# Compress Polyphone Pronunciation Prediction Model with Shared Labels

Pengfei Chen(✉), Lina Wang, Hui Di, Kazushige Ouchi, and Lvhong Wang

Research and Development Center, Toshiba, China
{chenpengfei,wanglina,dihui}@toshiba.com.cn,
kazushige.ouchi@toshiba.co.jp, wlv1990@gmail.com

**Abstract.** It is well known that deep learning model has huge parameters and is computationally expensive, especially for embedded and mobile devices. Polyphone pronunciations selection is a basic function for Chinese Text-to-Speech (TTS) application. Recurrent neural network (RNN) is a good sequence labeling solution for polyphone pronunciation selection. However, huge parameters and computation make compression needed to alleviate its disadvantages. Meanwhile, Large-scale-labels classification leads to more complicated network and heavy computation cost. In contrast to existing quantization with low precision data format and projection layer, we propose a novel method based on shared labels, which focuses on compressing the fully-connected layer before Softmax for models with a huge number of labels in TTS polyphone selection. The basic idea is to compress large number of target labels into a few label clusters, which will share the parameters of fully-connected layer. Furthermore, we combine it with other methods to further compress the polyphone pronunciation selection model. The experimental result shows that for Bi-LSTM (Bidirectional Long Short Term Memory) based polyphone selection, shared labels model decreases about 52% of original model size and accelerates prediction by 44% almost without performance loss. It is worth mentioning that the proposed method can be applied for other tasks to compress model and accelerate calculation.

**Keywords:** Bi-LSTM · Polyphone pronunciation prediction · Model compression · Shared labels

## 1 Introduction

Polyphone pronunciation prediction is a basic module of G2P (Grapheme-to-Phoneme) in Chinese Text-to-Speech (TTS) system, which provides the right pronunciation for Chinese character. The algorithms of polyphone pronunciation prediction include dictionary-based algorithm, statistical machine learning-based algorithm like Conditional Random Field (CRF) in (Lafferty et al.) [4], and deep learning-based algorithm like RNN. Dictionary-based method may fail for polyphone words problem, such as "朝阳" can be read as "chao2yang2" and


M. Sun et al. (Eds.): CCL 2020, LNAI 12522, pp. 403–414, 2020.
https://doi.org/10.1007/978-3-030-63031-7_29

"zhao1yang2", and Out of Vocabulary (OOV) problem. CRF and RNN perform well for polyphone pronunciation selection with context features. However, CRF needs manually designed context features. Neural network always has more parameters, larger model size and more expensive computation cost. In the application for embedded device, small model size and quick computation are necessary. So compression and acceleration of neural network is a hot research field in recent years.

There are several methods to compress deep learning model: low precision of data format, quantification, pruning, low rank factorization, and knowledge distillation. Low precision of data format replaces double or float with 16-bit float, which can reduce model size by quarter or half, but it cannot reduce running time when corresponding computations are not supported by existing instruction sets. Quantification and pruning methods pack some weights into one and prune the weights close to zero. It can reduce the model size but cannot accelerate the computation. Low rank factorization changes the network structure by factorizing large matrix into small matrixes. It can reduce the model size and accelerate the computation.

Our interest focuses on how to compress model that contains huge number of labels, and each input has a fixed label set. For example, each polyphone Chinese character has several (less than 10) fixed pronunciations. If we take all pronunciations of polyphone characters as target labels, there are too many parameters in fully-connected layer before Softmax. Therefore, the computation of fully-connected layer and Softmax will be costly. One idea is to use 1 character-1 model method, which can reduce the network complexity and computation cost, but it needs larger memory because of too many polyphone characters. This inspired us how we can share parameters in a single model which has comparable size to one-character model.

We propose a method to share parameters by means of different characters sharing pronunciation labels. We randomly assign labels for each character's pronunciation under conditions of avoiding label conflicts and assuring that the target label set is small. This yields smaller model size and less computation cost. Then we train our Bi-LSTM model with newly tagged corpus, and compare it with other compression methods in model size, memory usage and decoding time. The experimental results show our method can compress the model to half size and accelerate computation speed while maintaining a comparable performance compared with original model, overcoming the problem of too many labels.

## 2 Related Work

LSTM (Hochreiter and Schmidhuber) [7] is excellent in sequence labeling by learning contextual information. Bi-LSTM can use the future and history information to improve performance. LSTM encodes the embedding of input sequentially into a vector, which will keep the history information. Bi-LSTM will concatenate the vectors of forward LSTM and backward LSTM, which can utilize future and history information. Followed by fully-connected layer and Softmax,

Bi-LSTM model can predict the label with the highest probability. Lample et al. [3] presented a neural architectures based on Bi-LSTM and CRF to predict name entity. Cai et al. [12] described a system composed of Bi-LSTM acting as an encoder and a prediction network for Chinese polyphone pronunciation prediction. The output size equals the number of all possible pronunciations of polyphone character.

