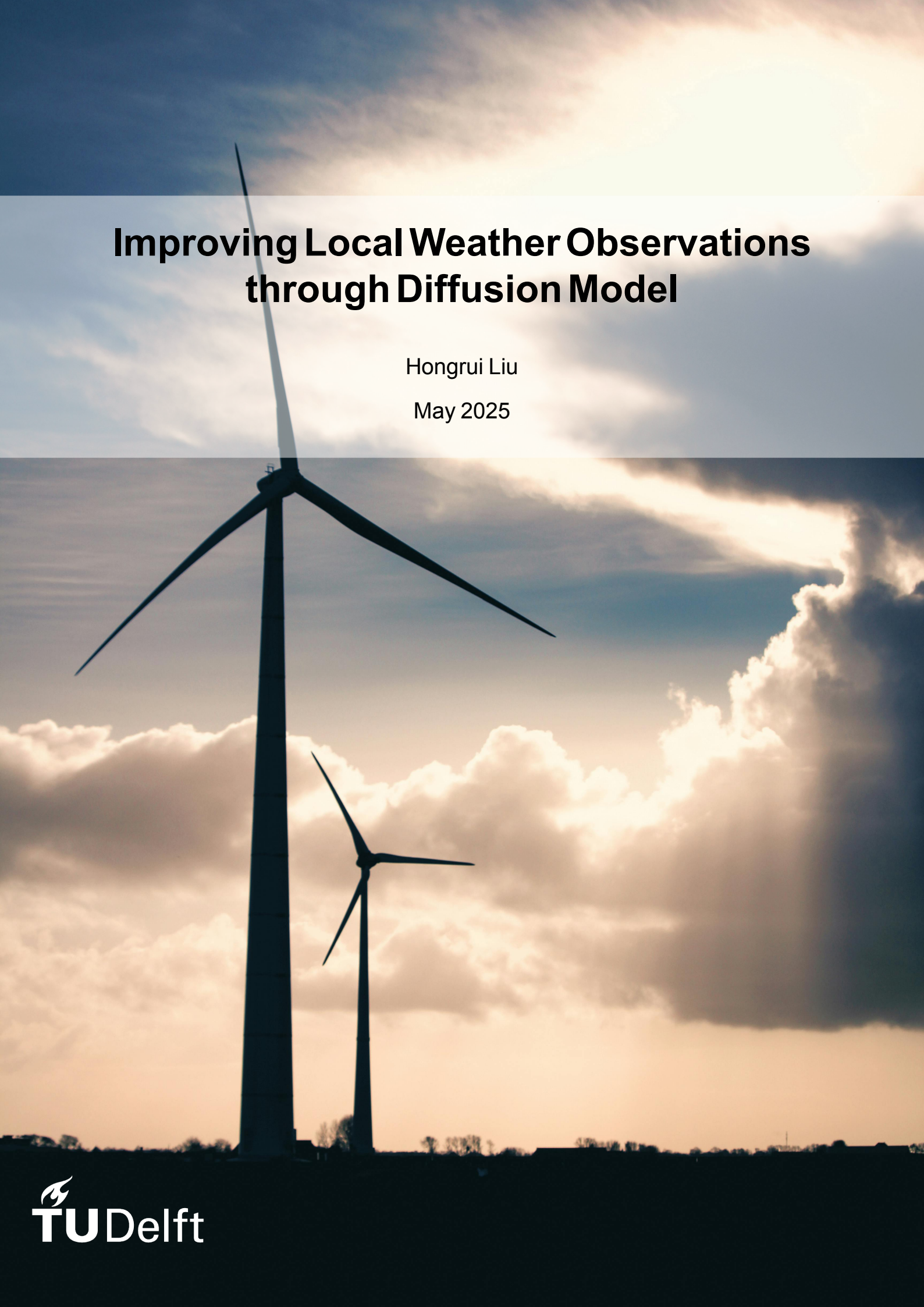


Improving Local Weather Observations through Diffusion Model

Hongrui Liu

May 2025

TU Delft

Improving Local Weather Observations through Diffusion Model

by

Hongrui Liu

to obtain the degree of Master of Science
in Applied Mathematics at the Delft University of Technology,
to be defended publicly on Monday May 26, 2025 at 13:00 PM.

Student number:	6005608	
Project duration:	November 1, 2024 – May 20, 2025	
Thesis committee:	Prof. dr. H. Kekkonen,	TU Delft, Chair and Responsible supervisor
	Prof. dr. J. Sun,	TU Delft, Core member and Daily supervisor
	Prof. dr. A. Heinlein,	TU Delft, Core member
	Prof. dr. R. Verzijlbergh,	TU Delft, Daily supervisor

TU Delft

Acknowledgments

Two years have passed in the blink of an eye. Every day spent in Delft is deeply etched in my memory. As I look back, my heart is filled with gratitude for all those who have supported and accompanied me on this journey.

First and foremost, I want to thank myself. Graduating from TU Delft has never been an easy feat. Over these two years, I faced numerous exams, countless assignments, and relentless deadlines. Yet, every ounce of effort and perseverance has been more than worth it.

I am profoundly grateful to my parents, Mr. Qilin Liu and Ms. Qiong Li. Their unwavering support and encouragement have been my greatest strength. Behind every decision I made, they stood silently by my side, offering unconditional love and sacrifice.

I also want to thank my friends—both those from China (Yishu Liu, Yu Chen, Ziqing Zhang, Xiaoling Xia, Yijun Li, etc.)and those I have come to know here in the Netherlands (Wei Liu, Yuran Wang, Xiner Zhou, Chen Liang, Jun Lin, Zhewen Gao, Yanqi Hong, Shipeng Liu, etc.). Thank you for patiently listening during my moments of doubt, sharing joyful gatherings, and joining me in study sessions. Your companionship has been a precious source of comfort and strength.

My sincere thanks go to my thesis supervisors, Dr. Hanne Kekkonen, Dr. Jing Sun and Dr. Remco Verzijlbergh. Your guidance and insightful feedback have greatly enriched my research and helped me refine my work. I am also deeply thankful to Whiffle for entrusting me with this project and providing invaluable support throughout my thesis journey. Furthermore, I appreciate all the professors who have taught me; your profound knowledge and rigorous academic spirit have been instrumental in broadening my horizons.

With a heart full of gratitude, I cherish every bit of support and encouragement I have received. It is because of you all that I stand where I am today. Moving forward, I will carry this gratitude with me and face the future with courage and determination.

Hongrui Liu

Delft, May 2025

Abstract

Reconstructing high-resolution wind fields from sparse, low-resolution observations is a critical yet ill-posed problem in meteorological modeling. Classical approaches, such as Computational Fluid Dynamics (CFD), are often too computationally intensive to meet the demands of real-time or large-scale industrial applications. Meanwhile, conventional data-driven methods like Convolutional Neural Networks (CNNs) tend to produce overly smoothed outputs and struggle to recover fine-scale structures, especially under severe data sparsity.

This thesis explores the use of diffusion-based generative models for super-resolution in wind field reconstruction. A progressive SR3 (Super-Resolution via Repeated Refinement) framework is developed, combining a multi-stage architecture with stochastic denoising processes to gradually reconstruct high-resolution outputs. Extensive experiments demonstrate that the progressive SR3 consistently outperforms CNN-based baselines in terms of reconstruction accuracy, perceptual quality, and robustness. Furthermore, a joint training strategy improves both performance and computational efficiency by enabling end-to-end optimization across stages.

The findings support the use of probabilistic diffusion models for meteorological super-resolution tasks and emphasize the effectiveness of progressive refinement in handling large upscaling factors. This approach provides a promising pathway for enhancing data-driven post-processing in atmospheric modeling.

The data is provided in https://gitlab.com/whiffle-public/whiffle-open-data, and code is available at: https://github.com/Homjui/Upscale_wind_field.

Keywords: diffusion models, SR3, super-resolution, wind field.

Contents

1 Introduction

Sustainability has become a critical global concern, driving substantial investments from governments and organizations worldwide. The European union, for instance, has allocated significant funding to initiatives promoting renewable energy sources, recognizing their potential to mitigate climate change by reducing carbon emissions [54]. Among these renewable energy sources, wind energy stands out due to its scalability and minimal environmental footprint [2]. However, optimizing wind energy production requires high-resolution wind data, which can provide precise support for research areas such as wind farm optimization [8], air quality monitoring [53], and wildfire risk assessment [17].

Accurately capturing wind behavior remains a significant challenge due to two primary difficulties. First, turbulence exhibits complex, inherently random dynamics influenced by various atmospheric factors, including humidity, pressure, and temperature [39]. Second, practical constraints on measurement infrastructure and associated costs limit the availability of turbulence observations, resulting in sparse data coverage. The combination of turbulence complexity and practical measurement limitations makes obtaining high-resolution wind data a persistent challenge.

To address this issue, researchers have explored multiple approaches to infer high-resolution turbulence data from sparse observations. Experimental techniques such as Particle Image Velocimetry (PIV) [1] enable the capture of detailed turbulent flow structures, while Computational Fluid Dynamics (CFD) [3] methods, including large eddy simulation (LES) [46] and Direct Numerical Simulation (DNS), provide numerical solutions for turbulence modeling. Despite their accuracy, these approaches are computationally expensive and time-consuming [6], limiting their practicality in real-time and large-scale industrial applications.

In recent years, data-driven methodologies have emerged as promising alternatives in turbulence mechanics [7, 44, 56]. By integrating sparse measurement data with machine learning models, researchers have leveraged deep learning techniques, such as Convolutional Neural Networks (CNNs) [13] and Generative Adversarial Networks (GANs) [9], to enhance spatial and temporal resolution, effectively reconstructing turbulence fields with reduced computational effort. With the advancement of deep learning methods, Diffusion Models (DMs) have demonstrated superior performance in high-resolution reconstruction problems [10] due to their ability to learn the underlying probability distribution of datasets. Motivated by these advancements, this study explores the application of DMs for turbulence field reconstruction, aiming to enhance the resolution of sparse wind observations efficiently and accurately.

1.1 Related Work

The thesis explores the application of DMs in enhancing the accuracy and resolution of local weather observations. This research intersects two key areas: meteorological upscaling tech-

niques and DMs. The first research domain, often referred to as super-resolution methods [13], includes both traditional numerical and statistical approaches as well as modern data-driven techniques. Although DMs, a class of deep generative models that have shown remarkable success in image processing tasks [10], are still in the early stages of recognition in meteorological applications, their probabilistic formulation and ability to generate high-quality data have made them increasingly relevant. Recent studies suggest their potential in improving the resolution and accuracy of weather observations [40].

1.1.1 Super-resolution methods in meteorological application

In many application scenarios of computer vision, from everyday photography to refining satellite [51] and medical images [22], Super-resolution tasks are of high demand. Super-resolution problem, which aims to enhance low-resolution images into high-resolution counterparts, received tremendous focus due to its ill-posed nature. In literature, a wide range of classical super-resolution methods have been explored, including prediction-based methods [25], edge-based methods [52] and statistical methods [27], etc. As deep learning continues to evolve, super-resolution techniques based on deep neural networks have gained increasing attention. Early methods leveraged CNNs, while more recent approaches have adopted GANs, which offer alternative solutions to the super-resolution problem.

CNNs have been widely employed as a fundamental approach for addressing the super-resolution problem, leveraging their ability to capture the nonlinear relationships between input and output data through spatial feature extraction via filtering operations. Dong et al. [11] pioneered the application of CNNs for super-resolution, which have since inspired various advancements in different domains. Building on this foundation, Fukami et al. [13] expanded CNNs to weather and climate applications where they utilized CNNs to reconstruct high-resolution turbulent flow fields from coarse input data, effectively capturing essential spatio-temporal dynamics with high fidelity. Similarly, Liu et al. [35] developed a CNN-based classification system for extreme climate event detection, achieving an accuracy of 89%–99% in identifying phenomena such as tropical cyclones, atmospheric rivers, and weather fronts.

Numerous studies have demonstrated that the adoption of CNNs significantly enhances model performance across a range of applications. Weyn et al. [57] constructed a deep CNN-based framework for global weather forecasting, achieving stable long-term predictions and outperforming coarse-resolution numerical weather prediction models in short- to medium-range forecasts. Liu et al. [34] further extended CNN architectures by introducing the static CNNs and the Multiple Temporal Paths CNNs (MTPC), designed to extract both spatial and temporal features. Their results showed that the MTPC model exhibited superior accuracy in reconstructing turbulent flows and outperformed static models in capturing anisotropic turbulence characteristics, highlighting the potential of CNN-based approaches for advancing data-driven weather and climate modeling.

Beyond supervised learning methods, unsupervised learning architectures, particularly GANs, provided useful inspirations on how to tackle super-resolution problems. In the GANs framework, a generator is trained to capture the underlying data distribution by generating synthetic samples that attempt to deceive a discriminator, which evaluates their authenticity [15]. Ledig et al. [31] were among the first to introduce GAN-based architectures for super-resolution, demonstrating that incorporating pixel- and feature-based loss functions led to the generation of more perceptually realistic images compared to previous state-of-the-art methods. Inspired by their work, many studies have extended GANs to weather and climate applications. Stengel et al. [51] applied GANs to super-resolve wind velocity and solar irradiance, demonstrating its potential for renewable energy resource assessment. Similarly, Chen et al.[9] applied GANs to radar echo data super-resolution, effectively preserving edge details and textural features critical for meteorological analysis.

In addition to standard GANs, researchers have explored variational and physics-informed GANs architectures to better capture complex problem settings and data structures. Xie et al. [58] developed a conditional GAN with temporal and spatial discriminators to generate coherent and detailed four-dimensional physics fields using low-resolution inputs. Their results showed that physics-aware data augmentation was able to enhance the accuracy and efficiency of the model. Yousif et al. [61] proposed a multi-scale super-resolution GAN with a physics-based loss function, which achieved accurate results with reduced training data and computational cost. Their approach successfully handled both laminar and turbulent flow regimes, showcasing the potential of physics-aware GANs for super-resolution tasks in geophysical and atmospheric sciences.

Beyond CNNs and GANs, various deep learning architectures have been explored for super-resolution tasks, including recursive neural networks [28] and residual attention networks [62], etc. These methods incorporates novel mechanisms to enhance feature extraction, stability, and reconstruction fidelity, while they are interconnected in their objective of learning high-resolution representations from degraded inputs. Wang et al. [55] considered that the primary differences among deep learning-based super-resolution methods can be categorized into several key aspects: the choice of network architectures [26], the formulation of loss functions [23] and the learning principles and optimization strategies employed during training [32].

Despite the substantial advancements in deep learning-based super-resolution models, determining the most effective architecture remains an open challenge, as each approach has its own strengths and limitations. CNN-based models have demonstrated strong performance in extracting spatial features and are computationally efficient, making them well-suited for real-time applications [13]. However, they often struggle with capturing long-range dependencies, which limits their ability to reconstruct fine-grained details in highly complex scenes [55]. GAN-based models, on the other hand, have been particularly successful in producing perceptually realistic high-resolution outputs [31], but they suffer from issues such as mode collapse and instability

during training [15]. Additionally, while GANs can generate sharp and visually appealing results, they may not always preserve physical consistency in scientific applications such as weather forecasting and fluid dynamics [51, 58, 61].

1.1.2 Diffusion models for high-resolution weather reconstruction

The introduction of DMs, pioneered by Ho et al. [19], represents a paradigm shift in generative modeling, offering a compelling alternative to the long-standing dominance of GANs [10]. These models have demonstrated remarkable success in image generation and reconstruction tasks, including super-resolution. Unlike adversarial training in GANs, which often suffers from mode collapse and instability, DMs employ a probabilistic denoising framework that ensures more stable and diverse outputs [47]. The fundamental mechanism of classical DMs, as established by Ho et al. [19], is based on a two-stage process: a forward noising process and a reverse denoising process. These structured noise-to-data mapping enables DMs to capture intricate spatial strucures, making them highly effectively for super-resolution applications.

Rombach et al. [38] demonstrated the ability of DMs to generate high-resolution and realistic images, showcasing their potential in addressing complex image tasks. Building on this foundation, researchers have extended DMs to the climate and weather domains, leveraging their generative capacity to enhance the availability and resolution of meteorological data. Merizzi et al. [37] applied DMs to address the temporal lag in reanalysis datasets by formulating the problem as a super-resolution task, thereby improving the accessibility of high-resolution climate data. Ling et al. [33] introduced a two-stage diffusion framework to enhance long-term rainfall prediction by mitigating issues such as blurriness and spatial distortions. Their approach first captures temporal dependencies in a low-resolution setting before refining spatial structures via high-resolution reconstruction, achieving a 5%-10% improvement over baseline methods. Hatanaka et al. [16] employed score-based DMs to refine coarse numerical weather prediction outputs into high-resolution satellite observations, demonstrating substantial improvements in day-ahead solar irradiance forecasting.

As tasks become increasingly complex, traditional DMs often struggle to meet the growing demands of high-resolution reconstruction and computational efficiency. Consequently, recent advancements have led to refined variations of DMs, tailored to specific super-resolution challenges. Conditional DMs, such as Super-Resolution via Repeated Refinement (SR3) [43] and Cascaded Diffusion Models (CDMs) [20], incorporate high-resolution conditioning signals to enhance reconstruction fidelity, outperforming conventional methods in both perceptual quality and structural coherence. Latent Diffusion Models (LDMs) [38], another prominent variant, map data into a lower-dimensional latent space for the diffusion process, significantly improving computational efficiency without sacrificing visual quality. Additionally, score-based DMs [49] leverage score matching to iteratively denoise and reconstruct high-resolution images, further enhancing the accuracy of fine-grained details in generated outputs. These innovations collect-

ively expand the applicability of DMs in scientific and engineering domains, particularly for high-resolution climate and atmospheric modeling.

Rybchuk et al. [40] pioneered the use of LDMs for reconstructing three-dimensional turbulence from sparse, noise-free observations. Their research highlighted the ability of LDMs to address inherent uncertainties through ensemble generation, providing diverse and realistic samples. In subsequent work, Rybchuk et al. [41, 42] expands LDMs into temporal dimensions. Their results demonstrated a high correlation with ground truth and a low bias in turbine inflow predictions. Mardani et al. [36] introduced Residual Corrective Diffusion Modeling (CorrDiff), a two-stage generative framework. Different from classical DMs, CorrDiff separates the deterministic and stochastic components of the data: the first stage uses a U-Net to predict large-scale atmospheric trends, while the second stage employs DMs to refine residual errors and capture fine-scale variability. This focus on residual modeling proves highly effective for weather data, where small-scale details are critical.

1.1.3 Research gap

Data-driven methods, particularly deep learning, have been extensively applied to weather upscaling problems, achieving computationally cheaper and more time-efficient results compared to traditional numerical approaches such as CFD [3]. Nevertheless, several important challenges remain:

- From model selection perspective, while DMs have been demonstrated as highly effective for super-resolution tasks in computer vision domains [10], their application in weather observation upscaling remains largely unexplored. Most existing research has focused on CNNs and GANs to reconstruct high-resolution data from sparse weather observations [13, 14], with little attention given to the potential of DMs. This research gap may be attributed to the relatively recent emergence of DMs. Given their superior performance in generating high-fidelity reconstructions with improved stability compared to GANs, investigating their role in weather upscaling presents a promising avenue for future research.

- From the dimension of super-resolution perspective, the majority of DMs weather super-resolution studies prioritize improving the temporal resolution of meteorological data, addressing issues such as time lag [37] and forecast accuracy [16, 33]. However, practical constraints on measurement infrastructure of meteorological data makes geographical upscaling equally crucial. Fine-grained spatial upscaling can enhance the ability to capture localized weather features with higher precision, enabling more accurate regional climate modeling and better applicability in real-world scenarios, such as wind energy forecasting [8] and extreme weather prediction [35].

- From super-resolution tasks perspective, existing studies applied DMs to meteorological super-resolution on rainfall [33] and solar irradiance [16], with relatively few investigations

into turbulence modeling. Whereas, wind flow exhibits high stochasticity and nonlinear dynamics, making it particularly challenging to capture and reconstruct accurately [24]. The absence of extensive research on turbulence upscaling highlights a critical gap, as high-resolution wind field reconstruction is essential for applications in sustainable energy utilization [8], aviation safety [53], and climate risk assessment [17].

