Role of Samplers in Diffusion-based Generative Speech Enhancement

Master Thesis

Ernests Lavrinovics

Copenhagen, Denmark
Aalborg University
Department of Architecture, Design and Media Technology
Medialogy

Department of Architecture, Design and Media Technology
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Role of Samplers in Diffusion-based Generative Speech Enhancement

Theme:
Generative AI based Speech Enhancement

Project Period:
Spring 2023

Project Group:
MED10

Participant(s):
Ernests Lavrinovics

Supervisor(s):
PhD Diego Caviedes Nozal
PhD Rasmus Kongsgaard Olsson
Prof.PhD Stefania Serafin

Copies: 1

Page Numbers: 60

Date of Completion:
May 25, 2023

Abstract:

In the recent years, speech enhancement has benefited from advances in diffusion-based generative modeling since new models have been proposed that have set new state-of-the-art results. The core idea of diffusion theory is to gradually perturb a data distribution with noise to then learn how to reverse this noise. This study aims to investigate potential optimizations of a diffusion system, namely a particular component of them called the noise scheduler.
The noise scheduler defines the amount of noise in a sample during the diffusion processes, and literature suggests that it is a crucial component to diffusion systems that can be optimized seperately from other components.
Four experiments were conducted, each testing different modifications of the noise scheduler. In conclusion, it was found that the modification of the noise scheduler could potentially optimize output quality, although with certain assumptions about the training of the model. Future work could include development of sampling algorithms, testing the noise schedulers and samplers against different training methods for the score model and extending on modifications of noise schedulers.

Contents

1 Introduction **3**

2 Background **7**
2.1 Foundations of Diffusion-Based Generative Models 7
2.1.1 Score-Based Generative Models (SGM) 8
2.1.2 Score-Based Modelling through Stochastic Differential Equations 9

3 State-of-the-art: Diffusion Models in Speech Enhancement **11**

4 Methodology **15**
4.1 Experimental Setup 15
4.2 Voicebank-DMD Dataset 16
4.3 Performed Experiments 16
4.3.1 Implementation Details 16
4.3.2 Experiment 1: Swapping Scheduler Functions 17
4.3.3 Experiment 2: Derivative Modification 17
4.3.4 Experiment 3: Timestep Offset 18
4.3.5 Experiment 4: Non-uniform Timesteps 19
4.4 Quantitative Metrics 20
4.4.1 PESQ 21
4.4.2 STOI 21
4.4.3 DNSMOS 21
4.4.4 WARP-Q 21

5 Results **23**
5.1 Unprocessed Voicebank-DMD Performance 23
5.2 SGMSE Baseline Results 24
5.3 Experiment 1: Scheduler Functions 25
5.4 Experiment 2: Derivative Modification 26
5.5 Experiment 3: Timestep Offset 27
5.6 Experiment 4: Non-uniform Timesteps 28

6 Discussion **29**

7 Conclusion **31**

8 Acknowledgments **33**

Bibliography **35**

9 Appendix **41**
9.1 Annealed Langevin Dynamics Algorithm 41
9.2 Noise Schedulers 43
9.3 Voicebank-DMD Subsplit 45
9.4 Plots of Experiment 1 46
9.5 Plots of Experiment 2 52
9.6 Plots of Experiment 3 57
9.7 Plots of Experiment 4 59

Chapter 1

Introduction

Speech enhancement deals with improving the perceptual aspects such as intelligibility and quality of impaired speech signals by minimizing additive noise and artifacts that can occur due to poor acoustic environments or impaired transmission systems. Over the decades, many different approaches have been proposed that rely on analytical means [1, 2], although advances in deep learning have created a new paradigm based on generative modelling due to higher output quality and better generalization [3, 4, 5]. The core idea of generative models is to learn the distribution of clean speech, instead of a direct noise-to-clean mapping. Previous works have used Generative Adversarial Network (GAN) architectures [6, 7, 8] and Variational Auto Encoders (VAE) [9, 10] to perform speech enhancement, while more recent works utilize diffusion-based models [11, 12, 13, 14] that outperform the VAE and GAN models. A core difference between GAN/VAE and diffusion-based approaches is that in diffusion the output is iteratively and progressively refined, whereas VAE and GAN normally uses a single-pass, although [8] proposes a system architecture that also uses chained GANs to have a degree of step-wise refinements.

Diffusion theory for speech enhancement has been successfully adopted from the image processing community [15, 16] where the models are applied to tasks such as image inpainting, image colorization, class-conditioned generation [11, 15, 17]. The image diffusion models can also be conditioned on textual prompts for generating images, for example Dall-E 2, Stable Diffusion and others [18]. These advances can also inspire synthetic dataset generation for fields which are notorious for sparsely available datasets, e.g. medical [19]. For audio processing, diffusion-based models have shown impressive results for speech enhancement [13, 14] music generation and denoising [20, 21], and speech source separation [22]. This suggests that the diffusion-based generative modelling theory is flexible and is not tied to a particular domain.

The backbone of diffusion-based generative models is a neural network that

estimates a noise gradient and a sampling algorithm that uses the noise gradient to generate samples by performing a denoising routine. Literature suggests [23] that these two components can be decoupled and optimized separately, therefore in theory, sampling algorithms and their underlying components can be interchanged to better accommodate any potential shortcomings of the trained model. See Figure 1.2 for a taxonomy of diffusion model.

Figure 1.1 shows the diffusion process within a speech enhancement context. During the forward process, a data distribution is gradually perturbed with Gaussian noise, and during the reverse process the noise is reverted. For noise reversion, the sampler and neural network both work in tandem to eliminate the noise component in a step-by-step manner and nudge the sample towards the final clean distribution.

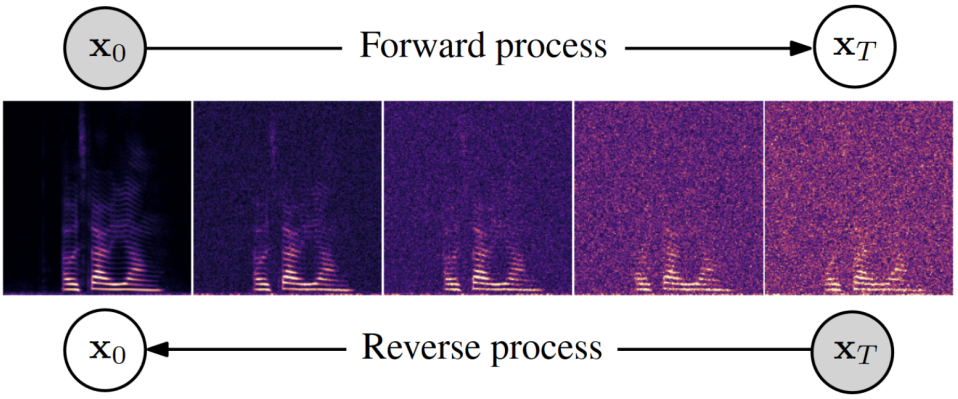


Figure 1.1: Illustration of an audio spectrogram and the iterative process of diffusion-based speech enhancement during forward and reverse processes. Illustration from [13]

The aim of this work is to make use of this decoupling and investigate optimizations in the sampler, more specifically a particular component of it called *noise schedule*. In essence, the noise schedule defines the amount of diffusion noise that a sample should have during forward and reverse diffusion processes. Meaning that lower schedule values would result in a less corrupted sample, and higher schedule values would result in a more corrupted sample. As suggested in literature [23, 24], the noise schedule is crucial to performance and does not always require following the same patterns as during training, and optimal schedule function can be task dependent [24]. In the diffusion community, improving the output quality and/or decreasing the computational costs of sampling are common research problems that are being addressed [25, 26, 27, 28, 11]. Therefore, the primary objective is to investigate the interplay between the generated output and the noise scheduler function to discover potential modifications that could improve the performance quantified by perceptual metrics. This is done through performing an empirical study by proposing a series of experiments that interchange and modify

the noise scheduler, while fixing other crucial variables such as the score network, sampling routine or stochasticity.

Figure 1.2: Taxonomy of diffusion models as per [11]

Chapter 2

Background

Diffusion models are part of a category *probabilistic generative models* [11] and in recent years have become a rapidly growing modelling approach within image, speech processing and NLP communities [11, 29, 12, 30] due to setting new state-of-the-art results for their respective applications. This chapter aims to describe the theoretical foundation of diffusion-based generative models and their different components.

2.1 Foundations of Diffusion-Based Generative Models

The seminal contribution [17] borrows concepts from non-equilibrium statistical physics where the evolution from state 0 to 1 is modelled as a stochastic process using Markov chains [31, 32], meaning that it can be viewed as a recursive process where the result at a particular time step *t* depends on the previous time step *t-1*.

The fundamental procedure of these models involves the progressive distortion of input data using noise injections (also called *perturbation kernels*) which usually consist of Gaussian noise. The data is recursively corrupted until it is practically indistinguishable from noise. This is followed by the reversal of the perturbations to generate a new data sample from a noise distribution. These two processes are referred to as the *diffusion* and *reverse-diffusion* processes, respectively, and each discrete perturbation or noise-reversion is a *diffusion step* (see Figure 2.1 for an illustration).

There are three dominant formulations of diffusion models: *denoising diffusion probabilistic models* [17, 16] (DDPMs), *score-based generative models* (SGMs) [15] and *stochastic differential equations* (Score SDEs) [33]. The following sections of this chapter of this work will go deeper into the theory behind the *SGM* and *Score SDE* architectures, since *Score SDE* builds upon *SGM*. Furthermore, later chapters describe experiments that are carried out using a model from the *Score SDE* family [13].

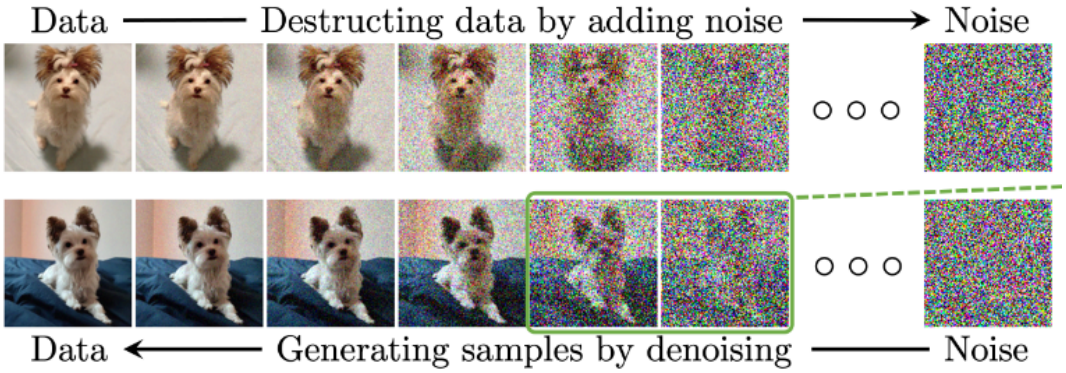


Figure 2.1: Illustration of the diffusion and reverse-diffusion processes. Each denoising step requires estimation of a *score function* that is described by a gradient pointing to the highest data likelihood. Image from [11].

2.1.1 Score-Based Generative Models (SGM)

The score-based generative model (SGM) [15] describes the inference architecture as a tandem of two main components: *score network* parametrized by θ which is used for estimating the noise gradient; and a *sampler* consisting of a Langevin dynamics algorithm that makes use of the aforementioned *score network* to generate samples.

Given a dataset consisting of independent and identically distributed samples $x_i \in \mathbb{R}$ of an unknown data distribution $p_{data}(x)$, a *score* is defined as the gradient of the log probability density function with respect to the data $\nabla_x \log p(x)$. The score is crucial during *reverse diffusion* steps, and it essentially is a vector field that describes how the noise within the generated sample should be shaped at a particular time step to nudge the sample towards the highest probability-density. The score is estimated by a parametrized neural network called *score network* that is trained by minimizing the following loss function 2.1 [34]

$$\frac{1}{2}\mathbb{E}_{p_{data}}\left[||s_\theta(x) - \nabla_x \log p_{data}(x)||^2\right] \tag{2.1}$$

where s_θ is a neural network output that aims to approximate the score, and $\nabla_x \log p_{data}(x)$ is the log-probability of the noise distribution that acts as a frame of reference for the model within the function. During training, the $\nabla_x \log p_{data}(x)$ can be computed because the data corruption process is controlled. In an ideal scenario, the training achieves $s_\theta(x) \approx \nabla_x \log p_{data}(x)$. The authors note that gradual data perturbation with varied levels of Gaussian noise is effective because it evenly corrupts the original sample. Furthermore, it allows for obtaining a sequence of gradually corrupted distributions that are used during training, that eventually converge to a distribution imperceptible of Gaussian noise irrespective of their prior distribution. These gradual perturbations allow training a single network

that estimates scores corresponding to all noise levels.

The *score* is then used to produce samples by a sampling routine, specifically authors propose an annealed Langevin dynamics algorithm. Formally, Langevin dynamics simulates the motion of a particle within a potential energy landscape. Within the generative modelling context, the *particle* is the synthetic sample and the potential energy landscape is the *score*. The Langevin method recursively computes Eq. 2.2 for generating discrete samples [15]

$$x_t = x_{t-1} + \frac{\epsilon}{2}\nabla_x \log p(x_{t-1}) + \sqrt{\epsilon} z_t. \tag{2.2}$$

where x_t is sample at time step t; $z_t \sim \mathcal{N}(0, I)$ is Gaussian noise; ϵ is noise-term magnitude; $\nabla_x log p(x_{t-1})$ is the score function.

During inference, Langevin dynamics algorithm is used with gradually decreasing noise denoted by σ. Typically, output with σ_{min} is to be approximately equal to the original data $p_{data}(x)$; σ_{max} is approximately equal to Gaussian noise with fixed mean and variance $\mathcal{N}(x; 0, \sigma_{max} I)$. See Appendix 9.1 for pseudocode and further details on Langevin dynamics.

2.1.2 Score-Based Modelling through Stochastic Differential Equations

As mentioned in the previous section, score-based modelling with Langevin Dynamics (SMLD) has two main components: *score* that is predicted using a neural network; *sampler* (e.g. Langevin dynamics algorithm) that uses the estimated score to generate samples.

The *Score-Based SDE* contribution [33] expresses the previously mentioned discrete Markov Chain approaches [16, 15] with stochastic differential equations. The use of SDEs enables to describe the forward-diffusion process as a continuum of distributions that evolve over time, instead of using a finite number of noise distributions when perturbing a data distribution. Eq. 2.3 describes the generic form SDE that the *score-based SDE* approach is based on. In practice, Eq. 2.3 is normally used during training to gradually corrupt a data distribution.

$$dx = f(x, t)dt + g(t)dw \tag{2.3}$$

Where t is time step, x is the data distribution, g is the diffusion coefficient and w is Wiener process (in practice represented as Gaussian noise).

A key property is that reversing this process satisfies a reverse-time SDE [35], that can be derived from the forward SDE, see Eq. 2.4. Therefore, it is possible to produce samples during *reverse-diffusion* by using a reverse-SDE if the score of a distribution is given at each intermediate time step t.

$$dx = \left[f(x, t) - g^2(t)\nabla_x \log p_t(x)\right] dt + g(t)d\overline{w} \tag{2.4}$$

where $\nabla_x log p_t(x)$ is the score, x is the perturbed data distribution, g is the diffusion coefficient and w is noise. In practice, the Eq. 2.4 is normally used to generate samples, this is the generic form of *samplers* for the SDE-based diffusion model family. Furthermore, it is important to note that samplers themselves can be multistep routines that combine reverse-time SDE solving with other processing steps.

The generic form SDE (Eq. 2.3) contains *g(t)* which is a function that gives diffusion coefficients. The diffusion coefficients act as a magnitude control of noise at a particular time step t. In practice when performing inference, the g(t) is discretized according to a certain number of time steps t and in literature this is referred to as a *noise scheduler*. It is suggested that the noise scheduler is crucial to performance, furthermore an optimal noise schedule can be task-dependent [23, 24]. If continuous time steps $t \in [0, 1]$ are used during inference, then a noise schedule during inference does not necessarily need to be the same as during training [23, 24].

