Model Compression by Iterative Pruning with Knowledge Distillation and Its Application to Speech Enhancement

Zeyuan Wei1, Hao Li2, Xueliang Zhang1

1Department of Computer Science, Inner Mongolia University 2Department of Electrical and Electronic Engineering, Southern University of Science and Technology
weizeyuan@mail.imu.edu.cn, lih9@sustech.edu.cn, cszzx@imu.edu.cn

Abstract

Over the past decade, deep learning has demonstrated its effectiveness and keeps setting new records in a wide variety of tasks. However, good model performance usually leads to a huge amount of parameters and extremely high computational complexity which greatly limit the use cases of deep learning models, particularly in embedded systems. Therefore, model compression is getting more and more attention. In this paper, we propose a compression strategy based on iterative pruning and knowledge distillation. Specifically, in each iteration, we first utilize a pruning criterion to drop the weights which have less impact on performance. Then, the model before pruning is used as a teacher to fine-tune the student which is the model after pruning. After several iterations, we get the final compressed model. The proposed method is verified on gated convolutional recurrent network (GCRN) and long short-term memory (LSTM) for single channel speech enhancement tasks. Experimental results show that the proposed compression strategy can dramatically reduce the model size by 40x without significantly reducing the performance degradation for GCRN.

Index Terms: pruning, quantization, knowledge distillation, speech enhancement

1. Introduction

In the real environment, noise is inevitable and affects many applications, such as mobile communication and automatic speech recognition. Speech enhancement aims to reduce noise and improve speech quality and intelligibility [1]. Compared with the conventional algorithms [2], deep learning-based speech enhancement developed very fast and achieved great success [3]. However, the large model requires huge computing and memory resources, which restrict its real applications to embedded systems in industry, such as mobile phones and earbuds. Meanwhile, direct training of a small model has unsatisfactory performance.

In order to expand the application scenarios of deep learning, model compression is getting more and more attention recently [4, 5]. Model compression, as its name indicates, is to compress a large model to a small one which has similar performances. In general, compression techniques include quantization, low-rank approximation, network pruning and knowledge distillation. The quantization is to convert floating-point numbers into fixed-point numbers with fewer bits. The commonly used quantitative methods include linear, aware quantization proposed in [6] and k-means clustering quantization [7, 8]. Low-rank approximation [9] is to decompose a large parameter matrix into several small ones by using classic math, such as singular value decomposition [10]. Network pruning was proposed many years ago [11] and widely studied recently. It is based on the observation that many parameters in a large neural network are unimportant. Pruning method is to find those parameters through certain rules to increase the sparsity of the network. Common pruning strategies include one-shot pruning [12] and iterative pruning [13]. In [14], iterative pruning is used to compress speech enhancement models, and trainable parameters of the models were reduced by more than 70% without significantly reducing the enhancement performance. Depending on the granularity, neurons or connections, there are structured [15] and unstructured pruning [16]. Different from the above three compression methods, knowledge distillation [17, 18, 19] is a learning framework that utilizes a teacher-student structure. Generally speaking, the teacher network is an efficient neural network or a collection of neural networks. The student network is a compact neural network. The output of teacher network is used as label to train student network. In [20, 21], knowledge distillation is applied to speech enhancement models. But the purpose is not for model compression.

In this paper, we combine the characteristics of iterative pruning, quantization and knowledge distillation to compress speech enhancement networks. We believe that the model before each pruning is also useful for the restoration of the model after. Therefore, we use the model before pruning as the teacher model to fine-tune the student model after pruning, by combining the outputs of the teacher model with the true labels. After that, clustering-based quantization is employed as well to further reduce the size of the neural network.

In order to evaluate the proposed method, we choose two representative models, LSTM and Gated CRN [22], for speech enhancement. Experimental results show that the model size is greatly reduced about 10x and 40x for LSTM and GCRN respectively without significant performance loss.

The paper is organized as following. In section 2, we briefly introduce the deep learning-based speech enhancement and describe the proposed method of model compression. Experimental setup and results are given in section 3. We conclude the whole work in section 4.

2. Method

2.1. DNN for speech enhancement

Given a clean speech \( s \) and a noise \( n \), the noisy speech can be modeled as:

\[ m = s + n \] (1)

where \( \{m, s, n\} \in \mathbb{R}^{T \times 1} \) and \( T \) represents time samples, respectively. The purpose of single channel speech enhancement is to obtain estimated clean speech \( \hat{s} \), with a mapping function:

\[ \hat{s} = f_{\theta}(m) \] (2)
where \( f_\theta \) is the mapping function, indicates GCRN/LSTM in this paper. \( \theta \) indicates model weights, respectively.

