ADACONV: ADAPTIVE CONVOLUTION FOR FLOW MATCHING-BASED TEXT-TO-SPEECH

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow Matching (FM) is widely used in the field of non-autoregressive text-to-speech (NAR-TTS). For FM-based NAR-TTS, a vector field (VF) needs to be predicted from the current sample state, conditioned on text, speaker information, and the timestep, to transport noise toward a Mel-spectrogram. In the current works, most researchers predict the VF using Transformer, where feed-forward network (FFN) is implemented via 1D convolutional (Conv1D) layers. However, when the training is completed, the kernel weights of Conv1D layers are fixed and cannot adaptively change with different inputs. This leads to high similarity across different frequency bands in the generated spectrograms and the lack of frequency details. To this end, we propose an Adaptive Convolution (AdaConv) block consisting of two components: Cross-Position Kernel Aggregation (CPKA) and Adaptive Gated Kernel Modulation (AGKM). CPKA learns multiple kernel-weight sets with different frequency responses and aggregates them across kernels and positions using Conv1D, thereby increasing kernel weight diversity. As the input evolves across time steps, AGKM computes position-wise gates from the current input to adaptively reweight the contribution of each position. It then extracts global information from the reweighted input and uses it to modulate the aggregated convolutional kernel weights. In this way, the AdaConv block makes the convolutional kernel weights adaptive to the input and helps generate Mel-spectrograms with more frequency details. Then, the AdaConv block is used to implement the FFN in Transformer to predict the VF. Experimental results demonstrate that our method improves both objective and subjective metrics.

1 INTRODUCTION

The text-to-speech (TTS) task typically follows a two-stage pipeline that generates intermediate acoustic representations (Mehta et al., 2024; Wang et al., 2025b) (e.g., speech tokens and Mel-spectrograms) from input text and condition information, followed by waveform synthesis via a neural vocoder. In the stage of acoustic representation, TTS models include two types of approaches: autoregression (AR) and non-autoregression (NAR). AR-TTS approaches (Du et al., 2024; Wang et al., 2025a) generate the speech by sequentially predicting next token, which leads to high inference latency (Chen et al., 2025c; Wang et al., 2025b). Instead, NAR-TTS approaches (Mehta et al., 2024; Zhu et al., 2025; Chen et al., 2025c) enable fast inference through parallel processing and achieve a more favorable balance between synthesis quality and inference efficiency. Hence, NAR-TTS has received widespread attention from researchers. Currently, NAR-TTS mainly fall into two categories: diffusion-based and flow matching-based (FM) methods. Diffusion-based NAR-TTS methods (Ju et al., 2024; Wang et al., 2025b) generate acoustic representations through a gradual transformation from noise to data, but that process requires a large number of iterative inference steps (Mehta et al., 2024). To mitigate the efficiency bottleneck, FM-based methods (Mehta et al., 2024; Chen et al., 2025c; Zhu et al., 2025) are given to train a neural network to predict the vector field (VF), which is crucial for transporting the noise to acoustic representations along straight probability paths (Lipman et al., 2022; Mehta et al., 2024) with fewer inference steps. In practice, the VF prediction is often conditioned on text, speaker information, and the timestep, and is achieved via a Transformer, where the feed-forward network (FFN) is implemented using 1D convolutional (Conv1D) layers.

However, once the training process completes, the kernel weights of Conv1D layers are fixed and cannot adaptively change with different inputs. Moreover, through experiments, we find that such

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

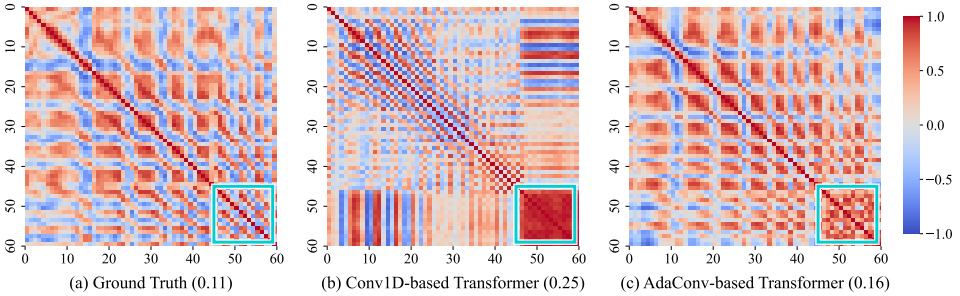


Figure 1: Similarity matrices of frequency bands for Ground Truth, Conv1D-based Transformer, and AdaConv-based Transformer. Parenthesized values (i.e., 0.11, 0.25, 0.16) are the mean similarities across different frequency bands. Conv1D-based Transformer and AdaConv-based Transformer respectively refer to the use of Conv1D layers and AdaConv block to implement FFN in Transformer.

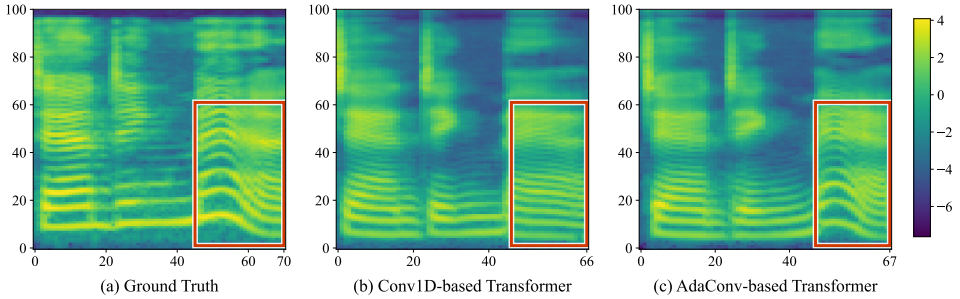


Figure 2: Comparison of frequency details between Ground Truth and generated Mel-spectrograms using the Conv1D-based and AdaConv-based Transformer.

fixed-weight kernels tend to produce high similarity across different frequency bands in generated spectrograms. As illustrated in Fig. 1(b), the spectrogram generated by the Conv1D-based Transformer exhibits a higher inter-band similarity than the ground truth (0.25 vs. 0.11). This elevated inter-band similarity results in the loss of frequency details, as evidenced by the blurred regions highlighted by the red boxes in Fig. 2(b). Inspired by existing input-adaptive and frequency-aware convolution methods (Nam et al., 2022; Chen et al., 2025b), we propose an Adaptive Convolution (AdaConv) block consisting of two components: Cross-Position Kernel Aggregation (CPKA) and Adaptive Gated Kernel Modulation (AGKM). CPKA learns multiple kernel-weight sets with different frequency responses. It then aggregates them across kernels and positions using Conv1D, thereby increasing kernel weight diversity and applying local adjustments at different positions. During FM generation, the sample state evolves across time steps, and the input processed by the FFN changes accordingly. AGKM computes position-wise gates from the current input and uses them to adaptively reweight the contribution of each position. Global information extracted from the reweighted input is subsequently used to modulate the aggregated convolutional kernel weights. Based on the advantages of the AdaConv block, it reduces the similarity across different frequency bands (comparing Fig. 1(b) and Fig. 1 (c)) and generates the spectrogram with more frequency details (comparing Fig. 2 (b) and Fig. 2 (c)). Furthermore, we implement the FFN using the AdaConv block and inject it into Transformer (Vaswani et al., 2017), Conformer (Gulati et al., 2020), and Zipformer (Yao et al., 2024) to predict the VF for the FM-based NAR-TTS task.

The main contributions of this paper are summarized as follows:

- We propose an AdaConv block for FM-based NAR-TTS, consisting of CPKA and AGKM. CPKA increases kernel weight diversity through joint aggregation across kernels and positions, while AGKM adapts position-wise input contributions and the aggregated kernel weights to the current input through gating and global modulation.
- Our proposed AdaConv block enables position-wise input contributions and the aggregated convolutional kernel weights to adapt to input variations across time steps, facilitating high-quality speech generation in FM-based NAR-TTS.
- We implement the FFN using the AdaConv block in Transformer, Conformer, and Zipformer. Experimental results show improvements in objective and subjective metrics, which illustrates the plug-and-play functionality of the AdaConv block.

2 RELATED WORKS

2.1 ADAPTIVE CONVOLUTION KERNEL WEIGHTS

Existing dynamic convolution methods validated the effectiveness of input-adaptive convolutional kernel weights. Related studies (He et al., 2019; Yang et al., 2019; Chen et al., 2020) showed that dynamic convolution improves the representation capacity and adaptability of convolutional neural networks. Thus, some works (He et al., 2019; Chen et al., 2020; Lou & Yu, 2025) learned multiple complete convolutional kernels and aggregated them using routing or attention weights generated from the input. Besides, there were also some works (Nam et al., 2022; Chen et al., 2025b;a) that incorporated frequency information into dynamic convolution, enabling convolutional kernel weights to model different frequency responses. Despite their different implementations, those methods shared a common design: input-dependent coefficients are mainly used on the kernel side, either to combine multiple complete kernels or to modulate kernel weights. When multiple kernels are involved, the aggregation is generally performed at the complete-kernel level.

