SepPrune: Structured Pruning for Efficient Deep Speech Separation

Yuqi Li[1*], Kai Li[2*], Xin Yin[3], Zhifei Yang[4], Zeyu Dong[5], Zhengtao Yao[6], Haoyan Xu[6], Yingli Tian[1], Yao Lu[7,8 ✉†]

[1]The City College of New York, CUNY
[2]Tsinghua University
[3]Zhejiang University
[4]Peking University
[5]Boston University
[6]University of Southern California
[7]CFAR, Agency for Science, Technology and Research (A*STAR)
[8]Institute of Cyberspace Security, Zhejiang University of Technology
yuqili010602@gmail.com, yaolu.zjut@gmail.com

Abstract

Although deep learning has substantially advanced speech separation in recent years, most existing studies continue to prioritize separation quality while overlooking computational efficiency, an essential factor for low-latency speech processing in real-time applications. In this paper, we propose *SepPrune*, the first structured pruning framework specifically designed to compress deep speech separation models and reduce their computational cost. *SepPrune* begins by analyzing the computational structure of a given model to identify layers with the highest computational burden. It then introduces a differentiable masking strategy to enable gradient-driven channel selection. Based on the learned masks, *SepPrune* prunes redundant channels and fine-tunes the remaining parameters to recover performance. Extensive experiments demonstrate that this learnable pruning paradigm yields substantial advantages for channel pruning in speech separation models, outperforming existing methods. Notably, a model pruned with *SepPrune* can recover 85% of the performance of a pre-trained model (trained over hundreds of epochs) with only one epoch of fine-tuning, and achieves convergence 36× faster than training from scratch.

Code — https://github.com/itsnotacie/SepPrune

Introduction

Speech separation, the task of isolating individual speakers from a mixed audio signal, has made remarkable progress in recent years (Jiang, Han, and Mesgarani 2025; Li, Chen et al. 2024; Xu, Li et al. 2024; Xiao, Zhang et al. 2025). However, most speech separation methods have focused primarily on improving model performance. While these "performance-first" approaches are effective in controlled, laboratory settings, they fall short in real-world applications (Dai, Li et al. 2025; Zhang, Huang et al. 2025; Wu et al. 2025; Ma, Huang, and Lin 2025; Liu et al. 2025) that demand low latency and constrained resource consumption, such as real-time voice communication (Maryn, Ysenbaert et al. 2017; Ke, Gong, and Du 2025), embedded auditory systems (Briere, Valin et al. 2008), etc.

To bridge this efficiency gap, recent efforts (Li, Yang, and Hu 2022; Zhou, Li et al. 2024; Zhou, Zhao et al. 2025; Zhang, Gao et al. 2025; Zhou, Yuan et al. 2025; You and Liu 2024; Lan and Tian 2025) have attempted to develop lightweight models through manual architectural design. However, such handcrafted models suffer from two fundamental limitations. First, they depend heavily on expert-driven architectural modifications and require substantial domain-specific knowledge. Second, and more importantly, these manual modifications are typically tailored to specific architectures, limiting their generalizability to other models. In light of the dual dilemma faced by manually designing architectures, this paper explores an alternative, non-invasive optimization strategy: *model pruning*.

Although model pruning has been shown to be effective in compressing vision and language models (Lu, Cheng et al. 2024; Zhou, Yuan et al. 2024; Liu, Meng et al. 2025), striking a balance between inference speed, memory usage, and accuracy, to the best of our knowledge, no pruning algorithm currently exists for end-to-end speech separation models. Unlike traditional vision (He, Zhang et al. 2016; Liu, Lin et al. 2021; Simonyan and Zisserman 2014; Li, Yang et al. 2025; Meng, Luo, and Liu 2025; Zhang et al. 2025) or language (Zeng, Wang et al. 2025; Xiao et al. 2024; Lou et al. 2025; Liu and Xiao 2025) models, speech separation models typically consist of three heterogeneous components: an audio encoder, a deep separation network, and an audio decoder. The computational complexity across these components is highly imbalanced. Consequently, indiscriminate pruning may damage already lightweight layers, leading to a collapse in performance.

To address these challenges, we propose *SepPrune*, the first structured pruning framework specifically designed for speech separation models. *SepPrune* consists of three stages.

[*]These authors contributed equally.
[†]Corresponding author.

First, it performs a computational structural analysis on existing speech separation models to identify the layers that contribute most significantly to the overall computation. Next, it introduces a differentiable pruning mechanism using Gumbel-Softmax and a modified Straight-Through Estimator to build a set of differentiable channel binary masks to learn which channels should be kept. Finally, *SepPrune* keeps the more important channels while removing the less important channels based on the binary masks, and fine-tunes the pruned model to recover the performance. Experiments show that *SepPrune* not only significantly reduces the number of parameters and FLOPs of models, but also outperforms the previous state-of-the-art channel pruning methods (Gao, Zhang et al. 2024; Lin, Ji et al. 2020b) on the three benchmark datasets of LRS2-2Mix (Li, Yang, and Hu 2022), Libri2Mix (Cosentino, Pariente et al. 2020) and EchoSet (Xu, Li et al. 2024). More notably, the pruned model obtained by *SepPrune* can recover $86\%+$ of the performance of the original model trained for 493 epochs with only 1 epoch of fine-tuning, converging $36\times$ faster than training from scratch.

In summary, our main contributions are as follows:

- We introduce *SepPrune*, the first pruning framework tailored specifically for deep speech separation models. *SepPrune* performs structural computational analysis on the target model to determine the layers with the highest computational cost. Furthermore, *SepPrune* introduces binary differentiable channel masks to select an optimal substructure. Based on the obtained masks, *SepPrune* performs channel pruning and fine-tunes the remaining weights to recover performance.
- Extensive experiments demonstrate that *SepPrune* outperforms existing channel pruning methods on three benchmark datasets (Libri2Mix, LRS2-2Mix and EchoSet) and various backbones (A-FRCNN-12, TDANet, SuDoRM-RF1.0x). It significantly reduces the complexity of the model while only causing minimal performance loss. Further experiments demonstrate its fast convergence and practical speedup effect.

