LEARNING EFFICIENT SPARSE STRUCTURES IN SPEECH RECOGNITION

Jingchi Zhang[⋆] *Wei Wen*[⋆] *Michael Deisher*[†] *Hsin-Pai Cheng* [⋆] *Hai Li*[⋆] *Yiran Chen*[⋆]

[⋆] Duke University, Durham, North Carolina, USA
[†]Intel Corporation, Hillsboro, Oregon, USA

ABSTRACT

Recurrent neural networks (RNNs), especially long short-term memories (LSTMs) have been widely used in speech recognition and natural language processing. As the sizes of RNN models grow for better performance, the computation cost and therefore the required hardware resource increase rapidly. We propose an efficient structural sparsity (ESS) learning method for acoustic modeling in speech recognition. ESS aims to generate a model that offers higher execution efficiency while maintaining the accuracy. A three-step training pipeline is developed in our work. First, we apply the group Lasso regularization method during training process and learn a structural sparse model from scratch. Then the learned sparse structures will be fixed and cannot be changed. Finally, we retrain the model and update the nonzero parameters in the model. We applied our ESS method on classic HMM+LSTM model on Kaldi toolkit. The experimental results show that ESS can remove 72.5% weight groups in the weight matrices when slightly increasing the word error rate (WER) 1.1%.

Index Terms— efficient structural sparsity, long short-term memory, acoustic modeling, speech recognition

1. INTRODUCTION

Recently, speech recognition models based on recurrent neural networks (RNNs), especially the long short-term memory (LSTM) models have become more popular in automatic speech recognition (ASR) [1]. Many different types of LSTM structures have been applied to various speech recognition tasks successfully, such as deep LSTM (DLSTM) [2], bidirectional LSTM (BLSTM) [3] and LSTM with recurrent projection layer (LSTMP) [4]. However, to further improve the model performance, common approaches are increasing the depth of the LSTM layers and and increasing the size of the matrices in LSTM model, which lead to higher computation cost and prolonged inference time.

In recent years, there have been extensive studies on accelerating deep neural networks (DNNs). For example, the pruning method [5] and sparsity regularization [6] can effectively reduce the volume of the weight parameters in convolutional neural networks (CNNs). Without considering the model structure, these methods zero out the elements below certain thresholds as they have negligible impact on the model accuracy. The produced matrix features *non-structural sparsity* as zero parameters are randomly distributed. Such non-structural sparse models can achieve only very limited computational speedup on hardware, even under a high sparsity level of more than 90% [7]. To bridge the discrepancy in algorithm and hardware implementation, Wen *et al.* [7] proposed a framework which can learn compact sparse structures such as filters, channels and layers in CNNs.

Learning sparse structures in RNNs is more difficult. In RNNs, all the time steps share a recurrent unit. Compressing the recurrent unit influences all the time steps. Thus, it is challenging to apply sparsity methods while maintaining acceptable accuracy. Sharan *et al.* [8] introduced a pruning method that can reduce 90% of the connections in RNN. However, the speedup of the actual computation could be limited due to the non-structural sparsity. The latest intrinsic sparse structural learning [9] is able to learn a sparsity structure in LSTM models and reduce the model size.

This work targets to improve the execution efficiency of acoustic modeling in speech recognition. We propose an efficient structural sparsity (ESS) learning method for HMM+LSTM based acoustic model. ESS attempts to learn sparse structures according to hardware requirements. ESS is applied on the training process by adding a purposely-designed group Lasso regularization term on the loss function. Using this new loss function, both accuracy and sparsity are optimized during the training process. As such, ESS produces models with a similar accuracy but in a very sparse structure and the computation is greatly accelerated on the hardware.

We tested the proposed ESS method on the classic HMM+LSTM model. The experiment using the nnet3 toolkit in Kaldi [10] showed that the new model trained by using our method can reach a sparsity level of 72.5% with 1.1% increase in the word error rate (WER).