Generally, the formalization of SDEs contribute to flexible sampling because any general-purpose SDE solver can be used to integrate the reverse-time SDE for sample generation. Furthermore, the generation process can be conditioned on information that is not available during training, this enables applications such as class-conditional generation, data inpainting and other inverse problems [11].

The training of the SDE-based model is performed by minimizing the following loss function Eq. 2.5 which is a modified form of the loss function 2.1

$$\theta^* = \arg\min_{\theta} \mathbb{E}_t \left[\lambda(t) \mathbb{E}_{x(0)} \mathbb{E}_{x(t)|x(0)} \left(\left\| s_\theta(x(t), t) - \nabla_{x(t)} \log p_{0t}(x(t)|x(0)) \right\|_2^2 \right) \right], \tag{2.5}$$

where $\lambda(t)$ is a weighting function, t is uniformly sampled time steps, $s_\theta(x(t), t)$ is the model's prediction, $\nabla_{x(t)} \log p_{0t}(x(t)|x(0))$ is the true score function computed for x(t) given x(0).

Expressing Discretized Sampling Processes in Continuous Time

Previous works [16, 15] express the reverse-diffusion process as a Markov Chains in discrete time steps. These formalizations can be expressed and are proposed by the authors [33] in continuous time as SDEs as well as a modification based on [16].

Since the main focus of this study are experiments with interchanged noise schedule functions, Appendix 9.2 contains the noise scheduler formalizations taken from [33, 23, 13].

Chapter 3

State-of-the-art: Diffusion Models in Speech Enhancement

Speech enhancement in itself is the task of recovering clean speech from noisy speech signals that are impaired by noise or reverberation [36]. Speech enhancement has been a hot research topic over the years due to the importance of speech in human communications [37, 38], data recovery [39], automated speech detection [40] or communications and hearing aids [41].

By analysing the underlying general statistics of speech, the clean speech signal is estimated by removing a noise estimate from noisy speech. Traditionally speech enhancement has been studied as a signal processing problem with analytical approaches, this was followed by data-driven supervised learning [1] and deep learning [42, 43, 44] using common network architectures such as *convolutional neural networks* and *recurrent neural networks*.

In recent years, speech enhancement has experienced a paradigm shift towards *generative modelling* by exploiting neural network architectures such as GAN [6, 8, 45] or diffusion-based [13, 14, 46, 12]. The fundamental difference is that generative models learn the underlying distribution of the training data to generate new samples with similar properties. Essentially generative models generate output that, ideally, should be imperceptible from the ground truth; as opposed to analytical approaches where input is directly operated on and modified. It is worth noting that the general theory described in Chapter 2 was put to use by research teams for tackling image processing tasks. Therefore, Chapter 3 reviews current state-of-the-art for conditional diffusion-based models for audio processing - specifically speech enhancement, conditioned on noisy speech to provide clean speech. This chapter also implicitly demonstrates the flexibility of diffusion theory. The SGMSE+ contribution [13] is described in more detail as it is used as a reference model for performing experiments described in Chapter 4.

CDiffuSE is a seminal contribution that adapts the DDPM [16] for speech en-

hancement task. The task adaption is performed by adapting the forward-diffusion process of using Gaussian noise with its mean being an interpolation of clean and noisy speech. The authors evaluate their model on *Voicebank-Demand*[40] dataset.

NU-Wave [47] is a pioneering work for speech upsampling and bandwidth extension using a diffusion-based generative model. The authors claim state-of-the-art results at the time of publishing when compared to other generative techniques such as GAN or baseline linear-interpolation.

UNIVERSE [14] motivation and main contribution is identified in the lack of flexible, general-purpose speech enhancement systems that can handle many different types of noise, e.g. echo, silent gaps, bandwidth reduction, clipping. All while retaining realism and without the introduction of critical voice artefacts [14]. The authors propose a generic model that can handle 55 distortion noise types and evaluate it by objective metrics as well as a listening test based on MUSHRA methodology. The base approach uses score-based SDE architecture and employ two neural networks to work in tandem, in a *predictor-corrector* scheme [33]. The authors note an observation that during their tests, listeners tend to prefer generative-based speech enhancement over output produced with regression. Furthermore, while their results suggest improvements over previous speech enhancement frameworks, the listening test subjects still preferred ground-truth more often than the audio generated with, *UNIVERSE* with most problematic noise types being generally low signal-to-noise ratios and long reverbs [14].

As mentioned previously, a common practice for forward and reverse diffusion steps is to corrupt data with Gaussian noise and afterwards invert it. **Cold diffusion** [48] for speech enhancement borrows the core idea from an image processing contribution [49] where the authors propose perturbing the data with a broader family of degradations, e.g. downsampling, masking, blurring. The work implies that such changes to sampling are valuable as real-world noise characteristics are rarely Gaussian noise and generalize better for arbitrary noise degradations. The authors perform sampling using a discrete Markov Chain approach, and propose a training scheme *unfolded training* for the core model that enables it to potentially correct for its mistakes by leveraging two approximations of a clean output from a degraded input.

Diffiner [50] is proposed that is trained using only clean audio, and it describes the diffusion process as a discrete Markov chain. The authors note that the model is intended to be used as a refiner for improving perceptual speech metric results. The basic idea of the model is that it relies on a pre-processed, already enhance speech signal, and it computes the noise variance between the enhanced vs noisy signals. The noise variance estimate is then used to guide the generative model to refine the signal.

Speech Enhancement and Dereverberation with Diffusion-based Generative Models

SGMSE+ [13] is recently published work that is based on SDE diffusion architecture. Authors propose three separate models: one for dereverberation, and two for ambient noise (traffic, café, outdoors, etc.) denoising. The task at hand is considered *conditional generation* because the generative process is conditioned on a noisy speech signal for which a clean version is to be generated. The authors follow the idea of incorporating the conditioning process directly into the forward and reverse processes by designing the following SDE which is the expansion of the generic form Eq.2.3 [13].

$$dx_t = \underbrace{\gamma(y - x_t)dt}_{:=f(x,t)} + \underbrace{\left[\sigma_{min}\left(\frac{\sigma_{max}}{\sigma_{min}}\right)^t \sqrt{2\log\left(\frac{\sigma_{max}}{\sigma_{min}}\right)}\right]}_{:=g(t)} dw \tag{3.1}$$

Where x_t is the present state with initial condition being x_0 that represents clean speech; y is noisy signal, w is Wiener process (in practice Gaussian noise); *f(x, t)* is *drift coefficient*; *g(t)* is *diffusion coefficient* that controls the amount of Gaussian noise σ injected at each forward step; γ is a *stiffness* constant.

As described in Section 2.1.2 and Eq. 2.4, the Eq. 3.1 has the following reverse SDE [13].

$$dx_t = \left[-f(x_t, y) + g(t)^2 s_\theta(x_t, y, t)\right] dt + g(t)d\overline{w} \tag{3.2}$$

Where $s_\theta(x_t, y, t)$ is a neural network parametrized by θ that estimates the *score*. Since equation 3.1 describes a Gaussian process, this enables direct sampling of x_t by using a *perturbation kernel* that dictates the type and magnitude of noise to be applied to the input at each diffusion step. The *perturbation kernel* is a normal distribution that the score model is fitted to. During training, the θ parameters are estimated, and the overall training objective is as follows:

$$argmin\mathbb{E}_{t,(x_0,y),z,x_t|(x_0,y)}\left[||s_\theta(x_t, y, t) + \frac{z}{\sigma(t)}||_2^2\right] \tag{3.3}$$

Eq. 3.3 essentially shows that the network is learning to minimize the L2-norm of the explicit output of the model s_θ that is summed with varied levels of Gaussian noise $\frac{z}{\sigma(t)}$ that depend on the time step t. Authors specify the noise range as $\sigma_{min} = 0.05$, $\sigma_{max} = 0.5$. The audio signals and time step are specified implicitly as parameters to functions. This learning objective creates a three-way mapping of a model's output at a time step t with a corresponding variance σ value, under the condition that z is always uniformly spread.

Sampling is inspired from [33] in a predictor-corrector (PC) scheme, where the predictor solves the reverse-time SDE and the corrector refines the sample. For

the predictor, the authors use Euler-Maruyama method, and for the corrector the authors use annealed Langeving Dynamics, which is described in more detail in Appendix 9.1.

For evaluating their models, the authors use two datasets; first one being WSJ0-CHiME3, which is a mix of WSJ0[51] clean speech and noise from CHiME3 [52]; second one being Voicebank-Demand [53]. The idea is to train on one dataset and test on the other to cross-validate the model's generalization. The authors quantify their results by performing a listening test based on MUSHRA [54] methodology. Also, quantitative evaluation metrics are computed such as POLQA [55], PESQ [56], ESTOI [57] as well as scale invariant Signal-To-Distortion, Signal-To-Inference and Signal-To-Artifact ratios [58].

Chapter 4

Methodology

As described in the Introduction, the core of this study is to explore the interplay of *noise schedulers* within samplers. This is done by taking a reference model, choosing a dataset, deciding on quantitative metrics and observing the changes in the resulting output when the corresponding noise scheduler functions are swapped as part of the sampling routine. Details of these steps are described in the following chapters.

4.1 Experimental Setup

For the experiments, the SGMSE+[13] [1] is taken as the reference model and described in more detail in Section 3. The SGMSE+ contribution proposes three separate speech enhancement models, two intended for general-purpose ambient noise (e.g. traffic, café), and one for dereverberation. This study uses the general-purpose ambient noise model that is trained on *WSJ0-CHiME3* dataset, which is WSJ0 [59] dataset of speech signals that are corrupted with noise from CHiME3 [52]. For the WSJ0-CHiME3 model, the authors evaluate it on *Voicebank-DMD* [53] dataset to better cross-validate generalization. The motivation for using the WSJ0-CHiME3 model is namely due to Voicebank-DMD dataset being freely available[2] as this choice to stay more consistent with the SGMSE+ publication.

Furthermore, the choice for the particular reference model is motivated by the fact that it claims to have achieved state-of-the-art results at the time of publishing, as well as its accessibility and relative simplicity of the system. Accessibility in the sense that all models from the original publication are open-sourced and Voicebank-DMD dataset is open for public access. And simplicity in the sense that the proposed work does not deploy two separate neural networks for a *predictor-*

[1] https://github.com/sp-uhh/sgmse

[2] https://datashare.ed.ac.uk/handle/10283/2791

corrector scheme, but rather employs the *corrector* directly in the sampling process.

4.2 Voicebank-DMD Dataset

As mentioned in the previous section, the experiments rely on *Voicebank-DMD* dataset, which is a combination of utterances from *Voicebank* [60] dataset with artificially added noise from *DEMAND* database [61]. The validation split of *Voicebank-DMD* contains 824 utterances of spoken English by native British speakers, 393 data points from a male speaker and 431 from a female speaker from Voicebank. The added noise varies in its signal-to-noise ratio between 17.5dB-2.5dB with 5dB increments, the added noise is from ambient environments such as transport, street, domestic and office.

4.3 Performed Experiments

Due to computational constraints, it is not feasible to run parameter sweep for empirical observations of model behaviour using a full dataset. Therefore, to enable faster feedback loops of generating results and drawing conclusions, preliminary experiments were carried out with a sub-split of the validation set of *Voicebank-DMD* dataset. The sub-split contains 35 pseudo-randomly selected data points, see Appendix 9.3 for a full list. The initial list of data points was obtained by using Python library *Numpy* random number generator and was fixed for all experiments. The following experiments were carried out while retaining the baseline hyperparameter configuration of the model as specified per [13].

4.3.1 Implementation Details

An important implementation detail to note is that the baseline scheduler function (Eq.9.6) outputs a different effective σ range than the σ_{min} and σ_{max} is provided at input. Therefore, all swapped scheduler functions in Experiment 1, as well as modifications to Mean Reverse VE (Experiment 2-4, Eq.9.7) function outputs are linearly scaled by using Eq 4.1 to match the effective σ range of the baseline scheduler function,

$$f(\sigma, x, y) = \frac{(\sigma - \sigma_{min})}{(\sigma_{max} - \sigma_{min})} * (y - x) + x \tag{4.1}$$

Where σ is a vector of values from a function that is being evaluated, x is the *effective minimum* and y is the *effective maximum* of the baseline function that the σ vector is scaled to. This is done to ensure a more fair comparison and to also match the range of σ values that the score model is actually trained on.

The tests are performed for a varied amount of diffusion step counts to observe the impact of decreasing step count on quality and how this can be balanced with scheduler function modifications or interchanges. The output of Eq 9.5 is reversed, since the authors of it assume an opposite direction of evolution in time as compared to other functions.

4.3.2 Experiment 1: Swapping Scheduler Functions

As per [23, 24] the scheduler functions σ are a crucial part of diffusion-based sampling, and they can be task-specific and do not need to be the same ones as during training of a model. Therefore, an experiment is performed of swapping out the baseline scheduler function Eq.9.6 with other scheduler functions to test for their interchangeability. The following functions proposed in literature - Linear9.4, Variance Exploding 9.1, Variance-Preserving 9.2, Sub Variance Preserving 9.3, Nvidia 9.5 with $\gamma = 1.5$ hyperparameter. Performed with diffusion step counts *n=10* and *n=30*.

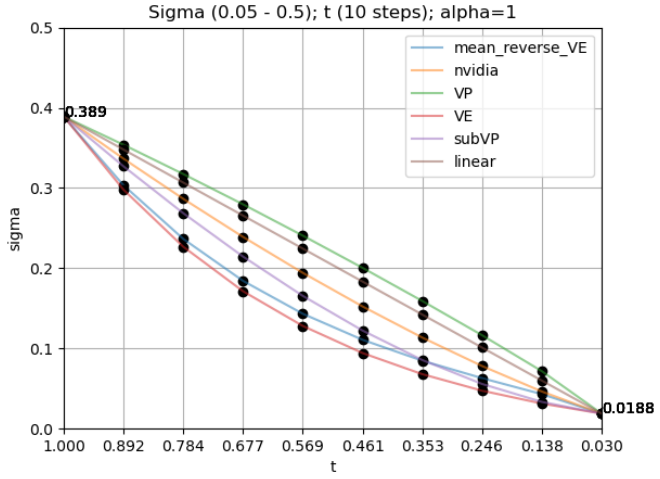


Figure 4.1: Experiment 1: Example overview of all tested scheduler functions

4.3.3 Experiment 2: Derivative Modification

Considering the baseline scheduler function Eq.9.6, introduce an *alpha* modifier to change its derivative, see Eq. 9.7. The alpha values were determined empirically based on observations from Experiment 1 which are later described in Section 5.3. The intention is to increase the amount of noise removed at the first-half of the time steps and smoothen out the $\Delta\sigma$ at the second half of the time steps. The assumption is that the output quality would start to improve at earlier time steps, which would

potentially lead to higher maximum values of the objective metrics. See Figure 4.2 for an illustration of the *alpha* modifier effect. Performed with diffusion step counts *n=10* and *n=30*.

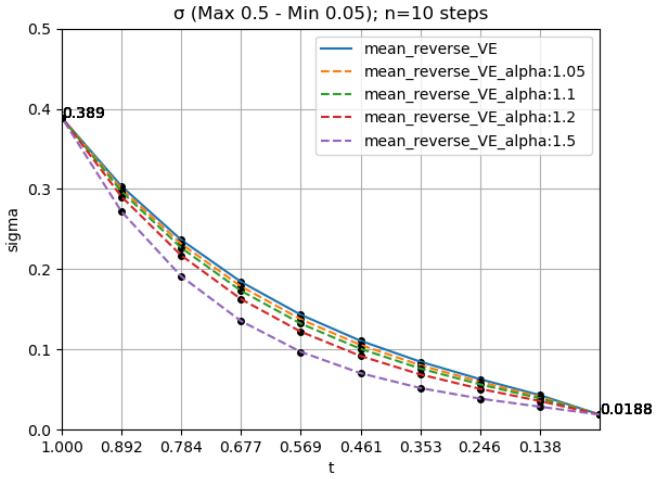


Figure 4.2: Experiment 2: Example overview of how the baseline scheduler function changes with the introduced modifier

4.3.4 Experiment 3: Timestep Offset

While retaining the σ values that are computed using baseline parameters, this experiment aims to non-linearly offset the corresponding time steps with a higher density towards the end of the function curve.

