Vicinagearth

RESEARCH

Open Access

Towards a flexible and unified architecture for speech enhancement

Linfeng Feng[1,2,3], Chi Zhang[2] and Xiao-Lei Zhang[1,2,3*]

Abstract

Deploying neural networks across devices with vastly different computational budgets is critical for realizing AI Flow at the network edge. This paper contributes to cooperative family-model systems by proposing a single network that can be dynamically sliced into subnetworks of varying sizes, enabling seamless adaptation to heterogeneous resource constraints across the device-edge-cloud continuum. To scale broadly, we make both the width and depth of the network flexible. For width scaling, we introduce FlexAttention, which enables a variable number of attention heads to adaptively adjust computational load. We also propose FlexRMSNorm, a normalization layer that dynamically adapts to different network widths. Combined with early-exit strategies, these components form a network that scales in both width and depth. Built from these flexible modules, we present SEFlow, a causal and sampling-rate-agnostic model that handles a wide range of speech enhancement tasks, including denoising, dereverberation, declipping, and packet loss concealment. Experimental results demonstrate that SEFlow is comparable to the state-of-the-art task-specific models across multiple speech enhancement tasks. Remarkably, even sub-networks as small as 1% of the full network remain effective in low-resource scenarios. Our demonstrations are available on the project homepage.

Keywords Speech enhancement, Flexible neural networks, Resource-constrained inference, AI Flow

1 Introduction

Deep neural networks (DNNs) have emerged as a transformative driving force behind the recent boom in Artificial Intelligence (AI), enabling breakthroughs across diverse domains [1–5]. Recently, the paradigm of AI Flow [6, 7] at the network edge has emerged as a transformative approach to distributed intelligence, where heterogeneous computational resources across devices, edge nodes, and cloud servers collaborate to deliver seamless AI services. This evolution coincides with the flourishing advancement of DNNs, where many works have achieved state-of-the-art performance across various modalities by scaling up model size—ranging from language models [8–10], vision applications [11–13], speech systems [14], video generation [15], to graph tasks [16]. However, realizing AI Flow's vision of ubiquitous edge intelligence faces critical challenges when deploying such large models across resource-constrained environments. The fundamental mismatch between fixed computational costs and the diverse platform capabilities—from personal laptops to cloud servers—creates bottlenecks in the intelligence flow. Under the AI Flow principle for efficient collaborative inference, a single family-model system should dynamically adapt to varying computational budgets across the device–edge–cloud continuum. Motivated by these challenges and the need to establish foundational infrastructure for AI Flow deployment, our key goal is to train a unified network that can be directly decomposed into sub-networks with significantly different computational costs, enabling seamless intelligence flow

*Correspondence:
Xiao-Lei Zhang
xiaolei.zhang@nwpu.edu.cn
[1] School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, China
[2] Institute of Artificial Intelligence (TeleAI), China Telecom, Beijing, China
[3] Research and Development Institute of Northwestern Polytechnical University in Shenzhen, Shenzhen, China

across diverse platforms without requiring fine-tuning or distillation.

In this context, we focus on designing a speech enhancement (SE) network that exemplifies AI Flow principles through widely adjustable computational cost. Speech enhancement serves as an ideal testbed for AI Flow implementation for several reasons. First, SE is a fundamental task in speech processing that naturally demands deployment across the entire device-edge-cloud spectrum. Second, it presents diverse computational requirements that mirror the heterogeneous nature of AI Flow environments: low-resource edge devices such as earphones and smartphones typically require models operating at less than 1 billion multiply–accumulate operations per second (GMACs/s), while edge servers can support tens of GMACs/s, and cloud-computing platforms can handle hundreds of GMACs/s. Specifically, different SE models use vastly different architectures and typically have different parameter efficiency. Therefore, computational cost (MACs) is more meaningful than model parameter count for SE models [17].

We construct a neural network with adjustable depth and width, enabling dynamic adaptation to varying computational budgets in the device-edge-cloud continuum. As illustrated in Fig. 1, our approach embodies the AI Flow training philosophy: we jointly train the full network alongside randomly-sampled subnetworks at each training step, creating a unified architecture that inherently supports diverse computational requirements. This cooperative training strategy ensures that after the training stage, subnetworks of dramatically different sizes—ranging from lightweight configurations suitable for resource-constrained edge devices to comprehensive models for powerful cloud servers. Following this principle, we propose *FlexAttention* and *FlexRMSNorm* as core components for dynamically controlling layer width. Combined with early-exit strategies for depth control, we construct a neural network architecture with adjustable depth and width that enables unprecedented scalability in both parameter count and computational cost.

In addition, we also aim for it to generalize across diverse application scenarios. Speech signals are often corrupted by various types of interference, leading to a range of sub-tasks in SE. Common interference includes environmental noise and room reverberation, which results in two SE sub-tasks—speech denoising [18] and speech dereverberation [19]. Since reverberation can be viewed as a special form of noise, these two sub-tasks are often dealt with together under the same framework [20]. Another type of degradation arises from amplitude limitations in recording devices, where the waveform gets clipped, resulting in a SE sub-task of declipping [21]. Notably, clipping in the time-domain introduces artifacts in the time-frequency (TF) domain, which can also be viewed as noise. Thus, declipping in the TF domain can naturally be treated as a denoising task [22]. Another SE sub-task is packet loss concealment (PLC) [23], where the input speech is corrupted due to packet loss during the transmission in communication networks.

To summarize, the aforementioned SE sub-tasks, including speech denoising, dereverberation, declipping, and PLC, are highly related and share strong similarities. However, existing research typically focuses on only one or a few of them, with independent models trained for each task. To address these tasks efficiently, another key goal of this work is to build a unified SE network that is generalizable across all the above sub-tasks without retraining. Our contributions are as follows:

- We propose a widely flexible neural network architecture. Specifically, we propose FlexAttention and FlexRMSNorm as width-flexible modules. FlexAttention enables an adjustable number of attention heads. In addition, we introduce FlexRMSNorm, a normalization layer with adjustable affine parameter size. The above modules, combined with the early-exit tech-

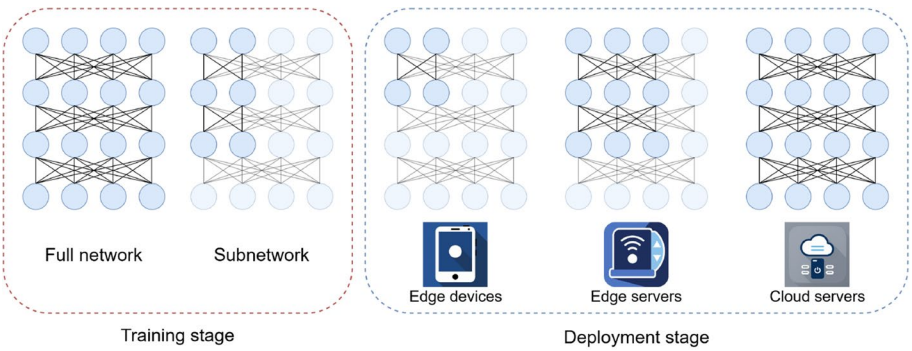


Fig. 1 Training and deployment of SEFlow. We jointly train the full-network and randomly-sampled-subnetworks at each step. After training, subnetworks of different sizes can be deployed to match the computational budgets of target devices

niques, make a single neural network architecture highly flexible. In addition, we design an acceleration strategy for multi-GPU training tailored to this flexible architecture.

- We propose a unified architecture, SEFlow, for speech enhancement, built upon the aforementioned flexible design. Trained via dynamic data augmentation by applying diverse combinations of distortions to clean speech, the model learns to handle a wide range of challenging degradations effectively. In addition, SEFlow is causal and sampling-rate-agnostic, supporting real-time inference across all common sampling rates. In addition, we design an acceleration strategy for multi-GPU training tailored to multi sampling rates.
- Experimental results show that, in terms of scalability, SEFlow maintains usable performance even when operating at just 1% of the full model's MACs. In terms of unification, SEFlow achieves performance competitive with the state-of-the-art task-specific models across a range of SE sub-tasks.

This paper is organized as follows. Section 2 presents related work. Section 3 describes the proposed flexible architecture in detail. Section 4 presents the design of unified architecture for speech enhancement. Section 5 outlines the experimental setup and results. Finally, Sect. 7 concludes the current work.

2 Related work

Recent studies have explored the deployment of a single network across platforms with varying computational capacities. Neural architecture search [24] has been applied to discover architectures that meet different constraints. Model pruning [25] and knowledge distillation [26] are commonly used to derive networks of different scales. The above methods are subject to fine-tuning.

In contrast, early-exit methods [27] accelerate inference by allowing intermediate layers (e.g., the output of a residual block) to produce predictions. However, early-exit primarily adjusts network depth while keeping the network width fixed, which limits the granularity of model partitioning. Slimmable networks [28] enable dynamic width adjustment in neural networks, applicable to linear layers and convolutional channels, etc. To improve the slimmable networks, [29] proposed training techniques such as the sandwich rule and inplace distillation. From the perspective of activating only a subset of neurons, we think that early-exit mechanisms control network depth by using only the early layers, while width flexibility enables adaptive neuron activation within each layer. An activated subset forms a sub-network, while the larger network can be regarded as an ensemble [30] of multiple smaller networks.

In the field of SE, several works [31–34] have employed early-exit [27] mechanisms to construct enhancement or separation models. Among them, [31, 32] further explored exit strategies based on the similarity between the outputs of consecutive layers, using a threshold to determine when to exit. Meanwhile, [35, 36] utilized slimmable networks [28] to build separation models. Miccini et al. [37] introduced dynamic channel pruning to control the cost of enhancement models.

