INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

# Improving WaveRNN with Heuristic Dynamic Blending for Fast and High-Quality GPU Vocoding

*Muyang Du*[1], *Chuan Liu*[1], *Jiaxing Qi*[1], *Junjie Lai*[1]

[1]NVIDIA Corporation

{myrond,riverl,jqi,julienl}@nvidia.com

## Abstract

Auto-regressive vocoders are typically less efficient at inference due to their serial nature, making it difficult to fully utilize graphics processing units (GPUs). In this context, batched inference with upsampled feature folding can be used to speed up vocoding. However, speech quality degradation caused by blending folded waveform segments making it hard to be applied to production. To address this issue, we propose a novel blending approach called heuristic dynamic blending (HDB), which effectively addresses the voice trembling and echo artifacts of conventional static blending. We also propose a parallel algorithm of HDB running on GPUs, which significantly reduces the additional time overhead introduced by the naive HDB algorithm. Experimental results demonstrate that using a multi-band WaveRNN with HDB can effectively improve parallelism for real-time GPU vocoding while maintaining high speech quality comparable to non-folding inference.

**Index Terms**: neural vocoder, speech synthesis, real-time, high-quality, WaveRNN, GPU vocoding.

## 1. Introduction

In recent years, neural network-based approaches have been widely applied to speech synthesis tasks such as text-to-speech (TTS) and voice cloning and have seen extraordinary success. The acoustic model[1, 2, 3] and vocoder are two essential components of these tasks. Among them, vocoder converts acoustic features, such as mel-spectrogram, generated by the acoustic model into waveform samples. There are two main types of neural network-based vocoders: autoregressive vocoders (AR), such as Conv-based WaveNet[4] and RNN-based WaveRNN[5], LPCNet[6], and non-autoregressive vocoders (Non-AR), such as Flow-based WaveGlow[7], GAN-based MelGAN[8], HiFi-GAN[9], and Diffusion-based DiffWave[10], WaveGrad[11].

AR vocoder typically requires upsampling the acoustic feature to the same resolution as the waveform and then generating one sample in each auto-regressive iteration based on the conditioning features and previously generated samples. In contrast, Non-AR vocoders such as HiFi-GAN perform a fully parallel conversion utilizing a stack of upsampling and convolution blocks. Benefiting from the natural property of Non-AR vocoders, it can fully use the computing capability of GPUs to achieve ultra-fast vocoding. However, Non-AR vocoders consume significantly more time and computational cost for training and have higher quality requirements for acoustic features. AR vocoders are easier to train and more robust to the imperfect synthetic acoustic features because of the randomness introduced by the sampling, but they are less efficient at inference.

Several methods have been proposed to speed up the vocoding of AR vocoders, such as synthesizing multiple sub-band samples[12, 13] (the multi-band strategy), synthesizing multiple consecutive samples simultaneously at each iteration[14], and sparsifying[15, 16] the model weights to reduce computational costs. However, these approaches still maintain the serial nature of the entire acoustic feature vocoding process, making it difficult to fully utilize GPUs. To further improve parallelism, a simple but effective approach is folding the upsampled conditioning features into multiple feature segments for batched vocoding and then blending the overlap area of waveform segments to get the final waveform. However, this approach suffers from observable quality degradation with the conventional static blending algorithm. In this paper, we propose a novel blending approach to improve the synthetic quality of folding inference to make it fully applicable for production. The main contributions are summarized as follows:

- We propose a novel blending algorithm called heuristic dynamic blending (HDB) for batched inference with feature folding, which effectively addresses the voice trembling and echo artifacts of the traditional static blending.
- We further propose a parallel algorithm of HDB running on GPUs, which significantly reduces the additional time overhead introduced by the naive HDB algorithm.
- We conduct multiple experiments to assess the effects of different blending and model configurations on speech quality. Results demonstrate that using a multi-band WaveRNN with HDB can vocode in real-time with speech quality comparable to non-folding inference.