This is done by projecting the baseline σ v to an *alpha-modified* curve and derive the corresponding t values from it, see Figure 4.3 for an example. In practice, the σ value projection happens in the following sequence:

Algorithm 1 Projection of σ values to a modified curve

1: Generate diffusion step vector X of length n
2: Generate diffusion step vector Y of length $n \times 100$
3: Sample $\sigma_{mrve}(X)$
4: Sample $\sigma_{mrve_{alpha}}(Y, \alpha = 0.8)$
5: Perform linear interpolation of σ_{mrve} values between $\sigma_{mrve_{alpha}}$ and Y to obtain X'
6: return (σ_{mrv}, X')

In the Algorithm 1, the idea behind Step 2 is to effectively oversample the

$\sigma_{mrve_{alpha}}$ function to decrease the rounding errors when performing the linear interpolation.

The aim of this experiment is to investigate the effects of changing t values that are specified implicitly during network training, as per training objective Eq.3.3. The effect of *alpha* values below 1 causes the sampling points to compress with a higher density towards the end of the function, as well as implicitly decreases the amount of noise to be removed at a given time step t when compared to the baseline function. The core idea behind this is also supported in [23] as the authors suggest that models trained on minimizing L2-norm may have a tendency to remove too much noise, hence this experiment tests an attempt to balance out this tendency. Performed with diffusion step counts *n=10*, *n=15* and *n=30*.

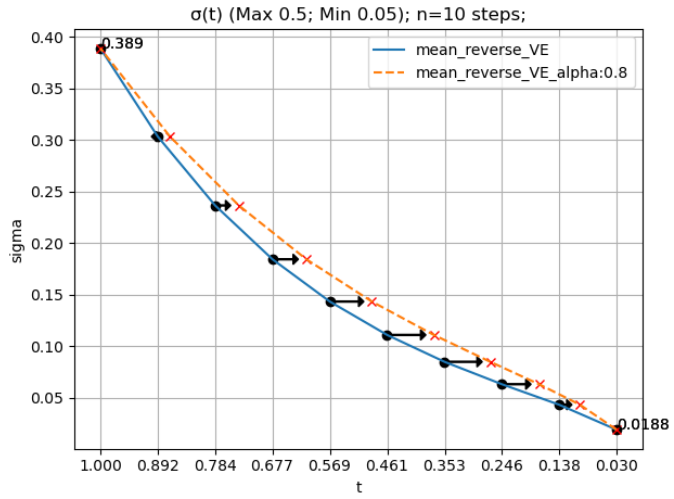


Figure 4.3: Experiment 3: Projecting baseline sigma values to an *alpha-modified* schedule function to derive the new t values. The blue curve with black dots depicts the baseline function with baseline discretization points, whereas the orange dashed curve with orange X markers represents the σ projection to an *alpha=0.8* modified baseline function.

4.3.5 Experiment 4: Non-uniform Timesteps

As shown in Experiment 3, the result of projecting σ values have two implicit effects on the relationship of t and σ. Firstly, an offset is created between the t and σ values that were used during the training, secondly, the discretization points are compressed with a higher density towards the end of the function. To better understand how these two effects affect the output, this experiment projects t values back on to the baseline function, essentially removing the σ and t offset but retaining the discretization point compression. As discussed and shown in later chapters, most of the quality improvements happen approximately in the second-

half or last-third of the diffusion steps. Therefore, this experiment tests whether non-uniform discretization steps would decrease the rough threshold at which the output quality starts to improve and potentially reach a higher peak due to the momentum.

The new σ and t values are obtained by taking the X' values specified as per Algorithm 1 and sampling the baseline scheduler Mean Reverse VE function (Eq. 9.6. This essentially creates a projection of the modified σ values back on to the baseline function, with the only difference being that the discretization points are spaced in a non-linear manner. See Figure 4.4 for an illustration. Performed with diffusion step counts *n=10*, *n=15* and *n=30*.

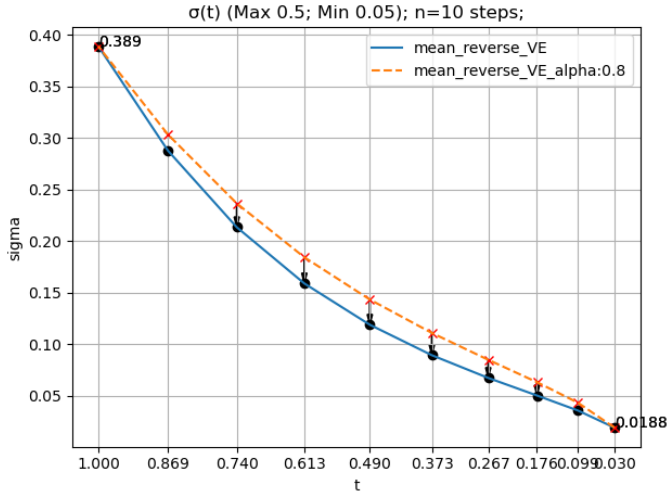


Figure 4.4: Experiment 4: Orange dashed curve with *'X'* represented the discretization from *Experiment 3* and the blue curve with black dots represents the newly obtained discretization points for the baseline function.

4.4 Quantitative Metrics

The results are quantified by computing perceptual quality metrics classified under two major families - *intrusive* (uses reference) and *non-intrusive* (no reference). A challenge with generative speech models is that the original and generated speech signals can have imperceptible structural differences, for example slight changes in phoneme or pitch. This means that the generated and reference speech signals may not align spectrotemporally and this effect may bias the results of certain perceptual metrics [62, 63, 64]. In total, four distinct metrics were chosen to better cross-validate the output quality and are further described in more details within the following.

4.4.1 PESQ

Published as an ITU P.862[56] recommendation for modelling subjective tests of perceived voice quality. PESQ uses a reference and a degraded signal to perform sample-by-sample analysis and is still used as part of speech evaluation [14, 13] even though it may be considered inappropriate [62, 50, 65] as it penalizes based on signal differences that are present but insignificant in generative speech. The metric outputs a MOS [66] (mean-opinion score) that is a coefficient between 1 (bad) to 5 (excellent). Implementation used from [3].

4.4.2 STOI

A metric intended for measuring speech intelligibility [67] and uses linear comparisons between the reference and degraded signal, also used previously in [14] and modified version in [13]. The output of this metric is a coefficient from 0 to 1, essentially indicating a positively correlating percentage of intelligibility - higher is better. Implementation used from [4]

4.4.3 DNSMOS

Non-intrusive metric based on a deep learning model that estimates a MOS score [68]. Trained on a labelled dataset of about 120k audio clips with associated MOS score. The dataset consists of 600 audio clips each processed by 200 noise suppression algorithms and afterwards crowdsourcing the associated MOS scores through listening tests. The metric computation is performed with DNSMOS implementation from GitHub[5] and it adopts an ITU-T P.835[69] methodology meaning that the results are separated into three submetrics - *speech quality*, *background noise*, *overall audio quality*. The results in this work are generated by using only the *overall audio quality* submetric.

4.4.4 WARP-Q

Specifically intended for evaluating generative neural speech [62]. It uses an analytical approach of preprocessing the audio signals by removing non-speech segments and afterwards it computes mel-frequency cepstrum coefficients (MFCC) which is a feature extraction technique [70, 71]. Afterwards, the MFCC's are compared using *subsequence dynamic time warping* [72] to minimize a cost function and find the minimum distance between the two audio signals. The final score is a coefficient

[3] https://github.com/ludlows/PESQ
[4] https://github.com/mpariente/pystoi
[5] https://github.com/microsoft/DNS-Challenge

with negative correlation, meaning that lower is better. Implementation used from [6].

[6] https://github.com/wjassim/WARP-Q

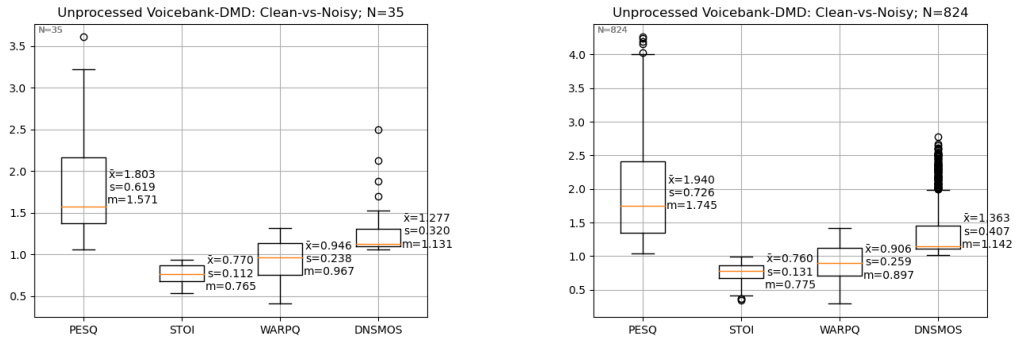
Chapter 5

Results

The following chapter summarizes the results obtained from the experiments described in Section 4, as well as baseline results of an unprocessed *Voicebank-DMD* dataset, and SGMSE baseline model configuration. It is important to note that box-plots are generated by accumulating the best performing diffusion-steps per each metric, per each datapoint. In most cases the best performing diffusion-step is the last one, although as later shown the last step may sometimes have a drop in performance.

5.1 Unprocessed Voicebank-DMD Performance

Evaluation metrics are computed for the full Voicebank-DMD dataset, as well as the 35 sample subsplit that is used to generate the results in the later subsections. This is done to set a global minimum for each metric that would enable a relative comparison between the processed and unprocessed data.



(a) 35 random sample subsplit.

(b) Full dataset.

Figure 5.1: Box plots of evaluation metrics computed for the unprocessed dataset

5.2 SGMSE Baseline Results

Computed by using default settings as per [13] for number of time steps *n=30* and *n=10*. Figure 9.9 depicts box-plots with quantitative metrics, and Figure 9.17 depicts the metric evolution throughout the whole of diffusion process for each data point. This is intended to give a qualitative intuition of how the results evolve during reverse-diffusion.

As per both Figures 9.9 and 9.17, it can be noted that a decrease of diffusion steps decreases the quality of results with respect to median and lowest-scoring data point. For PESQ and WARP-Q metrics, it is possible to observe an increased standard deviation, although the increase is only towards the positive range. This suggests that, for certain data points, the model was able to perform a substantially higher quality reverse-diffusion when higher resolution diffusion steps were present.

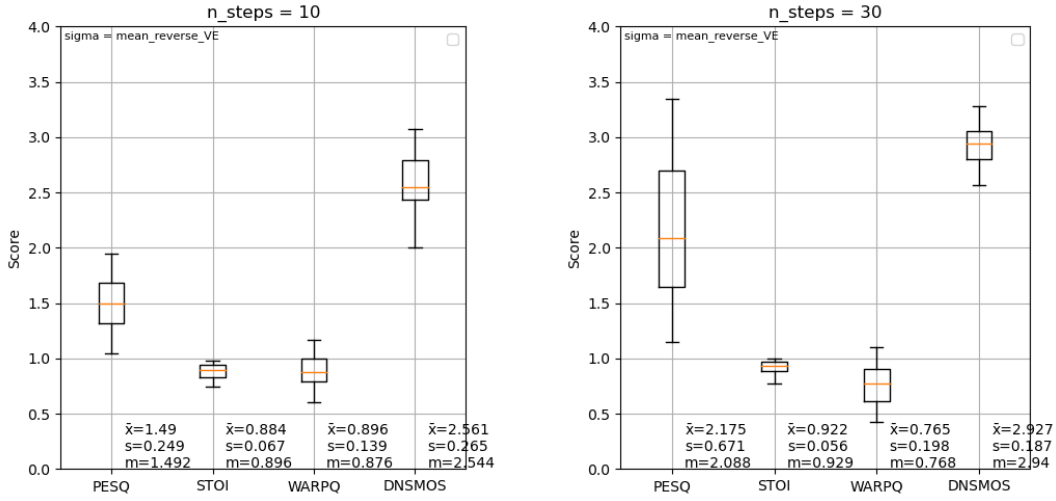


Figure 5.2: Box plots of reverse diffusion

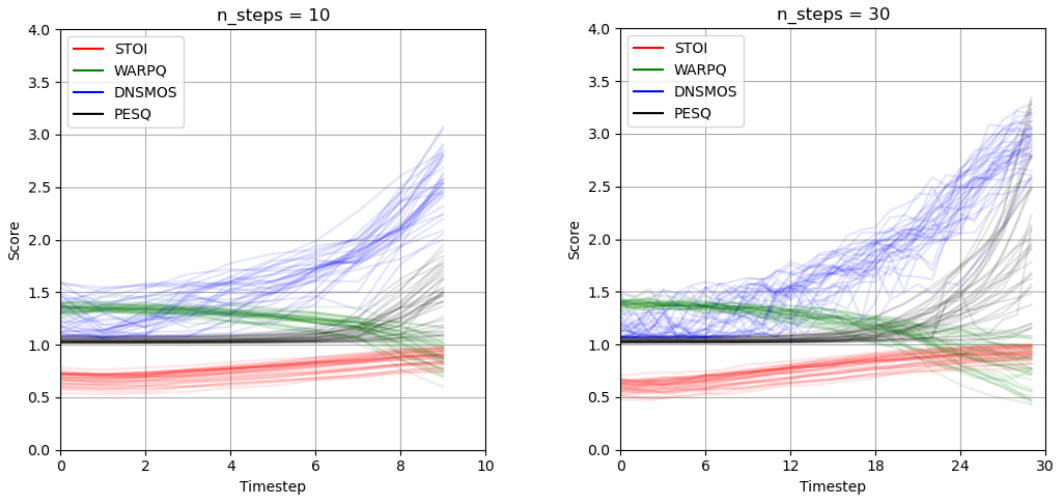


Figure 5.3: Overview of score evolution for all datapoints during reverse-diffusion. Timesteps are 0-indexed.

5.3 Experiment 1: Scheduler Functions

A summary of results is presented in Tables 5.1, with full plots available in Appendix 9.4. As expected, the results are consistently better with the default number of diffusion-steps *n=30*.

The improvement of metric scores appears to be non-linear depending on the

		PESQ	STOI	WARPQ	DNSMOS
Linear	n=10	1.472 (± 0.262)	0.869 (± 0.141)	0.951 (± 0.141)	**2.607** (±0.285)
	n=30	2.163 (± 0.69)	0.922 (± 0.058)	0.761 (± 0.198)	2.925 (± 0.192)
SubVP	n=10	1.503 (± 0.254)	0.889 (± 0.065)	**0.888** (± 0.145)	2.569 (± 0.246)
	n=30	2.174 (± 0.673)	0.921 (± 0.057)	0.768 (± 0.197)	2.913 (± 0.197)
VE	n=10	**1.572** (± 0.281)	**0.89** (± 0.065)	0.894 (± 0.144)	2.602 (± 0.259)
	n=30	2.123 (± 0.67)	0.919 (± 0.58)	0.774 (± 0.193)	2.897 (± 0.206)
VP	n=10	1.47 (± 0.282)	0.861 (± 0.068)	0.975 (± 0.141)	2.575 (± 0.254)
	n=30	2.155 (± 0.686)	0.921 (± 0.058)	**0.756** (± 0.198)	2.921 (± 0.199)
Nvidia	n=10	1.47 (± 0.236)	0.88 (± 0.067)	0.908 (± 0.14)	2.575 (± 0.254)
	n=30	**2.199** (± 0.674)	0.922 (± 0.057)	0.762 (± 0.198)	**2.934** (± 0.187)
Baseline	n=10	1.49 (± 0.249)	0.884 (± 0.067)	0.896 (± 0.139)	2.561 (± 0.265)
	n=30	2.175 (± 0.671)	**0.922** (± 0.056)	0.765 (± 0.198)	2.927 (± 0.187)

Table 5.1: Summary of results for Experiment 1 for n=10 and n=30 with mean value for each metric and standard deviation denoted in the brackets

scheduler function. For example, the SubVP and VE functions appear to ever-so improve the results with respect to baseline at *n=10*, although this improvement fades in *n=30* where the *Nvidia* schedule function appears to take a slight lead. Also, as shown in Figures 9.2, 9.8 and 9.10, for *n=10* the last diffusion-step appears to degrade the output quality with *linear*, *variance-preserving* and *nvidia* scheduler functions. Generally, from the output evolution plots in Appendix 9.4 it can be noted that most quality improvements happen in approximately the second-half of the diffusion-process.