Given these research gaps, this thesis aims to explore the application of DMs in turbulence geographical upscaling, filling a critical void in the field. By leveraging the strengths of DMs in generating high-fidelity reconstructions, this study seeks to enhance the precision of wind flow modeling. Improving the resolution of turbulence data has significant practical implications, particularly in optimizing wind energy generation, where a more accurate representation of wind patterns can lead to better resource allocation and efficiency in sustainable energy utilization.

The thesis research is closely related to the work of Rybchuk et al. [40], who focus on the reconstruction of inpainting of turbulence using DMs. Both studies leverage DMs to recover fine-grained spatial turbulence data in local regions. However, there are several key distinctions between this thesis and their research.

First, in terms of objectives, while Rybchuk et al. [40] address the problem of inpainting masked areas within a wind field, this thesis focuses on reconstructing high-resolution turbulence data from limited sparse observations. Although both tasks can be regarded as interpolation and extrapolation of incomplete dataset, the fundamental difference lies in the nature of the missing information. Inpainting typically assumes large, contiguous missing regions, which poses challenges in preserving global spatial consistency. In contrast, sparse reconstruction involves scattered and irregularly sampled data points, which makes it particularly difficult to recover fine-scale structures and local coherence [4]. Both are inherently difficult problems but demand different strategies due to the distinct characteristics of their missing data.

Second, in terms of methodology, both studies adopt DMs as the core framework, but they utilize different model variants. Rybchuk et al. [40] employ LDMs, which operate in a compressed latent space to improve efficiency [38], whereas this thesis adopts SR3, a super-resolution DMs designed to directly enhance spatial resolution conditioning on low-resolution inputs [43]. The primary distinction lies in between computational efficiency and reconstruction accuracy. LDMs reduce computational costs by working in a lower-dimensional representation, potentially sacrificing fine details, while SR3 preserves high-frequency features by operating in pixel space, making it more suitable for super-resolution problems.

1.2 Structure of Thesis

The remainder of this thesis is organized as follows:

- Section 2 presents the mathematical background necessary to understand the core methodology. It first introduces the super-resolution problem from an inverse problem perspective

and discusses common downsampling methods and quality metrics. It then summarizes foundational deep learning concepts, including neural network structures, autoencoders, and optimization techniques.

- Section 3 introduces the Super-Resolution via Repeated Refinement (SR3) framework. It details the theory of diffusion models, then formalizes SR3 for super-resolution tasks, where the training and sampling procedures are thoroughly discussed and rigorously derived.
- Section 4 describes the experimental setup and implementation. It includes dataset preparation, model configurations, training procedures, and baseline comparisons with CNNs. Experiments cover both direct and progressive upscaling strategies, hyperparameter tuning and joint learning.
- Section 5 discusses the experimental results, providing insights into model behavior and architectural choices. The section also presents final conclusions and suggests directions for future work.

2 Mathematical Backgrounds

2.1 Super-resolution

Super-resolution refers to the process of reconstructing high-resolution image from its corresponding low-resolution image. This technique has numerous practical applications, including medical diagnosis, face recognition, and autonomous driving. Mathematically, a super-resolution model aims to recover a high-dimensional data $\mathbf{y} \in \mathbb{R}^{D\times D}$ from its low-resolution counterpart $\mathbf{x} \in \mathbb{R}^{d\times d}$, where the low-resolution data is typically downscaled by a factor of s, such that $d = D/s$. Consequently, super-resolution can be viewed as either an interpolation or extrapolation process, depending on the desired resolution.

2.1.1 Downsampling methods

Given the high-resolution data $\mathbf{y} \in \mathbb{R}^{D\times D}$, the corresponding low-resolution data $\mathbf{x} \in \mathbb{R}^{d\times d}$ (with a scaling factor $s = D/d$) is typically generated through a downsampling operation $\mathcal{D}_s$:

$$\mathbf{x} = \mathcal{D}_s(\mathbf{y}) + \epsilon, \tag{1}$$

where ϵ represents noise or other uncertainties introduced during the downsampling process. The downsampling operator $\mathcal{D}_s$ can take various forms, depending on how the information is reduced from high-resolution to low-resolution. Common downsampling techniques include pooling, interpolation, and sparsification.

- Pooling operates by selecting a summary statistic, such as the maximum or average, from a small region of the high-resolution data $\mathbf{y}$. This can be represented mathematically as:

  $$\mathbf{x}_{i,j} = \max_{(p,q)\in\mathcal{N}_{i,j}} \mathbf{y}_{p,q},$$

  where $\mathcal{N}_{i,j}$ is a local neighborhood around grid point (i, j) in the high-resolution data $\mathbf{y}$, and $\mathbf{x}_{i,j}$ is the corresponding low-resolution grid point value. Pooling reduces the spatial dimensions of the data while retaining key features.

- Interpolation generates lower-resolution data by averaging or selecting grid point values from the high-resolution data $\mathbf{y}$ based on a specified factor s. This can be represented mathematically as:

  $$\mathbf{x} = \mathcal{I}_s(\mathbf{y}),$$

  where $\mathcal{I}_s$ denotes the interpolation operator. Common interpolation methods include

nearest-neighbor, bilinear, and bicubic interpolation. This approach allows dimension reduction in the least cost of damaging the perceptual quality.

- Sparsification refers to selectively retaining a subset of the most important or most relevant components from the high-resolution data $\mathbf{y}$. This can be represented mathematically as:

$$\mathbf{x} = \mathcal{S}_s(\mathbf{y}),$$

where $\mathcal{S}_s$ denotes a sparsification operation, which reduces the number of retained coefficients. This approach is often used in compressed sensing where only the most informative components are kept.

2.1.2 The inverse problem view

Once the downsampling operator $\mathcal{D}_s$ is defined, the super-resolution problem can be formulated as learning its inverse $\mathcal{D}_s^{-1}$, which aims to reconstruct high-resolution data from corresponding low-resolution observations, as illustrated in Figure 1.

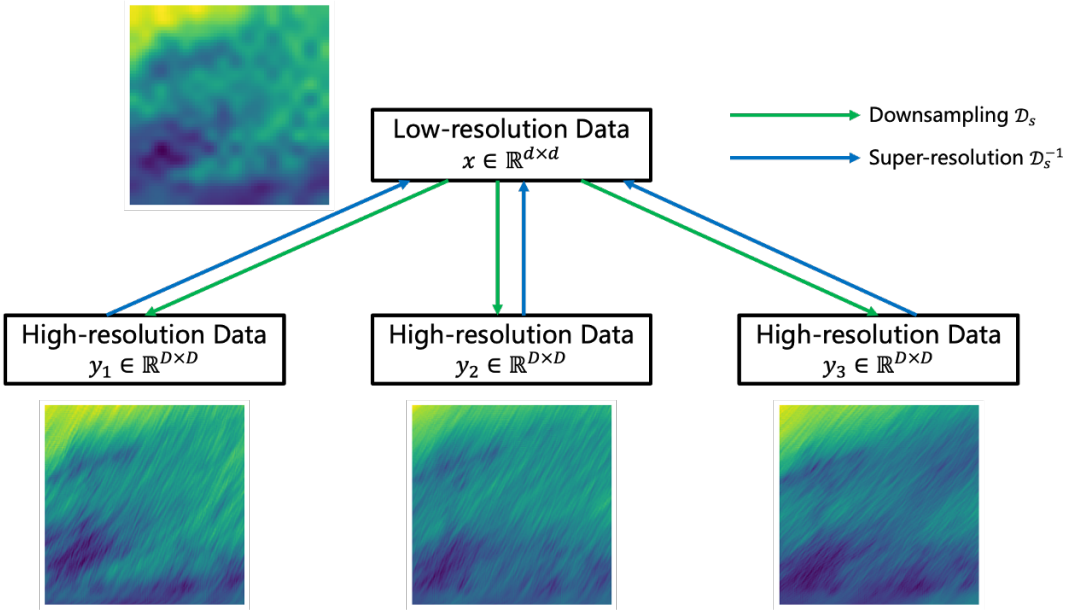


Figure 1: Super-resolution problem structure. From an inverse perspective, a single low-resolution data $\mathbf{x} \in \mathbb{R}^{d\times d}$ can correspond to multiple plausible high-resolution reconstructions $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3 \in \mathbb{R}^{D\times D}$, which highlights the ill-posed nature of the super-resolution problems.

However, from the perspective of inverse problems, super-resolution is inherently ill-posed. The key reason lies in the fact that the downsampling operation $\mathcal{D}_s$ is lossy: it discards fine-grained details such as high-frequency components. As a result, the inverse mapping $\mathcal{D}_s^{-1}$ is not unique. That is, multiple plausible high-resolution realizations can correspond to the same low-resolution input, especially when they share similar coarse structures. This one-to-many mapping nature introduces ambiguity and uncertainty into the reconstruction process, making the super-resolution task fundamentally challenging.

2.1.3 Super-resolution techniques

To address the ill-posed nature of the super-resolution problem, a variety of techniques have been developed to approximate the inverse of the downsampling operator $\mathcal{D}_s^{-1}$. Broadly, these approaches can be categorized into two groups: classical methods and data-driven methods.

(a) Classical methods

Classical approaches to super-resolution focus on reconstructing high-resolution image by leveraging predefined mathematical models or image structure priors. These methods do not require training on large datasets and often rely on statistical techniques or analytical assumptions about the nature of image data [55].

Among classical methods, interpolation techniques, such as bilinear and bicubic interpolation, form the most straightforward and widely-used family. These methods assume local smoothness or continuity in image intensity values and use this assumption to estimate unknown high-resolution pixels.

However, the relationship between pixels is often more intricate than what can be captured by simple interpolation rules, especially for dataset representing natural or physical processes. As a result, interpolation-based methods are limited in their ability to accurately recover fine-scale structures and high-frequency components lost during super-resolution. Nevertheless, their computational efficiency and ease of implementation make them a useful baseline approach or preprocessing step for validating and initializing more sophisticated super-resolution models, especially in scenarios where ground truth data is scarce or unavailable [55].

There are various interpolation methods applied in the literature, such as spline, nearest-neighbor and Lanczos interpolation. Two widely-used interpolation techniques in super-resolution tasks are:

- Bilinear interpolation estimates the value of the high-resolution grid point value $\mathbf{y}_{m,n}$ by computing a weighted average of the four nearest neighboring values in the low-resolution grid, denoted as (m_1, n_1), (m_1, n_2), (m_2, n_1), and (m_2, n_2). The interpolation can be formulated as:

$$\mathbf{y}_{m,n} \approx \sum_{i \in \{1,2\}} \sum_{j \in \{1,2\}} (m_i - m)(n_j - n)\mathbf{x}_{m_i,n_j}. \tag{2}$$

  Although bilinear interpolation is computationally efficient and simple to implement, it often introduces noticeable blurring, as it relies on a linear approximation of the underlying relationships and does not account for complex data structures.

- Bicubic interpolation extends this idea by considering a larger 4×4 neighborhood surrounding the target point (m, n) and applying cubic convolution along both spatial dimensions.

The interpolated value is computed as:

$$\mathbf{y}_{m,n} \approx \sum_{i=-1}^{2} \sum_{j=-1}^{2} P(m-i) \cdot P(n-j) \cdot I(i,j), \tag{3}$$

where $P(\cdot)$ is a cubic interpolation kernel (e.g., the Keys or Catmull-Rom kernel), and $I(i,j)$ denotes the grid point intensity in the low-resolution input. Bicubic interpolation generally produces smoother and sharper results than bilinear interpolation, as it leverages a larger spatial context and better captures intensity variations and edge information [55]. However, it is also more computationally demanding.

(b) Data-driven methods

With the emergence of machine learning, data-driven approaches have become the dominant paradigm in super-resolution. These methods aim to learn an approximate inverse mapping $\mathcal{D}_s^{-1}$ directly from data, by modeling the relationship between paired low-resolution inputs $\mathbf{x}$ and high-resolution targets $\mathbf{y}$.

Broadly speaking, there are two primary perspectives for modeling $\mathcal{D}_s^{-1}$ in data-driven settings:

- Discriminative models treat super-resolution as a supervised regression task, where the goal is to learn a deterministic mapping $f : \mathbf{x} \mapsto \mathbf{y}$. Such as CNNs [11].
- Generative models approach super-resolution from a probabilistic perspective, modeling the conditional distribution $p(\mathbf{y} \mid \mathbf{x})$. This formulation allows for capturing the inherent uncertainty and the one-to-many nature of the inverse problem. Such as GANs [31] and DMs [10].

2.1.4 Quality assessment

In super-resolution tasks, especially for image-based applications, evaluation metrics go beyond standard accuracy measures. Since the goal is to recover high-quality images from low-resolution inputs, both pixel-wise fidelity and perceptual similarity are essential. The following metrics are commonly used to assess reconstruction quality:

- Accuracy

  Mean Squared Error (MSE) is a pixel-wise measure of the average squared difference between the reconstructed image $\hat{\mathbf{y}}$ and the ground truth image $\mathbf{y}$ with $D \times D$ pixels. It is defined as:

$$\text{MSE} = \frac{1}{D^2} \sum_{i=1}^{D} \sum_{j=1}^{D} \left(\mathbf{y}_{i,j} - \hat{\mathbf{y}}_{i,j}\right)^2.$$

A lower MSE indicates higher reconstruction fidelity.

- Peak derivation ratio

  Peak signal-to-noise ratio (PSNR) is one of the most widely used objective metrics to quantify the quality of lossy image reconstruction. It is derived from the MSE and the maximum possible pixel value P:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{P^2}{\text{MSE}} \right).$$

  PSNR is expressed in decibels (dB), and a higher PSNR typically corresponds to better reconstruction quality.

- Structural similarity

  Unlike MSE and PSNR, which are pixel-wise error metrics, structural similarity index (SSIM) is designed to model perceptual similarity more closely to the human visual system. It considers structural information based on three components: luminance, contrast, and structure. The SSIM between two images $\mathbf{y}$ and $\hat{\mathbf{y}}$ is defined as:

$$\text{SSIM}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{(2\mu_{\mathbf{y}}\mu_{\hat{\mathbf{y}}} + C_1)(2\sigma_{\mathbf{y}\hat{\mathbf{y}}} + C_2)}{(\mu_{\mathbf{y}}^2 + \mu_{\hat{\mathbf{y}}}^2 + C_1)(\sigma_{\mathbf{y}}^2 + \sigma_{\hat{\mathbf{y}}}^2 + C_2)},$$

  where $\mu_{\mathbf{y}}$, $\mu_{\hat{\mathbf{y}}}$ are the mean intensities, $\sigma_{\mathbf{y}}$, $\sigma_{\hat{\mathbf{y}}}$ are standard deviations, and $\sigma_{\mathbf{y}\hat{\mathbf{y}}}$ is the covariance between the two images. C_1 and C_2 are small constants to stabilize the division. SSIM values range from -1 to 1, with 1 indicating perfect structural similarity.

2.2 Deep Neural Networks

Data-driven approaches are widely utilized in solving real-world problems. These methods aim to extract information, uncover patterns, and develop mathematical models to facilitate inference and prediction, providing theoretical support for various studies. This discipline is commonly referred to as statistical learning, or machine learning in the field of computer science.

The core idea of statistical learning is to quantitatively derive a mathematical function f that maps an input vector $\mathbf{x}$ to its corresponding output $\mathbf{y}$ based on a dataset $\mathcal{D}$ containing N input-output pairs. This dataset is represented as $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$. In this context, the function f represents the underlying relationship between the input $\mathbf{x}$ and the output $\mathbf{y}$, which can be learned from the data. Statistical methods are employed to extract information from the dataset, and the goal is to "learn" an estimated form of the mapping function f, denoted as $\hat{f}$.

Traditional models, such as linear regression and logistic regression, assume that f has a known form, such as a linear or polynomial function. Consequently, the estimated function $\hat{f}$ is modeled as a parametric function. For example, the parametric polynomial f

$$y = f(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_N x^N.$$

In these models, the primary objective is to estimate the coefficients $\mathbf{w} = (w_1, w_2, \cdots, w_N)$ based on the given dataset $\mathcal{D}$. However, as the complexity of the problem increases, it becomes apparent that not all relationships between input $\mathbf{x}$ and output $\mathbf{y}$ can be expressed in a predefined functional form. Therefore, more complex, nonlinear mappings are needed to estimate f, leading to the development of neural networks.

2.2.1 Neural networks structure

To generalize the linear model, one common approach is to express the output as a linear combination of nonlinear basis functions $h_i(\mathbf{x})$, instead of directly using the input features $\mathbf{x}$. This allows the model to capture more complex relationships between input and output. The model can be written as:

$$\hat{\mathbf{y}} = f(\mathbf{x}; \mathbf{w}) = H\left(\sum_{i=1}^{D} w_i h_i(\mathbf{x})\right), \tag{4}$$

where $h_i(\mathbf{x})$ denotes the i-th basis function, w_i is the corresponding weight, and $H(\cdot)$ is an optional activation function. For regression tasks, H is typically the identity function, while for classification tasks, it may represent a non-linear activation such as the sigmoid or softmax.

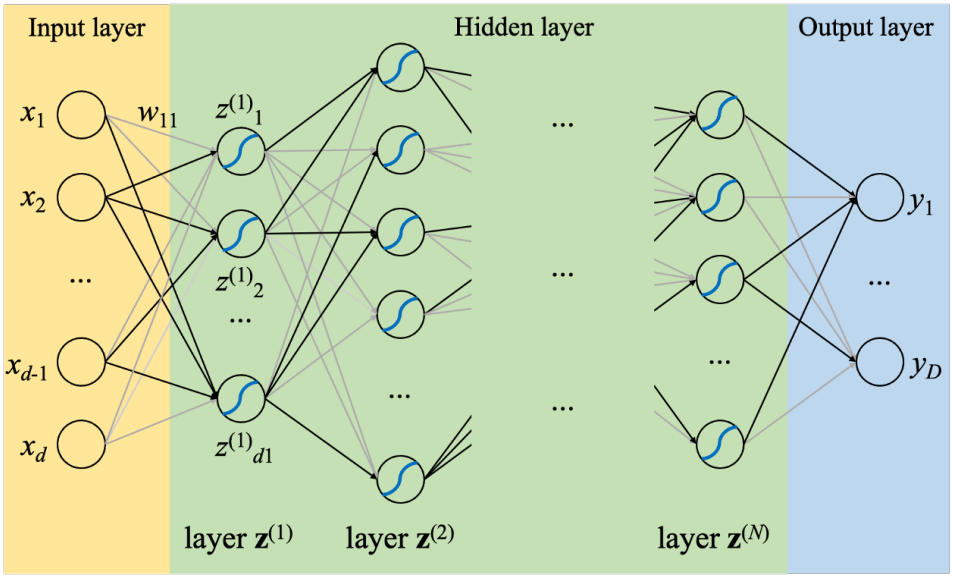


Figure 2: Neural Network structure. A neural network consists of an input layer (yellow block), several hidden layers (green block), and an output layer (blue block). The connections between neurons in different layers represent weights $\mathbf{W}$: black lines indicate non-zero weights, while gray lines indicate weights equal to zero. Each hidden neuron applies a nonlinear activation function $h(\cdot)$ (represented by the blue curve inside each node) to a weighted sum of its inputs.

Neural networks (NNs) extend the basis function formulation (4) into a layered and compos-

itional structure. As illustrated in Figure 2, the input is a vector $\mathbf{x} \in \mathbb{R}^d$ and the output is $\mathbf{y} \in \mathbb{R}^D$. The network is composed of an input layer, an output layer, and N intermediate layers known as hidden layers, with respective dimensions $d_1, d_2, \ldots, d_N$. Each hidden layer consists of multiple computational units, commonly referred to as neurons, which apply a weighted sum followed by a nonlinear activation function to their inputs. Denoting the activation of the k-th hidden layer as $\mathbf{z}^{(k)}$, the neural network can be formulated as:

$$
\begin{aligned}
z_j^{(1)} &= h\left(\sum_{i=1}^{d} w_{ij}^{(0)} x_i + b_j^{(0)}\right) = h_j\left(\mathbf{W}^{(0)}\mathbf{x} + \mathbf{b}^{(0)}\right), \\
z_j^{(2)} &= h\left(\sum_{i=1}^{d_1} w_{ij}^{(1)} z_i^{(1)} + b_j^{(1)}\right) = h_j\left(\mathbf{W}^{(1)}\mathbf{z}^{(1)} + \mathbf{b}^{(1)}\right), \\
&\cdots \\
z_j^{(N)} &= h\left(\sum_{i=1}^{d_{N-1}} w_{ij}^{(N-1)} z_i^{(N-1)} + b_j^{(N-1)}\right) = h_j\left(\mathbf{W}^{(N-1)}\mathbf{z}^{(N-1)} + \mathbf{b}^{(N-1)}\right), \\
y_i &= \sum_{i=1}^{d_N} w_{ij}^{(N)} z_i^{(N)} + b_j^{(N)} = \left(\mathbf{W}^{(N)}\mathbf{z}^{(N)} + \mathbf{b}^{(N)}\right)_j.
\end{aligned}
$$

Here, $h(\cdot)$ is a nonlinear activation function such as the sigmoid function $h(a) = \frac{1}{1+\exp(-a)}$ or the ReLU function $h(a) = \max(0, a)$, which introduces nonlinearity into the network.