## 2. Related Work

### 2.1. Multi-Band WaveRNN with $\mu$-law Companding

Figure 1 presents the multi-band WaveRNN used to demonstrate our proposal. It contains an upsampling network and an iterative network. The upsampling network $U$ takes frame-rate acoustic features as input and produces sub-band sample-rate conditional features for the iterative network. $U$ generates both the upsampled Mel feature with a stack of upsample (nearest interpolation + convolution) blocks, and the upsampled supplementary feature with a stack of residual blocks[17] followed by one single interpolation layer. The iterative network $I$ contains a pre-linear layer, two stacked GRU[18] layers, three post-linear layers, and a final sampling layer to auto-regressively generate 9-bit $\mu$-law[19] quantized sub-band samples extracted by Pseudo-QMF (PQMF)[20] analysis. In each iteration, $I$ generates $B$ sub-band samples simultaneously from a multinomial distribution.

During training, a fixed-length Mel fragment $M$ and its corresponding quantized sub-band signal fragment $y$ are used to jointly optimize the $U$ and $I$ using cross entropy as the objective function, as shown in the following equation:

$$\min_{\theta,\delta} -\sum_{n=1}^{N}\sum_{b=1}^{B}\sum_{c=1}^{C} y_{n,b,c}\log(I(U(M;\theta);\delta)_{n,b,c}) \quad (1)$$



10.21437/Interspeech.2023-690

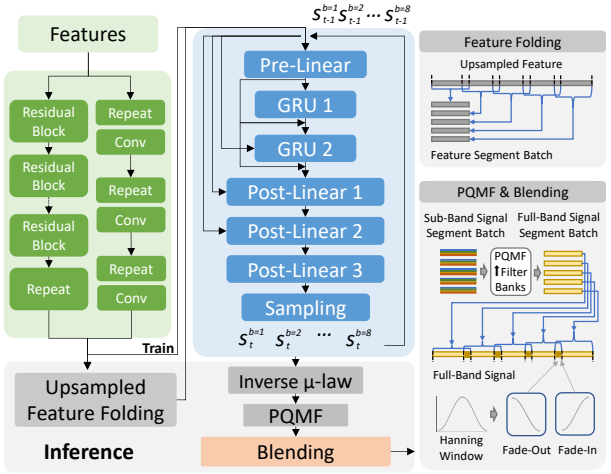


Figure 1: *Overview of the Multi-Band WaveRNN with μ-law, upsampled feature folding, and signal segments blending.*

where $\theta$ and $\delta$ are the trainable parameters of the upsampling network and iterative network, respectively. $C$ denotes the number of classes for sampling, which is set to 512 for the 9-bit output. $B$ represents the number of sub-bands, and $N$ indicates the length of the sub-band signal fragment used for training.

# 3. Method

## 3.1. Batched Inference by Feature Folding

As shown in Figure 1, a simple approach to improve vocoding parallelism is to fold the upsampled features into multiple segments of the same length (a feature segment batch), then perform batched auto-regression on the iterative network. For simplicity, we refer to this approach as the folding inference. During folding, an overlap area is required between adjacent segments. Once the auto-regression, inverse $\mu$-law, and PQMF synthesis are complete, the full-band signal segments are recovered to the complete signal by blending the reserved overlap areas. With folding applied, the number of vocoding iterations is no longer related to the length of the upsampled feature but depends solely on the configurable segment length. This can significantly reduce the real-time factor on GPU, especially when synthesizing audio with a long duration.

## 3.2. Conventional Static Blending and Limitations

Define the segment length and the overlap length used for folding the upsampled feature as $\hat{SL}$ and $\hat{OL}$ (with the requirement of $2\hat{OL} \leq \hat{SL}$). A straightforward approach is to use the same overlap for blending as for folding, which we call it as the static blending. Note that we apply blending to the full-band signal segments after PQMF synthesis, so the segment and overlap length for blending are $SL\ (= \hat{SL} \cdot B)$ and $OL\ (= \hat{OL} \cdot B)$, respectively. To avoid audible discontinuities in the overlap area after blending, we use the Hanning window to generate coefficients for fade-in $\alpha$ and fade-out $\beta$ as:

$$w[j] = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi j}{2OL-1}\right)\right],\ j \in [0, 2OL) \qquad (2)$$
$$\alpha = (w[j])_{\ 0 \leq i \leq OL},\ \beta = (w[j])_{\ OL \leq j < 2OL}$$

Then the full-band signal segment $S_i$ and $S_{i+1}$ is blended as:

$$S_{i+1,j} = \beta_j * S_{i,-OL+j} + \alpha_j * S_{i+1,j},\ j \in [0, OL) \qquad (3)$$