Vanilla GCRN [22] can achieve a satisfactory result. However, the large size of the model and the amount of computation limit the model deployment in an embedded system. The LSTM-based model used in this paper consists of 2 unidirectional LSTM layers with 256 units each and 2 fully connected (FC) layers with 128 units each, with batch normalization between the last LSTM and first FC layers. ReLU activation is applied after the first FC layer and sigmoid after the second. Mean square error (MSE) is used as the loss function:

\[
L_n = |\bar{s} - \bar{s}|^2
\]  

(3)

### 2.2. Iterative pruning and knowledge distillation

Pruning methods include structured pruning and unstructured pruning. Unstructured pruning can achieve smaller loss and higher pruning ratio than structure pruning. Therefore, it is more suitable for hardware devices that cannot be accelerated. Commonly used pruning strategies include one-shot pruning [23] and iterative pruning [13]. One-shot pruning prunes the model once, resulting in the model losing too many parameters at once, and the model after one-shot pruning cannot be fine-tuned well. Iterative pruning is gradually pruning, which means the number of parameters reduces slowly each time, and the model will be fine-tuned well in the end. Therefore, this paper uses iterative pruning strategy to compress model. In addition, in order to increase the pruning ratio without degrading performance, \( l_1 \) regularization is adopted:

\[
\Gamma_1 = \frac{\lambda}{g(\bar{\omega})} \sum_{\omega \in \bar{\omega}} |\omega|\]

(4)

where, \( \bar{\omega} \) denotes the set of all non-zero weights, and \( \lambda \) is a predefined hyperparameter, respectively. \( g(.) \) is cardinality of a set. The new loss function can be described as:

\[
L_{l1} = L_n + \Gamma_1
\]

(5)

First, to determine the proportion of pruning, we analyze the sensitivity of each weight tensor. Second, unstructured pruning is executed according to the preset pruning rate. Third, knowledge distillation is used to fine-tune the model. The vanilla model is usually used as the teacher, but it can make a big gap between the teacher model and the student model.

Leading to a poor convergence for the student model. To reduce the gap, after one-time pruning, we adopt the model before each pruning as the teacher model and the model after each pruning as the student model. Using the teacher model to fine-tune the student model. The procedure is shown in Figure 1. The loss function for distillation is:

\[
\text{Algorithm 1: Iterative pruning and knowledge distillation}
\]

**Data:** 1. \( \bar{\omega}_j \): Indicates all non-zero weights in \( j \)-th weight tensor \( W_j \), \( \forall j \).
2. \( v \): Dataset used for validation.
3. \( \zeta \): Controlable loss threshold.
4. \( \eta(v, \theta) \): Indicates the loss obtained by input validation set \( v \) into the network with non-zero parameter \( \theta \).
5. \( f_\theta^t(.) \): \( f_\theta^t(.) \) indicates the network after each pruning, \( f_\theta^t(.) \) indicates the network before each pruning.
6. \( \rho_j \): pruning ratio for \( j \)-th weight tensor.

**Result:** Network after pruning and fine-tuning

1. for each iteration \( i \)
   2. for each weight tensor \( W_j \)
      3. \( \rho_j = 0\%
      4. Let \( \varphi \subseteq \bar{\omega} \) be the set of \( \rho\% \) of the smallest absolute value of non-zero weight in each weight tensor \( W_j \)
      5. \( s = \eta(v, \theta); \forall \omega \in \varphi \)  \( ) - \eta(v, \theta) \)
      6. if \( s < \zeta \) then
         7. \( \rho_j \equiv \rho_j + 5\%
      8. else
         9. break;
      10. end
     11. end
     12. end
13. end

Get \( \rho_j \) for weight tensor \( W_j \), \( \forall j \). \( \rho_j \) is used to prune the network. When the pruning is done, network is fine-tuned by knowledge distillation.

The loss function is described as:

\[
L = L_n + L_d + \Gamma_1
\]

942

Figure 1: Architectures of iterative knowledge distillation.
Table 1: Performance of GCRN after 1 to 5-th pruning and quantization.

