INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Tensor decomposition for minimization of E2E SLU model toward on-device processing

Yosuke Kashiwagi[1], *Siddhant Arora*[2], *Hayato Futami*[1], *Jessica Huynh*[2], *Shih-Lun Wu*[2], *Yifan Peng*[2], *Brian Yan*[2], *Emiru Tsunoo*[1], *Shinji Watanabe*[2]

[1] Sony Group Corporation, Japan,
[2] Carnegie Mellon University, USA
yosuke.kashiwagi@sony.com

Abstract

Spoken Language Understanding (SLU) is a critical speech recognition application and is often deployed on edge devices. Consequently, on-device processing plays a significant role in the practical implementation of SLU. This paper focuses on the end-to-end (E2E) SLU model due to its small latency property, unlike a cascade system, and aims to minimize the computational cost. We reduce the model size by applying tensor decomposition to the Conformer and E-Branchformer architectures used in our E2E SLU models. We propose to apply singular value decomposition to linear layers and the Tucker decomposition to convolution layers, respectively. We also compare COMP/PARFAC decomposition and Tensor-Train decomposition to the Tucker decomposition. Since the E2E model is represented by a single neural network, our tensor decomposition can flexibly control the number of parameters without changing feature dimensions. On the STOP dataset, we achieved 70.9% exact match accuracy under the tight constraint of only 15 million parameters.

Index Terms: spoken language understanding, E2E, on-device, sequential distillation, tensor decomposition, STOP

1. Introduction

Spoken language understanding (SLU) is one of the essential applications of speech recognition. SLU is used in voice interfaces such as smart speakers, and improving its accuracy is required for usability [1–5]. On the other hand, these voice interfaces often work on edge devices due to latency and privacy issues. Such on-device processing requires smaller models to preserve power consumption. Therefore, miniaturization of the SLU model for on-device processing is a critical issue for voice interfaces [6–15].

For example, Radfar et al. reduce parameters by sharing the audio encoder and estimate slot tags, values, and intents [2, 6]. Also, Le et al. model NLU efficiently based on RNN-T using embedded features of both the encoder and the predictor of RNN-T [7]. In addition, Tyagi et al. propose early decision-making using the BranchyNet scheme to address the latency and computational complexity issues [9]. Although these are practical approaches, many are based on the traditional, relatively simple E2E model structure. On the other hand, recent E2E models have been used with more complex structures by combining convolution and self-attention operations such as Conformer [16] and E-Branchformer [17, 18]. Therefore, in addition to using singular value decomposition (SVD) [19–21] for two-dimensional matrices in the self-attention network, convolution network having high-order tensors requires tensor decomposition techniques. We specifically target the Conformer and E-Branchformer-based E2E SLU model, which has demonstrated high accuracy in the SLU task [17, 22].

Tensor decomposition techniques are widely used for model miniaturization [23]. Although various tensor decomposition techniques have been proposed, we mainly explore SVD, and Tucker decomposition [24] to target on-device fast processing. They enable inference without reconstructing the original parameter tensors from the factored tensors. Decomposition with smaller ranks can then be used to reduce computational complexity, enabling a reduction in the number of parameters and faster computation [25]. The paper demonstrates effective combinations of model compression techniques for Conformer and E-Branchformer, specifically through the use of tensor decompositions such as Tucker decomposition. This is because these models have convolution layers with higher-order tensors as parameters, making them more efficiently compressed using these techniques. By defining the model compression ratio, we show that the model size can be flexibly changed by determining the rank of the decomposition from this ratio. Furthermore, we evaluate CANDECOMP/PARAFAC (CP) decomposition [26] and Tensor-Train decomposition [27] when used instead of the Tucker decomposition.