According to our observations, static blending can effectively reduce the appearance of crackling noise in the overlap area when $\hat{OL}$ is set at a sufficient length. However, we notice that audio constructed using static blending has observable voice trembling and echo artifacts. These artifacts can become more noticeable when synthesizing higher sample-rate audio. We attribute these problems to misalignment in the overlap area of two adjacent segments. In other words, the overlap area $(S_{i+1,j})_{0 \leq j \leq OL}$ is either a *time-advanced* or *time-delayed* version of $(S_{i,j})_{-OL \leq j \leq -1}$. This is due to the superposition of: 1. the inaccurate calculation of the GRU states without long-term past dependencies after folding, and 2. the randomness introduced by multinomial sampling. While increasing $\hat{OL}$ can alleviate voice trembling, it makes the echo artifact more noticeable and reduces the ratio of valid payload in each segment.

## 3.3. Heuristic Dynamic Blending

To address the limitations of static blending, we propose a novel approach called heuristic dynamic blending (HDB) that blends adjacent segments using a dynamic overlap length. We define our proposal as a heuristic algorithm because it only produces locally optimal overlap, but experiments have shown it can effectively solve the artifacts of voice trembling and echo. This makes HDB a simple and effective solution for improving the parallelism of auto-regressive vocoders such as WaveRNN while maintaining high audio quality.

■ **Naive Implementation (Algorithm 1)**. Based on the overlap length $OL$ used in static blending, we take a fraction of $OL$ as $OF$, which is used to define the search range $\mathbb{Z}$ of dynamic overlap as $[OL-OF, OL+OF]$ for blending. As illustrated in Algorithm 1, $W$ is the output waveform. For each adjacent segment pair $S_i$ and $S_{i+1}$ in the segment batch, we first compute the Manhattan distance [21] for each dynamic overlap length $j$ as $Dist_j$, $j \in \mathbb{Z}$. Then we divide $Dist_j$ by $j$ to get the mean sample distance $\bar{Dist}_j$. Finally, we obtain the locally optimal dynamic overlap as $OL^D_{i,i+1}$ by minimizing $\bar{Dist}_j$ to blend the tail $OL^D_{i,i+1}$ samples of $S_i$ with the head $OL^D_{i,i+1}$ samples of $S_{i+1}$ (step 9), followed by replacing the tail $OL^D_{i,i+1}$ samples of $W$ with the head $OL^D_{i,i+1}$ samples of $S_{i+1}$ (step 10), and appending the remaining samples of $S_{i+1}$ to the end of $W$ (step 11). The computation complexity $\mathcal{C}$ of HDB is primarily due to the calculation of the Manhattan distance, as given by:

$$\mathcal{C} = (\mathcal{N}_S - 1) \cdot (2 \cdot OF + 1) \cdot SL \qquad (4)$$

where $\mathcal{N}_S$ is the number of segments. As shown above, HDB incurs additional computational complexity that grows linearly with the number of segments. To address this, we propose a parallel implementation (Algorithm 2) to accelerate HDB on the GPUs. This allows HDB to take advantage of the GPU's parallel computing capabilities, significantly reducing the overall computation time and improving performance.

■ **Parallel Implementation (Algorithm 2)**. As shown in Figure 3, we first parallelize the calculation of the Manhattan distance for the dynamic overlaps between two adjacent segments. Define two matrices $\mathbf{A}$ and $\mathbf{B}$. We initialize each row of $\mathbf{A}$ with the last $OL + OF + 1$ samples $(S_{i,j})_{-OL-OF-1 \leq j \leq -1}$ of segment $i$ and each row of $\mathbf{B}$ with the first $OL + OF + 1$ samples $(S_{i+1,j})_{0 \leq j \leq OL+OF}$ of segment $i+1$. We then roll the elements in the $k$th row of $\mathbf{B}$ by $k$ units to the right (RowWiseRoll), followed by subtracting $\mathbf{B}$ from $\mathbf{A}$ and computing the element-wise absolute ($f$) of the difference matrix $\mathbf{Dists}$. Next, we apply a pregenerated mask $\mathbf{M}$ to the $\mathbf{Dists}$. The last

