

# Diffusion Buffer: Online Diffusion-based Speech Enhancement with Sub-Second Latency

Bunlong Lay<sup>1,2</sup>, Rostislav Makarov<sup>1</sup>, Timo Gerkmann<sup>1</sup>

<sup>1</sup>University of Hamburg, Signal Processing, Germany

<sup>2</sup>Hamburger Informatik Technologie-Center e.V., Germany

bunlong.lay@uni-hamburg.de, rostislav.makarov@uni-hamburg.de,  
timo.gerkmann@uni-hamburg.de

## Abstract

Diffusion models are a class of generative models that have been recently used for speech enhancement with remarkable success but are computationally expensive at inference time. Therefore, these models are impractical for processing streaming data in real-time. In this work, we adapt a **sliding window diffusion framework** to the speech enhancement task. Our approach progressively corrupts speech signals through time, assigning more noise to frames close to the present in a buffer. This approach outputs denoised frames with a delay proportional to the chosen buffer size, enabling a trade-off between performance and latency. Empirical results demonstrate that our method outperforms standard diffusion models and runs efficiently on a GPU, achieving an input-output latency in the order of 0.3 to 1 seconds. This marks the first practical diffusion-based solution for online speech enhancement<sup>1</sup>.

**Index Terms:** Speech enhancement, diffusion models, online

## 1. Introduction

The objective of speech enhancement (SE) is to retrieve the original clean speech signal from a noisy mixture that is affected by additive environmental noise [1]. **Online SE**, which refers to enhancing the speech signal with a limited latency as it is being received, holds significant importance in a variety of applications. The ability to perform SE in an online fashion is critical for ensuring clear and intelligible communication in video conferencing, VoIP calls, and other live interaction platforms. The specific latency constraints depend on the use case. In some cases, such as in live captioning for broadcast media or online streaming, a relatively large latency of 0.1 up to 1 second is also acceptable. However, developing online SE systems is challenging. It requires low processing time under given hardware requirements which makes the use of computationally costly methods impractical. Therefore, online SE has to use a relatively small model that is capable of handling real-world data unseen during the development of these systems.

Traditional SE approaches attempt to leverage statistical relationships between the clean speech signal and the surrounding environmental noise [2]. In recent years, various machine learning techniques have been introduced, treating SE as a discriminative learning task [3, 4, 5]. In contrast to these approaches that learn a direct mapping from a corrupted input file to clean speech, so-called **diffusion models** transform a known tractable prior distribution to a specific data distribution, which is in general intractable. The fundamental concept behind these generative models involves iteratively adding Gaussian noise

to the data *forward process*, thereby transforming data into a tractable distribution such as the Gaussian distribution. A neural network (NN) is trained to invert this diffusion process as part of a so-called *reverse process* [6]. In the context of SE, the prior distribution is the distribution of mixture signals corrupted by Gaussian noise. Hence, solving the reverse process means enhancing the corrupted mixtures as the reverse process transforms the distribution of corrupted mixtures into the distribution of clean speech signals. Diffusion models have proven to achieve excellent performance results when tested on unseen data and different corruption types [7, 8, 9]. This ability makes diffusion models interesting for real-world scenarios where audio data is streamed live.

However, the reverse process typically employs a large NN that is called several times. Therefore, current diffusion-based models for SE are too slow for the task of online processing of streamed audio data. Recent advancements in diffusion-based approaches [10, 11], have demonstrated the ability to generate videos by **progressively denoising temporal data over time**. Inspired by these methods, we propose adapting the concept of temporal denoising for diffusion-based SE. To achieve this, we introduce a **Diffusion Buffer** (DB) containing the most recent  $B$  frames of streamed noisy data. The latest input frame is always placed at the end of this buffer. Within this buffer, frames closer to the present time remain noisier, while frames further in the past (towards the beginning of the buffer) are progressively denoised. A frame that reaches the very beginning of the buffer is removed from the DB and directed to the output, making room for a new noisy frame from the stream. We show that by adjusting the buffer size we can trade enhancement performance for latency. Moreover, we demonstrate that in terms of PESQ and WVMOS we perform even slightly better than vanilla score-based diffusion models [12, 7] when we take 60 reverse steps. The proposed method achieves comparable results to the offline case when the latency introduced by the buffer is set to 320–960 ms. A key advantage of this method is that the **score model is only called once per input frame**. In fact, with the proposed method streamed data is processed in an online-fashion on a laptop with an NVIDIA RTX 4080 GPU with algorithmic latencies of 320–960 ms. The real-time factor (RTF) is below 1 and thus the input-output latencies are approximately given by the algorithmic latency plus hop size.

