Compressing Deep Neural Networks for Efficient Speech Enhancement

Ke Tan and DeLiang Wang
The Ohio State University, United States

OUTLINE

1. Background and Motivations

2. Algorithm Description

3. Experiments

4. Conclusion

OUTLINE

1. Background and Motivations

2. Algorithm Description

3. Experiments

4. Conclusion

Background

- Increasing interest in deploying deep learning based enhancement systems for real-world applications and products.

- To achieve strong enhancement performance would require a large deep neural network (DNN).
  - computation and memory consuming
  - it is difficult to deploy such DNNs in latency-sensitive applications or on resource-limited devices

- It becomes an increasingly important problem to reduce memory and computation in DNNs for speech enhancement.

Motivations

- Two ways to derive a lightweight DNN:
  - to directly design a DNN with a small number of trainable parameters
  - to train a reasonably large DNN and then compress it

- Previous studies show that starting with training a large, over-parameterized DNN seems important for achieving high performance.

- It remains unclear for speech enhancement whether specific compression techniques are effective and how different techniques can be combined to achieve high compression rates.

- A generic compression pipeline for different speech enhancement models would be desired.

Motivations

- In this study, we developed two model compression pipelines for DNN-based speech enhancement.
- Each pipeline consists of three techniques: sparse regularization, iterative pruning and clustering-based quantization.
- Evaluation results show that the proposed approach substantially reduces the sizes of four different speech enhancement models, without significant performance degradation.

OUTLINE

1. Background and Motivations

2. Algorithm Description

3. Experiments

4. Conclusion

- A typical procedure of network pruning comprises three stages:
  - training a large DNN that achieves satisfactory performance
  - removing a specific set of weights in the trained DNN with a certain criterion
  - fine-tuning the pruned DNN

- The granularity of tensor sparsity impacts the efficiency of hardware architecture.
  - Fine-grained sparsity: individual weights are set to zero → **unstructured pruning**
    - difficult to apply hardware acceleration
  - Coarse-grained sparsity: groups of weights are set to zero → **structured pruning**
    - more hardware-friendly

- The key issue in network pruning is to define the pruning criterion, which determines the set of weights to be removed.

- For unstructured pruning, the importance of a specific set $\mathcal{U}$ of weights as the increase in the error induced by removing them: ($\mathcal{V}$ – validation set, Θ - set of all trainable parameters in the DNN)

$$\mathcal{I}_{\mathcal{U}} = \mathcal{L}(\mathcal{V}, \Theta | w = 0, \forall w \in \mathcal{U}) - \mathcal{L}(\mathcal{V}, \Theta).$$

- For structured pruning, the importance of a specific set $\mathcal{U}$ of weight groups as the increase in the error induced by removing them:

$$\mathcal{I}_{\mathcal{U}} = \mathcal{L}(\mathcal{V}, \Theta | \mathbf{g} = \mathbf{0}, \forall \mathbf{g} \in \mathcal{U}) - \mathcal{L}(\mathcal{V}, \Theta).$$

Algorithm

Iterative Pruning

Algorithm 1 Per-tensor sensitivity analysis for unstructured pruning

Input: (1) Validation set $\mathcal{V}$; (2) set $\mathcal{W}_l$ of all nonzero weights in the l-th weight tensor $\mathbf{W}_l$, $\forall l$; (3) loss function $\mathcal{L}(\mathcal{V}, \Theta)$, where Θ is the set of all nonzero trainable parameters in the DNN; (4) predefined tolerance value α_1.