Related Works

Model Pruning. Model pruning is a widely used technique to compress pre-trained models by eliminating redundant parts, which can be roughly divided into three categories: weight pruning (Bai, Wang et al. 2022; Chen, Xiang et al. 2023; Hoang and Liu 2023; Liu, Chen et al. 2022; Sun, Liu et al. 2023; Wang, Li, and Sun 2023; Wang, Qin et al. 2021; Yang, Zhen et al. 2025; Yao, Li, and Xiao 2024), channel pruning (Guo, Wang et al. 2020; He, Zhang, and Sun 2017; Ling, Wang, and Liu 2024; Li, Kadav et al. 2016; Liu, Mu et al. 2019; Ma, Fang, and Wang 2023; Zhuang, Tan et al. 2018) and layer pruning (Chen and Zhao 2018; Li, Lu et al. 2024; Lu, Yang et al. 2022; Lu, Zhu et al. 2024; Lu, Cheng et al. 2024; Tang, Lu, and Xuan 2023; Wu, Zhu et al. 2023). Specifically, in weight pruning, the unimportant weights are set to zero to reduce the total number of parameters. Although this method can achieve an extremely high compression rate, it depends on specialized hardware (Han, Liu et al. 2016; Park, Li et al. 2016) for real speed-ups, so its inference speed gains in real-world deployments are often modest. In contrast, both channel pruning and layer pruning can achieve inference acceleration on standard computing devices. However, layer pruning removes the whole layer at once, which can severely impair model performance. Consequently, this paper focuses on channel pruning. It targets the channel dimension of layers, removing less important channels to reduce model size, while striving to preserve the model's structure and performance.

Speech Separation. The purpose of speech separation is to separate a single speech signal from a speech mixture. These methods can be roughly divided into two categories: time domain and time-frequency domain. Time domain methods directly utilize the original audio signal to achieve separation. For example, Conv-TasNet (Luo and Mesgarani 2019) employs a linear encoder to create speech waveform representations optimized for speaker separation, with a linear decoder converting them back. A temporal convolutional network with stacked 1D dilated convolutional blocks is used to recognize masks and effectively capture long-term dependencies. DPT-Net (Chen, Mao, and Liu 2020) introduces direct context-awareness in speech sequence modeling through an improved transformer that integrates recurrent neural networks into the original transformer. In contrast, time-frequency domain methods need to first convert the audio signal into a spectrogram representation using Short-Time Fourier Transform (STFT) to achieve separation. For instance, TF-GridNet (Wang, Cornell et al. 2023) employs stacked multi-path blocks containing intraframe spectral, sub-band temporal, and full-band self-attention modules to jointly exploit local and global spectro-temporal information for separation. BSRNN (Luo and Yu 2023) explictly splits the spectrogram of the mixture into subbands and perform interleaved band-level and sequence-level modeling. While significant advances have been achieved in speech separation performance, current research mainly focuses on laboratory benchmarks while overlooking critical deployment requirements in practical systems, particularly the need for low-latency processing and computationally efficient operation.

Efficient Speech Separation. In real-world applications, speech separation models need to not only pursue separation quality, but also consider the computational efficiency for real-time processing. To this end, TDANet (Li, Yang, and Hu 2022) proposes an efficient lightweight architecture using top-down attention, achieving competitive performance with lower computational costs. Li et al. (Li and Luo 2024) introduce a dynamic neural network that trains a large model with dynamic depth and width during the training phase and selects a subnetwork from it with arbitrary depth and width during the inference phase. Recently, Tiger (Xu, Li et al. 2024) utilizes prior knowledge to divide frequency bands and compresses fre-quency information. We employ a multi-scale selective attention module to extract contextual features, while introducing a full-frequency-frame attention module to capture both temporal and frequency contextual information. Although these efficient speech separation methods have achieved promising results, their reliance

on novel lightweight architecture designs leaves the critical challenge of compressing existing high-parameter models largely unaddressed.

Preliminary

Speech separation aims to extract individual speech signals of different speakers from a mixture, which can be formulated as:

$$x = \sum_{1}^{C} s_i + \epsilon. \tag{1}$$

$s_i \in \mathcal{R}^{1\times T}$ and $x \in \mathcal{R}^{1\times T}$ denote the waveform of speaker i and a multi-speaker audio signal with the length T, respectively. $\epsilon \in \mathcal{R}^{1\times T}$ denotes the noise signal and C denotes the number of speakers.

For speech separation tasks, most current state-of-the-art models (Hu, Li et al. 2021; Li, Yang, and Hu 2022; Tzinis, Wang, and Smaragdis 2020; Xu, Li et al. 2024) use a three-stage modular design of "an audio encoder $\rightarrow$ a separation network $\rightarrow$ an audio decoder". Specifically, the audio encoder converts the mixed audio signal into a mixture audio representation. Subsequently, the separation network utilizes a deep neural network to produce a set of speaker-specific masks. Each target speech representation is then obtained by element-wise multiplying the mixture audio representation with its corresponding mask. Finally, the target waveform is reconstructed using the target speech representation through an audio decoder.

Method

Channel Pruning for Speech Separation Models

This paper aims to slim a pre-trained speech separation model by pruning its channels. Given a L-layer pre-trained model, channel pruning aims to find a set of binary masks

$$\mathcal{M}_{L\times 1} = \{m_1, m_2, \cdots, m_L\}, \tag{2}$$

where each mask $m_l \in \{0,1\}^{C_l}$ corresponds to the l-th layer and C_l is its number of channels. The objective is to mark each channel for removal (0) or retention (1) so as to reduce model complexity while maintaining its performance. To obtain the mask, a common paradigm in prior work is to minimize the loss $\mathcal{L}$ after pruning, which can be formulated as:

$$\min_{\Theta,\mathcal{M}} \mathbb{E}_{(\mathbf{x})}\Big[\mathcal{L}\big(f(\mathbf{x},\Theta,\mathcal{M})\big)\Big], \tag{3}$$

where Θ and $\mathbf{x}$ represent the pre-training weights and input dataset respectively. However, the direct joint optimization of discrete masks and continuous weights is neither computationally tractable nor easy to converge. To this end, we decouple the joint optimization by first optimizing the masks and then fine-tuning the weights.

$$\underbrace{\min_{\Theta}}_{\text{Weight Learning}} \quad \underbrace{\min_{\mathcal{M}} \mathbb{E}_{(\mathbf{x})}\Big[\mathcal{L}\big(f(\mathbf{x},\Theta,\mathcal{M})\big)\Big]}_{\text{Mask Learning}}. \tag{4}$$

Although this objective formulated by Eq. (4) may seem simple to implement, it still has two challenges that make it difficult to work in practice:

- Masks are discrete variables and difficult to optimize using gradient descent;
- The number of mask combinations explodes, making optimization extremely difficult.

