INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Compressing 1D Time-Channel Separable Convolutions using Sparse Random Ternary Matrices

Gonçalo Mordido[1*], *Matthijs Van keirsbilck*[2], *and Alexander Keller*[2]

[1]Hasso Plattner Institute, Germany
[2]NVIDIA, Germany

goncalo.mordido@hpi.de, matthijsv@nvidia.com, akeller@nvidia.com

Abstract

We demonstrate that 1x1-convolutions in 1D time-channel separable convolutions may be replaced by constant, sparse random ternary matrices with weights in $\{-1, 0, +1\}$. Such layers do not perform any multiplications and do not require training. Moreover, the matrices may be generated on the chip during computation and therefore do not require any memory access. With the same parameter budget, we can afford deeper and more expressive models, improving the Pareto frontiers of existing models on several tasks. For command recognition on Google Speech Commands v1, we improve the state-of-the-art accuracy from 97.21% to 97.41% at the same network size. Alternatively, we can lower the cost of existing models. For speech recognition on Librispeech, we halve the number of weights to be trained while only sacrificing about 1% of the floating-point baseline's word error rate.

1. Introduction

Speech and command recognition tasks tend to have strict low-latency requirements, which may be challenging to meet while using deep neural networks. Specifically, the employment of such networks on edge devices, which are bandwidth, energy, and area constrained, is often too costly in practical applications. Sparsity and quantization techniques are viable solutions, reducing the model size and computational cost while being suited for existing hardware.

With this motivation in mind, we combine sparsity and quantization to make compact speech and command recognition models even more compact. In particular, we leverage recent residual networks [1] containing time-channel separable convolutions [2, 3, 4]. Despite such compact designs, most weights are on the 1x1-convolutions in the residual paths, which are equivalent to fully-connected layers applied to every time step. Hence, we propose to decrease the memory and computational cost by replacing these trained floating-point weights by constant, sparse random ternary matrices.

Our approach enables (i) *faster training*, since the weights need not be updated, (ii) *faster on-chip inference*, due to sparse operations with no multiplications, and (iii) *higher compression rates*, because weights may not be stored but just computed on-the-fly resulting in (iv) *reduced memory bandwidth*.

At a similar network size, we outperform several floating-point baselines on multiple data sets and tasks. We improve the current state-of-the-art on speech command recognition from 97.21% to 97.41% using MatchboxNet [3] on Google Speech Commands [5]. On AN4 [6], we improve the Pareto efficiency in several floating-point configurations of QuartzNet [2], outperforming deeper and wider baselines at only $\approx 25\%$ and $\approx 65\%$ of the network size, respectively. On Librispeech [7], when reducing the model size from 19M to 11M parameters we lose only 0.3% accuracy, achieving 4.30% word error rate (WER) on dev-clean, as compared to the 3.98% WER baseline.

*Work done during a research internship at NVIDIA.

2. Related work

After the advances of neural networks in the image domain, they have recently gained popularity in speech recognition. Specifically, recurrent [8] and convolutional [9] neural networks, as well as a combination of both [10], quickly have become common. Encoder-decoder attention networks have also been proposed [11], and, following their adoption in the language domain, Transformers have been used for speech tasks as well [12, 13]. However, recurrent networks are less suitable for parallel hardware due to their sequential nature, and attention mechanisms have a high computation and memory footprint.

Recent works show that convolutional networks may outperform recurrent networks even on sequence tasks [14] while being better suited to modern parallel hardware. There have been many advances in the design of efficient convolutional models, especially in the computer vision community [15, 16]. Similar network designs have been adapted to speech tasks, replacing 2D-convolutions by 1D-convolutions, *e.g.* in Jasper [17], a residual convolutional network [1].

Depth-wise separable convolutions have fewer parameters than standard convolutions while improving generalization performance [15, 18]. Directly applying these lightweight convolutions to Jasper results in QuartzNet [2], which reduces the model size by $17\times$ at similar accuracy. QuartzNet's residual blocks consist of time-channel separable convolutions, *i.e.* across the time dimensions (temporal-convolutions) and across the feature dimensions (1x1-convolutions), followed by batch normalization (BN) and rectified linear units (ReLU). MatchboxNet [3] adopts this design to achieve state-of-the-art results on some speech command recognition data sets. Similar models have also been proposed for speaker recognition and verification [4] and cross-language learning and domain adaption [19].

