Given a specific word in PTB vocabulary, we search the learned output embedding matrix for ten most similar words to this word. The table shows that the embedding layer of both the BELM and the BELM+KD model maintain semantic information. For instance, 'Monday' is close to words that refer to dates in all the models. It appears that BELM+KD learns better word representations than plain BELM since it learns from both original data and the teacher model output. In BELM+KD, words close to 'more' are all comparative adjectives, which is not the case with BELM. Also, it might be unreasonable for BELM to regard 'father' or 'u.s.' as related words of 'first'.

Model	PPL	Size (byte)
Small LSTM (baseline)	86.7	25.7M
BinaryConncet [13]	133.2	3.5M
BWN [14]	106.9	3.5M
BELM [16]	94.0	4.2M
Soft BELM (ours)	**85.2**	3.9M
BLLM [16]	106.7	0.8M
Alternating LSTM, $k = 2$ [18]	103.1	1.7M
Alternating LSTM, $k = 3$ [18]	93.8	2.6M
Binary weights LSTM [17]	92.2	23.0M
Soft FBLM (ours)	**88.1**	0.9M

Table 6. Comparison of several binarization methods on PTB dataset.

5. CONCLUSIONS AND FUTURE WORK

This paper proposes a novel soft binarization method for NN LM compression. With our approach, the performance gap between the full precision model and the compressed model is largely closed. Experiments on the PTB dataset show that binarized language models generalize well over long sequences. By using a small number of scalars and learning from soft labels generated by the teacher model, BELM is consistently better than full precision LSTM LM while requiring 32$\times$ fewer parameters in the embedding layers. And the distilled FBLM can obtain an overall compression rate of 30 with negligible loss in terms of PPL and WER measure. Our future work will concentrate on the combination of binarization with other compression techniques.

6. REFERENCES

[1] Thorsten Brants, Ashok C Popat, Peng Xu, Franz J Och, and Jeffrey Dean, "Large language models in machine translation," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[2] Qi Liu, Lijuan Wang, and Qiang Huo, "A study on effects of implicit and explicit language model information for dblstm-ctc based handwriting recognition," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 461–465.

[3] Lalit R Bahl, Peter F Brown, Peter V de Souza, and Robert L Mercer, "A tree-based statistical language model for natural language speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7, pp. 1001–1008, 1989.

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

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[6] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[7] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[8] Artem M Grachev, Dmitry I Ignatov, and Andrey V Savchenko, "Neural networks compression for language modeling," in *International Conference on Pattern Recognition and Machine Intelligence*. Springer, 2017, pp. 351–357.

[9] Patrick Chen, Si Si, Yang Li, Ciprian Chelba, and Cho-Jui Hsieh, "Groupreduce: Block-wise low-rank approximation for neural language model shrinking," in *Advances in Neural Information Processing Systems*, 2018, pp. 10988–10998.

[10] Xiang Li, Tao Qin, Jian Yang, and Tieyan Liu, "Lightrnn: Memory and computation-efficient recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4385–4393.

[11] Yunchuan Chen, Lili Mou, Yan Xu, Ge Li, and Zhi Jin, "Compressing neural language models by sparse word representations," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, vol. 1, pp. 226–235.

[12] Kaiyu Shi and Kai Yu, "Structured word embedding for low memory neural network language model," *Proc. Interspeech 2018*, pp. 1254–1258, 2018.

[13] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.

[14] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[15] Lu Hou, Quanming Yao, and James T. Kwok, "Loss-aware binarization of deep networks," in *International Conference on Learning Representations*, 2017.

[16] Xuan Liu, Di Cao, and Kai Yu, "Binarized lstm language model," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, vol. 1, pp. 2113–2121.

[17] Arash Ardakani, Zhengyun Ji, Sean C. Smithson, Brett H. Meyer, and Warren J. Gross, "Learning recurrent binary/ternary weights," in *International Conference on Learning Representations*, 2019.

[18] Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha, "Alternating multi-bit quantization for recurrent neural networks," in *International Conference on Learning Representations*, 2018.

[19] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 535–541.

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

[21] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al., "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[22] Mikael Boden, "A guide to recurrent neural networks and backpropagation," *the Dallas project*, 2002.

[23] Yiwen Guo, Anbang Yao, Hao Zhao, and Yurong Chen, "Network sketching: Exploiting binary structure in deep cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5955–5963.

[24] Shu-Chang Zhou, Yu-Zhi Wang, He Wen, Qin-Yao He, and Yu-Heng Zou, "Balanced quantization: An effective and efficient approach to quantized neural networks," *Journal of Computer Science and Technology*, vol. 32, no. 4, pp. 667–682, 2017.

[25] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[26] Yoshua Bengio, Nicholas Léonard, and Aaron Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[27] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[28] Ofir Press and Lior Wolf, "Using the output embedding to improve language models," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017, pp. 157–163.

[29] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov, "Fasttext. zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

[30] Asit Mishra and Debbie Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," *arXiv preprint arXiv:1711.05852*, 2017.

[31] Kazuki Irie, Zhihong Lei, Ralf Schlüter, and Hermann Ney, "Prediction of lstm-rnn full context states as a subtask for n-gram feedforward language models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6104–6108.

[32] Yangyang Shi, Mei-Yuh Hwang, Xin Lei, and Haoyu Sheng, "Knowledge distillation for recurrent neural network language modeling with trust regularization," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7230–7234.

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