2.2 FLOW MATCHING-BASED TEXT-TO-SPEECH

FM simplifies the generation process to an efficient linear evolution by constructing straight probability paths between noise and data and learning the VF along those paths (Lipman et al., 2022), which enables transport from noise to data with fewer inference steps. Consequently, FM is suitable for NAR-TTS, as it balances the intricate modeling of continuous acoustic representations with the high-speed inference requirements of speech synthesis (Mehta et al., 2024; Chen et al., 2025c). Some recent works (Mehta et al., 2024; Zhu et al., 2025; Yang et al., 2025) employed different Transformer structures to predict the VF for FM-based NAR-TTS. In the works of Matcha-TTS (Mehta et al., 2024), F5-TTS (Chen et al., 2025c) and ZipVoice (Zhu et al., 2025), the authors adopted Conformer (Gulati et al., 2020), Diffusion Transformer (Peebles & Xie, 2023) and Zipformer (Yao et al., 2024) to predict the VF for generating the speech corresponding to the text and speaker. In those Transformer-based TTS models, the FFN is implemented with Conv1D layers. However, when training is completed, the parameters of Conv1D layers are fixed and cannot be changed with the input, which leads to a lack of frequency details and reduces the quality of speech.

3 METHOD

3.1 ADAPTIVE CONVOLUTION BLOCK

To make the convolutional kernel weights adaptive to the input, we propose the AdaConv block as illustrated in Fig. 3. It contains two components: Cross-Position Kernel Aggregation and Adaptive Gated Kernel Modulation.

3.1.1 CROSS-POSITION KERNEL AGGREGATION

For a Conv1D layer, the weight $\boldsymbol{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k}$ consists of C_{out} convolutional filters, each with a kernel size of k and C_{in} input channels. To enhance the adaptability of the convolutional kernel, CPKA introduces N learnable kernel-weight sets, denoted by $\{\boldsymbol{W}_1, \cdots, \boldsymbol{W}_N\}$, where each kernel has different weights and the size of $\boldsymbol{W}_i (i = 1, \ldots, N)$ is $C_{\text{out}} \times C_{\text{in}} \times k$.

Inspired by frequency-domain weight learning methods (Chen et al., 2025b), CPKA transforms multiple kernel-weight sets into the frequency domain. Mel-spectrograms exhibit strong dependencies among different frequency components (Ren et al., 2022). Motivated by this property, CPKA retains the complete frequency-domain representation of each kernel rather than imposing predefined disjoint frequency groups. It then aggregates the kernel weights across kernels and positions using Conv1D. Accordingly, each Conv1D kernel is transformed into the frequency domain by the one-dimensional discrete Fourier transform, producing $\boldsymbol{W}'_N$ as shown in Equation (1). Then, to ensure a consistent two-dimensional format before stacking the different frequency-domain kernels, we reshape each kernel from a tensor into a parameter matrix $\boldsymbol{W}''_N$, as formulated in Equation (2).

$$\boldsymbol{W}'_N = \mathcal{F}(\boldsymbol{W}_N) \in \mathbb{C}^{C_{\text{out}} \times C_{\text{in}} \times (k/2+1)}, \tag{1}$$

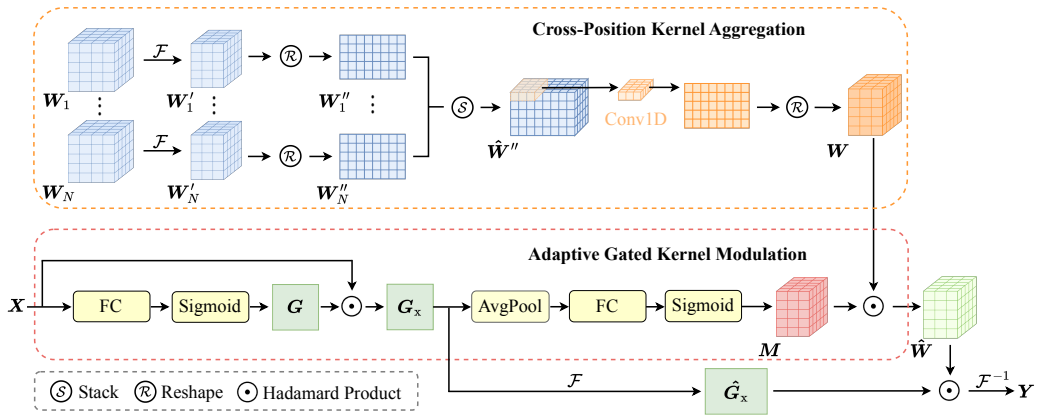


Figure 3: Overview of our proposed **Adaptive Convolution block**. $\boldsymbol{X}$ and $\boldsymbol{Y}$ represent the input and output. $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the one-dimensional discrete Fourier transform and its inverse. $\boldsymbol{W}_1, \cdots, \boldsymbol{W}_N$ are N distinct kernel weights in the Conv1D layer, and FC represents a fully connected layer.

where $\mathcal{F}$ is the one-dimensional discrete Fourier transform.

$$\boldsymbol{W}_N'' = \mathcal{R}(\boldsymbol{W}_N') \in \mathbb{R}^{C_{\text{out}} C_{\text{in}} \times 2(k/2+1)}, \tag{2}$$

where $\mathcal{R}(\cdot)$ refers to the reshape operation that merges the input and output channels, and concatenates the real and imaginary parts along the channel dimension.

Those matrices are subsequently stacked into a tensor $\hat{\boldsymbol{W}}''$, as shown in Equation (3), to support joint aggregation across kernels and positions. Building on the stacked complete frequency-domain representations, CPKA applies a Conv1D layer to jointly aggregate the kernel weights across kernels and positions, modeling the dependencies among different frequency components during kernel construction. The aggregated result is then reshaped into the standard frequency-domain kernel format, denoted by $\boldsymbol{W}$, as formulated in Equation (4).

$$\hat{\boldsymbol{W}}'' = \mathcal{S}(\boldsymbol{W}_1'', \cdots, \boldsymbol{W}_N'') \in \mathbb{R}^{N \times C_{\text{out}} C_{\text{in}} \times 2(k/2+1)}, \tag{3}$$

where $\mathcal{S}(\cdot)$ stacks its inputs along the first dimension.

$$\boldsymbol{W} = \mathcal{R}\Big(\mathrm{C}(\hat{\boldsymbol{W}}'')\Big) \in \mathbb{C}^{C_{\text{out}} \times C_{\text{in}} \times (k/2+1)}, \tag{4}$$

where $\hat{\boldsymbol{W}}''$ is stacked along the first dimension, $\mathrm{C}(\cdot)$ represents a Conv1D layer.

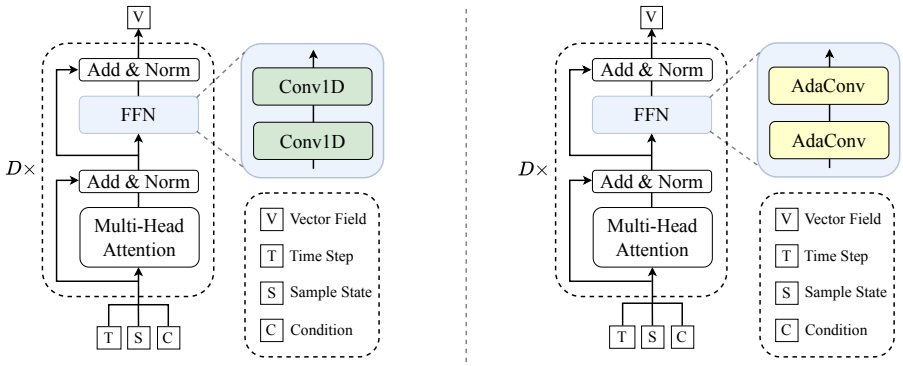
3.1.2 Adaptive Gated Kernel Modulation

The input of the AdaConv block is $\boldsymbol{X} \in \mathbb{R}^{C_{\text{in}} \times T}$, where C_{in} and T are the number of input channels and the length of the input. During FM generation, the sample state evolves from noise toward a Mel-spectrogram over multiple time steps, while the input processed by the FFN varies accordingly. To adapt the convolution to those input variations, AGKM performs input-side adaptation before convolution and global information extraction. The position-wise gating coefficients are computed from the current input through a linear projection followed by a sigmoid activation and are applied element-wise to $\boldsymbol{X}$ to adaptively reweight the contribution of each position, yielding the reweighted input $\boldsymbol{G}_{\text{x}}$, as shown in Equation (5).

$$\boldsymbol{G}_{\text{x}} = \sigma(\mathrm{FC}(\boldsymbol{X}^\top))^\top \odot \boldsymbol{X} \in \mathbb{R}^{C_{\text{in}} \times T}, \tag{5}$$

where $\top$ indicates the transpose operation, $\mathrm{FC}(\cdot)$ is a fully connected layer, $\sigma(\cdot)$ is the sigmoid activation function and $\odot$ represents the Hadamard Product.

Following the position-wise adjustment, the reweighted input $\boldsymbol{G}_x$ is used as both the convolution input and the source of global information for kernel modulation. Temporal average pooling is applied to $\boldsymbol{G}_{\text{x}}$ to obtain the global information $\boldsymbol{g}$, as formulated in Equation (6). The resulting $\boldsymbol{g}$ is then passed through an FC layer followed by a sigmoid activation to compute the modulation coefficients.



(a) Conv1D-based Transformer (b) AdaConv-based Transformer

Figure 4: Comparison of the Conv1D-based and AdaConv-based Transformer for VF prediction. In the AdaConv-based Transformer, the Conv1D layers in each FFN are replaced with the proposed AdaConv blocks. The sample state is the intermediate representation at the current timestep, and the condition includes speaker information and text. D indicates that the dashed-box component in the figure is executed D times.