3 Flexible architecture

3.1 Overview

To describe the flexible design, we first introduce the overall workflow of the full network. Given a corrupted speech signal in the TF domain, denoted as $Y \in \mathbb{C}^{T\times F}$, where T is the number of time frames and F is the number of frequency bins. The speech signal Y is first encoded by a band encoder f_{enc} into the embedding $Z^0 \in \mathbb{R}^{T\times K\times D}$, where K is the number of frequency bands and D is the embedding dimension. This embedding is then modeled by a stack of B residual blocks. We denote the output of the b-th block as $Z^{(b)} \in \mathbb{R}^{T\times K\times D}$. The residual connection [38] is applied as $Z^b = Z^{b-1} + Z^{(b)}$. After passing through all B blocks, we obtain $Z^B \in \mathbb{R}^{T\times K\times D}$, which is then decoded by a band decoder f_{dec} to produce the final enhanced speech signal $\hat{X} \in \mathbb{C}^{T\times F}$.

Based on the above full network, the core idea of our flexible architecture design lies in controlling B and D to adjust the network's depth and width—without fine-tuning and distillation. Specifically, for depth, we replace B with a configurable $\bar{B} \in \{1, \ldots, B\}$. For width, we replace D with a configurable $\bar{D} \le D$. Assuming a full multi-head attention has H heads, each with dimension $d = D/H$, we can adjust the width by replacing H with a configurable $\bar{H} \in \{1, \ldots, H\}$, leading to a adjustable width of $\bar{D} = \bar{H}d$. Sections 3.3 and 3.2 provide detailed descriptions of these designs.

3.2 Flexible modules for adjustable width

This section introduces the flexible modules for controlling network width. We first review the standard linear layer, and then present corresponding adjustable designs for the attention and normalization layers.

3.2.1 Flexible linear

We decompose the width control of the linear layer into the control of its input width and output width. A linear layer with I inputs and J outputs computes each output as a weighted sum of all inputs plus a bias term:

$$y_j = b_j + \sum_{i=1}^{I} w_{ji} x_i, \quad \forall j = 1, \ldots, J \tag{1}$$

Clearly, each of the J outputs depends on its own set of learnable parameters. Therefore, we can control the output width of a linear layer by slicing the weight matrix $\mathbf{W} \in \mathbb{R}^{J \times I}$ to $\mathbb{R}^{\bar{J} \times I}$ and the bias vector $\mathbf{b} \in \mathbb{R}^{J}$ to $\mathbb{R}^{\bar{J}}$, where $\bar{J} \leq J$. For controlling the input width, comparing Eq. (9) with Eq. (1), we find that, in both cases, a feature is formed by summing contributions from multiple neurons. Motivated by the principle of early-exit which activates only early layers, we control the input width by activating only early neurons, i.e., the first $\bar{I}$ neurons, where $\bar{I} \leq I$. The above two components formulate the proposed **FlexLinear**, as illustrated in Fig. 2(a):

$$y_j = b_j + \sum_{i=1}^{\bar{I}} w_{ji} x_i, \quad \forall j = 1, \ldots, \bar{J} \tag{2}$$

Its computation involves slicing the weight matrix $\mathbf{W} \in \mathbb{R}^{J \times I}$ to $\bar{\mathbf{W}} \in \mathbb{R}^{\bar{J} \times \bar{I}}$ and the bias vector $\mathbf{b} \in \mathbb{R}^{J}$ to $\bar{\mathbf{b}} \in \mathbb{R}^{\bar{J}}$ for parallel computation. Because the input and output of a residual block must keep the same shape, I and J are scaled proportionally. As a result, its scalability with respect to the width follows a quadratic relationship.

3.2.2 Flexible attention

Attention [39] is a popular module for sequence modeling in recent years. It typically divides the embedding dimension across multiple heads. This naturally leads to the idea of controlling the attention width by adjusting the number of heads. Based on this, we propose FlexAttention. Suppose a standard multi-head attention has H heads, each with dimension $d = D/H$, and has an input $\mathbf{x} \in \mathbb{R}^{D \times T}$ with sequence length T and embedding dimension D. The proposed **FlexAttention** sets the maximum number of heads to H, and selectively uses only $\bar{H} \leq H$ heads. Given $\bar{\mathbf{x}} \in \mathbb{R}^{\bar{D} \times T}$, $\bar{D} = \bar{H}d$, it can be divided into $\bar{H}$ heads. Specifically, as illustrated in Fig. 2(b), the input is then projected into queries, keys, and values using FlexLinear mappings:

$$\mathbf{Q} = \mathbf{W}^q \bar{\mathbf{x}} \in \mathbb{R}^{\bar{H}d \times T}, \quad \mathbf{K} = \mathbf{W}^k \bar{\mathbf{x}} \in \mathbb{R}^{\bar{H}d \times T}, \quad \mathbf{V} = \mathbf{W}^v \bar{\mathbf{x}} \in \mathbb{R}^{\bar{H}d \times T}, \tag{3}$$

where $\mathbf{W}^q$, $\mathbf{W}^k$, and $\mathbf{W}^v$ are all implemented via the proposed FlexLinear. When $\bar{H}$ heads are selected, the shapes of these matrices are adjusted to $\bar{H}d \times \bar{H}d$. The outputs are then split into $\bar{H}$ heads, with the h-th head corresponding to $\mathbf{Q}_h \in \mathbb{R}^{T \times d}$, $\mathbf{K}_h \in \mathbb{R}^{T \times d}$, $\mathbf{V}_h \in \mathbb{R}^{T \times d}$. The scaled dot-product attention is computed independently for each of the h heads:

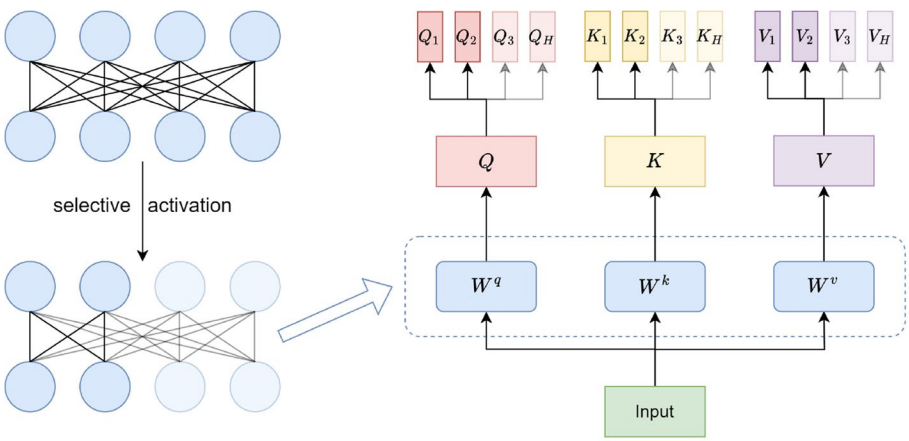
$$\text{Attention}(\mathbf{Q_h}, \mathbf{K_h}, \mathbf{V_h}) = \text{Softmax}\left(\frac{\mathbf{Q_h}\mathbf{K_h}^\top}{\sqrt{d}} + \mathbf{M}\right)\mathbf{V_h} \in \mathbb{R}^{T \times d}, \tag{4}$$

where $\mathbf{M} \in \mathbb{R}^{T \times T}$ is an optional additive mask, e.g., for causal or real-time tasks. Specifically, to enforce causality, entries in $\mathbf{M}$ corresponding to forbidden future positions are set to $-\infty$. When added to the attention logits before the softmax operation, these $-\infty$ values result in attention weights of 0 for those positions. Finally, the output is then concatenated and projected back to the original dimensionality:

$$\mathbf{y} = \mathbf{W}^o(\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})) \in \mathbb{R}^{\bar{D} \times T}. \tag{5}$$

where $\mathbf{W}^o \in \mathbb{R}^{\bar{D} \times \bar{D}}$ is a FlexLinear layer.

By controlling the head count, FlexAttention is able to support varying input embedding dimensions



(a) FlexLinear (b) FlexAttention

Fig. 2 Illustration of FlexLinear and FlexAttention. FlexLinear can adjust the width of the input and output, while FlexAttention can adjust the number of heads of the attention. For simplicity, we omit the subsequent parts after obtaining the features of the attention heads

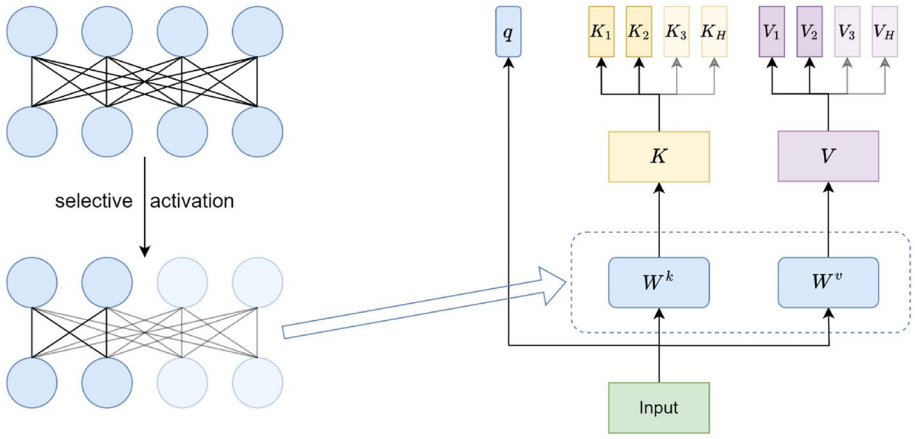


Fig. 3 Illustration of FlexFusionAttention. FlexFusionAttention can adjust the number of heads of the attention. For simplicity, we omit the subsequent parts after obtaining the features of the attention heads

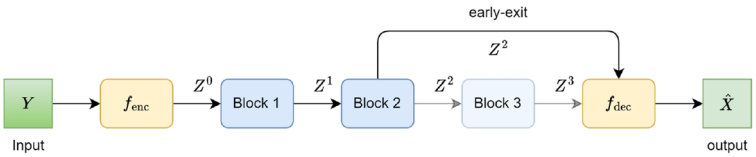


Fig. 4 Illustration of early-exit. The full network consists of B residual blocks. Under the early-exit, decoding can be performed after the $\bar{B}$-th block

$\bar{D} \in \{d, 2d, \ldots, Hd\}$. It contains four FlexLinear modules as learnable components, and thus its scalability with respect to width follows a quadratic relationship.