On top of compact architectural design, other compression techniques may be used to further improve model efficiency. While pruning [20] attempts to find and remove unimportant weights, neurons, or layers, factorization [21] compresses the weights using a low-dimensional approximation, and quantization [22, 23] allows one to store and perform operations in low-bit precision reducing the computational cost significantly [24]. Considering ternary weights, *e.g.* $\{-1, 0, 1\}$, multiplications are reduced to bit-wise, sparse operations in hardware, which are far less costly in terms of energy, speed, and area [25].

In another line of work, it has been shown that neural networks with random weights, *e.g.* echo state networks [26] and extreme learning machines [27], can have surprisingly power-

http://dx.doi.org/10.21437/Interspeech.2021-141

ful capabilities [28]. These findings have also been shown to translate to automatic speech recognition tasks [29].

In this work, we combine ternary quantization with random weights by replacing weights in 1x1-convolutions with constant, random ternary weights. As these weights neither need to be stored nor updated, a significant reduction in the memory and compute costs is possible for both training and inference.

3. Random ternary matrices

By analyzing the weights of the 1x1-convolutions in the sequence of residual blocks, we observe that they do not expose any obvious visual structure. Specifically, after applying trained ternarized quantization (TTQ [25]), the resulting trained matrices maintain model performance even though they can hardly be distinguished from random ternary matrices (see Figure 1).

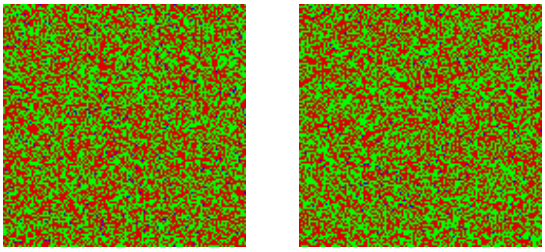

Figure 1: *A trained ternary quantized weight matrix of a 1x1-convolution [25] (left) can hardly be distinguished from a randomly generated ternary matrix (right).*

As previously mentioned, neural networks with random weights show powerful capabilities [26, 27, 30]. Therefore, we propose to use random ternarized matrices for the 1x1-convolutions in the residual blocks by keeping them constant while training the rest of the network. For each neuron, the ternarized weights simply compute a scaled difference of sums

$$\text{ReLU}\left(\text{BN}\left(\sum_{i\in I^+} x_i - \sum_{i\in I^-} x_i\right)\right), \qquad (1)$$

where the set I^+ contains all the indices of the connections with the weight $+1$ and I^- contains all the indices of the connections with the weight -1. The union $I := I^- \cup I^+$ may be sparse, *i.e.* may not contain the indices of all neurons in the previous layer. This scaled sum is a far cheaper operation than the traditional floating-point matrix multiplication.

Similar to echo state networks [26], a sufficient routing capacity is enabled by having trained layers before and after the constant random ternary layers. This matches existing observations [25, 22], where starting quantization too early in the network degrades performance. With this in mind, we only ternarize and do not update the weights in the 1x1-convolutions inside the residual blocks (which possess the majority of the weights). Since the weights of the preceding temporal-convolution and the subsequent batch normalization layers are still updated, a scaling of the inputs and outputs may be learned, respectively.

Implicitly, the tensor product of the aforementioned scaling vectors defines the weights of the ternary matrix, empowering our approach. We note that the ternary matrix in combination with batch normalization is reminiscent of *operational amplifiers*, computing a difference of sums that is amplified before applying a non-linearity, *e.g.* ReLU. Moreover, our algorithm is related to extreme learning machines [27], where a more theoretical foundation can be derived from [31, Algorithm ELM].

3.1. Implementation details

Given a randomly initialized matrix, the random (and constant) ternary matrix may be generated by applying the thresholding procedure presented by Zhu et al. [25]. A parameter t then serves as a threshold, where all absolute values less than t are quantized to zero and the remaining values are quantized to -1 and +1 according to their sign. If the matrices are initialized using uniformly distributed samples from $[-1, 1]$, $t \in [0, 1]$ will be the expected fraction of sparsity of the matrix: $t = 0$ yields a dense, binary matrix, $t = 1$ yields a zero matrix, and $t \in]0, 1[$ yields a ternary matrix. Listing 1 shows a simple implementation of our approach using PyTorch.

