ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095006

SHARING LOW RANK CONFORMER WEIGHTS FOR TINY ALWAYS-ON AMBIENT SPEECH RECOGNITION MODELS

Steven M. Hernandez[1], *Ding Zhao*[2], *Shaojin Ding*[2], *Antoine Bruguier*[2],
Rohit Prabhavalkar[2], *Tara N. Sainath*[2], *Yanzhang He*[2], *Ian McGraw*[2]

[1]Virginia Commonwealth University, [2]Google

ABSTRACT

Continued improvements in machine learning techniques offer exciting new opportunities through the use of larger models and larger training datasets. However, there is a growing need to offer these new capabilities on-board low-powered devices such as smartphones, wearables and other embedded environments where only low memory is available. Towards this, we consider methods to reduce the model size of Conformer-based speech recognition models which typically require models with greater than 100M parameters down to just 5M parameters while minimizing impact on model quality. Such a model allows us to achieve always-on ambient speech recognition on edge devices with low-memory neural processors. We propose model weight reuse at different levels within our model architecture: (i) repeating full conformer block layers, (ii) sharing specific conformer modules across layers, (iii) sharing sub-components per conformer module, and (iv) sharing decomposed sub-component weights after low-rank decomposition. By sharing weights at different levels of our model, we can retain the full model in-memory while increasing the number of virtual transformations applied to the input. Through a series of ablation studies and evaluations, we find that with weight sharing and a low-rank architecture, we can achieve a WER of 2.84 and 2.94 for Librispeech dev-clean and test-clean respectively with a 5M parameter model.

Index Terms— Model compression, conformer, weight sharing, low rank decomposition, embedded speech recognition

1. INTRODUCTION

Automatic speech recognition (ASR) is an essential component in a growing number of spoken language interfaces on mobile devices. Recently, long running applications such as transcribed recording, live captioning, and generalized keyword spotting are emerging, and are even more challenging on edge devices due to the limited resources. Always-on ambient speech recognition, the most ambitious use scenario, leverages advances in both deep learning and embedded neural processing hardware to enable always-running ASR on low power edge devices.

To achieve efficient always-on recognition with edge devices, instead of running a standard ASR model, alternative approaches are usually considered, such as recognizing a single keyword [1] or just a small set of intents [2]. However, this is not possible for keywordless interactions and also requires newly trained models each time new intents are made available. As such, in this work, we aim to focus on the extendability achieved by a generalized speech recognition model. However, recent advances in machine learning come at the expense of ever increasing model sizes (i.e., 100M parameters and higher). These models are not tractable for always-running on neural accelerators such as edge TPUs [3], which are limited to fewer than 6M parameters due to hardware memory constraints. In fact, to achieve inference using such large model sizes, the model must be split into smaller chunks which are then continuously transferred from memory to TPU, leading to poor energy usage and poor latency for ambient speech recognition tasks.

In this paper, we look for methods to reduce the size of Conformer-based [4] speech recognition models to achieve always-on ambient speech recognition, which can efficiently leverage specialized hardware such as edge TPUs. To do this, we propose model weight reuse at different levels within our Conformer architecture such as: (i) repeating full conformer layers, (ii) sharing specific modules across conformer layers, (iii) sharing specific sub-components within each conformer module, and (iv) sharing low-rank sub-weights after low-rank decomposition. Unlike other model compression techniques like low-bit quantization [5] and sparsity [6] which assume the use of specialized hardware features which we will discuss in Section 2, both sharing and low rank architectures can be achieved with existing neural accelerators. By sharing weights across layers, we can increase the number of virtual transformations applied to our input data without increasing the physical size of the model weights in memory. Increasing the number of virtual transformations in our model allows for more complex transformations on our model input which emulates the transforms typically found by increasing the number of layers in a model.

2. RELATED WORKS

Performing machine learning model inference on-board low power edge devices has recently achieved greater attention for tasks such as device-free wireless sensing [7], computer vision [8], and numerous other tasks [9]. At the core of edge model inference is achieving model compression for use on low power and low resourced devices.