To address this, we introduce *SepPrune*, which makes the masks optimizable and decouples their selection from the subsequent weight fine-tuning.

Channel Pruning via Differentiable Masks

In this subsection, we delve into our *SepPrune*. As illustrated in Fig. 1, our method consists of three core stages:

- **Structural Computational Analysis**: Identify the layer contributing most to the overall computation.
- **Mask Learning with Gumbel-Softmax**: Learn a binary channel mask via Gumbel-Softmax to select a substructure that minimizes task loss.
- **Channel Pruning and Weight Refinement**: Perform channel pruning based on the obtained mask and fine-tune the remaining weights to recover performance.

Structural Computational Analysis. Different from traditional convolutional neural networks (Simonyan and Zisserman 2014; He, Zhang et al. 2016) and transformers(Liu, Lin et al. 2021; Touvron, Lavril et al. 2023; Achiam, Adler et al. 2023), speech separation models usually consists of a set of an audio encoder, a deep separation network and an audio decoder. In speech separation models, the parameter distribution and computational complexity of different modules are often highly unbalanced. If we do not identify the "heavyweight" layers first and blindly prune all modules uniformly, we are likely to weaken the already lightweight layers or over-prune the key layers, resulting in a significant performance degradation. Therefore, we first perform structural computational analysis to these models. In this paper, we mainly use the TDANet (Li, Yang, and Hu 2022), A-FRCNN (Hu, Li et al. 2021) and SudoRM-RF (Tzinis, Wang, and Smaragdis 2020) to conduct experiments. Specifically, given a pre-trained model $f(\Theta)$, we use the widely-used protocols, i.e., number of parameters (denoted as Params) and required Float Points Operations (denoted as FLOPs), to evaluate model size and computational requirement. To ensure the reproducibility of the results, we uniformly utilize the ptflops to perform precise statistics on Params and FLOPs. As shown in Table 1, we find that the separation network accounts for more than 82% of the total parameters and 76% of the FLOPs of these speech separation models. It can be seen that the separation network is the module with the greatest pruning benefit, so in this paper we mainly perform channel pruning on this module to minimize the computational overhead.

Mask Learning with Gumbel-Softmax. After locating the parts with the most parameters, our next goal is to find the channels that need to be pruned (masked) by optimizing Eq. (4). As we mentioned in the previous paragraph, the objective formulated by Eq. (4) faces two critical challenges. First, the exhaustive search space for binary masks can be prohibitively large even at low pruning ratios. For instance, masking a layer with 128 channels at 25% sparsity

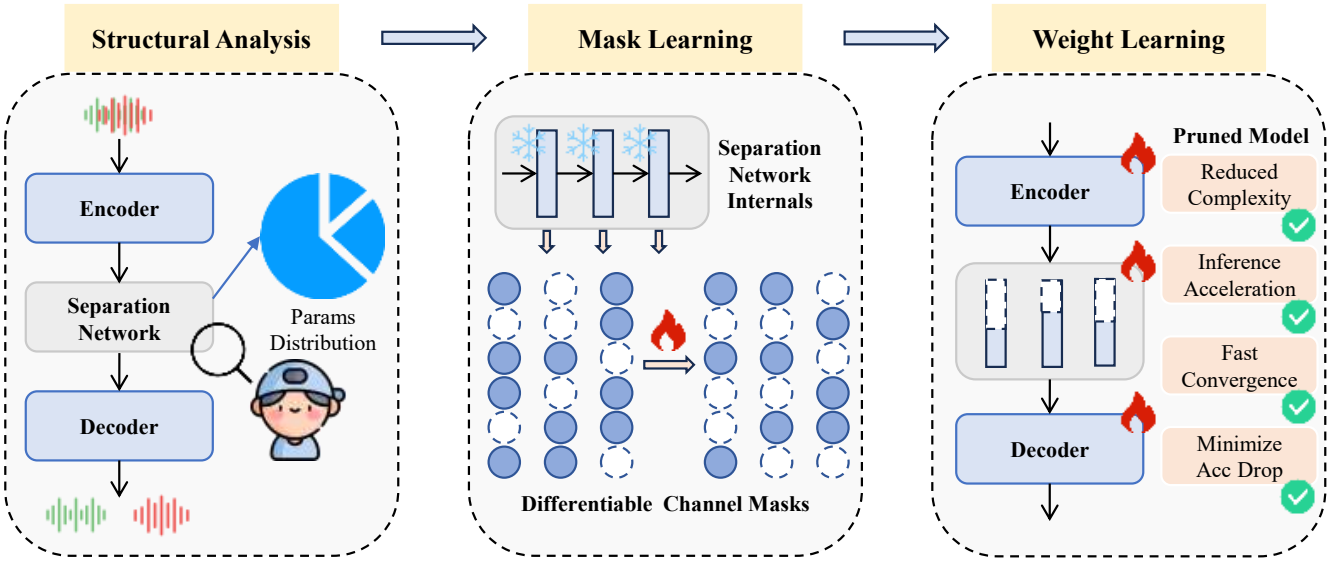

Figure 1: The overall pipeline of the proposed *SepPrune*.

Model	Total Params	Total FLOPs	Params of the SM	FLOPs of the SM	SM Parameter Ratio	SM FLOPs Ratio
A-FRCNN-12	5.13 M	28.58 GMac	4.22 M	26.56 GMac	82.31%	92.94%
A-FRCNN-16	5.13 M	37.44 GMac	4.22 M	35.42 GMac	82.31%	94.59%
SuDoRM-RF1.0x	2.72 M	4.65 GMac	2.35 M	3.57 GMac	86.50%	76.86%
TDANet	2.35 M	4.77 GMac	2.29 M	4.59 GMac	97.44%	96.22%
TDANet Large	2.35 M	9.20 GMac	2.29 M	9.09 GMac	97.44%	98.84%

Table 1: Statistics of the number of parameters and FLOPs of different models. SM denotes the separation network.

requires evaluating C_{128}^{32} possible solutions, making it difficult to use strategies such as evolutionary algorithms (Lin, Ji et al. 2020a) or reinforcement learning (Wang and Kindratenko 2024). It is worth noting that some layers of the speech separation model have 512 or more channels, and more than one layer needs to be masked, which means that the search space is actually much larger than C_{128}^{32}. Besides, while gradient-based optimization would be ideal for this high-dimensional search space, the discrete nature of binary masks $(0/1)$ fundamentally blocks gradient flow. To overcome both of these challenges, we introduce the Gumbel-Softmax (Fang, Li et al. 2024; Fang, Yin et al. 2024; Gumbel 1954; Jang, Gu, and Poole 2016) technique to convert discrete masks into differentiable "soft" probability distributions, allowing us to efficiently explore the exponential mask space using gradient descent.