Listing 1: *Example implementation given an uniformly distributed weight from* $[-1, 1]$ *and a threshold* $\mathrm{t} \in [0, 1]$.

```
# compute signs
pos = (weight.data > 0.).int2()
neg = (weight.data < 0.).int2()

# compute mask
mask = (torch.abs(weight.data) > t).int2()

# make ternary
weight.data = (pos - neg) * mask

# do not update weight during training
weight.requires_grad = False
```

3.2. Implementation variants

Our approach is widely applicable and suited to existing hardware, allowing for a range of efficient implementations. Specifically, the random ternary matrices may be computed on-the-fly on the chip without accessing memory for weights. For example, a pseudo-random number generator may be used to determine the ternary values, where, given an initial state, a specific matrix may be deterministically generated at any time. Alternatively, each ternary value may be determined by a hash function of the matrix entry coordinates, where different matrices may be realized by different hash functions.

Specific, structured sparsity patterns supported by certain accelerators[1] may be applied instead, taking full advantage of hardware support. Further compute acceleration in the ternary layers may be leveraged by custom layer implementations.

4. Results

We conducted experiments using MatchboxNet-NxMxW [3] on Google Speech Commands (GSC) [5] and QuartzNet-NxMxW [2] on AN4 [6] for 200 epochs and Librispeech [7] for 300 epochs. As for the notation, N stands for the number of residual blocks, M for the number of depth-wise separable convolutions in the residual blocks, and W for the number of channels. We refer to the original papers for additional architectural details.

GSCv1 consists of 65k one-second spoken utterances from various speakers. There are 30 command classes chosen for possible IoT or robotics application scenarios, numerical digits, as well as phoneme variety and command similarity. AN4

[1] https://developer.nvidia.com/blog/exploiting-ampere-structured-sparsity-with-cusparselt/

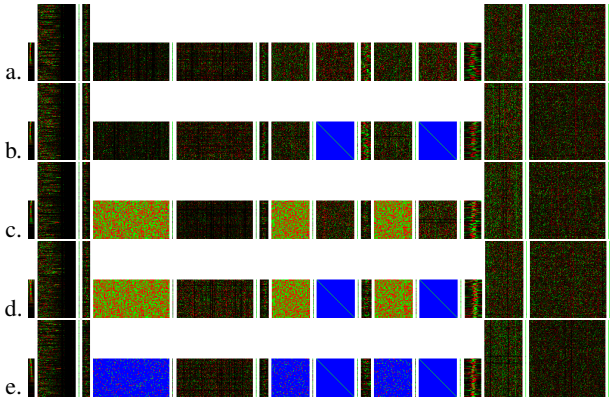

Figure 2: *MatchboxNet-3x1x64 matrices: colors indicate negative weights (red) or positive weights (green). To illustrate sparsity, zero is shown in blue in constant matrices. Row a) shows the baseline's trained floating-point weights; b) uses identity skip links; c) replaces the 1x1-convolutions in residual layers by constant, random ternary matrices in which red is -1 and green is $+1$; d) combines b) and c); e) uses sparse ternary weights.*

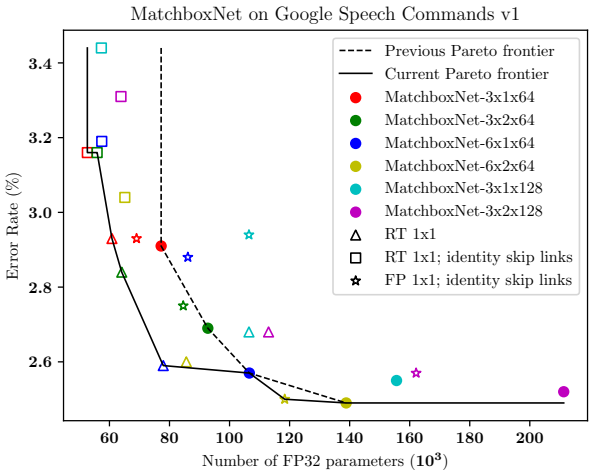

Figure 3: *Test error rate of different MatchboxNet configurations with respect to the number of model parameters on GSCv1. "FP" refers to floating-point and "RT" to random ternary weights. Our constant, random ternary matrices improve the Pareto frontier. Restricting the skip links to identity matrices greatly reduces the network size but increases error.*

is an alphanumeric data set consisting of around 1k utterances averaging 3 seconds long, describing personal information and control words. Librispeech contains 10k hours of segmented and aligned narrations of public audiobooks and is commonly used to evaluate large-scale speech recognition models.