Therefore, NNs learn from the data by optimizing a series of transformations through layers of interconnected neurons, capturing the mapping function f using connection weights between layers $\mathbf{W}^{(i)}$ and biases $\mathbf{b}^{(i)}$. This process can be expressed iteratively as:

$$
\begin{aligned}
\mathcal{NN}^h_{\mathbf{W}^{(i)},\mathbf{b}^{(i)}} &= \mathbf{b}^{(N)} + \mathbf{W}^{(N)} h\left(\mathbf{b}^{(N-1)} + \mathbf{W}^{(N-1)} h\left(\cdots h\left(\mathbf{b}^{(0)} + \mathbf{W}^{(0)}\mathbf{x}\right)\cdots\right)\right) \\
&= \mathbf{b}^{(N)} + \mathbf{W}^{(N)}\mathbf{z}^{(N)} \circ \mathbf{z}^{(N-1)} \circ \cdots \circ \mathbf{z}^{(1)} \circ \mathbf{x}.
\end{aligned}
$$

2.2.2 Autoencoder

In the network structure depicted in Figure 2, the input is represented as $\mathbf{x} \in \mathbb{R}^d$ and the output as $\mathbf{y} \in \mathbb{R}^D$, where typically $d > D$. However, when the number of output units matches the number of input units, the network is trained to generate an output $\mathbf{y}$ that closely approximates the corresponding input $\mathbf{x}$. This specific type of neural network was introduced by Hinton and Zemel [18] and is known as an Autoencoder (AE). The aim of AE is to reconstruct a output $\mathbf{y}$ that is close to input $\mathbf{x}$.

The structure of an AE is illustrated in Figure 3. The network consists of two primary compon-

ents: the encoder and the decoder. These can be viewed as two successive functional mappings, denoted as F_1 and F_2, respectively.

The encoder mapping F_1 projects the original d-dimensional input data $\mathbf{x}$ onto a lower M-dimensional subspace, known as the latent space, producing a latent representation $\mathbf{z}(\mathbf{x})$. The decoder mapping F_2 subsequently reconstructs the input data from the latent representation, yielding an output $\mathbf{y}(\mathbf{z})$ that approximates $\mathbf{x}$. An illustrative example of an AE in action is presented in Figure 4.

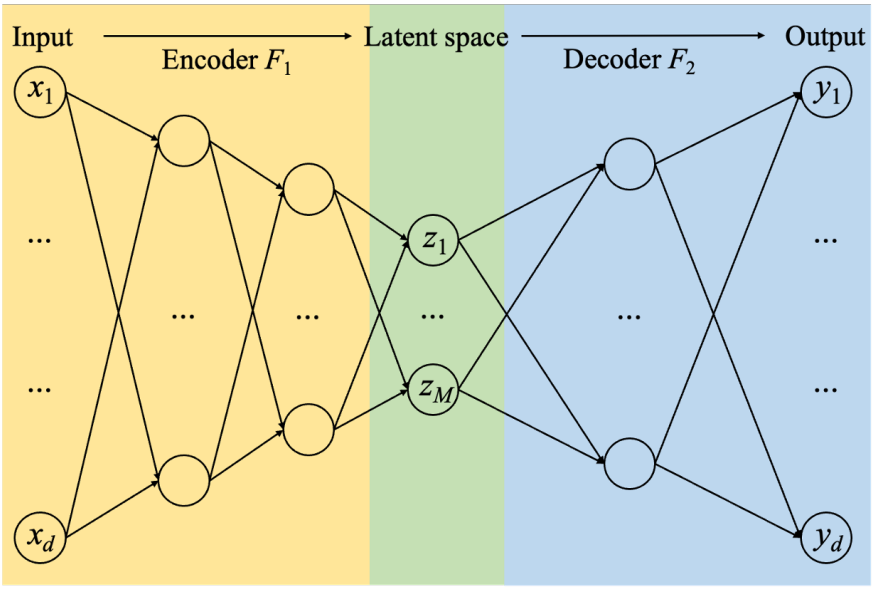


Figure 3: Autoencoder Structrue. An autoencoder consists of three main components: an encoder (yellow block), a latent space (green block), and a decoder (blue block). The encoder F_1 maps the input vector $\mathbf{x}$ from a high-dimensional space to a lower-dimensional latent space $\mathbf{z}$, while the decoder F_2 reconstructs the original input from the latent representation, producing an output $\mathbf{y}$ that has the same dimensionality as the input.

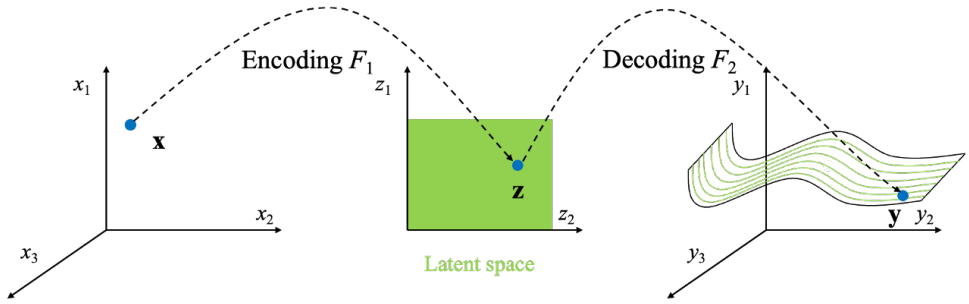


Figure 4: Autoencoder example. The encoder F_1 maps a d-dimensional input space ($d = 3$) into a M-dimensional latent space ($M = 2$), reducing the dimensionality of the data. The decoder F_2, which then reconstructs the original data from the latent representation, defines how the latent space is embedded in the d-dimensional input space. Since F_2 can be nonlinear, the resulting embedding of M-dimensional latent space can be non-planar.

A key extension of the AE is the Variational Autoencoders (VAEs) [29], which introduces a probabilistic interpretation of the latent space. In VAEs, the latent representation $\mathbf{z}$ is treated as a random variable with a prior distribution $p(\mathbf{z})$. Typically, this prior is chosen as an isotropic Gaussian distribution, i.e., $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, to enforce a structured latent space. Then this structure

serves as a fundamental block in the construction of rich generative models, for example, diffusion models explored in this thesis.

2.2.3 Training methods

Once the network architecture is defined, the next step is to train the model to approximate the underlying function f from data. Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, the neural network aims to learn a mapping $\hat{f}$ such that the predicted output $\hat{f}(\mathbf{x}_i)$ is as close as possible to the true target $\mathbf{y}_i$.

This is achieved by minimizing a predefined loss function $\mathcal{L}(\mathbf{w})$, which quantifies the discrepancy between the predictions $\hat{f}(\mathbf{x})$ and the ground truth $\mathbf{y}$. For example, in the case of an autoencoder, the loss function quantifies the discrepancy between the input $\mathbf{x}$ and its reconstructed counterpart $\mathbf{y}$ in MSE magnitude. It is commonly defined as:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n\|^2,$$

where $\mathbf{w}$ represents the parameters of the neural network to be optimized.

After defining the loss function, the training process seeks to estimate the optimal function $\hat{f}$ by minimizing the empirical loss over the training set $\mathcal{D}$. This is typically formulated as the minimization of a loss function $\mathcal{L}(\mathbf{w})$ with respect to the model parameters $\mathbf{w}$.

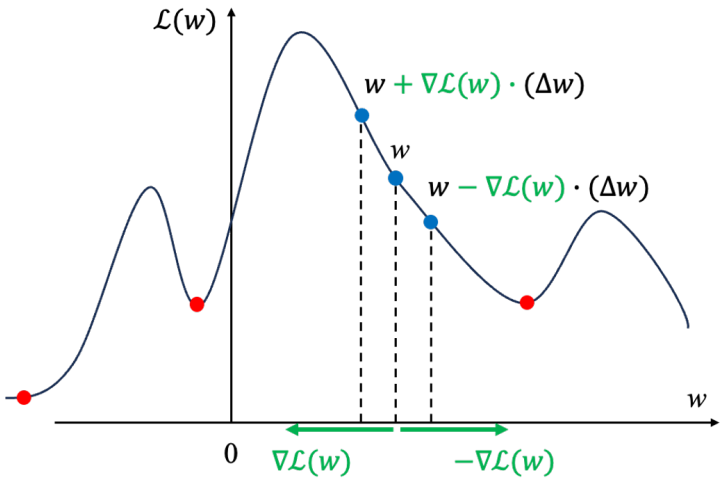


Figure 5: Gradient descent. The loss function $\mathcal{L}(\mathbf{w})$ landscape are complex with multiple local minima (red points). The goal of gradient descent is to minimize the loss function by iteratively updating the parameter $\mathbf{w}$ in the direction of the negative gradient, $-\nabla\mathcal{L}(\mathbf{w})$. Starting from a point $\mathbf{w}$, the algorithm takes a step $\Delta\mathbf{w}$ along $-\nabla\mathcal{L}(\mathbf{w})$ to reach the next iterate (blue points), descending the loss function.

In simple cases, where the model is linear and the loss function is convex, this optimization problem can sometimes be solved analytically. However, when more expressive models are used, such as deep neural networks that introduce multiple layers of nonlinearity, the resulting loss

function often becomes highly non-convex. This complexity arises from the structure of the model used to approximate f, rather than from f itself, which remains unknown in most real-world scenarios. As illustrated in Figure 5, the non-convex nature of the loss landscape can lead to numerous local minima or saddle points, making the optimization problem more challenging. To cope with this, iterative optimization methods such as gradient descent are widely used, although they do not guarantee convergence to the global minimum.

Intuitively, in the weight space, the loss function can be visualized as a surface with peaks and valleys, as shown in Figure 5. Minimizing the loss function is analogous to descending toward the lowest valley. Suppose the current weight vector is $\mathbf{w}$, representing a point in the weight space, and a small perturbation is applied to reach $\mathbf{w} + \Delta\mathbf{w}$. The corresponding change in the loss function can be approximated as

$$\Delta\mathcal{L} \approx (\Delta\mathbf{w})^T \nabla\mathcal{L}(\mathbf{w}),$$

where $\nabla\mathcal{L}(\mathbf{w})$ denotes the gradient of the loss function, which points in the direction of the steepest ascent. To minimize the loss, the weight update should proceed in the opposite direction of the gradient:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta\nabla\mathcal{L}(\mathbf{w}^{(k)}),$$

where $\eta > 0$ is the learning rate, which controls the step size of the update. By iteratively applying this update rule, the optimization algorithm gradually moves toward a local minimum, effectively reducing the loss function over time. However, as mentioned earlier, the complex structure of the loss function $\mathcal{L}(\mathbf{w})$ makes finding the global minimum particularly challenging. Due to the presence of multiple local minima, gradient descent may converge to a suboptimal solution, as the iterative updates can become trapped in a local minimum, unable to escape.

To mitigate this issue, LeCun et al. proposed an online version of gradient descent [30], which is based on a sum of independent observations. Instead of computing the loss over the entire training set, the dataset is divided into M smaller subsets, often called mini-batches. This allows the total loss function to be expressed as a sum over the losses computed on each mini-batch:

$$\mathcal{L}(\mathbf{w}) = \sum_{m=1}^{M} \mathcal{L}_m(\mathbf{w}),$$

where $\mathcal{L}_m(\mathbf{w})$ denotes the loss evaluated on the m-th mini-batch. For each step, updating the weight vector $\mathbf{w}$ using a portion of randomly selected data points at each iteration

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \nabla \mathcal{L}_m(\mathbf{w}^{(k)}).$$

This method is also known as stochastic gradient descent (SGD). Unlike standard gradient descent, which requires computing the gradient over the entire dataset in each iteration, SGD makes incremental updates by cycling through the dataset sequentially or by selecting individual data points randomly with replacement. Therefore, SGD introduces stochasticity into the optimization process, which can help the algorithm escape local minima and explore a broader region of the loss landscape. Furthermore, because each update step is based on only one data point, SGD significantly reduces the computational cost per iteration, making it more suitable for large-scale machine learning problems.

3 Super-resolution via Repeated Refinement

In this section, DMs are first introduced as a class of generative models. Due to their strong generative capabilities, they have recently attracted increasing attention in the context of weather upscaling, although their use in this domain is still in its early stages. However, their mathematical foundation is complex and nontrivial. To establish a rigorous basis for this thesis, a systematic derivation is provided, starting from the fundamental principles of DMs and leading to the formalization of their training and sampling algorithms.

Building on this foundation, Super-Resolution via Repeated Refinement (SR3) is then introduced as an extension of the standard DMs framework, designed to overcome the limitation that input and output must have the same dimensionality. This extension enables the application of DMs to super-resolution tasks, where high-resolution images are generated from low-resolution counterparts. Following a similar approach to the derivation of standard DMs, the fundamental concepts of SR3 are first presented, followed by a mathematical formulation of its training and sampling algorithms. These derivations establish the theoretical groundwork for the methods employed in this thesis.

3.1 Diffusion Models

DMs are a class of generative deep neural networks that have gained significant popularity in recent years. Unlike VAEs and GANs, which rely on a latent space $\mathbf{z}$ assumed to follow a fixed prior distribution, DMs adopt a fundamentally different generative mechanism. The core idea behind DMs is to progressively corrupt a given training sample $\mathbf{x}$ by adding Gaussian noise over multiple timesteps, ultimately transforming it into a sample drawn from a known prior distribution $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This process, illustrated as a left-to-right transformation in Figure 6, is referred to as the forward (noising) process.

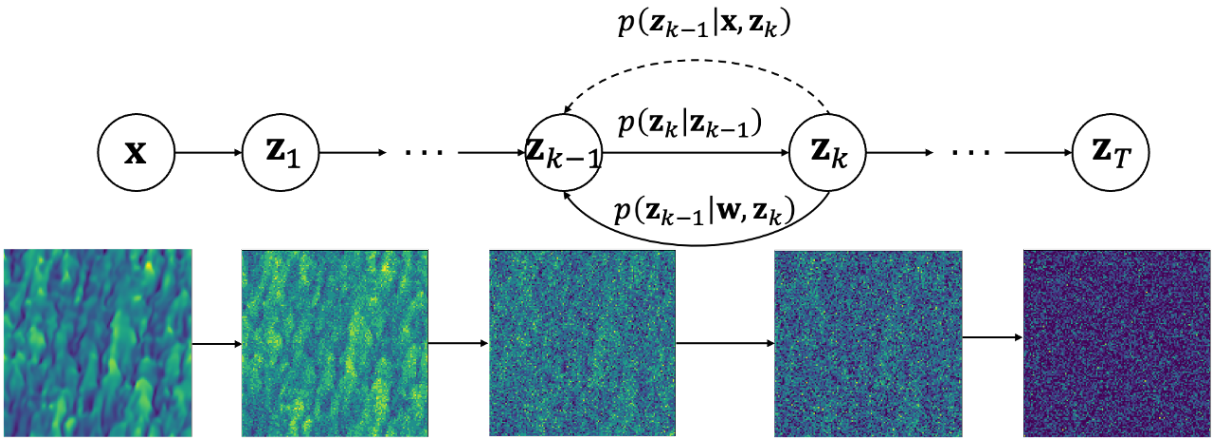


Figure 6: Framework of DMs. The top diagram illustrates the forward (rightward arrows) and reverse (leftward arrows) processes. The forward process follows the transition distribution $p(\mathbf{z}_k \mid \mathbf{z}_{k-1})$, while the reverse process ideally depends on $p(\mathbf{z}_k \mid \mathbf{x}, \mathbf{z}_{k-1})$ (dashed arrow). Since $\mathbf{x}$ is unknown during inference, it is approximated by a parameterized distribution $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})$ (backward arrow). The bottom sequence shows the diffusion process applied to turbulence field data, where Gaussian noise is progressively added to the original data $\mathbf{x}$, eventually yielding a noise sample $\mathbf{z}_T$ devoid of meaningful information.

Once the data has been mapped to this simple prior Gaussian distribution, a deep neural network is trained to approximate the reverse (denoising) process. The model iteratively removes Gaussian noise in a stepwise manner to reconstruct the original data sample $\mathbf{x}$, as depicted in the right-to-left transformation in Figure 6.

3.1.1 Forward process

In DMs, the forward process is designed to gradually introduce Gaussian noise into the data, transforming the original input into an isotropic Gaussian distribution over multiple time steps. Given an input $\mathbf{x} \in \mathbb{R}^{n \times n}$ (e.g., an image), each input unit (or pixel) undergoes an independent blending with Gaussian noise. The first-step noise-corrupted intermediate representation $\mathbf{z}_1$ is obtained by the following update rule:

$$\mathbf{z}_1 = \sqrt{1-\beta_1}\mathbf{x} + \sqrt{\beta_1}\epsilon_1,$$

where $\epsilon_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ represents Gaussian noise, and $\beta_1 \in (0, 1)$ controls the variance of the noise perturbation.

This process is repeated iteratively, where at each time step, additional independent Gaussian noise is added to generate a sequence of progressively noisier representations $\mathbf{z}_2, \mathbf{z}_3, \ldots, \mathbf{z}_T$:

$$\mathbf{z}_k = \sqrt{1-\beta_k}\mathbf{z}_{k-1} + \sqrt{\beta_k}\epsilon_k, \tag{5}$$

where $\epsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and mutually independent. It is worth noting that the noise variance schedule satisfies $0 < \beta_{k-1} < \beta_k < 1$, ensuring that the mean of $\mathbf{z}_k$ shifts closer to $\mathbf{0}$, while its variance approaches the identity matrix $\mathbf{I}$.

In other way round, this noising process can be interpreted as a Markov chain, where the transition probability is given by:

$$p(\mathbf{z}_k \mid \mathbf{z}_{k-1}) = \mathcal{N}\left(\sqrt{1-\beta_k}\mathbf{z}_{k-1}, \beta_k\mathbf{I}\right). \tag{6}$$

The Markov property implies that $\mathbf{z}_k$ is conditionally independent of $\mathbf{x}$ given $\mathbf{z}_{k-1}$, mathematically saying, $p(\mathbf{z}_k \mid \mathbf{z}_{k-1}, \mathbf{x}) = p(\mathbf{z}_k \mid \mathbf{z}_{k-1})$. This allows us to express the marginal distribution of $\mathbf{z}_k$ as an integral over the intermediate latent variables:

$$p(\mathbf{z}_k \mid \mathbf{x}) = \int \cdots \int p\left(\mathbf{z}_1, \ldots, \mathbf{z}_k \mid \mathbf{x}\right) d\mathbf{z}_1 \cdots d\mathbf{z}_{k-1}$$
$$= \int \cdots \int p\left(\mathbf{z}_1 \mid \mathbf{x}\right) \prod_{i=2}^{k} p\left(\mathbf{z}_i \mid \mathbf{z}_{i-1}\right) d\mathbf{z}_1 \cdots d\mathbf{z}_{k-1}.$$

For $p(\mathbf{z}_i \mid \mathbf{z}_{i-1})$, using (5). Then a closed-form expression for $\mathbf{z}_k$ can be derived by recursively applying the diffusion process from $\mathbf{z}_1$ to $\mathbf{z}_k$:

$$\mathbf{z}_k = \sqrt{(1-\beta_k)(1-\beta_{k-1})\cdots(1-\beta_1)}\mathbf{x} + \sum_{i=1}^{k} \sqrt{\beta_i} \left(\prod_{j=i+1}^{k} \sqrt{1-\beta_j} \right) \epsilon_i.$$

Defining $\alpha_k = \prod_{i=1}^{k}(1-\beta_i)$, due to the linearity and independence of the noise variables, $\mathbf{z}_k$ can be rewritten as:

$$\mathbf{z}_k = \sqrt{\alpha_k}\mathbf{x} + \sqrt{1-\alpha_k}\epsilon_k, \quad \epsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{7}$$

This expression confirms that $\mathbf{z}_k$ follows a Gaussian distribution:

$$p(\mathbf{z}_k \mid \mathbf{x}) = \mathcal{N}\left(\sqrt{\alpha_k}\mathbf{x}, (1-\alpha_k)\mathbf{I}\right).$$

Here, α_k encapsulates the cumulative effect of the noise schedule, progressively diminishing the influence of $\mathbf{x}$ while increasing the stochastic noise component. Notably, (7) highlights that the forward process can be efficiently simulated in a single step without iterating through all intermediate time steps. Instead, the total noise corruption at step k, represented by ϵ_k, can be directly sampled from a standard Gaussian distribution, significantly reducing computational complexity.