5.4 Experiment 2: Derivative Modification

As per Figure 4.2, four modified versions of the baseline scheduler function were tested with an introduced *alpha modifier*. Summary of results is presented in Table 5.2 with full plots available in Appendix 9.5.

Table 5.2 depicts inconsistent improvements among metrics for *n=10* reverse-diffusion step count, and a consistent decrease in performance for *n=30*. The reverse-diffusion evolution as per Appendix 9.5 also show that generally the higher *alpha* values tend to decrease the amount of steps at which significant improvements in output quality can be observed. It is noted that this particular modification to the *scheduler* function takes an insignificant positive effect on certain metrics at certain *alpha* values, and generally can yield a little-to-negative effect on output quality.

		PESQ	STOI	WARPQ	DNSMOS
Alpha=1.05	n=10	1.498 (± 0.26)	0.882 (± 0.067)	0.905 (± 0.139)	2.563 (± 0.272)
	n=30	2.17 (± 0.671)	0.921 (± 0.056)	**0.764** (± 0.2)	**2.928** (± 0.186)
Alpha=1.1	n=10	1.513 (± 0.272)	0.881 (± 0.067)	0.913 (± 0.14)	2.566 (± 0.28)
	n=30	2.167 (± 0.67)	0.921 (± 0.057)	0.768 (± 0.201)	2.912 (± 0.191)
Alpha=1.2	n=10	**1.548** (± 0.304)	0.879 (± 0.066)	0.928 (± 0.144)	**2.586** (± 0.287)
	n=30	2.156 (± 0.67)	0.921 (± 0.057)	0.767 (± 0.2)	2.912 (± 0.194)
Alpha=1.5	n=10	1.504 (± 0.296)	0.864 (± 0.064)	1.017 (± 0.114)	2.498 (± 0.284)
	n=30	2.089 (± 0.68)	0.916 (± 0.06)	0.782 (± 0.206)	2.888 (± 0.217)
Baseline	n=10	1.49 (± 0.249)	**0.884** (± 0.067)	**0.896** (± 0.139)	2.561 (± 0.265)
	n=30	**2.175** (± 0.671)	**0.922** (± 0.056)	0.765 (± 0.198)	2.927 (± 0.187)

Table 5.2: Summary of results for Experiment 2 for n=10 and n=30 with mean value for each metric and standard deviation denoted in the brackets. Best performance is highlighted per each unique number of diffusion steps n.

5.5 Experiment 3: Timestep Offset

As shown in Figure 4.3, a modified version of the baseline function was tested, for which new t values were projected for corresponding baseline σ values. Table 5.3 summarizes the results with an additional data point for *n=15* sampling steps. The table shows consistent improvements of the *alpha=0.8* function and suggests that this may be a viable modification to a scheduler function for increasing the performance. It can also be observed that the relative rate of improvement between *alpha vs baseline* for a given n starts to consistently decline as n grows. Full plots are available in Appendix 9.6

		PESQ	STOI	WARPQ	DNSMOS
Alpha=0.8	n=10	**1.945** (± 0.526)	**0.903** (± 0.064)	**0.834** (± 0.169)	**2.809** (± 0.234)
	n=15	**2.135** (± 0.606)	**0.915** (± 0.06)	**0.779** (± 0.189)	**2.918** (± 0.227)
	n=30	**2.211** (± 0.669)	**0.923** (± 0.55)	**0.759** (± 0.201)	**2.944** (± 0.181)
Baseline	n=10	1.49 (± 0.249)	0.884 (± 0.067)	0.896 (± 0.139)	2.561 (± 0.265)
	n=15	1.93 (± 0.471)	0.911 (± 0.06)	0.805 (± 0.178)	2.858 (± 0.238)
	n=30	2.175 (± 0.671)	0.922 (± 0.56)	0.765 (± 0.198)	2.927 (± 0.187)

Table 5.3: Summary of results for Experiment 3 for n=10 and n=30 with mean value for each metric and standard deviation denoted in the brackets. Best performance is highlighted per each number of diffusion steps n.

5.6 Experiment 4: Non-uniform Timesteps

The results in Table 5.4 shows that the non-uniform discretization has a varied and inconsistent performance. For *n=10*, the PESQ and WARPQ values are improved but their standard deviations decrease in performance, STOI and DNSMOS have minor improvements. Similarly, for *n=15* and *n=30*, most improvements could be cancelled out by the increase of standard deviation besides PESQ for *n=30* and *n=15*. Similarly, as in Experiment 3, the relative improvement for both functions per each metric appears to follow a logarithmic progression where there is a greater performance leap is from *n=10* to *n=15* than *n=15* to *n=30*. See Appendix 9.7 for box and score evolution plots.

		PESQ	STOI	WARPQ	DNSMOS
Alpha=0.8	n=10	1.464 (± 0.238)	0.878 (± 0.067)	0.912 (± 0.134)	**2.567** (± 0.265)
	n=15	**2.1** (± 0.603)	0.91 (± 0.061)	**0.791** (± 0.193)	**2.884** (± 0.253)
	n=30	**2.187** (± 0.671)	0.922 (± 0.056)	**0.761** (± 0.203)	**2.936** (± 0.192)
Baseline	n=10	**1.49** (± 0.249)	**0.884** (± 0.067)	**0.896** (± 0.139)	2.561 (± 0.265)
	n=15	1.93 (± 0.471)	**0.911** (± 0.06)	0.805 (± 0.178)	2.858 (± 0.238)
	n=30	2.175 (± 0.671)	0.922 (± 0.56)	0.765 (± 0.198)	2.927 (± 0.187)

Table 5.4: Summary of results for Experiment 4 for n=10 and n=30 with mean value for each metric and standard deviation denoted in the brackets. Best performance is highlighted per each number of diffusion steps *n*.

Chapter 6

Discussion

The results generally show that for 3 out of 4 Experiments, the modifications do not appear to yield significant improvements, yet further tests for statistical significances between the sample groups could be performed to quantify that.

In Experiments 1 and 2, the results are inconsistent where no particular function or modification dominates by a great amount. At a closer inspection of Figure 9.11 and it can be noted that the plotted scheduler functions from log-files do not match up the expectations described in Methodology 4.2. This is an indication of a faulty implementation, and that Experiment 1 should be repeated. In essence, The scheduler functions that had been tested, have had their σ values projected on to the Mean Reverse VE function, therefore in practice they depict the results of different variants of non-linear time step spacings, similar to what is done in Experiment 4.

In Experiment 2, it is possible to observe minor gradual improvements for *PESQ* and *DNSMOS* metrics for *n=10* diffusion steps, although *STOI* and *WARPQ* all other metrics are degrading, although the improvements are minor with increasing deviations, therefore further statistical tests could be performed to test the sample group distributions against the baseline distribution. For *n=30* there is also a consistent drop in performance as the *alpha* factor is increased. Furthermore, qualitatively observing the evolution plots in Appendix 9.5, it can be noted that the increase of *alpha* modifier also increases the diffusion step at which significant improvements occur when compared against baseline (Figure 9.17), this is an indication that the modification impairs the diffusion system. Overall, the positive *alpha* modification does not appear to definitively give a positive impact on the performance.

Interestingly enough, in Experiment 3, the applied modifications consistently appear to improve the performance through better performing means and lower standard deviations (except for WARPQ). This could be explained by the theory proposed in [23] that models trained on minimizing the L2 norm tend to regress

towards the mean and have a tendency to remove too much noise, therefore this can be compensated by adding more stochasticity noise with the scheduler. Considering Figure 4.3, it shows that for each given time step, the noise magnitude is higher than that of the baseline function. This artificial increase on the noise-term is also proposed in Algorithm 2 in [23] although embedded all-together in a different sampling methodology. The increase of diffusion time steps t, decreases the relative performance improvement, which may further back the claim that increase of diffusion steps further stabilizes the whole system because the samples are generated more gradually. Furthermore, for *n=10* number of diffusion steps, Experiment 3 modification is the only one that enables the system to actually outperformed the unprocessed data shown in Section 5.1.

If a closer look is taken at the score evolution plots in Appendix 9.6, it is noticeable that the last diffusion step tends to significantly drop in performance. It is important to note that the box-plots and results are presented by taking the absolute best performing time step, not necessarily the last step. The drop in performance could be affected by the fact that the last step of the function is normalized back to the baseline function, resulting in a more intense drop of σ values. Therefore, the last step could also potentially benefit from an offset.

In Experiment 4, the results are inconsistent in their improvements, better performing means are roughly balanced out by worse performing standard deviations. This could be explained that the methodology projects the modified values back on to the original function as per Figure 4.4, therefore the deviations in results could occur due to non-uniform sampling time steps.

Intuitively, by looking at the score evolution plots in Figure 9.17, an assumption could be made that a non-uniform time step distribution with a higher density towards the end of the function could decrease the threshold at which noticeable improvements begin to form, potentially leading to higher max values. In Experiment 4, the methodology implicitly compresses the sampling points towards the end of the function without any σ/t offsets with respect to the baseline scheduler. Unfortunately, the score evolution plots in Appendix 9.7 do not appear to show a that significant improvements start to form at earlier time steps. This indicates that the quality improvements in the later diffusion steps may be more dependent on the progression of the diffusion process rather than the discretization steps themselves.

Chapter 7

Conclusion

The work investigated the impact of noise scheduler modifications on the performance across four different experiments. The modifications in 3 out of 4 experiments showcased inconsistency in results where no single function considerably outperformed the baseline, although there is a strong foundation to believe that Experiments 1 have faulty results, therefore should be repeated. Furthermore, results can still be further analysed by performing statistical significance tests between the samples and the baseline.

As diffusion step count increases, the functions tend to become more interchangeable and stable, with fewer deviations between the results. The findings also depicted that the last diffusion step in Experiment 3 tends to suffer from a noticeable performance drop when evaluated with lower diffusion step counts. This could be due to the methodology of how the noise scheduler function is modified with intermediate time steps modified, but first and last steps retained as per the original function. Nevertheless, a scheduler function could be optimized to yield better output quality as suggested in Experiment 3 and supported by literature, although this particular modification may only be applicable to models trained by minimizing an L2 norm.

This work suggest a natural continuation of exchanging sampling routines alltogether by interchanging the reverse-time SDE solvers (predictors) or any additional optimization techniques that can further refine the output (correctors). Such experiments can be repeated with the same general methodology as described in this work, with inclusion of interchanged scheduler functions to further quantify and observe their impact on results. Furthermore, a more in-depth investigation can be performed on the sampling stability to explain why the decreased step-count tends to cause a performance drop in the last diffusion step, for example by experimenting with linearly offset σ values while retaining t values. This could be done empirically through a mapping of decreased diffusion steps and modifications to the noise scheduler to observe if there is a consistent correlation of lower-

end sigma values and the performance drop, this would further comprehend the outcomes of Experiment 4. As also described in literature [23], improvements on the *score network* can also be quantified by testing different training methods of the score model, continuing the tests with stochastic differential equation formalism, or extracting a corresponding ordinary differential equation to then compare the quality of output.

Chapter 8

Acknowledgments

First and foremost I would like to thank my supervisors PhD Diego Caviedes Nozal (GN Audio), PhD Rasmus Kongsgaard Olsson (GN Audio) and Prof.PhD Stefania Serafin (AAU) for providing me with the support vital for carrying out this work. Also, I would like to thank Assoc.Prof PhD Cumhur Erkut (AAU) for the great formal and initial introduction to machine learning during the first semester of MSc Medialogy, as well as the rest of the study team for providing a pleasant learning environment. Also, I would like to acknowledge my family for the support throughout my studies and the rest of MSc Medialogy and SMC graduation year 2023 students for the great social environment, fruitful discussions and psychoemotional support especially Antoni Grabowski and Manos Dimogerontakis.

Bibliography

[1] Philipos C. Loizou. *Speech Enhancement: Theory and Practice*. 2nd. USA: CRC Press, Inc., 2013. ISBN: 1466504218.

[2] Jacob Benesty, Jingdong Chen, and Emanuël AP Habets. *Speech enhancement in the STFT domain*. Springer Science & Business Media, 2011.

[3] Szu-Wei Fu et al. "Metricgan+: An improved version of metricgan for speech enhancement". In: *arXiv preprint arXiv:2104.03538* (2021).

[4] Xugang Lu et al. "Speech enhancement based on deep denoising autoencoder." In: *Interspeech*. Vol. 2013. 2013, pp. 436–440.

[5] Dario Rethage, Jordi Pons, and Xavier Serra. "A wavenet for speech denoising". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 5069–5073.

[6] Santiago Pascual, Antonio Bonafonte, and Joan Serra. "SEGAN: Speech enhancement generative adversarial network". In: *arXiv preprint arXiv:1703.09452* (2017).

[7] Deepak Baby and Sarah Verhulst. "Sergan: Speech enhancement using relativistic generative adversarial networks with gradient penalty". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 106–110.

[8] Huy Phan et al. "Improving GANs for speech enhancement". In: *IEEE Signal Processing Letters* 27 (2020), pp. 1700–1704.

[9] Yoshiaki Bando et al. "Statistical speech enhancement based on probabilistic integration of variational autoencoder and non-negative matrix factorization". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 716–720.

[10] Huajian Fang et al. "Variational autoencoder for speech enhancement with a noise-aware encoder". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 676–680.

[11] Ling Yang et al. "Diffusion models: A comprehensive survey of methods and applications". In: *arXiv preprint arXiv:2209.00796* (2022).

[12] Chenshuang Zhang et al. “A survey on audio diffusion models: Text to speech synthesis and enhancement in generative ai”. In: *arXiv preprint arXiv:2303.13336* 2 (2023).

[13] Julius Richter et al. “Speech enhancement and dereverberation with diffusion-based generative models”. In: *arXiv preprint arXiv:2208.05830* (2022).

[14] Joan Serrà et al. “Universal speech enhancement with score-based diffusion”. In: *arXiv preprint arXiv:2206.03065* (2022).

[15] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in neural information processing systems* 32 (2019).

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[17] Jascha Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 2256–2265. URL: https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[18] Chenshuang Zhang et al. “Text-to-image Diffusion Model in Generative AI: A Survey”. In: *arXiv preprint arXiv:2303.07909* (2023).

[19] Richard J Chen et al. “Synthetic data in machine learning for medicine and healthcare”. In: *Nature Biomedical Engineering* 5.6 (2021), pp. 493–497.

[20] Qingqing Huang et al. “Noise2music: Text-conditioned music generation with diffusion models”. In: *arXiv preprint arXiv:2302.03917* (2023).

[21] Eloi Moliner, Jaakko Lehtinen, and Vesa Välimäki. *Solving Audio Inverse Problems with a Diffusion Model*. 2023. arXiv: 2210.15228 [eess.AS].

[22] Robin Scheibler et al. “Diffusion-based Generative Speech Source Separation”. In: *arXiv preprint arXiv:2210.17327* (2022).

[23] Tero Karras et al. “Elucidating the design space of diffusion-based generative models”. In: *arXiv preprint arXiv:2206.00364* (2022).

[24] Ting Chen. “On the importance of noise scheduling for diffusion models”. In: *arXiv preprint arXiv:2301.10972* (2023).

[25] Alexia Jolicoeur-Martineau et al. “Gotta go fast when generating data with score-based models”. In: *arXiv preprint arXiv:2105.14080* (2021).

[26] Tim Salimans and Jonathan Ho. “Progressive distillation for fast sampling of diffusion models”. In: *arXiv preprint arXiv:2202.00512* (2022).