## 2. Background

We consider a speech enhancement task for speech signals corrupted by additive noise. The input is a noisy mixture  $\mathbf{Y} = \mathbf{S} + \mathbf{E}$  in the complex short-time Fourier transform (STFT) domain, consisting of a clean speech signal  $\mathbf{S} \in \mathbb{C}^{F \times K}$  and environmental noise  $\mathbf{E} \in \mathbb{C}^{F \times K}$ , where  $K$  and  $F$  are the number

<sup>1</sup>The source code is publicly available at  
<https://github.com/sp-uhh/Diffusion-Buffer>

of frames and number of frequency bins in  $\mathbf{Y}$ , respectively. The output is an estimate  $\hat{\mathbf{S}}$  of the clean speech signal. While bold symbols (e.g.  $\mathbf{Y}$ ) refer to spectrograms, we denote a single coefficient of a time-frequency bin by  $Y(l, k) \in \mathbb{C}$  for  $1 \leq l \leq F$  and  $1 \leq k \leq K$ .

## 2.1. Stochastic Differential Equations

The following formulation applies simultaneously to all time-frequency bins of  $Y(l, k)$ . This means that the following equations are one-dimensional. For readability, we will omit indices  $l, k$  in this subsection and reintroduce them in the following sections whenever the indices are important. Following the approach in [12, 7], we model the forward process of the score-based generative model with a stochastic differential equation (SDE) defined on  $0 \leq t < T_{\max}$ :

$$dX_t = f(X_t, Y)dt + g(t)dw, \quad (1)$$

where  $w$  is the standard Wiener process [13],  $X_t$  is the current process state with initial condition  $X_0 = S$ , and  $t$  is a continuous diffusion time-step variable describing the progress of the process ending at the last diffusion time-step  $T_{\max}$ . The drift coefficient  $f(X_t, Y)dt$  can be integrated by Lebesgue integration [14], and the diffusion coefficient  $g(t)dw$  follows Ito integration [13]. The diffusion coefficient  $g$  regulates the amount of Gaussian noise that is added to the process, and the drift  $f$  mainly affects the mean of  $X_t$  (see [13, (6.10)]) in the case of linear SDEs. The process state  $X_t$  follows a Gaussian distribution [15, Ch. 5], called the *perturbation kernel*:

$$p_{ot}(X_t|X_0, Y) = \mathcal{N}_{\mathbb{C}}(X_t; \mu_t(X_0, Y), \sigma_t^2 I). \quad (2)$$

We call  $\mu_t(X_0, Y)$  the *mean evolution* and  $\sigma_t$  the *variance evolution* as they describe how the mean and variance of the process state  $X_t$  are evolving over the diffusion time  $t$ . If we can find analytically closed-form solutions for the mean and variance evolution, then (2) allows us to efficiently compute the process state  $X_t$  for each  $t$  by calculating

$$X_t = \mu_t(X_0, Y) + \sigma_t z, \quad (3)$$

with  $z \sim \mathcal{N}_{\mathbb{C}}(0, 1)$ . By Anderson [16], each forward SDE as in (1) can be associated to a reverse SDE:

$$dX_t = [-f(X_t, Y) + g(t)^2 \nabla_{X_t} \log p_t(X_t|Y)] dt + g(t)d\bar{w}, \quad (4)$$

where  $d\bar{w}$  is a Wiener process going backwards in time. In particular, the reverse process starts at  $t = T$  and ends at  $t = 0$ . Here  $T < T_{\max}$  is a parameter that needs to be set for practical reasons as the last diffusion time-step  $T_{\max}$  is only reached in limit. The *score function*  $\nabla_{X_t} \log p_t(X_t|Y)$  is approximated by a NN called *score model*  $s_{\theta}(X_t, Y, t)$ , which is parameterized by a set of parameters  $\theta$ . Assuming that  $s_{\theta}$  is available, we can generate an estimate of the clean speech  $X_0$  from  $Y$  by solving the reverse SDE (4).