Output: Pruning ratio β_l for weight tensor $\mathbf{W}_l$, $\forall l$.

for each tensor $\mathbf{W}_l$ **do**
    for β in $\{0\%, 5\%, 10\%, \ldots, 90\%, 95\%, 100\%\}$ **do**
        Let $\mathcal{U} \subseteq \mathcal{W}_l$ be the set of the $\beta(\%)$ of nonzero weights with the smallest absolute values in tensor $\mathbf{W}_l$;
        $\mathcal{I}_{\mathcal{U}} \leftarrow \mathcal{L}(\mathcal{V}, \Theta | w = 0, \forall w \in \mathcal{U}) - \mathcal{L}(\mathcal{V}, \Theta)$;
        if $\mathcal{I}_{\mathcal{U}} > \alpha_1$ **then**
            $\beta_l \leftarrow \beta - 5\%$;
            break
        end if
    end for
    if β_l is not assigned any value **then**
        $\beta_l \leftarrow 100\%$;
    end if
end for
return β_l for weight tensor $\mathbf{W}_l$, $\forall l$

Algorithm 2 Per-tensor sensitivity analysis for structured pruning

Input: (1) Validation set $\mathcal{V}$; (2) set $\mathcal{G}_l$ of all nonzero weight groups in the l-th weight tensor $\mathbf{W}_l$, $\forall l$; (3) loss function $\mathcal{L}(\mathcal{V}, \Theta)$, where Θ is the set of all nonzero trainable parameters in the DNN; (4) predefined tolerance value α_1.

Output: Pruning ratio β_l for weight tensor $\mathbf{W}_l$, $\forall l$.

for each tensor $\mathbf{W}_l$ **do**
    for β in $\{0\%, 5\%, 10\%, \ldots, 90\%, 95\%, 100\%\}$ **do**
        Let $\mathcal{U} \subseteq \mathcal{G}_l$ be the set of the $\beta(\%)$ of nonzero weight groups with the smallest ℓ_1 norms in tensor $\mathbf{W}_l$;
        $\mathcal{I}_{\mathcal{U}} \leftarrow \mathcal{L}(\mathcal{V}, \Theta | \mathbf{g} = \mathbf{0}, \forall \mathbf{g} \in \mathcal{U}) - \mathcal{L}(\mathcal{V}, \Theta)$;
        if $\mathcal{I}_{\mathcal{U}} > \alpha_1$ **then**
            $\beta_l \leftarrow \beta - 5\%$;
            break
        end if
    end for
    if β_l is not assigned any value **then**
        $\beta_l \leftarrow 100\%$;
    end if
end for
return β_l for weight tensor $\mathbf{W}_l$, $\forall l$

Sparse Regularization

- For unstructured pruning, we use ℓ_1 regularization to impose weight-level sparsity, which encourages less important weights to become zero, reducing the resulting performance degradation:

$$\mathcal{R}_{\ell_1} = \frac{\lambda_1}{n(\mathcal{W})} \sum_{w \in \mathcal{W}} |w|$$

where $\mathcal{W}$ denotes the set of all weights. The function $n(\cdot)$ calculates the number of elements in a set.

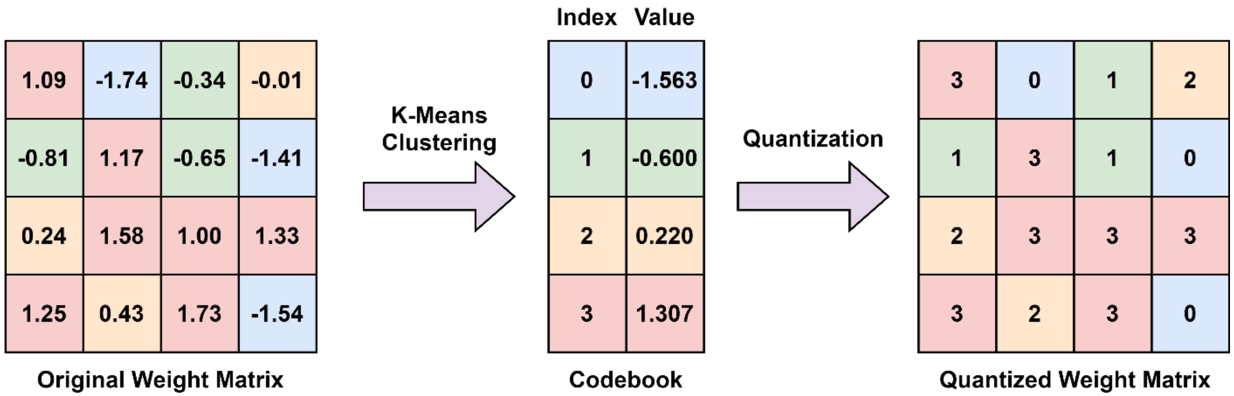
- For structured pruning, we use a group sparse regularizer [2]:

$$\mathcal{R}_{\mathrm{SGL}} = \frac{\lambda_1}{n(\mathcal{W})} \sum_{w \in \mathcal{W}} |w| + \frac{\lambda_2}{n(\mathcal{G})} \sum_{\mathbf{g} \in \mathcal{G}} \sqrt{p_{\mathbf{g}}} \, \|\mathbf{g}\|_2$$

where $\mathcal{G}$ denotes the set of all weight groups. The number of weights in each weight group $\mathbf{g}$ is represented by $p_{\mathbf{g}}$.

[1] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. A sparse-group lasso. Journal of Computational and Graphical Statistics, 22(2):231–245, 2013.

- To further compress the pruned DNN, we propose to use clustering-based quantization.
- Specifically, the weights in each tensor are partitioned into K clusters through k-means clustering.
- Once the clustering algorithm converges, we reset all the weights that fall into the same cluster to the value of the corresponding centroid.



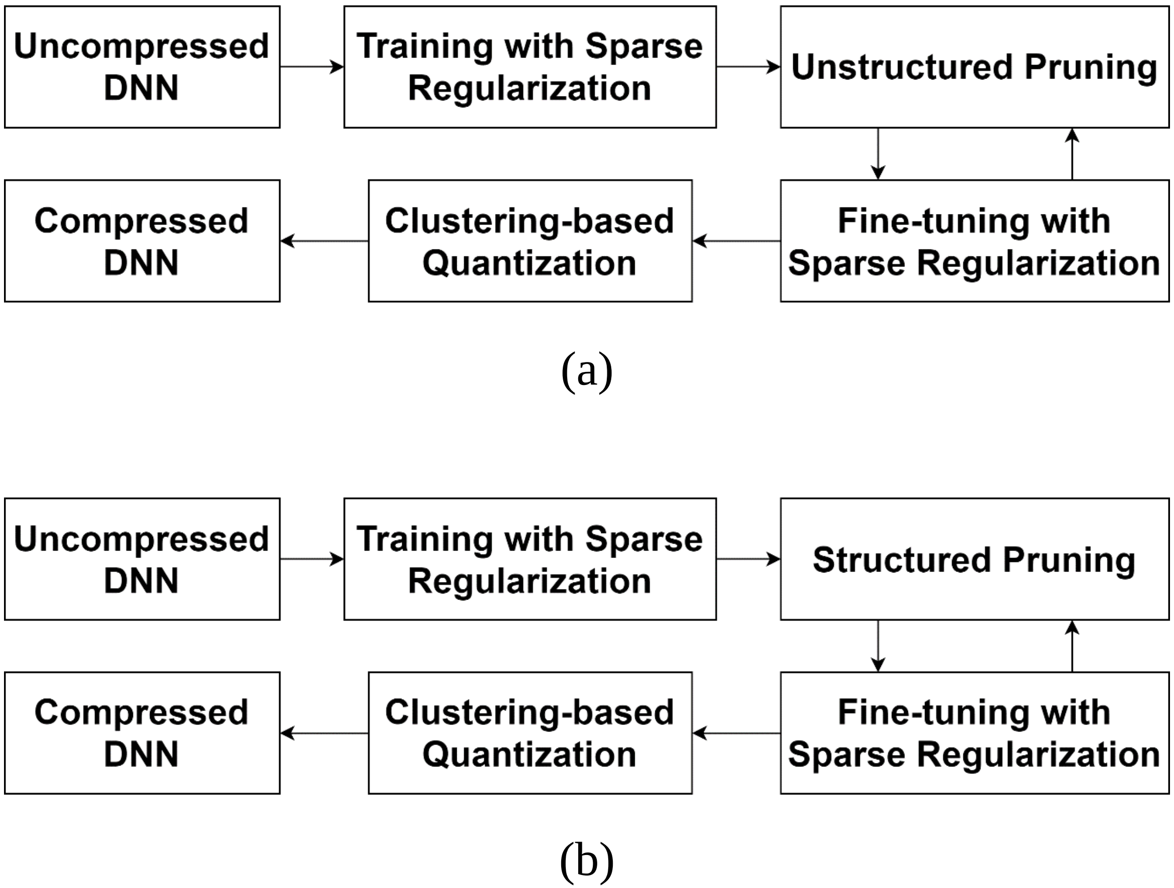


Fig. 1. Illustration of the proposed compression pipelines.

OUTLINE

1. Background and Motivations

2. Algorithm Description

3. Experiments

4. Conclusion

Data Preparation

- Speech corpus: training set of WSJ0, including 12776 utterances from 101 (= 89 + 6 + 6) speakers.

- (1) Training noises: 10,000 noises from a sound effect library. (2) Test noises: babble (BAB) and cafeteria (CAF) noises from an Auditec CD.

- Training set: 320,000 mixtures, SNR between -5 and 0 dB.

- Validation set: 846 mixtures, SNR between -5 and 0 dB.

- Test sets: 846 mixtures for each of the two noises and each of three SNRs (-5, 0, 5 dB).