<table>
<thead>
<tr>
<th>Test SNR</th>
<th>Mixure</th>
<th>STOI</th>
<th>PESQ</th>
<th>Test Noise</th>
<th>Babble</th>
<th>Cafeateria</th>
<th>STOI</th>
<th>PESQ</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5db</td>
<td>-0db</td>
<td>5db</td>
<td>AVG</td>
<td>-5db</td>
<td>0db</td>
<td>5db</td>
<td>AVG</td>
<td>-5db</td>
<td>0db</td>
</tr>
<tr>
<td>Mixture</td>
<td>58.4</td>
<td>71.4</td>
<td>82.9</td>
<td>70.9</td>
<td>56.8</td>
<td>71.1</td>
<td>82.0</td>
<td>70.0</td>
<td>1.50</td>
</tr>
<tr>
<td>Iter1</td>
<td>80.2</td>
<td>90.1</td>
<td>94.4</td>
<td>88.2</td>
<td>77.2</td>
<td>89.2</td>
<td>93.5</td>
<td>86.6</td>
<td>2.06</td>
</tr>
<tr>
<td>Iter2</td>
<td>80.2</td>
<td>90.1</td>
<td>94.5</td>
<td>88.3</td>
<td>77.1</td>
<td>89.1</td>
<td>93.5</td>
<td>86.6</td>
<td>2.03</td>
</tr>
<tr>
<td>Iter3</td>
<td>80.2</td>
<td>90.1</td>
<td>94.4</td>
<td>88.2</td>
<td>77.2</td>
<td>89.2</td>
<td>93.5</td>
<td>86.6</td>
<td>2.04</td>
</tr>
<tr>
<td>Iter4</td>
<td>79.7</td>
<td>90.0</td>
<td>94.4</td>
<td>88.1</td>
<td>76.6</td>
<td>89.0</td>
<td>93.4</td>
<td>86.4</td>
<td>2.03</td>
</tr>
<tr>
<td>Iter5</td>
<td>79.7</td>
<td>89.9</td>
<td>94.3</td>
<td>88.0</td>
<td>76.7</td>
<td>88.9</td>
<td>93.4</td>
<td>86.3</td>
<td>2.03</td>
</tr>
<tr>
<td>Quantization</td>
<td>78.3</td>
<td>89.0</td>
<td>93.8</td>
<td>87.1</td>
<td>75.4</td>
<td>88.3</td>
<td>92.9</td>
<td>85.5</td>
<td>1.98</td>
</tr>
</tbody>
</table>

Specific details are described in Algorithm 1.

The performance of the model after pruning. To verify its effectiveness, some experiments are conducted as follows. The perceptual evaluation of speech quality (PESQ) [26] and short-term objective intelligibility (STOI) [27] are used to evaluate speech quality and intelligibility, respectively. STOI values typically range from 0 to 1, which can be roughly interpreted as percent correct. PESQ values range from -0.5 to 4.5.

3.3. Experimental results

3.3.1 Effectiveness of knowledge distillation

Typically, pruning operations degrade model performance. The purpose of knowledge distillation in this paper is to fine-tune the performance of the model after pruning. To verify its effectiveness, some experiments are conducted as follows. Proposed from a sound effect library (available at https://www.soundideas.com) [25] are used. Specifically, we mix a randomly selected training utterance with a random cut from the 10000 noises. The SNR is randomly sampled between -5 and 0 dB. The training set includes 300,000 mixtures. Then, creating 30,000 mixtures as valid set by using the same method. For the test set, the two challenging noises (babble and cafeteria) from an Auditec CD (available at http://www.auditec.com) are used to generate 300 mixtures at each SNR of -5, 0, and 5dB.

The perceptual evaluation of speech quality (PESQ) [26] and short-term objective intelligibility (STOI) [27] are used to evaluate speech quality and intelligibility, respectively. STOI values typically range from 0 to 1, which can be roughly interpreted as percent correct. PESQ values range from -0.5 to 4.5.

2.3. Clustering-based quantization

We divide k-means to cluster the weights of each layer. After K-means converge, the network before each pruning, α is predefined weighting parameters, respectively. Therefore, the final loss can be defined as:

$$L = L_o + L_d + \Gamma_i$$

Specific details are described in Algorithm 1.

The cluster quantization strategy used in this paper is similar to the pruning strategy. Each weight tensor determines a cluster center as small as possible according to the quantification strategy in this paper. To a certain extent, the performance loss caused by quantization can be reduced by using different number clustering centers for each weight tensor. On the other hand, the network quantization is flexible with the use of cluster strategy. Refer to [14] for more details.