## 2.2. Latency considerations for streaming data

In contrast to the offline scenario, where an entire signal is available and can be operated on, the streaming scenario is more challenging: the input signal arrives one frame at a time and the output must be produced within a predefined amount of time to prevent adding further latency to the system. In our case, we must process a chunk of overlapping frames faster than the hop length's time. Application of score-based diffusion models

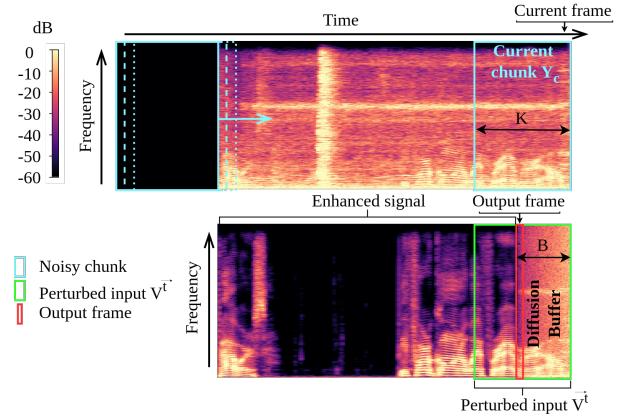


Figure 1: *The DB scheme for streamed data. The top part shows the noisy stream  $Y_s$ . The bottom presents the enhanced signal. In  $V^{\vec{t}}$ , we see exponentially increasing noise in the DB. The output frame is indicated in red at the beginning of the DB.*

such as [12, 7] on streaming data requires solving the reverse process within the time constraints defined above. Since the reverse SDE calls the score model several times, current score-based models such as [12, 17] are not usable for streaming data as even a single call of the score model exceeds the hop length as chosen in [12, 17].

## 3. Proposed method: Diffusion Buffer

Inspired by [10, 11], we propose to align the diffusion time-steps with the time axis of the noisy mixture. To this end, we introduce a Diffusion Buffer containing the last  $B$  frames, whereas the current frame  $\mathbf{R} \in \mathbb{C}^{F \times 1}$  is placed at the end of this buffer and past frames are closer to the beginning of the buffer. Within this buffer, frames that are closer to the end are modeled to be at larger diffusion time-steps and therefore contain more noise. Equivalently, frames that are further in the past are progressively denoised until they become fully denoised. Consequently, with each new frame added into the buffer, we output a denoised frame that lies further in the past. This is depicted in Fig. 1. Since there is a considerable delay between the current frame  $R$  and the output frame, this approach increases algorithmic latency. The amount of added latency can be adjusted by the choice of  $B$ , allowing for a trade-off between added latency and enhancement performance, as will be shown below. An important advantage of this approach is that within the time of each hop length, the reverse process only requires a single evaluation of the score. This is crucial, as multiple calls to the score model are computationally infeasible for online processing.

Mathematically, we formulate this process as follows: we fix  $B \leq K$  as the number of reverse steps taken to enhance the frames of the noisy mixture. Let  $\vec{t} = (t_1, \dots, t_B)$  be an ascending sequence of diffusion time-steps, i.e.  $0 < t_1 < \dots < t_B = T_{\max}$ . Then we call  $\mathbf{V}^{\vec{t}} \in \mathbb{C}^{F \times K}$  the *perturbed input* and define it as

$$V^{\vec{t}}(l, k) := \begin{cases} S(l, k) & \text{if } k < K - B, \\ X_{t_{g(k)}}(l, k) & \text{if } k \geq K - B. \end{cases} \quad (5)$$

where  $g(k) = k - (K - B) + 1$  and  $X_{t_{g(k)}}(l, k)$  can be computed via (3). In this formulation, the current frame from streamed data is placed at the last frame  $V(l, K)^{\vec{t}}$  which contains the most amount of Gaussian noise injected by the diffu-

sion process. Moreover, we see that the frames  $V^{\vec{t}}(l, k)$  for  $K - B \leq k \leq K$  are becoming clean when close to  $K - B$  and noisy when  $k$  is close to  $K$ . All other frames outside of this buffer are already assumed to be clean. The collection of frames  $V^{\vec{t}}(l, k)$  with  $K - B \leq k \leq K$  can also be thought of as a buffer on which we perform the diffusion process. Hence, we call it the *Diffusion Buffer*.