In our experiments, we compared our simplified networks to the original baselines under identical training settings, *i.e.* without fine-tuning. We adopted the training setups publicly available at NVIDIA NeMo[2], so that our results may be easily replicated as well as applied to additional models and tasks.

4.1. Speech command recognition (MatchboxNet on GSC)

Figure 2 shows the weight matrices resulting from multiple model size reductions. Figure 3 compares the test accuracy of several MatchboxNet configurations with respect to the number of model parameters. With 200 epochs, our MatchboxNet-3x1x64 baseline achieves 97.09% test accuracy, which is close to the originally reported mean of 97.21% accuracy over five trials [3]. We observe that using random ternary matrices improves the Pareto efficiency of the models, reaching a better trade-off of error rate for model size and cost. Hence, using random ternary weights with and without identity skip links allows us to increase model width and depth efficiently. We trained our models with varied t values and plot the best models for each configuration. Note that the sparsity levels t have little influence on the overall performance as discussed later in Section 4.2.1.

Although ternarizing the skip links reduces performance in our experiments, using identity skip links avoids the otherwise quadratic weight matrices and performs close to the baseline for the smaller model configurations. The skip links even may be omitted. Yet deeper models train better with skip links [1]. Interestingly, tests using constant floating-point random matrices did not result in any improvements as compared to our more efficient random ternary matrices.

Table 1 compares our random ternary variants to the floating-point models reported by Majumdar et al. [3]. At 77K parameters, our random ternary MatchboxNet-6x1x64 sets a new state-of-the-art of 97.41% accuracy. Our smaller variants further reduce model size while maintaining high accuracy.

Table 1: *Accuracy and model size on Google Speech Commands version 1. "FP" denotes floating-point weights and "RT" denotes our constant, random ternary weights.*

Model	FP32 param.	Acc. (%)
DenseNet-BC-100 [32]	$800 \cdot 10^3$	96.77
ResNet-15 [33]	$238 \cdot 10^3$	95.80
EdgeSpeechNet-A [34]	$107 \cdot 10^3$	96.80
MatchboxNet-3x1x64 [3] (FP)	$77 \cdot 10^3$	97.21
MatchboxNet-6x1x64 (RT)	$77 \cdot 10^3$	97.41
MatchboxNet-3x2x64 (RT)	$64 \cdot 10^3$	97.16
MatchboxNet-3x1x64 (RT)	$60 \cdot 10^3$	97.07

4.2. Automatic speech recognition

In Section 4.2.1, we use the fast iterations enabled by using the small AN4 data set to study a range of sparsity levels as well as the Pareto efficiency on multiple QuartzNet configurations. We then extrapolate these findings and evaluate our method against large-scale QuartzNet models on Lirispeech (Section 4.2.2).

4.2.1. QuartzNet on AN4

Figure 4 shows that at 90% average sparsity, i.e. $t = 0.9$, different configurations of QuartzNet perform comparably to the less computationally efficient fully dense models. We find a similar pattern for MatchboxNet. Therefore, we can choose a highly sparse network to minimize memory and compute cost. The performance starts deteriorating with increased sparsity levels, as the 1x1-convolutions eventually become zero matrices.

Figure 5 compares several QuartzNet configurations in

[2]https://github.com/NVIDIA/NeMo

terms of word error rate and network size. Our constant random ternary variants, with and without identity skip links, create a new Pareto frontier improving the accuracy of smaller models.

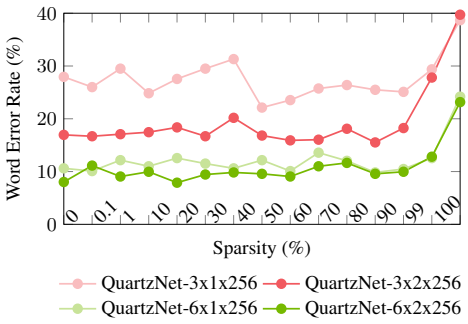

Figure 4: *Word error rate of different QuartzNet models with sparse ternary weights on the AN4 data set for varying sparsity levels. High sparsity levels deliver performance similar to fully connected layers, especially for bigger networks, that are less sensitive to sparsity.*