Model compression has commonly been achieved through a number of methods such as sparsity pruning [6, 10, 11], low-bit quantization [12, 13, 14], knowledge distillation [15, 16], and low-rank matrix factorization [17, 18]. These techniques can typically be applied regardless of the model architecture which allows them to be generalized to different tasks. However, some methods assume access to specific hardware features that may not be available on edge devices. Model sparsity techniques offers the ability to prune weights until an exact model size is achieved. However, without structured sparsity [19], the resulting model requires irregular memory access and without hardware support, memory usage and computation become inefficient. Quantization is typically applied to reduce model weights from 32-bit floating point values down to 8-bit integer values, and is also applied to lower quantization levels (i.e., 1-bit, 2-bit, or 4-bit [5, 14]) and even mixed-precision quantization [20]. However, computations on low-bit quantization level

This work was done while Steven M. Hernandez was an intern at Google.

models are not available on typical real-world hardware. On the other hand, techniques like knowledge distillation and low-rank decomposition are computed off-device and thus performing inference on these compressed models is identical to non-compressed models.

3. METHODS

3.1. Conformer Model

For our ambient ASR task, we leverage the conformer model architecture [4], an extension to the transformer model architecture [21]. For the intents of this work, we will focus on reducing the size of the conformer encoder since we find that it takes up greater than 90% of the overall model size. The size of the encoder is primarily a result of the N conformer blocks, thus we will also focus on ways to reduce the size of the encoder by both reducing the size of the individual conformer blocks as well as reducing the need for large values of N.

We define the i-th conformer block $\mathbb{C}_{(i)}$ in our model as $\mathbb{C}_{(i)}\left(F^{(i)}_{\text{start}}, A^{(i)}, C^{(i)}, F^{(i)}_{\text{end}}\right)$ where F_{start}, A, C, F_{end} are the parameters for the feed forward start, attention, convolution and, feed forward end modules respectively within the i-th conformer block $\mathbb{C}_{(i)}$. Fig. 1 illustrates the largest size sub-components for each of the modules. It is important to recognize these largest sub-components when reducing the size of our model because these are the weight matrices which we should focus on compressing. Notice, that while the architecture of our conformer contains many unique features, the size of the conformer blocks is primarily the result of several linear layers. Thus, if we can apply a compression technique to simple linear layers, then they can be similarly applied throughout the entire conformer model.

3.2. Repeat Full Layers

Suppose we have a conformer model with N conformer blocks $\mathbb{C}_{(i)}$ where $i \in \{1, \ldots, N\}$, then we can repeat each i-th layer $R[i]$ times as suggested in [22] by sharing the i-th layer's parameters. Our conformer transformation will then be described as a series of n-fold iterative functions defined as:

$$f^{\circ n} = \underbrace{f \circ f \circ \cdots \circ f}_{n}, \tag{1}$$

where f is some transformation function, and n is the number of times that the function is composed over itself. As such, our conformer transformation could be described as:

$$\underbrace{\mathbb{C}_{(N)} \circ \cdots \circ \mathbb{C}_{(N)}}_{R[N]} \circ \cdots \circ \underbrace{\mathbb{C}_{(2)} \circ \cdots \circ \mathbb{C}_{(2)}}_{R[2]} \circ \underbrace{\mathbb{C}_{(1)} \circ \cdots \circ \mathbb{C}_{(1)}}_{R[1]}, \tag{2}$$

or

$$\mathbb{C}_{(N)}^{\circ R[N]} \circ \cdots \circ \mathbb{C}_{(2)}^{\circ R[2]} \circ \mathbb{C}_{(1)}^{\circ R[1]}. \tag{3}$$

By repeating layers, we retain a set number of physical conformer layers (N) while increasing the number of virtual conformer layers or the number of transformations ($R \times N$). By performing n-fold iterative conformer transformations, we expect that we can transform our input with more complex transformations without directly increasing our cost from a model size perspective.

