Learnable Sparsity Structured Pruning for Acoustic Pre-trained Models

Siyuan Wang
Department of Electronic Engineering, Tsinghua University, China
wsy21@mails.tsinghua.edu.cn

Haoyu Wang
Department of Electronic Engineering, Tsinghua University, China
w-hy21@mails.tsinghua.edu.cn

Jian Li
Sinovoice, China
lijian@sinovoice.com.cn

Wei-Qiang Zhang*
Department of Electronic Engineering, Tsinghua University, China
wqzhang@tsinghua.edu.cn

ABSTRACT

Large-scale pre-trained models bring significant gains to many speech-related tasks. However, it is still challenging to use these large models when computing power of terminal equipment is limited. Pruning is an effective method to reduce memory footprint and cost calculation. The imperfect evaluation criteria of existing pruning methods and the complex fine tuning process result in a relatively high loss of accuracy. To solve these problems, we propose a structured pruning method, which introduced the upper confidence bound of importance scores to assess the potential of each component of the model more accurately. In addition, we also introduce a set of learnable pruning threshold parameters that can be learned via stochastic gradient descent, thereby reducing the hyper-parameter tuning. We apply our method to HuBERT models on automatic speech recognition (ASR) task. Our result shows that for all pruning granularity and pruning ratios, our method yields higher accuracy and speedup ratios in the inference process.When sparsity was 60%, our method performed only 0.63% down.

CCS CONCEPTS

• **Computing methodologies**; • **Speech recognition; Unsupervised learning; Transfer learning.**;

KEYWORDS

Neural network pruning, knowledge distillation, model compression, representation learning

ACM Reference Format:
Siyuan Wang, Haoyu Wang, Jian Li, and Wei-Qiang Zhang*. 2023. Learnable Sparsity Structured Pruning for Acoustic Pre-trained Models. In *2023 6th International Conference on Signal Processing and Machine Learning (SPML) (SPML 2023), July 14–16, 2023, Tianjin, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3614008.3614020

SPML 2023, July 14–16, 2023, Tianjin, China

ACM ISBN 979-8-4007-0757-5/23/07.
https://doi.org/10.1145/3614008.3614020

1 INTRODUCTION

Self-supervised learning (SSL) methods for learning representations of speech have been successful in many tasks in the past few years. They can obtain the learning of meaningful and distinctive features from unlabeled speech data. Pre-trained models based on Transfomer structures such as Wav2Vec2 [21], HuBERT [9], and WavLM [2] have made this representation an important component and have achieved significant results on tasks related to the speech domain in this way. These models, however, typically have a large number of parameters and take long inference time, which requires more storage space and more computational power. The limitations make these models unsuitable for low-resource devices and difficult for researchers in academia. Therefore, model compression has become a research hot-spot and focus in the field of deep learning.

In order to reduce the model size and improve accuracy, a variety of model compression techniques are proposed [7]. Most commonly used techniques include low-rank approximation [13, 16], weight sharing [4], knowledge distillation [10, 11, 23], quantization [1, 6, 22, 28], and pruning [3, 17, 18, 25]. In this paper, we focus on pruning. Pruning methods have been shown to be extremely effective at reducing the storage size of models fine-tuned for a specific task. It aims to search for an accurate sub-network in a larger pre-trained model. It can be broadly classified into two categories based on the granularity of removed components: (i) Unstructured pruning, that is, zeroing out insignificant parameters based on heuristic methods such as magnitude pruning [8], regularization of L0 [15]. (ii) Structured pruning refers to the structural pruning of Transformer networks [24], from pruning heads [18], to removing entire layers [5]. Although various pruning techniques have been proposed, no matter which kind of pruning methods, there are some defects. Unstructured pruning can lead to competitive performance, but it is difficult to accelerate due to irregular sparsity. As for structured pruning, in order to obtain a reasonable accuracy, complex hyper-parameter tuning operations are required. Under the conditions of high sparsity, the accuracy loss is still relatively high, although they produce hardware-friendly sub-networks.