### 3.1. Training

For training, we fix the number of frames  $K$ , frequency bands  $F$ , the number of reverse steps  $B$  in the Diffusion Buffer, and the smallest diffusion time-step  $\epsilon > 0$ . We first sample a pair of clean and noisy files from the dataset. Then we pad the clean and noisy files with  $K - 1$  leading zeros in the STFT domain to mimic initialization when processing streamed data. In addition, we randomly crop a segment of  $K$  frames from the clean and noisy file which we call  $S = X_0$  and  $Y$  respectively for the sequel. Second, we uniformly randomly sample an ascending sequence  $\vec{t} = (t_1, \dots, t_B)$  of Diffusion time-steps with  $t_1 = \epsilon > 0$ . Third, based on (3), we compute for  $K - B \leq k \leq K$

$$X_{t_{g(k)}}(l, k) = \mu_{t_{g(k)}}(X_0(l, k), Y(l, k)) + \sigma_{t_{g(k)}} z(l, k) \quad (6)$$

with  $z(l, k) \sim \mathcal{N}_{\mathbb{C}}(0, 1)$ . We arrange  $z(l, k)$  in a matrix as  $\mathbf{Z} \in \mathbb{C}^{F \times B}$ , likewise  $\Sigma \in \mathbb{C}^{F \times B}$ . Fourth, we use (6) to calculate  $V^{\vec{t}}(l, k)$  as in (5) for all frequencies  $1 \leq f \leq F$  and frames  $1 \leq k \leq K$ . Last, we optimize on the denoising score matching loss:

$$\arg \min_{\theta} \mathbb{E}_{t, (\mathbf{X}_0, \mathbf{Y}), \mathbf{Z}, \mathbf{V}^{\vec{t}} | (\mathbf{X}_0, \mathbf{Y})} \left[ \| \mathbf{s}_{\theta}(\mathbf{V}^{\vec{t}}, \mathbf{Y}, t) + \frac{\mathbf{Z}}{\Sigma} \|_2^2 \right] \quad (7)$$

where the division of matrices  $\frac{\mathbf{Z}}{\Sigma}$  is meant to be elementwise. Note that for the score model input we have  $\mathbf{V}^{\vec{t}}, \mathbf{Y} \in \mathbb{C}^{F \times K}$ , but the score model's output is in  $\mathbb{C}^{F \times B}$ .

---

### Algorithm 1 Proposed Online Speech Enhancement

---

**Require:** Reverse process  $u_{\theta}$  with trained score model  $s_{\theta}$ , noisy stream  $\mathbf{Y}_s$ , chunk size  $K$ , fixed diffusion time-steps  $\vec{t} = (t_1, \dots, t_B)$

- 1:  $\hat{\mathbf{S}} \leftarrow []$  ▷ initialize output
- 2:  $\mathbf{V}^{\vec{t}} \leftarrow [0, \dots, 0]$  ▷  $K$  empty frames
- 3: **for** frame  $\mathbf{R}$  in  $\mathbf{Y}_s$  **do**
- 4:    $\mathbf{Y}_c \leftarrow$  last  $K$  received frames ▷ initialize with 0
- 5:    $\mathbf{V}^{\vec{t}}$  pops ▷ removes its first frame
- 6:    $\mathbf{V}^{\vec{t}}$  appends  $\mathbf{R} + \sigma_{t_B} \mathbf{Z}$  ▷  $\mathbf{Z} \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}, \mathbf{I}_{F \times 1})$
- 7:    $\mathbf{V}^{\vec{t}} \leftarrow u_{\theta}(\mathbf{V}^{\vec{t}}, \mathbf{Y}_c, \vec{t})$  ▷ only one  $s_{\theta}$  call
- 8:    $\hat{\mathbf{S}}$  appends  $B$ -th last frame of  $\mathbf{V}^{\vec{t}}$
- 9: **end for**
- 10: **Return**  $\hat{\mathbf{S}}$