Specifically, let $F_i \in \mathcal{R}^{B\times C_i\times H_i}$ denote the feature representation of layer i, where C_i is the number of channels, B is the batch size and H_i represents the feature length. To prune redundant channels, we assign a learnable importance score $\alpha_i \in \mathcal{R}^{C_i}$ for each layer, where each scalar $a_{i,j}$ represents the importance score of the j-th channel. Then we apply the Gumbel-Softmax technique to the weights α_i:

$$\pi_i = \frac{\exp\left(\left(\log\left(\alpha_i\right) + g_i\right)/\tau\right)}{\sum_j \exp\left(\left(\log\left(\alpha_j\right) + g_j\right)/\tau\right)}, \tag{5}$$

where $g_i = -\log(-\log(\mathcal{U}))$ is the random noise drawn from the Gumbel distribution, with $\mathcal{U} \sim \text{Uniform}(0,1)$ and τ is a temperature term. Subsequently, we further utilize the improved Straight-Through Estimator (Bengio, Léonard, and Courville 2013) to binarize π_i to $m_i \in \{0,1\}^{C_i}$. In the backward propagation phase, we preserve the identity mapping of gradients while bounding their magnitudes within the interval [-1, 1] to mitigate the risk of the gradient explosion.

$$\begin{cases} m_i = \frac{\text{sign}(\pi_i - \epsilon)+1}{2} & \text{FP}, \\ \bigtriangledown_{\pi_i} = \text{Clip}(\pi_i, -1, 1) = \max\left(-1, \min(1, \pi_i)\right) & \text{BP}, \end{cases} \tag{6}$$

where ϵ is a hyperparameter used to control the masking (pruning) ratio. Leveraging the mask m_i and the feature representation F_i, we can obtain the masked feature $\hat{F}_i = m_i \odot F_i$. For simplicity, we omit α_i and use $\mathcal{A}$ to represent the set of learned weight α_i. Then $\mathcal{A}$ is derived by optimizing Eq. (7) using gradient descent.

$$\min_{\mathcal{A}} \mathbb{E}_{(\mathbf{x})}\Big[\mathcal{L}\big(f(\mathbf{x}, \Theta, \mathcal{A})\big)\Big], \quad \mathcal{A} \leftarrow \mathcal{A} - \eta_{\mathcal{A}} \frac{\partial \mathcal{L}}{\partial \mathcal{A}}. \tag{7}$$

Finally, we can obtain the $\mathcal{M}$ based on $\mathcal{A}$.

Channel Pruning and Weight Refinement. After obtaining the set of binary channel masks $\mathcal{M}$, we perform channel pruning by retaining the channels indexed $m_{i,j} = 1$ and removing those with $m_{i,j} = 0$. The pruned model is

fine-tuned to recover performance degradation caused by channel removal. Let $\hat{\Theta}$ denote the parameters of the pruned model, initialized from the surviving weights of the original model. The optimization objective is:

$$\min_{\hat{\Theta}} \mathbb{E}_{(\mathbf{x})}\Big[\mathcal{L}\big(f(\mathbf{x},\hat{\Theta})\Big], \quad \hat{\Theta} \leftarrow \hat{\Theta} - \eta_{\hat{\Theta}}\nabla_{\hat{\Theta}}\mathcal{L}. \tag{8}$$

Experiments

Model and Dataset

We report the performance of *SepPrune* on LRS2-2Mix (Li, Yang, and Hu 2022), Libri2Mix (Cosentino, Pariente et al. 2020) and EchoSet (Xu, Li et al. 2024). All of these datasets are at a sampling rate of 16 kHz. We describe the datasets used in detail below. As for models, we utilize TDANet (Li, Yang, and Hu 2022), A-FRCNN (Hu, Li et al. 2021) and SudoRM-RF (Tzinis, Wang, and Smaragdis 2020) to conduct experiments. These models are widely-used in the speech separation community (Li, Chen et al. 2024; Xu, Li et al. 2024).

Training and Evaluation

As for training the original models, to make a fair comparison with previous speech separation methods, we trained all models for 500 epochs in line with (Xu, Li et al. 2024). It is worth noting that our pruning method does not actually require this step. Since there is no pre-trained model directly available, we train the model ourselves. The batch size is set to 1 at the utterance level. We use the Adam optimizer with an initial learning rate of 0.001 and negative SI-SDR as the training loss (Le Roux, Wisdom et al. 2019). Besides, SDRi and SI-SDRi (Vincent, Gribonval, and Févotte 2006) are used for evaluation, with higher values indicating better performance. Once the best model has not been found for 15 consecutive epochs, we adjust the learning rate to half of the previous one. In addition, if the best model has not been found for 30 consecutive epochs, we stop the training early. As for mask learning, we set the initial learning rate to 0.1 and train all masks for 500 iterations. Since training the speech separation model is very expensive, in order to minimize the cost of mask learning, we only use 500 iterations. We verify the effects of different iterations in the ablation experiment section . Without specific instructions, we default to setting $\epsilon = 0.7$. In addition, the hyperparameters used for fine-tuning the pruned model are consistent with the model training. The Params and FLOPs are calculated for one second of audio at 16 kHZ. For all experiments, we used $8\times$NVIDIA V100 and $4\times$NVIDIA A100 for training and testing.