2. Tensor decomposition

2.1. Singular value decomposition

The SVD-based parameter reduction technique is stable and widely used [19–21]. The SVD is applied to the weight matrix $\boldsymbol{W} \in \mathbb{R}^{I \times J}$ with low-rank bases as:

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}, \tag{1}$$

where $\boldsymbol{S}$ is a diagonal matrix. The matrix size of each $\boldsymbol{U}$, $\boldsymbol{S}$ and $\boldsymbol{V}$ is $I \times R$, $R \times R$ and $R \times J$, respectively. The parameter size can be controlled by changing R. Since $\boldsymbol{S}$ is a square matrix, $\boldsymbol{U}\boldsymbol{S}$ or $\boldsymbol{S}\boldsymbol{V}$ can be pre-composed to reduce the parameters. In our model, we composed $\boldsymbol{S}\boldsymbol{V}$. Thus, the final number of decomposed parameters can be reduced into $IR+RJ$. Therefore, the parameter compression ratio γ_{svd} can be described as:

$$\gamma_{\text{svd}} = \frac{IR + RJ}{IJ}. \tag{2}$$

2.2. Tucker decomposition

Tucker decomposition [24] is known to be effective for applying high-order tensors. Here we discuss the case of 1d convolution, which has a 3-dimensional parameter tensor. The parameter tensor of the convolution $\boldsymbol{W} \in \mathbb{R}^{I \times J \times K}$ can be decomposed into a core tensor $\boldsymbol{C} \in \mathbb{R}^{R \times S \times T}$ and factor matrices $\boldsymbol{U}^1 \in \mathbb{R}^{I \times R}$, $\boldsymbol{U}^2 \in \mathbb{R}^{J \times S}$ and $\boldsymbol{U}^3 \in \mathbb{R}^{K \times T}$ using Tucker

10.21437/Interspeech.2023-1299

Algorithm 1 Size halving for Tucker decomposition

Require: $\hat{\gamma}_{\text{tucker}} > 0.0$
$R \leftarrow I, S \leftarrow J, T \leftarrow K$
$\gamma_{\text{tucker}} \leftarrow \infty$
while $\gamma_{\text{tucker}} > \hat{\gamma}_{\text{tucker}}$ **do**
  $R \leftarrow R/2, S \leftarrow S/2, T \leftarrow T/2$
  if $R < 1$ **then**
    $R \leftarrow 1$
  end if
  if $S < 1$ **then**
    $S \leftarrow 1$
  end if
  if $T < 1$ **then**
    $T \leftarrow 1$
  end if
  $\gamma_{\text{tucker}} \leftarrow \frac{RST+IR+JS+KT}{IJK}$
end while

decomposition as:

$$\boldsymbol{W} = \sum_{r,s,t} \boldsymbol{C}_{r,s,t} \times \boldsymbol{U}^1_r \otimes \boldsymbol{U}^2_s \otimes \boldsymbol{U}^3_t, \tag{3}$$

where $\otimes$ describes a Kronecker product. The parameter compression ratio γ_{tucker} can be described as:

$$\gamma_{\text{tucker}} = \frac{RST + IR + JS + KT}{IJK}. \tag{4}$$

2.3. CP decomposition

CP decomposition [26] decomposes tensor $\boldsymbol{W} \in \mathbb{R}^{I\times J\times K}$ into a weight vector $\lambda \in \mathbb{R}^R$ and factor matrices $\boldsymbol{U}^1 \in \mathbb{R}^{I\times R}$, $\boldsymbol{U}^2 \in \mathbb{R}^{J\times R}$ and $\boldsymbol{U}^3 \in \mathbb{R}^{K\times R}$ as:

$$\boldsymbol{W} = \sum_r \lambda_r \times \boldsymbol{U}^1_r \otimes \boldsymbol{U}^2_r \otimes \boldsymbol{U}^3_r. \tag{5}$$

The parameter compression ratio γ_{cp} can be described as:

$$\gamma_{\text{cp}} = \frac{R(1 + I + J + K)}{IJK}. \tag{6}$$

The CP decomposition is the special case where the core tensor of the Tucker decomposition has only diagonal component values and the same rank for each dimension in eq. (3). Therefore, it allows for more parameter reduction but is less expressive than the Tucker decomposition. Furthermore, the rank of the CP decomposition is restricted to the smallest dimension. Since the kernel size of the convolution layer is often small compared to the number of channels, the rank is often limited to the kernel size. Therefore, CP decomposition is difficult to adjust with flexible parameter sizes.