Besides normal attention, we adopt fusion attention module [40] in this work. We also propose a similar *flexible fusion attention*, specially designed for sequence features fusion, e.g. speech processing. Fusion attention is primarily used to aggregate sequence features, reducing a sequence of shape $T \times D$ into a single vector of shape D. As shown in Fig. 3, the only difference between FlexFusionAttention and standard FlexAttention is the removal of $\mathbf{W}^q \in \mathbb{R}^{\bar{H}d \times \bar{H}d}$, replaced by a learnable query vector $\mathbf{q} \in \mathbb{R}^d$. It skip the $\mathbf{W}^q$ projection in Eq. (3), and directly apply $\mathbf{q}$ in a variant of Eq. (4):

$$\text{Attention}(\mathbf{q}, \mathbf{K_h}, \mathbf{V_h}) = \text{Softmax}\left(\frac{\mathbf{q}\mathbf{K_h}^\top}{\sqrt{d}}\right)\mathbf{V_h} \in \mathbb{R}^d, \tag{6}$$

The concatenated features from the $\bar{H}$ heads then form a single vector of feature dimension $\bar{D}$.

3.2.3 Flexible normalization

RMSNorm [41] is a normalization layer widely adopted in recent prominent transformers such as LLaMA [2], Qwen [8], and DeepSeek [10]. We propose a flexible version of RMSNorm, named **FlexRMSNorm**, which adaptively adjusts the number of normalized dimensions via flexible weighted vector $\mathbf{w} \in \mathbb{R}^D$. Given an input tensor $\bar{\mathbf{x}} \in \mathbb{R}^{\bar{D}}, \bar{D} \le D$, the FlexRMSNorm computes:

$$\text{FlexRMSNorm}(\bar{\mathbf{x}}) = \frac{\bar{\mathbf{x}}}{\sqrt{\frac{1}{\bar{D}}\sum_{i=1}^{\bar{D}} x_i^2 + \varepsilon}} \odot \bar{\mathbf{w}} \in \mathbb{R}^{\bar{D}} \tag{7}$$

where $\bar{\mathbf{w}}$ is a sliced version of $\mathbf{w}$ that retains only the first $\bar{D}$ elements, $\odot$ denotes the Hadamard product, and ε is a small constant to prevent the denominator being zero.

3.3 Early-exit for adjustable depth

This section describes early-exit technique for controlling network depth. Figure 4 shows an example of early-exit, where $B = 3$ and selected $\bar{B} = 2$. As shown in Sect. 3.1, the output of each residual block shares the same shape. This makes it straightforward to use a shared decoder f_{dec} to decode $Z^{\bar{B}}$, which can be formulated as:

$$X = f_{\text{dec}}(Z^{\bar{B}}) \tag{8}$$

We briefly analyze the complexity of the subnetwork obtained via early-exit, along with the underlying

principle that makes early-exit effective. Because most of the computational cost spends in the residual blocks, both the parameter count and MACs scale down proportionally to $\bar{B}/B$. This strategy typically enables an order-of-magnitude scalability. According to the principle of residual connection [38], i.e. $Z^b = Z^{b-1} + Z^{(b)}$, we can derive:

$$Z^B = Z^0 + \sum_{b=1}^{B} Z^{(b)} \tag{9}$$

Therefore, decoding the full network output $f_{\text{dec}}(Z^B)$ can be seen as decoding the fusion of features across all blocks. In general, lower-layer features capture the main content, while higher-layer features refine the details. Early-exit effectively forces the high-layer neurons to deactivate, sacrificing some fine-grained improvements to save computation.

3.4 Efficient training strategy for flexible networks

As described above, we control the network depth by adjusting $\bar{B}$, and control the network width by adjusting $\bar{H}$. During training, $\bar{B}$ and $\bar{H}$ are randomly sampled at each step. For every batch, we duplicate it: one copy is used to train a randomly sampled sub-network, while the other is used to train the full network. This ensures coverage of various sub-networks while guaranteeing that each neuron is updated at least once per step. However, naively sampling sub-networks can lead to a bottleneck in distributed training. Specifically, GPUs assigned smaller sub-networks may finish early and have to wait for others, causing inefficiency. To address this, we use an independent random generator at each training step, with the step index as its seed. This generator pseudo-randomly samples a sub-network index i_{net} from $\{0, \ldots, BH-1\}$, ensuring that all GPUs train the same sub-network in each step. This index determines the depth and width of the current sub-network:

$$\bar{B} = \left\lfloor \frac{i_{\text{net}}}{B} \right\rfloor + 1, \quad \bar{H} = i_{\text{net}} \bmod B + 1 \tag{10}$$

4 Unified architecture for speech enhancement

4.1 Backbone

The backbone follows the BS-RoFormer [42], which is also the winner system of the NeurIPS'24 SE Challenge [43]. It adopts RoFormer for sequence modeling, where rotary position embedding [44] is applied to inject positional information into attention.

First, the frequency axis F is split into K sub-bands before encoding. Details of the sub-band splitting and sampling-rate-agnostic design are provided in Sect. 4.2. The full f_{enc} consists of K parallel FlexRMSNorm and FlexLinear layers, each with $\bar{D}$ output neurons. These layers map sub-bands of different bandwidths into embeddings of the same shape, $Z_Y^k \in \mathbb{R}^{T\times\bar{D}}$, which are then stacked together. Let the bandwidth of the k-th sub-band be w_k. The k-th FlexLinear takes $3w_k$ input neurons, corresponding to the real part Y_r, imaginary part Y_i, and log-magnitude $\log|Y|$.

For residual blocks, each residual block contains two Transformers that perform dual-path sequence modeling along the time dimension T and frequency dimension K, respectively. Our Transformer module design closely follows [42], except that we replace all linear, Attention, and RMSNorm layers with our flexible counterparts. We remove the use of random Dropout, since employing sub-networks already introduces deactivation. For Transformers modeling along the time dimension, we enable the causal mask as defined in Eq. (4), ensuring that the entire network is causal.

The design of f_{dec} is almost identical to [42], except that the first linear layers are replaced with FlexLinear layers with $\bar{D}$ input neurons. In addition, we adopt a complex-mapping strategy for output, instead of complex-masking. This is also mathematically considered for packet loss.

4.2 Band split scheme

To make the model sampling-rate-agnostic, we assume F and K correspond to a high sampling rate of 48 kHz. When processing audio with a lower sampling rate, applying the STFT of same resolution yields $Y \in \mathbb{C}^{T\times\bar{F}}$ with $\bar{F} < F$. Accordingly, we use only $\bar{K} < K$ sub-bands for sequence modeling, which reduces costs.

We consider seven common sampling rates: $\{8, 16, 22.05, 24, 32, 44.1, 48\}$ kHz. Our goal is to split bands as uniformly as possible on the Mel scale, while ensuring that $\bar{K}$ aligns exactly with the effective frequency ranges $\{4, 8, 11.025, 12, 16, 22.05, 24\}$ kHz for these sampling rates. To achieve this, we first divide 24000 Hz into 41 equal parts in Mel scale. We find that the numbers of sub-bands corresponding to the seven effective frequencies above are approximately $\{22, 29, 32, 33, 36, 40, 41\}$. Based on this, we design a two-stage band split strategy: first, we divide the full band into seven coarse bands corresponding to the seven effective frequencies; then, each coarse band is further split uniformly on the Mel scale into $\{22, 7, 3, 1, 3, 4, 1\}$ fine sub-bands, respectively, summing to a total of 41. As shown in Fig. 5, this design yields 41