Comparisons with State-of-The-Art Methods

To evaluate the effectiveness of *SepPrune*, we compare our method with existing channel pruning methods (Random, Hrank (Lin, Ji et al. 2020b) and UDSP (Gao, Zhang et al. 2024)) on three benchmark datasets, including Libri2Mix, LRS2-2Mix, and EchoSet. Since these methods do not experiment on speech separation models, we reproduce them ourselves. As shown in Table 2, we report the performance in terms of SDRi and SI-SDRi (in dB), along with the number of parameters (Params) and FLOPs after pruning. Across all datasets and model structures (TDANet (Li, Yang, and Hu 2022), A-FRCNN-12 (Hu, Li et al. 2021) and SuDoRM-RF1.0x (Tzinis, Wang, and Smaragdis 2020)), *SepPrune* consistently achieves superior performance under the same pruning ratio. For example, on LRS2-2Mix dataset with the A-FRCNN-12 model, *SepPrune* not only outperforms all other pruning methods, but also achieves an SDRi of 12.59 dB and an SI-SDRi of 12.25 dB, both of which exceed the performance of the original model (10.90 dB and 10.50 dB, respectively). Besides, on the most challenging EchoSet dataset, *SepPrune* outperforms all baseline pruning strategies, yielding the highest SDRi and SI-SDRi in all cases. In summary, these results demonstrate its effectiveness and strong generalization ability.

Separation Effciency

In the previous section, we have verified the effectiveness of *SepPrune*. To further demonstrate the efficiency of *SepPrune*, we measure the time and display memory overhead required for the pruned model during training and inference. Specifically, we perform the backward process (training) and forward process (inference) $1,000$ times on one second of audio at a sampling rate of 16 kHz, and then take the average to represent the training and inference speeds. We report the GPU time and GPU display memory usage during training and inference, respectively. We utilize a single card when calculating GPU (NVIDIA A100) time. As shown in Table 3, *SepPrune* not only brings effective training acceleration, but also significantly saves GPU memory during training (up to 50.2%). Besides, *SepPrune* achieves actual inference time acceleration, accelerating A-FRCNN-12 by 1.09 times, TDANet by 1.08 times, and SuDoRM-RF1.0x by 1.13 times. Notably, the limited GPU display memory savings observed during inference stem from the fact that, in speech-separation models, the pruned parts themselves occupy only a small fraction of the total memory footprint. Besides, the models are already lean (2-5M parameters). For such models, fixed overheads (I/O, framework calls) constitute a large portion of inference time, diminishing the returns from pruning channels. In addition, the RNN architecture in speech separation models imposes a strict serial processing flow: the computation for the current frame is dependent on the hidden state from the previous one. The overall speed is thus bottlenecked by this un-parallelizable loop. In summary, *SepPrune* provides a practical solution for model acceleration.

SepPrune Enables Fast Convergence

To further evaluate the efficiency of *SepPrune* in actual pruning scenarios, we design two experiments to fine-tune pruned models with only 1 epoch on the LRS2-2Mix dataset. These experiments are designed with the expectation that pruned models will recover most of their performance with minimal fine-tuning in real deployments. Specifically, we prune three typical speech separation models (TDANet, A-FRCNN-12 and SuDoRM-RF1.0x) and fine-tune them for 1 epoch on the LRS2-2Mix dataset, and then compare them

Model	Method	LRS2-2Mix				Libri2Mix				EchoSet			
		Params	FLOPs	SDRi	SI-SDRi	Params	FLOPs	SDRi	SI-SDRi	Params	FLOPs	SDRi	SI-SDRi
A-FRCNN-12	-	5.13	28.58	10.90	10.50	5.13	28.58	15.54	15.03	5.13	28.58	8.56	7.66
	Random	3.06	16.52	12.03	11.68	3.09	16.76	15.03	14.78	3.08	16.57	7.41	6.26
	Hrank	3.06	16.52	12.45	12.11	3.09	16.76	15.32	15.00	3.08	16.57	8.01	6.99
	UDSP	3.06	16.52	12.49	12.08	3.09	16.76	15.23	14.94	3.08	16.57	7.93	6.94
	SepPrune	3.06	16.52	**12.59**	**12.25**	3.09	16.76	**15.60**	**15.10**	3.08	16.57	**8.12**	**7.13**
TDANet	-	2.35	4.77	12.74	12.45	2.35	4.77	13.42	12.92	2.35	4.77	8.78	7.93
	Random	1.92	4.33	11.91	11.52	1.67	4.07	13.02	12.08	1.64	4.04	7.51	6.49
	Hrank	1.92	4.33	12.33	12.01	1.67	4.07	13.23	12.46	1.64	4.04	7.69	6.67
	UDSP	1.92	4.33	12.46	12.26	1.67	4.07	13.56	12.79	1.64	4.04	7.99	7.03
	SepPrune	1.92	4.33	**12.72**	**12.41**	1.67	4.07	**13.70**	**13.21**	1.64	4.04	**8.52**	**7.62**
SuDoRM-RF1.0x	-	2.72	4.65	11.43	11.10	2.72	4.65	13.52	12.99	2.72	4.65	7.82	6.77
	Random	1.54	2.91	9.53	9.06	0.98	2.09	12.14	11.82	1.11	2.28	6.52	5.74
	Hrank	1.54	2.91	10.29	9.89	0.98	2.09	12.36	12.01	1.11	2.28	6.85	5.89
	UDSP	1.54	2.91	10.12	9.67	0.98	2.09	12.31	12.04	1.11	2.28	7.04	5.98
	SepPrune	1.54	2.91	**10.37**	**9.98**	0.98	2.09	**12.79**	**12.24**	1.11	2.28	**7.29**	**6.07**

Table 2: Performance comparison with existing pruning methods on Libri2Mix, LRS2-2Mix and EchoSet. "-" indicates the original model. Bold denotes the best performance and underline is the second-best. SDRi and SI-SDRi are recorded in dB.

Model	Type	Train				Inference			
		GPU Time	GPU Memory	Speed Up	Memory Savings	GPU Time	GPU Memory	Speed Up	Memory Savings
A-FRCNN-12	Original Model	160.67ms	3714MB	1.04×	38.50%	104.23ms	642MB	1.09×	2.20%
	Pruned Model	155.14ms	2284MB			95.38ms	628MB		
TDANet	Original Model	435.75ms	5130MB	1.06×	22.14%	223.59ms	640MB	1.08×	0.01%
	Pruned Model	409.56ms	3994MB			207.52ms	639MB		
SuDoRM-RF1.0x	Original Model	157.43ms	4084MB	1.04×	50.20%	89.87ms	612MB	1.13×	1.60%
	Pruned Model	150.97ms	2034MB			79.25ms	602MB		

Table 3: Efficiency comparisons of the original model and pruned model. Experiments are conducted on the LRS2-2Mix dataset.

with the original unpruned models and retrained models (train from scratch) of the same size as the pruned models.