3. Experimental Results

3.1. Datasets

WSJ0 SI-84 [24], which includes 7138 utterances from 83 speakers (42 males and 41 females), is used to evaluate the proposed method. 77 speakers are used for training and remaining 6 are used for evaluation. For the training set, 10000 noises from a sound effect library (available at https://www.soundideas.com) [25] are used. Specifically, we mix a randomly selected training utterance with a random cut from the 10000 noises. The SNR is randomly sampled between -5 and 0 dB. The training set includes 300,000 mixtures. Then, creating 30,000 mixtures as valid set by using the same method. For the test set, the two challenging noises (babble and cafeteria) from an Auditec CD (available at http://www.auditec.com) are used to generate 300 mixtures at each SNR of -5, 0, and 5dB.

The perceptual evaluation of speech quality (PESQ) [26] and short-term objective intelligibility (STOI) [27] are used to evaluate speech quality and intelligibility, respectively. STOI values typically range from 0 to 1, which can be roughly interpreted as percent correct. PESQ values range from -0.5 to 4.5.

3.2. Experimental setup

For all experiments, we take a piece of speech through the sampling rate of 16kHz. The GCRN is trained in the frequency domain and according to the configuration described in [22]. For LSTM, we use 320 points FFT to get 161-D spectral features, then through a Mel-spectrogram transform matrix, the speech is mapped to a 128-dimensional mel space as input. The each pruning controllable loss threshold ζ is set empirically to 0.04, and the number of iteration i is set to 5, respectively. The distillation fine-tuning ratio α is set to 0.1. The configuration of quantization is the same as [14].

3.3. Experimental results

3.3.1. Effectiveness of knowledge distillation

Typically, pruning operations degrade model performance. The purpose of knowledge distillation in this paper is to fine-tune the performance of the model after pruning. To verify its effectiveness, some experiments are conducted as follows. Proposed
Table 2: Performance of LSTM after 1 to 5-th pruning and quantization.

<table>
<thead>
<tr>
<th>Test SNR</th>
<th>Mixture</th>
<th>Baseline</th>
<th>Iter1</th>
<th>Iter2</th>
<th>Iter3</th>
<th>Iter4</th>
<th>Iter5</th>
<th>Quantization</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5db</td>
<td>58.4</td>
<td>71.4</td>
<td>82.9</td>
<td>70.9</td>
<td>56.8</td>
<td>71.1</td>
<td>82.0</td>
<td>70.0</td>
</tr>
<tr>
<td>0db</td>
<td>85.4</td>
<td>85.4</td>
<td>91.9</td>
<td>83.4</td>
<td>71.2</td>
<td>84.6</td>
<td>90.9</td>
<td>82.2</td>
</tr>
<tr>
<td>5db</td>
<td>90.9</td>
<td>1.77</td>
<td>2.28</td>
<td>2.71</td>
<td>2.29</td>
<td>1.82</td>
<td>2.30</td>
<td>2.66</td>
</tr>
</tbody>
</table>

The proposed method has good performance in speech enhancement and model compression ratio.

3.3.3. Influence of distillation on model size

Table 3 presents the size model after each pruning by distillation, fine-tune, and non-distillation fine-tune. The results show that the model whose performance is fine-tuned by distillation obtained a higher pruning rate in the next pruning. This is because, with distillation to fine-tune, the model has better performance before the next pruning. For the same ζ, more non-zero weights can be removed. It suggests that using distillation to fine-tune the model not only achieves higher performance but also obtains a smaller model than the without distillation.

4. Conclusions

In this paper, knowledge distillation, iterative pruning, quantization are combined to compress model. After each pruning, we use the model before pruning as the teacher model to restore the student model after pruning by combining the outputs of the teacher model with the true labels. The experimental results show that the model before pruning is useful, and it is effective to use knowledge distillation to fine-tune the pruned model. It also proves that pruning a large model is better than training a small model directly.

3.3.2. Validity of the compression algorithm

To verify the effectiveness of the compression algorithm, we compress GCRN and LSTM networks, respectively. Table 1 and 2 show the performance of GCRN and LSTM networks after 5 iterations pruning and quantization at different SNR. Quantized performance is referred to as Quantization. As can be seen from Table 1, the model size for vanilla and 5-th iterations model is 37.27MB and 0.87MB, respectively, which means the compression rate is 40x, and the enhancement performance degrade rarely. From Table 3, the LSTM is compressed to 10x, and the performance is the same as before compression. The effectiveness of the proposed compression algorithm is proved by compress two networks.
5. References