As k approaches a sufficiently large T ($T \to \infty$), the input data $\mathbf{x}$ is entirely destroyed, converging to an isotropic Gaussian distribution:

$$p(\mathbf{z}_T) = p(\mathbf{z}_T \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right).$$

This confirms that, given enough steps, all information in the input is lost, making the reverse process essential for reconstructing the original data distribution.

3.1.2 Reverse process

Once the forward process is completed, the output $\mathbf{z}_T$ serves as the input to the reverse process, as shown in Figure 6, which aims to reconstruct the original input $\mathbf{x}$. Similar to the forward process, the reverse process follows a Markov chain, where the transition probability is given by

$$\begin{aligned} p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k\right) &= \frac{p\left(\mathbf{z}_k \mid \mathbf{z}_{k-1}\right) p\left(\mathbf{z}_{k-1}\right)}{p\left(\mathbf{z}_k\right)} \\ &= \frac{p\left(\mathbf{z}_k \mid \mathbf{z}_{k-1}\right) \int p\left(\mathbf{z}_{k-1} \mid \mathbf{x}\right) p\left(\mathbf{x}\right) d\mathbf{x}}{p\left(\mathbf{z}_k\right)}. \end{aligned}$$

However, this posterior probability is intractable due to the integral over the unknown input data distribution $p(\mathbf{x})$. To address this, an alternative approach is to define the reversed transition probability conditioned on the original input, i.e., $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x})$. Under this formulation, it turns out to be a Gaussian distribution due to the conjugacy property:

$$p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x}\right) = \frac{p\left(\mathbf{z}_k \mid \mathbf{z}_{k-1}, \mathbf{x}\right) p\left(\mathbf{z}_{k-1} \mid \mathbf{x}\right)}{p\left(\mathbf{z}_k \mid \mathbf{x}\right)} = \mathcal{N}\left(\mathbf{m}_k\left(\mathbf{z}_k, \mathbf{x}\right), \sigma_k^2 \mathbf{I}\right), \tag{8}$$

where the mean and variance are given by [19]:

$$\begin{aligned} \mathbf{m}_k\left(\mathbf{z}_k, \mathbf{x}\right) &= \frac{\left(1-\alpha_{k-1}\right)\sqrt{1-\beta_k}\mathbf{z}_k + \sqrt{\alpha_{k-1}}\beta_k \mathbf{x}}{1-\alpha_k}, \\ \sigma_k^2 &= \frac{\beta_k\left(1-\alpha_{k-1}\right)}{1-\alpha_k}. \end{aligned}$$

This formulation is intuitive: given a noisy observation $\mathbf{z}_k$, it is inherently challenging to determine the exact lower-noise state $\mathbf{z}_{k-1}$ from which it originated. However, with additional knowledge of the original input $\mathbf{x}$, the reverse inference process becomes significantly more tractable. This motivates the modeling of the reverse transition probability $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k)$ as a Gaussian distribution.

Feller [12] demonstrated that for continuous diffusion processes, the forward and reverse transition probabilities exhibit the same functional form. When the noise variance $\beta_k \ll 1$, meaning that each step introduces only a small perturbation and a large number of steps are required to reach the prior distribution $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the discrete diffusion process closely approximates a continuous diffusion process. In this regime, the transition probability of the forward process $p(\mathbf{z}_k \mid \mathbf{z}_{k-1})$ and the reverse process $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k)$ share the same functional form.

Since the exact reverse transition probability $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x})$ is intractable due to the dependence on the unknown original data $\mathbf{x}$, it is approximated using a parameterized function

$p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})$, where $\mathbf{w}$ represents the learnable parameters of a deep neural network. Specifically, the reverse transition is modeled as a Gaussian distribution with fixed variance $\beta_k \mathbf{I}$, matching the forward noise scale:

$$p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w}) = \mathcal{N}(\mu(\mathbf{z}_k, \mathbf{w}, k), \beta_k \mathbf{I}), \tag{9}$$

where $\mu(\mathbf{z}_k, \mathbf{w}, k)$ is a neural network with parameters $\mathbf{w}$, trained to approximate the true reverse transition mean. This approximation is motivated by the small noise assumption, under which the local behavior of the diffusion process allows the reverse transition covariance to be closely approximated by the forward noise variance $\beta_k \mathbf{I}$ [12].

It should be noted that the form of the parameterized reverse probability is not unique. Nichol and Dhariwal [10] further refined this approach by incorporating the local curvature of $p(\mathbf{z}_{k-1})$ in the vicinity of $\mathbf{z}_k$ into the neural network. They proposed learning the covariance matrix $\Sigma(\mathbf{z}_k, \mathbf{w}, k)$ of $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})$, thereby introducing additional flexibility into the model architecture.

To sum up, the overall reverse process within the Markov chain framework is then expressed as:

$$p(\mathbf{z}_1, \ldots, \mathbf{z}_T, \mathbf{x} \mid \mathbf{w}) = p(\mathbf{z}_T) \left\{ \prod_{k=2}^{T} p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w}) \right\} p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{w}). \tag{10}$$

3.1.3 Training objective

Since the reverse process is modeled as a deep neural network parameterized by $\mathbf{w}$, training the reverse process involves finding the optimal set of parameters. The first step is to define an appropriate objective function for the optimization. A natural choice is the likelihood function conditioned on the input data $\mathbf{x}$:

$$l(\mathbf{w} \mid \mathbf{x}) = p(\mathbf{x} \mid \mathbf{w}) = \int \cdots \int p(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_T \mid \mathbf{w}) \, d\mathbf{z}_1 \cdots d\mathbf{z}_T$$

where $p(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_T \mid \mathbf{w})$ is defined by the reverse process described in (10). Intuitively, this likelihood function captures the probability of observing the data $\mathbf{x}$ by integrating over all possible latent trajectories $(\mathbf{z}_1, \ldots, \mathbf{z}_T)$ that could have generated the observed data. However, the integration space is high-dimensional and computationally intractable due to the complexity of the underlying probability distributions. This challenge motivates the search for an alternative training objective, which leads to the formulation of the Evidence Lower Bound (ELBO).

Similar to VAEs and GANs, the training of DMs relies on maximizing the ELBO [29], denoted by $\mathcal{L}(\mathbf{w})$, which serves as a tractable lower bound on the log-likelihood function:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \log p(\mathbf{x} \mid \mathbf{w}) - \mathrm{KL}(p(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x}, \mathbf{w})) \\ &= \log p(\mathbf{x} \mid \mathbf{w}) + \int p(\mathbf{z}) \log \left\{ \frac{p(\mathbf{z} \mid \mathbf{x}, \mathbf{w})}{p(\mathbf{z})} \right\} d\mathbf{z} \\ &= \int p(\mathbf{z}) \log \left\{ \frac{p(\mathbf{x}, \mathbf{z} \mid \mathbf{w})}{p(\mathbf{z})} \right\} d\mathbf{z}.\end{aligned}$$

In this formulation, the Kullback-Leibler (KL) divergence term $\mathrm{KL}(p(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x}, \mathbf{w}))$ measures the discrepancy between the prior distribution $p(\mathbf{z})$ and the posterior distribution $p(\mathbf{z} \mid \mathbf{x}, \mathbf{w})$. Since the KL divergence is always non-negative, we have the following inequality:

$$\log p(\mathbf{x} \mid \mathbf{w}) \geq \mathcal{L}(\mathbf{w}).$$

Thus, maximizing the ELBO $\mathcal{L}(\mathbf{w})$ indirectly maximizes the log-likelihood of the model. The selection of the prior distribution $p(\mathbf{z})$ plays a critical role in this optimization process. Although any valid probability distribution could theoretically serve this purpose, a carefully chosen $p(\mathbf{z})$ can simplify the expression for the ELBO, making it computationally tractable.

In VAEs, $p(\mathbf{z})$ is typically parameterized using a neural network with trainable parameters [29]. Since $p(\mathbf{x}, \mathbf{z} \mid \mathbf{w})$ is also parameterized by a neural network governed by $\mathbf{w}$, optimizing $p(\mathbf{z})$ brings the overall optimization of the model parameters $\mathbf{w}$ closer to the objective of maximizing the likelihood function $p(\mathbf{x} \mid \mathbf{w})$. This connection forms the foundation for efficient training of DMs and their ability to generate high-quality outputs.

In DMs, $p(\mathbf{z})$ is chosen by a fixed distribution $p(\mathbf{z}_1, \ldots, \mathbf{z}_T \mid \mathbf{x})$, which depends on input data $\mathbf{x}$. According to the Markov property of the forward process, the joint distribution is given by:

$$p(\mathbf{z}_1, \ldots, \mathbf{z}_T \mid \mathbf{x}) = p(\mathbf{z}_1 \mid \mathbf{x}) \prod_{k=2}^{T} p(\mathbf{z}_k \mid \mathbf{z}_{k-1}),$$

where $p(\mathbf{z}_1 \mid \mathbf{x})$ and $p(\mathbf{z}_k \mid \mathbf{z}_{k-1})$ is given in (6). Plugging it back to ELBO inllustrated above and the only adjustable parameters are $\mathbf{w}$ of $p(\mathbf{x}, \mathbf{z} \mid \mathbf{w})$ in the reverse process. Then, the ELBO can be derived as:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \int \cdots \int p(\mathbf{z}_1, \ldots, \mathbf{z}_T \mid \mathbf{x}) \ln \left\{ \frac{p(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_T \mid \mathbf{w})}{p(\mathbf{z}_1, \ldots, \mathbf{z}_T \mid \mathbf{x})} \right\} d\mathbf{z}_1 \cdots d\mathbf{z}_T \\ &= \int \cdots \int \left\{ p(\mathbf{z}_1 \mid \mathbf{x}) \prod_{k=2}^{T} p(\mathbf{z}_k \mid \mathbf{z}_{k-1}) \right\} \ln \left\{ \frac{p(\mathbf{z}_T) \left\{ \prod_{k=2}^{T} p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w}) \right\} p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{w})}{p(\mathbf{z}_1 \mid \mathbf{x}) \prod_{k=2}^{T} p(\mathbf{z}_k \mid \mathbf{z}_{k-1})} \right\} d\mathbf{z}_1 \cdots d\mathbf{z}_T.\end{aligned}$$

For simplicity, define:

$$\begin{aligned}\mathbb{E}_{\mathbf{z}_1:\mathbf{z}_T}\left[f(\mathbf{z})\right] &= \int\cdots\int p\left(\mathbf{z}_1,\ldots,\mathbf{z}_T \mid \mathbf{x}\right)\left[f(\mathbf{z})\right]d\mathbf{z}_1\cdots d\mathbf{z}_T \\ &= \int\cdots\int\left\{p\left(\mathbf{z}_1 \mid \mathbf{x}\right)\prod_{k=2}^{T}p\left(\mathbf{z}_k \mid \mathbf{z}_{k-1}\right)\right\}\left[f(\mathbf{z})\right]d\mathbf{z}_1\cdots d\mathbf{z}_T.\end{aligned}$$

Then ELBO can be expressed as

$$\begin{aligned}\mathcal{L}\left(\mathbf{w}\right) &= \mathbb{E}_{\mathbf{z}_1:\mathbf{z}_T}\left[\ln\frac{p\left(\mathbf{z}_T\right)\left\{\prod_{k=2}^{T}p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k,\mathbf{w}\right)\right\}p\left(\mathbf{x} \mid \mathbf{z}_1,\mathbf{w}\right)}{p\left(\mathbf{z}_1 \mid \mathbf{x}\right)\prod_{k=2}^{T}p\left(\mathbf{z}_k \mid \mathbf{z}_{k-1}\right)}\right] \\ &= \mathbb{E}_{\mathbf{z}_1:\mathbf{z}_T}\left[\ln p\left(\mathbf{z}_T\right)+\ln p\left(\mathbf{x} \mid \mathbf{z}_1,\mathbf{w}\right)-\ln p\left(\mathbf{z}_1 \mid \mathbf{x}\right)+\sum_{k=2}^{T}\ln\frac{p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k,\mathbf{w}\right)}{p\left(\mathbf{z}_k \mid \mathbf{z}_{k-1}\right)}\right] \\ &= \mathbb{E}_{\mathbf{z}_1:\mathbf{z}_T}\left[\ln p\left(\mathbf{z}_T\right)+\ln\frac{p\left(\mathbf{x} \mid \mathbf{z}_1,\mathbf{w}\right)}{p\left(\mathbf{z}_1 \mid \mathbf{x}\right)}+\sum_{k=2}^{T}\ln\frac{p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k,\mathbf{w}\right)p\left(\mathbf{z}_{k-1} \mid \mathbf{x}\right)}{p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k,\mathbf{x}\right)p\left(\mathbf{z}_k \mid \mathbf{x}\right)}\right] \\ &= \mathbb{E}_{\mathbf{z}_1:\mathbf{z}_T}\left[\ln p\left(\mathbf{z}_T\right)+\ln\frac{p\left(\mathbf{x} \mid \mathbf{z}_1,\mathbf{w}\right)}{p\left(\mathbf{z}_1 \mid \mathbf{x}\right)}+\ln\frac{p\left(\mathbf{z}_1 \mid \mathbf{x}\right)}{p\left(\mathbf{z}_T \mid \mathbf{x}\right)}+\sum_{k=2}^{T}\ln\frac{p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k,\mathbf{w}\right)}{p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k,\mathbf{x}\right)}\right] \\ &= \mathbb{E}_{\mathbf{z}_1:\mathbf{z}_T}\left[\ln p\left(\mathbf{x} \mid \mathbf{z}_1,\mathbf{w}\right)+\ln\frac{p\left(\mathbf{z}_T\right)}{p\left(\mathbf{z}_T \mid \mathbf{x}\right)}+\sum_{k=2}^{T}\ln\frac{p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k,\mathbf{w}\right)}{p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k,\mathbf{x}\right)}\right],\end{aligned}$$

where the third equation holds by Bayesian formula and Markov property

$$p\left(\mathbf{z}_k \mid \mathbf{z}_{k-1}\right)=p\left(\mathbf{z}_k \mid \mathbf{z}_{k-1},\mathbf{x}\right)=\frac{p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k,\mathbf{x}\right)p\left(\mathbf{z}_k \mid \mathbf{x}\right)}{p\left(\mathbf{z}_{k-1} \mid \mathbf{x}\right)}.$$

According to the forward process of DMs, the corrupted input will eventually converge to an isotropic Gaussian distribution, i.e., $\mathbf{z}_T \sim \mathcal{N}\left(\mathbf{0},\mathbf{I}\right)$, which implies that all information from the original input has been completely lost at the final timestep T. Consequently, no trainable parameters are associated with $p\left(\mathbf{z}_T\right)$, allowing to omit this term from ELBO optimization. Similarly, since $p\left(\mathbf{z}_T \mid \mathbf{x}\right)$ is a fixed distribution without learnable parameters, it can also be excluded from the ELBO objective.

Thus, the ELBO can be reformulated using the definition of the KL divergence as follows:

$$
\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \mathbb{E}_{\mathbf{z}_1:\mathbf{z}_T}\left[\ln p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{w}) + \sum_{k=2}^{T} \ln \frac{p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})}{p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x})}\right] \\
&= \int \cdots \int p(\mathbf{z}_1, \ldots, \mathbf{z}_T \mid \mathbf{x}) \left[\ln p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{w}) + \sum_{k=2}^{T} \ln \frac{p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})}{p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x})}\right] d\mathbf{z}_1 \cdots d\mathbf{z}_T \\
&= \underbrace{\int p(\mathbf{z}_1 \mid \mathbf{x}) \ln p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{w})\, d\mathbf{z}_1}_{\text{reconstruction term}} - \underbrace{\sum_{k=2}^{T} \int p(\mathbf{z}_k \mid \mathbf{x}) \, \mathrm{KL}\left(p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x}) \| p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})\right) d\mathbf{z}_k}_{\text{consistency term}}.
\end{aligned}
$$

The first term is commonly referred to as the reconstruction term, which measures the likelihood of generating the observed data $\mathbf{x}$ from the first latent output $\mathbf{z}_1$. Maximizing this likelihood contributes positively to the ELBO objective. In contrast, the second term, known as the consistency term, enforces agreement between the true reverse $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x})$, which conditions on the original data $\mathbf{x}$, and the learned reverse process $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})$, which only depends on the former latent state $\mathbf{z}_k$. Minimizing the KL divergence between these two ensures that the learned reverse model faithfully approximates the true reverse dynamics.

Since both he true reverse $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x})$ and the learned reverse process $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})$ are Gaussian distributions as defined in (8) and (9), the KL divergence between them can be computed in closed form based on the analytical properties of Gaussian distributions, which is given as

$$
\mathrm{KL}\left(p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x}) \| p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})\right) = \frac{1}{2\beta_k} \|\mathbf{m}_k(\mathbf{z}_k, \mathbf{x}) - \mu(\mathbf{z}_k, \mathbf{w}, k)\|^2 + C, \tag{11}
$$

where β_k denotes the variance of the Gaussian noise at timestep k, $\mathbf{m}_k(\mathbf{z}_k, \mathbf{x})$ represents the true mean function conditioned on $\mathbf{x}$, and $\mu(\mathbf{z}_k, \mathbf{w}, k)$ corresponds to the parameterized mean predicted by the model. The constant C is independent of $\mathbf{w}$ and can thus be ignored during optimization.

Given the KL divergence in (11), the consistency term reduces to a simple squared-loss function. Consequently, maximizing the ELBO becomes equivalent to simultaneously maximizing the reconstruction likelihood and minimizing the squared-error terms across all consistency steps, thereby encouraging the learned distribution $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})$ to approximate the true distribution $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x})$. It should be noted that the computational cost of optimizing $\mathcal{L}(\mathbf{w})$ is primarily dominated by the consistency term, as it requires optimization over all timesteps k.

3.1.4 Modified training obejective

As discussed in the previous section, the ELBO consists of the KL divergences between the true forward noising process and the learned reverse denoising process at each step. Maximizing the

ELBO is equivalent to forcing the learned distribution $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})$ to approximate the true distribution $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x})$. However, as described in (7), the latent output $\mathbf{z}_k$ at any timestep k can be directly obtained by computing the total noise corruption applied to $\mathbf{x}$ up to timestep k, rather than iteratively adding noise at each step.

Ho et al. [19] proposed a modified ELBO formulation that yields higher quality results by altering the objective of neural network. Instead of predicting the denoised output at each Markov chain step, a new deep neural network $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)$ is used to predict the total noise component added to the original input $\mathbf{x}$ to generate the latent noisy output $\mathbf{z}_k$ at timestep k. This modification improves sample quality because predicting the noise is generally easier than predicting the clean data directly, leading to a more stable training process and better convergence properties [19].

Following the total noising approach in (7), the mean $\mathbf{m}_k(\mathbf{z}_k, \mathbf{x})$ of the reverse conditional distribution $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x})$ can be expressed as:

$$\mathbf{m}_k(\mathbf{z}_k, \mathbf{x}) = \frac{1}{\sqrt{1-\beta_k}}\left(\mathbf{z}_k - \frac{\beta_k}{\sqrt{1-\alpha_k}}\epsilon_k\right),$$

where ϵ_k represents the total noise added to $\mathbf{x}$ to generate $\mathbf{z}_k$. This formulation highlights that the reverse process is governed by the noise structure introduced during the forward diffusion process.