As shown in Table 4, the original models take 493 epochs (TDANet), 136 epochs (A-FRCNN-12), and 86 epochs (SuDoRM-RF1.0x) to complete training, while *SepPrune* only fine-tune for 1 epoch and restore the performance of most models to more than 85%, fully demonstrating the efficiency of *SepPrune* in the training stage. Besides, we further explore whether it is more efficient to do a small amount of fine-tuning on the pruned model or to train a model of the same size from scratch under the same parameter budget. As shown in Table 5, the performance of the randomly initialized small model after 1 epoch of training is far behind the effect of fine-tuning the pruned model for 1 epoch. Training a model of the same size as the pruned model directly from scratch to achieve the same effect as fine-tuning the pruned model for 1 epochs requires dozens of epochs (36 for TDANet and 31 for A-FRCNN-12), which fully demonstrates the huge efficiency advantage of *SepPrune*. The performance recovery effect of SuDoRM-RF1.0x is significantly inferior to that of the other two models. We believe that this is mainly due to the fact that a large number of structures of the model are removed during the pruning process, making it difficult to quickly rebuild the model performance with only 1 epoch of fine-tuning. Despite this, its performance is still better than a randomly initialized model of the same size trained from scratch for 1 epoch, which shows that even if a large amount of structure is pruned, the retained pre-trained weights can still bring better initial performance than training from scratch with very limited epochs. In summary, *SepPrune* can not only effectively restore model performance, but also significantly reduce costs by dozens of times of training acceleration.

Ablation Study

We adopt the TDANet and A-FRCNN-12 trained on the LRS2-2Mix dataset in the ablation studies. The training configuration of ablation experiments is same as before.

Which optimization method is better: joint optimization of weights and masks, or step-by-step optimization? To verify whether optimizing masks and weights step by step is better than joint optimization, we use the masks obtained by joint optimization and separate optimization for channel pruning respectively. As shown in Table 6, the model obtained by step-by-step optimization achieves higher separation performance than the jointly-optimized one, improving SDRi by 0.54 dB and SI-SDRi by 0.72 dB. We believe that this is because step-by-step optimization focuses on mask search first, making the preserved structure fit the task more accurately.

The effect of mask learning with different numbers of iterations. For cost-saving reasons, we only perform 500 iterations of mask learning. Here, we conduct an ablation experiment with different iterations to evaluate the influence of the number of mask learning iterations on the final pruning effect. Specifically, we set the iteration to $\{300, 500, 700, 900, 1100\}$ for experiments. As shown in Table 7, different numbers of mask learning iterations do

Model	Fine-tuning 1 Epoch		Well-trained Model			Performance Recovery Rate	
	SDRi	SI-SDRi	SDRi	SI-SDRi	Training Epochs	SDRi	SI-SDRi
TDANet	11.17	10.81	12.74	12.45	493	87.68%	86.83%
A-FRCNN-12	9.43	8.94	10.90	10.50	136	86.51%	85.14%
SuDoRM-RF1.0x	5.18	4.06	11.43	11.10	86	45.32%	36.58%

Table 4: Performance recovery after pruning with only 1 epoch of fine-tuning on the LRS2-2Mix dataset. "Fine-tuning 1 Epoch" denotes the pruned model with 1 epoch fine-tuning. "Well-trained Model" denotes the pre-trained original model.

Model	Fine-tuning 1 Epoch		Training 1 Epoch		Comparable Performance			Training Acceleration
	SDRi	SI-SDRi	SDRi	SI-SDRi	SDRi	SI-SDRi	Training Epochs	
TDANet	11.17	10.81	4.31	2.80	11.13	10.75	36	36×
A-FRCNN-12	9.43	8.94	3.43	1.76	9.60	9.17	31	31×
SuDoRM-RF1.0x	5.18	4.06	4.43	2.96	5.03	3.85	2	2×

Table 5: Comparison of fine-tuning a pruned model and training a model of the same size from scratch. "Fine-tuning 1 Epoch" denotes the pruned model with 1 epoch fine-tuning. "Training 1 Epoch" means training a model of the same size as the pruned model from scratch for 1 epochs. "Comparable Performance" denotes training a model from scratch with the same size as the pruned model achieves performance comparable to fine-tuning the pruned model for 1 epoch.

Method	SDRi	SI-SDRi
Joint Optimization	12.05	11.53
Step-by-step Optimization	12.59	12.25

Table 6: Importance of optimizing masks and weights in steps. Experiments are conducted using A-FRCNN-12 on the LRS2-2Mix dataset.

Iteration	Params	FLOPs	SDRi	SI-SDRi
300	1.91 M	4.32 GMac	12.74	12.43
500	1.92 M	4.33 GMac	12.72	12.41
700	1.92 M	4.33 GMac	12.68	12.40
900	1.92 M	4.33 GMac	12.73	12.42
1100	1.90 M	4.31 GMac	12.71	12.41

Table 7: Pruned models obtained by mask learning with different numbers of iterations. Experiments are conducted using TDANet on the LRS2-2Mix dataset.

not affect the final performance or compression rate, so we set it to 500 by default in this paper to minimize the training cost while ensuring the pruning effect.

The influence of the value of ϵ. In this paper, ϵ is a hyperparameter used to control the pruning ratio. Therefore, we set $\epsilon = \{0.5, 0.6, 0.7, 0.8, 0.9\}$ to conduct experiments. As shown in Table 8, changing the value of ϵ can effectively adjust the model complexity and performance. In this study, when $\epsilon = 0.7$, the model achieves a good balance between computational complexity and performance. Therefore, ϵ is set to 0.7 by default.

Conclusion

In this paper, we have presented *SepPrune*, the first pruning framework tailored specifically for deep speech separation models. *SepPrune* first performs a structural calculation analysis on existing models to determine the layers with the highest computational cost. Subsequently, *SepPrune* introduces differentiable masks to perform gradient-driven channel mask search and implements channel pruning based on the obtained masks. Experiments demonstrate that *SepPrune* outperforms the existing channel pruning methods. Besides, *SepPrune* can recover more than 85% of the performance of the original model with just 1 epoch of fine-tuning and converge much faster than training a model of the same size from scratch. Finally, *SepPrune* offers a novel pruning paradigm for the design of lightweight speech separation models on devices with limited resources.

