**IMPROVING DESIGN OF INPUT CONDITION INVARIANT SPEECH ENHANCEMENT**

Wangyou Zhang, Jee-weon Jung2,∗ , Yanmin Qian

Shanghai Jiao Tong University, China 2Carnegie Mellon University, USA

**Team Members:**

* **23BEC7117 : Abhishek Bharadwaj**
* **22BEC7186 : Sourav Vir Singh**

**Absract**

**Creating one single system that can improve speech quality across all types of input situations is still a big challenge in the research world, even though it’s something many experts want. The idea is to build a model that can work smoothly with audio of different lengths, at different sound quality levels (sampling rates), and with various microphone setups—even when the background is noisy or echoes a lot. This kind of ability is what we mean when we say “input condition invariant speech enhancement.” Some progress has been made recently, and a model was introduced that showed good results in ideal or controlled situations. But when it came to using multiple microphones in real-life environments, the model’s performance dropped a lot. This shows that while we're on the right path, there’s still a need to improve how these systems handle more complex, unpredictable audio inputs.**

**In this work, we present new system designs aimed at improving the ability of a speech enhancement model to perform well, even when the type of input varies widely. While the model already performs decently in controlled or simulated environments, our goal is to reduce the drop in performance when the model is tested in actual, real-life situations. To make this possible, we started by revisiting and modifying several core parts of the system. One of the main problems we found was with the part responsible for handling multiple input audio channels—it didn’t adapt well when exposed to unfamiliar data or settings. So, we redesigned this component to make it stronger and more adaptable. In addition to that, we came up with a two-phase training method. This training approach boosts the model’s learning speed and helps it perform better across a wide range of different input scenarios.**

**Secondly, we introduced two newly designed time-frequency blocks that work in two separate paths. These blocks are more efficient than older methods—they not only deliver better results but also require fewer internal settings and less computing power. Their main advantage is that they help improve the speech enhancement system without making it heavier or slower. To test how well our updated model performs, we ran several tests using different publicly available datasets. These experiments were important in checking whether the model works properly in real-life noisy environments. The outcomes clearly show that our upgraded approach handles real-world situations much better than older models, which often struggled outside of lab conditions.**

**To support open research and allow others to build upon our work, we plan to share a complete set of instructions and resources that explain how the model was created and trained. These detailed steps will make it easy for others to reproduce the results we achieved. Overall, this research brings us closer to designing a speech enhancement system that works well under a wide range of real-life situations—whether the audio is recorded with different microphone setups, comes at varying sound quality levels, or uses different sampling rates.**

**Building a universal speech enhancement (SE) system capable of handling diverse input conditions—such as varying audio durations, sampling frequencies, and microphone configurations—remains a challenging and underexplored area. This paper presents novel architectures aimed at improving the robustness and performance of such input condition-invariant SE systems, especially in real-world environments. Key contributions include a redesigned channel-modeling module to improve generalization, a two-stage training strategy for better training efficiency, and two novel dual-path time-frequency blocks that reduce computational cost while improving performance. Extensive experiments on public datasets demonstrate significant performance improvements, especially under real-world conditions, marking a major step toward practical and universal SE solutions. The study also promises to release reproducible model recipes for the research community.**

**Keywords— Enhancing speech clarity in all types of environments, works across different sound sampling rates, adaptable to setups with any number of microphones without needing changes.**

**I. INTRODUCTION**

**Speech enhancement, or SE, is basically a technique used to clean up audio by removing background disturbances like noise and echoes that usually lower the clarity of spoken words. The main goal is to make the speech sound clearer and easier to understand, especially in places with lots of noise or echo. Although SE involves several tasks, this study mainly looks at two key challenges—removing background noise (called denoising) and reducing the echo effect (known as dereverberation). These two issues are especially common when trying to improve speech quality in real-life environments.In recent years, especially over the past ten, there has been a lot of development in machine learning methods to improve speech quality. These advanced methods have shown great success in handling different types of noisy situations. Some of the popular approaches include separating useful speech signals from noise by using time and frequency-based filtering techniques. Others focus on analyzing the sound directly over time using special neural networks that learn to clean speech. There are also some newer designs that combine multiple techniques to get even better results.**

**However, most existing methods predominantly assume fixed input conditions, such as a single microphone or a predefined sampling frequency. Consequently, the SE models developed under these constraints often fail to generalize across different scenarios, particularly when transitioning between input conditions. The discrepancy between trained and unseen input conditions is a well-known challenge that leads to degraded model performance.**

**In recent times, researchers have introduced a new method called the Unconstrained Speech Enhancement and Separation Network, also known as USES. This system aims to create a flexible solution that can work under different types of input conditions. It is specially built to manage three major variations: one, changes in audio sampling rates; two, the presence of different numbers and arrangements of microphones; and three, input audio that varies in length. To tackle these challenges, the USES system makes use of a technique known as Short-Time Fourier Transform (STFT), which helps break down sound signals for better analysis. On top of that, it includes a dual-path model that processes both the timing and frequency aspects of the audio. Another helpful component added to this setup is the TAC (transform-average-concatenate) method. This technique improves the model’s ability to learn efficiently while keeping it steady across a wide range of input types. This ability of the USES model—to work well regardless of how the input conditions change—is what makes it a strong step forward in building a truly universal speech enhancement system.**

**Even though the USES system performs quite well across several datasets, its effectiveness drops noticeably when tested in real-world situations—especially in complex scenarios like the CHiME-4 recordings, which involve multiple microphones. This drop in performance is mainly because of the current model design, which doesn’t handle such situations efficiently. A major issue lies in how the TAC (Transform-Average-Concatenate) components work. These modules try to simplify the input from multiple microphones by averaging the audio signals and then converting them into a single format for each channel. While this can work in some cases, it doesn’t do a good job when there’s a big difference in sound quality between the microphones—especially when the noise levels vary. Another limitation is that the TAC modules handle single-microphone and multi-microphone data in separate ways. Because of this, it becomes difficult for the model to learn a common way to process both types of input, which affects its ability to work well across a wide range of situations.**