In this work, we propose a structured pruning approach method named Learnable Sparsity Structured Pruning(LSSP). In our method, we introduced the upper confidence bound of importance scores, to evaluate which components should be pruned out in the model. In addition, in order to simplify the hyperparameter tuning process, we introduce a set of learnable pruning ratio parameters that can

be learned by stochastic gradient descent, and a new regularization term. We apply our method for HuBERT models on ASR tasks. The results show for all pruning granularity, and pruning ratios, our methods all yield higher accuracy and speedup ratios. Especially, when the sparsity is 60%, the accuracy only decreases by 0.63% and the speedup ratio reaches 2.4.Next,we will demonstrate from a variety of novel methods, experiments, and ablation experiments.

2 METHODS

The proposed method can be applied to any transformer-based acoustic model. For this work, we selected HuBERT, which performed well on SUPERB [27].

2.1 HuBERT

The HuBERT model can be divided into two parts as follows: CNN Encoder and Transformer Encoder. The input is down sampled and raise the dimension by CNN Encoder, and then randomly masked and fed into Transformer Encoder. The labels are obtained by clustering MFCC features or another model's hidden units.

2.2 Upper Confidence Bound of Importance Scores

The Background Of Movement Pruning Regardless of whether the pruning is structured or unstructured, the training strategies of existing pruning methods can be grouped into the following two categories: (i) one-time pruning [14] and (ii) iterative pruning [8]. While one-time pruning removes all redundant parameters at once and fine-tunes them after ranking their importance, iterative pruning removes only a portion of the redundant parameters at a time and continues with the next round of pruning after a short fine-tuning. However, one-time pruning often requires pre-training for downstream tasks, which results in poor performance. In this work, we focus on the latter scenario. Movement pruning [20] performed better in iterative pruning. It combines the weight of the model with gradient information to determine the importance. Instinctively, Movement pruning selects weights that tend to move from 0 in training, meaning it focuses more on first-order information about weights. But because the score is estimated based on mini-batch data and receives complex optimization strategies during training, it has a fatal error: It does not accurately reflect its contribution to model performance. Specifically, some weights often alternate between being clipped and activated, making training unstable.

Optimization Methods To address this problem, we were inspired by PLATON [29] to apply upper confidence bound (UCB) to capture the uncertainty of significance scores in our proposed structured pruning approach.

Let $W \in R^{n\times n}$ be a general representation of the weight matrix in the model (only the square matrix is discussed). To decide which weights in W should be removed, we introduce an importance score matrix $S \in R^{n\times n}$ and a masking matrix $M \in R^{n\times n}$. The pruning process is the optimization of S to select the most important weights, and different pruning methods optimize S in different ways. The actual computation involved is the Hadamard product of the masking matrix and the weights, i.e., for input x , the corresponding output is $a = (W \odot M)x$. A common strategy for generating the masking matrix M is to retain the top v % of the weights. This function is defined as:

$$Top_{v\%}(S)_{i,j} = \begin{cases} 1, & if\ S_{i,j}\ is\ in\ top\ v\% \\ 0, & otherwise \end{cases}$$

For the movement pruning, its score matrix S is defined as follow:

$$S = |W \odot \nabla L\,(W)|$$

This formula is derived from the first-order Taylor expansion of L with respect to W, approximates the change in loss when a weight or component is subtracted. We have improved on this in two ways: (i) exponential moving average, which reduces the non-negligible variability due to mini-batch data and complex optimization strategies.(ii) uncertainty qualification by local time variations, and trend algorithm discover the potential of weights. Such quantification can be considered a UCB of estimated importance.

Exponential Moving Average This method forces the model to retain weights that suddenly drop in importance scores due to instability in training.Specifically, in the t-th itration of the model, we define the smoothing value as:

$$S_m^{(t)} = \beta_1 S_m^{(t-1)} + (1-\beta_1)\ S^{(t)}$$

where $\beta_1 \in (0,1)$ is a hyperparameter. The moving average can effectively alleviate the sudden changes in the scores during the training process, which makes the training more stable, and the exponential level can emphasize more recent information, which is more conducive to the convergence of the model.