---

### 3.2. Online inference for streamed data

Once we have a trained model as described in Section 3.1, we run inference as described in Algorithm 1 and illustrated in Fig. 1. Let  $\mathbf{Y}_s$  be an infinite stream of data in the STFT domain and assume we receive the frame  $\mathbf{R}$  in the for-loop of Algorithm 1. We then add in line 6 an amount of Gaussian noise so that the random variable  $\mathbf{R}' = \mathbf{R} + \sigma_{t_B} \mathbf{Z}$  follows the perturbation kernel (2), meaning  $\mathbf{R}'$  is at diffusion time-step  $t_B$ .

As usual, we then run one reverse step for  $\mathbf{R}'$  where the score function is approximated by the trained score model. Consequently,  $\mathbf{R}'$  is now at diffusion time-step  $t_{B-1}$ . In fact, we run the reverse step for all frames within the DB. In particular, this means that the  $B$ -th last frame which was before the reverse step at diffusion time-step  $t_1 = \epsilon$  is now at diffusion time-step 0. We therefore have enhanced this frame and output it to the listener. The output frame is also depicted in Fig. 1 which is at the beginning of the DB. Hence, the latency of this approach is given by  $h_s \cdot B$ , where  $h_s$  is the hop length in seconds, plus the compute time for one output frame, which should be smaller than the hop-length  $h_s$  for online processing. Note that the reverse process only enhances frames within the DB, all other frames are not processed as they are already enhanced (see also Fig. 1). As we slide over the streamed data as shown in Fig. 1, we ensure that each frame has undergone  $B$  reverse steps before it is enhanced. The zero part on the left of  $\mathbf{Y}_s$  in Fig. 1 is required for initialization. For this, we intentionally padded the training data with leading zeros in the first step in Section 3.1 to match the initialization.

## 4. Experimental Setup

### 4.1. Data representation

Each audio input, sampled at 16 kHz, is converted to a complex-valued STFT. As in [7], we use a window size of 510 samples (32 ms), a hop length of 256 samples ( $h_s = 16$ ms), and a periodic Hann window. The input to the score model is cropped randomly to  $K = 128$  time frames, resulting in approximately 2 seconds of data (see Section 3.1). A magnitude compression is used to compensate for the typically heavy-tailed distribution of STFT speech magnitudes [18]. Each complex coefficient  $v$  of the STFT representation is transformed as  $\beta|v|^{\alpha} e^{i\angle(v)}$  with  $\beta = 0.15$  and  $\alpha = 0.5$ , as in [7, 12].

### 4.2. Score model

For the score model  $s_{\theta}(\mathbf{X}_t, \mathbf{Y}, t)$ , we employ the Noise Conditional Score Network (NCSN++) architecture. The original NCSN++ architecture used in [7, 19] had 65M parameters. We reduced the network capacity by adapting its channel dimension from 128 to 96, reducing the number of Downsampling/Upsampling blocks from 6 to 4, and decreasing the number of residual blocks from 2 to 1. The resulting network has only 18M parameters.

For the proposed DB method we need to adapt the network in two ways. First, the time-embeddings  $t$  originally were only added to the channel dimension of the features of NCSN++. In our case, we now have a sequence of diffusion time-steps that need to be adapted to the channel and frame dimension of the features of the network. To this end, we use a Conv2D layer with stride to match the channel and frame dimension to the feature. This resulted in an additional 300.000 parameters. Second, the network output was originally of the same shape as its input  $\mathbf{Y}$ . Since we train on the loss from (7), we need to crop the output of the NN to contain only the last  $B$  frames, instead of taking all  $K$  frames.

We train the network on the loss described in Section 3.1. The optimizer we use is the ADAM optimizer [20] with a learning rate of  $10^{-4}$  and a batch size of 32. For smoothing the network parameters along the training epochs, we employ an exponential moving average of the score model's parameters [7, 19] (decay of 0.999). We trained for 250 epochs.

