PruneSLU: Efficient On-device Spoken Language Understanding through Vocabulary and Structural Pruning

Anonymous submission to Interspeech 2025

Abstract

Recent advancements in Spoken Language Understanding (SLU) have been driven by pre-trained speech processing models. However, deploying these models on resource-constrained devices remains challenging due to their large parameter sizes. This paper presents PruneSLU, a new method for compressing pre-trained SLU models while maintaining performance. Our approach combines vocabulary pruning and structural layer-wise pruning to reduce model size while preserving essential knowledge. After pruning, the model undergoes knowledge refinement using integration distillation and contrastive learning. Experiments on the STOP and SLURP datasets demonstrate that PruneSLU compresses a 39M model to 15M while retaining 98% of its original performance, outperforming previous compression techniques.

Index Terms: Spoken Language Understanding, Model Pruning, On-device Processing

1. Introduction

Spoken Language Understanding (SLU) plays a crucial role in conversational AI, which is fundamental to applications such as virtual assistants [1, 2]. Despite advancements, deploying SLU models on devices like smartphones remains challenging due to constraints in computational resources, memory, and power. Therefore, creating SLU models that are efficient enough to run directly on such devices is essential.

Research in on-device SLU has primarily focused on addressing these constraints through two main approaches. The first approach involves designing a compact SLU model from scratch [3, 4]. The second strategy builds on existing pre-trained models, reducing their size through model compression techniques [5]. This latter approach, known as model compression or distillation, has shown significant promise. It uses a smaller student model to mimic the behavior of a larger, pre-trained teacher model, thereby preserving the critical knowledge embedded within the teacher model.

In this paper, we introduce a new method to compress pre-trained models for on-device SLU tasks (Fig. 1). Unlike previous methods that often require creating a new student model [5], our approach involves pruning the teacher model by removing less effective weights. Our pruning strategy is based on two key methods: vocabulary pruning and structural pruning. Vocabulary pruning reduces the model's tokens to the most task-based tokens, cutting down memory usage and computational demands. Structural pruning involves selectively removing less critical layers, thus simplifying the architecture. We explore five strategies for layer pruning: top-layer, alternative-layer, bottom-layer, magnitude-based, and loss-based. By applying these techniques, we aim to create lightweight yet high-performing SLU systems. To mitigate the loss of knowledge due to pruning, we employ a model retraining process, transferring knowledge from the teacher to the pruned student model using entropy loss, distillation loss, and contrastive loss techniques. Our proposed framework, PruneSLU, surpasses existing methods, achieving Exact Match (EM) scores exceeding the state-of-the-art (SOTA) under the constraints of 15M and 30M parameters on the STOP dataset [6]. It also demonstrates significant model compression—up to 260%—with only a 5% reduction in performance on the SLURP dataset [7]. In summary, our contributions are threefold:

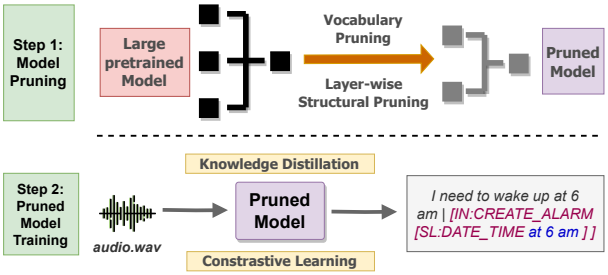

Figure 1: *An overview of our framework.*

- **(I)** Proposing a new pruning method tailored for on-device SLU applications, integrating vocabulary pruning and structural pruning, along with a knowledge refinement approach using distillation and contrastive learning.
- **(II)** Demonstrating the effectiveness of the proposed method with extensive experiments applications.
- **(III)** Making our code and models publicly available at `https://anonymous.4open.science/r/PruneSLU-E7FF`.

2. Methodology

End-to-end (E2E) models based on pre-trained speech processing architectures like Whisper [8] have demonstrated strong performance in spoken language understanding (SLU) [9, 10]. These models typically comprise n encoder layers, m decoder layers, a hidden size of dim, and a vocabulary V. However, their large size presents challenges for on-device deployment.