Uncertainty Qualification In addition to moving average, we directly consider uncertainties in the estimation of important scores to reduce abrupt changes. Specifically, we quantify the uncertainty of an estimation by local time variation in sensitivity, defined as:

$$U^t = \left|S^{(t)} - S_m^{(t)}\right|$$

$$U_m^{(t)} = \beta_2 U_m^{(t-1)} + (1-\beta_2)\,U^{(t)}$$

Uncertainty quantifies variability by taking into account the difference between the current significance score and its historical average. The larger the value, the greater the uncertainty of the weight, which means its S_m is less trustworthy, in the sense it can be considered as the upper confidence bound of S_m.

Ideally, the criteria would be a combination of the two, followed by screening, so we define the ultimate significance score as:

$$S_{UCB}^{(t)} = S_m^{(t)} \odot U_m^{(t)}$$

When some weight or component has a low $S_m$but a high U_m, the model tends to retain and develop its potential.

Table 1: Summary of pruning mode

Modes	MHA	FFN
Hybrid	Block	Dim
Structure	Heads	Dim
Unstructure	1*1	1*1

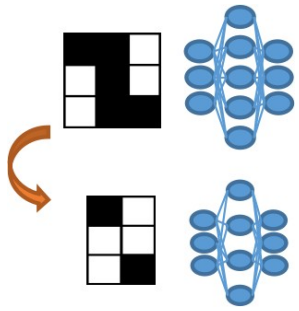

Figure 1: Block pruning diagram

2.3 Structured Pruning Methods

In this work, we apply the block pruning method [12] as the basic framework. The core idea of this method is the same as the Movement pruning method, except that the pruning object is expanded to the parameter block.Specifically, for each weight matrix $W \in R^{n\times n}$, we set a fixed block structure (a,b). All weights within each block are delimited into a group and share a common importance score (using S_{UCB} as designed in the previous section), which is derived from the corresponding score matrix $S_{UCB} \in R^{n/a\times n/b}$. Finally, the mask weight is calculated by extending the masking matrix:

$$W_{new} = W \odot M(S_{UCB})_{[n/a\times n/b]}$$

The number of parameters in the transform-based models is mainly concentrated in the feed-forward network layer (FFN) and the multi-head attention layer (MHA). Where FFN consists of two matrices W_1 and W_2 and its shape is $R^{d_{model}\times d_{inner}}$ or $R^{d_{inner}\times d_{model}}$, where d_{inner} is the hidden size. The MHA parameters composed of four matrices, W_K,W_Q,W_V and W_O , and are shaped like $R^{d_{model}\times d_{model}}$. In our experiment, we used three modes to prune the model, Hybrid, Structure and Unstructure, as shown in Table 1. The specific shape is as follows:

- Block :(32,32),square blocks.
- Dim :(1,d_{model}) and (d_{model}, 1),FFN hidden size.
- Heads :(d_{heads},d_{model}),attention heads.

Additionally, like Figure 1. If the model masks some complete heads of the attention or masks some inner dimensions of the FFN, we will further compact the model and perform a second fine-tuning. In generating the mask matrix, we use two schemes, one of is the Top_v method mentioned above, called hard-pruning [20], and the other soft-pruning [20]. For soft-pruning, M is not controlled by a fixed percentage v, but by a fixed threshold t, such as M = (S > t). In order to control sparsity, additional regularization terms are added to control the gradual reduction of importance scores. The coefficient control the penalty intensity and thus the sparsity level. The formula is as follows:

$$\lambda_{total} * (\lambda_{FFN}||\sigma(S_{FFN})\,||+\lambda_{MHA}||\,\sigma\,(S_{MHA})\,||$$

where λ_{total}, λ_{FFN} and λ_{MHA} are hyper-parameter,$||A|| = \sum_{i,j} A_{i,j}$ and σ is the sigmoid function.

Because of the different sensitivity of different matrices, soft pruning can set the matrices sparsity flexibly, which makes it better than hard pruning. But its sparsity is controlled by the regularization term coefficient. It needs complex experiments to achieve the target sparsity.