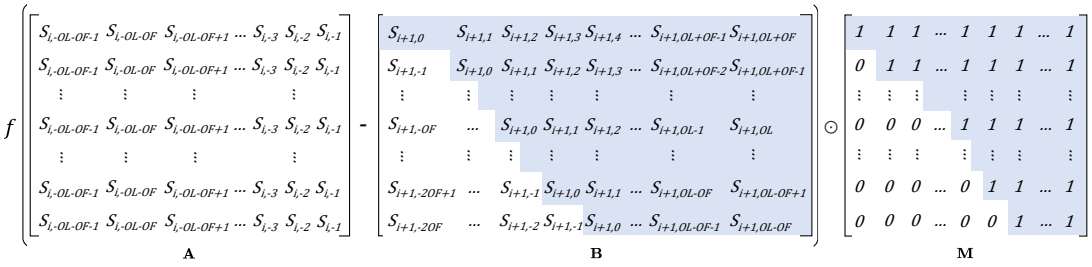

Figure 2: *Parallel approach to calculate the Manhattan distance of different overlap lengths between two adjacent signal segments.*

**Algorithm 1** Naive Implementation of HDB.

**Input:** $S, SL, OL, OF$
**Output:** $W$

$W \leftarrow S_0$
**for** $0 \leq i < \mathcal{N}_S$ **do**
  $Dists \leftarrow [\,]$
  **for** $OL - OF \leq j \leq OL + OF$ **do**
    $Dist_j = \text{Manhattan}(\text{tail}(S_i, j), \text{head}(S_{i+1}, j))$
    $\bar{Dist}_j \leftarrow D_j \div j$
    **add** $\bar{Dist}_j$ **to** $Dists$
  $OL^D_{i,i+1} = \text{argmin}(Dists) + OL - OF$
  $S_{i+1} \leftarrow \text{blend}(S_i, S_{i+1}, OL^D_{i,i+1})$
  $\text{tail}(W, OL^D_{i,i+1}) \leftarrow \text{head}(S_{i+1}, OL^D_{i,i+1})$
  $W \overset{+}{\leftarrow} \text{tail}(S_{i+1}, SL - OL^D_{i,i+1})$
**return** $W$

**Algorithm 2** Parallel Implementation of HDB.

**Input:** $\mathbf{S}, SL, OL, OF, \mathbf{M}$ ▷ Shape of $\mathbf{S}$: $(\mathcal{N}_S, SL)$
**Output:** $W$

$W \leftarrow \mathbf{S}[0, :]$
$\mathbf{A} \leftarrow \mathbf{S}[: -1, -OL - OF - 1 :]$, $\mathbf{B} \leftarrow \mathbf{S}[1 :, : OL + OF + 1]$
$\mathbf{A} \leftarrow \mathbf{A}.\text{unsqueeze}(1).\text{repeat}(1, 2 * OF + 1, 1)$
$\mathbf{B} \leftarrow \mathbf{B}.\text{unsqueeze}(1).\text{repeat}(1, 2 * OF + 1, 1)$
$\mathbf{B} \leftarrow \text{RowWiseRoll}(\mathbf{B}, axis = 2)$
$\mathbf{Dists} \leftarrow f(\mathbf{A} - \mathbf{B}) \odot \mathbf{M}$
$\bar{\mathbf{Dists}} \leftarrow \text{sum}(\mathbf{Dists}, axis = 2) \oslash \text{sum}(\mathbf{M}, axis = 2)$
$\mathbf{OL^D} \leftarrow OL + OF - \text{argmin}(\bar{\mathbf{Dists}}, axis = 1) - 1$
**for** $0 \leq i < \mathcal{N}_S$ **do**
  Same as step 9, 10, 11 of the naive implementation.
**return** $W$

$2OF + 1 - i$ elements of row $i$ in $\mathbf{M}$ are set to 1, indicating the valid difference values. After that, we calculate the mean value of the $k$-th row as the mean sample difference for the dynamic overlap $OL + OF - k$. Finally, we can obtain $OL^D_{i,i+1}$ by finding the row index of $\bar{\mathbf{Dists}}$ with the minimum mean sample difference. Based on the process above, the obtaining of local optimal dynamic overlaps $\mathbf{OL^D}$ of the entire segment batch $\mathbf{S}$ can be further fully parallelized on GPU by batched processing, as illustrated in Algorithm 2 in PyTorch-style pseudo-code.

# 4. Experiments

## 4.1. Experimental Setup

**Dataset.** The LJSpeech[22] dataset is used for training and evaluation. It contains 13,100 22.05kHz audio clips of a single female speaker and has a total of 24 hours. In our experiments, 100 randomly selected audio clips are reserved for evaluation.