2.4. Tensor-Train decomposition

Tensor-Train [27] is another decomposition method for high-order tensor. The original tensor $\boldsymbol{W} \in \mathbb{R}^{I\times J\times K}$ is decomposed into a product of 3-order tensors $\boldsymbol{G}^1_{1,i,r} \in \mathbb{R}^{1\times I\times R}$, $\boldsymbol{G}^2_{r,j,s} \in \mathbb{R}^{R\times J\times S}$ and $\boldsymbol{G}^3_{s,k,1} \in \mathbb{R}^{S\times K\times 1}$ by the Tensor-Train decomposition as:

$$\boldsymbol{W}_{i,j,k} = \sum_{r,s} \boldsymbol{G}^1_{1,i,r} \times \boldsymbol{G}^2_{r,j,s} \times \boldsymbol{G}^3_{s,k,1}. \tag{7}$$

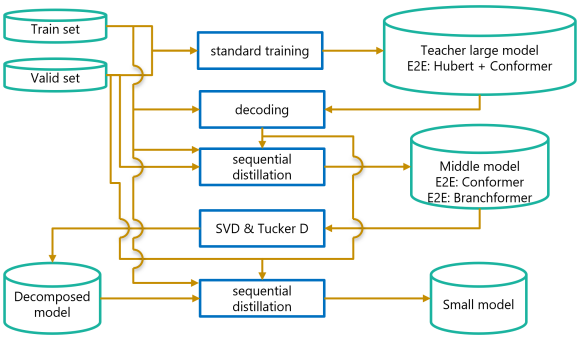

Figure 1: *Training procedure.*

The parameter compression ratio γ_{tt} can be described as:

$$\gamma_{\text{tt}} = \frac{IR + RJS + SK}{IJK}. \tag{8}$$

Tensor-Train decomposition provides flexible parameter reduction, as it involves $N-1$ hyperparameters corresponding to the rank of N-th order tensors. However, unlike Tucker decomposition, it cannot accelerate inference because the original tensor must be reconstructed during inference.

3. Decomposed E2E SLU model

3.1. Training overview

We investigate the tensor decomposition techniques in E2E SLU models with higher-order tensors as parameters. E2E SLU is a task that estimates semantic label sequences $\boldsymbol{Y}$ directly from input speech feature sequences $\boldsymbol{X}$. Fig.1 describes the overview of the training procedure of our model. We use two-step sequential distillation to train a small model. Recent SOTA models use self-supervised learning techniques, which have a large number of parameters (over 300M) [28]. However, on-device processing requires a smaller model, typically around 30M or less. Therefore, We do not apply tensor decomposition directly to the large model because the teacher model is too large to minimize within small parameters with sufficient rank. We train small model through a middle-size model to achieve the target size with a compression ratio near 0.3. As shown in the blue boxes in Fig.1, the training procedure has 5 steps.

1. A large teacher model is trained with ground truth labels. We use the E2E Conformer model with Hubert-based feature extractor[1] [28], which has achieved the best performance in prior study [29].
2. The training data is decoded using the trained teacher model to generate teacher labels.
3. The middle model is trained with both the distilled and ground truth labels. The middle model is used as the seed model to miniaturize the final small model using the decomposition, so its structural type is the same as the final small model.
4. Tensor decomposition and SVD techniques are applied to the middle model to reduce the parameters. We set the target size to 15M or 30M as in a previous study [7].
5. Finally, the decomposed model is finetuned with distillation to generate the small model.