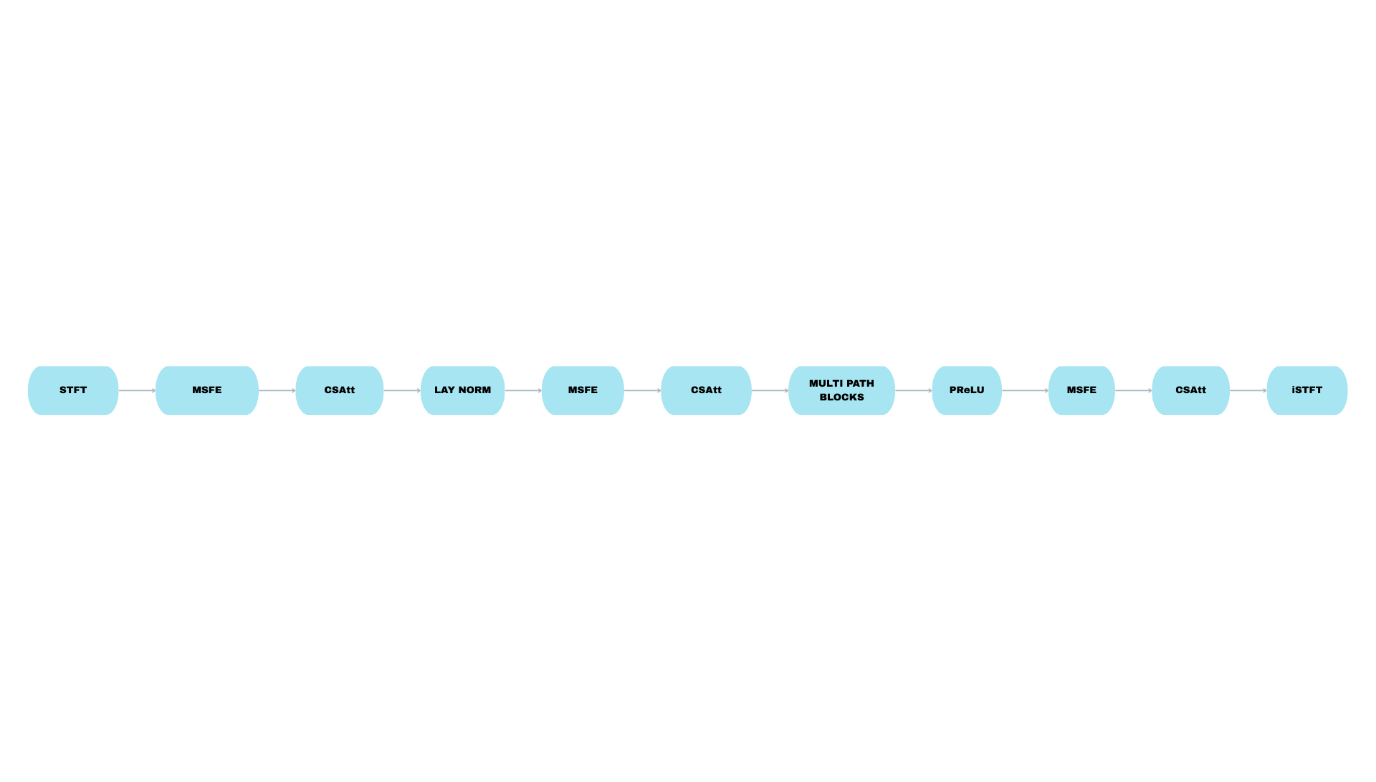
**In this study, we focus on improving the capabilities of the existing USES model, especially when it needs to work with a variety of input settings. Instead of sticking to one fixed setup, our aim is to make the system flexible and efficient for different types of audio inputs. To move in this direction, we propose two improved versions of the model—named USES-Comp and USES-Swin. These upgraded designs are capable of handling both single-microphone and multiple-microphone inputs effectively, while also making the system lighter by cutting down on unnecessary computation. A key part of our method involves reworking how time and frequency information in the audio is processed, using fresh modeling strategies that can extract richer features. Alongside this, we also improve how the system handles data from different microphone channels, ensuring better performance in real-world environments. Moreover, we make changes to how the model is trained. Instead of training everything at once, we break the process into two separate stages. The first stage focuses on simpler single-channel input, while the second one prepares the model to manage multi-channel input. This step-by-step learning process not only boosts training efficiency but also helps the model adapt better to varied conditions during actual use. It ensures smoother learning when combining data from both types of inputs.**

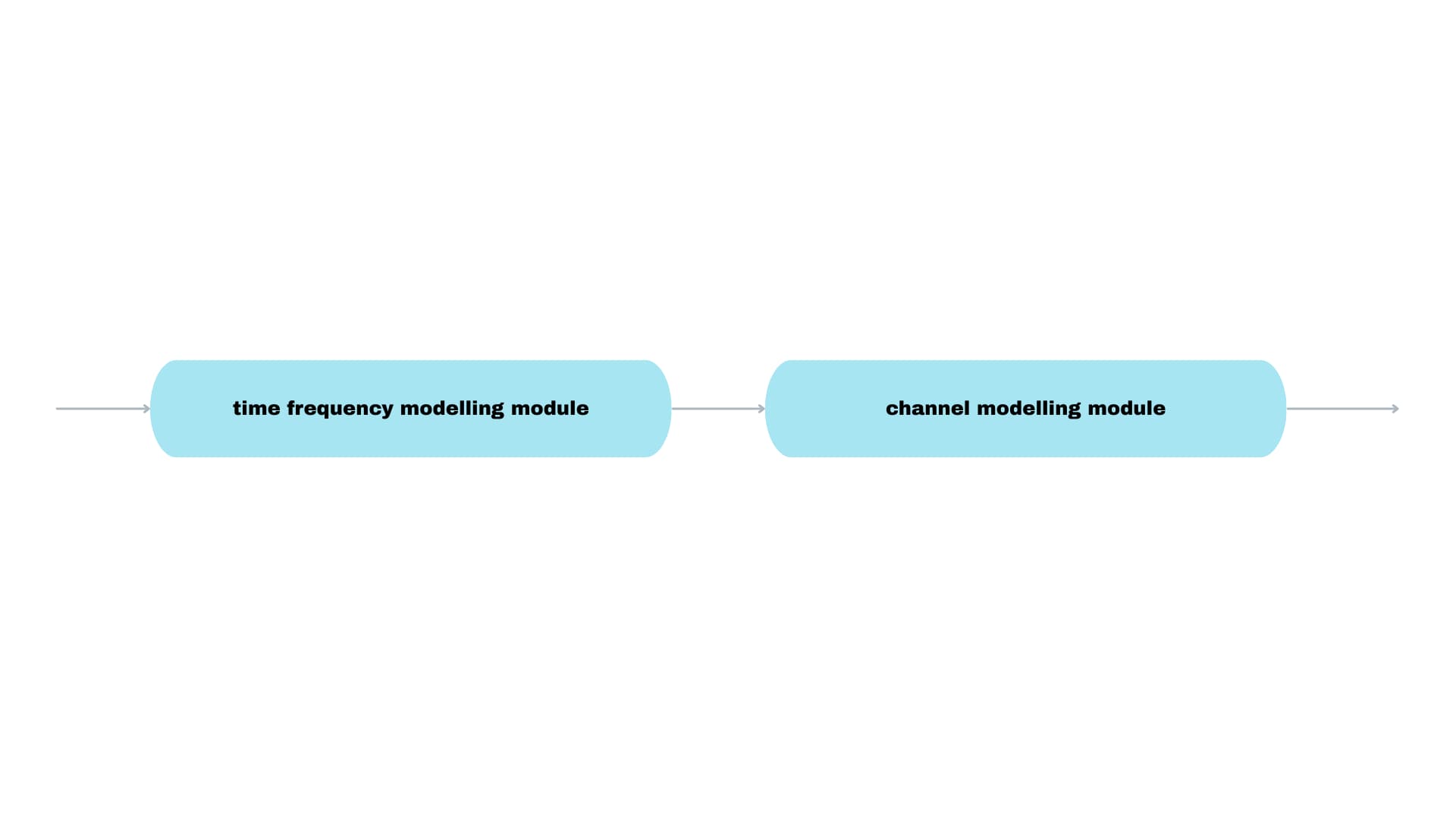
**To test how well our improved approach works, we carried out experiments using five well-known and openly available speech datasets: VoiceBank+DEMAND, DNS, CHiME-4, REVERB, and WHAMR!. These datasets cover a wide range of real-world sound conditions, including different types of background noise and microphone setups. After running several tests, the results clearly showed that our updated model does a better job at enhancing speech and handling different types of input compared to the older USES system. Not only does it perform better in terms of sound quality, but it also adapts more effectively to various environments and setups, proving its strength in generalization.**