Speech Enhancement Models

- We use four models with different designs including DNN types, training targets and processing domains.

- (1) **Feedforward DNN (FDNN)**: 3 hidden layers with 2048 units in each layer. Input: magnitude spectrogram. Target: IRM.
- (2) **LSTM**: 4 LSTM hidden layers with 1024 units in each layer, and the output layer is a fully-connected layer followed by ReLU function. The LSTM performs spectral mapping in the magnitude domain.
- (3) **Temporal convolutional neural network (TCNN) [2]**: time-domain enhancement.
- (4) **Gated convolutional recurrent network (GCRN) [3]**: complex spectral mapping.

[2] A. Pandey and D. L. Wang. TCNN: Temporal convolutional neural network for real-time speech enhancement in the time domain. In IEEE ICASSP, pages 6875–6879. IEEE, 2019.
[3] K. Tan and D. L. Wang. Learning complex spectral mapping with gated convolutional recurrent networks for monaural speech enhancement. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 28:380–390, 2020.

Table 1. Comparisons between pruned models and comparably-sized unpruned models.

Metric	STOI (%)			PESQ			# Param.
SNR	-5 dB	0 dB	5 dB	-5 dB	0 dB	5 dB	
Mixture	57.86	70.14	81.48	1.51	1.80	2.12	-
FDNN_P	63.72	77.63	86.59	1.63	2.06	2.44	1.15 M
FDNN_B	62.89	76.25	85.54	1.58	1.98	2.35	1.45 M
LSTM_P	75.76	86.62	92.25	1.98	2.46	2.85	2.93 M
LSTM_B	72.25	83.98	90.31	1.85	2.31	2.68	3.14 M

Table 2. Comparisons between uncompressed and compressed models.