3.3. Sub-component Customization

While we expect that the higher complexity transformations offered by repeating layers will be of benefit to our model architecture, there is still a clear intuition that a model with $R \times N$ virtual layers will likely not be able to perform better than a model with $R \times N$ physical layers due to the increased number of distinct parameters available in the model. Thus, we may wish to allow for layer repeating but with some slight customization per layer. By allowing customization, we can retain our ability to reduce model size through sharing, but we also allow conformer blocks to perform unique transformations instead of strictly iterative function composition.

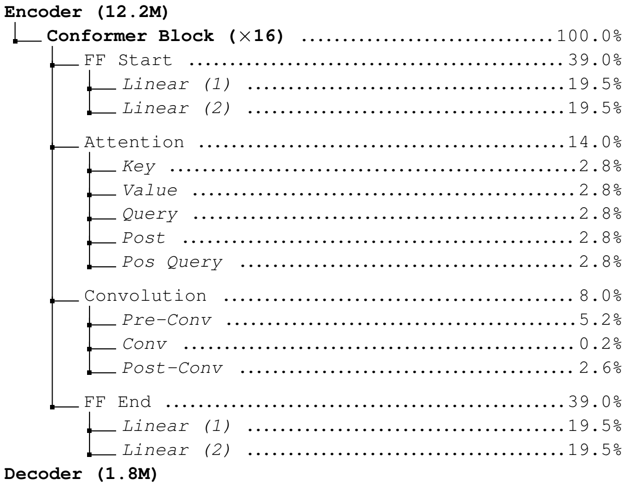

Fig. 1. High level composition of our base 14M parameter Conformer model (B0). We focus on reducing the size of the 16 conformer blocks within the encoder portion through sharing (i) full conformer blocks, (ii) modules, and (iii) sub-components.

Our first effort towards this is to only share certain modules within our conformer blocks. Sharing specific modules such as feed forward or attention modules was initially reviewed in [23] towards a reduced size BERT model. We define sharing indices $\mathcal{I}_{FS}, \mathcal{I}_A, \mathcal{I}_C$, $\mathcal{I}_{FE}$ which correspond to our feed forward start, attention, convolution, and feed forward end modules respectively[1]. Each of the described sharing indices $\mathcal{I}_x$, are subject to the following constraints: $|\mathcal{I}_x| = N$, $min(\mathcal{I}_x) = 1$, $max(\mathcal{I}_x) \leq N$. With these sharing indices, we can now define our i-th conformer block as:

$$\mathbb{C}_{(i)}\left(F_{start}^{\left(\mathcal{I}_{FS}^{(i)}\right)}, A^{\left(\mathcal{I}_{A}^{(i)}\right)}, C^{\left(\mathcal{I}_{C}^{(i)}\right)}, F_{end}^{\left(\mathcal{I}_{FE}^{(i)}\right)}\right). \tag{4}$$

Digging deeper into the structure of our conformer modules, even smaller sub-components can be found. In Fig. 1 we see that the largest sub-components are primarily linear layers. Beyond these sub-components, there are also some much smaller modules which have miniscule effect on decreasing the model size, yet may allow for better customization of our shared conformer layers. As such, we suggested that these smaller components can be excluded from our model sharing system, thus allowing improved model performance. Furthermore, we expect that certain sub-components may hold great importance in providing improved model performance and thus should not be shared even though they do contribute to increasing model size.

3.4. Low-Rank Factorization

As illustrated in Fig. 1, the largest sub-components within our conformer are normal linear layers composed of a weight matrix M and bias b. The complexity of the conformer comes from the specific architecture of the model rather than the complexity of the specific sub-components. Since the weight matrix is much larger in size than the bias, we will focus on reducing the size of M. Supposing that $M \in \mathbb{R}^{m \times n}$, we can apply low-rank decomposition [17] to reduce

[1] Other modules are ignored due to the large relative size of these modules.

	# Conformer Layers		WER		Model Size
	Physical	Virtual	dev	test	
SL0	1	1	10.69	10.53	2.55M
SL1	1	2	9.15	9.18	2.55M
SL2	1	3	8.39	8.71	2.55M
SL3	4	4	4.13	4.30	4.84M
SL4	4	8	3.50	3.76	4.84M
SL5	4	12	3.20	3.45	4.84M
SL6	4	16	3.31	3.67	4.84M

Table 1. Sharing full layers by passing layer output into itself.