[1] https://dl.fbaipublicfiles.com/hubert/hubert_large_ll60k.pt

Table 1: *Experimental results on STOP dataset within 15M parameters using Tucker decomposition.*

			Compression ratio encoder / decoder	Distillation	# Epochs	# Parameters	Valid/Test EM
	Deliberation SLU [7]					$<$ 15M	—— / 67.9
E2E	Conformer	Teacher				431M	68.8 / 69.4
		Scratch		✓	50	14M	34.7 / 34.8
		Middle		✓	50	47M	62.6 / 62.5
		Small	0.300 / 0.295	✓	50	15M	65.0 / 65.4
	E-Branchformer	Scratch		✓	50	15M	60.9 / 61.4
		Middle		✓	50	53M	68.6 / 69.0
		Small	0.250 / 0.300	✓	50	15M	69.6 / 70.1
		Small	0.250 / 0.300	✓	200	15M	**70.4 / 70.9**

Table 2: *Experimental results on STOP dataset within 30M parameters using Tucker decomposition.*

			Compression ratio encoder / decoder	Distillation	# Epochs	# Parameters	Valid/Test EM
	Deliberation SLU [7]					$<$ 30M	—— / **73.87**
E2E	Conformer	Teacher				431M	68.8 / 69.4
		Scratch		✓	50	29M	45.3 / 45.7
		Middle		✓	50	47M	62.6 / 62.5
		Small	0.650 / 0.650	✓	50	30M	65.4 / 65.7
	E-Branchformer	Scratch		✓	50	30M	66.4 / 66.8
		Middle		✓	50	53M	68.6 / 69.0
		Small	0.550 / 0.600	✓	50	30M	**71.0** / 71.4

The middle model and small model have Conformer or E-Branchformer encoders. Conformer and E-Branchformer include convolution layers that have 3rd or 4th-order tensor parameters. Thus, they can be efficiently compressed by tensor decomposition techniques such as Tucker decomposition.

3.2. Sequential distillation

Sequential distillation is used to downsize sequential models such as encoder-decoder models [30]. Frame-by-frame distillation is performed by the KL-divergence minimization criterion using softmax outputs with temperature hyperparameters. In contrast, sequential distillation uses the labels obtained by inference of the teacher model as target labels for student model training. It is also reported to be more effective when extended to N-best from 1-best [31]. However, in our setting, when extended to 5-best, the performance is slightly degraded. Instead, our system uses sequential distillation, which combines the 1-best obtained by the teacher model and ground truth labels.

3.3. Determination of rank from compression ratio

In our model, the Tucker decomposition is mainly used to reduce the parameters of the convolution layer. The reasons for adopting the Tucker decomposition are that it is expected to be able to represent the convolution weight tensor with fewer parameters than CP decomposition and that it contributes not only to parameter reduction but also to speed up [25]. The tucker decomposition can be inferred without reconstructing the original tensor by swapping the order of computation as in SVD. This technique is very compatible with on-device processing. On the other hand, Tensor-Train decomposition cannot accelerate inference because the original tensor must be reconstructed during inference.

The rank of the decompositions is determined by the specified compression ratio. We apply SVD to the linear layers and Tucker decomposition to the convolution layers in the models. The number of nodes in the middle of the SVD and the size of the core tensor of the Tucker decomposition are determined from the given compression ratio (eq. (2)). In the case of SVD, the number of middle nodes R can be calculated as follows:

$$R = \gamma_{\text{svd}} \frac{IJ}{I+J}. \tag{9}$$

On the other hand, since the core tensor shape cannot be uniquely determined from the given compression ratio (eq. (4)), the rank of the Tucker decomposition is calculated by iteratively halving all dimensions until the current ratio is under the given one as in Algorithm 1. In line 4, all dimensions are halved, and each time the compression ratio is recalculated in line 14. Lines 5–13 compensate so that each dimension is not less than 1.

For comparison, the Tucker decomposition is replaced by CP and Tensor-Train decomposition. CP decomposition, similar to SVD, can uniquely determine rank from compression ratio (eq. (6)). However, Tensor-Train decomposition cannot be uniquely determined from the given compression ratio (eq. (8)). Therefore, the dimension is iteratively reduced, as Tucker decomposition.

4. Experimental evaluation

4.1. Experimental settings

We evaluated our system on the STOP dataset [29]. STOP dataset consists of over 200,000 audio files from over 800 speakers and text and semantic parses. They are divided into train, valid, and test sets. Evaluation criteria were performed by exact match accuracy (EM), which is the percentage of perfect match of the label sequences.