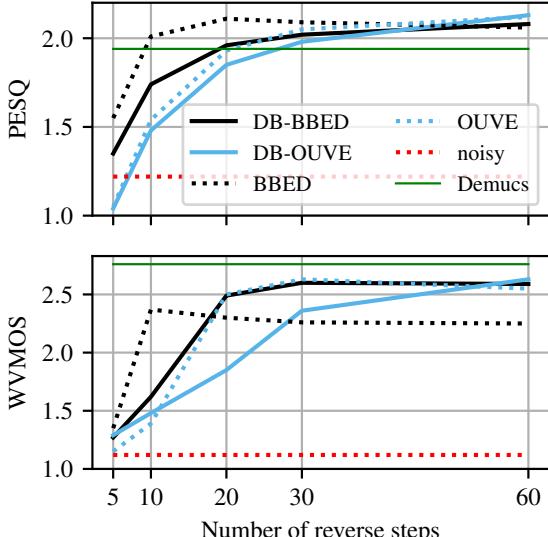


Figure 2: Comparing OUVE, BBED against their DB versions on the filtered EARS-WHAM test set. We experimented with different number of reverse steps. This is also the buffer size  $B$  for the DB versions with algorithmic latency equal to  $h_s \cdot B$ .

#### 4.3. Datasets and metrics

We use the publicly available dataset EARS-WHAM [21], originally recorded in 48 kHz. We downsampled the data to 16 kHz and filter it by removing files of type “whisper”, “highpitch” and “lowpitch”. We also used only pairs of clean and noisy with a Signal-to-Noise ratio in [0, 15] db. This dataset has 54 hours for training, 1.1 hours of validation, and 2 hours for testing.

We evaluate the performance on the perceptual metric wideband PESQ [22]. We also evaluate on a reference-free metric WVMOS [23] using a NN to predict MOS values. Note that this metric sometimes produces negative MOS values. In these cases, we simply set the WVMOS value to 1.0. Moreover, the RTF is defined as the processing time for one iteration in Algorithm 1 divided by  $h_s$ .

#### 4.4. SDEs and baselines

We will compare against OUVE [7] and BBED [12] with the data representation from Section 4.1. We use the parameterization as described in [24]. Specifically, the drift term of OUVE is  $f(\mathbf{X}_t, \mathbf{Y}) = \gamma(\mathbf{Y} - \mathbf{X}_t)$ , and BBED’s drift term is  $\frac{\mathbf{Y} - \mathbf{X}_t}{1-t}$ . Moreover, both SDEs use the diffusion term  $g(t) = ck^t$ . We refer to [24] for the closed-form solution for the mean and variance evolution, which is used to compute (3) or (6). We select the parameters  $c, k, \gamma$  as in [24]. Precisely, for OUVE we have  $\gamma = 1.5$ ,  $c = 0.01$ ,  $k = 10$  with a reverse starting point of 1.0. For BBED we select parameters  $c = 0.08$ ,  $k = 2.6$  with a reverse starting point of 0.8. For inference, we do not use an online streaming framework but rather enhance noisy files by utterance-based processing as it is done in [7, 12]. We use NCSN++ as described in Section 4.2 with the reduced capacity.

The proposed method DB is trained with the same parameterizations of BBED and OUVE which we call DB-BBED and DB-OUVE respectively. Additionally, we employ the NCSN++ parameterization used for the baselines. We also incorporate the necessary changes outlined in Section 4.2.

We also compare against the real-time capable discriminative network DEMUCS [25] operating in the time-domain.

## 5. Results

We train the proposed DB-OUVE and DB-BBED on the filtered 16 kHz EARS-WHAM dataset and experimented with the buffer length  $B = 5, 10, 20, 30, 60$ .