2. RELATED WORK

There have been a lot of studies on removing redundant structures in CNNs. For example, by applying l_1 regularization on the loss function in training procedure, a large portion of

parameters in CNNs can be reduced [6]. Connection pruning method [5] can reduce 90% of the weight parameters and keep the accuracy at the similar level. Pruning methods have be successfully applied to RNNs [8] and compressed gated recurrent unit (GRU) models. Recently, the use of group Lasso regularization has been proved to be an effective way to remove structures in CNNs, such as neurons, filters, channels and layers [7]. The group Lasso based sparsity method also has been successfully applied to RNN and LSTM [9]. The approach can learn a structural sparse model and accelerate the computation without any specific hardware deployment.

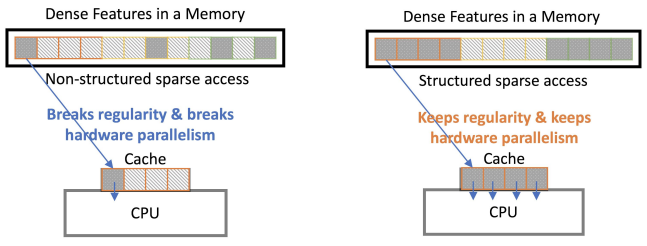
3. LEARNING EFFICIENT SPARSE STRUCTURES

3.1. Computational efficiency explanation

Conventional sparsity learning or pruning methods produce non-structural sparsity or connectivity, which results in poor data locality and therefore has limited or even negative acceleration on the computation.

Fig. 1 illustrates the difference in the data locality induced by non-structural sparsity model and structural sparsity model and the corresponding execution in CPU. It can also be applied on GPU because of the same properties. The dark gray blocks in the figure represent the features that are needed in the computation. The striped blocks denote the features that are not necessary in computation. In a non-structural sparse model, as illustrated in Fig. 1(a), when the CPU fetches the first needed feature, it automatically fetches all the features in the weight group into the cache for improving execution parallelism. Unfortunately, those features are not necessary in the computation. When another feature is needed in the next step, the data might not have been cached yet so the CPU needs to fetch it from memory. Compared to getting data from the local cache directly, fetching from memory is much more time-consuming. As can be observed that non-structural sparse models cannot make good use of memory locality, which severely lengthens its execution time.

On the contrary, a structural sparse model locates all the required weights in the same cache line. As shown in Fig. 1(b), the weights in the same group will be fetched into the cache together. The weight group without any needed weights is omitted. In this way, much less data exchange occurs between the CPU and memory. And a large weight matrix with structural sparsity takes less computation time to calculate all the weight groups. This approach improves the cache locality and accelerates the computation.



(a) Non-structural sparsity (b) Structural sparsity

Fig. 1: An illustration of the data locality induced by non-structural and structural sparsity models and the corresponding impact on hardware execution.

3.2. An overview of ESS

Our proposed ESS learning method is composed of three steps. As shown in Fig. 2, the process starts with the structural sparsity learning from scratch. At the end of this step, a sparse LSTM model at the desired sparsity level will be generated. The accuracy of the model is relatively low and will be recovered in the following two steps. In the second step, we fix the zero parameters learned in the previous step to prevent the structural sparsity from updating in the next step. At the end, the sparse model will be used as an initial input model and be trained for additional epochs. Here, the extra group Lasso regularization term in the first step is disabled. The model is retrained to recover the accuracy. Because all the zero parameters are fixed in the second step, only nonzero elements can be updated. In this way, the learned sparsity is maintained while the accuracy gradually increases. The pseudocode of ESS is shown in Algorithm 1.

3.3. Learning structural sparsity

State-of-the-art LSTM-based speech recognition models use a deep LSTMP architecture to model temporal sequences [4]. The weight matrix w in a LSTMP unit is usually a very large matrix consisting of several weight group matrices w_k. Due to its large size, calculating w is expensive and time-consuming. The optimization goal is to remove as many weight groups w_k as possible and form a desired sparse structure. Learning sparse structure in LSTM was proven to be effective in [9], and we extend it here to acoustic models.