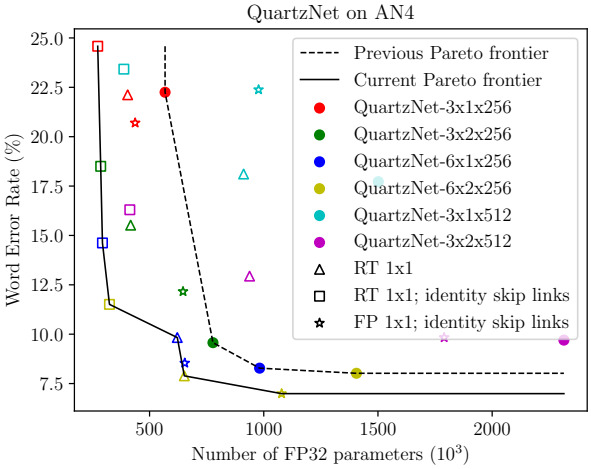

Figure 5: *Test word error rate of different QuartzNet configurations with respect to the number of model parameters on AN4. "FP" refers to floating-point weights and "RT" to constant, random ternary weights. The best Pareto front is formed by our random ternary variants, with and without identity skip links.*

4.2.2. *QuartzNet on Librispeech*

Using the original QuartzNet-15x5 [2], which has 15 residual blocks, we randomize and ternarize all 1x1-convolutions in the last 6 blocks, reducing the number of trainable parameters by half. Since early layers contain fewer parameters and are more sensitive to quantization [24, 22], compressing later layers presents greater benefits. We trained the skip links and used $t = 0.001$, although sweeping for optimal t values (similar to our previous experiments) would likely improve results.

Table 2 compares the greedy WER of our variant to the bigger models as reported by Kriman et al. [2]. Despite reducing the number of floating-point parameters by half, our variant achieves competitive results. Namely, the WER is only increased by $\approx 1\%$ (test-clean) as compared to the floating-point QuartzNet-15x5 baseline. By also learning the skip links, we manage to reduce the WER by $\approx 0.5\%$ (test-clean) while only learning 2M additional parameters.

We compared our variants to QuartzNet configurations reported by Kriman et al. [2] in Table 3. At identical network sizes, we observe that our random ternary variants either maintain or increase parameter efficiency on dev-clean and dev-other, respectively, especially when also learning the skip links.

Table 2: *Word error rate and model size on Librispeech. "FP" denotes floating-point weights and "RT" denotes our random, ternary weights.*

Model	FP32 param.	Test clean	Test other
JasperDR-10x5 [17]	$333 \cdot 10^6$	4.32	11.82
TDS Conv [35]	$37 \cdot 10^6$	5.36	15.64
QuartzNet-15x5 [2] (FP)	$19 \cdot 10^6$	3.90	11.28
QuartzNet-15x5 (RT, skip FP)	$11 \cdot 10^6$	4.49	13.21
QuartzNet-15x5 (RT)	$9 \cdot 10^6$	4.93	14.21

Table 3: *Word error rate and model size of several QuartzNet variants on Librispeech. "FP" means floating-point weights and "RT" denotes our random, ternary weights. Extrapolation is obtained from the Pareto frontier of the FP baselines to match the model size of our smallest ternary variant.*

Model	FP32 param.	Dev clean	Dev other
QuartzNet-15x5 [2] (FP)	$19 \cdot 10^6$	3.98	11.58
QuartzNet-10x5 [2] (FP)	$13 \cdot 10^6$	4.14	12.33
QuartzNet (FP, extrapolated)	$9 \cdot 10^6$	4.84	14.20
QuartzNet-5x5 [2] (FP)	$7 \cdot 10^6$	5.39	15.69
QuartzNet-15x5 (RT, skip FP)	$11 \cdot 10^6$	4.30	12.87
QuartzNet-15x5 (RT)	$9 \cdot 10^6$	4.84	13.75

5. Conclusion and future work

Random ternary 1x1-convolutions improve the efficiency of speech residual networks, reducing memory and computational cost significantly during training and inference. Our method is energy efficient and simple to support in hardware: while computation may be realized on-chip without multiplications, memory access may be avoided by computing the weights on-the-fly. Hence, our random ternary layers are a simple, yet effective way to further compact existing network designs.

Due to its similarity to operational amplifiers, our algorithm may be explored in analog hardware in the future. Testing the applicability of our approach to other compact convolutional speech models, *e.g.* Conformer [13], is worth investigating.

6. Acknowledgements

The third author is very thankful to Cédric Villani for a discussion on structure to be discovered in neural networks during the AI for Good Global Summit 2019 in Geneva. This work has been partially funded by the Federal Ministry of Education and Research (BMBF, Germany) in the project Open Testbed Berlin - 5G and Beyond - OTB-5G+ (Förderkennzeichen 16KIS0980).