ϵ	Params	FLOPs	SDRi	SI-SDRi
0.5	3.69 M	20.25 GMac	12.52	12.19
0.6	3.42 M	18.55 GMac	12.58	12.23
0.7	3.06 M	16.52 GMac	12.59	12.25
0.8	2.94 M	15.73 GMac	12.35	12.03
0.9	2.71 M	14.49 GMac	12.07	11.75

Table 8: Pruning models obtained with different ϵ. Experiments are conducted using A-FRCNN-12 on the LRS2-2Mix dataset.

Limitations. Although this paper verifies the universality and effectiveness of *SepPrune* on multiple mainstream models (Li, Yang, and Hu 2022; Hu, Li et al. 2021; Tzinis, Wang, and Smaragdis 2020), we have not yet conducted evaluations on the latest state-of-the-art models, such as Tiger (Xu, Li et al. 2024) and SPMamba (Li, Chen et al. 2024). In the future, we will work on conducting experiments on more representative models to further verify the applicability and generalization ability of *SepPrune*. Furthermore, we will delve into the reasons for the poor speedup performance on certain model architectures.

References

Achiam, J.; Adler, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bai, Y.; Wang, H.; et al. 2022. Dual lottery ticket hypothesis. *arXiv preprint arXiv:2203.04248*.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Briere, S.; Valin, J.-M.; et al. 2008. Embedded auditory system for small mobile robots. In *ICRA*, 3463–3468.

Chen, J.; Mao, Q.; and Liu, D. 2020. Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation. *arXiv preprint arXiv:2007.13975*.

Chen, S.; and Zhao, Q. 2018. Shallowing deep networks: Layer-wise pruning based on feature representations. *TPAMI*, 41(12): 3048–3056.

Chen, Z.; Xiang, J.; et al. 2023. Rgp: Neural network pruning through regular graph with edges swapping. *TNNLS*.

Cosentino, J.; Pariente, M.; et al. 2020. Librimix: An open-source dataset for generalizable speech separation. *arXiv preprint arXiv:2005.11262*.

Dai, R.; Li, C.; et al. 2025. Unbiased Missing-modality Multimodal Learning. In *ICCV*, 24507–24517.

Fang, G.; Li, K.; et al. 2024. TinyFusion: Diffusion Transformers Learned Shallow. *arXiv preprint arXiv:2412.01199*.

Fang, G.; Yin, H.; et al. 2024. Maskllm: Learnable semi-structured sparsity for large language models. *arXiv preprint arXiv:2409.17481*.

Gao, S.; Zhang, Y.; et al. 2024. BilevelPruning: unified dynamic and static channel pruning for convolutional neural networks. In *CVPR*, 16090–16100.

Gumbel, E. J. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Guo, S.; Wang, Y.; et al. 2020. Dmcp: Differentiable markov channel pruning for neural networks. In *CVPR*, 1539–1547.

Han, S.; Liu, X.; et al. 2016. EIE: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3): 243–254.

He, K.; Zhang, X.; et al. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *ICCV*, 1389–1397.

Hoang, D. N.; and Liu, S. 2023. Revisiting pruning at initialization through the lens of ramanujan graph. *ICLR*.

Hu, X.; Li, K.; et al. 2021. Speech separation using an asynchronous fully recurrent convolutional neural network. *NeurIPS*, 34: 22509–22522.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jiang, X.; Han, C.; and Mesgarani, N. 2025. Dual-path mamba: Short and long-term bidirectional selective structured state space models for speech separation. In *ICASSP*, 1–5.

Ke, Z.; Gong, H.; and Du, D. H. 2025. PM-Dedup: Secure Deduplication with Partial Migration from Cloud to Edge Servers. *arXiv preprint arXiv:2501.02350*.

Lan, Q.; and Tian, Q. 2025. ACAM-KD: adaptive and cooperative attention masking for knowledge distillation. *arXiv preprint arXiv:2503.06307*.

Le Roux, J.; Wisdom, S.; et al. 2019. SDR–half-baked or well done? In *ICASSP*, 626–630.

Li, H.; Kadav, A.; et al. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

Li, K.; Chen, G.; et al. 2024. Spmamba: State-space model is all you need in speech separation. *arXiv preprint arXiv:2404.02063*.

Li, K.; and Luo, Y. 2024. Subnetwork-to-go: Elastic neural network with dynamic training and customizable inference. In *ICASSP*, 6775–6779.

Li, K.; Yang, R.; and Hu, X. 2022. An efficient encoder-decoder architecture with top-down attention for speech separation. *arXiv preprint arXiv:2209.15200*.

Li, Y.; Lu, Y.; et al. 2024. SGLP: A similarity guided fast layer partition pruning for compressing large deep models. *arXiv preprint arXiv:2410.14720*.

Li, Y.; Yang, C.; et al. 2025. Frequency-aligned knowledge distillation for lightweight spatiotemporal forecasting. In *ICCV*, 7262–7272.

Lin, M.; Ji, R.; et al. 2020a. Channel pruning via automatic structure search. *arXiv preprint arXiv:2001.08565*.

Lin, M.; Ji, R.; et al. 2020b. Hrank: Filter pruning using high-rank feature map. In *CVPR*, 1529–1538.

Ling, G.; Wang, Z.; and Liu, Q. 2024. SlimGPT: Layer-wise Structured Pruning for Large Language Models. *NeurIPS*, 37: 107112–107137.

Liu, H.; and Xiao, L. 2025. RE-GRPO: Leveraging hard negative cases through large language model guided self training. *Neurocomputing*, 132543.

Liu, J.; Meng, S.; et al. 2025. Aligning Vision to Language: Annotation-Free Multimodal Knowledge Graph Construction for Enhanced LLMs Reasoning. In *ICCV*.

Liu, S.; Chen, T.; et al. 2022. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. *arXiv preprint arXiv:2202.02643*.

Liu, X.; Lu, Y.; Wang, X.; and Wu, X. 2025. Training-free multi-style fusion through reference-based adaptive modulation. In *ACPR*, 149–163.

Liu, Z.; Lin, Y.; et al. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 10012–10022.