[27] Daniel Watson et al. “Learning fast samplers for diffusion models by differentiating through sample quality”. In: *International Conference on Learning Representations*. 2022.

[28] Shitong Shao et al. “Catch-Up Distillation: You Only Need to Train Once for Accelerating Sampling”. In: *arXiv preprint arXiv:2305.10769* (2023).

[29] Florinel-Alin Croitoru et al. “Diffusion models in vision: A survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[30] Yuansong Zhu and Yu Zhao. “Diffusion Models in NLP: A Survey”. In: *arXiv preprint arXiv:2303.07576* (2023).

[31] Radford M Neal. “Annealed importance sampling”. In: *Statistics and computing* 11 (2001), pp. 125–139.

[32] C. Jarzynski. “Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach”. In: *Physical Review E* 56.5 (1997), pp. 5018–5035. DOI: 10.1103/physreve.56.5018. URL: https://doi.org/10.1103%2Fphysreve.56.5018.

[33] Yang Song et al. “Score-based generative modeling through stochastic differential equations”. In: *arXiv preprint arXiv:2011.13456* (2020).

[34] Yang Song et al. “Sliced score matching: A scalable approach to density and score estimation”. In: *Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 574–584.

[35] Brian D.O. Anderson. “Reverse-time diffusion equation models”. In: *Stochastic Processes and their Applications* 12.3 (1982), pp. 313–326. ISSN: 0304-4149. DOI: https://doi.org/10.1016/0304-4149(82)90051-5. URL: https://www.sciencedirect.com/science/article/pii/0304414982900515.

[36] DeLiang Wang and Jitong Chen. “Supervised speech separation based on deep learning: An overview”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.10 (2018), pp. 1702–1726.

[37] W. B. Kleijn et al. “Optimizing Speech Intelligibility in a Noisy Environment: A unified view”. In: *IEEE Signal Processing Magazine* 32.2 (2015), pp. 43–54. DOI: 10.1109/MSP.2014.2365594.

[38] Astrid Schmidt-Nielsen. “Intelligibility and acceptability testing for speech technology”. In: *Applied speech technology* (1995), pp. 194–231.

[39] Orchisama Das, Bhaswati Goswami, and Ratna Ghosh. “Application of the tuned Kalman filter in speech enhancement”. In: *2016 IEEE First International Conference on Control, Measurement and Instrumentation (CMI)*. 2016, pp. 62–66. DOI: 10.1109/CMI.2016.7413711.

[40] Felix Weninger et al. “Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR”. In: *Latent Variable Analysis and Signal Separation: 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings 12*. Springer. 2015, pp. 91–99.

[41] Li-Ping Yang and Qian-Jie Fu. “Spectral subtraction-based speech enhancement for cochlear implant patients in background noise”. In: *The Journal of the Acoustical Society of America* 117.3 (Mar. 2005), pp. 1001–1004. ISSN: 0001-4966. DOI: 10.1121/1.1852873. eprint: https://pubs.aip.org/asa/jasa/article-pdf/117/3/1001/12248886/1001_1_online.pdf. URL: https://doi.org/10.1121/1.1852873.

[42] Se Rim Park and Jinwon Lee. “A fully convolutional neural network for speech enhancement”. In: *arXiv preprint arXiv:1609.07132* (2016).

[43] Anurag Kumar and Dinei Florêncio. “Speech Enhancement In Multiple-Noise Conditions using Deep Neural Networks”. In: *CoRR* abs/1605.02427 (2016). arXiv: 1605.02427. URL: http://arxiv.org/abs/1605.02427.

[44] Yong Xu et al. “A regression approach to speech enhancement based on deep neural networks”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.1 (2014), pp. 7–19.

[45] Chris Donahue, Bo Li, and Rohit Prabhavalkar. “Exploring speech enhancement with generative adversarial networks for robust speech recognition”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 5024–5028.

[46] Yen-Ju Lu et al. “Conditional diffusion probabilistic model for speech enhancement”. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 7402–7406.

[47] Junhyeok Lee and Seungu Han. “NU-Wave: A Diffusion Probabilistic Model for Neural Audio Upsampling”. In: *Interspeech 2021*. ISCA, 2021. DOI: 10.21437/interspeech.2021-36. URL: https://doi.org/10.21437%2Finterspeech.2021-36.

[48] Hao Yen et al. *Cold Diffusion for Speech Enhancement*. 2023. arXiv: 2211.02527 [eess.AS].

[49] Arpit Bansal et al. “Cold diffusion: Inverting arbitrary image transforms without noise”. In: *arXiv preprint arXiv:2208.09392* (2022).

[50] Ryosuke Sawata et al. “A Versatile Diffusion-based Generative Refiner for Speech Enhancement”. In: *arXiv preprint arXiv:2210.17287* (2022).

[51] John Garofolo et al. “Csr-i (wsj0) complete ldc93s6a”. In: *Web Download. Philadelphia: Linguistic Data Consortium* 83 (1993).

[52] Jon Barker et al. "The third 'CHiME'speech separation and recognition challenge: Dataset, task and baselines". In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE. 2015, pp. 504–511.

[53] Cassia Valentini-Botinhao et al. "Noisy speech database for training speech enhancement algorithms and tts models". In: *University of Edinburgh. School of Informatics. Centre for Speech Technology Research (CSTR)* (2017).

[54] B Series. "Method for the subjective assessment of intermediate quality level of audio systems". In: *International Telecommunication Union Radiocommunication Assembly* (2014).

[55] John g. Beerends et al. "Perceptual objective listening quality assessment (POLQA), the third generation itu-t standard for end-to-end speech quality measurement part i—temporal alignment". In: *Journal of the Audio Engineering Society* 61.6 (2013), pp. 366–384.

[56] Antony W Rix et al. "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs". In: *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*. Vol. 2. IEEE. 2001, pp. 749–752.

[57] Jesper Jensen and Cees H Taal. "An algorithm for predicting the intelligibility of speech masked by modulated noise maskers". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.11 (2016), pp. 2009–2022.

[58] Jonathan Le Roux et al. "SDR–half-baked or well done?" In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 626–630.

[59] Douglas B Paul and Janet Baker. "The design for the Wall Street Journal-based CSR corpus". In: *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*. 1992.

[60] Christophe Veaux, Junichi Yamagishi, and Simon King. "The voice bank corpus: Design, collection and data analysis of a large regional accent speech database". In: *2013 international conference oriental COCOSDA held jointly with 2013 conference on Asian spoken language research and evaluation (O-COCOSDA/CASLRE)*. IEEE. 2013, pp. 1–4.

[61] Joachim Thiemann, Nobutaka Ito, and Emmanuel Vincent. "The Diverse Environments Multi-channel Acoustic Noise Database (DEMAND): A database of multichannel environmental noise recordings". In: *Proceedings of Meetings on Acoustics* 19.1 (June 2013). 035081. ISSN: 1939-800X. DOI: 10.1121/1.4799597. eprint: https://pubs.aip.org/asa/poma/article-pdf/doi/10.1121/1.4799597/14782232/035081_1_online.pdf. URL: https://doi.org/10.1121/1.4799597.

[62] Wissam A. Jassim et al. “Speech quality assessment with WARP-Q: From similarity to subsequence dynamic time warp cost”. In: *IET Signal Processing* 16.9 (2022), pp. 1050–1070. DOI: https://doi.org/10.1049/sil2.12151. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/sil2.12151. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/sil2.12151.

[63] Wissam A Jassim et al. “Speech quality factors for traditional and neural-based low bit rate vocoders”. In: *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2020, pp. 1–6.

[64] Yuma Koizumi et al. “DNN-based source enhancement to increase objective sound quality assessment score”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.10 (2018), pp. 1780–1792.

[65] Chandan KA Reddy et al. “A scalable noisy speech dataset and online subjective test framework”. In: *arXiv preprint arXiv:1909.08050* (2019).

[66] ITU-T Recommendation P.800. “Methods for subjective determination of transmission quality”. In: *Geneva: International Telecommunication Union* (1996).

[67] Cees H Taal et al. “An algorithm for intelligibility prediction of time–frequency weighted noisy speech”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.7 (2011), pp. 2125–2136.

[68] Chandan KA Reddy, Vishak Gopal, and Ross Cutler. “DNSMOS: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6493–6497.

[69] ITU-T Recommendation P.835. “Subjective test methodology for evaluating speech communication systems that include noise suppression algorithm”. In: *Geneva: International Telecommunication Union* (2003).

[70] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. “Comparative evaluation of various MFCC implementations on the speaker verification task”. In: *Proceedings of the SPECOM*. Vol. 1. 2005. Citeseer. 2005, pp. 191–194.

[71] Joan Serra et al. “Chroma binary similarity and local alignment applied to cover song identification”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 16.6 (2008), pp. 1138–1151.

[72] Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Vol. 5. Springer, 2015.

[73] Zhifeng Kong et al. “Diffwave: A versatile diffusion model for audio synthesis”. In: *arXiv preprint arXiv:2009.09761* (2020).

Chapter 9

Appendix

9.1 Annealed Langevin Dynamics Algorithm

Algorithm 2 is the pseudo code of the Langevin dynamics sampler as proposed by [15].

Algorithm 2 Annealed Langevin dynamics

Require: $\{\sigma_i\}_{i=1}^{L}$, ε, T
1: Initialize $\tilde{x}_0$
2: **for** $i \leftarrow 1$ to L **do**
3: $\quad \alpha_i \leftarrow \varepsilon \cdot \sigma_i^2 / \sigma_L^2$
4: $\quad$ **for** $t \leftarrow 1$ to T **do**
5: $\quad\quad$ Draw $z_t \sim \mathcal{N}(0, I)$
6: $\quad\quad \tilde{x}_t \leftarrow \tilde{x}_{t-1} + \frac{\alpha_i}{2} s_\theta(\tilde{x}_{t-1}, \sigma_i) + \sqrt{\alpha_i} z_t$
7: $\quad$ **end for**
8: $\quad \tilde{x}_0 \leftarrow \tilde{x}_T$
9: **end for**
10: **return** $\tilde{x}_T$

The algorithm requires a sequence of σ values that represent the noise variance, ϵ a coefficient that is a tunable hyperparameter for changing the *step size* and T number of steps to execute the core of Langevin dynamics algorithm before updating the step-size.

Step 1. Initializing the starting point of our sample, in practice this would be Gaussian noise.

Step 2. Initialize the outer loop of the algorithm over a series of different noise levels (variances)

Step 3. Calculate the step size α_i that controls the amount of change applied in the algorithm.

Step 4. Initialize the inner loop for computing the Langevin dynamics formula

Step 5. Draw a Gaussian noise sample

Step 6. Compute Eq. 2.2

Step 8. Update the output state and prepare for the next iteration of the whole algorithm

The idea behind the two nested loops are that the Step 4. to Step 6. can perform *exploration* of the data space, with gradually decreasing step-size in the outer loop on Step 3, this allows for higher quality sampling and aids to prevent of getting stuck in a local minima [15].

9.2 Noise Schedulers

As mentioned in Chapter 2, the diffusion process is described by first corrupting a data distribution with varied levels of Gaussian noise, the transitions between two discretized points in time are called *perturbation kernels*. The reversion of Gaussian noise is learnable as per 2.5 because the forward process is controlled and the ground-truth perturbation kernels in the forward-diffusion are known.

The generic form Gaussian distribution is formalized as follows: $X \sim \mathcal{N}(\mu, \sigma)$, where X is a sample drawn from a normal distribution $\mathcal{N}$ that is centred around some mean μ with a variance of σ. This can be expanded to an equation of $p_{0t}(x(t)|x(0)) = \mathcal{N}(\mu, \sigma)$, where $p_{0t}(x(t)|x(0)$ describes the transition from 0 to t for a state x(t) given x(0).

In diffusion models, the *variance* parameter of Gaussian noise is commonly a function, that when discretized, is referred to as the noise scheduler in the literature. The following subsections present different σ variance formalizations that are used for in experiments described in Chapter 4.

Variance Exploding

Described in [11] Equation 31.

$$\sigma(t, \sigma_{max}, \sigma_{min}) = \sigma_{\min} \left(\frac{\sigma_{max}}{\sigma_{min}} \right)^{t} \tag{9.1}$$

Variance Preserving

Described in [11] Equation 33.

$$\sigma(t, \sigma_{max}, \sigma_{min}) = \sqrt{1 - e^{-\frac{1}{2}t^2(\sigma_{max} - \sigma_{min}) - t\sigma_{min}}} \tag{9.2}$$

Sub-Variance Preserving

Described in [11] Equation 34.

$$\sigma(t, \sigma_{max}, \sigma_{min}) = 1 - e^{-\frac{1}{2}t^2(\sigma_{max} - \sigma_{min}) - t\sigma_{min}} \tag{9.3}$$

Linear

Linear schedule is used in [73], and can be constructed as follows considering SGMSE+ constraints.

$$\sigma(t, \sigma_{max}, \sigma_{min}) = (\sigma_{max} - \sigma_{min}) * t + \sigma_{min} \tag{9.4}$$

Nvidia

Described in [23] Equation 5.

$$\sigma(t, \sigma_{max}, \sigma_{min}, \gamma) = \left(\sigma_{max}^{\frac{1}{\gamma}} + t\left(\sigma_{min}^{\frac{1}{\gamma}} - \sigma_{max}^{\frac{1}{\gamma}}\right)\right)^{\gamma} \tag{9.5}$$

SGMSE+ Variance Exploding

Described in [13] Equation 6. Referred to as Mean Reverse VE in Chapter 4 and 5

$$\sigma(t, \sigma_{max}, \sigma_{min}, \gamma)^2 = \frac{\sigma_{min}^2\left((\frac{\sigma_{max}}{\sigma_{min}})^{2t} - e^{-2\gamma t}\right)\log(\frac{\sigma_{max}}{\sigma_{min}})}{\gamma + \log(\frac{\sigma_{max}}{\sigma_{min}})} \tag{9.6}$$

SGMSE+ Variance Exploding with an Alpha Modifier

Based on [13] Equation 6 with an added α exponent highlighted in the function. Referred to as Mean Reverse VE Alpha in Chapter 4 and 5

$$\sigma(t, \sigma_{max}, \sigma_{min}, \gamma, \alpha)^2 = \frac{\sigma_{min}^2\left((\frac{\sigma_{max}}{\sigma_{min}})^{2t\boldsymbol{\alpha}} - e^{-2\gamma t}\right)\log(\frac{\sigma_{max}}{\sigma_{min}})}{\gamma + \log(\frac{\sigma_{max}}{\sigma_{min}})} \tag{9.7}$$

9.3 Voicebank-DMD Subsplit

Sorted list of pseudo randomly selected datapoints from the *Voicebank-DMD* validation split for performing preliminary tests.

1. p232_011
2. p232_017
3. p232_032
4. p232_060
5. p232_084
6. p232_090
7. p232_108
8. p232_159
9. p232_186
10. p232_213
11. p232_254
12. p232_256
13. p232_266
14. p232_269
15. p232_286
16. p232_301
17. p232_362
18. p232_413
19. p257_011
20. p257_015
21. p257_030
22. p257_049
23. p257_064
24. p257_086
25. p257_087
26. p257_132
27. p257_142
28. p257_241
29. p257_247
30. p257_259
31. p257_313
32. p257_317
33. p257_329
34. p257_343
35. p257_370

9.4 Plots of Experiment 1

Box and evolution plots of Linear, SubVP, VE, VP and Nvidia schedule functions for Experiment 1.