To address this issue, we propose a new pruning method that effectively reduces model parameters while maintaining performance, making it suitable for on-device deployment (typically under 30M parameters) [5]. Given a pre-trained model with P parameters, partitioned into non-vocabulary (P_{nv}) and vocabulary (P_v) components, we apply two pruning techniques: (1) *vocabulary pruning*, which retains only essential tokens to

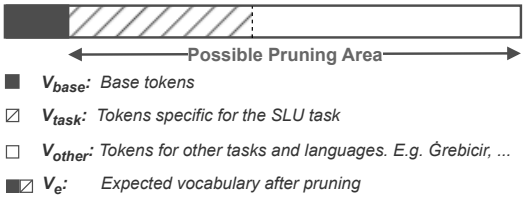

Figure 2: *Illustration of vocabulary division: General base tokens (V_{base}), SLU task-specific tokens (V_{task}), and tokens for other tasks or languages (V_{other}): $V_e = V_{base} \cup V_{task}$.*

reduce P_v, and (2) *layer-wise structural pruning*, which reduces P_{nv} by removing encoder and decoder layers. The resulting pruned model, with P_e parameters, consists of n_e encoder layers, m_e decoder layers, and a compact vocabulary V_e, where $n_e \leq n$, $m_e \leq m$, and $V_e \subset V$.

2.1. Vocabulary Pruning

Multilingual pre-trained speech processing models, such as Whisper [8], can be extremely large, with some models containing over a billion (B) parameters (e.g., Whisper-large contains 1.5B parameters). A significant portion of these parameters is typically concentrated in the embedding matrix, which scales with the vocabulary size. For instance, the Whisper-tiny model has 39 million (M) parameters, with 51% of its size attributed to the vocabulary ($P_v = dim \times |V| = 384 \times 51,865 \approx 20M$). However, many tokens may be irrelevant to specific downstream tasks like SLU. Pruning these non-essential tokens can significantly reduce model size while maintaining performance in the task.

In this paper, we propose a method for pruning a given vocabulary V to a more compact vocabulary V_e. Our approach begins with establishing a base vocabulary V_{base}, consisting of the most frequent tokens from the original tokenizer, such as the top 2,000 most frequent tokens. These base tokens ensure the model retains its basic functionality by being able to tokenize any input sequence effectively [11]. If $|V_{base}|$ is still smaller than $|V_e|$, we then incorporate additional tokens from V, focusing on those most relevant to the SLU task (Fig. 2). This additional set, V_{task}, includes tokens that are crucial for the downstream SLU task. The process involves: (i) tokenizing the SLU training data; (ii) ranking the tokens by frequency; (iii) excluding tokens already present in V_{base}; and (iv) iteratively adding tokens and their associated sub-tokens to V_e until the desired vocabulary size is reached. This method ensures a tailored vocabulary balancing compactness with the capability to handle task-specific language nuances.

2.2. Layer-wise Structural Pruning

In addition to vocabulary pruning, we apply layer-wise structural pruning to further reduce the model's size. Prior research shows that pruning the encoder can significantly compromise accuracy, as the encoder is crucial for transforming input audio into a rich, contextual representation [12], which is essential for accurate comprehension and processing. Therefore, our approach prioritizes pruning the decoder layers before considering any modifications to the encoder. Specifically, we aim to reduce the number of decoder layers to a minimum (one layer) before pruning the encoder, as the decoder can often perform effectively with fewer layers in domain-specific adaptation [13].

For the decoder, we define the layer-wise pruning task as follows: Consider a pre-trained speech processing model's decoder (L) with d layers, where L_0 is the input layer and L_{d-1} is the last layer. Our objective is to develop an algorithm that identifies a subset $R_k \subset L$ with $|R_k| = k$, ensuring that the performance of the pruned model with R_k is only marginally reduced compared to the full model. To achieve this, we propose five strategies for selecting R_k.

(I) Top-layer pruning: This strategy assumes lower layers, capturing general features, are critical for overall performance, while specialized top layers, which focus on specific objectives, can be pruned with minimal accuracy impact [14]. Thus, we remove the top $d - k$ layers from the model.

(II) Alternative-layer pruning: Based on the assumption that neighboring layers may contain redundant or overlapping information [15], this approach prunes every alternate layer. The pruned model retains layers L_0, L_2, ..., L_{2k-2}.

(III) Bottom-layer pruning: Although lower layers are crucial for basic feature extraction and understanding, we explore this strategy for the completeness of our experiments. Here, we remove the lowest $d - k$ layers, starting from layer L_0.

(IV) Magnitude-based pruning: Building on successful unstructured pruning techniques, which suggest that smaller magnitude weights hold less information [16, 17], we extend this to layer-wise pruning. We sum the absolute magnitudes of all weights in a layer: $S_i = \sum_{j=1}^{n_i} |w_{ij}|$, where w_{ij} is the j-th weight in layer L_i, and n_i is the total number of weights in layer L_i. The $d - k$ layers with the smallest total magnitudes S_i are considered least important and are pruned.