Those coefficients are broadcast to form $\boldsymbol{M}$ with the same shape as the aggregated kernel weights $\boldsymbol{W}$, as shown in Equation (7). Thus, Equation (8) is employed to modulate the aggregated kernel weights $\boldsymbol{W}$ using $\boldsymbol{M}$ through a Hadamard Product, yielding the kernel weights $\hat{\boldsymbol{W}}$.

$$\boldsymbol{g} = \mathrm{AvgPool}(\boldsymbol{G}_{\mathrm{x}}) \in \mathbb{R}^{C_{\mathrm{in}} \times 1}, \tag{6}$$

where $\mathrm{AvgPool}(\cdot)$ is average pooling.

$$\boldsymbol{M} = \mathcal{V}(\sigma(\mathrm{FC}(\boldsymbol{g}))) \in \mathbb{R}^{C_{\mathrm{out}} \times C_{\mathrm{in}} \times (k/2+1)}, \tag{7}$$

where $\mathcal{V}(\cdot)$ broadcasts the output to match the shape of $\boldsymbol{W}$.

$$\hat{\boldsymbol{W}} = \boldsymbol{M} \odot \boldsymbol{W} \in \mathbb{C}^{C_{\mathrm{out}} \times C_{\mathrm{in}} \times (k/2+1)}. \tag{8}$$

With the modulated kernel weights $\hat{\boldsymbol{W}}$, convolution is performed on the adjusted input $\boldsymbol{G}_{\mathrm{x}}$. Since $\hat{\boldsymbol{W}}$ is represented in the frequency domain, we utilize the one-dimensional discrete Fourier transform to convert $\boldsymbol{G}_{\mathrm{x}}$ into the frequency domain, obtaining $\hat{\boldsymbol{G}}_{\mathrm{x}}$. In this domain, convolution is formulated as a Hadamard Product according to the Convolution Theorem (Jähne, 2005). Accordingly, Equation (9) computes the Hadamard Product between $\hat{\boldsymbol{G}}_{\mathrm{x}}$ and $\hat{\boldsymbol{W}}$, and the result is transformed back to the time domain, yielding $\boldsymbol{Y}$.

$$\boldsymbol{Y} = \mathcal{F}^{-1}(\hat{\boldsymbol{G}}_{\mathrm{x}} \odot \hat{\boldsymbol{W}}) \in \mathbb{R}^{C_{\mathrm{out}} \times T}, \tag{9}$$

where $\mathcal{F}^{-1}$ represents the inverse Fourier transform.

As mentioned earlier, most existing FM-based NAR-TTS models employ Transformer-based backbones for VF prediction, in which the FFNs are implemented using Conv1D layers with fixed weights, as shown in Fig. 4(a). Unlike them, our AdaConv block adaptively reweights position-wise contributions in the input and modulates the aggregated convolutional kernel weights as the input varies across FM time steps, helping reduce inter-band similarity and generate Mel-spectrograms with more frequency details.

3.2 Comparison with Dynamic Convolution Methods

Our proposed AdaConv block differs from existing dynamic convolution methods in two aspects. First, existing methods mainly perform input adaptation on the kernel side through input-dependent kernel combination or modulation (He et al., 2019; Yang et al., 2019; Chen et al., 2020; Nam et al., 2022; Chen et al., 2025b). In contrast, the AdaConv block first adaptively reweights position-wise contributions in the input and then modulates the aggregated convolutional kernel weights as the input varies **across time steps**. Second, CPKA jointly aggregates kernel weights **across kernels and positions** using Conv1D, rather than combining multiple complete kernels with input-dependent coefficients (Yang et al., 2019; Chen et al., 2020; Nam et al., 2022) or directly modulating kernel weights (Chen et al., 2025b).

Table 1: Training configurations for seen-speaker and zero-shot TTS. Trans. and Conf. are Transformer and Conformer

Model	Dataset	Backbone	Epochs	Learning Rate	Warmup	CFG	GPUs
Matcha-TTS	LJSpeech	Trans. / Conf.	800	4×10^{-4}	50 epochs	–	4
Matcha-TTS	VCTK	Trans. / Conf.	800	2×10^{-4}	50 epochs	–	4
Matcha-TTS (SFM)	LJSpeech	Trans. / Conf.	800	4×10^{-4}	50 epochs	–	4
Matcha-TTS (SFM)	VCTK	Trans. / Conf.	800	2×10^{-4}	50 epochs	–	4
ZipVoice	LibriTTS	Zipformer	600	1×10^{-2}	10k steps	0.7	8

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We use LJSpeech and VCTK (Ito & Johnson, 2017; Veaux et al., 2017) as the single-speaker and multi-speaker benchmarks. LibriTTS (Zen et al., 2019) is used as a large-scale multi-speaker corpus. For cross-sentence zero-shot evaluation, we use the LibriSpeech-PC test-clean subset (Meister et al., 2023). It contains 1,127 utterances from unseen speakers, with durations ranging from 4 to 10 seconds.

Implementation Details. All experiments are conducted on NVIDIA GeForce RTX 3090 Ti GPUs with 24 GB of memory. We train all models using the Adam optimizer with dynamic batching. Gradient clipping with a maximum norm of 1.0 is applied to stabilize training. During inference, we solve the probability flow ODE using the Euler method. The generated Mel-spectrograms are converted into waveforms using the Vocos neural vocoder (Siuzdak, 2024). The model-specific training configurations are summarized in Table 1.

Evaluation Metrics. We evaluate model performance using objective and subjective metrics: *(1) Speaker Similarity (SIM and SIM-o):* Speaker embeddings are extracted using WavLM-base-plus-sv (Snyder et al., 2018; Chen et al., 2022), and cosine similarity is computed. SIM compares generated speech with the corresponding ground truth speech, whereas SIM-o (Le et al., 2023) compares it with the original prompt speech. *(2) Word Error Rate (WER):* WER measures content fidelity using transcriptions from Whisper-large-v3 (Radford et al., 2023) and is computed as the normalized edit distance between the ASR transcription of generated speech and reference texts. *(3) UTMOS:* UTMOS (Saeki et al., 2022) predicts perceptual naturalness directly from the waveform. *(4) Comparative MOS (CMOS):* CMOS measures relative naturalness through pairwise comparison with the original target speech, using a 7-point scale from -3 to $+3$. *(5) Similarity MOS (SMOS):* SMOS measures perceived speaker similarity between generated and reference speech. Each pair is rated on a 5-point scale from 1 ("not similar at all") to 5 ("extremely similar"). For CMOS and SMOS evaluation, 20 volunteers rated the same 10 utterances for each model, with model identities hidden and samples presented in random order.

Comparison Methods. We compare the proposed AdaConv block with Matcha-TTS (Mehta et al., 2024), Matcha-TTS (SFM) (Yang et al., 2025), and ZipVoice (Zhu et al., 2025). Both Matcha-TTS and Matcha-TTS (SFM) are evaluated with Transformer (Vaswani et al., 2017) and Conformer (Gulati et al., 2020) backbones, whereas ZipVoice employs a Zipformer backbone (Yao et al., 2024). As a direct comparison, we include the Dynamic Convolution (DConv) block (Chen et al., 2020) , which dynamically aggregates multiple parallel convolutional kernels using input-dependent attention weights. As the official PyTorch implementation is unavailable, we use a third-party implementation and its one-dimensional variant for acoustic representation modeling. For a fair comparison, the DConv block and the AdaConv block replace the same Conv1D layers and use the same kernel size, stride, padding, and input and output dimensions. Moreover, the number of parallel kernels K in the DConv block is set to the same value as the number of kernel-weight sets N in the AdaConv block. All variants follow the same training configuration as their corresponding baseline models.

Model Configuration. Unless otherwise specified, the number of kernel-weight sets (N) in the AdaConv block is set to 8. We replace the Conv1D-based FFNs in all layers of the Transformer and Conformer backbones with the AdaConv-based FFNs. For the 16-layer Zipformer, we apply the AdaConv-based FFN only in the final two layers for lightweight plug-and-play integration.

Table 2: Quantitative comparison of the baseline models with their corresponding DConv block and AdaConv block variants using Transformer and Conformer backbones on LJSpeech and VCTK, and a Zipformer backbone on LibriSpeech-PC test-clean.

Model	SIM / SIM-o ↑	WER ↓	UTMOS ↑	CMOS ↑	SMOS ↑
Transformer trained on LJSpeech					
Ground Truth	1.000	1.779	4.380	0.00	–
Matcha-TTS	0.971	1.613	4.186	−0.43	–
Matcha-TTS + DConv block	0.965	1.654	4.251	−0.45	–
Matcha-TTS + AdaConv block (Ours)	0.972	1.534	4.249	−0.18	–
Matcha-TTS (SFM)	0.973	1.715	**4.257**	−0.20	–
Matcha-TTS (SFM) + DConv block	0.966	1.641	4.152	−0.47	–
Matcha-TTS (SFM) + AdaConv block (Ours)	**0.974**	**1.393**	4.247	**−0.11**	–
Conformer trained on LJSpeech					
Ground Truth	1.000	1.779	4.380	0.00	–
Matcha-TTS	0.965	1.713	4.149	−0.50	–
Matcha-TTS + DConv block	0.962	1.643	4.203	−0.17	–
Matcha-TTS + AdaConv block (Ours)	0.973	1.630	**4.297**	−0.13	–
Matcha-TTS (SFM)	0.971	1.768	4.193	−0.20	–
Matcha-TTS (SFM) + DConv block	0.969	1.660	4.210	−0.16	–
Matcha-TTS (SFM) + AdaConv block (Ours)	**0.973**	**1.432**	4.261	**−0.07**	–
Transformer trained on VCTK					
Ground Truth	1.000	1.557	3.999	0.00	–
Matcha-TTS	0.921	1.534	4.008	−0.28	–
Matcha-TTS + DConv block	0.916	1.558	4.105	−0.25	–
Matcha-TTS + AdaConv block (Ours)	0.921	1.445	4.121	−0.18	–
Matcha-TTS (SFM)	0.918	0.886	4.106	−0.19	–
Matcha-TTS (SFM) + DConv block	0.913	1.052	4.116	−0.23	–
Matcha-TTS (SFM) + AdaConv block (Ours)	**0.923**	**0.863**	**4.131**	**−0.15**	–
Conformer trained on VCTK					
Ground Truth	1.000	1.557	3.999	0.00	–
Matcha-TTS	0.916	1.551	4.005	−0.20	–
Matcha-TTS + DConv block	0.910	1.574	4.100	−0.24	–
Matcha-TTS + AdaConv block (Ours)	0.918	0.928	4.101	−0.20	–
Matcha-TTS (SFM)	0.917	1.121	4.093	−0.19	–
Matcha-TTS (SFM) + DConv block	0.914	1.092	**4.112**	−0.21	–
Matcha-TTS (SFM) + AdaConv block (Ours)	**0.921**	**0.874**	4.092	**−0.16**	–
Zipformer trained on LibriTTS					
Ground Truth	0.694	1.774	4.100	0.00	3.65
ZipVoice	0.608	1.503	4.155	+0.15	**3.62**
ZipVoice + DConv block	0.603	1.476	4.161	+0.13	3.57
ZipVoice + AdaConv block (Ours)	**0.611**	**1.432**	**4.168**	**+0.21**	3.60