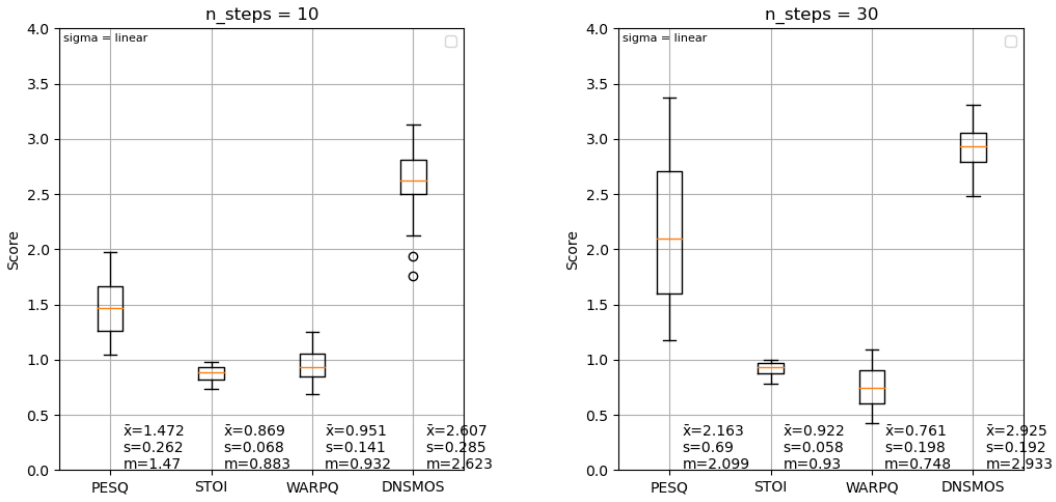


Figure 9.1: Box plots of reverse diffusion for linear

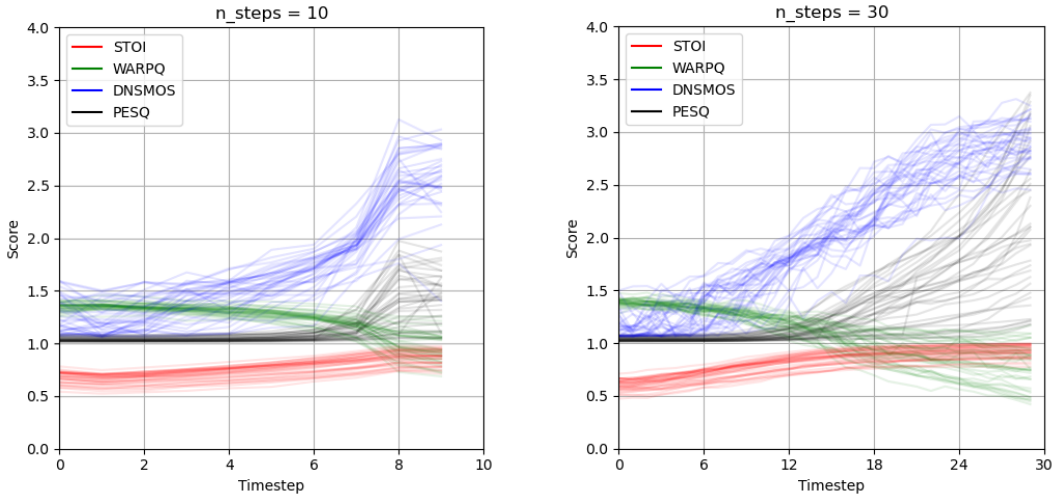


Figure 9.2: Overview of 0-indexed score evolution for all datapoints during reverse-diffusion for linear

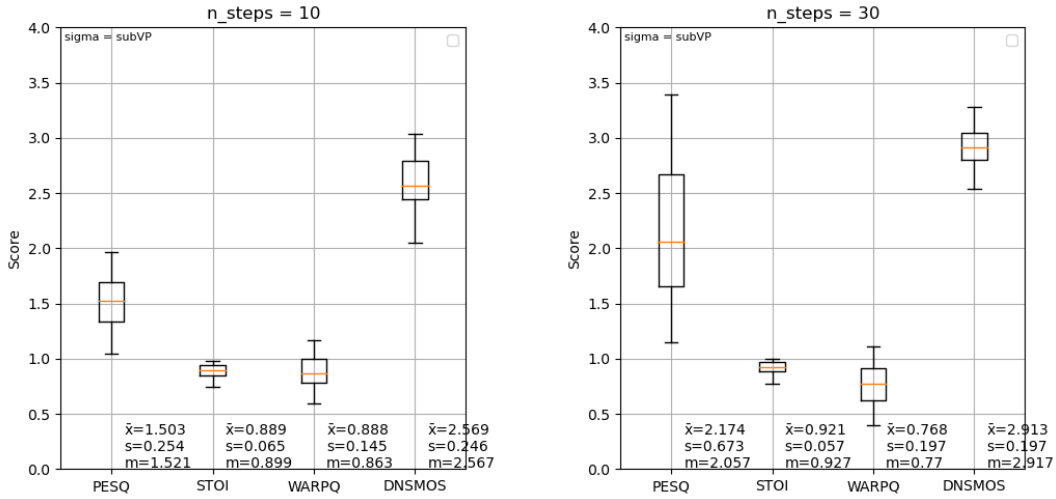


Figure 9.3: Box plots of reverse diffusion for subVP

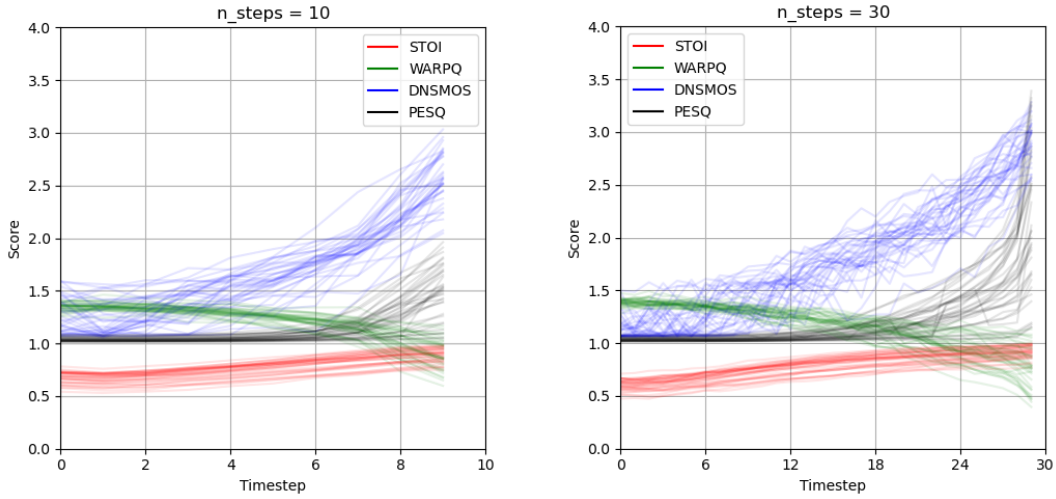


Figure 9.4: Overview of 0-indexed score evolution for all datapoints during reverse-diffusion for subVP

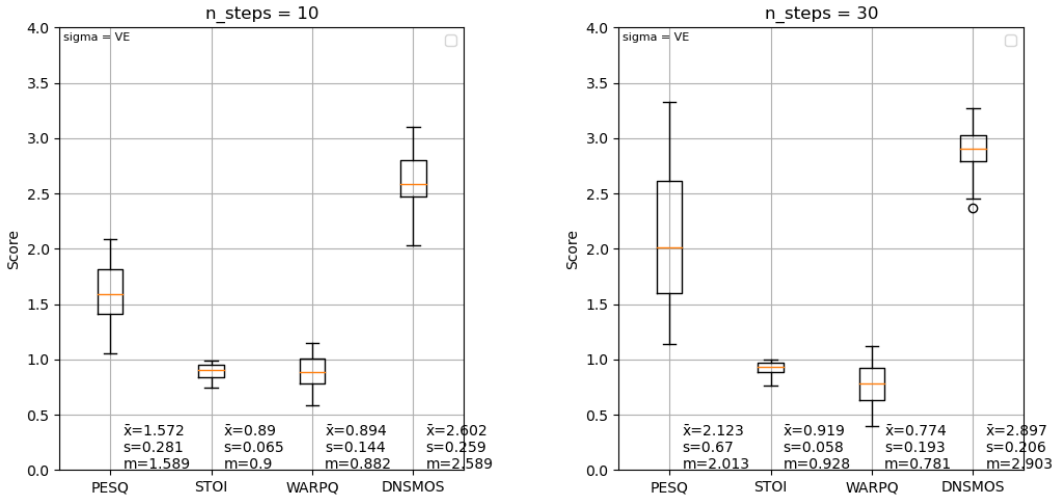


Figure 9.5: Box plots of reverse diffusion for VE

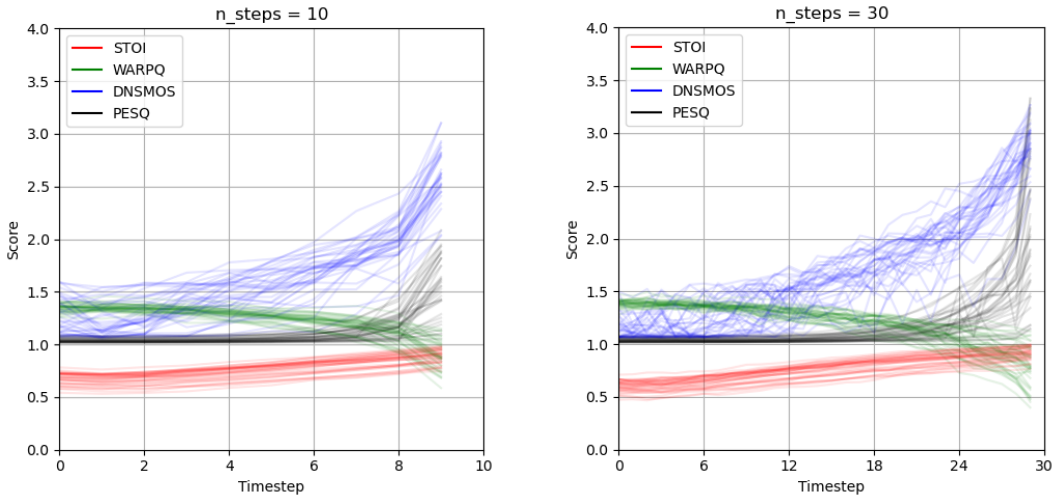


Figure 9.6: Overview of 0-indexed score evolution for all datapoints during reverse-diffusion for VE

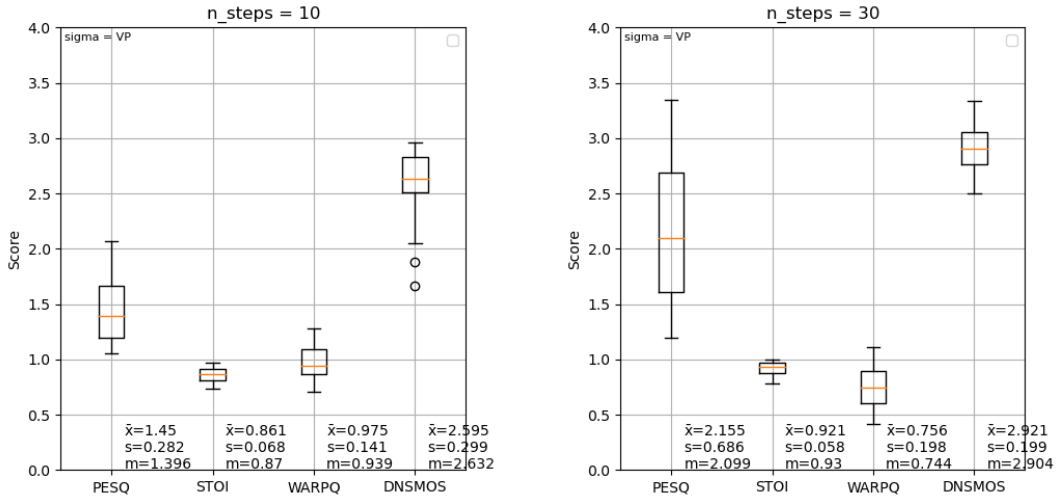


Figure 9.7: Box plots of reverse diffusion for VP

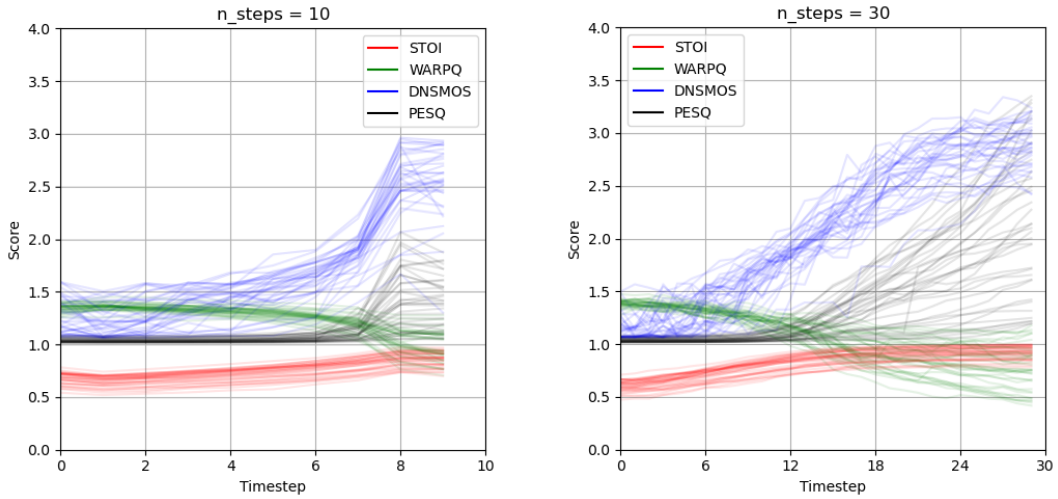


Figure 9.8: Overview of 0-indexed score evolution for all datapoints during reverse-diffusion for VP

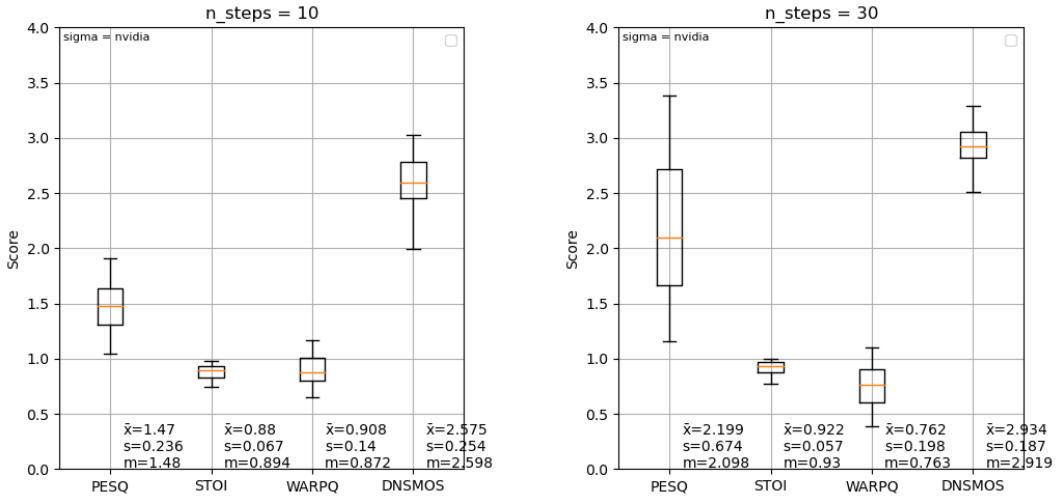


Figure 9.9: Box plots of reverse diffusion for nvidia

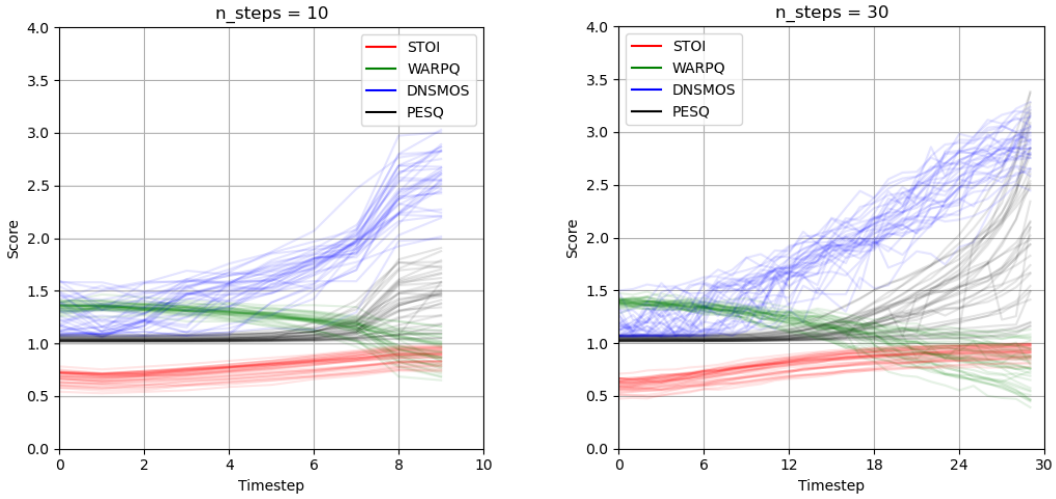


Figure 9.10: Overview of 0-indexed score evolution for all datapoints during reverse-diffusion for nvidia

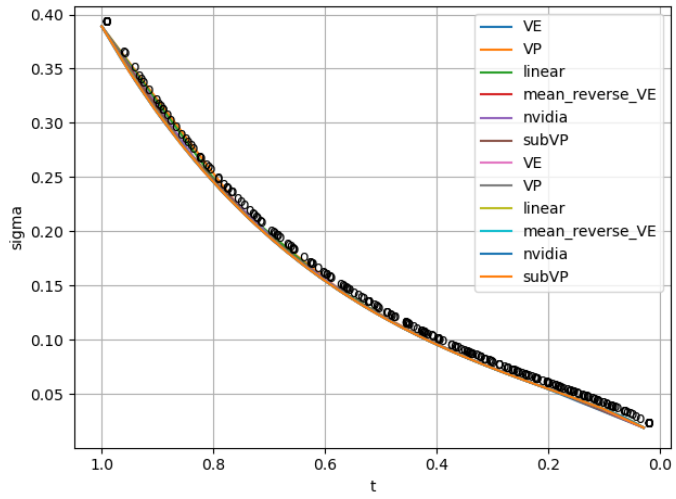


Figure 9.11: Overview of all scheduler functions stacked and plotted from log files generated during Experiment 1. Seperated plots are available in the supplementary material.