To estimate this noise component, a deep neural network $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)$, parameterized by $\mathbf{w}$, is introduced to predict the total noise ϵ_k. Consequently, the mean $\mu(\mathbf{z}_k, \mathbf{w}, k)$ of the reverse conditional distribution $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})$ can be rewritten in terms of the predicted noise as:

$$\mu(\mathbf{z}_k, \mathbf{w}, k) = \frac{1}{\sqrt{1-\beta_k}}\left(\mathbf{z}_k - \frac{\beta_k}{\sqrt{1-\alpha_k}}\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)\right). \tag{12}$$

Substituting into Equation 11, the KL divergence can be rewritten in terms of the original input $\mathbf{x}$ and total noise ϵ_k as:

$$\mathrm{KL}\left(p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x}) \| p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})\right) = \frac{\beta_k}{2(1-\alpha_k)(1-\beta_k)}\|\mathbf{g}(\underbrace{\sqrt{\alpha_k}\mathbf{x} + \sqrt{1-\alpha_k}\epsilon_k}_{\mathbf{z}_k}, \mathbf{w}, k) - \epsilon_k\|^2 + C.$$

Regarding the reconstruction term, it can be written similarly to the consistency term, with $k = 1$ being a special case:

$$\ln p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{w}) = \frac{-1}{2(1-\beta_1)}\|\mathbf{g}(\underbrace{\sqrt{\alpha_1}\mathbf{x} + \sqrt{1-\alpha_1}\epsilon_1}_{\mathbf{z}_1}, \mathbf{w}, 1) - \epsilon_1\|^2 + C.$$

Thus, the ELBO becomes:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= -\sum_{k=1}^{T} \frac{\beta_k}{2(1-\alpha_k)(1-\beta_k)} \int p(\mathbf{z}_k \mid \mathbf{x}) \left\|\mathbf{g}\left(\sqrt{\alpha_k}\mathbf{x} + \sqrt{1-\alpha_k}\epsilon_k, \mathbf{w}, k\right) - \epsilon_k\right\|^2 d\mathbf{z}_k \\ &= -\mathbb{E}_{\epsilon_k} \sum_{k=1}^{T} \frac{\beta_k}{2(1-\alpha_k)(1-\beta_k)} \left\|\mathbf{g}\left(\sqrt{\alpha_k}\mathbf{x} + \sqrt{1-\alpha_k}\epsilon_k, \mathbf{w}, k\right) - \epsilon_k\right\|^2,\end{aligned}$$

where $\mathbf{z}_k$ is constructed as a noisy version of $\mathbf{x}$ by adding Gaussian noise $\epsilon_k \sim \mathcal{N}(0, I)$. Consequently, the integral over $p(\mathbf{z}_k \mid \mathbf{x})$ can be interpreted as an expectation over ϵ_k. In practice, this expectation is typically approximated by a single Monte Carlo sample of ϵ_k at each iteration.

Furthermore, $\mathbf{x}$ is sampled from the empirical data distribution $p_{\mathcal{D}}(\mathbf{x})$, and the timestep k is uniformly sampled from $\{1, \ldots, T\}$. Combining these sampling strategies, the ELBO can be interpreted as the weighted sum of the squared differences between the predicted noise and the actual noise across T timesteps:

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathbf{x}}\mathbb{E}_{\epsilon_k,k}\|\mathbf{g}(\sqrt{\alpha_k}\mathbf{x} + \sqrt{1-\alpha_k}\epsilon_k, \mathbf{w}, k) - \epsilon_k\|^2. \tag{13}$$

It is worth noting that Ho et al. [19] empirically observed that diffusion models achieve better performance when the weighting factor $\frac{\beta_k}{2(1-\alpha_k)(1-\beta_k)}$ is omitted. This adjustment ensures that each timestep contributes equally to the loss, rather than being weighted according to its associated noise level. As a result, the final training objective commonly used for diffusion models removes the original weighting term from the ELBO.

3.1.5 Denoising diffusion probabilistic models

Focusing on the training objective of DMs presented in (13), the squared error term has a straightforward interpretation. For a given step k in the Markov chain and a given training data point $\mathbf{x}$, a total noise vector ϵ_k is sampled to generate the corresponding noisy latent state $\mathbf{z}_k$ at that step. The loss function measures the squared difference between the predicted noise, denoted as $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)$, and the actual sampled noise ϵ_k. The overall training process of DMs, known as Denoising Diffusion Probabilistic Models (DDPMs) [19], is illustrated in Figure 7.

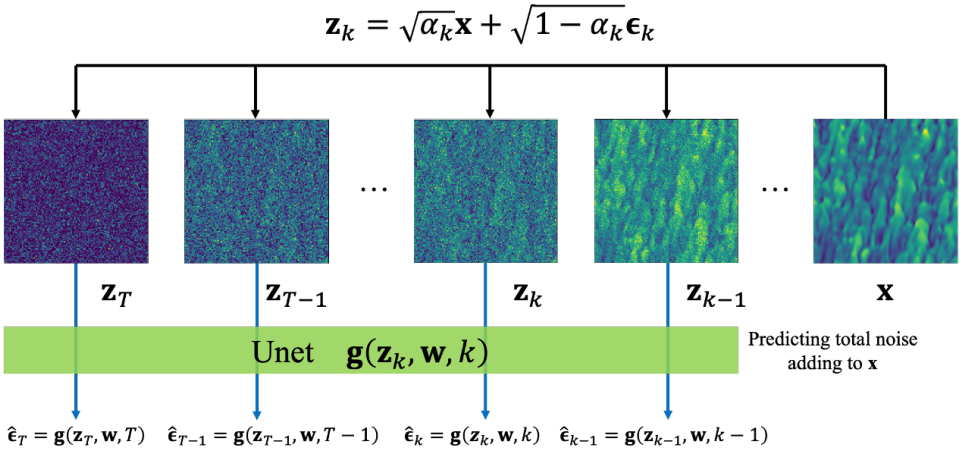


Figure 7: DDPMs training process. At each timestep k, a noisy latent variable $\mathbf{z}_k$ is generated from the original data $\mathbf{x}$ as $\mathbf{z}_k = \sqrt{\alpha_k}x + \sqrt{1-\alpha_k}\epsilon_k$, where ϵ_k represents the added total noise with respect to original input $\mathbf{x}$. This process is illustrated by the black line. The neural network $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)$ (green block) is trained to predict ϵ_k, depicted by the blue line in the figure.

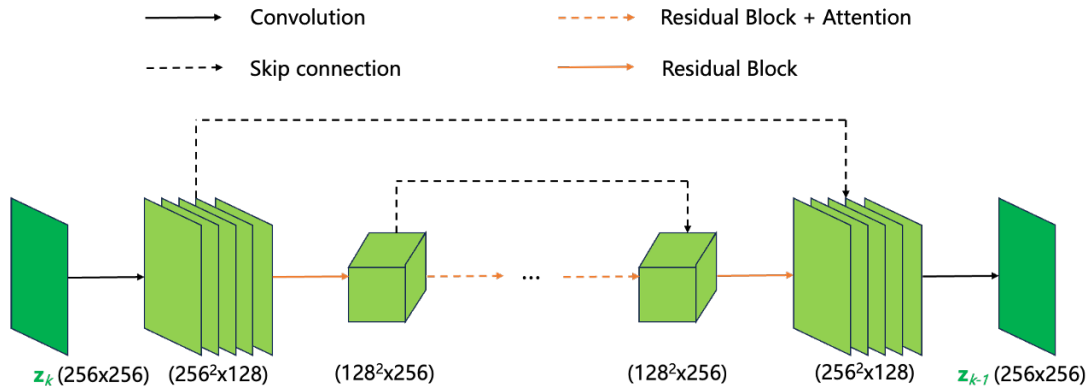


Figure 8: An example of U-Net architecture of DMs. The U-Net consists of downsampling layers, corresponding upsampling layers, and a bottleneck, with skip connection between symmetric layers to preserve spatial information. This architecture is used to predict the total added noise ϵ_k from a noisy latent state $\mathbf{z}_k$. The model takes the input and outputs in the same size (256×256).

To predict the total noise added to the latent state $\mathbf{z}_k$, DMs often adopt the U-Net architecture, given in Figure 8, which has demonstrated excellent performance in maintaining spatial structures during noise estimation. A geeral U-Net consists of downsampling blocks, upsampling blocks, and skip connections that transfer spatial information between corresponding layers, ensuring the preservation of fine-grained details necessary for high-fidelity generation tasks [19].

The overall training process of DDPMs is perturbing the original clean data $\mathbf{x}$ by adding noise to form $\mathbf{z}_k$ at each timestep k followed (7). Then the neural network $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)$ is then trained to predict ϵ_k.

Algorithm 1 outlines the procedure for training a DDPMs using SGD. In each iteration, the gradient vectors of the loss function are computed by randomly sampling data points from the dataset $\mathcal{D}$. Each sampled data point $\mathbf{x}$ is subjected to a diffusion process by adding noise at a randomly selected timestep t along the Markov chain. Subsequently, an optimization step

is performed to accumulate the gradients over a mini-batch, followed by updating the model parameters.

Algorithm 1 Denoising diffusion probabilistic models (DDPMs) training [5]

Input: Training data $\mathcal{D} = \{\mathbf{x}_n\}$, Noise schedule $\{\beta_1, \ldots, \beta_T\}$
Output: Neural network parameters $\mathbf{w}$

for $k \in \{1, \ldots, T\}$ **do**
  $\alpha_k \leftarrow \prod_{i=1}^{k}(1 - \beta_i)$ ▷ Calculate α from β
end for
repeat
  $\mathbf{x} \sim \mathcal{D}$ ▷ Sample a data point
  $k \sim \{1, \ldots, T\}$ ▷ Sample a point along the Markov chain
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Sample a noise vector
  $\mathbf{z}_k \leftarrow \sqrt{\alpha_k}\mathbf{x} + \sqrt{1 - \alpha_k}\,\epsilon$ ▷ Evaluate noisy latent variable
  $\mathcal{L}(\mathbf{w}) \leftarrow \|\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k) - \epsilon\|^2$ ▷ Compute loss term
  Take optimizer step
until converged
return $\mathbf{w}$

Once the model is trained, the generation of new samples is performed via a sequential denoising process, starting from $\mathbf{z}_T$, which is a pure Gaussian noise $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Notably, the sampling procedure differs from training: instead of predicting the total noise at arbitrary timesteps, sampling progresses step-by-step alongside the Markov chain, gradually refining the latent states $\mathbf{z}_k$ towards the target outputs.

The sampling procedure is detailed in Algorithm 2 and illustrated in Figure 9. At each denoising step, the network predicts the noise component $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)$, which is used to compute the mean $\mu(\mathbf{z}_k, \mathbf{w}, k)$ of the reverse diffusion distribution followed (12).

Each denoising step from $\mathbf{z}_k$ to $\mathbf{z}_{k-1}$ involves three sub-steps:

1. Denoising estimation: compute $\mu(\mathbf{z}_k, \mathbf{w}, k)$ using the noise prediction network $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)$.
2. Sampling: generate a sample $\mathbf{z}_{k-1}$ from the distribution $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w})$, leveraging the Gaussian distribution properties.
3. Noise adjustment: add the scaled noise term $\sqrt{\beta_k}\epsilon$, where ϵ represents the stochastic noise introduced at timestep $k - 1$ during the forward process.

In the final step, a noise-free output $\mathbf{x}$ is obtained without adding any additional noise.

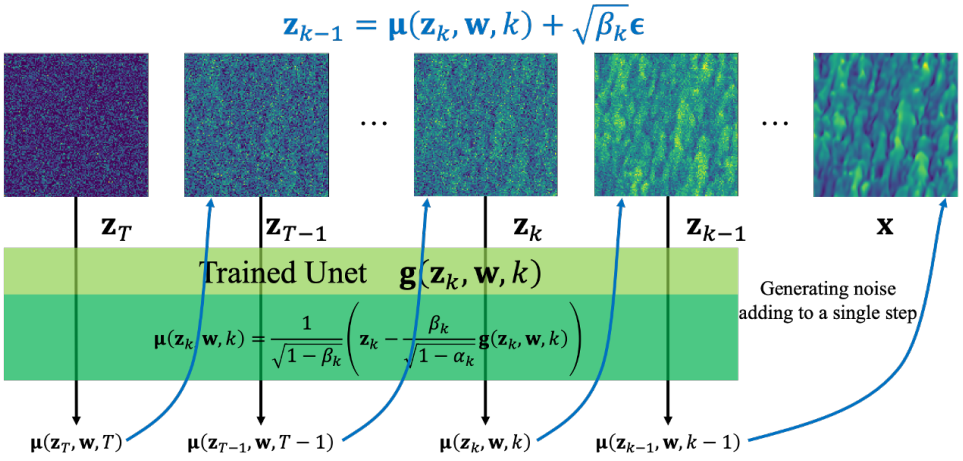


Figure 9: DDPMs sampling process. Starting from pure noise $\mathbf{z}_T$, the latent variable $\mathbf{z}_k$ is iteratively denoised using the trained neural network $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)$ (green block), which predicts the noise (black line). The estimated mean $\mu(\mathbf{z}_k, \mathbf{w}, k)$ is used to compute a less noisy latent state through $\mathbf{z}_{k-1} = \mu(\mathbf{z}_k, \mathbf{w}, k) + \sqrt{\beta_k}\epsilon$ (blue line). Through this iterative denoising process, the model gradually reconstructs the original input $\mathbf{x}$.

Algorithm 2 Denoising diffusion probabilistic models (DDPMs) sampling [5]

Input: Trained denoising network $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)$, Noise schedule $\{\beta_1, \ldots, \beta_T\}$
Output: Sample vector $\mathbf{x}$ in data space

$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Sample from final latent space
for $k = T, T-1, \ldots, 2$ **do**
  $\alpha_k \leftarrow \prod_{i=1}^{k}(1-\beta_i)$ ▷ Calculate α from β
  $\mu\left(\mathbf{z}_k, \mathbf{w}, t\right) \leftarrow \frac{1}{\sqrt{1-\beta_k}}\left(\mathbf{z}_k - \frac{\beta_k}{\sqrt{1-\alpha_k}}\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k)\right)$
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Sample a noise vector
  $\mathbf{z}_{k-1} \leftarrow \mu(\mathbf{z}_k, \mathbf{w}, k) + \sqrt{\beta_k}\epsilon$ ▷ Add scaled noise
end for
$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}}\left(\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}}\mathbf{g}(\mathbf{z}_1, \mathbf{w}, 1)\right)$ ▷ Final denoising step
return $\mathbf{x}$

Recent works have proposed improvements to the original DDPM framework to address the computational inefficiency arising from the sequential sampling steps. For instance, Song et al. [49] reformulated the reverse diffusion process as an ordinary differential equation, enabling deterministic sampling with significantly fewer steps. Additionally, methods like stochastic differential equations allowed trade-offs between computation and sample quality by tuning the discretization scheme [50]. Recent approaches, such as consistency models, further improved efficiency by training models to directly map noise to data in a single step [48].

3.2 Super-Resolution via Repeated Refinement

DMs require the input $\mathbf{x}$ and output $\mathbf{z}_T$ to have the same dimensionality. This is because, at each timestep, noise is added with varying intensities while maintaining the same spatial distribution. In image processing tasks, every pixel in the reverse process must be denoised in a one-to-one correspondence with the input pixels. If the dimensions of the input and output differ, the structural integrity of the noise distribution is disrupted, preventing DMs from learning an

effective mapping from noise to a clear image.

To address this limitation, Saharia et al. [43] proposed the Super-Resolution via Repeated Refinement (SR3) framework, which extends the capabilities of DMs for modeling conditional probabilities. Similar to classical DDPMs, SR3 performs the forward process by progressively adding Gaussian noise to samples from an empirical distribution, such as the conditional distribution $p(\mathbf{y} \mid \mathbf{x})$, through T timesteps until all information is lost, resulting in a standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The reverse process then aims to recover the original distribution by progressively denoising within a sequence of refinement steps.

It is important to note that, in standard DMs, the reverse process aims to reconstruct an output $\mathbf{z}_0$ that is as close as possible to the original input $\mathbf{x}$. However, in SR3, the objective shifts: the reverse process is conditioned on the low-resolution input $\mathbf{x}$ and is designed to generate a high-resolution output $\mathbf{z}_0$ that closely matches the target $\mathbf{y}$. Here, the SR3 model is trained on paired data $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ serves as the low-resolution input, and the goal is to predict $\mathbf{y}$, the corresponding high-resolution target.

3.2.1 Conditional forward and reverse process

Given a dataset of N input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, where each low-resolution input $\mathbf{x}_i \in \mathbb{R}^{d \times d}$ corresponds to a high-resolution output $\mathbf{y}_i \in \mathbb{R}^{D \times D}$, the goal of SR3 is to learn a mapping from $\mathbf{x}_i$ to $\mathbf{y}_i$ that accurately captures the underlying conditional relationship between low- and high-resolution data. Unlike DMs that focus on learning the marginal distributions $p(\mathbf{x})$ or $p(\mathbf{y})$, the SR3 framework focuses on modeling the conditional probability distribution $p(\mathbf{y} \mid \mathbf{x})$. Specifically, SR3 learns a parameterized approximation of $p(\mathbf{y} \mid \mathbf{x})$ by employing DDPMs to map low-resolution inputs $\mathbf{x}$ to their corresponding high-resolution outputs $\mathbf{y}$. The conditional forward and reverse diffusion processes are illustrated in Figure 10.

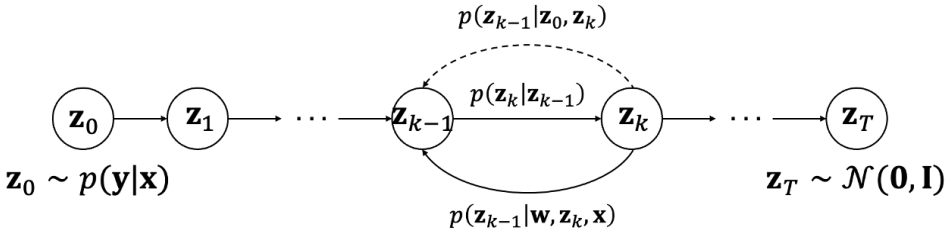


Figure 10: Framework of SR3. SR3 consists of a forward process (rightward arrows) and a reverse process (leftward arrows). The forward process starts from $\mathbf{z}_0$, representing $p(\mathbf{y} \mid \mathbf{x})$, and progressively adds noise via $p(\mathbf{z}_k \mid \mathbf{z}_{k-1}, \mathbf{x})$, ending at $\mathbf{z}_T$. The reverse process then denoises $\mathbf{z}_T$ back to $\mathbf{z}_0$ using $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x}, \mathbf{w})$ (dashed arrows), approximating the true reverse transition $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{x}, \mathbf{z}_0)$.

For the conditonal forward process, the conditional diffusion process is conducted following the diffusion kernel defined in (7). The marginal distribution of the corrupted sample $\mathbf{z}_k$ at timestep k given the original input $\mathbf{z}_0$ is modeled as:

$$p(\mathbf{z}_k \mid \mathbf{z}_0) = \mathcal{N}\left(\sqrt{\alpha_k}\mathbf{z}_0, (1-\alpha_k)\mathbf{I}\right), \tag{14}$$

where $\alpha_k = \prod_{i=1}^{k}(1-\beta_i)$, and β_i controls the variance of the Gaussian noise added at each timestep. The forward process of SR3 is similar to that of classical DMs, through progressively corrupting the original input by adding noise until it reaches a nearly isotropic Gaussian distribution.

In the conditional reverse process, the SR3 model seeks to recover the high-resolution output by reversing this noising process. The transition probability for the reverse diffusion step can be derived using the conjugacy property of Gaussian distributions [43], leading to:

$$p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{z}_0\right) = \mathcal{N}\left(\mathbf{m}_k\left(\mathbf{z}_k, \mathbf{z}_0\right), \sigma_k^2\mathbf{I}\right),$$

where the mean $\mathbf{m}_k$ and variance σ_k^2 are given by:

$$\begin{aligned}
\mathbf{m}_k\left(\mathbf{z}_k, \mathbf{z}_0\right) &= \frac{\left(1-\alpha_{k-1}\right)\sqrt{1-\beta_k}\,\mathbf{z}_k + \sqrt{\alpha_{k-1}}\beta_k\,\mathbf{z}_0}{1-\alpha_k}, \\
\sigma_k^2 &= \frac{\beta_k\left(1-\alpha_{k-1}\right)}{1-\alpha_k}.
\end{aligned}$$

This reverse diffusion process iteratively refines the noisy input toward a high-resolution output that is consistent with the conditional distribution $p\left(\mathbf{y} \mid \mathbf{x}\right)$.