The teacher model was based on the Conformer model. The model used a HuBERT-based feature extractor and convolution layers to reduce the input feature length. The encoder had 12 attention blocks, each with 512 dimensions with 8 heads. The

decoder had 6 attention blocks, and each block also had 512 dimensions with 8 heads. The number of parameters of the teacher model was 431M. The Conformer-based middle model had 47M parameters. Its encoder had 10 attention blocks, and each block had 384 dimensions with 6 attention heads. The decoder had 3 blocks with the same dimensions and attention heads. On the other hand, the E-Branchformer-based middle model had 53M parameters. Its encoder had 10 attention blocks, each with 384 dimensions with 6 heads. The decoder had 3 blocks with the same dimensions and heads.

For the training of the middle and small models, we used speed perturbation. The teacher and ground truth labels were both used with speed perturbation for the distillation. On the other hand, the validation data consisted of ground truth text, even in the sequential distillation. The input feature was log mel-filter bank having 80 bins. Furthermore, we used SpecAugment technique [32]. After that, we applied utterance-level mean normalization. The target semantic parse labels were divided using the byte-pair encoding (BPE) model with 500 tokens. Adam optimization with warmup scheduling was used in our training. Finally, we used an averaged model of the top 10 accuracy checkpoints.

4.2. Comparison of Conformer and E-Branchformer

Table 1 shows the experimental results of our system on the STOP dataset with 15M parameter limitation. In this experiment, we used tucker decomposition (Sec 2.2) to reduce the convolution parameters. For tensor decomposition, we set the compression ratio of the encoder and decoder as 0.3 and 0.295, respectively, in Conformer. In the case of the E-Branchformer, the compression ratios were set to 0.25 and 0.3 for the encoder and decoder. By using tensor decompositions from the middle Conformer, the small Conformer achieved 65.4 EM. Furthermore, the E-Branchformer encoder significantly improved the performance. Our system, E-Branchformer-based E2E SLU, achieved 70.1% EM for the test set with a parameter count of 15M, which was better than the previous study [7]. This result was a higher performance than the teacher model (69.4%). This was due to that the teacher model was based on a Conformer encoder and ground truth labels were also used for sequential distillation. Furthermore, under this condition, it appeared that the small model had not fully converged, so we continued training it for an additional 200 epochs. As a result, the accuracy was increased to 70.9%.

In addition we made comparisons with 30M parameters. Table 2 shows the experimental results of our system on the STOP dataset with 30M parameter limitation. In this limitation, we set the compression ratios of the encoder and decoder as 0.65 and 0.65 in Conformer. In the E-Branchformer, the encoder and decoder compression ratios were set as 0.55 and 0.6, respectively. The performance of the E-Branchformer was higher than Conformer but lower than the deliberation model [7]. The comparison with 15M indicates that the E2E model is more advantageous when constructing the smaller model. Our E2E model does not explicitly separate ASR and NLU, so unlike [7], we can reduce the parameters in a balanced manner so that SLU performance remains high.

4.3. Comparison of different compression ratios

Consequently, we investigated the performance changes with various compression ratios. Figure 2 shows the EM with varying compression ratio in E-Branchformer with Tucker decomposition. We used the same compression ratio for both encoder and decoder in this experiment. According to the figure, The EM drops sharply when the compression ratio falls under 0.3. Therefore, it is important to set an appropriate compression ratio when applying tensor decomposition.

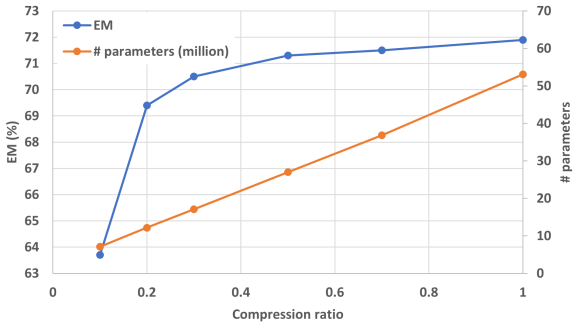

Figure 2: *EM of our system varying compression ratio with Tucker decomposition.*