In detail, a group Lasso penalty is added into the loss function to encourage sparse structure. The penalty term is:

$$R(w) = \sum_{k=1}^{K} ||w_k||_2, \tag{1}$$

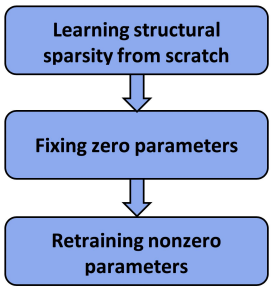


Fig. 2: The three-step training pipeline of ESS.

Algorithm 1: *ESS*: efficient sparse structure algorithm

Input : weight matrix $w^{(0)}$, sparsity regularization coefficient λ, threshold τ, learning rate η
Output: updated weight matrix w

add group Lasso regularization on the loss function
for *iteration* $i = 0, ..., M-1$ **do**
  estimate gradient $g^{(i)}$ of $w^{(i)}$ on loss function
  update: $w^{(i+1)} \leftarrow w^{(i)} - \eta \cdot \left(g^{(i)}\right)$
  zero out each element in $w^{(i+1)}$ whose absolute value is smaller than τ
end
remove group Lasso regularization on the loss function
for *iteration* $j = M, ..., M+N-1$ **do**
  estimate gradient over $w^{(j)}$: $g^{(j)}$
  zero out gradients in $g^{(j)}$ w.r.t. zero parameters in $w^{(j)}$
  update $w^{(j+1)} \leftarrow w^{(j)} - \eta \cdot g^{(j)}$
end
return $(w^{(M+N)})$

where $||\cdot||_2$ is the l_2 norm. The step to update each weight group w_k in Stochastic Gradient Descent (SGD) training becomes:

$$w_k \leftarrow w_k - \eta \cdot \left(\frac{\partial E(w)}{\partial w_k} + \lambda \cdot \frac{w_k}{||w_k||_2} \right), \tag{2}$$

where η is the learning rate, $E(w)$ is the loss calculated by the original loss function and λ is a positive coefficient which controls the degree of structural sparsity. In Equation (2), vector w_k represents a weight group matrix. $-\frac{w_k}{||w_k||_2}$ denotes the group Lasso penalty applied on w_k. Moreover, $-\frac{w_k}{||w_k||_2}$ is a unit vector along a direction opposite to w_k, which makes w_k squeezed towards zero. Although a large portion of weight groups can be forced to zero, not all of them can. When there are more zero weight groups in w, the accuracy of the model could decrease so that $\frac{\partial E(w)}{\partial w_k}$ in Equation (2) becomes larger, preventing the loss function from continuing to decrease. The training eventually converges when it reaches a balance between the accuracy term $\frac{\partial E(w)}{\partial w_k}$ and the sparsity term $-\frac{w_k}{||w_k||_2}$. The trade-off between the accuracy and sparsity is controlled by the parameter λ.

To avoid the denominator becoming zero in the group Lasso penalty term $-\frac{w_k}{||w_k||_2}$, we add a tiny constant ε into the computation of l_2 norm, such as:

$$||w_k||_2 \triangleq \sqrt{\varepsilon + \sum_j (w_{kj})^2}, \tag{3}$$

where w_{kj} is the j-th element in w_k. The constant ε prevents the results of l_2 norm from being zero. In our experiment, ε is set to be 10^{-8}.

Although w_k is squeezed gradually in each iteration, it is hard for w_k to reach exactly zero because of the continuous fluctuation in updating. Fortunately, w_k will be dynamically stabilized and fluctuate around zero. In order to zero out these small weight groups completely, we set a threshold τ and the parameters smaller than τ are removed during each iteration.

3.4. Fixing zero and retraining nonzero parameters

Learning sparse structure potentially degrades the model accuracy. Intuitively, a sparse model containing more zero elements could have a lower accuracy. To maintain the accuracy, we include an accuracy restoring step to the training process. Once the model reaches the desired sparsity rate, we fix the obtained sparse structure and switch to the basic training method to improve its accuracy. In SGD training, for instance, the step to update the weight group w_k can be expressed as

$$w_k^{(n)} \leftarrow w_k^{(n)} - \eta \cdot \left(\frac{\partial E(w)}{\partial w_k^{(n)}} \cdot \theta(w_k) \right), \tag{4}$$

where

$$\theta(\xi) = \begin{cases} 0, \xi = 0 \\ 1, \xi \neq 0 \end{cases}. \tag{5}$$

By adding the θ function, the zero elements will remain unchanged and only nonzero elements will be updated to minimize the loss function. In this way, the sparse structure learned in the previous step is kept meanwhile the model accuracy is improved.