Liu, Z.; Mu, H.; et al. 2019. Metapruning: Meta learning for automatic neural network channel pruning. In *ICCV*, 3296–3305.

Lou, Y.; Hu, H.; Ma, S.; Zhang, Z.; Wang, L.; Ge, J.; and Tao, X. 2025. DRF: LLM-AGENT Dynamic Reputation Filtering Framework. In *International Conference on Neural Information Processing*, 127–141. Springer.

Lu, Y.; Cheng, H.; et al. 2024. Reassessing Layer Pruning in LLMs: New Insights and Methods. *arXiv preprint arXiv:2411.15558*.

Lu, Y.; Yang, W.; et al. 2022. Understanding the dynamics of DNNs using graph modularity. In *ECCV*, 225–242.

Lu, Y.; Zhu, Y.; et al. 2024. A generic layer pruning method for signal modulation recognition deep learning models. *TCCN*.

Luo, Y.; and Mesgarani, N. 2019. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *TASLP*, 27(8): 1256–1266.

Luo, Y.; and Yu, J. 2023. Music source separation with band-split RNN. *TASLP*, 31: 1893–1901.

Ma, S.; Huang, Y.; and Lin, Y. 2025. DramaBench: A Six-Dimensional Evaluation Framework for Drama Script Continuation. *arXiv preprint arXiv:2512.19012*.

Ma, X.; Fang, G.; and Wang, X. 2023. Llm-pruner: On the structural pruning of large language models. *NeurIPS*, 36: 21702–21720.

Maryn, Y.; Ysenbaert, F.; et al. 2017. Mobile communication devices, ambient noise, and acoustic voice measures. *Journal of Voice*, 31(2): 248–e11.

Meng, S.; Luo, Y.; and Liu, P. 2025. Grounding creativity in physics: A brief survey of physical priors in aigc. *arXiv preprint arXiv:2502.07007*.

Park, J.; Li, S.; et al. 2016. Faster cnns with direct sparse convolutions and guided pruning. *arXiv preprint arXiv:1608.01409*.

Simonyan, K.; and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*.

Sun, M.; Liu, Z.; et al. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Tang, H.; Lu, Y.; and Xuan, Q. 2023. Sr-init: An interpretable layer pruning method. In *ICASSP*, 1–5.

Touvron, H.; Lavril, T.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tzinis, E.; Wang, Z.; and Smaragdis, P. 2020. Sudo rm-rf: Efficient networks for universal audio source separation. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6.

Vincent, E.; Gribonval, R.; and Févotte, C. 2006. Performance measurement in blind audio source separation. *TASLP*, 14(4): 1462–1469.

Wang, B.; and Kindratenko, V. 2024. Rl-pruner: Structured pruning using reinforcement learning for cnn compression and acceleration. *arXiv preprint arXiv:2411.06463*.

Wang, H.; Qin, C.; et al. 2021. Recent advances on neural network pruning at initialization. *arXiv preprint arXiv:2103.06460*.

Wang, Y.; Li, D.; and Sun, R. 2023. NTK-SAP: Improving neural network pruning by aligning training dynamics. *arXiv preprint arXiv:2304.02840*.

Wang, Z.-Q.; Cornell, S.; et al. 2023. TF-GridNet: Making time-frequency domain models great again for monaural speaker separation. In *ICASSP*, 1–5.

Wu, J.; Zhu, D.; et al. 2023. Efficient layer compression without pruning. *TIP*, 32: 4689–4700.

Wu, Y.; Liu, X.; Zhao, C.; and Wu, X. 2025. Prompt-Guided Dual Latent Steering for Inversion Problems. *arXiv preprint arXiv:2509.18619*.

Xiao, C.; et al. 2024. Confusion-resistant federated learning via diffusion-based data harmonization on non-IID data. *NeurIPS*, 37: 137495–137520.

Xiao, X.; Zhang, Y.; et al. 2025. Prompt-based Adaptation in Large-scale Vision Models: A Survey. arXiv:2510.13219.

Xu, M.; Li, K.; et al. 2024. TIGER: Time-frequency Interleaved Gain Extraction and Reconstruction for Efficient Speech Separation. *arXiv preprint arXiv:2410.01469*.

Yang, Y.; Zhen, K.; et al. 2025. Wanda++: Pruning Large Language Models via Regional Gradients. *arXiv preprint arXiv:2503.04992*.

Yao, J.; Li, C.; and Xiao, C. 2024. Swift sampler: Efficient learning of sampler by 10 parameters. *NeurIPS*, 37: 59030–59053.

You, H.; and Liu, B. 2024. Application of pseudometric functions in clustering and a novel similarity measure based on path information discrepancy. In *ICONIP*. Springer.

Zeng, X.; Wang, H.; et al. 2025. LENSLLM: Unveiling Fine-Tuning Dynamics for LLM Selection.

Zhang, H.; Huang, B.; et al. 2025. Sensitivity-LoRA: Low-Load Sensitivity-Based Fine-Tuning for Large Language Models. In *Findings of EMNLP*, 13185–13199.

Zhang, H.; Xu, H.; Liu, H.; Yu, X.; Zhang, X.; and Wu, C. 2025. Conditional variational underwater image enhancement with kernel decomposition and adaptive hybrid normalization. *Neurocomputing*, 130845.

Zhang, J.; Gao, J.; et al. 2025. Time-llama: Adapting large language models for time series modeling via dynamic low-rank adaptation. In *ACL Student Research Workshop*, 1145–1157.

Zhou, S.; Li, L.; et al. 2024. LiDAR-PTQ: Post-Training Quantization for Point Cloud 3D Object Detection.

Zhou, S.; Yuan, Z.; et al. 2024. Information Entropy Guided Height-aware Histogram for Quantization-friendly Pillar Feature Encoder. *arXiv preprint arXiv:2405.18734*.

Zhou, S.; Yuan, Z.; et al. 2025. Pillarhist: A quantization-aware pillar feature encoder based on height-aware histogram. In *CVPR*, 27336–27345.

Zhou, Y.; Zhao, Z.; et al. 2025. Dropping Experts, Recombining Neurons: Retraining-Free Pruning for Sparse Mixture-of-Experts LLMs. In *Findings of EMNLP*, 15169–15186.

Zhuang, Z.; Tan, M.; et al. 2018. Discrimination-aware channel pruning for deep neural networks. *NeurIPS*, 31.