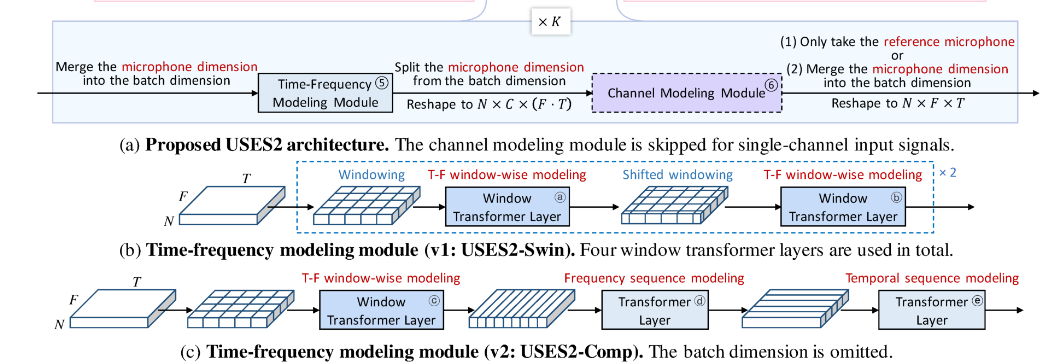
**Speech enhancement (SE) focuses on improving speech quality by reducing noise and reverberation, especially in real-world conditions. While recent learning-based methods have shown strong results, most assume fixed input settings—like a single microphone or specific sampling frequency—which limits their adaptability to varied scenarios.**

**To address this, the USES framework was introduced, designed to handle different sampling rates, microphone numbers, and input durations. It uses STFT-based encoding and a dual-path time-frequency (T-F) modeling approach, along with a transform-average-concatenate (TAC) method for channel modeling.**

**However, USES struggles in real-world conditions due to limitations in its channel modeling, especially when dealing with microphones that have varying signal qualities. In this work, the authors propose two improved models—USES2-Comp and USES2-Swin—which feature enhanced channel modeling, advanced T-F processing, and a two-stage training strategy. These improvements boost both performance and generalization, moving closer to a universal SE system.**

****



****

**Figure 1: Overview of the Suggested USES2 Model for Speech Enhancement Under Different Input Settings**

**This illustration shows the architecture of the proposed speech enhancement system called USES2, which is built to work well even when input conditions like sampling rate or number of microphones change. The diagram highlights various layers of the model, with specific focus on the convolutional layers. Their filter sizes and the number of output features (also called feature maps) are marked using grey color for easy identification. The design combines both spectral and spatial processing paths to handle sound signals efficiently in different real-world environments.**

**A. Multi-Scale Feature Extractor (MSFE)**

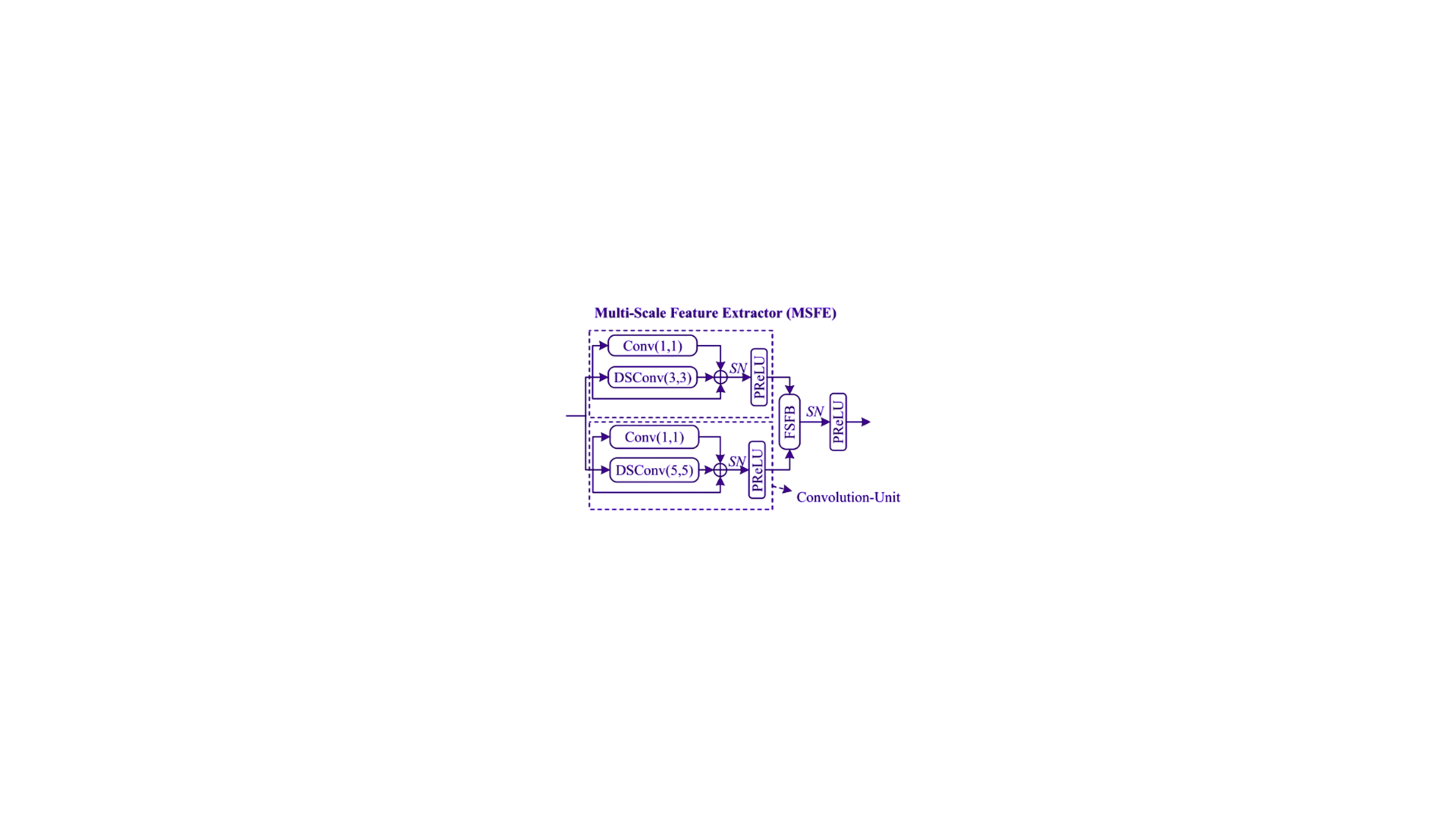
**To improve the system’s ability to extract useful features from audio signals, we introduced a module called the Multi-Scale Feature Extractor (MSFE), inspired by earlier research [28]. This component, as displayed in Figure 1, is built using two main processing units called convolution units (CUs), along with another block known as the Feature Selective Fusion Block (FSFB).**

**After processing through the FSFB, the signals go through a special activation function called PReLU (Parametric ReLU) [29]. Then, a flexible normalization method known as Switchable Normalization (SN) [30] is applied. SN is smart enough to learn which type of normalization is best for the data by considering different viewpoints like per channel, entire layer, or small data batches.**