**Acoustic Specifications.** The mel spectrogram with 80 bins is used as the input feature of the upsampling network. It is extracted with an FFT size of 1024, a hop length of 256, and a window length of 1024 from the normalized waveform. For faster and more stable convergence, we normalize mel to a symmetric interval of -4 to 4. In order to better evaluate the improvement of the proposal on models with different number of sub-bands $B$, we train two multi-band models (4-band and 8-band). To minimize the PQMF reconstruction error, the cutoff ratio is set to $0.142, 0.079$ for $4, 8$ sub-bands, respectively.

**Model Architecture.** For the upsampling network, different upsample factors are used for the 4-band and 8-band models. Specifically, upsample factors of 64 ($B$=4), 128 ($B$=8) are used in the interpolation layer after the residual blocks and upsample factors of [4,4,4] ($B$=4), [4,4,8] ($B$=8) are used in the interpolation layer inside the upsample blocks. There are 10 residual blocks and each block contains two 1D convolutions with kernel size 1 and in/out channels 128 followed by batch normalization. The convolutions in upsample blocks have a in/out channel of 1 and kernel sizes of [9,9,9] ($B$=4), [9,9,17] ($B$=8). As for the iterative network, all the GRU layers and linear layers have 512 hidden units, except for the last linear layer, which has 2048 units ($B$=4), and 4096 units ($B$=8).

**Training & Evaluation.** The models are trained using the Adam optimizer[23], with a initial learning rate of 1e-4 and a batch size of 32. Training is done on a NVIDIA Tesla V100 GPU, using single precision and gradient clipping[24] to ensure stability. Performance evaluation is done on the same GPU with Intel Xeon E5-2698 v4 CPU. As for speech quality, we evaluate using objective and subjective measures, including the Perceptual Evaluation of Speech Quality (PESQ)[25], Narrow-Band PESQ (NBPESQ), Short-time Objective Intelligibility (STOI)[26] and Mean Opinion Score (MOS). The MOS results are gathered by having 10 outsourced online listeners score 20 evaluation audio samples from 1 to 5 for each configuration. Audio samples can be found on our GitHub page[1].

Table 1: *Quality comparison of WaveRNN models with different numbers of sub-bands and different blending algorithms.*

| Model | Blend | PESQ | NBPESQ | STOI | RTF |
|---|---|---|---|---|---|
| MB-4 | - | 3.869 | 3.858 | 0.649 | 7.53 |
| MB-4-**F** | **S** | 3.481 | 3.653 | 0.646 | 0.22 |
| MB-4-**F** | **D** | **3.767** | **3.812** | **0.656** | 0.22 |
| MB-8 | - | 3.842 | 3.839 | 0.647 | 3.87 |
| MB-8-**F** | **S** | 3.617 | 3.729 | 0.646 | 0.22 |
| MB-8-**F** | **D** | **3.758** | **3.815** | **0.651** | 0.22 |

## 4.2. Discussion

### *4.2.1. Objective Evaluation*

Table 1 shows the objective speech quality scores of the multi-band (MB) models with 4 and 8 sub-bands under different blending algorithms. The suffix **F** denotes folding inference, **S**

[1] http://muyangdu.github.io/WaveRNN-Heuristic-Dynamic-Blending/

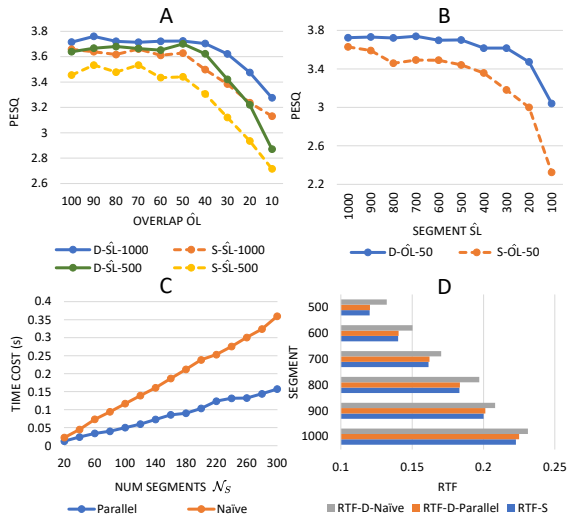


Figure 3: ***A***. *PESQ of $\hat{SL}$ 1000, 500, and $\hat{OL}$ varies from 100 to 10.* ***B***. *PESQ of $\hat{OL}$ 50 and $\hat{SL}$ varies from 1000 to 100.* ***C***. *The time cost of the Parallel and Naive Dynamic Blending.* ***D***. *RTF of the Static, Parallel Dynamic, and Naive Dynamic Blending.*