4.2 Experimental Results

Overall Performance. Table 2 compares AdaConv with Conv1D and DConv to assess whether its gains arise from dynamic convolution alone. All variants replace the same Conv1D layers and use identical training configurations. For seen-speaker TTS, AdaConv consistently achieves lower WER than both Conv1D and DConv across Transformer and Conformer on LJSpeech and VCTK, while generally maintaining or improving SIM, UTMOS, and CMOS. Although the DConv block reduces WER in several settings, it decreases SIM relative to the Conv1D baseline. For example, for Matcha-TTS (SFM) with the Transformer backbone on LJSpeech, the DConv block reduces WER from 1.715 to 1.641 but decreases SIM from 0.973 to 0.966, whereas the AdaConv block achieves a WER of 1.393 with a SIM of 0.974. For zero-shot TTS, the AdaConv block reduces the WER of ZipVoice by *4.7%* while improving SIM-o, UTMOS, and CMOS. Those results suggest that the gains are not due to dynamic convolution alone, but to the specific combination of position-wise gating and global kernel modulation.

Ablation Study on CPKA. We study the effect of the number of kernel-weight sets (N) in the AdaConv block, as shown in Table 3. The experiments use *Matcha-TTS (SFM)* with Transformer

Table 3: Effect of the kernel-weight set count (N) in the AdaConv block, evaluated using *Matcha-TTS (SFM)* with Transformer and Conformer on LJSpeech and *ZipVoice* with Zipformer on LibriSpeech-PC test-clean; Conv1D is included for comparison.

Design	N	Transformer			Conformer			Zipformer		
		SIM↑	WER↓	UTMOS↑	SIM↑	WER↓	UTMOS↑	SIM-o↑	WER↓	UTMOS↑
Conv1D	–	0.973	1.715	4.257	0.971	1.768	4.193	0.608	1.503	4.155
AdaConv	4	0.973	1.601	4.214	0.972	1.642	4.206	0.610	1.491	4.157
AdaConv	8	**0.974**	1.393	4.247	**0.973**	1.432	**4.261**	0.611	1.432	**4.168**
AdaConv	16	0.974	**1.381**	**4.261**	0.973	**1.431**	4.260	**0.623**	**1.428**	4.167

Table 4: Effect of AGKM on the Conv1D layer and the AdaConv block with $N = 8$, evaluated using *Matcha-TTS (SFM)* with Transformer and Conformer backbones on LJSpeech, and *ZipVoice* with a Zipformer backbone on LibriSpeech-PC test-clean.

Design	N	Transformer			Conformer			Zipformer		
		SIM↑	WER↓	UTMOS↑	SIM↑	WER↓	UTMOS↑	SIM-o↑	WER↓	UTMOS↑
Conv1D	–	0.973	1.715	4.257	0.971	1.768	4.193	0.608	1.503	4.155
Conv1D+AGKM	–	0.974	1.603	**4.259**	0.972	1.614	4.203	0.610	1.487	4.159
AdaConv −AGKM	8	0.972	1.539	4.113	0.971	1.616	4.096	0.607	1.470	4.161
AdaConv	8	**0.974**	**1.393**	4.247	**0.973**	**1.432**	**4.261**	**0.611**	**1.432**	**4.168**

Table 5: Ablation studies of the AdaConv block, evaluated using *Matcha-TTS (SFM)* with Transformer and Conformer backbones on LJSpeech, and *ZipVoice* with a Zipformer backbone on LibriSpeech-PC test-clean.

Design	N	Transformer			Conformer			Zipformer		
		SIM↑	WER↓	UTMOS↑	SIM↑	WER↓	UTMOS↑	SIM-o↑	WER↓	UTMOS↑
Baseline	–	0.973	1.715	**4.257**	0.971	1.768	4.193	0.608	1.503	4.155
AdaConv	8	**0.974**	**1.393**	4.247	**0.973**	**1.432**	**4.261**	**0.611**	**1.432**	**4.168**

and Conformer on LJSpeech, and *ZipVoice* with Zipformer on LibriSpeech-PC test-clean. Increasing N from 4 to 8 improves the objective metrics across backbones. Compared with the Conv1D layer, $N = 8$ reduces WER by *18.8%*, *19.0%*, and *4.7%* for Transformer, Conformer, and Zipformer, exceeding the reductions with $N = 4$ by 12.1, 11.9, and 3.9 percentage points. Increasing N to 16 changes WER by less than 1% and yields only marginal changes in UTMOS, indicating diminishing returns. Thus, $N = 8$ provides the best trade-off between performance and computational efficiency.

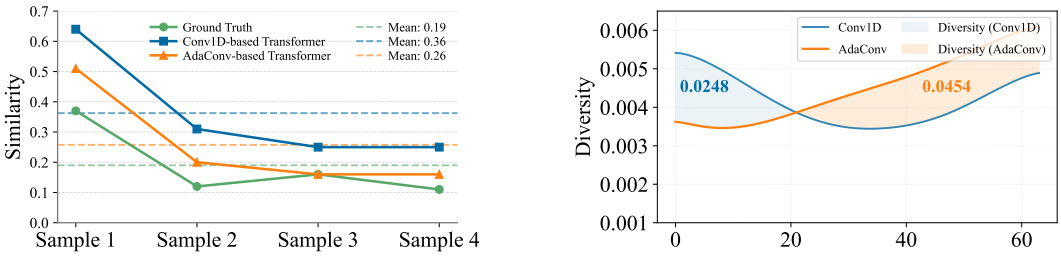
Ablation Study on AGKM. We evaluate AGKM, which reweights position-wise contributions in the input and uses global information from the reweighted input to modulate the aggregated convolutional kernel weights. As shown in Table 4, the Transformer and Conformer experiments use *Matcha-TTS (SFM)* on LJSpeech, while the Zipformer experiment uses *ZipVoice* on LibriSpeech-PC test-clean. Removing AGKM increases WER by *7.0%*, *9.5%*, and *1.1%* for the Conv1D variants with Transformer, Conformer, and Zipformer. For the AdaConv block with $N = 8$, the corresponding relative increases are *10.5%*, *12.8%*, and *2.7%*. Those results confirm the effectiveness of jointly adapting the input and convolutional kernel weights across different backbones. Additional analysis of the gating mechanism is provided in Appendix C.1.

Ablation Study on Adaptive Convolution. Table 5 presents the overall ablation results of the proposed AdaConv block in both seen-speaker and zero-shot TTS tasks across backbones. The AdaConv block shows consistent improvements over the baseline across most evaluation metrics, particularly achieving significant reductions in WER, which demonstrates its effectiveness in improving speech intelligibility. Those gains are associated with Mel-spectrograms generated by AdaConv-based models that contain richer frequency details. Consistent improvements are observed across Transformer, Conformer, and Zipformer, verifying the generality of the proposed method. In addition, the AdaConv block maintains or improves SIM-o and UTMOS, indicating that WER reduction does not compromise speaker similarity or perceptual quality. Overall, AdaConv enhances backbone modeling capability and generalizes well to both seen-speaker and zero-shot TTS.

Analysis of Adaptive Convolution Block. We analyze Mel-spectrograms generated by Conv1D- and AdaConv-based Transformers, along with their kernel diversity, to assess the ability of AdaConv

Table 6: Comparison of TTS model inference speed and block-level relative parameter counts.