Metric	STOI (%)			PESQ			Storage	CR
SNR	-5 dB	0 dB	5 dB	-5 dB	0 dB	5 dB		
Mixture	57.86	70.14	81.48	1.51	1.80	2.12	-	-
FDNN_U	64.87	78.64	87.25	1.65	2.09	2.47	34.54 MB	1×
FDNN_C	63.58	77.51	86.50	1.64	2.06	2.44	0.10 MB	343×
LSTM_U	75.74	86.47	92.04	2.00	2.47	2.84	115.27 MB	1×
LSTM_C	75.83	86.62	92.25	1.98	2.46	2.84	2.49 MB	46×
TCNN_U	79.76	89.72	93.96	2.04	2.52	2.86	19.28 MB	1×
TCNN_C	77.77	88.46	93.26	1.95	2.44	2.79	0.56 MB	34×
GCRN_U	81.03	90.43	94.43	2.14	2.65	3.01	37.27 MB	1×
GCRN_C	80.66	90.15	94.26	2.15	2.65	3.02	1.11 MB	34×
GCRN_C-SP	81.05	90.32	94.35	2.15	2.66	3.03	4.11 MB	9×

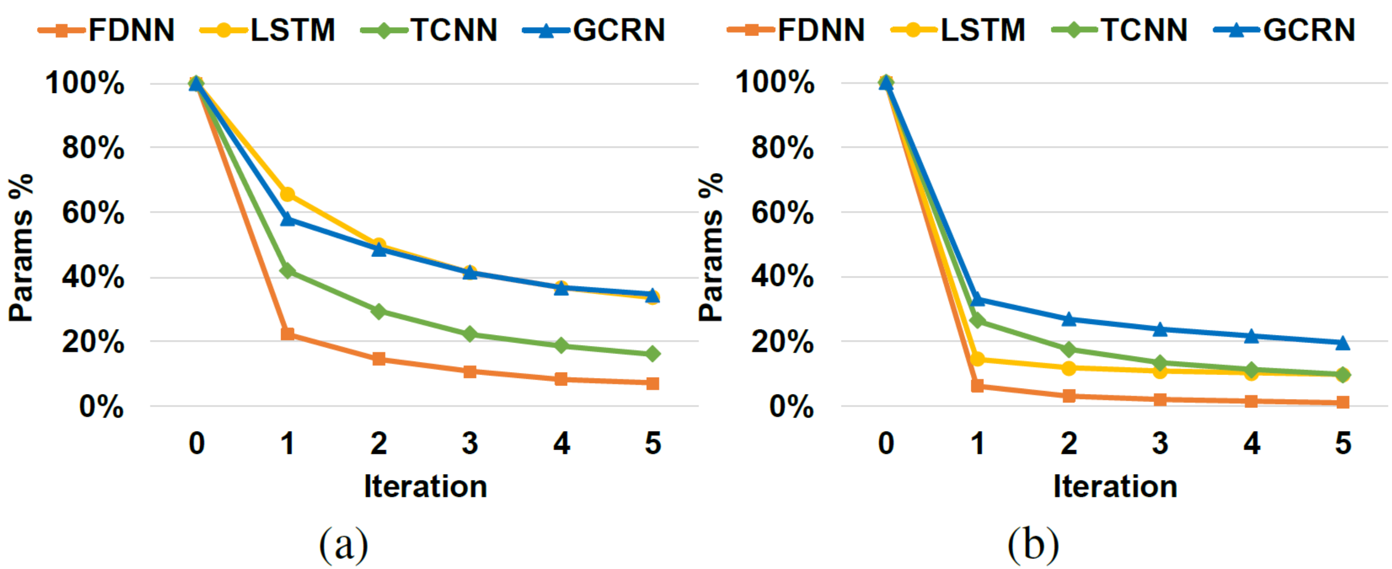


Fig. 2. The percent of the original number of trainable parameters at different pruning iterations. (a).Without, and (b).With sparse regularization. Note that unstructured pruning is performed.