(V) Loss-based pruning: Unlike data-independent methods, loss-based pruning uses a validation dataset to guide pruning. We iteratively omit one layer at a time, computing the evaluation loss. The $d - k$ layers that least impact the evaluation loss are pruned, offering a more nuanced understanding of each layer's importance to the SLU task.

2.3. Pruning Constraint

To extend the pruning technique across various model parameter sizes, we define the expected parameters as P_e. The model's parameters can fall into two categories: vocabulary-related, in the embedding matrix, and non-vocabulary, primarily in the encoder/decoder layers. We treat the encoder and decoder layers collectively, as our pruning strategy first focuses on the decoder, then the encoder. Thus, the total number of expected parameters can be expressed as:

$$P_e \approx dim \times |V_e| + P_{\text{layer}} \times d_e \tag{1}$$

Here, *dim* represents the model's hidden dimension, P_{layer} denotes the number of parameters per layer, d_e is the number of pruned layers, and $|V_e|$ indicates the pruned vocabulary size. The pruning process involves determining $|V_e|$ and d_e given P_e (expected parameter size), with *dim* and P_{layer} defined by the model's configuration. Since vocabulary and layer pruning are interdependent, each affecting performance differently, equation (1) can be addressed by proposing a pruning sequence—vocabulary first, then layers, or the reverse. For example, if vocabulary pruning is done first, followed by layer pruning, d_e can be calculated as:

$$d_e = \begin{cases} d & \text{if } \text{dim} \times |V_{\text{base}}| + P_{\text{layer}} \times d \leq P_e \\ \lceil \frac{P_e - \text{dim} \times |V_{\text{base}}|}{P_{\text{layer}}} \rceil & \text{otherwise} \end{cases} \tag{2}$$

Alternatively, if the sequence involves pruning the layers first, followed by vocabulary pruning, d_e can be calculated as:

$$d_e = \begin{cases} 1 & \text{if } \dim \times |V| + P_{\text{layer}} \geq P_e \\ \lceil \frac{P_e - \dim \times |V|}{P_{\text{layer}}} \rceil & \text{otherwise} \end{cases} \quad (3)$$

Once d_e is determined, the value of $|V_e|$ can then be calculated as: $|V_e| = \lfloor (P_e - P_{\text{layer}} \times d_e) \,/\, \dim \rfloor$. In the experiments (Section 3.2), we will analyze different pruning sequences and methodologies, comparing their effectiveness. This analysis will identify the optimal strategy for balancing model size reduction with performance retention, guiding pruning choices in practical applications.

2.4. Pruned Model Retraining

To restore the performance of pruned models on the SLU task, we implement a retraining strategy that includes both standard training with gold label data and knowledge distillation. In distillation, a smaller, student model learns to replicate the behavior of a larger, more complex teacher model by imitating its output distributions [18] and hidden states across intermediate layers via contrastive loss [19, 20]. However, distillation can be challenging when there is a significant size gap between models or when they utilize different vocabularies [5]. To address these challenges, rather than using the base pre-trained speech processing model as the teacher, we fine-tune the vocabulary-pruned model to serve as the teacher, with the fully pruned model acting as the student.

Our approach to retraining begins with the standard cross-entropy (CE) loss applied to the ground truth labels:

$$L_{CLM} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \quad (4)$$

where N is the number of samples, y_i and $\hat{y}_i$ are the true label and the predicted probability for the i-th sample, respectively.

Previous research has demonstrated that combining knowledge distillation loss with cross-entropy loss can enhance model performance [18]. The distillation objective is typically defined as the cross-entropy loss between the output distributions of the teacher (p_t) and student (p_s) models:

$$L_{distill} = D_{KL}(p_s || p_t) \quad (5)$$

Moreover, recent studies suggest that incorporating knowledge from the intermediate layers of the teacher model can further boost the student's performance [21, 22]. While previous methods often align intermediate representations using mean squared error loss, we hypothesize that the student model should not only closely mirror the teacher's hidden states but also generate representations that distinctly differentiate each sample from others. To achieve this, we introduce contrastive learning [19, 20] into our training process. The aim is to align the representations of positive pairs in latent space while increasing the distance between negative pairs. The contrastive loss function measures the similarity between the hidden states $h_{teacher}$ and $h_{student}$:

$$\begin{aligned} h_{teacher} &= \mathcal{H}_t^{(q)} \\ h_{student} &= \mathcal{H}_s^{(k)} \end{aligned} \quad (6)$$

where $\mathcal{H}_t^{(q)}$ and $\mathcal{H}_s^{(k)}$ represent the outputs of the teacher and student layers q and k after LayerNorm. Here, the q-th layer of the student model mirrors the k-th layer of the teacher model post-pruning. This information is used to update the model's parameters, thereby enabling the pruned model to produce more consistent representations with the full model. Given a training batch B, the positive pair at the i^{th} position is $(h_{teacher}^{(i)}, h_{student}^{(i)})$, while the negative pairs consist of other samples in the batch. The training objective for contrastive learning is given by:

$$L_{contras} = -\log \frac{\exp(\text{sim}(h_{teacher}^{(i)}, h_{student}^{(i)})/\tau)}{\sum_{j \in B} \exp(\text{sim}(h_{teacher}^{(i)}, h_{student}^{(j)})/\tau)} \quad (7)$$

where sim is the cosine similarity function, and τ is a temperature hyperparameter [19]. B represents the samples in the mini-batch. Finally, to optimize the model's learning process, we combine all loss functions into a final objective function:

$$L_{final} = L_{CLM} + \alpha * L_{distill} + \beta * L_{contras} \quad (8)$$

where α and β are hyperparameters that balance the contributions of the loss components.

3. Experiment

Dataset and Evaluation Metric. To evaluate our proposed method, we conducted experiments on the well-known STOP dataset [6], which encompasses a diverse range of 82 intents and 84 slots. We utilized the full-resource version, comprising $120k$, $33k$, and $75k$ training, development, and testing samples, respectively. Following [6], our primary evaluation metric was the Exact Match (EM) score. Additionally, we employed the EM-Tree metric, which measures the accuracy of generating correct parsed trees ignoring the accuracy of the span texts. To further demonstrate our method's generalizability, we analyzed its performance on the SLURP dataset [7]. SLURP consists of 50,628 audio files, with 8,690 samples allocated for development and 13,078 for testing. The evaluation metrics for SLURP included intent accuracy (IC) and SLU F1 (SF) score, as outlined in the original paper [7].

Experimental Settings. For our experiments, we used the pre-trained Whisper model [8] as the backbone, specifically Whisper-tiny with 39M parameters and Whisper-base with 74M parameters. We fine-tuned the models on the validation set to select the best hyperparameters, followed by test set evaluation for final results. Specifically, we fine-tuned on the STOP dataset for {1, **3**, 5, 10} epochs[1], using learning rates {1e-05, 2e-05, 3e-05, **4e-05**, 5e-05}, 5000 bootstrap steps, and a batch size of 16. For hyperparameters, we set τ to 1.0 and selected α, β from {**0.1**, 0.2, 0.5, 1.0}. We chose the optimal base tokens vocabulary size $|V_{base}|$ from {500, 1000, **2000**, 3000, 4000, 5000}. Training time for the 15M model with an A100 GPU is approximately 1 hour. For the SLURP dataset, we used the same settings as the STOP dataset.

Baselines. For the baselines, we used the tensor decomposition method [5] and the deliberation-based approach [4] from previous works. We compared our method against these baselines in two settings: 15M and 30M parameters.

3.1. Main Results

STOP. Table 1 compares our method with baselines at 30M and 15M parameters. In both cases, our method outperforms the best previous approach [5], with gains of 1.69 EM in the 30M and 1.41 EM in the 15M setting. We compress the 39M model

[1]The **bold** values indicate best performance on the validation set.

Method	Pre-trained Model	Num. Param	STOP	
			EM	EM-Tree
WhiSLU-large [9]	Whisper-large	1550 M	76.68	86.37
Tensor decomposition [5]	Hubert-base	30.0 M	71.40	-
Tensor decomposition [5]	Hubert-base	15.0 M	70.90	-
Deliberation SLU [4]	None	15.0 M	67.90	-
PruneSLU-full (ours)	whisper-tiny	39.0 M	73.15	84.08
PruneSLU-30M (ours)	whisper-tiny	30.0 M	**73.09**	**83.99**
PruneSLU-15M (ours)	whisper-tiny	15.0 M	**72.31**	**83.55**

Table 1: *Main results of our method on STOP test set.*

to 15M while retaining 98% of its performance. Compared to the SOTA WhiSLU-large method [9] of the STOP dataset, our 15M model achieves comparable results with 100 times fewer parameters, underscoring the effectiveness of our approach.