9.5 Plots of Experiment 2

Box and evolution plots for a modified derivative of the baseline σ function. *Alpha* $\in$ [1.05, 1.1, 1.2, 1.5]

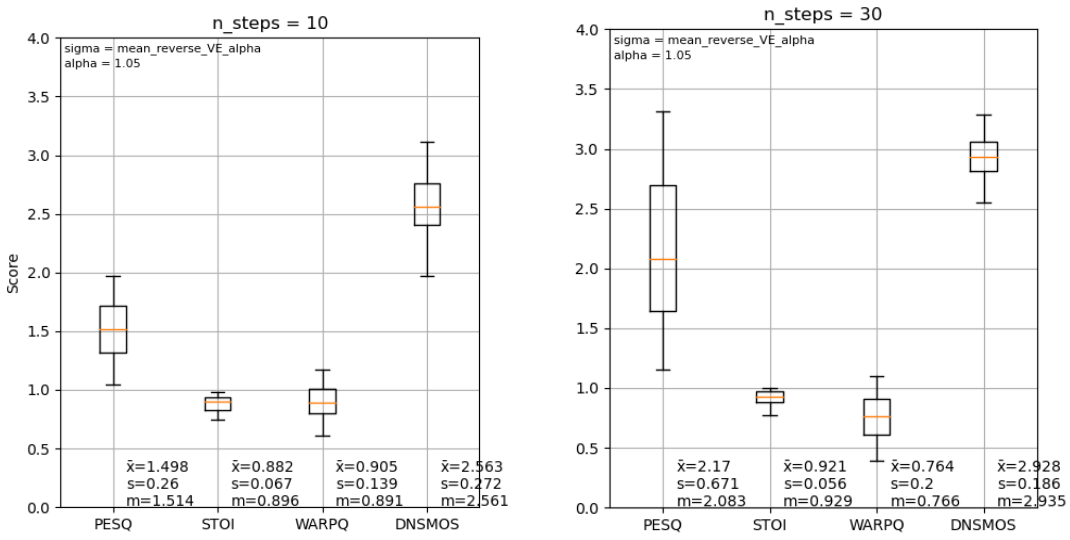


Figure 9.12: Box plots of reverse diffusion for alpha=1.05

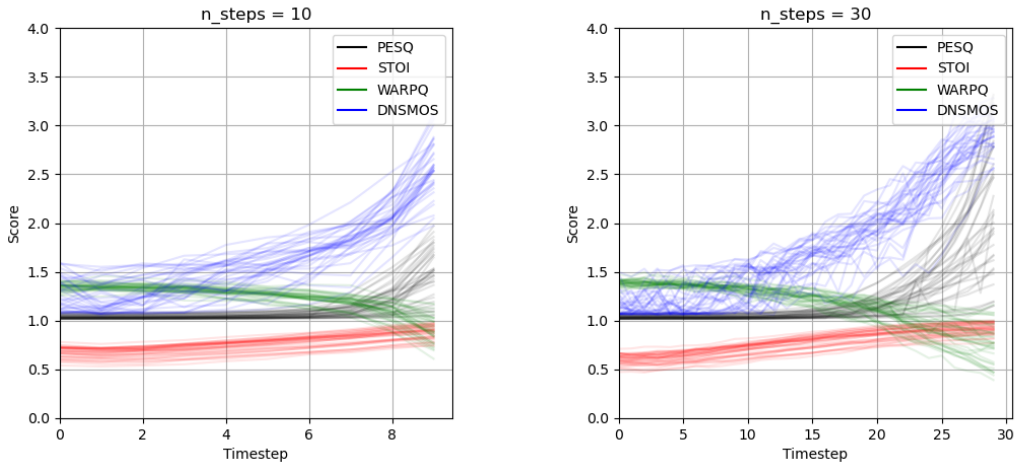


Figure 9.13: Overview of 0-indexed score evolution for all datapoints during reverse-diffusion for alpha=1.05

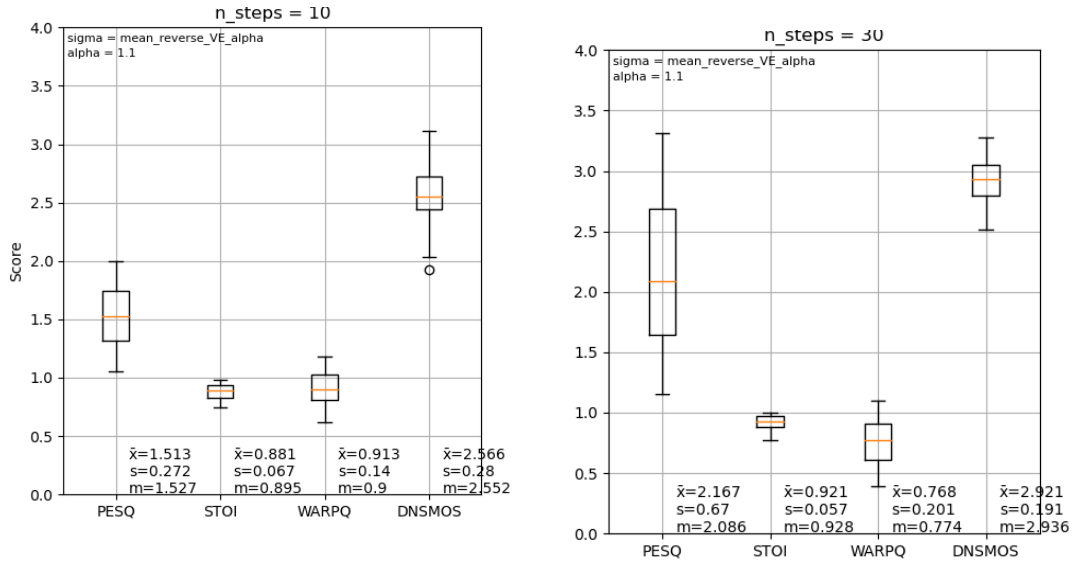


Figure 9.14: Box plots of reverse diffusion for alpha=1.1

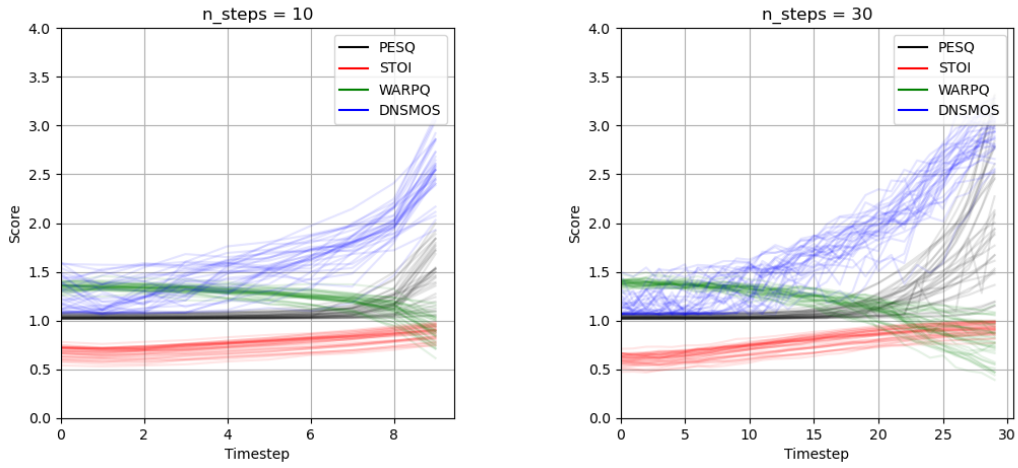


Figure 9.15: Overview of 0-indexed score evolution for all datapoints during reverse-diffusion for alpha=1.1

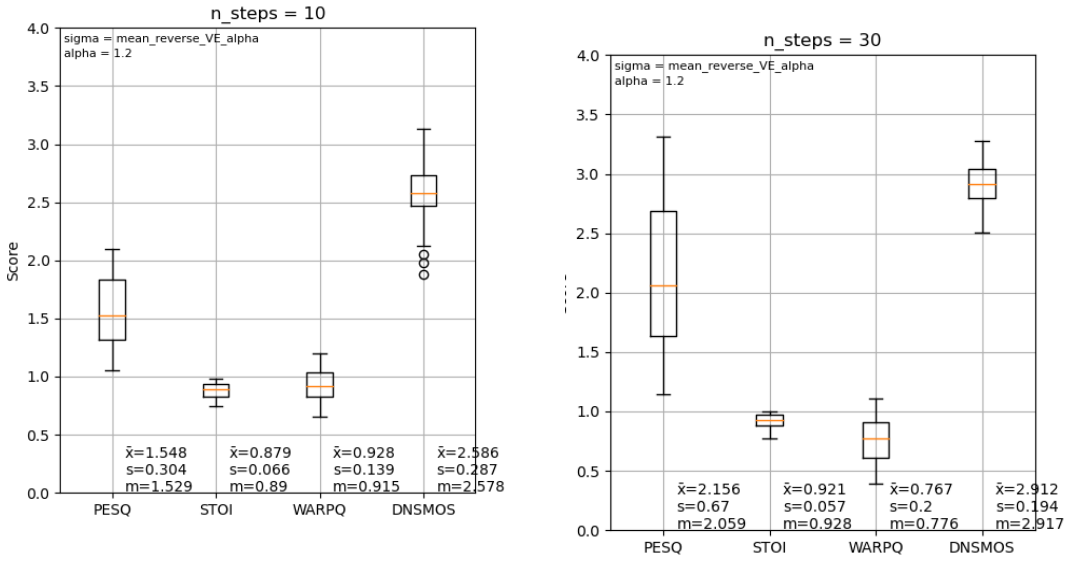


Figure 9.16: Box plots of reverse diffusion for alpha=1.2

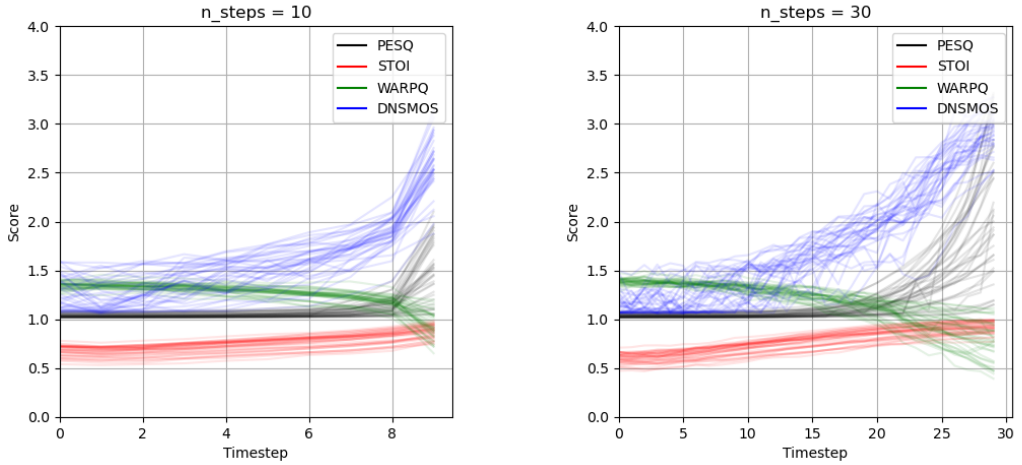


Figure 9.17: Overview of 0-indexed score evolution for all datapoints during reverse-diffusion for alpha=1.2

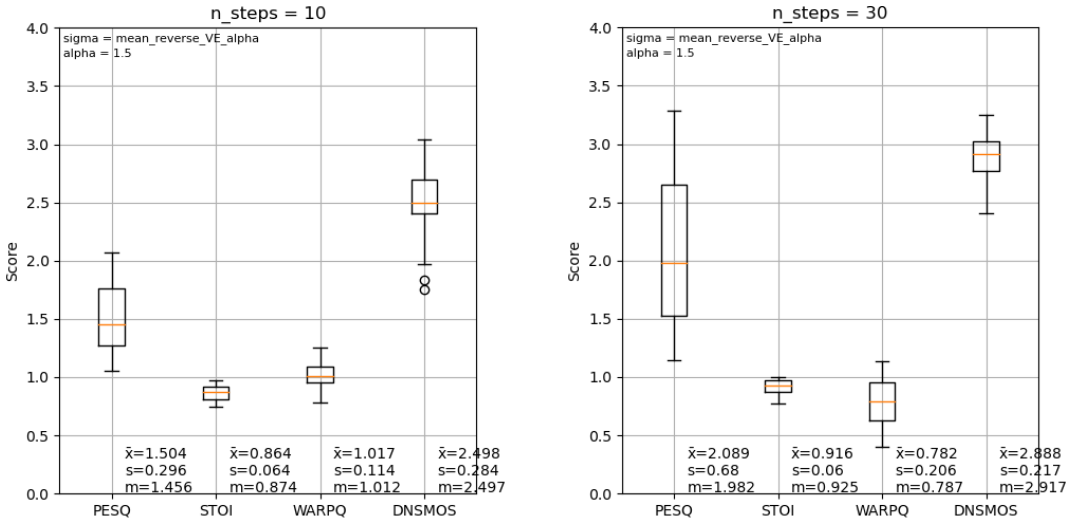


Figure 9.18: Box plots of reverse diffusion for alpha=1.5

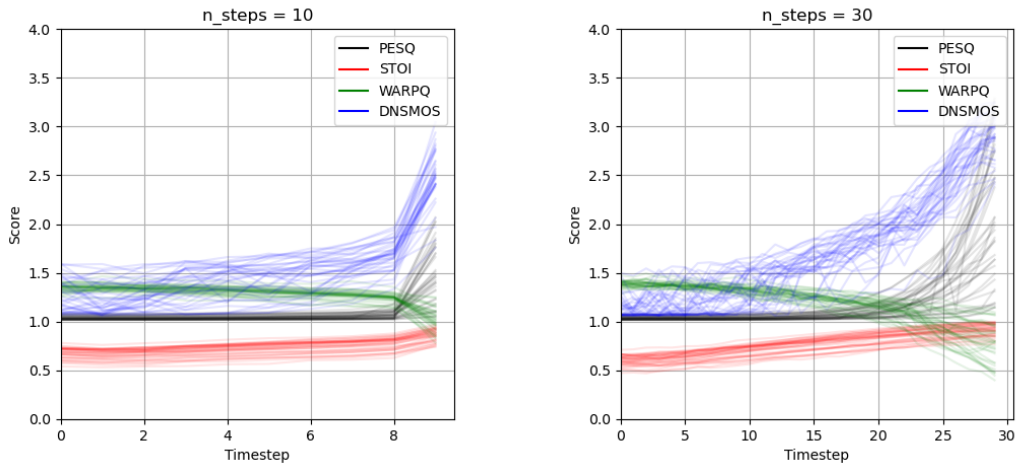


Figure 9.19: Overview of 0-indexed score evolution for all datapoints during reverse-diffusion for alpha=1.5