4. EXPERIMENTS

The open-source Kaldi toolkit is used to conduct experiments in this work. Specifically, we use the model generated by nnet3-based recipe provided in Kaldi *tedlium/s5_r2* as baseline. The only difference is that we did not use i-vectors. The training corpus for the experiments is TED-LIUM (the second release) [11]. It contains 207 hours, 1475 TED talks from 1242 different speakers. The corpus is split into three sets: training, development and testing.

The model topology is described in Kaldi *tedlium/s5_r2*. It contains three LSTMP layers, indicating that the proposed structural sparsity can be applied to three large matrices. We adjusted our sparsity method based on the target hardware architecture in order to get the best speedup performance.

4.1. Hardware implementation

The target hardware platform is inspired by the Intel® Gaussian Neural Accelerator (GNA) [12]. GNA consists of parallel integer compute cores, a memory mapping unit, DMA units, layer sequencing controller, and layer processing controller [13]. It is capable of inference processing for large

Table 1: The model sparsity and WER under different sparsity configurations

Method	λ	Sparsity in LSTM			Sparsity	WER	
group-8		1	2	3	mean	develop	test
baseline	0	0	0	0	0	11.5%	11.4%
ESS	0.15	39.1%	59.6%	36.0%	45.8%	12.0%	11.8%
ESS	0.35	62.2%	82.5%	68.6%	72.5%	12.6%	12.6%
ESS	0.65	74.9%	88.5%	71.7%	78.9%	13.3%	13.3%
Method	λ	Sparsity in LSTM			Sparsity	WER	
group-16		1	2	3	mean	develop	test
baseline	0	0	0	0	0	11.5%	11.4%
ESS	0.15	36.6%	56.7%	37.2%	44.6%	11.8%	11.7%
ESS	0.35	55.8%	77.0%	63.8%	67.0%	12.5%	12.4%
ESS	0.65	67.9%	84.9%	70.4%	75.4%	13.1%	13.1%

neural networks used in continuous speech recognition with high performance and low power consumption. For this hardware, weight matrices are represented as `int8` or `int16` and the hardware fetches data in chunks. So weight groups such as 8 `int16` elements or 16 `int8` elements are amenable to the accelerator memory architecture. If the matrix cannot be divided by 8 or 16, the programmer could apply zero padding to the matrix.

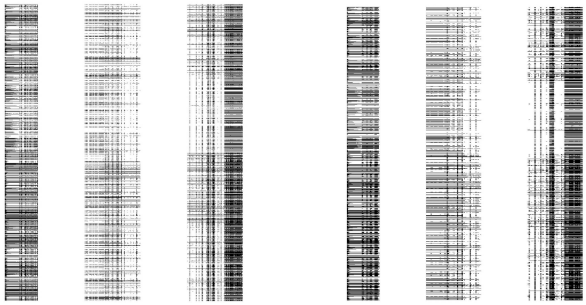
4.2. Experiment results

In our experiment, sparsity is defined as the ratio of the number of zero groups to the total number of weight groups. We applied ESS to all three LSTMP layers in the model. Grouping by both 8 `int16` elements (*group-8*) and 16 `int8` elements (*group-16*) were explored for generalization.

Table 1 summarizes our experimental results, which details the sparsity of each LSTMP layer as well as the entire model.

When applying *group-8* and setting the sparsity parameter λ to 0.35, the overall sparsity of the LSTM model is 72.5%. In theory, the model can achieve up to 3.64× speedup as 72.5% of the calculation can be omitted. Compared to the baseline, the WER of the ESS model increases 1.1% and 1.2% on the development and test sets, respectively. It is also worth pointing out that when a model is trained with $\lambda = 0.15$, the WER only increases 0.4%. Our results show that the ESS method can keep the WER at a similar level and dramatically decrease the number of nonzero weight groups.