Design	Params	Transformer (RTF↓)		Conformer (RTF↓)		Zipformer (RTF↓)	
		GPU	CPU	GPU	CPU	GPU	CPU
Conv1D	1.0×	0.0212	0.1003	0.0302	0.2321	0.1157	7.7938
DConv block ($N = 8$)	8.1×	0.0288	0.1834	0.0462	0.3205	0.1196	8.1565
AdaConv block ($N = 4$)	5.6×	0.0256	0.1692	0.0445	0.2812	0.1179	8.0059
AdaConv block ($N = 8$)	10.9×	0.0300	0.1976	0.0480	0.3306	0.1237	8.3912



(a) Similarity Comparison

(b) Kernel Weights Diversity

Figure 5: (a) Mean similarities across frequency bands for the ground truth and outputs generated by the Conv1D-based Transformer and the Transformer with the AdaConv block across samples. (b) Frequency-domain kernel weight diversity, measured by the standard deviation of normalized frequency responses across channels.

to recover frequency details. For a reliable quantitative analysis, we focus on selected spectrogram regions containing meaningful speech content, avoiding silent intervals and noise artifacts. As shown in Fig. 5(a), the Conv1D-based Transformer exhibits a high mean similarity of *0.36*, which deviates significantly from the Ground Truth (*0.19*) and produces blurred spectral details. In comparison, the AdaConv block yields a much lower mean similarity of *0.26*, aligning more closely with the ground truth spectral distribution. This improvement is attributed to the convolutional kernel weights adaptive to input data in the AdaConv block. Fig. 5(b) illustrates that the AdaConv block achieves a much higher cumulative diversity gain (*0.0454*) compared to the fixed Conv1D layers (*0.0248*). By employing convolutional kernel weights adaptive to input data, the AdaConv block dynamically processes various inputs, effectively mitigating the similarity across different frequency bands inherent in fixed kernel weights. As a result, the synthesized Mel-spectrograms (see App. C.4) exhibit richer frequency details.

Inference Speed and Parameter Count Table 6 compares the block-level parameter counts and system-level real-time factors (RTFs) of Conv1D, DConv, and AdaConv. We evaluate Transformer and Conformer with Matcha-TTS (SFM) on LJSpeech, and Zipformer with ZipVoice on LibriSpeech-PC test-clean. RTF measures Mel-spectrogram generation without vocoder synthesis under identical software and hardware settings (NVIDIA GeForce RTX 3090 Ti and AMD EPYC 7542 CPU). AdaConv has $5.6\times$ and $10.9\times$ the parameters of Conv1D for $N = 4$ and $N = 8$, respectively, but incurs only modest RTF increases. For example, with Zipformer, the GPU RTF increases from 0.1157 to 0.1237 at $N = 8$, with similar trends on Transformer and Conformer. Thus, AdaConv enables input-adaptive kernels while retaining high inference efficiency.

5 Conclusion

In this paper, we propose the AdaConv block for FFNs in FM-based NAR-TTS. It combines CPKA, which learns multiple kernel-weight sets with different frequency responses and aggregates them across kernels and positions, with AGKM, which adaptively reweights position-wise contributions in the input and uses global modulation to adapt the aggregated convolutional kernel weights to the current input. On the Transformer backbone, the AdaConv block reduces inter-band similarity and generates Mel-spectrograms with more frequency details. It also consistently achieves lower WER across Transformer, Conformer, and Zipformer backbones. Experiments demonstrate improvements in objective and subjective metrics. Moreover, the AdaConv block consistently achieves lower WER than the tested DConv block variants, suggesting that its gains are not solely attributable to dynamic convolution. Its effectiveness across different backbones and TTS settings shows broad applicability.

AI USE STATEMENT

Generative AI tools were not used for any tasks requiring disclosure under the ICLR 2027 AI Policy. Their use was limited to language editing aimed at improving the clarity and readability of the manuscript. Any AI-assisted edits were subsequently reviewed and revised by the authors. Responsibility for the final content of the paper, including text edited with the aid of generative AI, rests entirely with the authors.

REFERENCES

Linwei Chen, Lin Gu, and Ying Fu. Frequency-dynamic attention modulation for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22620–22632, 2025a.

Linwei Chen, Lin Gu, Liang Li, Chenggang Yan, and Ying Fu. Frequency dynamic convolution for dense image prediction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 30178–30188, 2025b.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. WavLM: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, 2022.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11030–11039, 2020.

Yushen Chen, Zhikang Niu, Ziyang Ma, Keqi Deng, Chunhui Wang, Jian Zhao, Kai Yu, and Xie Chen. F5-TTS: A fairytaler that fakes fluent and faithful speech with flow matching. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025c.

Zhihao Du, Qian Chen, Shiliang Zhang, Kai Hu, Heng Lu, Yexin Yang, Hangrui Hu, Siqi Zheng, Yue Gu, Ziyang Ma, Zhifu Gao, and Zhijie Yan. CosyVoice: A scalable multilingual zero-shot text-to-speech synthesizer based on supervised semantic tokens. *arXiv preprint arXiv:2407.05407*, 2024.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech*, pp. 5036–5040, 2020.

Junjun He, Zhongying Deng, and Yu Qiao. Dynamic multi-scale filters for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3562–3572, 2019.

Keith Ito and Linda Johnson. The LJ Speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.

Bernd Jähne. *Digital image processing*. Springer, 2005.

Zeqian Ju, Yuancheng Wang, Kai Shen, Xu Tan, Detai Xin, Dongchao Yang, Eric Liu, Yichong Leng, Kaitao Song, Siliang Tang, Zhizheng Wu, Tao Qin, Xiangyang Li, Wei Ye, Shikun Zhang, Jiang Bian, Lei He, Jinyu Li, and Sheng Zhao. NaturalSpeech 3: Zero-shot speech synthesis with factorized codec and diffusion models. In *International Conference on Machine Learning (ICML)*, 2024.

Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, and Wei-Ning Hsu. Voicebox: Text-guided multilingual universal speech generation at scale. In *NeurIPS*, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Meng Lou and Yizhou Yu. Overlock: An overview-first-look-closely-next convnet with context-mixing dynamic kernels. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 128–138, 2025.

Shivam Mehta, Ruibo Tu, Jonas Beskow, Éva Székely, and Gustav Eje Henter. Matcha-TTS: A fast TTS architecture with conditional flow matching. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 11341–11345, 2024.

Aleksandr Meister, Matvei Novikov, Nikolay Karpov, Evelina Bakhturina, Vitaly Lavrukhin, and Boris Ginsburg. Librispeech-pc: Benchmark for evaluation of punctuation and capitalization capabilities of end-to-end asr models. In *2023 IEEE automatic speech recognition and understanding workshop (ASRU)*, pp. 1–7. IEEE, 2023.

Hyeonuk Nam, Seong-Hu Kim, Byeong-Yun Ko, and Yong-Hwa Park. Frequency dynamic convolution: Frequency-adaptive pattern recognition for sound event detection. *INTERSPEECH*, 2022.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pp. 28492–28518. PMLR, 2023.

Yi Ren, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu. Revisiting over-smoothness in text to speech. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8197–8213, 2022.

Takaaki Saeki, Detai Xin, Wataru Nakata, Tomoki Koriyama, Shinnosuke Takamichi, and Hiroshi Saruwatari. UTMOS: utokyo-sarulab system for voicemos challenge 2022. In *Interspeech*, pp. 4521–4525, 2022.

Hubert Siuzdak. Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis. In *International Conference on Learning Representations (ICLR)*, 2024.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5329–5333. IEEE, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, 2017.

Christophe Veaux, Junichi Yamagishi, and Kirsten MacDonald. CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit. *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*, 2017.

Xinsheng Wang, Mingqi Jiang, Ziyang Ma, Ziyu Zhang, Songxiang Liu, Linqin Li, Zheng Liang, Qixi Zheng, Rui Wang, Xiaoqin Feng, Weizhen Bian, Zhen Ye, Sitong Cheng, Ruibin Yuan, Zhixian Zhao, Xinfa Zhu, Jiahao Pan, Liumeng Xue, Pengcheng Zhu, Yunlin Chen, Zhifei Li, Xie Chen, Lei Xie, Yike Guo, and Wei Xue. Spark-TTS: An efficient llm-based text-to-speech model with single-stream decoupled speech tokens. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025a.

Yuancheng Wang, Haoyue Zhan, Liwei Liu, Ruihong Zeng, Haotian Guo, Jiachen Zheng, Qiang Zhang, Xueyao Zhang, Shunsi Zhang, and Zhizheng Wu. Maskgct: Zero-shot text-to-speech with masked generative codec transformer. In *ICLR*, 2025b.

Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in neural information processing systems*, 32, 2019.

Dong Yang, Yiyi Cai, Yuki Saito, Lixu Wang, and Hiroshi Saruwatari. Shallow flow matching for coarse-to-fine text-to-speech synthesis. *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

Zengwei Yao, Liyong Guo, Xiaoyu Yang, Wei Kang, Fangjun Kuang, Yifan Yang, Zengrui Jin, Long Lin, and Daniel Povey. Zipformer: A faster and better encoder for automatic speech recognition. *International Conference on Learning Representations (ICLR)*, 2024.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. LibriTTS: A corpus derived from LibriSpeech for text-to-speech. In *Proc. Interspeech*, pp. 1526–1530, 2019.

Han Zhu, Wei Kang, Zengwei Yao, Liyong Guo, Fangjun Kuang, Zhaoqing Li, Weiji Zhuang, Long Lin, and Daniel Povey. Zipvoice: Fast and high-quality zero-shot text-to-speech with flow matching. *2025 IEEE automatic speech recognition and understanding workshop (ASRU)*, 2025.

A STATEMENTS

Limitations. Our experiments show that the AdaConv block helps FM-based NAR-TTS generate Mel-spectrograms with more frequency details, but its applicability to other acoustic representations, such as speech tokens and raw waveforms, remains unverified. Moreover, AdaConv introduces additional parameters and inference overhead when extensively applied across the backbone. Its plug-and-play nature, however, enables a flexible trade-off between computational cost and synthesis performance by selectively replacing only a subset of Conv1D-based FFNs.

Ethics Statement. Text-to-speech systems can support beneficial applications such as human–computer interaction, accessibility, and content creation, while also posing potential risks related to impersonation, misinformation, and speech generation without a speaker's consent. These risks highlight the importance of responsible development and use of speech generation technologies. The methods and experiments in this work are presented for research purposes. For real-world deployment, we encourage the adoption of appropriate safeguards and consent mechanisms, as well as compliance with applicable data licenses and privacy requirements.