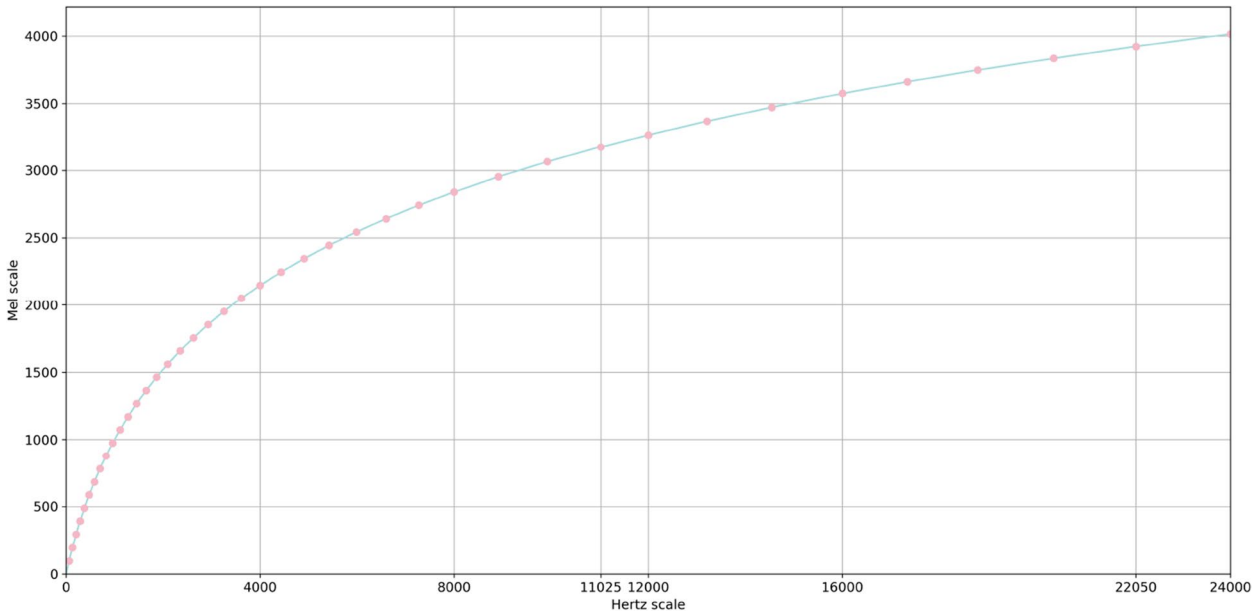


Fig. 5 Divide the subbands evenly on the Mel scale. The line indicates the Hz-to-Mel frequency mapping, and points mark the sub-band boundaries

sub-bands that are approximately uniform on the Mel scale and enables sampling-rate-agnostic inference.

4.3 Efficient training strategy for multi-sampling-rate

As shown in Sect. 4.2, the dataset supports seven sampling rates: {8, 16, 22.05, 24, 32, 44.1, 48} kHz. This presents an opportunity for optimization during training. If audio samples are randomly selected and resampled to 48 kHz to form a mini-batch, it clearly results in unnecessary computational overhead. Even with a batch size of 1, under multi-GPU training, GPUs processing lower sampling rates finish earlier and then idle while waiting for those handling higher sampling rates—leading to a bottleneck.

To address this challenge, we propose a Distributed Grouped Sampler that enforces uniform sampling rates within each training batch across all GPUs. The algorithm consists of three phases:

Phase 1: Sample Grouping. At initialization, the dataset is traversed to group sample indices by their native sampling rates, forming seven groups corresponding to the supported rates {8, 16, 22.05, 24, 32, 44.1, 48} kHz.

Phase 2: Distributed Batch Formation. At the beginning of each epoch, samples in each group are shuffled. The groups are then evenly partitioned across GPUs using round-robin assignment. Batches are formed exclusively within each group to ensure sampling rate consistency.

Phase 3: Sequential Batch Yielding. Batches are yielded in a round-robin manner across groups. Depleted groups are skipped until all groups are exhausted, marking the end of the epoch. This design ensures synchronized GPU workloads and avoids idle time caused by sampling rate mismatch.

This strategy eliminates redundant resampling and balances computational load across GPUs. The number of batches per epoch is given by $\sum_{sr} \lfloor \frac{|G_{sr}|}{B_a \times N_{gpu}} \rfloor$, where $|G_{sr}|$ is the number of samples in group sr, B_a is the batch size, and N_{gpu} is the number of GPUs.

4.4 VAD decoder

As shown in Fig. 6, we incorporate voice activity detection (VAD) [45] as an auxiliary task to enhance the performance of speech enhancement (SE). To this end, we design a lightweight VAD decoder, denoted as f_{vad}, which imposes negligible computational overhead.

After obtaining $Z^B \in \mathbb{R}^{T \times K \times \bar{D}}$, it is fed into the VAD decoder f_{vad} to produce the VAD prediction $\hat{\mathbf{v}} \in [0,1]^T$. The VAD decoder is implemented using FlexFusionAttention. Specifically, a FlexFusionAttention layer first aggregates information along the K dimension, treating it as the sequence length, yielding $V \in \mathbb{R}^{T \times \bar{D}}$. A subsequent FlexLinear layer with sigmoid activation maps V to the final prediction $\hat{\mathbf{v}} \in [0,1]^T$.

4.5 Loss function

The loss function is designed to optimize complex-spectrogram reconstruction, magnitude-spectrogram

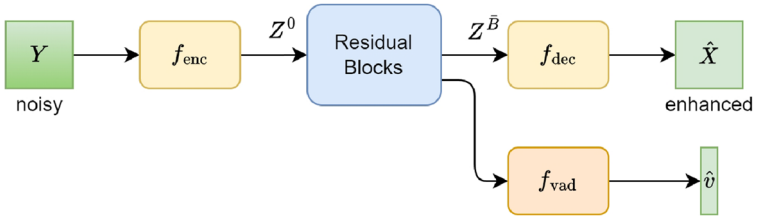


Fig. 6 Illustration of the unified architecture for speech enhancement

reconstruction [14], and VAD [45], where the latter two serve as auxiliary tasks to enhance the overall performance.

Given the clean complex-spectrogram of the target speaker $X \in \mathbb{C}^{T\times F}$, the corresponding VAD labels are represented as a binary vector $\mathbf{v} \in \{0,1\}^T$, obtained via a simple energy thresholding method. VAD is implemented using a lightweight module, as detailed in Sect. 4.4. Let the VAD prediction is $\hat{\mathbf{v}} \in [0,1]^T$. The loss function is defined as:

$$\mathcal{L} = \alpha\left(\left|\hat{X}_r - X_r\right|_1 + \left|\hat{X}_i - X_i\right|_1 + \left||\hat{X}| - |X|\right|_1\right) + \beta \sum_{t=1}^{T}\left(-\nu_t \log \hat{\nu}_t - (1-\nu_t)\log(1-\hat{\nu}_t)\right) \tag{11}$$

where X_r and X_i denote the real and imaginary parts of X, respectively. In this paper, we set $\alpha = 1/3$ and $\beta = 1/10$.

5 Experiments

5.1 Experimental setup

5.1.1 Datasets

For the training data, we perform dynamic data augmentation using clean speech, noise, and room impulse responses (RIRs). We collect clean speech from DNS5 Speech [46], WSJ [47], LibriTTS [48], VCTK [49], and EARS [50], totaling approximately 800 hours of raw audio. We clean DNS5 Speech and WSJ using DNSMOS [51] by retaining only samples with an overall score above 3.0, and further deduplicate DNS5 Speech. This results in about 400k clean utterances, with an average duration of 4 seconds per sample, which amounts to 440 hour clean speech. In addition, many audio samples in DNS5 Speech, WSJ, and LibriTTS exhibit artificially high sampling rates. To address this, we estimate the actual bandwidth of each sample by identifying the sub-band whose energy is 60dB below the peak. We then select the lowest sampling rate that sufficiently covers this effective bandwidth.

For noise, we compile samples from DNS5 Noise [46], WHAM! [52], FSD50K [53], and ESC-50 [54]. After removing samples with prominent speech content, we obtain around 90,000 noise clips.

For RIRs, we simulate 1.44 million RIRs using Pyroomacoustics [55], with reverberation time $T_{60} \in [0.03, 2.98]$ seconds. We categorize rooms into three types: small, medium, and large. For each category, we simulate 200 rooms. In every room, 24 microphones are randomly placed, and RIRs are generated for 100 different source positions with respect to this microphone array. This results in a total of $3 \times 200 \times 24 \times 100 = 1,440,000$ unique RIRs. For all room types, the room height is sampled uniformly from 2 m to 5 m, and the wall absorption coefficient is sampled uniformly from 0.2 to 0.8. The room length and width are sampled as follows: Small rooms: 3 m to 10 m. Medium rooms: 10 m to 30 m. Large rooms: 30 m to 50 m. This setting yields RIRs corresponding to a wide range of reverberation strengths.

We evaluate the denoising and dereverberation capabilities of SEFlow using the blind test set (with reverberation) from the INTERSPEECH 2020 DNS Challenge. For evaluating the packet loss concealment (PLC) ability of SEFlow, we use the blind test set from the INTERSPEECH 2022 Audio Deep PLC Challenge. As there is no established benchmark for the declipping task, we instead showcase its performance in our demos. Additionally, for ablation studies, we trained toy models using the training, validation, and test splits of VCTK [49].

5.1.2 Dataset simulation

When adding noise, the signal-to-noise ratio (SNR) is uniformly sampled from $[-5, 20]$ dB. When adding noise, there is a 50% probability that RIRs will be used to generate point source noise. Clipping is performed by flattening waveform exceeding the 90%-percentile amplitude. Packet loss is simulated using a Markov chain with stay and transition probabilities of 0.95 and

0.05, respectively, and the maximum consecutive loss duration is randomly sampled from 1 to 10 packets, with each packet being 10 ms.

All models are trained and tested with randomized combinations: noise, reverberation, clipping, and packet loss are applied with probabilities 1.0, 0.5, 0.3, and 0.3, respectively, in any arbitrary combination.

5.1.3 Model configuration

The short term Fourier transform (STFT) is computed with a window length of 32 ms and a hop length of 16 ms. The number of discrete Fourier transform (DFT) points is set equal to the window length in samples, and thus depends on the sampling rate. For the full SEFlow, we set the network depth $B = 12$, width $D = 256$, and number of attention heads $H = 4$. For the toy models, $B = 6$, $D = 192$ and $H = 4$.