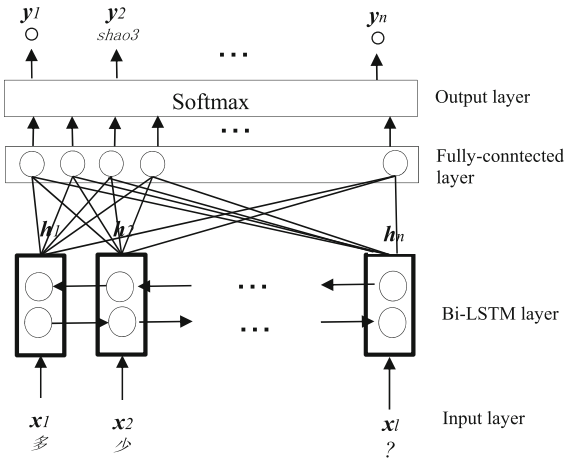


**Fig. 1.** Polyphone pronunciation sequence labeling by Bi-LSTM

There is a long history of model compression in Nature Language Processing (NLP) tasks. Many compression approaches have been proposed, including low precision data format and quantization, network pruning and parameter sharing, tensor decomposition, knowledge distillation. We briefly review the most popular methods in this section.

**Low Precision Data Format and Quantization.** Low precision data format can reduce the model size exponentially. Quantization compresses value to a less bits data to reduce the number of bytes of the weight parameter. For example, we can compress a float value to an 8-bit integer, one of 256 equally-sized intervals within the range. 16-bit, 8-bit, and even 1-bit quantization were proposed to compress network. Gupta et al. [8] used 16-bit wide fixed-point number representation to train neural network, without degradation in the accuracy of classification tasks. Courbariaux and Bengio [5] proposed a binarized Neural Networks (BNNs) with binary weights and activations, which can drastically reduce memory size, without any loss in classification accuracy.

**Network Pruning and Parameter Sharing.** Pruning can remove the parameters below threshold from network. Parameter sharing groups weights into hash brackets for sharing. Network pruning and parameter sharing not only can reduce the structure complexity of model, but also can improve generalization of network.

Grachev et al. [1] introduced pruning to network compression for language modeling, and compared performance and model size with baseline model. Han et al. [9] trained a network to learn which connections are important firstly. Then they pruned the unimportant connections and retrained the network to fine tune the weights of the remaining connections. Their method improved the energy efficiency and storage of neural networks without affecting accuracy. Chen et al. [10] presented HashedNets which used a low-cost hash function to group weights into hash buckets, and all weights within the same hash bucket share a single parameter value.

**Tensor Decomposition.** Tensor decomposition approaches can factorize weight matrix into smaller matrixes to reduce the number of parameters, which can compress the size of model and accelerate the calculation. Grachev et al. [1] compared low-rank (LR) factorization and tensor train (TT) decomposition in LSTM compression in language modeling. In their result, LR LSTM 650-650 is the most useful model for practical application.

**Knowledge Distillation.** Hinton et al. [2] first proposed knowledge distillation to transfer knowledge from teacher model to student model. This method can be used to improve the compressed model (student model) by exploiting original model (teacher model).

## 3 Shared Labels Model

### 3.1 Framework

Next we will introduce how to implement our novel proposal. For polyphone Chinese characters, there are as many as 1304 pronunciations in our corpus. 1305 labels, including "O" for non-polyphone characters, are used in our original model, which leads to the facts that the fully-connected layer before Softmax contains a large proportion of the weights and Softmax computation is costly. Considering the number of any polyphone pronunciations is less than 10, only 10 shared labels are used in our novel proposal to reduce the memory usage and computational cost.