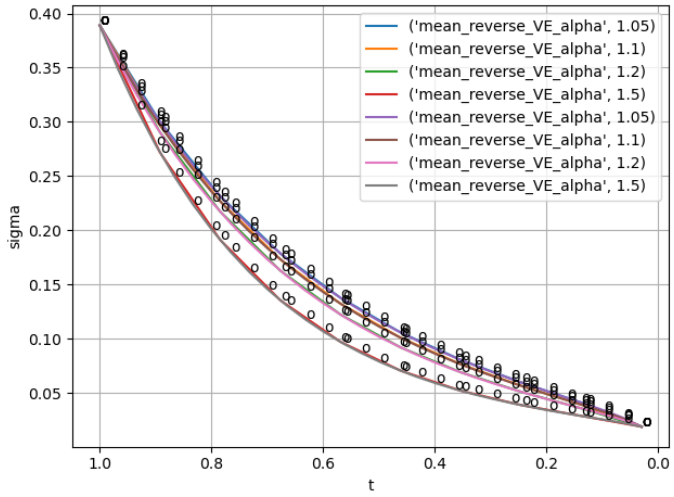


Figure 9.20: Overview of all scheduler functions used in Experiment 2, stacked and plotted from log files generated during Experiment 2. Seperated plots are available in the supplementary material. Dots denote sampling points

9.6 Plots of Experiment 3

Box plots and score evolution for Experiment 3 of modified t step values for fixed σ values computed from the baseline function.

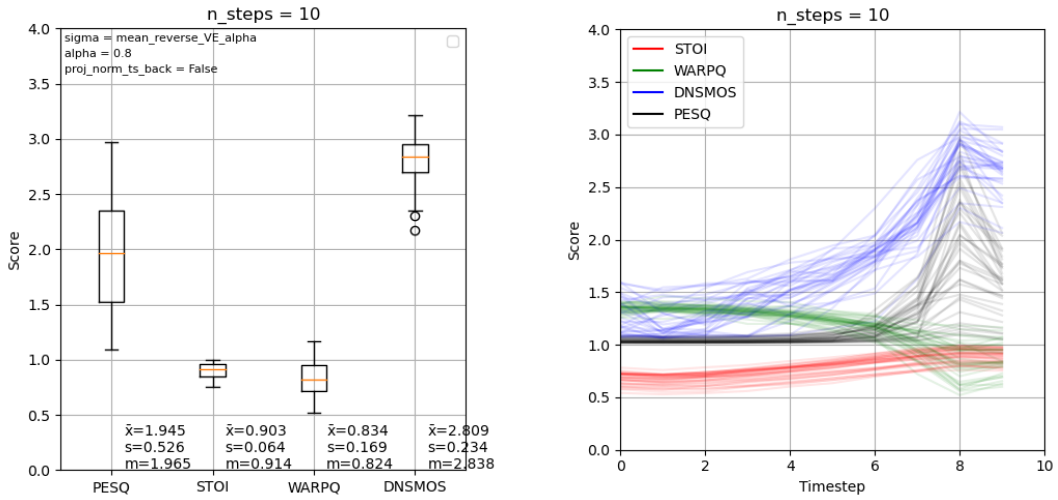


Figure 9.21: Box-plot and score evolution for modified baseline *scheduler* with number of 0-indexed timesteps *t=10*.

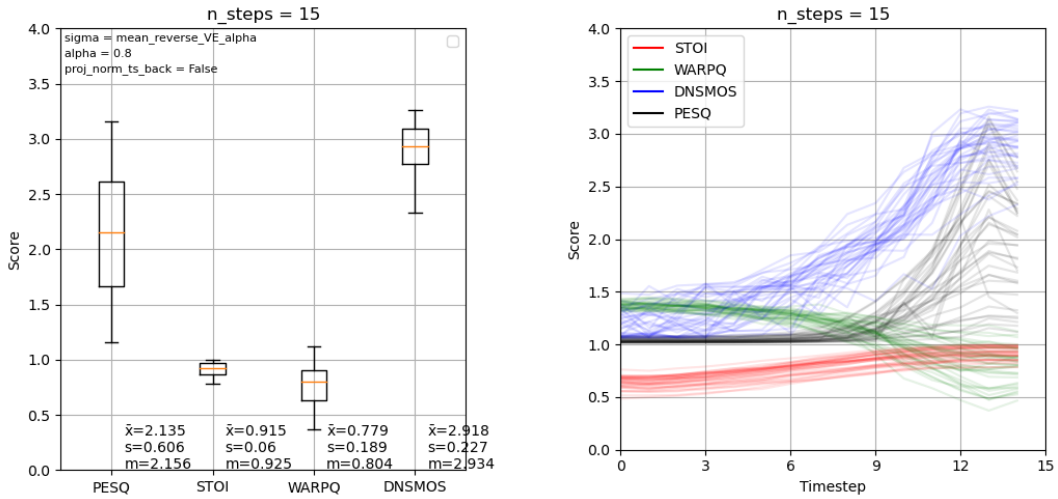


Figure 9.22: Box-plot and score evolution for modified baseline *scheduler* with number of 0-indexed timesteps *t=15*.

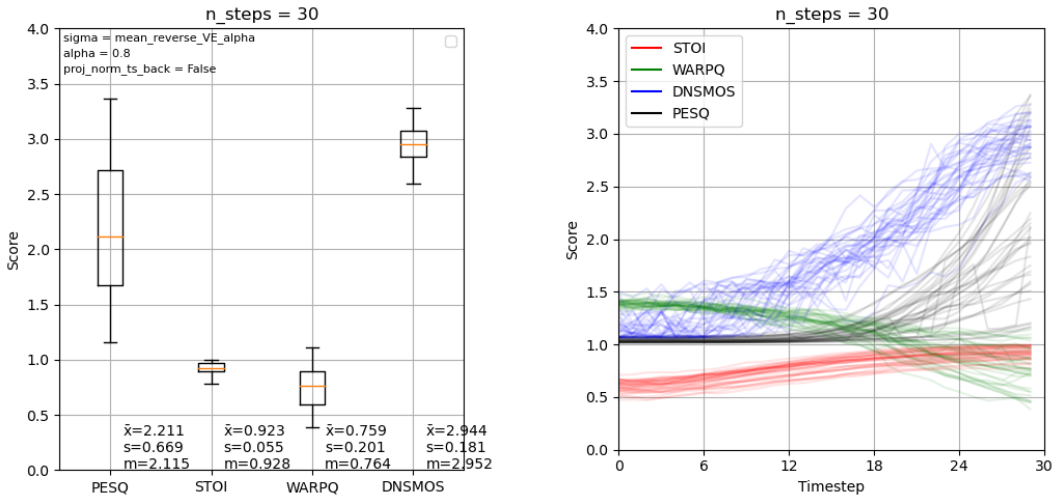


Figure 9.23: Box-plot and score evolution for modified baseline *scheduler* with number of 0-indexed timesteps *t=30*.

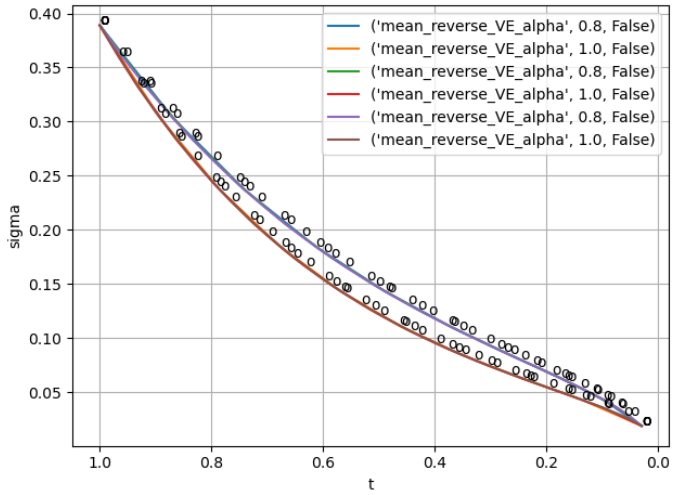


Figure 9.24: Overview of all scheduler functions stacked and plotted from log files generated during Experiment 3. Seperated plots are available in the supplementary material.

9.7 Plots of Experiment 4

Box plots and score evolution for Experiment 4 of computed σ values based on t values from Experiment 3.

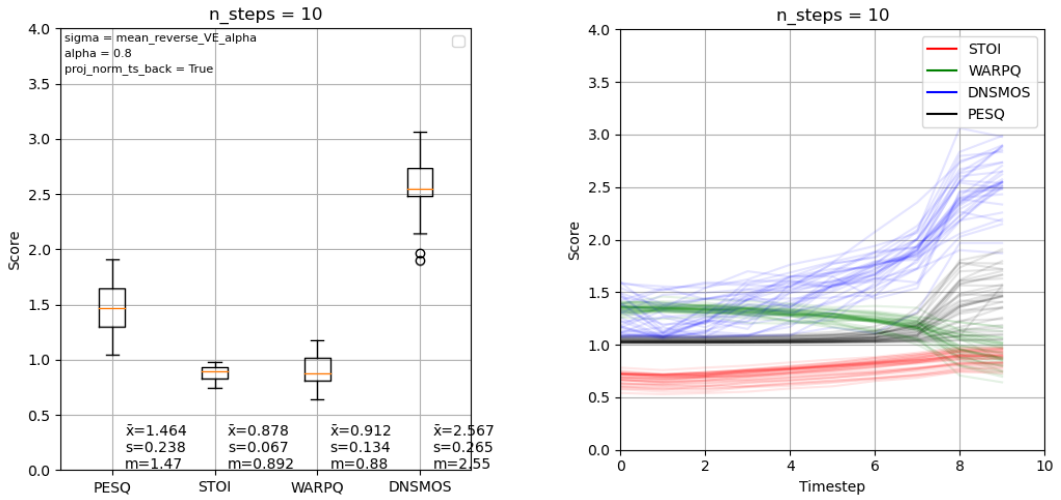


Figure 9.25: Box-plot and score evolution for modified non-linear discretization step spacing based on a modified baseline function from Experiment 3. Number of 0-indexed timesteps *t=10*.

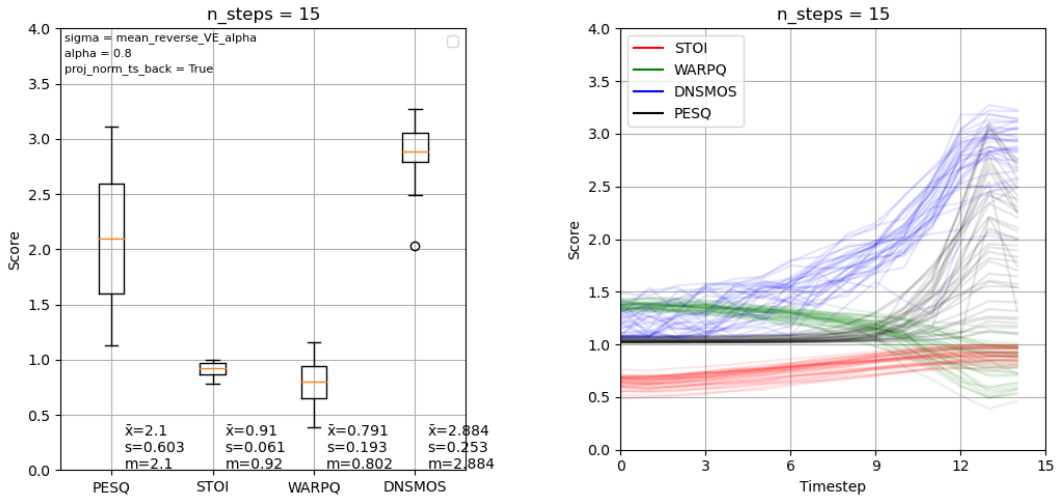


Figure 9.26: Box-plot and score evolution for modified non-linear discretization step spacing based on a modified baseline function from Experiment 3. Number of 0-indexed timesteps *t=15*.

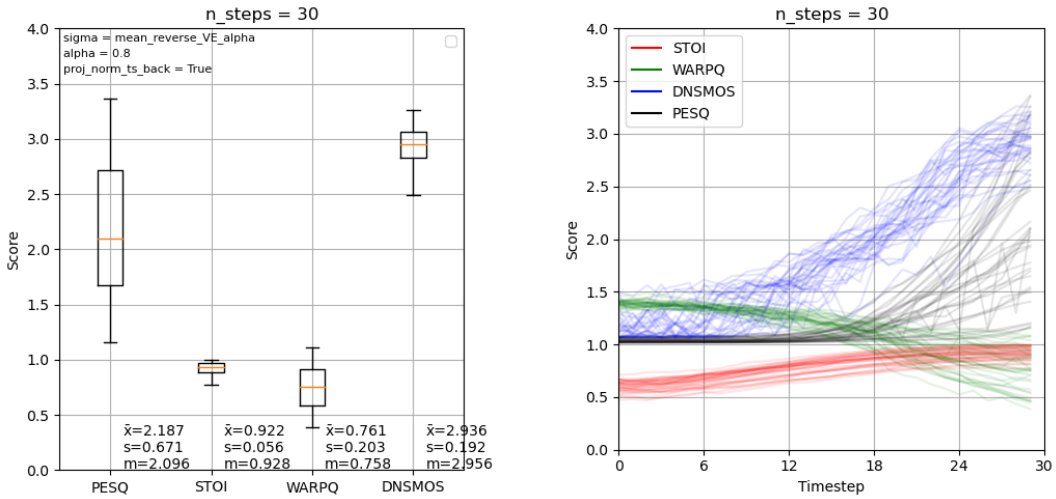


Figure 9.27: Box-plot and score evolution for modified non-linear discretization step spacing based on a modified baseline function from Experiment 3. Number of 0-indexed timesteps *t=30*.

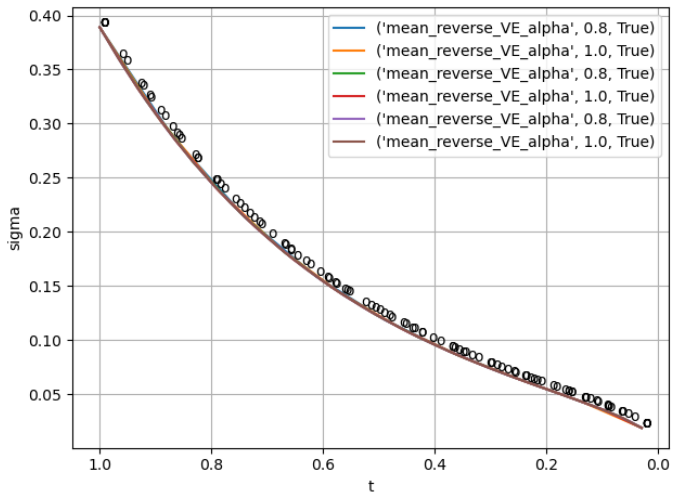


Figure 9.28: Overview of all scheduler functions stacked and plotted from log files generated during Experiment 4. Seperated plots are available in the supplementary material.