First, we discuss, as in Section 2.2 why the vanilla diffusion models OUVE and BBED do not operate in real-time on streamable data as their RTFs  $\gg 1$ . Compared to their original implementation in [7, 12] where the network has 65M parameters, we use in this work a reduced version of NCSN++. We intentionally reduced the NCSN++ architecture as described in Section 4.2 to ensure that the network’s processing time on a laptop with an NVIDIA RTX 4080 is smaller than the duration of one hop length of  $h_s < 16$  ms on chunks with  $K = 128$  frames. In fact, the processing time of taking one reverse step with the reduced NCSN++ is  $\approx 14$  ms (RTF is  $\frac{14}{16}$ ). When now applying BBED or OUVE to streamable data, during each hop length, the reduced NCSN++ is called  $N$  times where  $N$  is the number of reverse steps (RTF =  $N \frac{14}{16}$ ). This approach becomes computationally infeasible for  $N > 2$  and will not operate in real-time on the given hardware. In addition for  $N = 1$ , we report results that OUVE and BBED are even much worse than the noisy input. In contrast, the versions with a DB call the score model only once during the iteration in Algorithm 1 at the cost of increasing the latency by  $h_s \cdot B$ .

Second, we observe in Fig. 2 that the baselines OUVE and BBED slightly outperform their DB versions in terms of PESQ for 5, 10, 20, 30 numbers of reverse steps. However, for 60 reverse steps, the proposed DB-BBED and DB-OUVE methods are marginally outperforming their baseline counterparts by 0.05 PESQ. For WVMOS the proposed DB methods are at least on par with the baselines for all reverse steps (except for OUVE and DB-OUVE at 5 reverse steps). DB-BBED even outperforms BBED for 20, 30, 60 reverse steps. In addition, we see from Fig. 2 that with already 20 reverse steps we achieve reasonable results. Hence, the proposed method performs well with an algorithmic latency of 320 - 960 ms.

Third, we observe that the proposed methods are outperformed by DEMUCS in WVMOS, but achieve higher PESQ values when  $N > 30$ . We report results that the proposed methods ( $N > 30$ ) are as intelligible as DEMUCS, as their ESTOI [26] values differ only by 0.01.

The SDE parameterization of BBED has a larger variance schedule compared to OUVE, as shown in [24]. A higher variance schedule reduces the number of reverse steps needed for enhancement, which is why BBED outperforms OUVE in terms of PESQ and WVMOS when fewer reverse steps are used (see dotted lines in Fig. 2). This advantage extends to score-based diffusion models with the proposed DB, where DB-BBED also outperforms DB-OUVE for a few number of reverse steps (see solid lines in Fig. 2). Better performance with fewer reverse steps is crucial as it reduces algorithmic latency.

## 6. Conclusion

In this work, we successfully adapted score-based diffusion models to process streamed audio data online. Inspired by [10, 11], we denoise noisy frames through physical time. Unlike standard diffusion models that do not take the physical time axis into consideration, we introduce the Diffusion Buffer, where the position of each frame is taken into account and frames further from the present are progressively enhanced. This approach is computationally feasible as the score model is evaluated only once while waiting for new noisy data. The proposed Diffusion Buffer enables a trade-off between SE performance and latency, given by the number of reverse steps. At latencies between 320 and 960 ms, the proposed methods are running online on a consumer GPU achieving SE performance comparable with existing offline diffusion-based methods.

## 7. Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 545210893, 498394658.

Funded by the Federal Ministry for Economic Affairs and Climate Action (Bundesministerium für Wirtschaft und Klimaschutz), Zentrales Innovationsprogramm Mittelstand (ZIM), Germany, within the project FKZ KK5528802VW4.

The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project fac101ac. NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) – 440719683.