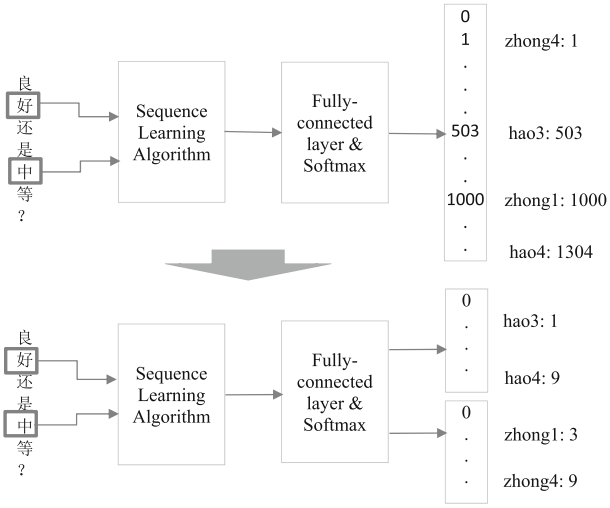


**Fig. 2.** Illustration of shared labels model

In our proposal, we use a character-based model for polyphone pronunciation prediction. We take a sentence as the input, such as "良好还是中等?". Each character's embedding comes from pre-trained embedding model. The inputs are fed into sequence learning algorithm to learn semantics and features, and the output label is predicted with 1 fully-connected layer and Softmax. The sequence learning method includes but not limited to Bi-LSTM. For the output layer, the original 1305 labels are mapped to 10 digital shared labels. For example, labels "hao3" and "hao4" for "好" are mapped to label-1 and label-9 respectively. The prediction is based on the 10 shared labels.

The digital labels for different polyphone characters can be the same but their real pronunciations may be different. For example, in the above sentence, "好 hao4" and "中 zhong4" share the digital label "9". We can translate the digital shared labels into real pronunciations with a dictionary based on the method of the following section.

### 3.2 Theoretical Analysis

Theoretically, the framework can reduce parameter number and accelerate computation speed. We use Bi-LSTM as sequence learning algorithm. Given sequence length $L$, character embedding size $V$, LSTM hidden size $H$, target label number $N$, the parameter number of Bi-LSTM layer is

$$N_1 = 2 * 4 * (V * H + H * H + H) \tag{1}$$

The parameter number of Fully-connected layer is

$$N_2 = 2 * H * N \tag{2}$$

The parameter number of model is the sum of these two parts:

$$N_{para} = N_1 + N_2 \tag{3}$$

For our polyphone pronunciation prediction task, the model is always not complex. In our implementation, we set $L$ as 100, $V$ as 100, $H$ as 200 and $N$ as 1305. The total parameter number of baseline model is 1,003,600, and that of shared labels model is 485,600, which reduces about 52% of size.

We take multiply-accumulate operations (MACCs) as measure of computations. One MACC includes one multiplication and one addition.

For vector multiplication:

$$y = w_0 * x_0 + w_1 * x_1 + \cdots + w_{n-1} * x_{n-1} \tag{4}$$

$w$ and $x$ are two vectors, result $y$ is a scalar.

A dot-product between two vectors of size $n$ uses $n$ MACCs. For a sequence of length $L$, the total MACCs of our model is

$$N_{MACCs} = L * [2 * 4 * (V + H) * H + 2 * H * N] \tag{5}$$

According the Eq. (5), the total MACCs of baseline model is 100.2 million. In our shared labels method, the total MACCs is 48.4 million, reducing about 52% compared with the baseline model.

If we take Floating Point Operations (FLOPs) as measure, there will be more reduction in computation because of less operations in Softmax.

### 3.3 Modules of Polyphone Pronunciation Selection with Shared Labels

In the training phase, we need to convert pronunciation to digital label. As mentioned above, we map 1305 labels to 10 shared labels. The mapping relations are saved in two dictionaries consisting of character-pronunciation-label_ID and character-label_ID-pronunciation. Details will be described in the next section. Then we handle the corpus with shared labels. We train Bi-LSTM model with pre-processed corpus.

In inference phase, we get the digital label from the prediction of model. Then we replace the digital label with real pronunciation by looking up dictionary.

**Assign Pronunciations to Label Clusters.** The relation between polyphone characters and their pronunciations is *N*:*N*. So it is important to map 1305 pronunciations to 10 shared labels. We take ID 0 for non-polyphone characters, whose pronunciation can be determined by looking up dictionary. Meanwhile, keeping balanced data number in each shared label will benefit to training speed and performance of the model. We assume that the label IDs subject to random distribution.