Table 1 also shows that as λ increases, the sparsity level grows while speech recognition accuracy drops. It reflects the trade-off between sparsity and WER. However, further increasing λ doesn't help to improve the sparsity level at a corresponding rate. For instance, when $\lambda = 0.65$, the model sparsity reaches to 78.9%, which is high but not significantly larger than 72.5% obtained at $\lambda = 0.35$. Compared to the sparsity gain, WER increases more. In brief, when λ is small, sparsity increases rapidly with minor accuracy loss. As λ gets larger, sparsity grows slower and WER increases faster. It can be intuitively explained that with fewer parameters, there is a limit on the WER that a model can reach. In practice, we tend to keep λ below a threshold to achieve the best effectiveness.



(a) *group-8* (b) *group-16*

Fig. 3: The visualization of weight matrices in LSTM units learned by ESS with $\lambda = 0.35$.

Fig. 3 visualizes the weight matrices in these LSTM units, where white color represents zero weight groups and black denotes nonzero weight groups. Note that white color only appears when all the elements in a weight group are 0. Fig. 3(a) shows the matrices trained with $\lambda = 0.35$ and *group-8*. Although the learned structures are not exactly the same among three weight matrices, they are all very sparse. In fact, the difference among the matrices indicates that no single sparse structure is chosen. Instead, the learning process makes it converge to an optimal configuration across all the matrices.

Both Table 1 and Fig. 3(b) show similar results for ESS with weight groups of 16 elements. The sparsity obtained by *group-16*, however, is slightly lower than that by *group-8*, under the same constraint λ. This is simply because it is harder to force all the elements in a larger group to be zero.

Finally, we'd like to point out that we did not observe any increase in decoding time due to acoustic modeling error.

5. CONCLUSION

In this work, we proposed an efficient structural sparsity (ESS) learning method for acoustic modeling in speech recognition. ESS, based on group Lasso regularization, forces the networks to learn sparse structures while maintaining accuracy. We showed that our method can generate structural sparse models that speedup computation better than non-structural sparse models. We applied our method to an HMM+LSTM architecture model and obtained a model with high sparsity and little WER increase.

Acknowledgements. This work is supported by the National Science Foundation CCF-1744082.

6. REFERENCES

[1] H. Sak, A. Senior, and F. Beaufays, "Long short-term memorry based on recurrent neural network arrchitectures for large vocabulary speech recognition," *arXiv:1402.1128*, 2014.

[2] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of ICASSP*, 2013.

[3] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *ASRU*, 2013, pp. 273–278.

[4] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Interspeech*, 2014.

[5] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Advances in Neural Information Processing Systems*, 2015.

[6] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[7] W. Wen, C. Wu, Y. W, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems*, 2016.

[8] S. Narang, G. Diamos, S. Sengupta, and E. Elsen, "Exploring sparsity in recurrent neural networks," *arXiv:1704.05119*, 2017.

[9] W. Wen, Y. He, S. Rajbhandari, W. Wang, F. Liu, B. Hu, Y. Chen, and H. Li, "Learning intrinsic sparse structures within long short-term memory," *arXiv:1709.05027*, 2017.

[10] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, M. Petr, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proceedings ASRU*, 2011, pp. 1–4.

[11] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM corpus with selected data for language modeling and more ted talks," in *LREC*, 2014, pp. 3935–3939.

[12] M. Deisher and A. Polonski, "Implementation of efficient, low power deep neural networks on next-generation intel client platforms," *http://sigport.org/1777*, 2017.

[13] G. Stemmer, M. Georges, J. Hofer, P. Rozen, J. Bauer, J. Nowicki, T. Bocklet, H. R. Colett, O. Falik, M. Deisher, and S. J. Downing, "Speech recognition and understanding on hardware-accelerated DSP," in *Proc. Interspeech*, 2017, pp. 2036–2037.