denotes static blending, and **D** denotes heuristic dynamic blending. The $\hat{SL}$ is set to 1000 for all the models to ensure that their iterative network loops the same number of vocoding iterations to obtain similar RTFs, while the $\hat{OL}$ for folding is set to 100 and 50 for the 4-band and 8-band models, respectively, to ensure that they apply the same $OL$ for blending. In all the configurations, 10% of the $OL$ is used as the $OF$. For non-folding inference, both the 4-band and 8-band models can generate high-quality speech with similar scores in all three quality metrics. However, their RTFs on the test set are far from real-time. In contrast, if folding is applied with static blending, we can achieve $\sim 4\times$ real-time but also incur a quality degradation. The drop is particularly noticeable in the 4-band (10.0% $\downarrow$ in PESQ) model as it have more folding segments compared to the 8-band (5.9% $\downarrow$ in PESQ) model when using the same $\hat{SL}$ for similar RTF. Similar drop can also be observed in NBPESQ.

If folding is applied with HDB, we observe a significant rebound in speech quality for both models. With HDB, the 4-band can achieves similar scores as the 8-band model and their PESQs only decrease by 2.6% and 2.2%, respectively, compared to non-folding synthesis. It is worth noting that HDB achieves even higher STOI than non-folding synthesis. STOI is proposed to evaluate the intelligibility of degraded speech signals (e.g., noise reduction). A higher STOI indicates a cleaner vocoded audio. We attribute this observation to the gap between training and non-folding inference. During training, the model is optimized using fixed-length audio fragments, but it is used to vocode arbitrary-length signals during non-folding inference. This gap can lead to the accumulated error in the GRU states that cause the iterative network to generate noisy samples when the inference signal length exceeds the length of the training fragment. In comparison, folding inference naturally bridge this gap by using a fixed-length short segment.

To further investigate the effect of segment and overlap length on sound quality, we conduct experiments using different $\hat{SL}$ and $\hat{OL}$ on the 8-band model. Figure 3 A shows the change in PESQ for $\hat{SL}$ values of 1000 and 500, with $\hat{OL}$ ranging from 100 to 10. In general, HDB has a higher PESQ than static blending. It can be observed that the PESQ of static blending begins to decrease when $\hat{OL} < 50$, while HDB can still maintain a high PESQ at $\hat{OL} = 40$. Furthermore, we study the variation of PESQ when $\hat{OL}$ is fixed at 50 and $\hat{SL}$ ranging from 1000 to 100, as shown in Figure 3 B. The score for static blending shows a drop at $\hat{SL} = 800$ and a second drop when $\hat{SL} < 500$. In contrast, HDB maintains a stable PESQ when $\hat{SL} > 400$.

To evaluate the performance of the parallel algorithm of our proposal, we measure the time consumption of parallel and naive algorithm under different $\mathcal{N}_S$ with $\hat{OL} = 50$, as shown in Figure 3 C. It can be observed that the parallel algorithm uses less than half of the blending time compared with the naive algorithm, which is especially important when $\mathcal{N}_S$ is high. Figure 3 D shows the average RTF of all the blending algorithm on the test set under various $\hat{SL}$. It can be confirmed that the parallel algorithm running on GPU achieves similar RTF as the static blending. With $\hat{SL}$ ranging from 1000 to 500, folding inference can achieve $\sim 4.5\times$ to $8.3\times$ real-time on the test set.