3.2.2 Training SR3

In the context of SR3, the target output $\mathbf{z}_0$ remains unobservable, making it challenging to explicitly determine the reversed transition probability $p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{z}_0\right)$. As discussed in the framework of DMs, when the number of timesteps is sufficiently large, the reverse process follows a form similar to the forward process. Therefore, the learnable reverse Markovian process in SR3 is parameterized as:

$$p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w}, \mathbf{x}\right) = \mathcal{N}\left(\mu\left(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}\right), \beta_k\mathbf{I}\right), \tag{15}$$

where $\mu\left(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}\right)$ is a neural network with parameters $\mathbf{w}$, trained to approximate the true reverse transition mean.

Following the total noise formulation in DMs which is defined in (14), the corrupted latent state $\mathbf{z}_k$ at step k can be expressed as:

$$\mathbf{z}_0 = \frac{1}{\sqrt{\alpha_k}}\mathbf{z}_k - \frac{\sqrt{1-\alpha_k}}{\sqrt{\alpha_k}}\epsilon_k,$$

where $\epsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the total Gaussian noise added during the forward process. By substituting the posterior probability $p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{z}_0\right)$, its mean is given by:

$$\mathbf{m}_k\left(\mathbf{z}_k, \mathbf{z}_0, \mathbf{x}\right) = \frac{1}{\sqrt{1-\beta_k}}\left\{\mathbf{z}_k - \frac{\beta_k}{\sqrt{1-\alpha_k}}\epsilon_k\right\}.$$

This indicates that the mean of the posterior probability depends only on $\mathbf{z}_k$ and the total noise added during the forward diffusion process up to time step k. To model this relationship, a deep neural network $\mathbf{g}\left(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}\right)$ is introduced to predict the total noise and reconstruct $\mathbf{z}_0$. Consequently, the approximated mean function for the learned reverse process in (15) is defined as:

$$\mu\left(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}\right) = \frac{1}{\sqrt{1-\beta_k}}\left\{\mathbf{z}_k - \frac{\beta_k}{\sqrt{1-\alpha_k}}\mathbf{g}\left(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}\right)\right\}. \tag{16}$$

Therefore, similar to the training objective in DMs, the training objective of SR3 is to minimize the KL divergence between the true posterior distribution $p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{z}_0\right)$ and the learned transition probability $p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w}, \mathbf{x}\right)$, formulated as:

$$\begin{aligned}
&\arg\min_{\mathbf{w}} \mathrm{KL}\left(p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{z}_0\right) \| p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w}, \mathbf{x}\right)\right) \\
&= \arg\min_{\mathbf{w}} \mathrm{KL}\left(\mathcal{N}\left(\mathbf{m}_k\left(\mathbf{z}_k, \mathbf{z}_0\right), \sigma_k^2\mathbf{I}\right), \mathcal{N}\left(\mu\left(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}\right), \beta_k\mathbf{I}\right)\right) \\
&= \arg\min_{\mathbf{w}} \frac{\beta_k}{2\left(1-\alpha_k\right)\left(1-\beta_k\right)} \|\mathbf{g}(\underbrace{\sqrt{\alpha_k}\mathbf{z}_0 + \sqrt{1-\alpha_k}\epsilon_k}_{\mathbf{z}_k}, \mathbf{w}, k, \mathbf{x}) - \epsilon_k\|^2,
\end{aligned}$$

where $\epsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $(\mathbf{x}, \mathbf{y})$ is sampled from the dataset $\mathcal{D}$. In practice, the weighting factor $\frac{\beta_k}{2(1-\alpha_k)(1-\beta_k)}$ is omitted to ensure that each diffusion timestep contributes equally to the overall loss. Consequently, the final objective function for training the parameters $\mathbf{w}$ is:

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})}\mathbb{E}_{(\epsilon_k,k)} \|\mathbf{g}(\underbrace{\sqrt{\alpha_k}\mathbf{z}_0 + \sqrt{1-\alpha_k}\epsilon_k}_{\mathbf{z}_k}, \mathbf{w}, k, \mathbf{x}) - \epsilon_k\|^2, \tag{17}$$

where the outer expectation is taken over the dataset distribution $p_{\mathcal{D}}(\mathbf{x}, \mathbf{y})$, and the inner expectation is over the diffusion timestep k and sampled total noise ϵ_k. This two-level expectation structure ensures that the model learns to predict the noise corruption across all stages of the diffusion process, conditioned on varying input pairs $(\mathbf{x}, \mathbf{y})$.

The corresponding SGD training procedure is outlined in Algorithm 3.

Algorithm 3 Super-Resolution Residual Refinement (SR3) Training

Input: Training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, Noise schedule $\{\beta_1, \ldots, \beta_T\}$
Output: Neural network parameters $\mathbf{w}$

for $k \in \{1, \ldots, T\}$ **do**
  $\alpha_k \leftarrow \prod_{i=1}^{k}(1 - \beta_i)$ ▷ Compute noise scaling factor
end for
repeat
  $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$ ▷ Sample input-output pair
  $k \sim \{1, \ldots, T\}$ ▷ Sample timestep from the Markov chain
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Generate noise vector
  $\mathbf{z}_k \leftarrow \sqrt{\alpha_k}\mathbf{y} + \sqrt{1 - \alpha_k}\,\epsilon$ ▷ Add noise to high-resolution target
  $\mathcal{L}(\mathbf{w}) \leftarrow \|\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}) - \epsilon\|^2$ ▷ Compute reconstruction loss
  Update parameters $\mathbf{w}$ using gradient descent
until converged
return $\mathbf{w}$

Similar to the training process of DMs as outlined in Figure 7 and Algorithm 1, SR3 is trained by fitting the total noise ϵ added to the original input $\mathbf{z}_0$ using a deep neural network $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x})$. It is important to note that the structure of this neural network differs from that of DMs, as it performs the fitting based on low-resolution $\mathbf{x}$, which is a key characteristic of conditional diffusion models. An example of U-Net architecture of SR3 is given in Figure 11.

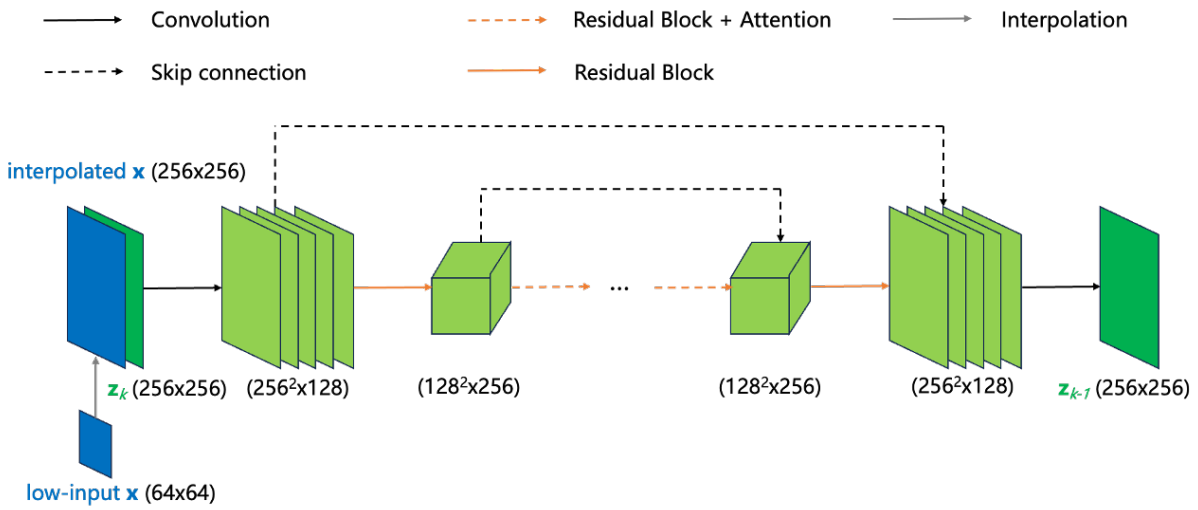


Figure 11: An examle of U-Net architecture of SR3. Similar to DMs, the network predicts noise from a noisy latent state $\mathbf{z}_k$. However, SR3 conditions on the low-resolution input $\mathbf{x}$, which is upsampled via interpolation and concatenated with $\mathbf{z}_k$ before being fed into the model. The figure shows the activiation dimensions for a $64 \times 64 \rightarrow 256 \times 256$ super-resolution.

3.2.3 Sampling via SR3

Once the total noise prediction neural network $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x})$ finishes training, the reverse transition probability $p(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w}, \mathbf{x})$ can be derived. Given a noisy latent state $\mathbf{z}_t$, the target $\mathbf{z}_0 \sim p(\mathbf{y} \mid \mathbf{x})$ can be estimated by:

$$\hat{\mathbf{z}}_0 = \frac{1}{\sqrt{\alpha_k}}\mathbf{z}_k - \frac{\sqrt{1-\alpha_k}}{\sqrt{\alpha_k}}\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}).$$

By substituting the mean of the posterior distribution of the normal form $p\left(\mathbf{z}_{k-1} \mid \mathbf{z}_k, \mathbf{w}, \mathbf{x}\right)$ defined in (16) and using the variance β_k (as defined by the forward process in [19]), each step in the iterative refinement under the low-resolution input $\mathbf{x}$ condition can be derived. This iterative process takes the following form:

$$\mathbf{z}_{k-1} \leftarrow \mu\left(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}\right) + \sqrt{\beta_k} = \frac{1}{\sqrt{1-\beta_k}}\left(\mathbf{z}_k - \frac{\beta_k}{\sqrt{1-\alpha_k}}\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x})\right) + \sqrt{\beta_k}.$$

The sampling procedure for SR3 follows the Markov chain in a sequential manner, similar to DMs. The detailed sampling algorithm is given in Algorithm 4.

Algorithm 4 Super-Resolution Residual Refinement (SR3) Sampling

Input: Trained refinement network $\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x})$, Low-resolution input $\mathbf{x}$, Noise schedule $\{\beta_1, \ldots, \beta_T\}$
Output: High-resolution output $\mathbf{y}$

$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Initialize with Gaussian noise
for $k = T, T-1, \ldots, 2$ **do**
  $\alpha_k \leftarrow \prod_{i=1}^{k}(1-\beta_i)$ ▷ Compute scaling factor
  $\mu\left(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}\right) \leftarrow \frac{1}{\sqrt{1-\beta_k}}\left(\mathbf{z}_k - \frac{\beta_k}{\sqrt{1-\alpha_k}}\mathbf{g}(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x})\right)$
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Sample noise vector
  $\mathbf{z}_{k-1} \leftarrow \mu(\mathbf{z}_k, \mathbf{w}, k, \mathbf{x}) + \sqrt{\beta_k}\epsilon$ ▷ Add noise
end for
$\mathbf{y} = \frac{1}{\sqrt{1-\beta_1}}\left(\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}}\mathbf{g}(\mathbf{z}_1, \mathbf{w}, 1, \mathbf{x})\right)$ ▷ Final high-resolution reconstruction
return $\mathbf{y}$

4 Experiments

As discussed in Section 1, limitations in the current measurement infrastructure and the inherent randomness of atmospheric turbulence hinder the accurate capture of wind behavior. While LES can provide high-resolution simulations, its computational cost renders it impractical for real-time or large-scale industrial applications. To address this challenge, this section explores using SR3, introduced in earlier sections, to enhance the spatial resolution of wind observations.

Specifically, real-world wind measurements are typically available at a coarse resolution of approximately 2 km, whereas both factual high-resolution observations and LES outputs are available at a much finer resolution of about 120 m. Therefore, the aiming of experiment part is to reconstruct fine-scale wind fields from coarse input data using the SR3 framework, which can be formulated as a 16× super-resolution task.

4.1 Experiment Setup

4.1.1 Dataset description

To enable 16× super-resolution using SR3 model, a suitable training dataset must first be constructed. In case of this thesis, the high-resolution ground-truth data is obtained from LES, which contains fine-grained turbulence field data over the Netherlands and adjacent coastal regions in a certain height. The LES dataset offers hourly snapshots, where each frame has a spatial resolution of 120 meters and covers a large geographical area, resulting in a data size of 3840×2560 grid points per record.

However, directly using such large-scale input would be computationally inefficient and impractical for training. To address this, each hourly LES field is divided into smaller patches of 256×256 grid points, enabling the model to be trained on localized wind field patterns. This approach not only reduces computational cost but also promotes the learning of more generalizable features for 16× super-resolution of turbulence.

Following the SR3 training process described in Algorithm 3, the training set consists of paired samples $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ is the low-resolution input and $\mathbf{y}$ is the corresponding high-resolution output. The high-resolution patches $\mathbf{y}$ are directly extracted from LES outputs, while the low-resolution counterparts $\mathbf{x}$ are generated by applying downsampling with a factor of 16 (i.e., from 256×256 to 16×16) and then upsampling back to 256×256 using interpolation. This mirrors the SR3 pipeline where the diffusion process is conditioned on an interpolated low-resolution image.

For the experiments in this thesis, each single-hour LES record, with a size of 2560×3840 grid points, is divided into smaller non-overlapping patches of 256×256 grid points. This partitioning method allows for the generation of 150 small patches from a single hourly record. To ensure that the training and testing sets do not overlap, data from January 2020 is used exclusively

for the training set, resulting in a total of $150 \times 24 \times 31 = 111600$ training samples. For the testing set, data from May 1, 2020 is selected, and the same patch extraction method is applied to generate the corresponding low-resolution inputs and high-resolution ground truth. Figure 12 illustrates the process to construct training set.

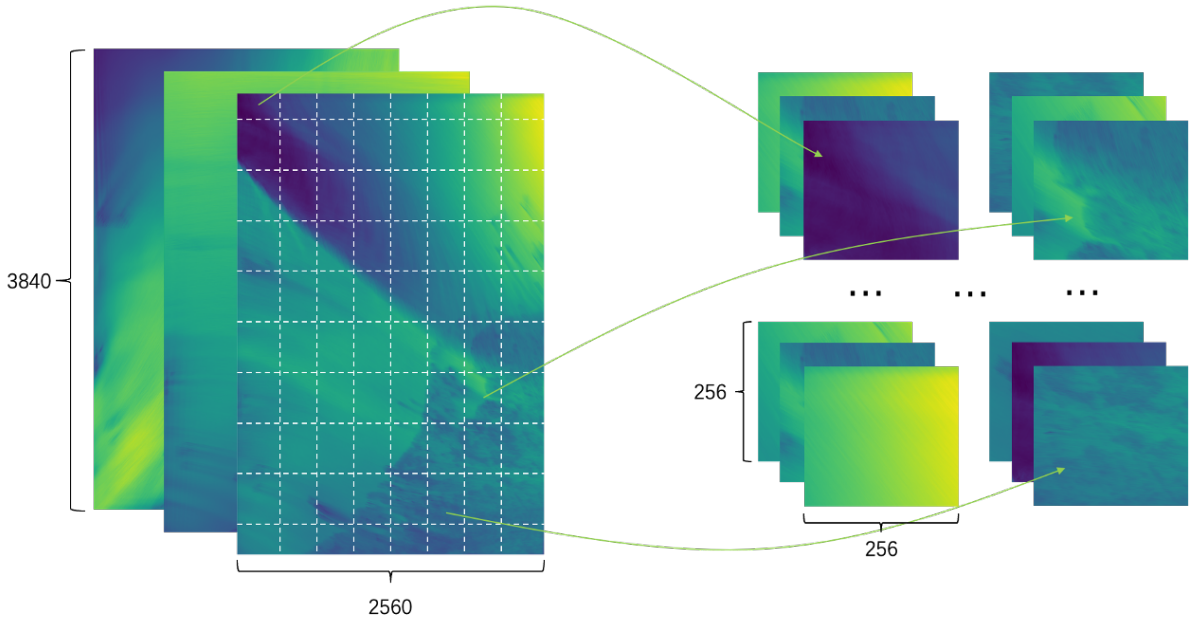


Figure 12: Each large image (left) represents one hour of LES data on a 3840×2560 spatial grid. The training set (right) is constructed by dividing these large datasets into multiple 256×256 patches. This patch-based construction both diversifies the training samples and improves training efficiency.

4.1.2 Model structure

Based on the construction of the training set, the experiment focuses on achieving 16× super-resolution from 16×16 to 256×256. Since super-resolution is inherently an ill-posed problem, excessively large scaling factors, such as 16×, can result in significant information loss in the input, making it difficult for the SR3 model to effectively reconstruct high-resolution details. Therefore, following the approach proposed by [43], a progressive SR3 strategy is employed. This involves using two SR3 models sequentially, where each model performs a 4× super-resolution.

The two stages are first stage ($16 \times 16 \rightarrow 64 \times 64$) and second stage ($64 \times 64 \rightarrow 256 \times 256$), as illustrated in Figure 13. The underlying U-Net architecture for both stages is detailed in the first and second rows of Table 1.

Tasks	Channel dimension	Depth multipliers	ResNet blocks	# Parameters
$16^2 \rightarrow 64^2$	64	$\{1, 2, 4, 8, 4, 2, 1\}$	2	550M
$64^2 \rightarrow 256^2$	64	$\{1, 2, 4, 8, 16, 8, 4, 2, 1\}$	3	1.2B
$16^2 \rightarrow 256^2$	128	$\{1, 2, 4, 8, 16, 8, 4, 2, 1\}$	3	1.5B

Table 1: U-Net architectures of SR3 models.The first row shows the configuration of the U-Net used in the first stage of progressive SR3, while the second row illustrates the architecture for the second stage. The final row presents the structure used in 16× SR3. The "Channel dimension" corresponds to the dimension of the first layer, and the "Depth multipliers" are applied to subsequent resolutions.

Additionally, to evaluate the performance of SR3, a baseline model using a standard CNNs with the same structure is trained as a benchmark. Since SR3 incorporates diffusion mechanisms over the conventional U-Net structure, comparing the performance of SR3 against the CNNs model allows to assess the benefits of these enhancements.

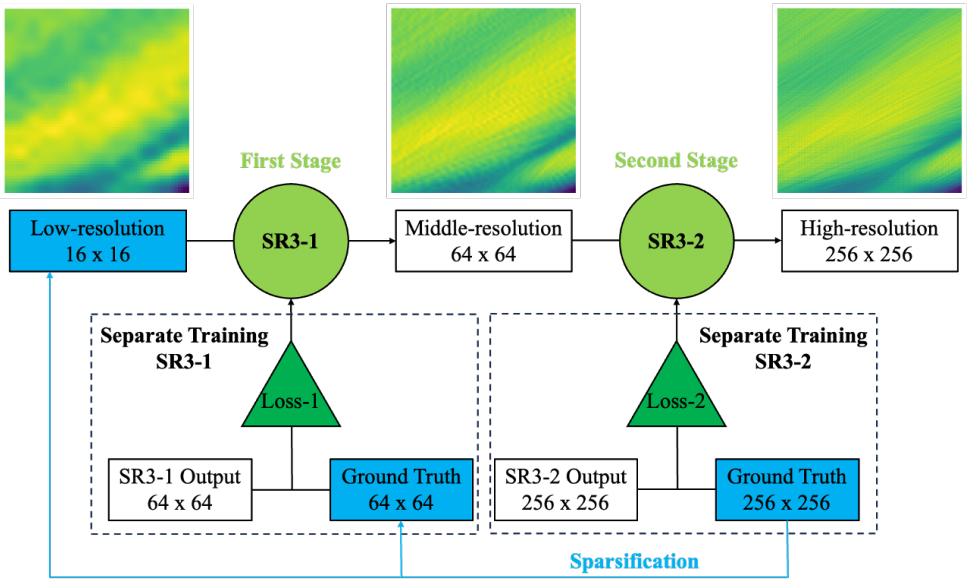


Figure 13: Progressive SR3 framework. The model consists of two independently trained SR3 networks, each performing 4× super-resolution. In the first stage, SR3-1 upsamples the 16×16 low-resolution input to an intermediate resolution of 64×64, while in the second stage, SR3-2 further enhances the resolution from 64×64 to 256×256.

4.1.3 Experiment design

Building on the previously introduced models, the experiments are designed to compare the performance of different super-resolution methods, focusing on two key experimental setups:

- Direct 16× SR3 vs. CNNs

  In the first experiment, the performance of the 16× SR3 model is evaluated by directly upscaling low-resolution images from 16×16 to 256×256 grid points. The SR3 model is compared against a CNN model with the same U-Net architecture, which serves as the baseline.