2.4 Learnable Pruning Threshold Parameters

Hard pruning is simple to operate, but not flexible. Soft pruning is the opposite. Therefore, we introduced a set of learning thresholds $[t_1, ..., t_n]$ and applied them to hard pruning to make up for its shortcomings. The formula is as follows:

$$Top_{v\%}(S)_{i,j} = \begin{cases} 1, & if\ S_{i,j}\ is\ in\ top\ \sigma\,(t_i)\,\% \\ 0, & otherwise \end{cases}$$

We introduce a new regularization term to drive the model to achieve the target sparsity, specifically as follows:

$$L_{regu}\,(t) = \left(\left(N_{remain} - N_{target}\right)/N_{total}\right)^2$$

Where N represents the number of model parameters. Therefore, the target loss is:

$$\text{Loss} = \text{Loss}_{ctc} + \lambda_{regu} * L_{regu}$$

Because of competition between ctc loss and regularization terms, models are often not pruned in late training to maintain performance. Therefore, we redesigned a normalized term coefficient to make the approximation of target sparsity more accurate. The form is as follows:

$$\lambda_{regu} = max\left(\lambda_{max}\frac{L_{regu}}{1-\left(N_{target}/N_{total}\right)^2}, 70\right) * a$$

$$a = \begin{cases} min\left(\left(\frac{T}{T_{max}-T}\right) * 5, 10\right), & \frac{T}{T_{max}-T} \geq 0.25 \\ \left(\frac{T}{T_{max}-T}\right) * 3, & otherwise \end{cases}$$

where λ_{max} is hyperparameter. T is the number of training steps. The main idea is to increase the penalty when there is a large gap between the current sparsity and the target sparsity, and vice versa. Secondly, a warm-up step is set to limit the penalty to a smaller range when the sparsity is far away from the target at the beginning of the training, in order to let the model stabilize first, which is more conducive to the later training.

3 EXPERIMENTS

3.1 Datasets

For the dataset, we selected audio of less than 14 seconds in the Librispeech [19] train-clean-100 as our training set and tested it on the Librispeech test-clean. Librispeech consists of about 1000 hours of English reading speech corpus and corresponding transcribed text files, sampled at 16 kHz.

3.2 Baseline

Our baseline model utilizes the framework and pre-trained models provided by Hugging Face [26], and was fine-tuned using ctc loss in the same training set. To be fair, the rate of learning and number of training steps were consistent with subsequent experiments.

Table 2: Summary of results. In this case, LSSP is short for our proposed method. MvP is short for Movement pruning. S1 represents the first pruning process and s2 represents the further fine-tuning process after the model is compacted.WER means word error rate.

Methods	WER s1/s2	Speed-UP	Pruning mode	Sparsity
HuBERT-Base	7.7./N/A	1.0	N/A	
MVP-Hard	14.92/12.77	2.5	Structure	60
MVP-Soft	12.64/11.22	2.4		
LSSP	12/10.85	2.4		
MVP-Hard	13.4/11.3	2.2	Hybrid	
MVP-Soft	10/8.84	2		
LSSP	9.4/8.3	2		
MVP-Hard	16.21/13.78	3.3	Structure	80
MVP-Soft	14.44/12.07	2.7		
LSSP	12.89/11.8	2.6		
MVP-Hard	14.7/13.2	2.9	Hybrid	
MVP-Soft	12.69/11.77	2.6		
LSSP	11.84/11.11	2.6		
MVP-Hard	10.93/N/A	1.0		85
MVP-Soft	9.47/N/A	1.0	Unstructure	
LSSP	8.88/N/A	1.0		

3.3 Implementation details

We trained the model on three GeForce RTX 3090 for 100,000 steps in a batch size of 24 hours and less than 21 hours. In terms of learning rate, we used a warm-up optimization strategy of learning rate, which linearly increased to 3e-5 in the first 10000 training steps and decreased to 0 in subsequent training steps. For sparsity control, our strategy is increase the sparsity rates gradually increase from 0 to 60%. And let it remained constant for the first 10% percent and the last 20% of step ensure the stability of the model. We select the exponential moving average parameters β_1 from {0.80, 0.90} and β_2 from ={0.850, 0.950}, and select the regularization term coefficient $\lambda_{\max}$ from {20000, 40000, 80000}.