Under these settings, the parameter count of the full network and the smallest sub-network are 27.19M and 1.69M, respectively. For a 16 kHz audio input, the corresponding computational cost is 24.71 GMACs/s and 0.19 GMACs/s, differing by two orders of magnitude. Based on the above configuration, we set the input audio duration to 4 seconds per sample during training. With a batch size of 2 and a single RTX 4090 GPU, the training pipeline processes approximately 2 batches per second on average.

5.1.4 Evaluation metrics

We adopt both non-intrusive and intrusive evaluation metrics. The non-intrusive metrics are DNSMOS [51] and PLCMOS [56]. DNSMOS includes speech signal quality (SIG), background noise quality (BAK), and overall quality (OVL). Intrusive metrics include Signal-to-Distortion Ratio (SDR), Scale-Invariant SDR (SI-SDR), Perceptual Evaluation of Speech Quality (PESQ) [57], and Extended Short-Time Objective Intelligibility (STOI) [58]. For downstream evaluations, we employ RawNet3 [59] as the speaker recognition model to compute speaker cosine similarity (SpkSim), and Whisper-base [60] as the automatic speech recognition (ASR) model to obtain word accuracy (WAcc).

5.2 Comparison results

As shown in Table 1, the large version of SEFlow achieves performance comparable to recent state-of-the-art models on denoising tasks. The current backbone design of SEFlow tends to favor noise suppression, occasionally resulting in slight speech over-suppression. Nevertheless, the models with computational complexity under 1G MACs/s can support real-time streaming application on resource-constrained devices such as headphones or smartphones. Even with an extremely lightweight configuration—using only a single layer and a single attention head—SEFlow achieves less than 200M MACs/s while still significantly outperforming the unprocessed noisy input in terms of OVRL. As shown in Table 2, as for PLC, a task that is inherently more distinct from others, SEFlow's performance with minimal configurations is relatively weaker. At least two blocks are required to achieve usable performance. However, when scaled to its full capacity, SEFlow again reaches performance comparable to recent state-of-the-art PLC models.

Table 1 DNSMOS SIG, BAK, and OVL scores on the Interspeech 2020 DNS Challenge blind test set (reverb version). The sampling rate is 16kHz

Method	Depth($\bar{B}$)-width($\bar{H}$)	Complexity	MACs(G)	Non-intrusive SE		
		Param(M)		SIG	BAK	OVL
Noisy	/	/	/	1.76	1.50	1.39
TasNet [61]	/	5.10	/	2.42	2.71	2.01
FRCRN [62]	/	6.90	/	2.93	2.92	2.28
Voicefixer [63]	/	111.70	/	3.07	3.72	2.67
SELM [64]	/	/	/	3.16	3.58	2.70
MaskSR [65]	/	361.80	/	3.40	4.04	3.09
GenSE [66]	/	/	/	3.49	3.73	3.19
AnyEnhance [67]	/	363.60	/	3.50	4.04	3.20
SEFlow	12-4	27.19	24.71	2.79	3.76	2.30
	9-3	14.24	10.69	2.80	3.68	2.32
	6-2	6.18	3.33	2.20	3.90	2.00
	3-1	1.83	0.48	1.84	4.00	1.76
	2-1	1.76	0.34	1.59	3.87	1.67
	1-1	1.69	0.19	1.51	3.95	1.63

Table 2 DNSMOS OVL and PLCMOS scores on Interspeech 2022 PLC-challenge blind test set. The sampling rate is 48kHz

Method	Depth($\bar{B}$)-width($\bar{H}$)	Complexity	MACs(G)	Non-intrusive SE	
		Param(M)		OVL	PLCMOS
Lossy	/	/	/	2.56	2.90
LPCNet [68]	/	/	/	3.09	3.74
PLCNet [69]	/	/	/	/	3.83
BS-PLCNet [70]	/	/	/	3.20	4.29
SEFlow	12-4	27.19	35.26	3.19	3.75
	9-3	14.24	15.31	3.13	3.64
	6-2	6.18	4.81	3.05	3.63
	3-1	1.83	0.73	2.94	3.63
	2-1	1.76	0.52	2.86	3.34
	1-1	1.69	0.31	2.41	2.11

Table 3 Comparison of training strategies under different conditions, where the training time is measured in hours

	Flexible architecture		Multi sample rate	
	GMACs/s	Training time	GMACs/s	Training time
Vanilla	24.46	9.07	35.26	9.07
Efficient	15.20	5.90	29.34	7.71

Table 3 lists the average MACs and training time per epoch required for different training strategies. We first focus on flexible sub-networks, where the sampling rate is fixed at 48kHz to simplify MACs calculation. When both B and H are sampled independently on each GPU, the MACs for a training step are determined by the largest sub-network among all GPUs, resulting in an average of 24.46GMACs. In contrast, synchronizing the sampled sub-network across all GPUs reduces the average to 15.20GMACs, saving 37.86% of the computation. Meanwhile, the training time is also reduced by 34.95%. Next, we examine the impact of multi-sampling-rate training, using the full network to simplify MACs calculation. In the vanilla setting, all samples are upsampled to 48kHz, resulting in 35.26GMACs. By ensuring a consistent sampling rate within each step, the average drops to 29.34GMACs, achieving a 16.79% reduction in computation. Meanwhile, the training time is also reduced by 14.99%.

5.3 Ablation studys

Table 4 presents the impact of different loss functions on model performance, all trained using the full network. The "Wav" loss denotes the L1 loss on time-domain waveform, as used in BS-RoFormer [42]. It can be observed that this loss leads to degraded performance on most metrics. In contrast, using the "Mag" loss [14]—i.e., applying L1 loss on the magnitude spectrum—results in lower SDR and SI-SDR, but significantly improves PESQ, STOI, and downstream task performance. Furthermore, incorporating our proposed VAD module as an auxiliary task consistently boosts performance across almost all metrics.

Table 5 presents the impact of varying depth and width on performance. As expected, adjusting depth leads to approximately linear changes in MACs. Remarkably, even when reduced to a single block, the model still provides noticeable enhancement for speech under compound distortions. In contrast, varying width leads to quadratic changes in MACs. When the width is reduced to one-quarter, the computational cost drops to only 8% of the full model. An interesting observation is that the "6-1" configuration consistently outperforms the early-exit "1–4" variant across all metrics, despite using only 37% of the MACs. Specifically, it achieves relative improvements of 12.3%, 8.6%, 15.4%, 38.3%, 47.7%, 24.8%, 12.0%, 7.4%, and 17.8% across the evaluated metrics. Compared to the noisy input, even with just 8% of the MACs, the model yields substantial enhancement. However, comparisons among the three "6-4" variants indicate that models trained under flexible configurations tend to underperform the full fixed-scale model. This performance gap is

Table 4 Ablation studys on loss functions

Loss	Non-intrusive SE			Intrusive SE				Downstream	
	SIG	BAK	OVL	SDR	SISDR	PESQ	STOI	SpkSim	WAcc
clean	3.317	3.894	2.987	∞	∞	4.500	1.000	100.00	90.56
noisy	2.299	2.070	1.762	4.623	3.309	1.300	0.523	57.27	54.13
STFT	3.006	3.900	2.724	12.271	11.616	2.077	0.686	64.68	63.89
STFT+Wav	3.023	3.899	2.738	11.881	11.499	2.067	0.688	63.87	61.50
STFT+Mag+Wav	3.050	3.882	2.754	11.172	10.891	2.258	0.698	72.63	66.13
STFT+Mag	3.056	3.882	2.757	11.630	10.860	2.327	0.701	73.83	65.76
STFT+Mag+VAD	3.065	3.887	2.768	11.856	11.185	2.323	0.703	74.09	67.63

Table 5 Ablation studies on depth and width scalability. "Full" refers to training the complete network without any scalability. The units of Param and MACs are M and G, respectively

	$\bar{B}$-$\bar{H}$	Computational		Non-intrusive SE			Intrusive SE				Downstream	
		Param	MACs	SIG	BAK	OVL	SDR	SISDR	PESQ	STOI	SpkSim	WAcc
clean	/	/	/	3.317	3.894	2.987	∞	∞	4.500	1.000	100.00	90.56
noisy	/	/	/	2.299	2.070	1.762	4.623	3.309	1.300	0.523	57.27	54.13
$\bar{B}$	1–4	9.50	2.20	2.477	3.522	2.171	7.227	6.182	1.459	0.550	59.66	45.46
	2–4	10.09	3.84	2.728	3.744	2.435	9.653	8.878	1.771	0.617	65.17	54.75
	3–4	10.68	5.48	2.864	3.800	2.566	10.653	9.964	1.957	0.650	68.90	60.16
	4-4	11.28	7.11	2.941	3.843	2.646	11.186	10.521	2.101	0.671	70.28	62.74
	5-4	11.87	8.75	2.975	3.866	2.685	11.544	10.885	2.187	0.683	71.25	64.72
	6-4	12.46	10.39	3.012	3.878	2.720	11.808	11.168	2.226	0.691	71.13	65.73
$\bar{H}$	6-1	1.32	0.82	2.781	3.826	2.505	9.996	9.130	1.821	0.616	64.05	53.54
	6-2	3.84	2.82	2.936	3.855	2.646	11.168	10.434	2.049	0.663	69.76	59.70
	6-3	7.55	6.01	2.985	3.863	2.691	11.625	10.887	2.161	0.679	71.28	63.88
	6-4	12.46	10.39	2.992	3.866	2.698	11.774	10.950	2.185	0.682	71.05	62.92
full	6-4	12.46	10.39	3.065	3.887	2.768	11.856	11.185	2.323	0.703	74.09	67.63