Table 3: *Experimental results (token accuracy and EM) on STOP dataset within 15M changing decomposition type.*

	TAcc.	EM
Tucker decomposition (Sec. 2.2)	**91.8**	70.1
CP decomposition (Sec. 2.3)	34.8	0.1
Tensor-Train decomposition (Sec. 2.4)	90.7	**70.2**

4.4. Comparison of different tensor decompositions

We further examined the performance with different tensor decomposition techniques. In addition to the Tucker decomposition used in our system, we applied CP decomposition and Tensor-Train decomposition to the convolution layer. Table 3 shows the performance on the STOP dataset. In this experiment, the compression ratio was set to 0.25 and 0.3 for the encoder and decoder, respectively, targeting 15M. CP decomposition improved the training accuracy from the initial parameters, but it reached a saturation point at small epochs. This was because the CP decomposition decomposed all dimensions with the same rank, which led to an excessive loss of expressiveness. In contrast, the Tensor-Train decomposition performed significantly well, as did the Tucker decomposition. EM was slightly better for the Tensor-Train decomposition, but we observed a large difference in token accuracy.

5. Conclusion

In this paper, we describe our investigation of the minimization of the E-Branchformer for on-device E2E SLU. We applied sequential distillation and tensor decomposition techniques to the E-Branchformer. Compared to the Conformer-based model, E-Branchformer with Tucker decomposition achieved higher performance in both 15M and 30M limitations. In addition, it obtained better performance than the deliberation model in 15M limitation. The experiment with different compression ratios showed that compression around 0.3 points was efficient for E-Branchformer. Our comparison of the Tucker decomposition with the CP decomposition and Tensor-Train decomposition showed that the Tucker decomposition was a relatively efficient decomposition. Finally, our system, E-Branchformer-based decomposed E2E SLU model, achieved 70.9% EM with 15M parameter limitation on STOP data.

6. References

[1] Dmitriy Serdyuk, Yongqiang Wang, Christian Fuegen, Anuj Kumar, Baiyang Liu, and Yoshua Bengio, "Towards end-to-end spoken language understanding," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5754–5758.

[2] Martin Radfar, Athanasios Mouchtaris, and Siegfried Kunzmann, "End-to-end neural transformer based spoken language understanding," *Proc. Interspeech 2020*, pp. 866–870, 2020.

[3] Jingjing Dong, Jiayi Fu, Peng Zhou, Hao Li, and Xiaorui Wang, "Improving spoken language understanding with cross-modal contrastive learning," *Proc. Interspeech 2022*, pp. 2693–2697, 2022.

[4] Ye Wang, Baishun Ling, Yanmeng Wang, Junhao Xue, Shaojun Wang, and Jing Xiao, "Adversarial knowledge distillation for robust spoken language understanding," *Proc. Interspeech 2022*, pp. 2708–2712, 2022.

[5] Siddhant Arora, Siddharth Dalmia, Xuankai Chang, Brian Yan, Alan Black, and Shinji Watanabe, "Two-pass low latency end-to-end spoken language understanding," *Proc. Interspeech 2022*, pp. 3478–3482, 2022.

[6] Martin Radfar, Athanasios Mouchtaris, Siegfried Kunzmann, and Ariya Rastrow, "FANS: Fusing ASR and NLU for on-device SLU," in *Interspeech 2021*, 2021.

[7] Duc Le, Akshat Shrivastava, Paden D. Tomasello, Suyoun Kim, Aleksandr Livshits, Ozlem Kalinli, and Michael L. Seltzer, "Deliberation model for on-device spoken language understanding," in *Interspeech 2022*, 2022, pp. 3468–3472.

[8] Alaa Saade, Joseph Dureau, David Leroy, Francesco Caltagirone, Alice Coucke, Adrien Ball, Clément Doumouro, Thibaut Lavril, Alexandre Caulier, Théodore Bluche, et al., "Spoken language understanding on the edge," in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*. IEEE, 2019, pp. 57–61.