#### 4.2.2. *Subjective Evaluation*

Table 2: *MOS with 95% Confidence Intervals*

| Model | Blend | 1000 # 50 | 500 # 50 |
|---|---|---|---|
| MB-8-F | **D** | $4.05 \pm 0.06$ | $3.89 \pm 0.06$ |
| MB-8-F | **S** | $3.73 \pm 0.07$ | $3.31 \pm 0.07$ |
| Model | Blend | 1000 # 100 | 500 # 100 |
| MB-8-F | **D** | $4.17 \pm 0.05$ | $4.02 \pm 0.06$ |
| MB-8-F | **S** | $3.90 \pm 0.07$ | $3.45 \pm 0.07$ |
| MB-8 | - | $4.21 \pm 0.05$ | |
| GT | - | $4.62 \pm 0.05$ | |

We conduct listening tests on the 8-band model to evaluate the MOS of speech audio synthesized using folding inference with both static blending (**S**) and heuristic dynamic blending (**D**), and compare them to those of non-folding inference and ground truth (GT). The tests were conducted using $\hat{SL}$ 1000, 500 and $\hat{OL}$ 100, 50 ($\hat{SL}$ # $\hat{OL}$). The results, shown in Table 2, indicate that HDB generally achieves a higher score compared to static blending. Increasing $\hat{OL}$ from 50 to 100 can lead to a noticeable improvement for both static blending and HDB. When $\hat{SL}$ is set to 1000, HDB can synthesize speech with a quality comparable to non-folding inference.

Additionally, we find that when $\hat{SL}$ is set to 100 for a lower RTF, HDB is still able to produce speech of acceptable quality while static blending suffers from severe quality degradation (please refer to the audio samples[1]). Subjective results further demonstrate that HDB offers practitioners more flexibility in balancing the trade-off between RTF and quality.

## 5. Conclusions

In this work, we propose a novel blending approach called heuristic dynamic blending (HDB) for WaveRNN batched inference with feature folding, which effectively addresses the voice trembling and echo artifacts of static blending. Furthermore, we propose a parallel implementation of HDB running on GPUs to reduce the additional time overhead introduced by the naive HDB. Experimental results demonstrate the effectiveness of the proposed approach, achieving real-time GPU vocoding with speech quality comparable to non-folding vocoding.

# 6. References


[1] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*. IEEE, 2018, pp. 4779–4783.

[2] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T. Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

[3] A. Łańcucki, "Fastpitch: Parallel text-to-speech with pitch prediction," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6588–6592.

[4] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio, corr, vol. abs/1609.03499," 2017.

[5] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 2415–2424.

[6] J. Valin and J. Skoglund, "LPCNET: improving neural speech synthesis through linear prediction," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*. IEEE, 2019, pp. 5891–5895.

[7] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*. IEEE, 2019, pp. 3617–3621.

[8] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 14 881–14 892.

[9] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[10] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

[11] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "Wavegrad: Estimating gradients for waveform generation," in *International Conference on Learning Representations*.

[12] P. L. Tobing and T. Toda, "High-fidelity and low-latency universal neural vocoder based on multiband wavernn with data-driven linear prediction for discrete waveform modeling," in *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, H. Hermansky, H. Cernocký, L. Burget, L. Lamel, O. Scharenborg, and P. Motlícek, Eds. ISCA, 2021, pp. 2217–2221.

[13] Q. Tian, Z. Zhang, H. Lu, L. Chen, and S. Liu, "Featherwave: An efficient high-fidelity neural vocoder with multi-band linear prediction," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 195–199.

[14] H. Kanagawa and Y. Ijima, "Multi-sample subband wavernn via multivariate gaussian," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8427–8431.

[15] S. Narang, G. Diamos, S. Sengupta, and E. Elsen, "Exploring sparsity in recurrent neural networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[16] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *J. Mach. Learn. Res.*, vol. 22, pp. 241:1–241:124, 2021.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778.

[18] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 103–111.

[19] S. Haykin and M. Moher, *Introduction to analog and digital communications*. Wiley, 2007.

[20] T. Q. Nguyen, "Near-perfect-reconstruction pseudo-qmf banks," *IEEE Transactions on signal processing*, vol. 42, no. 1, pp. 65–76, 1994.

[21] P. E. Black, "Manhattan distance"" dictionary of algorithms and data structures," *http://xlinux. nist. gov/dads//*, 2006.

[22] K. Ito and L. Johnson, "The lj speech dataset," 2017.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[24] J. Zhang, T. He, S. Sra, and A. Jadbabaie, "Why gradient clipping accelerates training: A theoretical justification for adaptivity," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

[25] I.-T. Recommendation, "Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," *Rec. ITU-T P. 862*, 2001.

[26] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "A short-time objective intelligibility measure for time-frequency weighted noisy speech," in *2010 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2010, pp. 4214–4217.