Table 6 Results on denoising task

$\bar{B}$-$\bar{H}$		Computational		Non-intrusive SE			Intrusive SE				Downstream	
		Param	MACs	SIG	BAK	OVL	SDR	SISDR	PESQ	STOI	SpkSim	WAcc
clean	/	/	/	3.475	4.014	3.180	∞	∞	4.500	1.000	100.00	92.59
noisy	/	/	/	3.059	2.388	2.241	8.340	8.339	1.507	0.627	83.21	80.04
full	1-1	1.13	0.19	2.907	3.805	2.610	13.347	13.101	1.815	0.645	73.50	72.21
flex				2.769	3.776	2.487	12.371	12.079	1.667	0.625	70.30	69.89
full	2-1	1.17	0.32	3.036	3.879	2.744	14.588	14.350	2.008	0.670	76.06	73.71
flex				2.938	3.834	2.649	13.683	13.406	1.877	0.654	74.83	72.75
full	3-1	1.21	0.44	3.106	3.896	2.807	15.094	14.931	2.106	0.687	77.66	75.97
flex				3.049	3.879	2.754	14.445	14.217	2.009	0.672	76.55	74.71
full	4-2	3.54	1.94	3.217	3.951	2.926	16.990	16.998	2.474	0.735	81.41	79.62
flex				3.145	3.929	2.856	16.271	16.088	2.291	0.714	79.42	78.47
full	5-3	7.22	5.07	3.266	3.976	2.979	17.977	17.924	2.719	0.757	82.95	81.57
flex				3.217	3.960	2.931	17.492	17.348	2.529	0.743	80.66	80.33
full	6-4	12.46	10.39	3.350	4.009	3.069	18.349	18.310	2.973	0.787	85.95	85.55
flex				3.311	3.988	3.024	18.992	18.895	2.843	0.774	83.92	83.75

acceptable given the flexibility gained and also suggests potential for future improvement.

In addiction, we separately compare the tasks of denoising, declipping, and packet loss concealment, which differ in both prevalence and difficulty. Note that the results of "clean" in Tables 6, 7, and 8 vary slightly due to minor differences in magnitude when preparing each test set.

We start with the denoising task, which is both the most common and relatively easy. As shown in Table 6, for each group of models with the same architecture, SEFlow performs only slightly worse than individually trained networks. Even when SEFlow is reduced to a single residual block ($B = 1$) with a single used head ($H = 1$)—resulting in as few as 0.19 G MACs/s—it still proves effective, showing improvements in both SDR and SI-SDR compared to the noisy input. The used ASV and ASR models are trained on noisy data and are inherently robust. To further boost downstream performance, a larger network with at least 5 residual blocks is required.

Next is the declipping task. As discussed in the main paper, declipping in the time-frequency domain

Table 7 Results on declipping task

	$\tilde{B}$-$\tilde{H}$	Computational		Non-intrusive SE			Intrusive SE				Downstream	
		Param	MACs	SIG	BAK	OVL	SDR	SISDR	PESQ	STOI	SpkSim	WAcc
clean	/	/	/	3.354	3.815	2.999	∞	∞	4.500	1.000	100.00	91.32
noisy	/	/	/	3.497	3.732	3.075	3.668	4.533	1.704	0.804	61.69	79.75
full	1-1	1.13	0.19	3.083	3.830	2.763	8.709	8.315	2.157	0.666	53.87	72.77
flex				3.029	3.835	2.721	7.856	7.817	2.023	0.654	55.33	71.47
full	2-1	1.17	0.32	3.129	3.801	2.793	9.621	9.296	2.409	0.715	58.06	75.58
flex				3.131	3.835	2.807	9.111	9.067	2.253	0.697	56.70	74.51
full	3-1	1.21	0.44	3.176	3.808	2.835	9.930	9.609	2.546	0.755	62.88	77.28
flex				3.153	3.823	2.822	9.802	9.514	2.419	0.722	59.99	75.74
full	4-2	3.54	1.94	3.273	3.820	2.929	11.199	10.950	3.012	0.836	70.27	82.10
flex				3.240	3.829	2.903	10.712	10.438	2.746	0.796	67.69	80.16
full	5-3	7.22	5.07	3.295	3.827	2.951	11.995	11.816	3.227	0.855	75.06	84.82
flex				3.276	3.826	2.933	11.474	11.254	3.025	0.835	72.15	81.70
full	6-4	12.46	10.39	3.371	3.858	3.030	13.485	13.524	3.519	0.910	81.76	86.43
flex				3.323	3.832	2.977	12.903	12.778	3.407	0.879	78.87	85.88

Table 8 Results on packet loss concealment task

Method B-H		Computational		Non-intrusive SE		Intrusive SE				Downstream	
		Param	MACs	OVL	PLCMOS	SDR	SISDR	PESQ	STOI	SpkSim	WAcc
clean	/	/	/	3.147	4.456	∞	∞	4.500	1.000	100.00	92.01
lossy	/	/	/	2.372	2.688	8.000	6.884	1.431	0.772	69.66	79.91
full	1-1	1.13	0.19	2.593	3.457	7.704	6.573	1.482	0.643	80.05	69.60
flex				2.536	3.269	7.608	6.449	1.427	0.637	78.48	70.11
full	2-1	1.17	0.32	2.680	3.528	7.826	6.699	1.628	0.673	85.04	67.68
flex				2.653	3.411	7.714	6.571	1.529	0.664	83.07	68.56
full	3-1	1.21	0.44	2.741	3.647	8.025	6.937	1.714	0.706	86.41	68.33
flex				2.714	3.465	7.809	6.680	1.644	0.688	85.15	68.34
full	4-2	3.54	1.94	2.850	3.746	8.862	7.952	1.920	0.770	89.04	71.30
flex				2.807	3.689	8.380	7.342	1.864	0.749	89.05	70.48
full	5-3	7.22	5.07	2.894	3.828	9.438	8.601	2.049	0.783	90.62	74.09
flex				2.853	3.818	8.803	7.841	1.985	0.773	90.32	73.11
full	6-4	12.46	10.39	2.956	3.923	9.817	9.033	2.223	0.817	91.90	76.56
flex				2.936	3.929	9.933	9.144	2.202	0.799	91.39	76.45

essentially removes artifacts, making it similar to denoising. As shown in Table 7, the non-intrusive SE metrics, appear unsuitable for this task, as they show no significant difference between clean and noisy signals. For each group of models with the same architecture, flex version performs only slightly worse than full trained networks. Even when the flex version is reduced to a single residual block ($B = 1$) with a single used head ($H = 1$)—resulting in only 0.19 G MACs/s—it still delivers clear improvements in SDR, SI-SDR, and PESQ over the noisy input. To further enhance downstream performance, a larger network with at least 4 residual blocks is needed.

Finally, we consider the packet loss concealment (PLC) task, with a focus on PLCMOS, a metric specifically designed to evaluate PLC performance. As shown in Table 8, for each group of models with the same architecture, flex version performs only slightly worse than full trained networks. Even when the flex version is reduced to a single residual block ($B = 1$) with a single used head ($H = 1$)—resulting in only 0.19 G MACs/s—it still delivers clear improvements in both PLCMOS and SpkSim compared to the lossy input.

For downstream tasks, we observe that SpkSim is quite sensitive to packet loss, whose utterance-level

embeddings are derived from frame-level fusion. Even so, our SEFlow, including the smallest variant, is able to provide significant restoration. In contrast, ASR shows poor performance across both individually trained networks and SEFlow. This points to limitations in the current backbone or loss design and suggests that ASR-specific fine-tuning may be necessary for improvement.

5.4 Single case analysis

We analyze the impact of sub-network size on performance using an audio sample with strong noise and reverberation. As shown in Fig. 7, the speech is almost completely masked by noise. Even the smallest sub-network, "3-1", manages to remove most of the noise, though it suffers from noticeable speech over-suppression. As both width and depth increase, the over-suppression gradually diminishes, and the enhanced outputs become increasingly closer to the clean reference. More demonstrations are available on the project homepage.

6 Discussion

In this section, we discuss several important aspects of the proposed architecture, including potential additional overheads and the environmental friendliness of such adaptive models.

6.1 Potential overheads

There are two main types of potential overhead associated with such dynamic mechanisms. First, training complexity: To support multi-size adaptation, it is typically necessary to train multiple sub-networks simultaneously. This increases memory usage and computational load during training, as gradient updates and parameter synchronization across different sub-network branches impose additional burdens. Second, inference-time decision/scheduling overhead: Dynamically selecting sub-network widths based on inputs or resource constraints requires extra decision-making logic (e.g., how to quickly match the optimal sub-network). Thus, how to dynamically select the optimal sub-network remains a worthwhile research direction.

6.2 Environmental friendliness

Such adaptive models hold significant value for the environmental friendliness of edge deployments. Traditional fixed-structure models operate at peak computational capacity regardless of input complexity, causing even simple samples (e.g., low-noise samples) to trigger full-depth/width inference. This results in unnecessary energy consumption for edge devices (such as IoT sensors and mobile terminals)—devices that often rely on battery power or connect to low-efficiency power grids, where redundant energy use directly exacerbates carbon emissions.

The adaptive mechanisms of our model can specifically alleviate this issue by reducing MACs and memory accesses—two primary sources of energy consumption in edge AI chips. While the current study has not conducted dedicated energy consumption tests, the inherent advantages of adaptive mechanisms and the practical demands of edge deployment highlight its potential for improving energy efficiency and reducing carbon footprints, offering directions for subsequent optimizations targeting environmental goals.