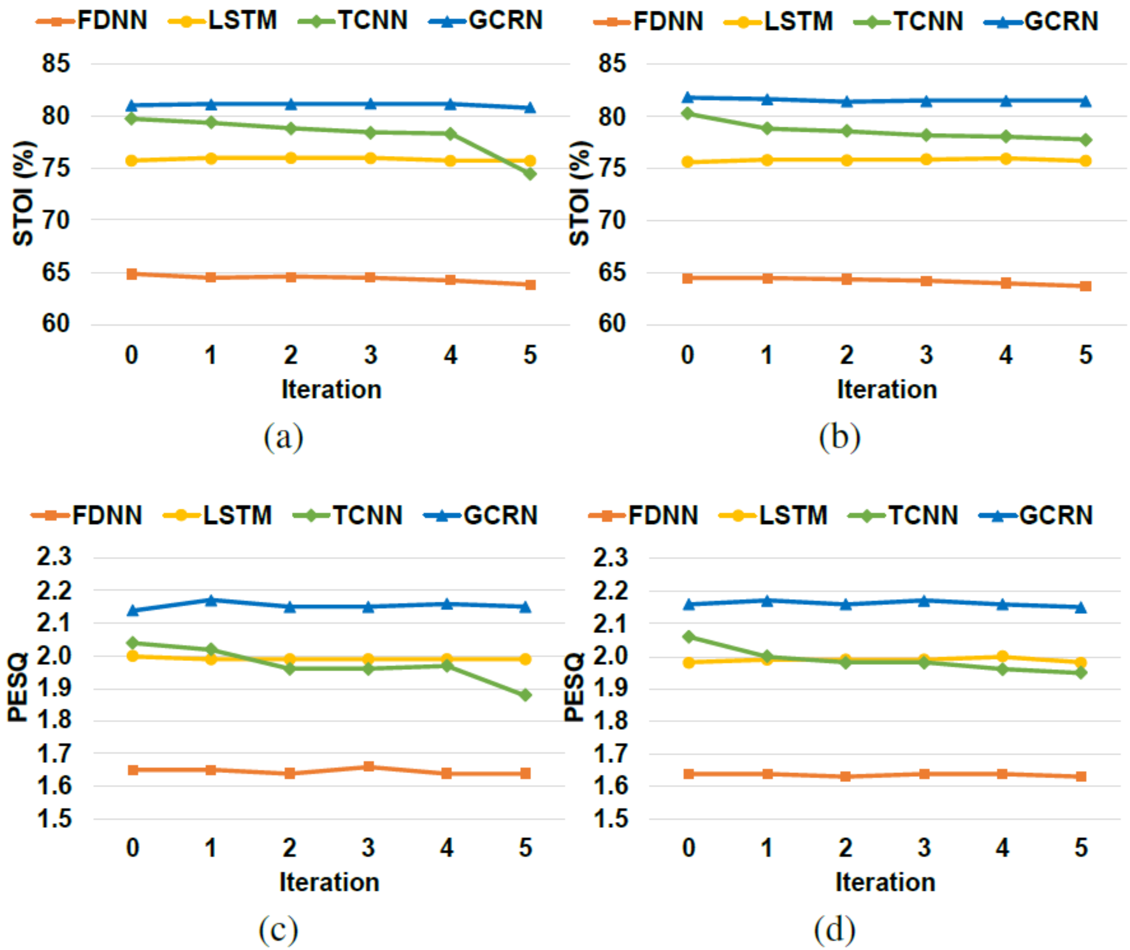


Fig. 3. STOI and PESQ scores for -5 dB SNR at different pruning iterations. (a)&(c). Without, and (b)&(d). With sparse regularization. Note that unstructured pruning is performed.

Demos

Babble, -5 dB

Unprocessed:

GCRN_U (37.27 MB):

GCRN_C (1.11 MB):

Clean:

Cafeteria, -5 dB

Unprocessed:

TCNN_U (19.28 MB):

TCNN_C (0.94 MB):

Clean:

OUTLINE

Conclusion

- In this study, we have proposed two new pipelines to compress DNNs for speech enhancement.
- Our experimental results show that the proposed pipelines substantially reduce the sizes of all the four models, without significant performance degradation.
- We also find that training and pruning an over-parameterized DNN achieves better enhancement results than directly training a small DNN that has a comparable size to the pruned DNN.