**Each convolution unit follows a design that uses multiple branches. These branches perform operations like depthwise 2D convolutions (DSConv), pointwise convolutions, and also include shortcut connections that help preserve information. These outputs are then merged and passed through the normalization and activation steps again.**

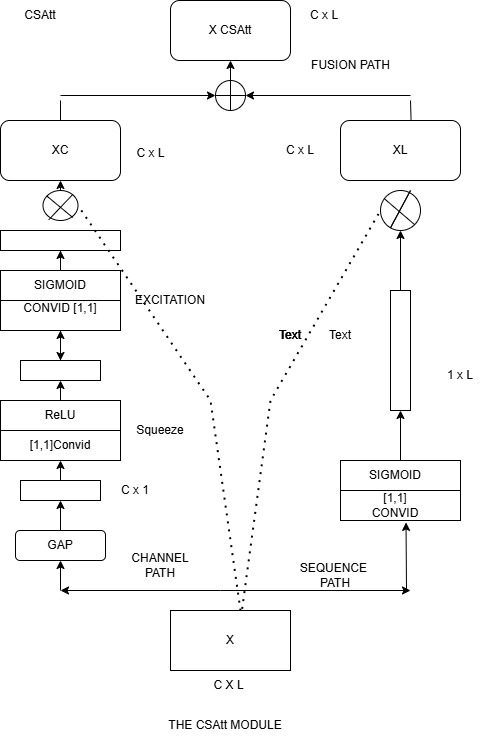
**The two convolution units in the MSFE use different filter sizes—one uses 3x3 filters, and the other uses 5x5 filters. This variation helps the model pick up patterns at different scales. By using such a mix of simple and complex operations across multiple paths, the system can capture a wider variety of features, making the extracted information richer and more expressive.**

**To make sure that the information from the two CUs is combined in an effective way, we added the FSFB module, which plays a key role in merging features from different levels.**

****

**B. Channel and sequence attention (CSAtt)**

To enhance the feature spaces, a channel and sequence attention (CSAtt) module is designed to learn objective-oriented feature weights in dual parallel attention paths. The channel attention path focuses on the feature dimension to excite the informative points and suppress disturbance features. The sequence attention path is expected to allocate dedicated weights to highlight salient frames and discard echo frames, which further enhances the feature discriminative ability. The **Channel and Sequence Attention (CSAtt)** module is a key component of the ROSE framework designed to enhance speech features by focusing on both informative channels and crucial time frames. It operates in two parallel paths: the **channel attention path**, which emphasizes feature dimensions with significant spoken information while suppressing noise-related channels, and the **sequence attention path**, which assigns higher weights to meaningful frames—especially central phoneme frames—and reduces the impact of echo-dominant or irrelevant frames. These two attention outputs are then fused to produce a refined feature map that improves the model's ability to enhance speech and support accurate ASR (Automatic Speech Recognition), particularly in challenging Air Traffic Control (ATC) scenarios.



**2. PROPOSED METHODS**

**2.1. Overview**

**The updated USES2 model follows a dual-path design that works in both time and frequency domains. This setup is naturally independent of the audio’s sampling rate, making it flexible across various input types—similar to the structure proposed in earlier research [12]. As shown in Figure 1(a), the model includes multiple layers of resolution, mainly composed of the following parts:**

1. **An encoder based on the Short-Time Fourier Transform (STFT) (components ① to ③)**
2. **A set of multi-path processing blocks (labeled as ④), which handle both frequency-based and spatial signal features**
3. **A decoder section that also uses STFT (components ⑦ to ⑨)**

**To begin with, the system takes an input audio signal and breaks it down into C different channels, where C is one or more. After that, the signal is converted into a complex spectrum that holds the dimensions C × 2 × F × T, where F and T refer to frequency bins and time frames, respectively.**

**Each time-frequency unit within this spectrum is then passed through two layers of 2D convolution (parts ② and ③). After this step, a layer normalization is applied to standardize the features.**

**Then comes the multi-path block stage (marked as ④), where a series of K blocks are stacked. The first Kₛ blocks process the features using:**

* **A time-frequency modeling unit (⑤)**
* **A channel modeling unit (⑥)**

**For the rest of the blocks, that is, the remaining K − Kₛ layers, only the time-frequency module is used, helping in further refinement of the spectral features.**

**Once this processing is complete, the multi-channel output is simplified to a single-channel form. This is done by selecting one reference channel and removing the others. A key difference from the method in [12] is that—for single-channel inputs, the system now skips the channel modeling phase altogether, which makes it more efficient.**

**This separation of design allows the model to handle both single- and multi-channel audio effectively, something that wasn’t possible in the older version [12].**

**The decoder portion of the framework includes the following parts:**

* **A PReLU activation function (⑦) that adjusts based on the data**
* **A point-wise convolution layer (⑧)**
* **A 2D transposed convolutional layer (⑨)**

**Finally, the processed output is transformed back to its original waveform using the inverse STFT (iSTFT).**

**2.2. Time-Frequency Modeling**

**The use of transformer-based models to separately handle the frequency and time dimensions, as seen in studies [12, 13], has shown promising outcomes in speech enhancement. Yet, treating these two aspects—frequency and time—as completely separate paths tends to limit how well the model can adapt or generalize to a variety of conditions [20, 21]. This limitation is especially noticeable because speech signals, over short durations, typically show a strong connection between nearby frequency components. So, when we focus on small localized areas within the time-frequency (T-F) space, the system can better capture spectral details and patterns.**

**Taking inspiration from the Swin Transformer, which has been widely used in image processing tasks [22], particularly due to its ability to process information at multiple levels, we introduce two updated modeling structures that make use of similar ideas for time-frequency handling in speech signals:**

1. **USES2-Swin – A model that uses Swin Transformer blocks for fully integrated T-F analysis**
2. **USES2-Comp – A structure that adds localized window-based modeling to existing time and frequency paths**

**V1: USES2-Swin (Fig. 1(b))**

In **USES2-Swin**, we employ **only the window-wise transformer layers** for spectral modeling, similar to the **Swin transformer** in **vision design** [22, 24]. The **windowing and shifted windowing operations** are applied before **block ③** and **between block ④** transformer layers.

* During the **windowing operation**, the **T-F representation** is **evenly partitioned** into **non-overlapping windows** of size **W\_F × W\_T**, where **W\_F** and **W\_T** denote the number of **frequency** and **time bins**, respectively.
* **Zero-padding** is applied before **windowing (and later removed)**, ensuring that the **number of frequency bins and frames** is **exactly divisible** by **W\_F** and **W\_T**, respectively.
* A **window transformer layer** is then applied to each **window independently** for **local T-F modeling**.

Following [22], a **standard transformer structure** [25] with **layer normalization** is used, incorporating a **learnable 2D relative positional bias** within each **multi-head self-attention module**.

Because the **STFT window duration is fixed**, we set the **constant T-F resolution** in the **spectrum** based on the **sampling frequency (SF)** [12]³. As a result, the **learned relative positional bias** remains usable **without fine-tuning** when handling signals from **different SFs**.