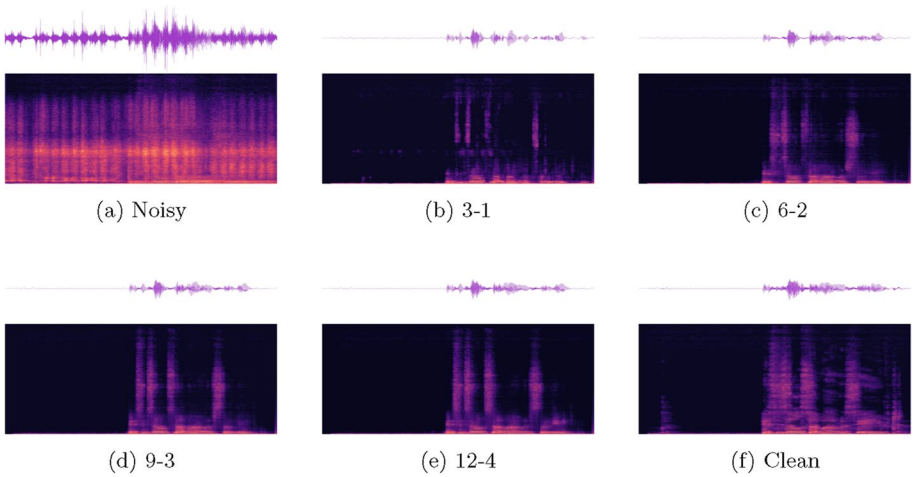


Fig. 7 Qualitative comparison of spectrograms from different subnetwork configurations. From left to right and top to bottom: **a** the noisy input, **b**-**e** enhanced outputs from subnetworks with increasing width and depth settings: (B=3, H=1), (B=6, H=2), (B=9, H=3), and (B=12, H=4), and **f** the clean reference

7 Conclusion

We have presented a flexible and unified architecture that advances the foundational infrastructure for AI Flow in multi-task speech enhancement scenarios. To enable seamless intelligence flow across the device-edge-cloud continuum, we propose a set of flexible modules—FlexAttention and FlexRMSNorm, facilitating adaptive resource utilization in cooperative inference systems. Combined with early-exit strategies, the resulting sub-networks can be dynamically deployed across heterogeneous computational nodes with varying budgets. This architecture exemplifies the AI Flow paradigm by supporting multiple speech enhancement tasks—including denoising, dereverberation, declipping, and packet loss concealment—through a single family-model system that adapts to diverse network conditions and computational constraints. The proposed system achieves strong performance on each individual task while maintaining the flexibility essential for cooperative edge intelligence. We think these designs establish crucial building blocks for future AI Flow implementations, inspiring research on scalable family-model architectures that enable ubiquitous speech enhancement services across distributed computing environments.

Abbreviations

DNN	Deep neural networks
SE	Speech Enhancement
MACs	Multiply–accumulate operations
GMACs/s	Billion multiply–accumulate operations per second
TF	Time-frequency
PLC	Packet loss concealment
VAD	Voice activity detection
RIR	Room impulse response
STFT	Short term Fourier transform
DFT	Discrete Fourier transform
SIG	Speech signal quality
BAK	Background noise quality
OVL	Overall quality
SNR	Signal-to-noise ratio
SDR	Signal-to-distortion ratio
SI-SDR	Scale-invariant signal-to-distortion ratio
PESQ	Perceptual evaluation of speech quality
STOI	Extended short-time objective intelligibility
SpkSim	Speaker cosine similarity
WAcc	Word accuracy
ASR	Automatic speech recognition
Param	The number of parameters

Acknowledgements

None.

Code availability

Code will be made available on reasonable request.

Authors' contributions

Linfeng Feng conceived the study, conducted the experiments, and led the manuscript writing. Xiao-Lei Zhang proposed the research direction and contributed to the writing and revision. Chi Zhang provided critical revision. All authors reviewed and approved the final manuscript for submission.

Funding

None.

Data availability

All data used is publicly available.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 1 August 2025 Revised: 22 October 2025 Accepted: 24 October 2025
Published online: 24 November 2025

References

1. X. Li, Positive-incentive noise. IEEE Trans. Neural Netw. Learn. Syst. **35**(6), 8708–8714 (2024)
2. H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., Llama: Open and efficient foundation language models (2023). arXiv preprint arXiv:2302.13971
3. H. Zhang, S. Huang, Y. Guo, X. Li, Variational positive-incentive noise: how noise benefits models. IEEE Trans. Pattern. Anal. Mach. Intell. (2025). https://doi.org/10.1109/TPAMI.2025.3575295
4. A. Polyak, A. Zohar, A. Brown, A. Tjandra, A. Sinha, A. Lee, A. Vyas, B. Shi, C.Y. Ma, C.Y. Chuang, et al., Movie gen: A cast of media foundation models (2024). arXiv preprint arXiv:2410.13720
5. H. Zhang, Y. Xu, S. Huang, X. Li, Data augmentation of contrastive learning is estimating positive-incentive noise (2024). arXiv preprint arXiv:2408.09929
6. J. Shao, X. Li, Ai flow at the network edge. IEEE Netw. 1 (2025). https://doi.org/10.1109/MNET.2025.3541208
7. H. An, S. Huang, S. Huang, R. Li, Y. Liang, J. Shao, Z. Wang, C. Yuan, C. Zhang, H. Zhang, et al., Ai flow: Perspectives, scenarios, and approaches (2025). arXiv preprint arXiv:2506.12479
8. A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, et al., Qwen2. 5 technical report. (2024). arXiv preprint arXiv:2412.15115
9. S. Huang, H. Zhang, X. Li, Enhance vision-language alignment with noise. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 16, (AAAI Press, Washington, DC, 2025) pp. 17449–17457
10. A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al., Deepseek-v3 technical report (2024). arXiv preprint arXiv:2412.19437
11. D. Tao, M. Song, X. Li, J. Shen, J. Sun, X. Wu, C. Faloutsos, S.J. Maybank, Bayesian tensor approach for 3-d face modeling. IEEE Trans. Circuits Syst. Video Technol. **18**(10), 1397–1410 (2008)
12. J. Han, D. Zhang, S. Wen, L. Guo, T. Liu, X. Li, Two-stage learning to predict human eye fixations via sdaes. IEEE Trans. Cybern. **46**(2), 487–498 (2015)
13. K. Jiang, Z. Shi, D. Zhang, H. Zhang, X. Li, Mixture of noise for pre-trained model-based class-incremental learning (2025). arXiv preprint arXiv:2509.16738
14. Z.Q. Wang, S. Cornell, S. Choi, Y. Lee, B.Y. Kim, S. Watanabe, Tf-gridnet: integrating full-and sub-band modeling for speech separation. IEEE/ACM Trans. Audio Speech Lang. Process. **31**, 3221–3236 (2023)
15. L. Zhao, L. Feng, D. Ge, R. Chen, F. Yi, C. Zhang, X.L. Zhang, X. Li. Uniform: A unified multi-task diffusion transformer for audio-video generation (2025). arXiv:2502.03897
16. S. Huang, Y. Xu, H. Zhang, X. Li, Learn beneficial noise as graph augmentation. In: *Proceedings of the 42nd International Conference on Machine Learning (ICML)* (Vancouver, 2025)
17. W. Zhang, K. Saijo, J. weon Jung, C. Li, S. Watanabe, Y. Qian, Beyond performance plateaus: A comprehensive study on scalability in speech enhancement. In: *Interspeech 2024*, pp. 1740–1744 (2024). https://doi.org/10.21437/Interspeech.2024-1266