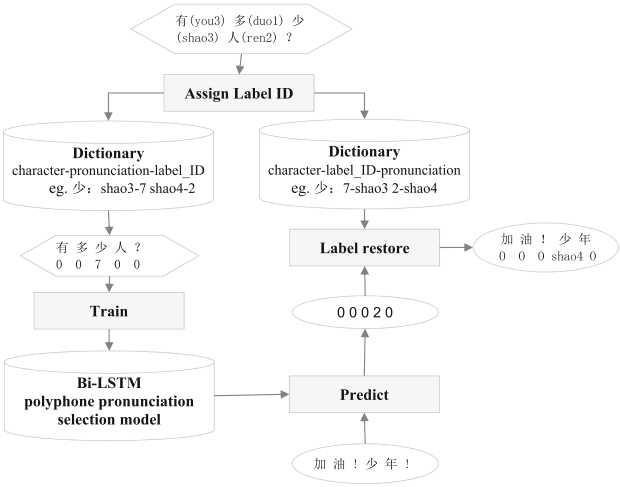


**Fig. 3.** Modules of polyphone pronunciation selection with shared labels model

The algorithm is as follow.

**Algorithm 1.** Assign Label ID for pronunciation

```
1: for char in poly_chars do
2:   for pron in char's prons do
3:     rand_id = randint(1,9)
4:     for char in homophone chars with this pron do
5:       for c_pron in char's prons do
6:         if c_pron.label_id == rand_id then
7:           goto 3
8:         else
9:           continue
10:        end if
11:        return rand_id
12:      end for
13:    end for
14:  end for
15: end for
```

Firstly, traverse each character's pronunciations (line 1,2), and assign a label randomly (line 3). If the label is the same with that of other pronunciations of current character and related homophone characters (line 4–6), reassign it randomly (line 7). Repeat this process until all pronunciations are assigned to a certain label. Because of randomness of label assignment, the labels distribution keeps balance.

## 4 Experiments

In this section, we compare our proposal shared labels model with standard Bi-LSTM model. Besides, we also test shared labels model combining with other methods, such as low precision float, projection layer. We compare their performance, memory usage, and speed respectively.

### 4.1 Data

Cai et al. [12] did their experiments with a public polyphonic character dataset, but it was unavailable when we tried to use it. So we use our own data as train and test sets. We have 188k sentences labeled with their pronunciations. We randomly select 1000 sentences as test set, and others as train set. There are 1127 polyphone characters in our corpus consisting of 1305 pronunciations (labels). Other Chinese characters are non-polyphone, which are labeled as "O", and their pronunciations are got by looking up dictionary.

### 4.2 Experimental Settings

Our experiments are done in tensorflow-GPU[1] version (train) and tensorflow-CPU version (test). Our CPU is Intel Xeon E5, and it does not support AVX512.

**Baseline Bi-LSTM Model.** The structure of the network is the same as in Sect. 3.2. In the training phase, we set the batch size to 16, learning rate to 0.1, and the dropout rate to 0.2. We adopt gradient descent optimization to learn the parameters.

**Shared Labels Model.** We have the same setting with baseline model, except the output size is 10.

**Bi-LSTM with Projection Layer.** For further compression, we add a projection layer between input layer and Bi-LSTM layer. Projection layer can factorize the big matrix into small matrixes, which can save memory and reduce the number of parameters.

[1] https://github.com/tensorflow/tensorflow.

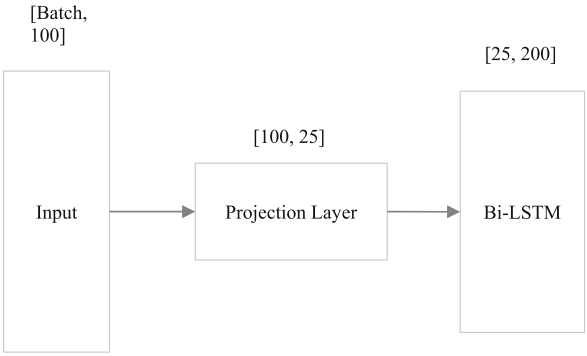


**Fig. 4.** Bi-LSTM with projection layer

**Knowledge Distillation.** Transfer learning is a promising method to improve the performance of compressed and simplified networks. We take 16-bit model with projection layer as student model and baseline as teacher model. We adopt the fusion of soft target and hard target as learning object.

**Shared Labels in CRF.** CRF is a statistical-based machine learning algorithm, which is popular used in sequence labeling problems. We use it to implement the polyphone selection with shared labels to check if it is workable.

### 4.3 Experiment Results

**Model Size and Performance.** We compare the models by F1-score and file size.

**Table 1.** F1-score and model size for different models

| Model | F1-score | Model size(Kb) |
|---|---|---|
| Baseline (Bi-LSTM) | 96.86 | 3925 |
| + Shared | 96.78 | 1897 |
| + 16bit | 96.85 | 1963 |
| + 16bit + Shared | 96.75 | 948 |
| + 16bit + PL | 94.40 | 1733 |
| + 16bit + PL + KD | 94.96 | 1733 |
| + 16bit + Shared + PL | 94.55 | 719 |

Note: Shared means shared labels model, PL means Bi-LSTM model with projection layer, KD means knowledge distillation.