Method	Pre-trained Model	Num. Param	SLURP	
			IC	SF
SpeechBrain Direct [23]	HuBERT-base	95 M	85.34	74.62
PruneSLU-full (ours)	whisper-tiny	39.0 M	76.53	73.23
PruneSLU-30M (ours)	whisper-tiny	30.0 M	76.26	72.85
PruneSLU-15M (ours)	whisper-tiny	15.0 M	76.22	71.42

Table 2: *Main results of our method on SLURP test set.*

SLURP. Table 2 presents our method's results on the SLURP dataset. The 15M model achieves 95% of the full model's performance using the SLU-F1 metric, showcasing our method's strong generalization across datasets.

3.2. Method Analysis

To further assess our method, we conducted a series of in-depth analyses using the STOP dataset.

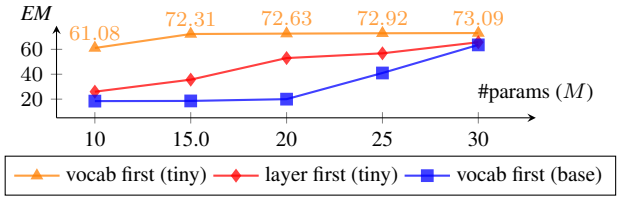

Figure 3: *Performance across different compression ratios with various pruning sequences and base models.*

Impact of Pruning Sequence. As shown in Fig. 3, we compared two pruning strategies on the Whisper-tiny base model: pruning vocabulary first versus pruning layers first. The results indicate that pruning vocabulary first is more effective, as it removes redundant tokens while preserving performance.

Effect of Target Sizes and Base Models. Here, we compare the performance of models compressed using the vocabulary-first pruning method across different parameter sizes (Fig. 3). We observe that the performance remains stable between 15M and 30M parameters but experiences a noticeable decline when further reduced to 10M. This drop is likely due to the necessity of pruning encoder layers at this scale, which aligns with previous research showing that pruning the encoder can significantly impair model performance [12]. Additionally, our findings indicate that compressing from Whisper-tiny (39M parameters) is more effective than from Whisper-base (74M parameters), suggesting that a base model closer in size to the target model facilitates better results.

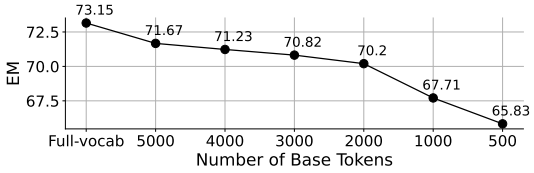

Figure 4: *Impact of number base tokens.*

Impact of Base Tokens. We also analyzed the effect of varying the number of base tokens from 500 to 5000 using Whisper-tiny. As depicted in Fig. 4, there is a noticeable performance drop when reducing from 2000 to 1000 base tokens, with only minimal parameter reduction. Thus, we adopt 2000 base tokens as the standard setting for our method.

Layer Pruning Method Analysis. We explored different layer-wise pruning methods using the Whisper-base model, with results shown in Table 3. Our analysis reveals that loss-based pruning performs best, likely because it leverages domain-specific evaluation data to guide the pruning process, thereby selecting the most appropriate layers for removal.

Pruning Method	EM
Top-layer Pruning	66.72
Bottom-layer Pruning	70.46
Alternate-layer Pruning	71.46
Magnitude-based Pruning	44.46
Loss-based Pruning	**71.93**

Table 3: *Impact of layer-wise pruning strategies ($|R_k| = 2$).*

Ablation Study on Loss Components. Finally, we conducted an ablation study on the loss components used in our method, as summarized in Table 4. Each component significantly contributes to the overall performance, and removing any one of them leads to noticeable degradation, highlighting the importance of each element in our final loss function.

Method	EM	Δ
PruneSLU	72.18	-
- without L_{CLM}	71.28	-0.90
- without L_{KL}	66.87	-5.31
- without $L_{contras}$	71.44	-0.74

Table 4: *Impact of each component in the final loss.*

4. Conclusion

We introduced PruneSLU, a framework that optimizes pre-trained models for on-device SLU through vocabulary and layer-wise structural pruning. PruneSLU reduced model size while maintaining high performance, surpassing SOTA methods on the STOP dataset with 1.69 EM in 30M setting and 1.41 EM in 15M setting. It preserved 95% of the full model's performance on SLURP, demonstrating generalizability. Extensive experiments validated the effectiveness of our pruning strategies and the benefit of combining knowledge distillation with contrastive learning to minimize performance loss.