18. H. Wang, D. Wang, Cross-domain diffusion based speech enhancement for very noisy speech. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics*, Speech and Signal Processing (ICASSP), pp. 1–5 (2023). IEEE
19. H. Guo, Y. Chen, X.L. Zhang, X. Li, Graph attention based multi-channel u-net for speech dereverberation with ad-hoc microphone arrays. In: *Proc. Interspeech 2024* (Kos, 2024) pp. 617–621
20. H. Wang, A. Pandey, D. Wang, A systematic study of DNN based speech enhancement in reverberant and reverberant-noisy environments. Comput. Speech Lang. **89**, 101677 (2025)
21. P. Záviška, P. Rajmic, A. Ozerov, L. Rencker, A survey and an extensive evaluation of popular audio declipping methods. IEEE J. Sel. Top. Signal Process. **15**(1), 5–24 (2020)
22. W. Mack, E.A. Habets, Declipping speech using deep filtering. In: *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 200–204 (2019). IEEE
23. L. Diener, S. Branets, A. Saabas, R. Cutler, The ICASSP 2024 audio deep packet loss concealment grand challenge. IEEE Open J. Signal Process. (2025). https://doi.org/10.1109/OJSP.2025.3526552
24. M. Poyser, T.P. Breckon, Neural architecture search: a contemporary literature review for computer vision applications. Pattern Recogn. **147**, 110052 (2024)
25. H. Cheng, M. Zhang, J.Q. Shi, A survey on deep neural network pruning: taxonomy, comparison, analysis, and recommendations. IEEE Trans. Pattern Anal. Mach. Intell. **46**(12), 10558–10578 (2024). https://doi.org/10.1109/TPAMI.2024.3447085
26. G. Hinton, O. Vinyals, J. Dean. Distilling the knowledge in a neural network (2015). arXiv:1503.02531
27. H. Rahmath P, V. Srivastava, K. Chaurasia, R.G. Pacheco, R.S. Couto, Early-exit deep neural network - a comprehensive survey. ACM Comput. Surv. , 37 (2024). https://doi.org/10.1145/3698767
28. J. Yu, L. Yang, N. Xu, J. Yang, T. Huang, Slimmable neural networks. In: *ICLR* (2019)
29. J. Yu, T.S. Huang, Universally slimmable networks and improved training techniques. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision* (IEEE, Seoul, 2019) pp. 1803–1811
30. X.L. Zhang, X. Li, Robust multilayer bootstrap networks in ensemble for unsupervised representation learning and clustering. Pattern Recognit. **156**, 110739 (2024). https://doi.org/10.1016/j.patcog.2024.110739
31. S. Chen, Y. Wu, Z. Chen, T. Yoshioka, S. Liu, J. Li, X. Yu, Don't shoot butterfly with rifles: Multi-channel continuous speech separation with early exit transformer. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6139–6143 (2021). IEEE
32. A. Li, C. Zheng, L. Zhang, X. Li, Learning to inference with early exit in the progressive speech enhancement. In: *2021 29th European Signal Processing Conference (EUSIPCO)*, pp. 466–470 (2021). IEEE
33. S. Kim, M. Kim, Bloom-net: Blockwise optimization for masking networks toward scalable and efficient speech enhancement. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 366–370 (2022). IEEE
34. R. Miccini, A. Zniber, C. Laroche, T. Piechowiak, M. Schoeberl, L. Pezzarossa, O. Karrakchou, J. Sparsø, M. Ghogho, Dynamic nsnet2: Efficient deep noise suppression with early exiting. In: *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6 (2023). IEEE
35. M. Elminshawi, S.R. Chetupalli, E.A. Habets, Slim-tasnet: A slimmable neural network for speech separation. In: *2023 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 1–5 (2023). IEEE
36. M. Elminshawi, S.R. Chetupalli, E.A. Habets, Dynamic slimmable network for speech separation. IEEE Signal Process. Lett. (2024). https://doi.org/10.1109/LSP.2024.3445304
37. R. Miccini, C. Laroche, T. Piechowiak, L. Pezzarossa, Scalable speech enhancement with dynamic channel pruning. In: *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5 (2025). IEEE
38. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, (IEEE, 2016), pp. 770–778
39. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need. Adv. Neural Inf. Process. Syst. **30** (2017) pp. 6000–6010
40. L. Feng, L. Zhao, B. Zhu, X.L. Zhang, X. Li, Audiospa: Spatializing sound events with text (2025). arXiv preprint arXiv:2502.11219
41. B. Zhang, R. Sennrich, Root mean square layer normalization. Adv. Neural Inf. Process. Syst. **32** (2019) pp. 12381–12392
42. W.T. Lu, J.C. Wang, Q. Kong, Y.N. Hung,Music source separation with band-split rope transformer. In: *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 481–485 (2024). IEEE
43. W. Zhang, R. Scheibler, K. Saijo, S. Cornell, C. Li, Z. Ni, A. Kumar, J. Pirklbauer, M. Sach, S. Watanabe, et al., Urgent challenge: Universality, robustness, and generalizability for speech enhancement (2024). arXiv preprint arXiv:2406.04660
44. J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, Y. Liu, Roformer: enhanced transformer with rotary position embedding. Neurocomputing **568**, 127,063 (2024)
45. X. Tan, X.L. Zhang, Speech enhancement aided end-to-end multi-task learning for voice activity detection. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6823–6827 (2021). IEEE
46. H. Dubey, A. Aazami, V. Gopal, B. Naderi, S. Braun, R. Cutler, A. Ju, M. Zohourian, M. Tang, M. Golestaneh et al., Icassp 2023 deep noise suppression challenge. IEEE Open J. Signal Process. (2024). https://doi.org/10.1109/OJSP.2024.3378602
47. D.B. Paul, J. Baker, The design for the wall street journal-based CSR corpus. In: *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, (Morgan Kaufmann Publishers, 1992)
48. H. Zen, V. Dang, R. Clark, Y. Zhang, R.J. Weiss, Y. Jia, Z. Chen, Y. Wu, Libritts: a corpus derived from librispeech for text-to-speech. In: *Interspeech 2019*, (2019), pp. 1526–1530. https://doi.org/10.21437/Interspeech.2019-2441
49. C.V. Botinhao, X. Wang, S. Takaki, J. Yamagishi, Investigating RNN-based speech enhancement methods for noise-robust text-to-speech. In: *9th ISCA speech synthesis workshop*, (ISCA, 2016), pp. 159–165
50. J. Richter, Y.C. Wu, S. Krenn, S. Welker, B. Lay, S. Watanabe, A. Richard, T. Gerkmann, Ears: An anechoic fullband speech dataset benchmarked for speech enhancement and dereverberation. In: *Interspeech 2024*, pp. 4873–4877 (2024). https://doi.org/10.21437/Interspeech.2024-153
51. C.K. Reddy, V. Gopal, R. Cutler, DNSMOS: a non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6493–6497 (2021). IEEE
52. G. Wichern, J. Antognini, M. Flynn, L.R. Zhu, E. McQuinn, D. Crow, E. Manilow, J.L. Roux, Wham!: Extending speech separation to noisy environments. In: *Interspeech 2019*, (2019), pp. 1368–1372. https://doi.org/10.21437/Interspeech.2019-2821
53. E. Fonseca, X. Favory, J. Pons, F. Font, X. Serra, Fsd50k: an open dataset of human-labeled sound events. IEEE/ACM Trans. Audio Speech Lang. Process. **30**, 829–852 (2021)
54. K.J. Piczak, ESC: Dataset for environmental sound classification. In: *Proceedings of the 23rd ACM international conference on Multimedia*, (IEEE, 2015), pp. 1015–1018
55. R. Scheibler, E. Bezzam, I. Dokmanić, Pyroomacoustics: A python package for audio room simulation and array processing algorithms. In: *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, (IEEE, 2018), pp. 351–355
56. L. Diener, M. Purin, S. Sootla, A. Saabas, R. Aichner, R. Cutler, Plcmos – a data-driven non-intrusive metric for the evaluation of packet loss concealment algorithms. In: *Interspeech 2023*, (2023), pp. 2533–2537. https://doi.org/10.21437/Interspeech.2023-1532
57. A. Rix, J. Beerends, M. Hollier, A. Hekstra, Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 2 (2001), pp. 749–752 vol.2. https://doi.org/10.1109/ICASSP.2001.941023
58. J. Jensen, C.H. Taal, An algorithm for predicting the intelligibility of speech masked by modulated noise maskers. IEEE/ACM Trans. Audio Speech Lang. Process. **24**(11), 2009–2022 (2016). https://doi.org/10.1109/TASLP.2016.2585878
59. J.w. Jung, Y.J. Kim, H.S. Heo, B.J. Lee, Y. Kwon, J.S. Chung, Pushing the limits of raw waveform speaker recognition. Proc. Interspeech (ISCA, 2022)

60. A. Radford, J.W. Kim, T. Xu, G. Brockman, C. McLeavey, I. Sutskever, Robust speech recognition via large-scale weak supervision. In: *International Conference on Machine Learning*, pp. 28492–28518 (2023). PMLR
61. Y. Luo, N. Mesgarani, Conv-tasnet: surpassing ideal time-frequency magnitude masking for speech separation. IEEE/ACM Trans. Audio Speech Lang. Process. **27**(8), 1256–1266 (2019)
62. S. Zhao, B. Ma, K.N. Watcharasupat, W.S. Gan, Frcrn: Boosting feature representation using frequency recurrence for monaural speech enhancement. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 9281–9285 (2022). IEEE
63. Haohe Liu and Xubo Liu and Qiuqiang Kong and Qiao Tian and Yan Zhao and DeLiang Wang and Chuanzeng Huang and Yuxuan Wang, VoiceFixer: A Unified Framework for High-Fidelity Speech Restoration. In: *Interspeech 2022*, pp. 4232–4236 (2022). https://doi.org/10.21437/Interspeech.2022-11026
64. Z. Wang, X. Zhu, Z. Zhang, Y. Lv, N. Jiang, G. Zhao, L. Xie, Speech enhancement using discrete tokens and language models. In: *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 11561–11565 (2024). IEEE
65. X. Li, Q. Wang, X. Liu, Masksr: Masked language model for full-band speech restoration. In: *Interspeech 2024*, pp. 2275–2279 (2024). https://doi.org/10.21437/Interspeech.2024-1584
66. J. Yao, H. Liu, C. Chen, Y. Hu, E. Chng, L. Xie, GenSE: Generative speech enhancement via language models using hierarchical modeling. In: *The Thirteenth International Conference on Learning Representations* (2025). https://openreview.net/forum?id=1p6xFLBU4J. Accessed 23 Jan 2025
67. J. Zhang, J. Yang, Z. Fang, Y. Wang, Z. Zhang, Z. Wang, F. Fan, Z. Wu, Anyenhance: a unified generative model with prompt-guidance and self-critic for voice enhancement. IEEE Trans. Audio Speech Lang. Process. **33**, 3085–3098 (2025). https://doi.org/10.1109/TASLPRO.2025.3587393
68. J.M. Valin, A. Mustafa, C. Montgomery, T.B. Terriberry, M. Klingbeil, P. Smaragdis, A. Krishnaswamy, Real-time packet loss concealment with mixed generative and predictive model (2022). arXiv preprint arXiv:2205.05785
69. B. Liu, Q. Song, M. Yang, W. Yuan, T. Wang, Plcnet: Real-time packet loss concealment with semi-supervised generative adversarial network. In: *INTERSPEECH*, (ISCA, 2022), pp. 575–579
70. Z. Zhang, J. Sun, X. Xia, C. Huang, Y. Xiao, L. Xie, Bs-plcnet: Band-split packet loss concealment network with multi-task learning framework and multi-discriminators. In: *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)* pp. 23–24 (2024). IEEE

Publisher's Note