From Table 1, we can see shared labels model is compressed to 48% of baseline model with no obvious performance loss. Combined with 16-bit float and shared label, the size of baseline model is further compressed to 24%, and F1-score only drops 0.11 point.

Compared with projection and knowledge distillation, shared model shows good result in model size reduction and keeps high performance at the same time.

a) By adding projection layer, 16-bit model is compressed a little, but the performance dropped a lot.
b) Knowledge distillation is useful to improve performance of projection model.
c) Compared with knowledge distillation, shared labels model has a much smaller size and comparable performance.
d) In general, shared labels outperforms projection in both model size and performance.

**Memory Usage.** We test the memory usage with open source tool Valgrind[2]. From Fig. 5, we can see that the memory usage of shared label model drops a lot compared with that of baseline model.

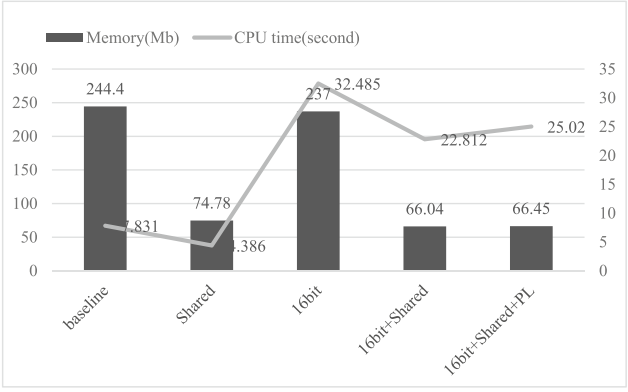


**Fig. 5.** Memory usage and CPU time of different models

**Decoding Time.** It takes less time for shared labels model compared with its counterpart model, with accelerating by 44% in Shared vs. baseline and 30% in 16bit+Shared vs. 16bit model. But for 16-bit model, the decoding is much slower than baseline, because our experiment machine does not support 16-bit float instructions.

[2] https://sourceware.org/git/?p=valgrind.git;a=summary.

**Result on Shared Labels in CRF.** For CRF with 1305 labels, the feature number is about 900 million which makes it unable to be loaded into memory as much as 256 GB. So the training cannot be continued. If we use 10 shared labels for CRF, the feature number is about 6,900,000, with an exponential reduction compared with previous model. The training speed is fast and its F1-score is as high as 0.9765 (Table 2).

**Table 2.** Result on CRF for polyphone pronunciation selection

| Model | F1-score | Model size (Kb) |
|---|---|---|
| CRF | NA (unable to train) | |
| Shared-CRF | 97.65 | 50292 |

## 5 Conclusion

We propose a novel shared labels method to compress polyphone pronunciation selection model. It decreases size of models consisting of huge number of labels by mapping labels to small shared labels. Our proposed method reduces the model size and memory usage remarkably, and accelerate decoding speed without performance loss compared with other methods.

In the future, we will verify the compressed model on embedded device and investigate other tasks which can apply this method.

## References

1. Grachev, A.M., Ignatov, D.I., Savchenko, A.V.: Neural networks compression for language modeling. In: Pattern Recognition and Machine Intelligence, vol. 10597 (2017)
2. Hinton, G., Dean, J., Vinyals, O.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop (2015)
3. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: Proceedings of NAACL-HLT, pp. 260–270 (2016)
4. John, L., Andrew, M., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In Proceedings ICML (2001)
5. Courbariaux, M., Bengio, Y.: Binarynet: training deep neural networks with weights and activations constrained to +1 or -1. CoRR, vol. abs/1602.02830 (2016)
6. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org (2015)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

8. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - vol. 37, ser. ICML 2015, pp. 1737–1746 (2015)
9. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, ser. NIPS 2015 (2015)
10. Chen, W., Wilson, J., Tyree, S., Weinberger, K.Q., Chen, Y.: Compressing neural networks with the hashing trick. In: JMLR Workshop and Conference Proceedings (2015)
11. Yu, C., Wang, D., Zhou, P., Zhang, T.: A survey of model compression and acceleration for deep neural networks. In: IEEE Signal Processing Magazine, Special Issue on Deep Learning for Image Understanding (2019)
12. Cai, Z., Yang, Y., Zhang, C., Qin, X., Li, M.: Polyphone disambiguation for mandarin Chinese using conditional neural network with multi-level embedding features. INTERSPEECH (2019)