M into three distinct sub-matrices: $U \in \mathbb{R}^{m \times k}$, $V \in \mathbb{R}^{n \times k}$, and $\Sigma \in \mathbb{R}^{k \times k}$ a diagonal matrix which can be found through:

$$\min_{U,\Sigma,V} ||M - U\Sigma V^T||, \tag{5}$$

using singular value decomposition (SVD). Given $k \ll \min(m, n)$, the number of parameters for any M in our model can be reduced.

While reducing k allows us to reduce the number of parameters for a given matrix, it also greatly increases the reconstruction error. To account for this, typically model fine-tuning is performed after decomposition to account for loss in model performance. In our case, we instead begin with a low-rank structure when training from scratch which allows us to forgo the need for training followed by fine tuning and also allows us to ignore the use of singular value decomposition. As such, for simplicity, Σ can be combined with U or V in our low-rank reconstruction structure as suggested in [24]. Thus, we can restate our reconstruction structure as $M \sim UV^T$.

4. EXPERIMENT DESIGN

4.1. Dataset

We evaluate on the LibriSpeech datasets [25] which consists of 960 hours of training data (i.e., train-clean and train-other) and we evaluate on dev-clean and test-clean. The spoken-word input data is structured as 80 log Mel-filterbank energy features with a window size of 25ms and a 10ms stride. The output is modelled using a word-piece model (WPM) embedding with a dimensionality of $1,024$.

4.2. Model Architecture

We begin our evaluations with a conformer architecture (B0) with 14M parameters and 16 conformer block layers consisting of 0.7M parameters each. This baseline architecture achieves a word-error rate (WER) of 2.18 on dev-clean and 2.53 on test-clean, however, our goal is to reduce this model down to approximately 5M parameters (a reduction of approximately -65%), thus this model is not applicable for always-on ambient ASR using low-power edge TPU devices. Notice, we begin with a 14M parameter model rather than a larger 100M+ parameter model since it is a common size for "small" models in the literature [4, 26]. We also design a handcrafted 5M parameter version of this model (B1) as a baseline with 8 conformer block layers with a size of 0.5M parameters each. This 5M baseline model achieves a WER of 3.53 on dev-clean and 3.72 on test-clean.

5. RESULTS

5.1. Repeat Full Layers

We begin our evaluations by reviewing the results of sharing full conformer layers. When sharing conformer layers, we repeat the conformer transformations in order over multiple repetitions. Suppose we have N conformer blocks repeated R times, we define the number of physical conformer layers as N and the number of virtual conformer layers as $N \times R$. By increasing R, we can achieve an increase in the number of transformations applied to our input data with the expectation that increasing the number of transformations will allow an improvement in the model quality. In Table 1, we begin by reviewing the model quality with a single physical conformer layer repeated different numbers of times. We observe that even with one physical conformer block (SL0-SL2), the WER decreases as the number of repetitions is increased (i.e., an increase in the number of virtual layers). With just 3 repetitions of the single conformer layer (SL2), our model is able to decrease the WER by -2.30 and -1.82 for dev and test respectively. Even so, the WER rates are still large, so while we demonstrated that increasing the number of virtual layers can improve the model quality, the quality is still good enough. To improve this, we increase to 4 physical conformer layers (SL3-SL6) which also brings our model size closer towards our goal of 5M parameters. We find that repeating the conformer block transformations three times (SL5) reduces the WER by -0.93 and -0.85 for dev and test respectively compared to just one iteration of each conformer block layer. However, we find that further increasing to four repetitions per conformer block (SL6) begins to degrade our model quality. Thus, there is a limit to the number of times conformer blocks should be repeated.