Reproducibility Statement. Information supporting the reproducibility of the proposed method and experimental results is provided throughout the main paper and appendix. The main paper describes the model architecture and configurations, training objective and settings, experimental setup, datasets, evaluation metrics, and comparison methods. Additional theoretical derivations, analyses, and evaluation details are provided in the appendix. These materials are intended to facilitate independent reproduction and verification of the results reported in this work.

B FM-BASED NAR-TTS BACKBONES

As illustrated in Fig. 6, our zero-shot FM-based NAR-TTS model consists of a Zipformer text encoder and a Zipformer VF predictor equipped with the AdaConv block. During training, the encoded text is average-upsampled to form the text condition, and the VF predictor is trained with the time step, noisy speech, and masked speech condition using the FM objective. During inference, the predictor is conditioned on the prompt speech and text to guide an ODE solver to transport Gaussian noise toward the target Mel-spectrogram. For computational efficiency, we replace the Conv1D-based FFNs only in the final two layers of the 16-layer predictor with AdaConv-based FFNs.

B.1 ADACONV-BASED CONFORMER BACKBONE

Fig. 7 illustrates the adaptation of the Conformer structure for VF prediction in the seen-speaker task. In this configuration, the Conv1D layers within the FFN are replaced by AdaConv blocks, forming the AdaConv-based Conformer. As depicted in the figure, the sample state represents the intermediate representation of the sample as it is transported from noise to data, while the condition comprises speaker information and text. This modification makes the kernel weights adaptive to the input and helps generate Mel-spectrograms with more frequency details than the fixed-weight Conv1D baseline. Unlike the zero-shot setting, where speaker information is extracted from a prompt utterance, speaker information is directly available in the seen-speaker setting.

C ANALYSIS OF THE ADACONV BLOCK

C.1 INPUT-ADAPTIVE ADJUSTMENT VIA POSITION-WISE GATING

To provide further insight into the position-wise gating mechanism in the AdaConv block, we analyze the input reweighting process in Equation (5). Rather than directly applying convolution to the input $\boldsymbol{X}$, AGKM computes position-wise gates $\sigma(\mathrm{FC}(\boldsymbol{X}^\top))^\top$ from the current input and uses them to adaptively reweight the contribution of each position. For the analyses in this appendix, we refer to this gating-based input reweighting as input-adaptive adjustment (IAA). IAA is not an additional component of the AdaConv block but denotes the input-side adjustment performed by AGKM. We analyze IAA in terms of input stability and robustness to controlled input noise.

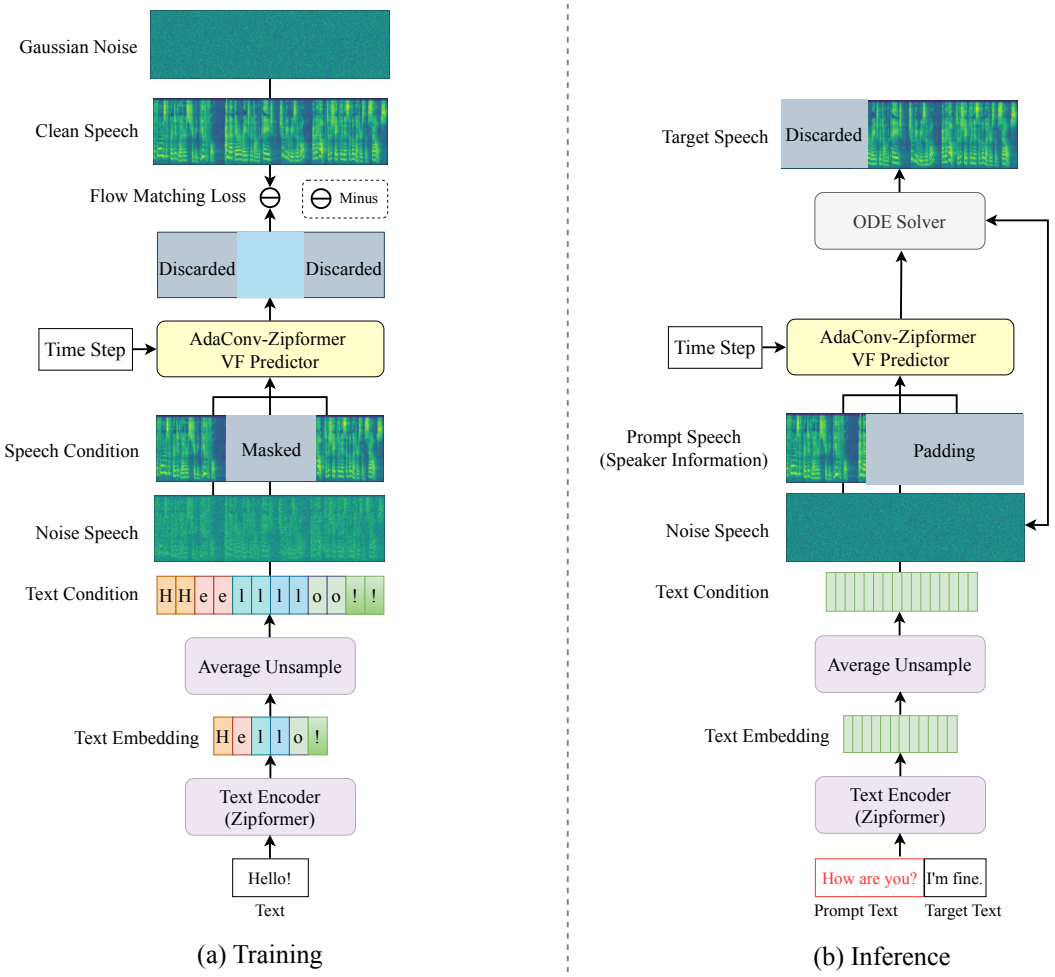
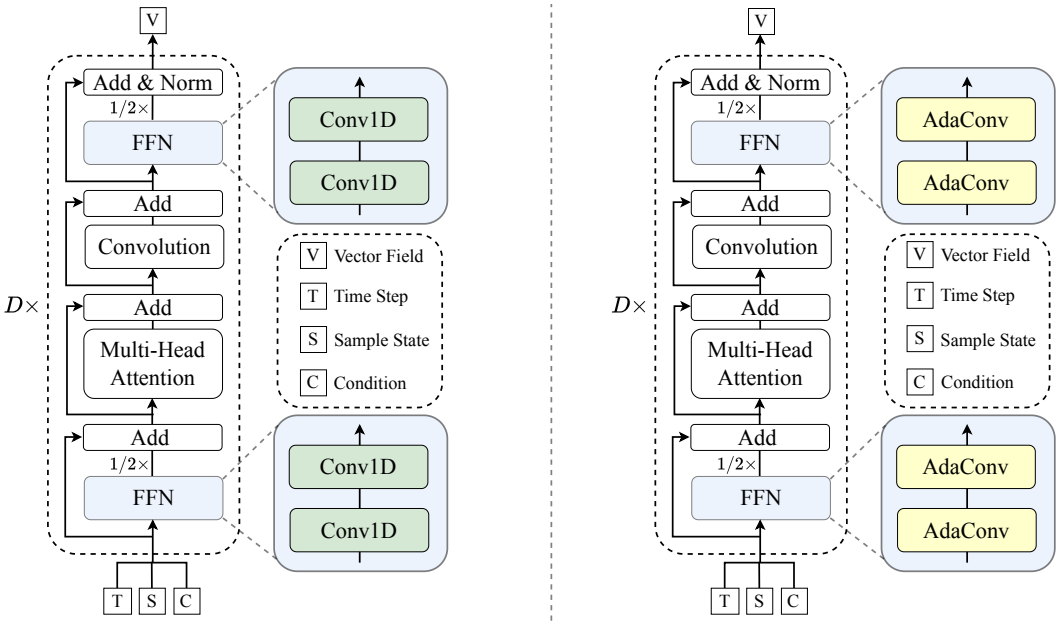


Figure 6: Overall architecture of the AdaConv-based Zipformer for FM-based NAR-TTS model.

Stabilizing Input Variations. As illustrated in Fig. 8, we analyze whether IAA stabilizes the input representations processed by the subsequent convolution operation. For each sample, the frame-wise mean is first computed by averaging the input values across channels at each frame. We then apply a sliding window along the temporal dimension and calculate the local variance within each window to measure local variability. The trajectories in Fig. 8 show that the original inputs (green lines) contain stronger frame-level fluctuations, whereas the IAA-adjusted inputs (orange lines) exhibit smoother trajectories across different samples. Specifically, column (a) shows that IAA suppresses rapid variations in the frame-wise mean, while column (b) shows that it reduces the sliding-window variance. These results indicate that the position-wise gates adapt the contribution of each position according to the current input and provide more stable representations for the subsequent convolution operation.

Noise-Robust Distribution Alignment. We further evaluate whether IAA can suppress perturbations in the input representations. Controlled noise is added to the original input, and the channel-wise mean and variance are computed by averaging the feature values over all temporal frames. Noise Shift measures the relative deviation of noisy statistics from clean references; the upward arrow (↑) indicates the direction of increase, not that larger values are preferable. As shown by the kernel density estimation curves in Fig. 9, the injected noise shifts and broadens the mean and variance distributions (orange lines) relative to the clean references (blue lines), as also reflected by the annotated Noise Shift values. After IAA, the adjusted distributions (green lines) move closer to the clean distributions across different samples. Those results indicate that the position-wise gating mechanism in AGKM adaptively reweights the contribution of each position under input perturbations, producing cleaner and more stable representations for the subsequent convolution operation.