5. References

[1] J. R. Bellegarda, "Spoken language understanding for natural interaction: The siri experience," *Natural Interaction with Robots, Knowbots and Smartphones: Putting Spoken Dialog Systems into Practice*, pp. 3–14, 2013.

[2] S. Rongali, B. Liu, L. Cai, K. Arkoudas, C. Su, and W. Hamza, "Exploring transfer learning for end-to-end spoken language understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 15, 2021, pp. 13 754–13 761.

[3] M. Radfar, A. Mouchtaris, S. Kunzmann, and A. Rastrow, "FANS: Fusing ASR and NLU for On-Device SLU," in *Proc. Interspeech 2021*, 2021, pp. 1224–1228.

[4] D. Le, A. Shrivastava, P. D. Tomasello, S. Kim, A. Livshits, O. Kalinli, and M. Seltzer, "Deliberation Model for On-Device Spoken Language Understanding," in *Proc. Interspeech 2022*, 2022, pp. 3468–3472.

[5] Y. Kashiwagi, S. Arora, H. Futami, J. Huynh *et al.*, "Tensor decomposition for minimization of E2E SLU model toward on-device processing," in *Proc. INTERSPEECH 2023*, 2023, pp. 710–714.

[6] P. Tomasello, A. Shrivastava, D. Lazar, P.-C. Hsu, D. Le *et al.*, "Stop: A dataset for spoken task oriented semantic parsing," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 991–998.

[7] E. Bastianelli, A. Vanzo, P. Swietojanski, and V. Rieser, "SLURP: A spoken language understanding resource package," in *Proceedings of EMNLP 2020*, Online, 2020, pp. 7252–7262.

[8] A. Radford, J. W. Kim, T. Xu *et al.*, "Robust speech recognition via large-scale weak supervision," in *International conference on machine learning*. PMLR, 2023, pp. 28 492–28 518.

[9] M. Wang, Y. Li, J. Guo, X. Qiao *et al.*, "Whislu: End-to-end spoken language understanding with whisper," in *Proc. Interspeech*, vol. 2023, 2023, pp. 770–774.

[10] H. Yang, M. Zhang, and D. Wei, "Irag: Iterative retrieval augmented generation for slu," in *2024 20th IEEE International Colloquium on Signal Processing & Its Applications (CSPA)*. IEEE, 2024, pp. 30–34.

[11] J. O. Alabi, D. I. Adelani, M. Mosbach, and D. Klakow, "Adapting pre-trained language models to african languages via multilingual adaptive fine-tuning," in *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea, 2022, pp. 4336–4349. [Online]. Available: https://aclanthology.org/2022.coling-1.382

[12] S. Gandhi, P. von Platen, and A. M. Rush, "Distil-whisper: Robust knowledge distillation via large-scale pseudo labelling," *arXiv preprint arXiv:2311.00430*, 2023.

[13] S. Shleifer and A. M. Rush, "Pre-trained summarization distillation," *arXiv preprint arXiv:2010.13002*, 2020.

[14] E. Voita, R. Sennrich, and I. Titov, "The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives," in *Proc. EMNLP-IJCNLP 2019*, Hong Kong, China, 2019, pp. 4396–4406.

[15] H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov, "Poor man's bert: Smaller and faster transformer models," *preprint arXiv:2004.03844*, vol. 2, no. 2, 2020.

[16] M. Zullich, E. Medvet, F. A. Pellegrino, and A. Ansuini, "Speeding-up pruning for artificial neural networks: introducing accelerated iterative magnitude pruning," in *2020 25th International Conference on Pattern Recognition*. IEEE, 2021, pp. 3868–3875.

[17] A. Bragagnolo, E. Tartaglione, A. Fiandrotti, and M. Grangetto, "On the role of structured pruning for neural network compression," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3527–3531.

[18] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[19] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6894–6910.

[20] T. Do, P. Nguyen, and M. Nguyen, "StructSP: Efficient fine-tuning of task-oriented dialog system by using structure-aware boosting and grammar constraints," in *Findings of the Association for Computational Linguistics: ACL 2023*, Toronto, Canada, 2023, pp. 10 206–10 220.

[21] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for BERT model compression," in *EMNLP-IJCNLP 2019*, Hong Kong, China, 2019, pp. 4323–4332.

[22] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for natural language understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 4163–4174.

[23] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe *et al.*, "Speechbrain: A general-purpose speech toolkit," *preprint arXiv:2106.04624*, 2021.