	Non-Shared Modules	Model Dim.	WER		Model Size
			dev	test	
SM0	F.F. Start	96	3.56	3.64	4.93M
SM1	Attention	128	3.19	3.48	4.99M
SM2	Convolution	136	3.13	3.35	5.03M
SM3	F.F. End	96	3.64	3.88	4.93M
SM4	Attention + Convolution	120	3.22	3.36	5.03M

Table 2. Sharing conformer layers (i.e., 4 Physical, 12 Virtual), while unsharing specific modules.

5.2. Sharing Conformer Modules

Next, we dig into the structure of the conformer blocks to identify the major modules which we can either enable or disable sharing. In Table 2, we use the shared conformer block model with 4 physical conformer layers repeated 3 times (SL5) as our base model and then select certain conformer modules to disable sharing (i.e., unshare). By unsharing individual modules, the model size increases, and thus, we must reduce the internal model dimension to compensate. Unsharing the convolution layer (SM2) offers the lowest WER rates at 3.13 and 3.35 for dev and test respectively. However, it is interesting to observe that unsharing the feed forward start (SM0) and feed forward end (SM3) modules significantly increases the WER rates. We can attribute this to the fact that both feed forward modules are so large in size, and thus, by not sharing these modules, we must greatly reduce the size of the model weight dimensionality hyperparameter to compensate.

5.3. Sharing Sub-Components

To further our understanding of how sharing of different components affects quality, we next look at disabling sharing (i.e., unsharing) for specific sub-components within our model. Again, we leverage the best model from Table 1 where we have 4 physical lay-

	Non-Shared Sub-Components		WER		Model Size
	Module	Sub-Component	dev	test	
SC0	F.F. Start	Linear (1)	3.10	3.23	6.02M
SC1	F.F. Start	Linear (2)	3.16	3.20	6.02M
SC2	Attention	Query	3.24	3.37	5.01M
SC3	Attention	Value	3.23	3.40	5.01M
SC4	Attention	Key	3.09	3.29	5.01M
SC5	Conv.	Pre-Conv.	5.49	5.81	5.17M
SC6	Conv.	Conv.	3.02	3.16	5.35M
SC7	Conv.	Post-Conv.	3.37	3.71	5.01M
SC8	F.F. End	Linear (1)	2.99	3.18	6.02M
SC9	F.F. End	Linear (2)	3.08	3.30	6.02M
SC10	All	Misc. Small	2.95	3.28	5.36M

Table 3. Effect of allowing certain conformer sub-components to be shared or not shared.

	# Conformer Layers		Rank (k)	WER		Model Size
	Physical	Virtual		dev	test	
LR0	4	4	N/A	4.13	4.30	4.84M
LR1	8	8	50	3.46	3.69	5.04M
LR2	12	12	20	3.70	3.75	4.98M
LR3	16	16	6	3.81	4.05	5.00M
LRS0	8	16	50	3.14	3.36	5.04M
LRS1	8	24	50	2.99	3.23	5.04M
LRS2	8	32	50	2.88	3.25	5.04M
LRS3	8	40	50	2.84	2.98	5.04M

Table 4. After applying low-rank architecture for feed forward modules. With and without sharing layers.

ers repeated three times giving a total of 12 virtual transformations (SL5) which achieved a WER of 3.20 for a model of size 4.84M. In Table 3, we unshare single weight variables at a time for each module. We see that unsharing these sub-components still keeps our model size close to 5M parameters except in the case of the linear sub-components in both feed forward start and end modules (SC0, SC1, SC8, and SC9). In addition to unsharing the individual module sub-components which were shown in Fig. 1, a number of other significantly smaller weights are also found within each module. These weights are small enough that they do not have a large impact on the overall model size (i.e., only an increase of 0.52M parameters). Thus, we also evaluate unsharing these miscellaneous small weights as well (SC10). We can see that unsharing the convolution sub-components within the convolution module allows for the lowest WER (SC6) while unsharing the other sub-components in the conformer layer each result in an increase in the WER. For the attention module, unsharing both query (SC2) and value (SC3) results in similar WER to the original model, yet unsharing key (SC4) does see a decrease in WER, thus implying that the attention key sub-components contains important information for our model.