## 8. References

- [1] R. C. Hendriks, T. Gerkmann, and J. Jensen, *DFT-domain based single-microphone noise reduction for speech enhancement: A survey of the state-of-the-art*. Morgan & Claypool, 2013.
- [2] T. Gerkmann and E. Vincent, “Spectral masking and filtering,” in *Audio Source Separation and Speech Enhancement*, E. Vincent, T. Virtanen, and S. Gannot, Eds. John Wiley & Sons, 2018.
- [3] D. Wang and J. Chen, “Supervised speech separation based on deep learning: An overview,” *IEEE Trans. on Audio, Speech, and Language Proc. (TASLP)*, vol. 26, no. 10, pp. 1702–1726, 2018.
- [4] Y. Luo and N. Mesgarani, “Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation,” *IEEE Trans. on Audio, Speech, and Language Proc. (TASLP)*, vol. 27, no. 8, pp. 1256–1266, 2019.
- [5] R. Chao, W.-H. Cheng, M. La Quatra, S. M. Siniscalchi, C.-H. H. Yang, S.-W. Fu, and Y. Tsao, “An investigation of incorporating mamba for speech enhancement,” *IEEE Spoken Language Technology Workshop*, 2024.
- [6] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in Neural Inf. Proc. Systems (NeurIPS)*, vol. 33, pp. 6840–6851, 2020.
- [7] J. Richter, S. Welker, J.-M. Lemercier, B. Lay, and T. Gerkmann, “Speech enhancement and dereverberation with diffusion-based generative models,” *IEEE Trans. on Audio, Speech, and Language Proc. (TASLP)*, 2023.
- [8] R. Scheibler, Y. Fujita, Y. Shirahata, and T. Komatsu, “Universal score-based speech enhancement with high content preservation,” *IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 2024.
- [9] B. Nortier, M. Sadeghi, and R. Serizel, “Unsupervised speech enhancement with diffusion-based generative models,” *IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 2023.
- [10] D. Ruhe, J. Heek, T. Salimans, and E. Hoogeboom, “Rolling diffusion models,” *Int. Conf. on Machine Learning (ICML)*, 2024.
- [11] J. Kim, J. Kang, J. Choi, and B. Han, “Fifo-diffusion: Generating infinite videos from text without training,” *Advances in Neural Inf. Proc. Systems (NeurIPS)*, 2024.
- [12] B. Lay, S. Welker, J. Richter, and T. Gerkmann, “Reducing the prior mismatch of stochastic differential equations for diffusion-based speech enhancement,” *Interspeech*, 2023.
- [13] I. Karatzas and S. E. Shreve, *Brownian Motion and Stochastic Calculus*, 2nd ed. Springer, 1996.
- [14] W. Rudin, *Real and Complex Analysis*, 3rd ed. McGraw-Hill, Inc., 1987.
- [15] S. Särkkä and A. Solin, *Applied Stochastic Differential Equations*. Cambridge University Press, 2019, no. 10.
- [16] B. D. Anderson, “Reverse-time diffusion equation models,” *Stochastic Processes and their Applications*, vol. 12, no. 3, pp. 313–326, 1982.
- [17] S. Welker, J. Richter, and T. Gerkmann, “Speech enhancement with score-based generative models in the complex STFT domain,” *Interspeech*, 2022.
- [18] T. Gerkmann and R. Martin, “Empirical distributions of DFT-domain speech coefficients based on estimated speech variances,” *Int. Workshop on Acoustic Echo and Noise Control*, 2010.
- [19] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” *Int. Conf. on Learning Representations (ICLR)*, 2021.
- [20] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *Int. Conf. on Learning Representations (ICLR)*, 2015.
- [21] J. Richter, Y.-C. Wu, S. Krenn, S. Welker, B. Lay, S. Watanabe, A. Richard, and T. Gerkmann, “EARS: An anechoic fullband speech dataset benchmarked for speech enhancement and dereverberation,” *Interspeech*, 2024.
- [22] A. Rix, J. Beerends, M. Hollier, and A. Hekstra, “Perceptual evaluation of speech quality (PESQ) - a new method for speech quality assessment of telephone networks and codecs,” *IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, vol. 2, pp. 749–752, 2001.
- [23] P. Andreev, A. Alanov, O. Ivanov, and D. Vetrov, “Hifi++: A unified framework for bandwidth extension and speech enhancement,” *IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 2023.
- [24] B. Lay and T. Gerkmann, “An analysis of the variance of diffusion-based speech enhancement,” *Interspeech*, 2024.
- [25] A. Defossez, G. Synnaeve, and Y. Adi, “Real time speech enhancement in the waveform domain,” 2020.
- [26] J. Jensen and C. H. Taal, “An algorithm for predicting the intelligibility of speech masked by modulated noise maskers,” *IEEE Trans. on Audio, Speech, and Language Proc. (TASLP)*, vol. 24, no. 11, pp. 2009–2022, 2016.