To expand the **receptive field** in **deep transformer layers**, the **shifted windowing operation** is introduced:

* It shifts the **window partition** by **⌊W\_F / 2⌋** and **⌊W\_T / 2⌋** along the **frequency and time dimensions**, respectively.
* By **stacking multiple transformer layers** with **windowing and shifted windowing**, the **T-F modeling module** effectively captures **global** T-F information.

Since this new structure can **natively** handle **arbitrary long signals** through a **window-based approach**, it **removes the memory token process** required in **[12]**.

**V2: USES2-Comp (Fig. 1(c))**

**The USES2-Comp architecture is introduced to supplement the basic time and frequency pathway modeling with additional local window-based processing.**

* **At the beginning of each time-frequency module, we include a window transformer layer that concentrates on extracting details from localized T-F areas.**
* **Similar to [12], we use learnable memory tokens—specifically shaped as 1 × N × 1 × G—and insert them before the transformer layer. These tokens are used during time-domain processing to help the model understand longer sequences, where G is the group size.**
* **Following this, we use two improved transformer layers (based on enhancements from [13, 26])—one for analyzing the frequency path (block ④) and another for time path modeling (block ⑥).**

**The consistent resolution of the T-F representation ensures that frequency gaps between nearby time frames are preserved, allowing smoother modeling of speech dynamics over time.**

**2.3. Channel Modeling Module (⑥)**

Alongside the upgraded time-frequency (T-F) modeling block (denoted as module ⑤), a refined channel modeling unit (marked as module ⑥ in Fig. 1(a)) has also been included. This addition is aimed at improving the system’s spatial processing capability—essentially how the model understands and handles input from different microphone channels.

In earlier designs, the basic version of the transform-average-concatenate (TAC) module didn’t perform well in situations where microphone channels had highly unbalanced signal quality, especially when the signal-to-noise ratios (SNRs) varied a lot. We believe this happens because TAC follows a fixed processing pattern. It doesn't learn to adapt well to new or diverse channel conditions, making it less flexible in unfamiliar scenarios.

To overcome this issue, one technique that was previously explored is channel-wise attention, as proposed in [27]. This method adjusts itself based on the number and arrangement of microphone inputs, and in some cases, it performs better in adapting to new setups. But, our initial testing revealed that this approach still struggles when applied to data from completely different environments. Specifically, it falls short when only a single microphone is used, which requires a more generalized design.

To tackle these shortcomings, we propose a new module named **TACₑₓₜ** short for Transform-Attention-Concatenate Extended. This design tries to take the best parts of both TAC and channel-wise attention. It keeps each channel's information independent, which is useful for generalization, but also adds an attention mechanism. This attention helps the model focus more on the useful parts of input across channels and better understand their relationships.

The process within the TACₑₓₜ block is as follows. Suppose the input feature is represented as:  
  **X ∈ ℝᴺˣ²ˣᶜˣᶠˣᵀ**,  
where **N** is the batch size, **C** is the number of channels, **F** is frequency bins, and **T** is time frames.

We begin with:

**Equation (1):**

Here, a fully connected (FC) linear layer is applied to **X**, followed by a parametric ReLU activation to get feature map **Y**.

**Equation (2):**  
     
An attention operation is carried out on **Y**, which is then passed through another FC layer and PReLU activation to obtain **Y′**.

**Equation (3):**  
    
The two outputs **Y** and **Y′** are joined together along the embedding axis and then processed through an FC layer, PReLU, and finally layer normalization (LN) to generate the output **X**.

In this setup:

* **FC(·)** refers to a trainable linear transformation,
* **LN(·)** denotes normalization applied along the embedding channel to stabilize training.

Now, the channel-wise attention mechanism inside this module works as follows:

**Equation (4):**  
    
The features **Y** go through a ReLU and FC layer, are normalized, and reshaped into a query matrix **Q**.

**Equation (5):**  
     
A similar transformation generates the key matrix **K**.

**Equation (6):**  
     
The value matrix **V** is also created from **Y** after applying the same sequence of operations.

**Equation (7):**  
  

Here, we calculate the attention map by performing a dot product between **Q** and **Kᵀ**, scale it using the square root of the dimension product (denoted by **√HFT**), apply softmax for normalization, and finally multiply the result with **V**. This is followed by FC, ReLU, and layer normalization, producing the final output **A**.

The attention output **A**, with dimensions  **ℝᴺˣᶜˣᶠˣᵀ** effectively brings together time and frequency information into a unified embedding space. This ensures that the attention map remains independent of the number of frequency bins or time steps—making the system more flexible across various audio formats and lengths.

**2.4. Two-Stage Training Strategy**

To improve how efficiently the model trains and how accurate it becomes, we introduce a training method that works in two distinct steps. Since the overall architecture is designed in a way that separates single-channel and multi-channel processes (as mentioned earlier in Section 2.1), each type can be trained more effectively on its own.

During the initial phase, the part of the model responsible for handling multiple input channels—referred to as channel modeling (marked as ⑥ in Fig. 1(a))—is turned off. This step focuses purely on refining how the model learns to enhance audio signals in terms of time and frequency, but only using single-channel data.

Once the first round of training is complete, the next step involves turning channel modeling back on. This second phase helps the model adapt to environments where audio comes from multiple microphones, improving its capability to handle such complex data.

This split-phase process gives the model the ability to handle both types of input more effectively. It cuts down the overall difficulty of training while also boosting the model’s adaptability.

Additionally, training in two steps makes the development process more flexible. By dealing with single-channel and multi-channel audio separately, the system can be fine-tuned more easily for specific tasks—especially useful in real-world scenarios where resources are limited and single-mic setups are common.

**Table 1 gives a detailed overview of the datasets that were used during the training and testing phases of the model. In this table, “#Ch” stands for the total number of microphone channels present in each dataset. A higher number of channels generally indicates more complex input for multi-channel speech processing tasks.**

You’ll also notice two labels: **“(Sim)”** and **“(Real)”**. These terms are used to tell whether the audio data was **artificially created** using simulation methods, or whether it was **captured directly from real-world environments** using physical recording devices.

In addition, the letters **“(A)”** and **“(R)”** are used in the table to indicate **the type of sound environment**. Specifically, **“(A)” refers to anechoic conditions**, which are environments without echo or reverberation—basically clean spaces. On the other hand, **“(R)” represents reverberant environments**, which have echoes and reflections, making speech enhancement more challenging.