(a) Conv1D-based Conformer

(b) AdaConv-based Conformer

Figure 7: Structural comparison between Conv1D-based and AdaConv-based Conformers for VF prediction. (a) The baseline model using Conv1D layers. (b) The proposed AdaConv-based variant, where the Conv1D layers within the FFN are replaced by AdaConv blocks. The sample state is the intermediate representation at the current time step, while the condition includes speaker information and text. The symbol D indicates that the component within the dashed box is stacked D times.

C.2 Kernel Frequency Responses and Diversity

We analyze the frequency responses and kernel weight diversity of the Conv1D layer and the AdaConv block. The frequency response reflects the filtering behavior of kernel weights in the frequency domain, while kernel weight diversity measures variation in normalized frequency responses across output channels. As shown in Fig. 5(b), the AdaConv block achieves higher cumulative kernel weight diversity than Conv1D (0.0454 vs. 0.0248), indicating that its kernel weights are more diverse and less redundant. Fig. 10 further presents the sample-wise frequency responses and kernel weight diversity. The Conv1D baseline has fixed kernel weights and thus exhibits the same frequency response across inputs, whereas the AdaConv block adapts its frequency response to the current input. Those results show that the AdaConv block provides input-adaptive frequency responses and improves kernel weight diversity.

C.3 Band Similarity and Kernel Weight Diversity

We analyze inter-band similarity and kernel weight diversity using selected spectrogram regions containing speech content, excluding silent intervals and noise artifacts. As shown in Fig. 5(a), the Conv1D-based Transformer has a mean inter-band similarity of *0.36*, which is much higher than that of the ground truth (*0.19*). The Transformer equipped with the AdaConv block reduces the mean similarity to *0.26*, bringing it closer to the ground truth. Meanwhile, Fig. 5(b) shows that the AdaConv block achieves higher cumulative kernel weight diversity than Conv1D (*0.0454* vs. *0.0248*). Those results are consistent with the observation that higher kernel weight diversity is associated with lower inter-band similarity.

C.4 Frequency Details in Mel-Spectrograms

To analyze inter-band similarity, we construct a frequency correlation matrix by computing the Pearson correlation coefficient between every pair of frequency bins over all temporal frames. Red re-

gions in the matrix indicate stronger inter-band correlations, whereas blue regions indicate more independent spectral structures. We further calculate the mean inter-band similarity by averaging the correlations between frequency-band pairs with sufficient separation, thereby reducing the influence of local smoothness near the diagonal. A higher score indicates greater spectral redundancy. As illustrated in Fig. 11, the Conv1D-based Transformer produces higher inter-band similarity and more blurred harmonic structures than the ground truth. For example, the similarity of Sample 1 increases from *0.37* for the ground truth to *0.64* for the Conv1D-based Transformer, while the AdaConv block reduces it to *0.51*. For Sample 3, the similarity decreases from *0.25* with Conv1D to *0.16* with the AdaConv block. This lower inter-band similarity corresponds to clearer harmonic variations and more frequency details in the generated Mel-spectrograms.

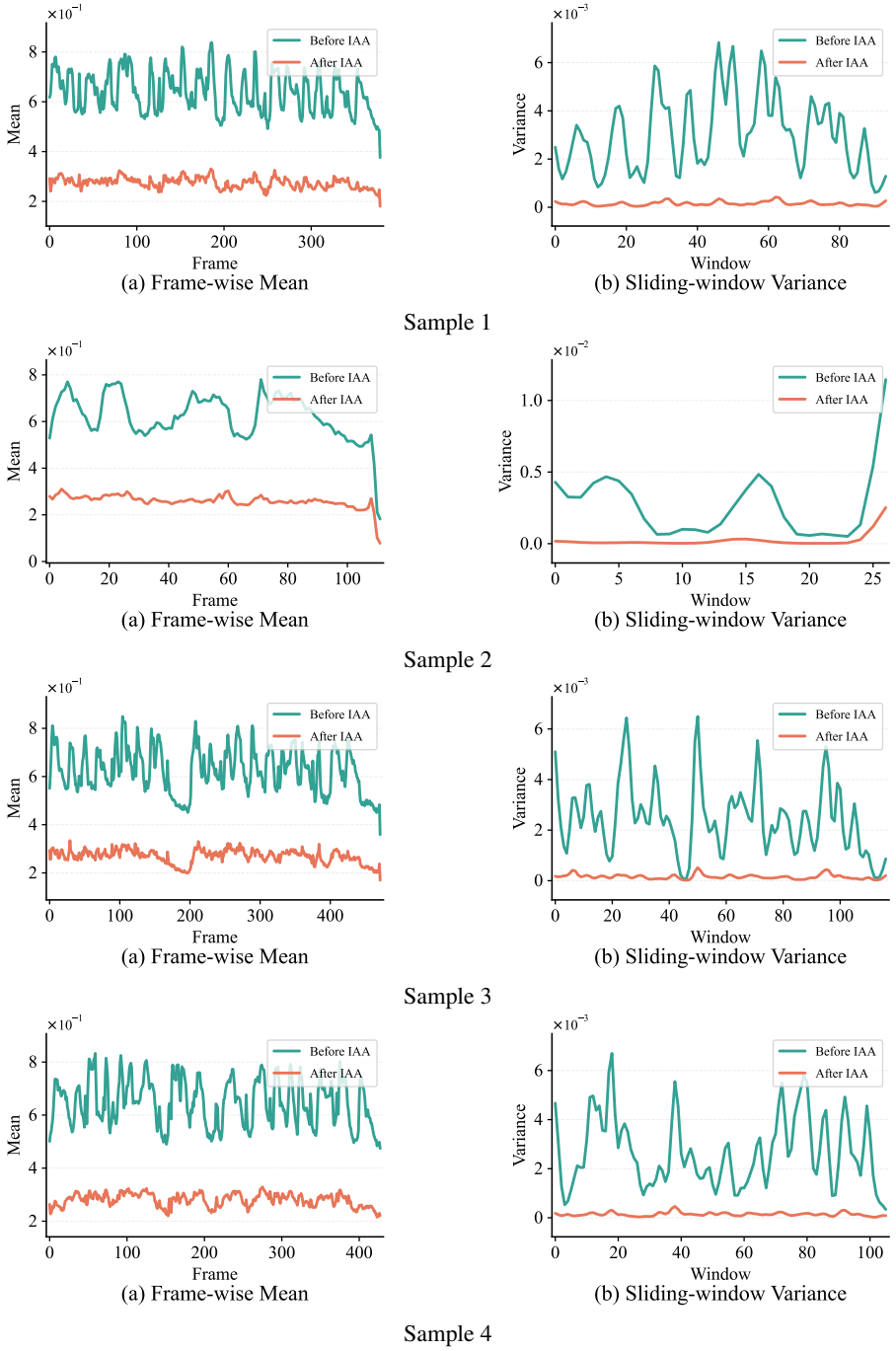


Figure 8: Stability comparison before and after input-adaptive adjustment (IAA). IAA denotes the input reweighting induced by the position-wise gating mechanism in AGKM. (a) Frame-wise mean of the input representations. (b) Sliding-window variance.