4 RESULTS

In this section, we compare our approach to the baseline model, as well as the hard and soft pruning methods described above, under the same conditions as the general hyper-parameters, at sparse levels 60%, 80% and 85%. The results of HuBERT-base model on Librispeech test-clean are shown in the following Table 2. To start with the structured pruning method, the Hybrid mode WER is significantly higher than Structure in pruning mode, but the acceleration ratio is generally smaller. This phenomenon shows that different block shapes have obvious different influence on pruning effect and inference speed. Small chunks mean higher accuracy and smaller acceleration ratio. Secondly, because the model acceleration ratio produced by the Hybird and Structured methods was not significantly different, we further observed that the model learned to remove the entire head in MHA even though we did not set the pruning scale to a full head size in the Hybrid method. The pruning effect is shown in the Figure 2. Based on this, we used a Hybrid mode as the primary method. As Table 2 shows, no matter the sparsity is 60%, 80% or 85%, our method can achieve the best results. In particular, when sparsity was 60%, our method performed only 0.6% worse compared to baseline after step2 fine-tuning.

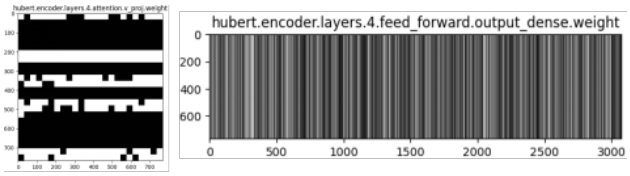

Figure 2: The matrix is pruned under Hybrid method. White is preserved,black is pruned

4.1 Ablation Experiment

In this section, we will analyze why our method can bring to better results in the form of ablation experiment.

The Influence Of UCB We first analyzed UCB, the results are shown in the Table 3. Because of the soft pruning method drives a diminishing importance score, which conflicts with our UCB, so we apply UCB only to hard pruning (even if hard pruning is less effective than soft pruning, the learnable threshold makes up for it). As we can see, at 60% sparsity.UCB improved the performance of hard pruning by 2%, but due to the inherent inflexibility of hard pruning, MVP-UCB was not as effective as soft pruning. Further, we can see in Figure 3, UCB changes significantly less than the importance score used in traditional movement pruning, resulting in a more stable training process and better model performance.

Finally, we tested the sensitivity of the MVP-UCB method to β_1 and β_2, and our results showed that our method was robust for these two hyper-parameters.

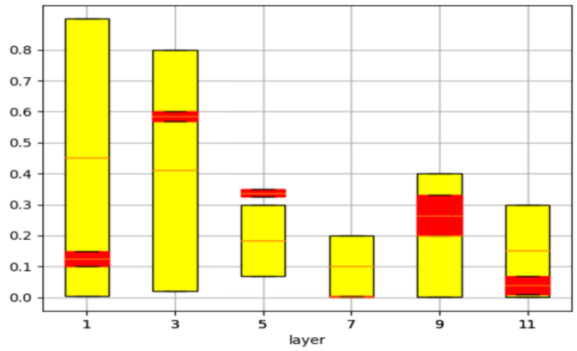

Figure 3: Score change range graph. Red is UCB. We take the mean value of the Score matrix corresponding to the matrix of each layer and calculate the score every 10 steps.

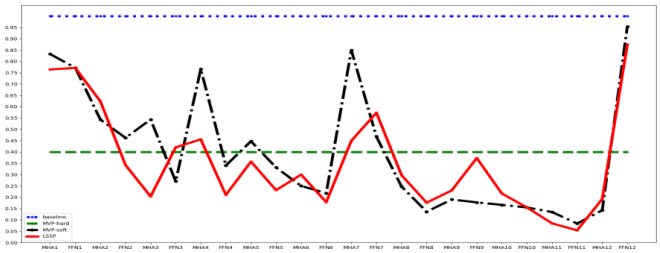

Figure 4: Compare the sparsity of the MHA matrix and FFN matrix

Table 3: Performance comparison of different methods. Only compare the results of step1. Among them, MVP-UCB means only the movement pruning that only adds UCB.