Each dataset is divided into three parts: **training, development, and testing**, with the total audio duration for each part clearly stated in hours. This structure helps in maintaining consistency during model evaluation.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Hours (train / dev / test) | SF | #Ch |
| VoiceBank+DEMAND [15] | 8.8 / 0.6 / 0.6 | 48 kHz | 1 |
| DNS1 (v1) [16] | (A)90 / (A)10 / (R)0.42 & (A)0.42 | 16 kHz | 1 |
| CHiME-4 [17] | (Sim)14.7 / (Sim)2.9 / (Sim)2.3 & (Real)2.2 | 16 kHz | 5 |
| REVERB [18] | (Sim)15.5 / (Sim)3.2 / (Sim)4.8 & (Real)0.7 | 16 kHz | 8 |
| WHAMR! [19] | R)58.0 / (R)14.7 / (R)9.0  (A)58.0 / (A)14.7 / (A)9.0 | 16 kHz | 2 |

**3. EXPERIMENTS**

**3.1. Datasets**

**To train the USES2 model, we created a large and diverse dataset that adds up to around 245 hours of audio. Instead of using a single source, we collected data from five different and widely-used speech datasets. These are the same datasets mentioned in the earlier study [12].**

**This combined training set includes audio recorded under various conditions—such as different microphone setups, background noise levels, and environments. You can find more detailed info about these input types in the table provided. Additional technical details are also available in reference [12]. For evaluation, each dataset has its own separate testing part. We used these to measure how well the model performs in different scenarios by applying performance metrics to each set individually.**

**3.2. Configurations**

**During the testing phase, both USES2-Swin and USES2-Comp models were configured with multiple parallel processing blocks—specifically, the Swin variant used three such blocks, while the Comp version made use of four. These blocks are responsible for handling different aspects of the signal, as shown in Figures 1(b) and 1(c). In both models, the first two blocks include a special component for managing input channels, which can be seen in Figure 1(a).**

**Each of these models has a total of 12 layers built using transformer architecture. For analyzing audio signals, a fixed window of 32 milliseconds is used, and it shifts by 16 milliseconds for each step. This setting remains the same, no matter what the original sampling rate of the audio is. Other settings—like the size of feature representations, the bottleneck layer size, and the internal structure of the transformer layers—were kept the same as in earlier related research work [22].**

**In setups where multiple audio channels are used, the system doesn't manually select a reference—it simply picks the first channel by default to act as the baseline for comparison. For training and evaluating the model, all processes were carried out using the ESPnet-SE framework [28], which is specifically built for speech enhancement tasks.**

**The training process was handled by the Adam optimization algorithm. Instead of starting with a high learning rate, the model begins with a small one and slowly increases it. This ramp-up continues until it reaches a peak value of 0.0004 over the first 4000 steps. After this warm-up phase, if the model's accuracy or performance doesn’t show any improvement across two back-to-back validation checks, the system automatically reduces the learning rate by 50% to prevent overfitting and stabilize learning.**

**To save time and computing power, the audio input is broken down into four-second segments during training. Each training batch includes four such segments.**

**When dealing with recordings from multiple microphones, the system randomly selects and shuffles up to four audio channels for each training step. The training process uses a special loss calculation that considers differences in sound at multiple levels of frequency resolution. This frequency-based loss is paired with a similar loss in the time domain, just like the method used in reference [29]. For calculating the frequency loss, different window sizes (256, 512, 768, and 1024) are used to analyze sound at multiple levels. The time-domain loss is given a fixed importance, with a weight of 0.5**.

**Metrics:** Performance is assessed using five key metrics:

* Wideband PESQ (PESQ-WB) [30]
* STOI [31]
* Signal-to-distortion ratio (SDR) [12]
* DNSMOS (OVRL) [33]
* Word error rate (WER), evaluated using the OpenAI Whisper model [7]

**3.3. Results Analysis**

**Model Performance on a Specific Dataset**

**To evaluate how well the new techniques work, detailed experiments were carried out using the CHiME-4 dataset. This dataset helps compare different model versions in both test environments: one that is simulated (artificial) and another that reflects real-world conditions.**

**Table 2 shows a step-by-step comparison of the models. It helps highlight how each change or improvement affects the final outcome. The first two rows in the table serve as benchmarks, giving a reference point for the original models before any upgrades were made.**

**The traditional version of the model, known as USES [12], performs reasonably well in a controlled (simulated) environment. However, when the same model is tested in real situations with background noise and varying inputs, its performance drops drastically. For instance, the Word Error Rate (WER)—which shows how many words were misinterpreted—jumps sharply to 78.1%.**

**This sudden increase in WER clearly signals that the original model is not reliable in practical settings. It fails to maintain accuracy when moved from a lab environment to actual use cases. The rest of the table then explores how the proposed improvements help fix this issue.**

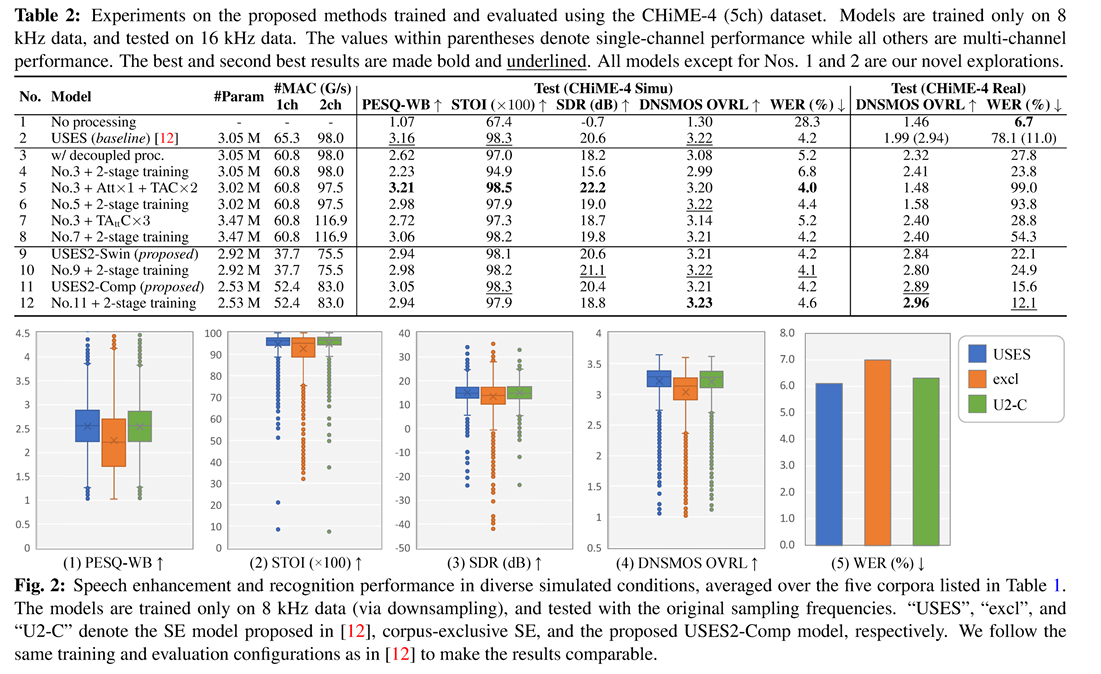
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Model | #Param | #MAC | #MAC |  | Test | (CHiME-4 | Simu) |  | Test (CHiME-4 |  |
| 1 | No processing | - | 1ch | 1ch | PESQ-WB | STOI (x 100) 1 | SDR (dB) 1 | DNSMOS OVRL | WER (%) t | DNSMOS OVRL | Real) |
| 2 | USES (baseline) [12] | 3.05 M |  |  | 1.07 | 67.4 | -0.7 | 1.30 | 28.3 | 1.46 | 1 WER (%) t |
| 3 | w/ decoupled proc. | 3.05 M | 65.3 | 65.3 | 3.16 | 98.3 | 20.6 | 3.22 | 4.2 | 1.99 (2.94) | 6.7 |
| 4 | No.3 + 2-stage training | 3.05 M | 60.8 | 60.8 | 2.62 | 97.0 | 18.2 | 3.08 | 5.2 | 2.32 | 78.1 (11.0) |
| 5 | No.3 + Attx1 + TACx 2 | 3.02 M | 60.8 | 60.8 | 2.23 | 94.9 | 15.6 | 2.99 | 6.8 | 2.41 | 27.8 |
| 6 | No.5 + 2-stage training | 3.02 M | 60.8 | 60.8 | 3.21 | 98.5 | 22.2 | 3.20 | 4.0 | 1.48 | 23.8 |
| 7 | No.3 + TACx3 | 3.47 M | 60.8 | 60.8 | 2.98 | 97.9 | 19.0 | 3.22 | 4.4 | 1.58 | 99.0 |
| 8 | No.7 + 2-stage training | 3.47 M | 60.8 | 60.8 | 2.72 | 97.3 | 18.7 | 3.14 | 5.2 | 2.40 | 93.8 |
| 9 | USES2-Swin (proposed) | 2.92 M | 60.8 | 60.8 | 3.06 | 98.2 | 19.8 | 3.21 | 4.2 | 2.40 | 28.8 |
| 10 | No.9 + 2-stage training | 2.92 M | 37.7 | 37.7 | 2.94 | 98.1 | 20.6 | 3.21 | 4.2 | 2.84 | 54.3 |
| 11 | USES2-Comp (proposed) | 2.53 M | 37.7 | 37.7 | 2.98 | 98.2 | 21.1 | 3.22 | 4.1 | 2.80 | 22.1 |
| 12 | No.11 + 2-stage training | 2.53 M | 52.4 | 52.4 | 3.05 | 98.3 | 20.4 | 3.21 | 4.2 | 2.89 | 24.9 |