- Two-stage 4× SR3 vs. CNNs

  In the second experiment, a progressive SR3 strategy is implemented, where two SR3 models are applied sequentially to upscale the image in two stages: from 16×16 to 64×64 in the first stage, and from 64×64 to 256×256 in the second stage. This progressive approach is compared against a similar two-stage CNNs, where each stage performs a 4× upscaling.

For both experiments, MSE and SSIM are used to assess the pixel-wise accuracy and perceptual similarity of the generated high-resolution data respectively. In addition, the training time per

epoch and sampling (inference) time are recorded to evaluate the computational efficiency of each model. Furthermore, visual comparisons of the generated high-resolution images will also be provided to assess the perceptual quality.

4.2 Experiment Results

4.2.1 High-resolution reconstruction via progressive upscaling

The experiments are conducted on an NVIDIA A100 GPU. The SR3 architecture is illustrated in Figure 11, and the corresponding architectural parameters are provided in Table 1. For baseline comparisons, the CNNs models adopt the same architecture as their corresponding SR3 models, but without the diffusion process.

To address the 16× super-resolution task, the progressive super-resolution framework decomposes the task into two successive 4× super-resolution stages. The first stage performs upscaling from 16×16 to 64×64, referred to as SR3-1. The second stage further upscales the result to 256×256, referred to as SR3-2. Each stage is trained independently to optimize performance. The traning results and hyperparameters of SR3-1 and SR3-2 are recorded in Table 2.

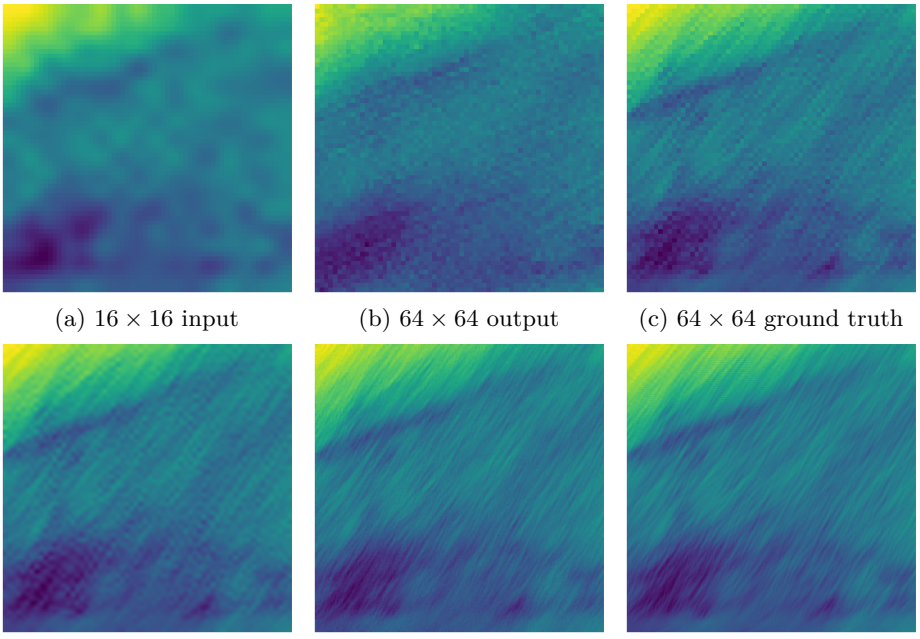
		SR3-1 (4×)	SR3-2 (4×)
Hyper-parameters	Learning rate	10^{-4}	10^{-3}
	Batchsize	8	8
	β starts	10^{-4}	10^{-4}
	β ends	0.02	0.02
	# diffusion timesteps	1000	1000
	Interpolation methods	`Bicubic`	`Bicubic`
	Training loss measure	L2	L2
	Optimization methods	`Adam`	`Adam`
	Earlystop patience	5	5
Training results	Training epochs	27	23
	Average time per epoch	650 s	3250 s
	Final training loss	0.18	0.06

Table 2: Hyperparameters and training results for the progressive SR3 models

Although both stages are designed to perform a 4× super-resolution task, there is a noticeable gap in training loss between SR3-1 and SR3-2, with 0.18 and 0.06 respectively. This discrepancy can be attributed to the difference in available information at each stage. Specifically, SR3-1 upscales from 16×16 to 64×64, where the input is extremely low resolution and lacks sufficient structural details. In contrast, SR3-2 performs upscaling from 64×64 to 256×256, where the input already contains richer spatial and semantic information. Consequently, SR3-2 is able to make more informed estimations about the high-frequency details of the output.

This phenomenon is also reflected in the sampling outputs from the two models given in Figure 14. The results generated by SR3-2 demonstrate finer structural details and more accurate reconstruction, which is more close to ground truth, benefiting from the higher-quality input.

For evaluation, testing was performed on an independent dataset from May 1, 2020. To ensure consistency, the testing set was constructed in the same manner as the training set in, by segmenting a large spatial-temporal record into multiple non-overlapping 256×256 image patches, as shown in Figure 12. Importantly, the testing data was from a much later time point than the training set, which was taken exclusively from January 2020. This deliberate temporal separation introduces significant variation in wind patterns between the two datasets, thereby enabling a more rigorous assessment of the model's generalization capability beyond seasonal or short-term dynamics.

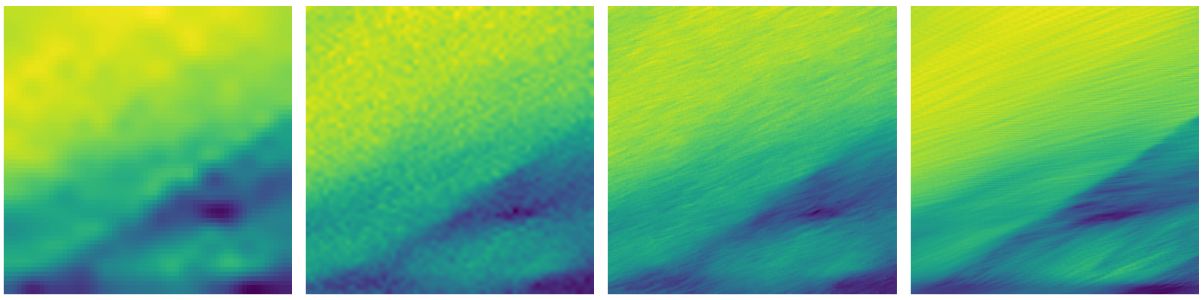


(a) 16×16 input (b) 64×64 output (c) 64×64 ground truth

(d) 64×64 input (e) 256×256 ouput (f) 256×256 ground truth

Figure 14: Results of SR3-1 and SR3-2. (a), (b), and (c) show the inference results for the SR3-1, where the input is scaled from 16×16 to 64×64. (d), (e), and (f) present inference results for the SR3-2, where the input is scaled from 64×64 to 256×256. The ground truth images are provided for comparison.

During sampling, the same model-specific hyperparameters were used as in training in Table 2, including the number of diffusion timesteps and the β (noise variance control) scheduling. For a single input, the total sampling time for the two-stage SR3 model was approximately 15 seconds for SR3-1 and 300 seconds for SR3-2. A visual example of the complete $16\times$ super-resolution result is shown in Figure 15.

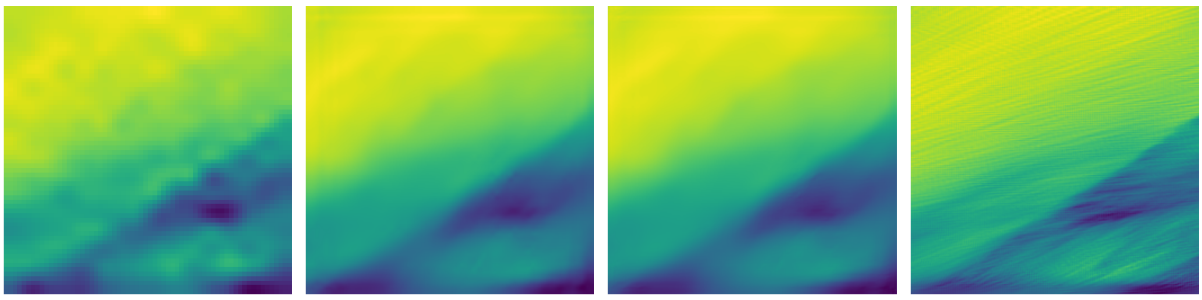


(a) 16×16 (b) 64×64 (c) 256×256 (d) ground truth

Figure 15: Results of the progressive SR3 model. The low-resolution input (a) is first upscaled by $4\times$ using SR3-1, producing a middle-resolution data (b). This is then further upscaled by another $4\times$ using SR3-2 to obtain the high-resolution reconstruction (c). The ground truth high-resolution data (d) is used for performance evaluation.

To evaluate the effectiveness of SR3 in performing super-resolution tasks, a baseline comparison is conducted using CNNs models that share the same U-Net architecture as SR3, but without the diffusion process. While U-Net is not the most typical choice for classical CNN-based super-resolution methods, which often use encoder-decoder or residual blocks such as in SRCNN [11], this architecture ensures a fair comparison in terms of model structure and capacity. This setup isolates the contribution of the diffusion process itself, rather than differences in network architecture. The architectural details are provided in Table 1. CNNs remain a widely-used baseline in super-resolution, and this design allows to assess how much performance gain is attributable specifically to the diffusion mechanism.

A visual example of the reconstruction results produced by the two-stage CNNs model is presented in Figure 16. This model follows the same progressive structure as SR3: the first CNNs performs $4\times$ upscaling from 16×16 to 64×64, and the second CNNs further upsamples the result to 256×256.



(a) 16×16 (b) 64×64 (c) 256×256 (d) ground truth

Figure 16: Results of the progressive CNNs model. The low-resolution input (a) is first upscaled by $4\times$ using the first-stage CNNs, producing a middle-resolution data (b). This is then further upscaled by another $4\times$ using the second-stage CNNs to obtain the high-resolution reconstruction (c). The ground truth high-resolution data (d) is used for performance evaluation. The CNNs architectures are the same of the U-Net structures in SR3.

The quantitative evaluation results comparing progressive SR3 and progressive CNNs are summarized in Table 3. These results are based on error metrics between the predicted high-resolution data and the ground truth. From the evaluation results, the SR3 model consistently

outperforms CNNs in terms of MSE and SSIM, highlighting its superior reconstruction capabilities. This performance gain is attributed to the fundamental difference in model mechanisms: while CNNs rely on deterministic upscaling operations, SR3 employs a probabilistic denoising diffusion process that captures complex data distributions and structural dependencies more effectively.

Evaluation metrics	Progressive SR3	Progressive CNNs
MSE	0.014	0.0256
Variance	0.0139	0.0171
SSIM	0.6	0.54
Average training time per epoch	650 + 3250 s	45 + 1670 s
Average sampling time per input	15 + 300 s	both less than 1 s

Table 3: Evaluation metrics of progressive SR3 and progressive CNNs on the testing set.

However, this performance advantage comes at a computational cost. Both training and sampling times of SR3 are significantly longer compared to CNNs. This is particularly evident during the sampling phase. For CNNs, generating a high-resolution image takes less than one second, while SR3 requires considerably more time, with about 300 seconds. This increased cost is due to the iterative nature of the sampling process, which is directly proportional to the number of diffusion timesteps. This behavior aligns with the expected dynamics of the SR3 sampling algorithm, as outlined in Algorithm 4, where each sample must be sequentially generated through a Markov chain of reverse noise steps.

Visual comparisons in Figure 15 and Figure 16 further reinforce the quantitative findings. SR3 reconstructions demonstrate superior detail preservation, while CNNs-generated data tend to be overly smooth and lack fine textures. This is likely due to the interpolation-based upscaling mechanisms commonly employed in CNNs, which rely on local grid point neighborhoods and often result in blurred edges and loss of high-frequency information [55].

4.2.2 High-resolution reconstruction via directly upscaling

In this section, a new SR3 model is constructed to address the 16× super-resolution task. Unlike the progressive SR3 mentioned in the previous section, which performs super-resolution through two successive 4× SR3 models, the direct SR3 model attempts to upscale the low-resolution input (16×16) to high-resolution output (256×256) in a single step. The network structure of the direct SR3 model is provided in Table 1.

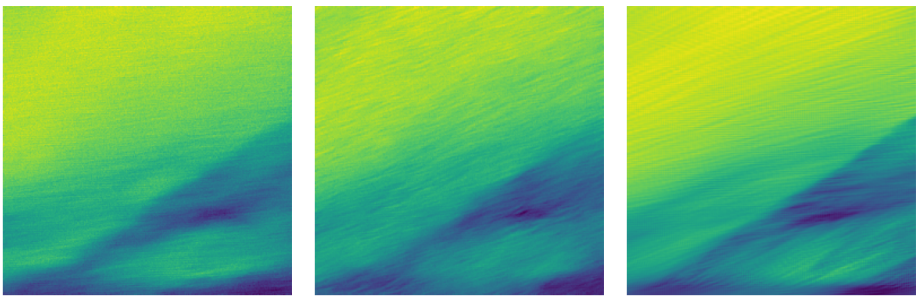
The experimental results in Table 4 demonstrate that the progressive strategy yields superior performance in both MSE and SSIM compared to the direct approach. Furthermore, the lower variance further indicates that the SR3 can generates more stable and consistent results. This performance advantage can be attributed to the ill-posed nature of the super-resolution problem, which becomes increasingly challenging as the upscaling factor grows.

Evaluation metrics	Direct 16× SR3	Progressive 16× SR3
MSE	0.0248	0.014
Variance	0.0195	0.0139
SSIM	0.57	0.6
Average training time per epoch	8300 s	650 + 3250 s
Average sampling time per input	759 s	15 + 300 s

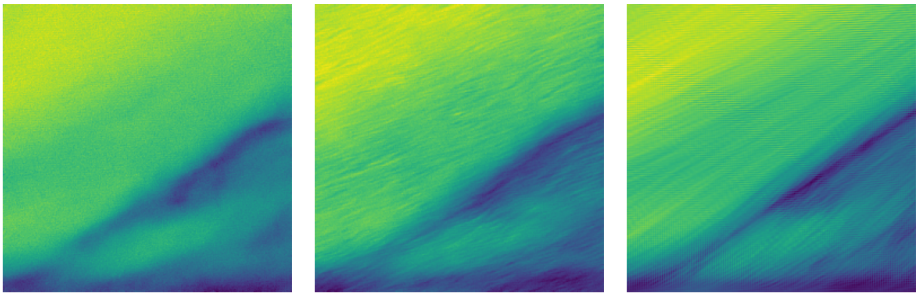
Table 4: Evaluation metrics of direct SR3 and progressive SR3 on the testing set.

From an inverse problem perspective, super-resolution aims to recover high-resolution data $\mathbf{y} \in \mathbb{R}^{D\times D}$ from its corresponding low-resolution observation $\mathbf{x} \in \mathbb{R}^{d\times d}$, where the scaling factor is defined as $s = D/d$. This process can be understood as estimating the inverse function $\mathcal{D}_s^{-1}$ of a downsampling operator $\mathcal{D}_s$, as introduced in (1). However, this inverse problem is inherently ill-posed due to the significant loss of high-frequency information during the downsampling process. The degree of information loss increases exponentially with the scaling factor s, making the recovery of fine details extremely difficult when s is large (e.g., $s = 16$).

In such cases, directly applying a 16× SR3 model often results in outputs that only preserve the coarse structure of the original data, while failing to reconstruct fine-grained textures and details. In contrast, the progressive SR3 framework decomposes the difficult 16× task into two simpler 4× sub-tasks, thereby reducing the reconstruction ambiguity and allowing each model stage to focus on more localized structures. This hierarchical design enables the model to better preserve both global context and local fidelity, as illustrated qualitatively in Figure 17.



(a) direct 16× SR3 (b) progressive 4× SR3 (c) ground truth



(d) direct 16× SR3 (e) progressive 4× SR3 (f) ground truth

Figure 17: Results of direct SR3 and progressive SR3 models. (a) shows the inference result from the direct SR3 model, while (b) displays the corresponding result from the progressive SR3 model. (c) presents the ground truth for comparison. (d), (e), and (f) illustrate another test sample, following the same order: direct SR3 result, progressive SR3 result, and ground truth, respectively.

In addition to its superior reconstruction quality, the progressive model also exhibits an advantage in computational efficiency. Specifically, the total sampling time for the progressive SR3 model is significantly lower than that of the direct SR3 model. As shown in Table 1, the direct model requires a substantially larger NN to handle the more complex one-shot 16× upscaling task. This increase in model size leads to longer inference times, making the direct approach less efficient overall.

When comparing the performance of the direct SR3 model with that of the progressive CNNs, the results appear similar in terms of MSE and variance. As illustrated in Figure 16 and Figure 17, both models struggle to recover fine-grained high-frequency details. However, in terms of SSIM, direct SR3 achieves a higher score compared to progressive CNNs, indicating that the direct SR3 appears to better preserve large-scale structural patterns and semantic layout, even if the grid-point-level accuracy is comparable.

4.2.3 Hyperparameters tuning

In this section, hyperparameter tuning was performed for the progressive SR3 model to optimize its performance. The tuned hyperparameters, listed in Table 2, include the learning rate, interpolation method, loss function, and optimization algorithm. These were selected through grid search, and the results are summarized in Table 5. For fair comparison, the CNNs baseline were implemented with the same architecture and training setup as SR3, but without the diffusion process. However, no dedicated hyperparameter tuning was performed for the CNNs, which may put them at a slight disadvantage in terms of optimal performance.

Hyperparameters	Tuning range	SR3-1	SR3-2
Learning rate	$\{10^{-3}, 10^{-4}\}$	10^{-4}	10^{-3}
Interpolation methods	{`Bicubic`, `Bilinear`}	`Bicubic`	`Bicubic`
Loss measure	{L1, L2}	L2	L2
Optimization methods	{`Adam`, `Momentum`}	`Adam`	`Adam`

Table 5: Hyperparameters tuning results of progressive SR3 model.

This outcome suggests that fine-tuning hyperparameters such as learning rate and interpolation methods can significantly influence the performance of the SR3 models. In particular, the choice of interpolation method directly affects the quality of the initial upsampled input, which in turn influences the difficulty of the subsequent denoising task. Since the diffusion process is conditioned on the upsampled input, as illustrated in Figure 11, a poor initial estimate can lead the denoising trajectory away from the ground truth. To investigate this effect, the following experiments compare two widely used interpolation strategies, bilinear and bicubic, to assess their impact on reconstruction quality.

Bilinear interpolation, as defined in (2), offers computational efficiency but often results in noticeable blurring, as it estimates the values using a linear approximation based on a limited local neighborhood. In contrast, bicubic interpolation, described in (3), utilizes a larger neigh-

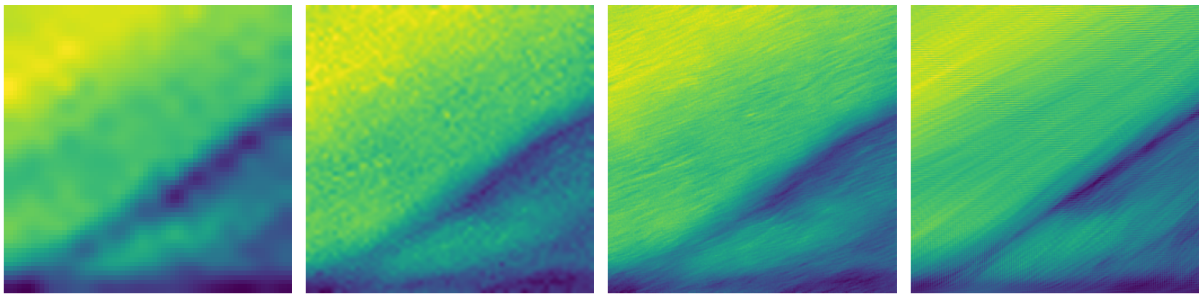
borhood and performs cubic convolution along both the horizontal and vertical axes. Although bicubic interpolation method is more computationally intensive, it typically produces smoother and more visually accurate results, especially around edges and fine textures.

In the context of super-resolution, bicubic interpolation is often favored as it retains more image detail, especially in high-resolution reconstructions [11, 31, 43]. This characteristic makes bicubic interpolation particularly effective for the SR3 model, where the goal is to generate high-quality, high-resolution images.