Methods	WER
MVP-Hard	13.4
MVP-Soft	10
MVP-UCB	11.3

The Influence Of Learnable Threshold To prove that our approach can also be as flexible as soft pruning, we compare the sparsity of MHA matrix and FFN matrix corresponding to each layer of the pruning model obtained by soft pruning and our method. The results are shown in Figure 4. Roughly the following pattern can be obtained: (i) FFN is easier to prune than MHA. (ii) The matrix near the input is less sparse than that near the output. Our method learned similar rules to soft pruning. In addition, our method requires only one hyperparameter λ_{max}, which greatly reduces complex operations.

5 CONCLUSIONS

In this paper, we propose a structured pruning method, which introduced upper confidence bound of importance scores, a set of learnable pruning threshold parameters, and corresponding regularization terms.Experiments show that our method can more accurately assess the importance of weights and learn the sensitive of different matrices, and the operation is relatively simple. For all pruning granularity, and pruning ratios, our method all yield higher accuracy and speedup ratios. In particular, when sparsity was 60%, our method performed only 0.63% down. Note that this is the result without distillation loss.

There are some meaningful works left to do, such as add teacher models and distillation loss, and using it to guide base model pruning, which we believe will yield some gains. Next, try pruning large model and see how well it works when both large and base model prune to the same absolute number of parameters. We leave these questions for future work.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant No. 62276153.

REFERENCES

[1] Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore. 2019. Efficient 8-bit quantization of transformer neural machine language translation model. arXiv preprint arXiv:1906.00532 (2019).
[2] Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, *et al.* 2022. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. IEEE Journal of Selected Topics in Signal Processing 16, 6 (2022), 1505–1518.
[3] Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2019. Fine-tune BERT with sparse self-attention mechanism. In Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP). 3548–3553.
[4] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. arXiv preprint arXiv:1807.03819 (2018).
[5] Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. arXiv preprint arXiv:1909.11556 (2019).
[6] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. 2020. Training with quantization noise for extreme model compression. arXiv preprint arXiv:2004.07320 (2020).
[7] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. 2016. EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News 44, 3 (2016), 243–254.
[8] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. Advances in neural information processing systems 28 (2015).
[9] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. IEEE/ACM Transactions on Audio, Speech, and Language Processing 29 (2021), 3451–3460.
[10] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. arXiv preprint arXiv:1909.10351 (2019).
[11] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. arXiv preprint arXiv:1908.08593 (2019).
[12] François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. 2021. Block pruning for faster transformers. arXiv preprint arXiv:2109.04838 (2021).
[13] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019).
[14] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Snip: Single-shot network pruning based on connection sensitivity.
[15] arXiv preprint arXiv:1810.02340 (2018).
[16] Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through L_0 regularization. arXiv preprint arXiv:1712.01312 (2017).
[17] Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. 2019. A tensorized transformer for language modeling. Advances in neural information processing systems 32 (2019).
[18] J Scott McCarley. 2019. Pruning a bert-based question answering model. arXiv preprint arXiv:1910.06360 142 (2019).
[19] Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? Advances in neural information processing systems 32 (2019).

[20] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 5206–5210.
[21] Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. Advances in Neural Information Processing Systems 33 (2020), 20378–20389.
[22] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. 2019. wav2vec: Unsupervised pre-training for speech recognition.
[23] arXiv preprint arXiv:1904.05862 (2019).
[24] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 8815–8821.
[25] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. arXiv preprint arXiv:1908.09355 (2019).
[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems 30 (2017).
[27] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. arXiv preprint arXiv:1905.09418 (2019).
[28] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, *et al.* 2019. Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019).
[29] Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y Lin, Andy T Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, *et al.* 2021. Superb: Speech processing universal performance benchmark. arXiv preprint arXiv:2105.01051 (2021).
[30] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. In 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS). IEEE, 36–39.
[31] Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022. Platon: Pruning large transformer models with upper confidence bound of weight importance. In International Conference on Machine Learning. PMLR, 26809–26823.