[9] Akshit Tyagi, Varun Sharma, Rahul Gupta, Lynn Samson, Nan Zhuang, Zihang Wang, and Bill Campbell, "Fast intent classification for spoken language understanding systems," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8119–8123.

[10] Thierry Desot, François Portet, and Michel Vacher, "End-to-end spoken language understanding: Performance analyses of a voice command task in a low resource setting," *Computer Speech & Language*, vol. 75, pp. 101369, 2022.

[11] Akshat Gupta, "On building spoken language understanding systems for low resourced languages," in *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Seattle, Washington, July 2022, pp. 1–11, Association for Computational Linguistics.

[12] Pu Wang et al., "Bottleneck low-rank transformers for low-resource spoken language understanding," *Proceedings Interspeech 2022*, 2022.

[13] Anderson R Avila, Khalil Bibi, Rui Heng Yang, Xinlin Li, Chao Xing, and Xiao Chen, "Low-bit shift network for end-to-end spoken language understanding," *Proc. Interspeech 2022*, pp. 2698–2702, 2022.

[14] Yingying Gao, Junlan Feng, Chao Deng, and Shilei Zhang, "Meta auxiliary learning for low-resource spoken language understanding," *Proc. Interspeech 2022*, pp. 2703–2707, 2022.

[15] Marco Dinarelli, Marco Naguib, and François Portet, "Toward low-cost end-to-end spoken language understanding," *Proc. Interspeech 2022*, pp. 2728–2732, 2022.

[16] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al., "Conformer: Convolution-augmented transformer for speech recognition," *Proc. Interspeech 2020*, pp. 5036–5040, 2020.

[17] Yifan Peng, Siddharth Dalmia, Ian Lane, and Shinji Watanabe, "Branchformer: Parallel mlp-attention architectures to capture local and global context for speech recognition and understanding," in *International Conference on Machine Learning*. PMLR, 2022, pp. 17627–17643.

[18] Kwangyoun Kim, Felix Wu, Yifan Peng, Jing Pan, Prashant Sridhar, Kyu J Han, and Shinji Watanabe, "E-Branchformer: Branchformer with enhanced merging for speech recognition," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 84–91.

[19] Jeffrey Josanne Michael, Nagendra Kumar Goel, Jonas Robertson, Shravan Mishra, et al., "Comparison of svd and factorized tdnn approaches for speech to text," *arXiv preprint arXiv:2110.07027*, 2021.

[20] Yuekai Zhang, Sining Sun, and Long Ma, "Tiny transducer: A highly-efficient speech recognition model on edge devices," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6024–6028.

[21] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," *Advances in neural information processing systems*, vol. 27, 2014.

[22] Siddhant Arora, Siddharth Dalmia, Pavel Denisov, Xuankai Chang, Yushi Ueda, Yifan Peng, Yuekai Zhang, Sujay Kumar, Karthik Ganesan, Brian Yan, et al., "Espnet-slu: Advancing spoken language understanding through espnet," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7167–7171.

[23] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, "Compressing recurrent neural network with tensor train," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 4451–4458.

[24] Ledyard R Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[25] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *arXiv preprint arXiv:1511.06530*, 2015.

[26] Richard A Harshman et al., "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[27] Ivan V Oseledets, "Tensor-Train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[28] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed, "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[29] Paden Tomasello, Akshat Shrivastava, Daniel Lazar, Po-Chun Hsu, Duc Le, Adithya Sagar, Ali Elkahky, Jade Copet, Wei-Ning Hsu, Yossi Adi, et al., "STOP: A dataset for spoken task oriented semantic parsing," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 991–998.

[30] Yoon Kim and Alexander M Rush, "Sequence-level knowledge distillation," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1317–1327.

[31] Raden Mu'az Mun'im, Nakamasa Inoue, and Koichi Shinoda, "Sequence-level knowledge distillation for model compression of attention-based sequence-to-sequence speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6151–6155.

[32] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," *Proc. Interspeech 2019*, pp. 2613–2617, 2019.