**Table 2 presents the results from our experiments, where we tested different speech enhancement methods using the CHiME-4 dataset, which includes recordings with five microphones. In this study, all models were trained using audio data that had been downsampled to 8 kHz. However, during testing, we used 16 kHz data to check how well the models perform under different sample rates.**

**In the table, some performance numbers are written inside brackets—these refer to results from models tested on single-microphone (single-channel) inputs. The rest of the results are based on data with multiple microphones (multi-channel).**

**To highlight the best models, we have used bold and underlined text to mark the top-performing and second-best methods.**

**It's important to note that except for model numbers 1 and 2, all the remaining models are our own new designs that we developed as part of this research.**



**Figure 2 shows how well the models performed in both speech enhancement and recognition when tested in different simulated environments. These results are averaged across five different datasets, which are listed earlier in Table 1. For training, all models were given audio data that had been downsampled to 8 kHz. However, when testing, the models were evaluated using the audio at its original sampling rate to see how well they adapted.**

**The names "USES", "excel", and "U2-C" represent different speech enhancement models. The "USES" model refers to the one introduced in a previous study [12]. The "excel" model was trained separately on each individual dataset, meaning it was specific to one type of input. Meanwhile, "U2-C" stands for the newly suggested USES2-Comp model, which is meant to work well across all types of input conditions.**

**To keep the comparison fair, the researchers used the same setup for training and testing as was done in the earlier study [12]. This makes sure that the performance results of the new and older models can be directly compared without bias.**

**Table 3: Performance evaluation on various real recordings**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test set |  | DNSMOS | OVRL |  |  | WER | (%) |  |
|  | noisy | USES | excl | U2-C | noisy | USES | excl | U2-C |
| CHiME-4 (Real, 5ch) | 1.46 | 1.58 | 2.89 | 3.08 | 6.7 | 85.9 | 15.6 | 10.3 |
| REVERB (Real, 8ch) | 1.57 | 3.11 | 2.10 | 3.07 | 5.8 | 5.1 | 4.9 | 5.1 |

Rows 9 to 12 illustrate USES2 performance with and without the proposed two-stage training. Both architectures outperform the USES model, whereas the proposed USES2-Comp with two-stage training demonstrates the best performance in real conditions, as evidenced by DNSMOS and WER. Additionally, both proposed models reduce parameter and computational costs compared to USES. Since USES2-Comp shows superior performance over USES2-Swin in Table 2, this architecture is adopted for the remaining experiments.

**Evaluation of USES2 in Diverse Conditions**

**The performance of the updated USES2 system was tested using several real-world speech datasets, and the findings are summarized in Table 3. One of the models used for this evaluation is called USES2-Comp, labeled as "U2-C". This model was trained using a two-phase training technique, where different types of audio inputs were included together in the dataset, as explained earlier in Section 3.1. In addition to this, another type of model was trained separately for each individual dataset using the same USES2-Comp structure. These are referred to as corpus-exclusive models (marked as "excl."), and they were used to check how well U2-C performs when compared with models that are tuned for specific data only.**

**To reduce the strain of training on large datasets, the audio files were first converted to a lower quality format, specifically 8 kHz. This helped speed up the initial training phase and made it easier to handle the data. Once the training was completed, the models were then tested using high-resolution audio with their original sampling rates to see how well they performed in real situations.**

**Among all the test sets, the CHiME-4 dataset stood out. It showed a major boost in performance when the U2-C model was used. This model nearly doubled the results compared to earlier methods. The biggest improvements were seen in two areas: the DNSMOS OVRL score, which checks how natural or clear the audio sounds, and the Word Error Rate (WER), which measures how many mistakes happen in speech recognition. These gains were especially significant when the recordings included sound from multiple microphones, making them more complex and closer to real-life environments.**

**This performance jump is mainly due to an improved component in the model—called the TAC\_extended module, which was introduced in Section 2.3. Unlike the older version (also known as the vanilla TAC), this updated module is much better at dealing with cases where audio signals come from microphones that have different levels of clarity or strength—something that often happens in real setups like CHiME-4.**

**Beyond just one dataset, the model also performed strongly on others, including those it wasn’t specifically trained on. This shows that the USES2-Comp model has a good ability to adapt and generalize to new or unexpected conditions, which is a valuable trait in practical applications like real-time speech recognition. Another important aspect is that this newer model keeps its performance high in simulated environments as well—it doesn't drop in quality when tested outside of real-world scenarios. As seen in Figure 2, both the original USES and the U2-C version gave similar results on five different datasets. Interestingly, both outperformed the corpus-specific (excl.) models, even though those were fine-tuned for individual datasets. When we put together all these observations—especially from Table 3 and Figure 2—it becomes clear that the USES2-Comp model isn’t just more flexible, it also uses fewer resources. It cuts down on the number of parameters and the overall processing cost, making it a smart and practical alternative to the traditional USES approach.**

**4. Conclusion**

**In this research, we put forward two improved versions of speech enhancement models, named USES2-Comp and USES2-Swin. These upgraded designs aim to make speech improvement systems work better even when the input conditions change. This means they can handle different types of audio recordings—whether the quality isn’t great, or the microphone arrangement is unusual—as often happens in everyday noisy environments. What makes these models stand out is their strong and steady performance, not just in real-world use, but also when tested in ideal lab-like conditions.** **This shows how adaptable and dependable they are in various scenarios**.**A major improvement we added was a two-phase training process. This training method helps the models learn more efficiently by first focusing on simpler, single-microphone inputs and then shifting to more complex, multi-microphone data. This step-by-step learning makes it easier for the model to adapt to both types of scenarios, avoiding imbalance in training. To test how well our ideas worked, we carried out several experiments using well-known open-source datasets. The results confirmed that the proposed architectures not only performed well but were also consistent and reliable under different testing conditions. Both USES2 models stood out for their ability to process audio clearly while keeping resource use (like memory and computing power) low.**