7. References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 1, 2, 4.1

[2] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang, "Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6124–6128. 1, 1, 2, 4, 4.2.2, 2, 3

[3] S. Majumdar and B. Ginsburg, "Matchboxnet–1d time-channel separable convolutional neural network architecture for speech commands recognition," *CoRR*, vol. abs/2004.08531, 2020. 1, 1, 2, 4, 4.1, 4.1, 1

[4] N. R. Koluguri, J. Li, V. Lavrukhin, and B. Ginsburg, "Speakernet: 1d depth-wise separable convolutional network for text-independent speaker recognition and verification," 2020. 1, 2

[5] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018. 1, 4

[6] A. Acero, *Acoustical and environmental robustness in automatic speech recognition*. Springer Science & Business Media, 2012, vol. 201. 1, 4

[7] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210. 1, 4

[8] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE, 2013, pp. 273–278. 2

[9] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014. 2

[10] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182. 2

[11] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 4945–4949. 2

[12] L. Dong, S. Xu, and B. Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5884–5888. 2

[13] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020. 2, 5

[14] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018. 2

[15] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobilenetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520. 2

[16] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114. 2

[17] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J. M. Cohen, H. Nguyen, and R. T. Gadde, "Jasper: An end-to-end convolutional neural acoustic model," *CoRR*, vol. abs/1904.03288, 2019. 2, 2

[18] K. Hayashi, T. Yamaguchi, Y. Sugawara, and S.-i. Maeda, "Einconv: Exploring unexplored tensor network decompositions for convolutional neural networks," *arXiv preprint arXiv:1908.04471*, 2019. 2

[19] J. Huang, O. Kuchaiev, P. O'Neill, V. Lavrukhin, J. Li, A. Flores, G. Kucsko, and B. Ginsburg, "Cross-language transfer learning, continuous learning, and domain adaptation for end-to-end automatic speech recognition," 2020. 2

[20] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016. 2

[21] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks." in *Interspeech*, 2018, pp. 3743–3747. 2

[22] G. Mordido, M. V. keirsbilck, and A. Keller, "Instant quantization of neural networks using Monte Carlo methods," *NeurIPS 2019 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (NeurIPS 2019 EMC^2)*, 2019. 2, 3, 4.2.2

[23] ——, "Monte carlo gradient quantization," *CVPR 2020 Joint Workshop on Efficient Deep Learning in Computer Vision (CVPR 2020 EDLCV)*, 2020. 2

[24] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2015. 2, 4.2.2

[25] C. Zhu, S. Han, H. Mao, and W. Dally, "Trained ternary quantization," *CoRR*, vol. abs/1612.01064, 2016. [Online]. Available: http://arxiv.org/abs/1612.01064 2, 3, 1, 3, 3.1

[26] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001. 2, 3, 3

[27] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006. 2, 3, 3

[28] C. Gallicchio and S. Scardapane, "Deep randomized neural networks," *Recent Trends in Learning From Data*, pp. 43–68, 2020. 2

[29] H. Shrivastava, A. Garg, Y. Cao, Y. Zhang, and T. Sainath, "Echo state speech recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5669–5673. 2

[30] M. D. Skowronski and J. G. Harris, "Automatic speech recognition using a predictive echo state network classifier," *Neural networks*, vol. 20, no. 3, pp. 414–423, 2007. 3

[31] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 513–529, April 2012. 3

[32] B. Li, F. Wu, S.-N. Lim, S. Belongie, and K. Q. Weinberger, "On feature normalization and data augmentation," *arXiv preprint arXiv:2002.11102*, 2020. 1

[33] J. Tang, Y. Song, L. Dai, and I. V. McLoughlin, "Acoustic modeling with densely connected residual network for multichannel speech recognition," in *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018*, B. Yegnanarayana, Ed. ISCA, 2018, pp. 1783–1787. [Online]. Available: https://doi.org/10.21437/Interspeech.2018-1089 1

[34] Z. Q. Lin, A. G. Chung, and A. Wong, "Edgespeechnets: Highly efficient deep neural networks for speech recognition on the edge," *arXiv preprint arXiv:1810.08559*, 2018. 1

[35] A. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions," in *Proc. Interspeech 2019*, 2019, pp. 3785–3789. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2460 2