5.4. Low-Rank (and Sharing)

Next we look towards low-rank architecture in Table 4. By reducing the k, we can subsequently achieve an increase in the number of physical layers. As we can see, with $k = 50$ (LR1), we are able to increase from 8 physical layers compared to only 4 when a low-rank architecture is not applied (LR0). We find that $k = 50$ also decreases the WER of the model by -0.67 and -0.61 for dev and test respectively. However, while we expect increasing the number of physical conformer layers should improve the quality, we find that k directly counteracts these WER improvements and thus while $k = 20$ (LR2) and $k = 6$ (LR3) achieve lower WER compared to the non low-rank architecture, they both perform worse than $k = 50$ (LR1). Continuing with $k = 50$ and the number of physical layers at 8, we also apply our layer sharing technique[2] to increase the number of virtual layers from 16 (LRS0) up until 40 (LRS3) by repeating each conformer layer. With this, we find WERs as low as 2.84 and 2.98 on dev and test are achievable at our bounds of 5M parameters when repeating the 8 physical conformer layers 5 times each (LRS3).

Model	WER		Model Size
	dev	test	
Conformer (S) [4] (B0)	2.18	2.53	14M
Handcrafted (B1)	3.53 (-0.00)	3.72 (-0.00)	4.9M
Share Layers (SL5)	3.20 (-0.33)	3.45 (-0.27)	4.84M
Share Modules (SM2)	3.13 (-0.40)	3.35 (-0.37)	5.09M
Shared Sub-C. (SC10)	2.95 (-0.58)	3.28 (-0.44)	5.36M
Low-Rank (LR2)	3.46 (-0.07)	3.69 (-0.03)	5.04M
L.R. Share (LRS3)	2.84 (-0.69)	2.98 (-0.74)	5.04M

Table 5. Overall best results for the evaluated compression methods.

5.5. Overview

Our overall best results for the evaluated methods are shown in Table 5. Each of our evaluated models was created based on an initial 14M parameter model (B0) described in [4]. We compare these models to a handcrafted 5M parameter model (B1) which was created by manually reducing hyperparameters (e.g., number of conformer blocks). The lowest overall WER was achieved by LRS3 because low-rank decomposition reduces the size of each physical conformer layer, thus allowing for a greater number of physical conformer layers while sharing layer weights through repeating offers an even greater number of virtual conformer layer transformations without increasing model size. While reducing model size does increase WER compared to larger models, our goal in this work is to create a model which fits completely within TPU memory, thus offering low-power, always-on ASR. With this, we can handle most ASR tasks, while defer to a larger model only when necessary.

6. CONCLUSION

In this work, we propose to reduce the size of Conformer-based models through parameter weight reuse at four levels: (i) repeating conformer block layer transformations, (ii) sharing specific conformer modules, (iii) sharing or not sharing sub-components per conformer module, and (iv) sharing low-rank decomposed sub-weights. By sharing model weight across layers, we find that we can increase the number of virtual transformations of our input data without further increasing the size of our model and thus we can retain our model in-memory for always-on ambient ASR leveraging low-power and low-resource neural accelerators such as edge TPU hardware. Through our evaluations, we find that sharing model weights and applying a low-rank Conformer architecture (LRS3) offers the greatest performance for our 5M parameter models, achieving a WER of 2.84 and 2.98 for LibriSpeech dev-clean and test-clean respectively.

[2] Preliminary results show only marginal improvements in combining low-rank and sub-component sharing due to the large search space.

7. REFERENCES

[1] Raziel Alvarez and Hyun-Jin Park, "End-to-end Streaming Keyword Spotting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6336–6340.

[2] Swayambhu Nath Ray, Minhua Wu, Anirudh Raju, Pegah Ghahremani, Raghavendra Bilgi, Milind Rao, Harish Arsikere, Ariya Rastrow, Andreas Stolcke, and Jasha Droppo, "Listen with Intent: Improving Speech Recognition with Audio-to-Intent Front-End," *arXiv preprint arXiv:2105.07071*, 2021.