**Looking ahead, our goal is to continue developing a more universal SE model—one that can manage even more types of background noise and signal problems. By doing so, we hope to make our approach suitable for a wider range of real-world uses, from mobile devices to complex multi-microphone setups.**

**REFERENCES**

[1] P. C. Loizou, Speech enhancement: theory and practice. CRC press, 2013. 1

[2] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, “A regression approach to speech enhancement based on deep neural networks,” IEEE/ACM Trans. ASLP., vol. 23, no. 1, pp. 7–19, 2014. 1

[3] Y. Wang, A. Narayanan, and D. Wang, “On training targets for supervised speech separation,” IEEE/ACM Trans. ASLP., vol. 22, no. 12, pp. 1849–1858, 2014. 1

[4] D. S. Williamson et al., “Time-frequency masking in the complex domain for speech dereverberation and denoising,” IEEE/ACM Trans. ASLP., vol. 25, no. 7, pp. 1492–1501, 2017. 1

[5] Z.-Q. Wang, P. Wang, and D. Wang, “Complex spectral mapping for single-and multi-channel speech enhancement and robust ASR,” IEEE/ACM Trans. ASLP., vol. 28, pp. 1778–1787, 2020. 1

[6] Y. Luo and N. Mesgarani, “Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation,” IEEE/ACM Trans. ASLP., vol. 27, no. 8, pp. 1256–1266, 2019. 1

[7] A. Pandey and D. Wang, “A new framework for CNN-based speech enhancement in the time domain,” IEEE/ACM Trans. ASLP., vol. 27, no. 7, pp. 1179–1188, 2019. 1 [8] Y. Luo, Z. Chen, and T. Yoshioka, “Dual-path RNN: efficient long sequence modeling for time-domain single-channel speech separation,” in Proc. IEEE ICASSP, 2020, pp. 46–50. 1

[9] S. Zhao, B. Ma, K. N. Watcharasupat, and W.-S. Gan, “FRCRN: Boosting feature representation using frequency recurrence for monaural speech enhancement,” in Proc. IEEE ICASSP, 2022, pp. 9281–9285. 1

[10] Z.-Q. Wang, S. Cornell, S. Choi, Y. Lee, B.-Y. Kim, and S. Watanabe, “TF-GridNet: Making time-frequency domain models great again for monaural speaker separation,” in Proc. IEEE ICASSP, 2023, pp. 1–5. 1

[11] L. Liu, H. Guan, J. Ma, W. Dai, G. Wang, and S. Ding, “A mask free neural network for monaural speech enhancement,” in Proc. Interspeech, 2023, pp. 2468–2472. 1

[12] W. Zhang, K. Saijo, Z.-Q. Wang, S. Watanabe, and Y. Qian, “Toward universal speech enhancement for diverse input conditions,” in Proc. IEEE ASRU, 2023. 1, 2, 3, 4

[13] L. Yang, W. Liu, and W. Wang, “TFPSNet: Time-frequency domain path scanning network for speech separation,” in Proc. IEEE ICASSP, 2022, pp. 6842–6846. 1, 2

[14] Y. Luo, Z. Chen, N. Mesgarani, and T. Yoshioka, “End-to-end microphone permutation and number invariant multi-channel speech separation,” in Proc. IEEE ICASSP, 2020, pp. 6394– 6398. 1

[15] C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi, “Speech enhancement for a noise-robust text-to-speech synthesis system using deep recurrent neural networks,” in Proc. Interspeech, 2016, pp. 352–356. 1, 3

[16] C. K. Reddy et al., “The INTERSPEECH 2020 deep noise suppression challenge: Datasets, subjective testing framework, and challenge results,” in Proc. Interspeech, 2020, pp. 2492– 2496. 1, 3

[17] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer, “An analysis of environment, microphone and data simulation mismatches in robust speech recognition,” Computer Speech & Language, vol. 46, pp. 535–557, 2017. 1, 3

[18] K. Kinoshita, M. Delcroix, T. Yoshioka, T. Nakatani, A. Sehr, W. Kellermann, R. Maas, S. Gannot, and B. Raj, “The REVERB challenge: A common evaluation framework for dereverberation and recognition of reverberant speech,” in Proc. IEEE WASPAA, 2013, pp. 1–4. 1, 3

[19] M. Maciejewski, G. Wichern, E. McQuinn, and J. Le Roux, “WHAMR!: Noisy and reverberant single-channel speech separation,” in Proc. IEEE ICASSP, 2019, pp. 696–700. 1, 3

[20] R. Gu and Y. Zou, “Temporal-spatial neural filter: Direction informed end-to-end multi-channel target speech separation,” arXiv preprint arXiv:2001.00391, 2020. 2

[21] J.-w. Jung, H.-S. Heo, H. Tak, H.-j. Shim, J. S. Chung, B.-J. Lee, H.-J. Yu, and N. Evans, “AASIST: Audio anti-spoofing using integrated spectro-temporal graph attention networks,” in Proc. IEEE ICASSP, 2022, pp. 6367–6371. 2

[22] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in Proc. ICCV, 2021, pp. 10 012– 10 022. 2

[23] H. Zhao, C. Zhang, B. Zhu, Z. Ma, and K. Zhang, “S3T: Selfsupervised pre-training with Swin transformer for music classification,” in Proc. IEEE ICASSP, 2022, pp. 606–610. 2

[24] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, “SwinIR: Image restoration using Swin transformer,” in Proc. ICCV, 2021, pp. 1833–1844. 2

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems, vol. 30, 2017. 2

[26] J. Chen, Q. Mao, and D. Liu, “Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation,” in Proc. Interspeech, 2020, pp. 2642–2646. 2

[27] A. Pandey, B. Xu, A. Kumar, J. Donley, P. Calamia, and D. Wang, “TPARN: Triple-path attentive recurrent network for time-domain multichannel speech enhancement,” in Proc. IEEE ICASSP, 2022, pp. 6497–6501. 3

[28] C. Li et al., “ESPnet-SE: End-to-end speech enhancement and separation toolkit designed for ASR integration,” in Proc. IEEE SLT, 2021, pp. 785–792. 3

[29] Y.-J. Lu, S. Cornell, X. Chang, W. Zhang, C. Li, Z. Ni, Z.- Q. Wang, and S. Watanabe, “Towards low-distortion multichannel speech enhancement: The ESPnet-SE submission to the L3DAS22 challenge,” in Proc. IEEE ICASSP, 2022, pp. 9201–9205. 3

[30] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, “Perceptual evaluation of speech quality (PESQ)—a new method for speech quality assessment of telephone networks and codecs,” in Proc. IEEE ICASSP, vol. 2, 2001, pp. 749–752. 3

[31] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, “An algorithm for intelligibility prediction of time–frequency weighted noisy speech,” IEEE Trans. ASLP., vol. 19, no. 7, pp. 2125–2136, 2011. 3

[32] E. Vincent, R. Gribonval, and C. Févotte, “Performance measurement in blind audio source separation,” IEEE Trans. ASLP., vol. 14, no. 4, pp. 1462–1469, 2006. 3

[33] C. K. Reddy, V. Gopal, and R. Cutler, “DNSMOS P.835: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors,” in Proc. IEEE ICASSP, 2022, pp. 886–890