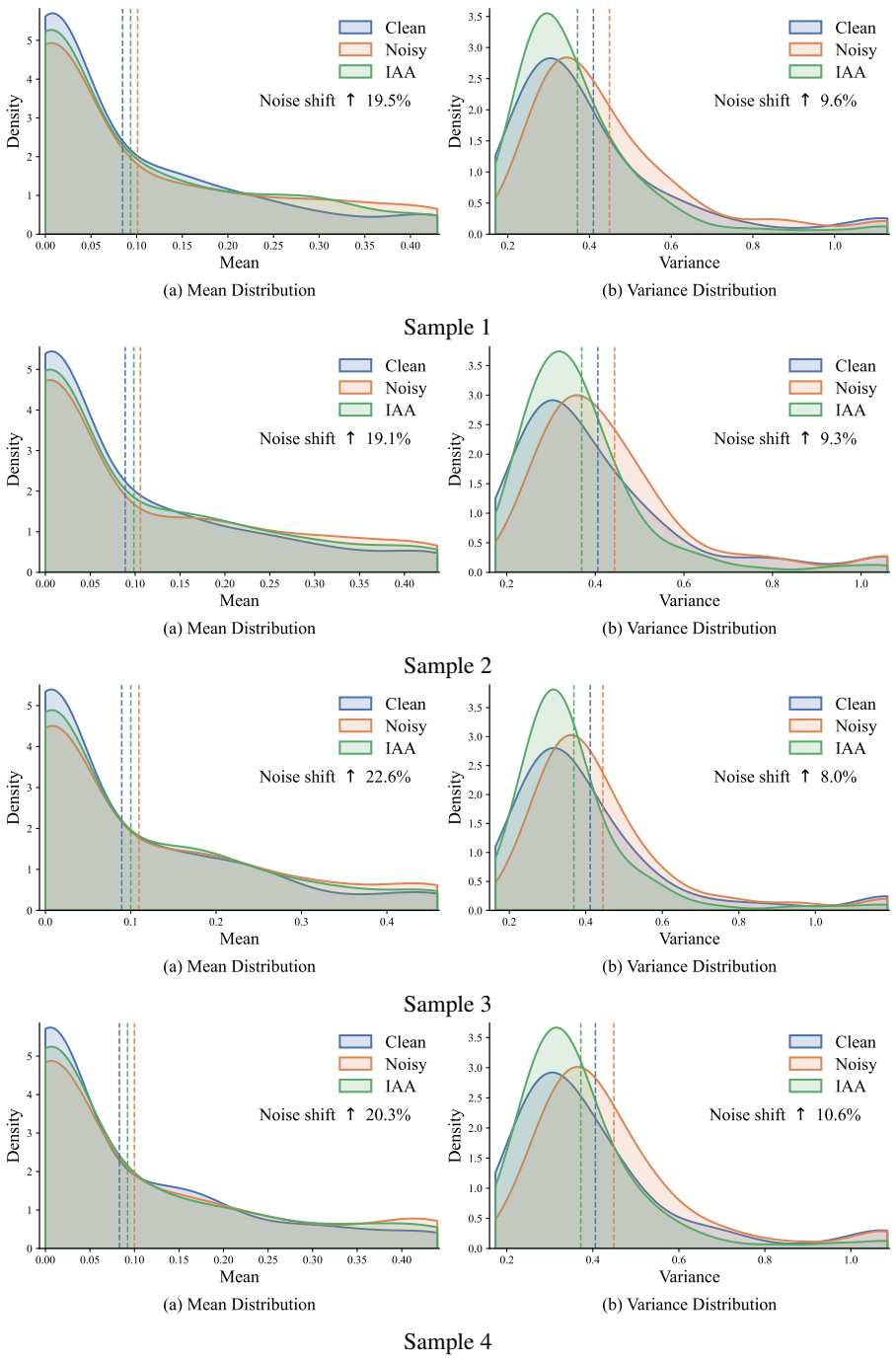
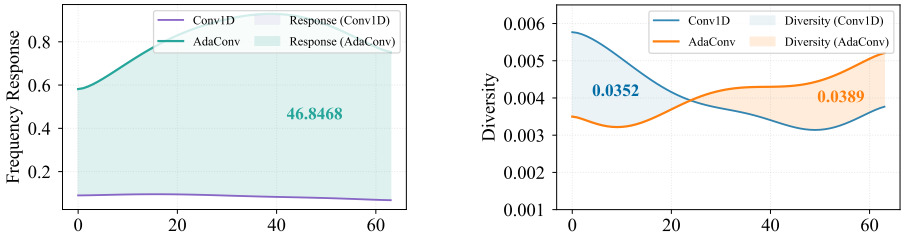
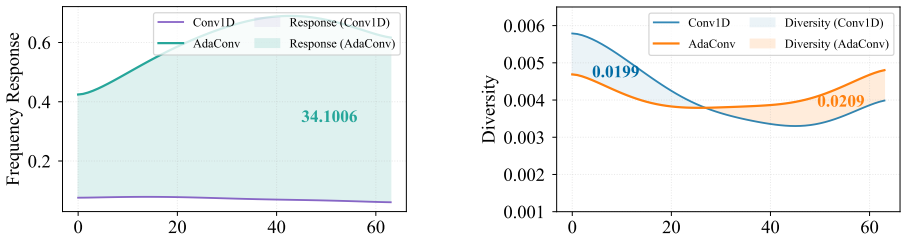


Figure 9: Distributions of channel-wise statistics for clean, noisy, and IAA-adjusted inputs. (a) Channel-wise mean distributions. (b) Channel-wise variance distributions. Controlled noise is added to the input representations, and IAA shifts the noisy distributions toward the clean references.



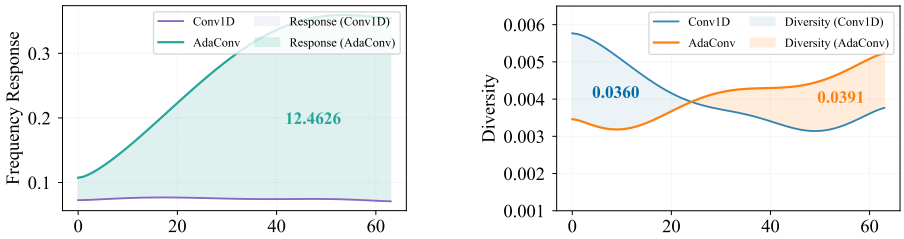
(a) Kernel Weights Frequency Response (b) Kernel Weights Diversity

Sample 1



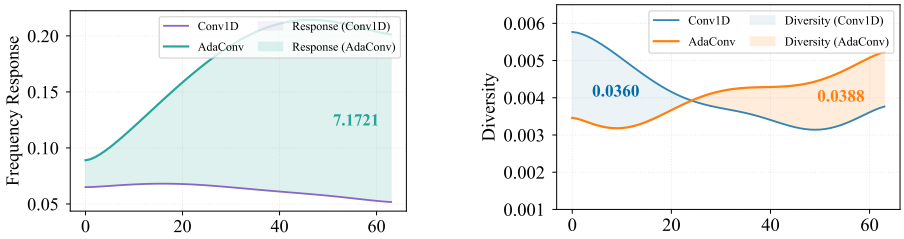
(a) Kernel Weights Frequency Response (b) Kernel Weights Diversity

Sample 2



(a) Kernel Weights Frequency Response (b) Kernel Weights Diversity

Sample 3



(a) Kernel Weights Frequency Response (b) Kernel Weights Diversity

Sample 4

Figure 10: Frequency responses and kernel weight diversity across input samples. The left column shows the frequency responses of the Conv1D and AdaConv kernel weights, while the annotated values denote the cumulative response gain of AdaConv over Conv1D across frequency positions. The right column shows kernel weight diversity in the frequency domain, measured by the standard deviation of normalized frequency responses across output channels. The plots compare the Conv1D-based and AdaConv-based Transformers.

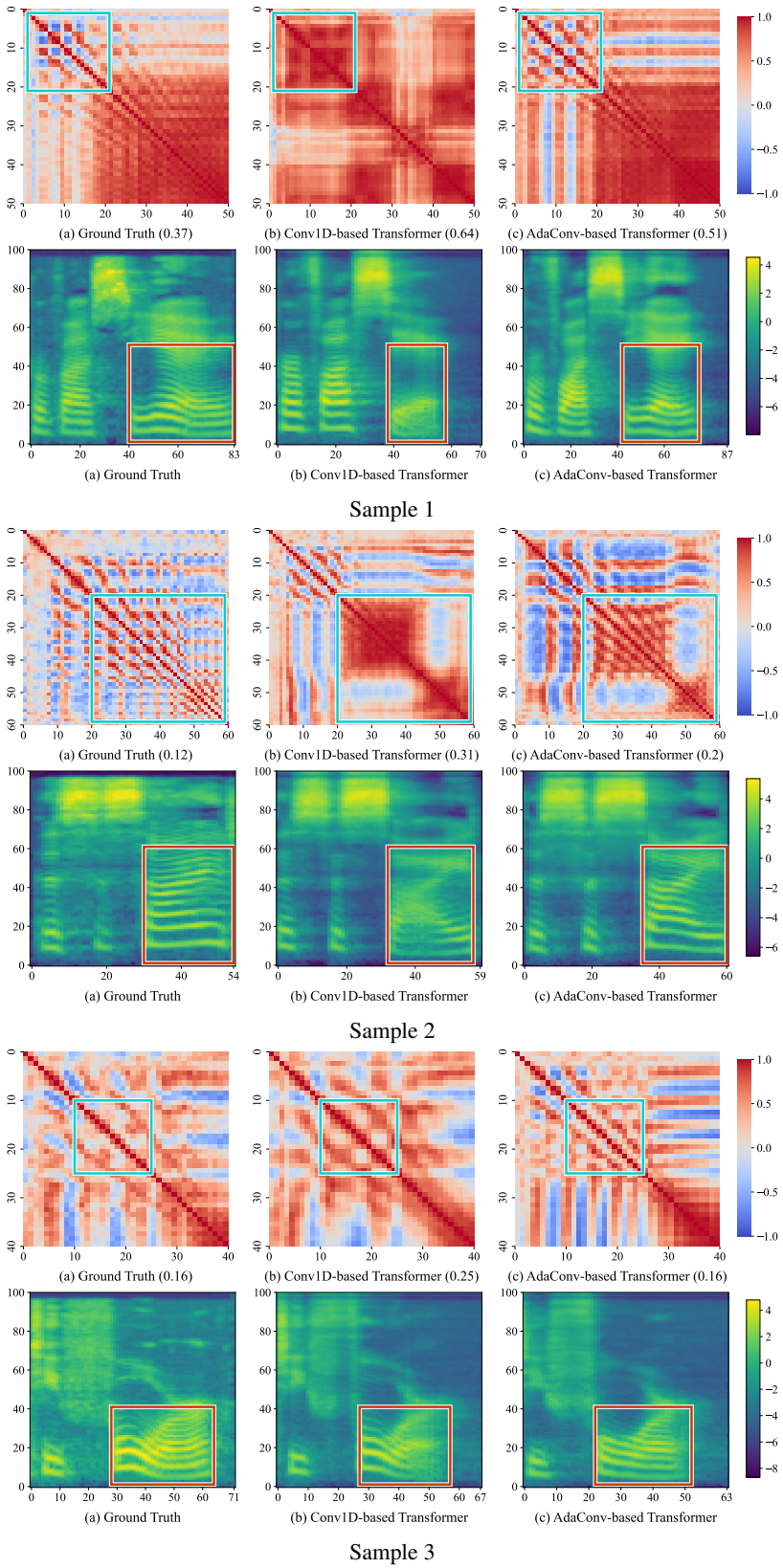


Figure 11: Comparison of frequency similarity matrices (top) and Mel-spectrograms (bottom) for different samples, featuring Ground Truth, Conv1D-based Transformer, and AdaConv-based Transformer. Parenthesized values are mean inter-band similarity.