[3] Mattia Antonini, Tran Huy Vu, Chulhong Min, Alessandro Montanari, Akhil Mathur, and Fahim Kawsar, "Resource Characterisation of Personal-Scale Sensing Models on Edge Accelerators," in *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, 2019, pp. 49–55.

[4] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, Helen Meng, Bo Xu, and Thomas Fang Zheng, Eds. 2020, pp. 5036–5040, ISCA.

[5] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, "Binarized Neural Networks," *Advances in neural information processing systems*, vol. 29, 2016.

[6] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both Weights and Connections for Efficient Neural Network," *Advances in neural information processing systems*, vol. 28, 2015.

[7] Steven M Hernandez and Eyuphan Bulut, "WiFi Sensing on the Edge: Signal Processing Techniques and Challenges for Real-World Systems," *IEEE Communications Surveys & Tutorials*, 2022.

[8] Kunran Xu, Huawei Zhang, Yishi Li, Yuhao Zhang, Rui Lai, and Yi Liu, "An Ultra-low Power TinyML System for Real-time Visual Processing at Edge," *arXiv preprint arXiv:2207.04663*, 2022.

[9] Colby R Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, et al., "Benchmarking TinyML Systems: Challenges and Direction," *arXiv preprint arXiv:2003.04821*, 2020.

[10] Zhaofeng Wu, Ding Zhao, Qiao Liang, Jiahui Yu, Anmol Gulati, and Ruoming Pang, "Dynamic Sparsity Neural Networks for Automatic Speech Recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6014–6018.

[11] Shaojin Ding, Tianlong Chen, and Zhangyang Wang, "Audio Lottery: Speech Recognition Made Ultra-Lightweight, Noise-Robust, and Transferable," in *International Conference on Learning Representations*, 2021.

[12] Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard, "Adaptive Quantization for Deep Neural Network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, vol. 32.

[13] Pierre-Emmanuel Novac, Ghouthi Boukli Hacene, Alain Pegatoquet, Benoît Miramond, and Vincent Gripon, "Quantization and Deployment of Deep Neural Networks on Microcontrollers," *Sensors*, vol. 21, no. 9, pp. 2984, 2021.

[14] Shaojin Ding, Phoenix Meadowlark, Yanzhang He, Lukasz Lew, Shivani Agrawal, and Oleg Rybakov, "4-bit Conformer with Native Quantization Aware Training for Speech Recognition," in *Proc. Interspeech 2022*, 2022, pp. 1711–1715.

[15] Gianmarco Cerutti, Rahul Prasad, Alessio Brutti, and Elisabetta Farella, "Neural Network Distillation on IoT Platforms for Sound Event Detection," in *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, Gernot Kubin and Zdravko Kacic, Eds. 2019, pp. 3609–3613, ISCA.

[16] Ze Yang, Linjun Shou, Ming Gong, Wutao Lin, and Daxin Jiang, "Model Compression with Two-stage Multi-teacher Knowledge Distillation for Web Question Answering System," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 690–698.

[17] Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin, "Language model compression with weighted low-rank factorization," *CoRR*, vol. abs/2207.00112, 2022.

[18] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao, "On Compressing Deep Models by Low Rank and Sparse Decomposition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7370–7379.

[19] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li, "Learning Structured Sparsity in Deep Neural Networks," *Advances in neural information processing systems*, vol. 29, 2016.

[20] Clemens JS Schaefer, Siddharth Joshi, Shan Li, and Raul Blazquez, "Edge Inference with Fully Differentiable Quantized Mixed Precision Neural Networks," *arXiv preprint arXiv:2206.07741*, 2022.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is All you Need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] Raj Dabre and Atsushi Fujita, "Recurrent Stacking of Layers for Compact Neural Machine Translation Models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 6292–6299.

[23] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," *arXiv preprint arXiv:1909.11942*, 2019.

[24] Jian Xue, Jinyu Li, and Yifan Gong, "Restructuring of Deep Neural Network Acoustic Models with Singular Value Decomposition," in *Interspeech*, 2013, pp. 2365–2369.

[25] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2015.

[26] Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu, "ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context," *Proc. Interspeech 2020*, pp. 3610–3614, 2020.