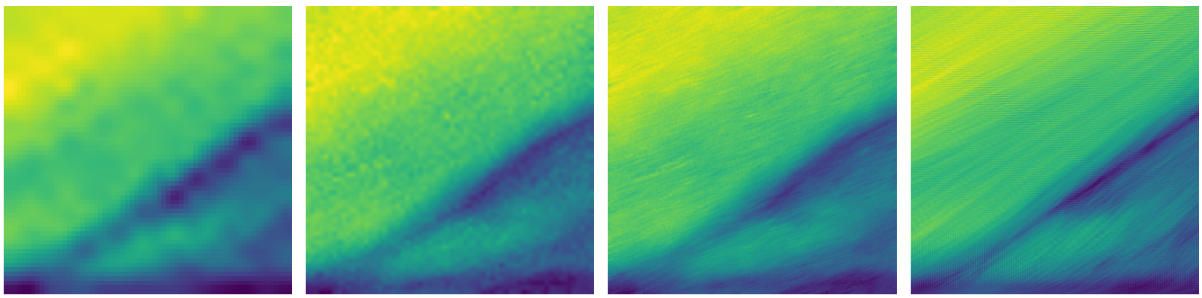
The results of the interpolation experiment, shown in Table 6, confirm that bicubic interpolation consistently outperforms bilinear interpolation across multiple metrics, including MSE, variance, and SSIM. Bicubic interpolation achieves better reconstruction quality, suggesting its superior ability to preserve structural details. This improvement can be attributed to its consideration of a larger neighborhood and smoother gradient transitions, which is essential for high-quality super-resolution.

Evaluation metrics	Bilinear interpolation	Bicubic interpolation
MSE	0.0154	0.014
Variance	0.0165	0.0139
SSIM	0.57	0.6
Average interpolation time	43 s	68 s

Table 6: Evaluation metrics of bicubic and bilinear interpolation methods on the testing set.



(a) 16×16 (b) 64×64 (`Bicubic`) (c) 256×256 (`Bicubic`) (d) ground truth



(e) 16×16 (f) 64×64 (`Bilinear`) (g) 256×256 (`Bilinear`) (h) ground truth

Figure 18: Results of bicubic and biliear interpolation methods. (a), (b), and (c) show the low-resolution (16×16), medium-resolution (64×64), and high-resolution outputs (256×256) using bicubic interpolation, respectively, with (d) as the ground truth. (e), (f), (g), and (h) show the corresponding results using bilinear interpolation.

Figure 18 visually demonstrates the difference in the quality of images reconstructed using

Bicubic and Bilinear interpolation. In the figures, Bicubic interpolation produces sharper, more detailed results, while Bilinear interpolation results in a softer, slightly blurred image.

Another interesting observation from the tuning process is the importance of the progressive reconstruction approach. When the first stage of the progressive SR3 model uses effective interpolation methods, it provides a solid foundation for the subsequent stages, which leads to improved final performance. Specifically, when the first stage interpolation is effective, it helps the second stage to further refine the reconstruction and achieve even higher-quality results. This highlights the value of using intermediate resolutions in progressive super-resolution models, where each stage builds upon the previous one to bridge the gap between low-resolution inputs and high-resolution outputs.

4.2.4 Joint learning of progressive SR3

In the previous section, the progressive SR3 model was trained in a decoupled manner—specifically, SR3-1 and SR3-2 were trained separately, as illustrated in Figure 13. The main advantage of this approach lies in the independence between the two stages, preventing interference between different SR3 models and allowing each to focus on its respective resolution scale. However, this separation comes at the cost of computational redundancy. Since both stages involve similar steps such as data loading and intermediate storage, the total training time and sampling latency increase significantly.

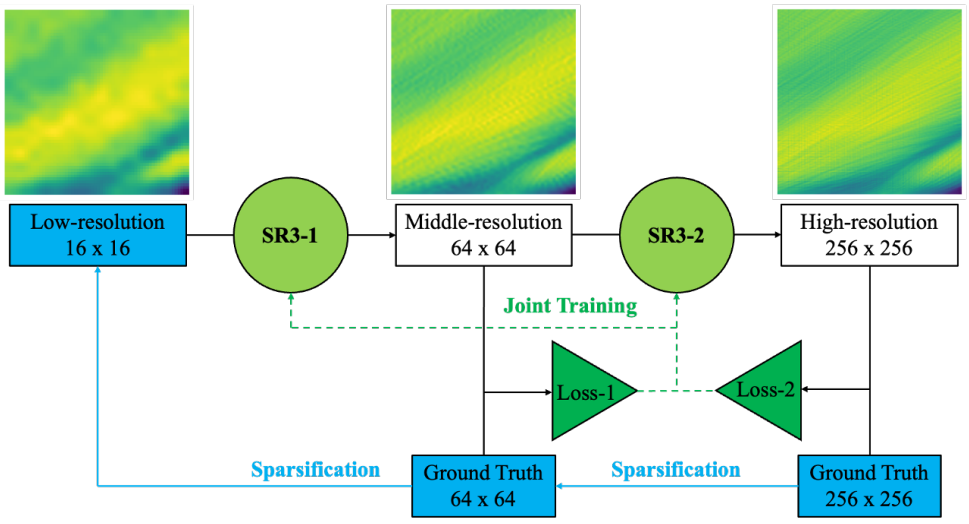


Figure 19: Joint training progressive SR3 framework. Unlike the progressive setup, the joint training model optimizes both SR3-1 and SR3-2 simultaneously. Both intermediate and final outputs are supervised using their respective ground truths through Loss-1 and Loss-2, enabling end-to-end learning and joint optimization of both networks for improved overall performance.

To address these inefficiencies, the separate training strategy is replaced with a joint training framework, as depicted in Figure 19, in which SR3-1 and SR3-2 are trained simultaneously within a unified optimization framework. In this setting, the total training loss is formulated as a weighted combination of the individual losses from each stage, and the model parameters are

updated end-to-end.. The total training loss $\mathcal{L}$ is given as

$$\mathcal{L}(\mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = \lambda_1 \mathcal{L}^{(1)}(\mathbf{w}^{(1)}) + \lambda_2 \mathcal{L}^{(2)}(\mathbf{w}^{(2)}), \tag{18}$$

where $\mathcal{L}^{(1)}$ and $\mathcal{L}^{(2)}$ denote the training losses for SR3-1 and SR3-2, respectively, and $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$ are the corresponding sets of trainable parameters. The weighting coefficients λ_1 and λ_2 provide flexibility in adjusting the contribution of each stage during training, enabling the model to emphasize the resolution level that is more critical to the specific super-resolution task.

In the experiments, the relative weighting between the two loss components plays a significant role in the quality of the final reconstruction. Assigning a higher weight to $\mathcal{L}^{(1)}$ (e.g., $\lambda_1 = 0.8$, $\lambda_2 = 0.2$) consistently leads to better overall performance. This indicates that producing a strong intermediate representation at the 64×64 resolution is essential for the second-stage model to perform effective refinement. In contrast, assigning excessive weight to $\mathcal{L}^{(2)}$ (e.g., $\lambda_1 = 0$ or $\lambda_2 = 1$) often results in degraded performance, with the final outputs appearing overly localized or pixelated. This is likely due to insufficient guidance in the intermediate stage, causing the second stage to overfit local features without reliable global structure. These observations motivate the use of a loss weighting strategy that prioritizes early-stage accuracy, typically by allocating a larger proportion of the total loss to $\mathcal{L}^{(1)}$.

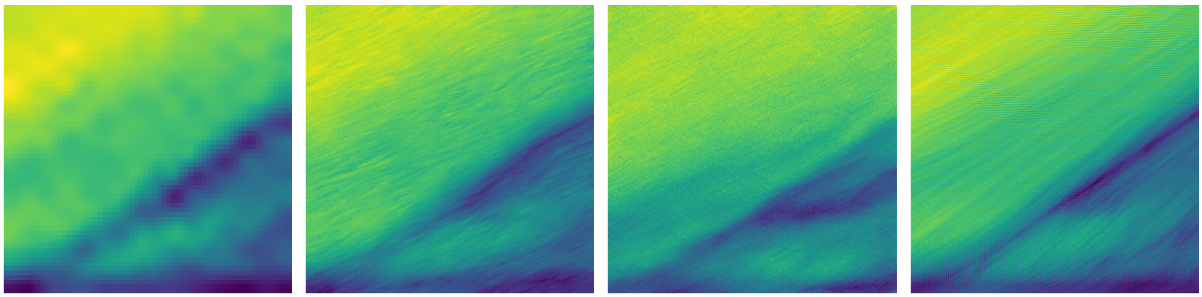
The experimental results presented in Table 7 compare the performance of the joint learning model (with weighting coefficients $\lambda_1 = 0.8$ and $\lambda_2 = 0.2$) against the separate learning strategy. While the separate learning model achieves slightly better scores in MSE and variance, the joint learning model performs noticeably better in terms of SSIM. This improvement suggests that the joint training strategy is more effective at preserving the structural and perceptual quality of the output, likely due to its end-to-end optimization that allows information to flow between both stages during training [59, 60]. Such interaction can guide the early SR3-1 to produce features that are more beneficial for the subsequent SR3-2, ultimately enhancing the overall structural consistency. A visual example is given in Figure 20.

Evaluation metrics	Separate learning	Joint learning
MSE	0.014	0.0172
Variance	0.0139	0.0198
SSIM	0.6	0.67
Average training time per epoch	650 + 3250 s	3750 s
Average sampling time per input	15 + 300 s	158 s

Table 7: Evaluation metrics of separate learning and joint learning strategies of the progressive SR3 model on the testing set.

Moreover, as previously discussed, joint training reduces redundant computational steps such as data loading and model initialization, which are duplicated in the separate training process. As shown in Table 7, this efficiency gain is reflected in the reduction in both training and sampling

processes. Although the reduction may appear modest in absolute terms, it becomes increasingly significant when scaling to larger datasets or higher-resolution tasks.



(a) 16×16 input (b) joint learning output (c) separate learning output (d) 256×256 ground truth

Figure 20: Results of joint learning and separate training of progressive SR3 models. (a) shows the low-resolution input (16×16). (b) and (c) present the high-resolution outputs from the jointly trained and separately trained progressive SR3 models, respectively. (d) displays the ground truth.

5 Discussion and Conclusion

5.1 Conclusions

This thesis investigates the application of diffusion-based generative models for high-resolution wind field reconstruction from sparse observations. By implementing and evaluating the SR3 framework, several important conclusions can be drawn:

- Generative modeling capacity: Compared to deterministic model such as CNNs, the stochastic denoising process of SR3 is more suitable for solving ill-posed super-resolution problems, where low-resolution input and its corresponding possible high-resolution outputs forms a one-to-many mapping, as shown in Figure 1. Deterministic models tends to offers an overly smooth results, while SR3 can capture the underlying distribution of possible solutions and enable the recovery of more structurally faithful details [50].

- Progressive refinement strategy: The design of progressive SR3 model divides a large 16× upscaling task into two manageable subproblems ($16 \times 16 \rightarrow 64 \times 64$ and $64 \times 64 \rightarrow 256 \times 256$), as illustrated in Figure 13. This approach outperforms the direct $16 \times 16 \rightarrow 256 \times 256$ upscaling, as it gradually bridges the semantic and structural resolution gap, improving stability and visual fidelity at each stage.

- Impact of intermediate quality: In a multi-stage SR3 pipeline, the quality of the intermediate reconstruction plays a crucial role in determining the final result. High-quality intermediate outputs provide more reliable priors for subsequent refinement, which suggests that more emphasis should be palced on optimizing earlier stages. For example, in joint training (Figure 19), increasing the weight of the loss function associated with the first stage can help guide the overall learning process more effectively.

- Interpolation as initialization: Interpolation methods serve as the initial upsampling step in super-resolution pipelines and have a notable impact on final output quality. Within the SR3 framework, this initial interpolation acts as a guide for the subsequent diffusion process, as shown in Figure 11. Experimental results show that bicubic interpolation leads to better final reconstructions than bilinear, as it better preserves edge information and gradient continuity [11, 31, 43].

- Joint vs. separate training: Joint training of the two SR3 stages yields improved structural reconstruction compared to separate training. This can be attributed to the benefits of end-to-end optimization, where gradients can flow across stages, allowing the model to coordinate learning across resolution levels [59, 60]. Additionally, joint training eliminates redundant operations such as repeated data loading and model re-initialization, resulting in notable gains in training and inference efficiency.

While these findings underscore the strengths of the SR3 approach, there are also important

practical considerations. Diffusion-based models, especially in multi-stage settings, are computationally intensive. Even with joint training optimizations, training and inference times remain significantly longer than other data-driven methods such as CNNs. Moreover, although the assumption of uniform low-resolution sampling may not reflect all real-world scenarios, it is a commonly used and reproducible setup in numerical weather modeling [13]. It serves as a valuable benchmark to isolate the model' s behavior under varying data densities.

In conclusion, the progressive SR3 framework presents a promising path for enhancing meteorological super-resolution pipelines. Its capacity to generate high-fidelity wind field reconstructions from sparse data may contribute to more accurate and data-driven weather modeling.

5.2 Future Research Directions

Despite the promising results achieved by the proposed SR3-based framework in generate high-fidelity wind field reconstructions problems, there remain several directions for further improvement and extension.

(a) Refinement of training objective

The current training objective minimizes the MSE between the predicted and true noise as illustrated in (17), which is standard in diffusion models. However, it ignore perceptual quality, which also serves as an important critiria in evaluating super-resolution results. Future work could consider adding reconstruction-based losses, such as SSIM, between the generated high-resolution image and the ground truth. Since SSIM is not meaningful when computed on noise, it should be applied to compare the final output $\hat{\mathbf{y}}$ with the reference $\mathbf{y}$:

$$\mathcal{L}(\mathbf{w}) = \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{SSIM}} = \mathcal{L}_{\text{MSE}} + \mu(1 - \text{SSIM}(\hat{\mathbf{y}}, \mathbf{y})).$$

Such a hybrid loss could encourage both statistical accuracy and perceptual sharpness.

(b) Improved guidance during sampling

Another promising direction is to enhance the fidelity and controllability of the sampling process. The diffusion framework in [45] demonstrates that guided sampling, where a low-resolution input or physics-based condition, such as PDE residual, is used to steer the reverse process, significantly improves reconstruction accuracy, particularly under sparse or mismatched input distributions. Similar guidance strategies, such as classifier-free guidance [21] or iterative sampling with noise injection and feedback [45], could be explored in SR3 to better exploit spatial priors and improve reconstruction accuracy.

References

[1] Ronald J Adrian. Twenty years of particle image velocimetry. *Experiments in fluids*, 39:159–169, 2005.

[2] Mohammed H Alsharif, Abu Jahid, Raju Kannadasan, and Mun-Kyeom Kim. Unleashing the potential of sixth generation (6g) wireless networks in smart energy grid management: A comprehensive review. *Energy Reports*, 11:1376–1398, 2024.

[3] John David Anderson and John Wendt. *Computational fluid dynamics*, volume 206. Springer, 1995.

[4] Arnav V Bhavsar and Ambasamudram N Rajagopalan. Range map superresolution-inpainting, and reconstruction from sparse data. *Computer Vision and Image Understanding*, 116(4):572–591, 2012.

[5] Christopher M Bishop and Hugh Bishop. *Deep learning: Foundations and concepts*. Springer Nature, 2023.

[6] Steven L Brunton and Bernd R Noack. Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews*, 67(5):050801, 2015.

[7] Michele Buzzicotti. Data reconstruction for complex flows using ai: Recent progress, obstacles, and perspectives. *Europhysics Letters*, 142(2):23001, 2023.

[8] Lichao Cao, Mingwei Ge, Xiaoxia Gao, Bowen Du, Baoliang Li, Zhi Huang, and Yongqian Liu. Wind farm layout optimization to minimize the wake induced turbulence effect on wind turbines. *Applied Energy*, 323:119599, 2022.

[9] Hongguang Chen, Xing Zhang, Yintian Liu, and Qiangyu Zeng. Generative adversarial networks capabilities for super-resolution reconstruction of weather radar echo images. *Atmosphere*, 10(9):555, 2019.

[10] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[11] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[12] William Feller. Retracted chapter: On the theory of stochastic processes, with particular reference to applications. In *Selected Papers I*, pages 769–798. Springer, 2015.

[13] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

[14] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution analysis via machine learning: a survey for fluid flows. *Theoretical and Computational Fluid Dynamics*, 37(4):421–444, 2023.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[16] Yusuke Hatanaka, Yannik Glaser, Geoff Galgon, Giuseppe Torri, and Peter Sadowski. Diffusion models for high-resolution solar forecasts. *arXiv preprint arXiv:2302.00170*, 2023.

[17] Warren E Heilman. Atmospheric turbulence and wildland fires: a review. *International journal of wildland fire*, 32(4):476–495, 2023.

[18] Geoffrey E Hinton and Richard Zemel. Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[20] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022.

[21] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[22] Yawen Huang, Ling Shao, and Alejandro F Frangi. Simultaneous super-resolution and cross-modality synthesis of 3d medical images using weakly-supervised joint convolutional sparse coding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6070–6079, 2017.

[23] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016.

[24] Ahsan Kareem. Emerging frontiers in wind engineering: Computing, stochastics, machine learning and beyond. *Journal of wind engineering and industrial aerodynamics*, 206:104320, 2020.

[25] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153–1160, 1981.

[26] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using

very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[27] Kwang In Kim and Younghee Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE transactions on pattern analysis and machine intelligence*, 32(6):1127–1133, 2010.

[28] Sookyung Kim, Sasha Ames, Jiwoo Lee, Chengzhu Zhang, Aaron C Wilson, and Dean Williams. Resolution reconstruction of climate data with pixel recursive model. In *2017 IEEE international conference on data mining workshops (ICDMW)*, pages 313–321. IEEE, 2017.

[29] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[30] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[31] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[32] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017.

[33] XuDong Ling, ChaoRong Li, FengQing Qin, LiHong Zhu, and Yuanyuan Huang. Two-stage rainfall-forecasting diffusion model. *IEEE Geoscience and Remote Sensing Letters*, 2024.

[34] Bo Liu, Jiupeng Tang, Haibo Huang, and Xi-Yun Lu. Deep learning methods for super-resolution reconstruction of turbulent flows. *Physics of fluids*, 32(2), 2020.

[35] Yunjie Liu, Evan Racah, Joaquin Correa, Amir Khosrowshahi, David Lavers, Kenneth Kunkel, Michael Wehner, William Collins, et al. Application of deep convolutional neural networks for detecting extreme weather in climate datasets. *arXiv preprint arXiv:1605.01156*, 2016.

[36] Morteza Mardani, Noah Brenowitz, Yair Cohen, Jaideep Pathak, Chieh-Yu Chen, Cheng-Chin Liu, Arash Vahdat, Karthik Kashinath, Jan Kautz, and Mike Pritchard. Residual diffusion modeling for km-scale atmospheric downscaling. 2024.

[37] Fabio Merizzi, Andrea Asperti, and Stefano Colamonaco. Wind speed super-resolution and validation: from era5 to cerra via diffusion models. *Neural Computing and Applications*, 36(34):21899–21921, 2024.

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[39] Matthias Roth. Review of atmospheric turbulence over cities. *Quarterly Journal of the Royal Meteorological Society*, 126(564):941–990, 2000.

[40] Alex Rybchuk, Malik Hassanaly, Nicholas Hamilton, Paula Doubrawa, Mitchell J Fulton, and Luis A Martínez-Tossas. Ensemble flow reconstruction in the atmospheric boundary layer from spatially limited measurements through latent diffusion models. *Physics of Fluids*, 35(12), 2023.

[41] Alex Rybchuk, Luis A Martinez-Tossas, Nicholas Hamilton, Paula Doubrawa, Ganesh Vijayakumar, Malik Hassanaly, Michael B Kuhn, and Daniel S Zalkind. A baseline for ensemble-based, time-resolved inflow reconstruction for a single turbine using large-eddy simulations and latent diffusion models. In *Journal of Physics: Conference Series*, volume 2505, page 012018. IOP Publishing, 2023.

[42] Alex Rybchuk, Luis A Martínez-Tossas, Stefano Letizia, Nicholas Hamilton, Andy Scholbrock, Emina Maric, Daniel R Houck, Thomas G Herges, Nathaniel B de Velder, and Paula Doubrawa. Ensemble-based, large-eddy reconstruction of wind turbine inflow in a near-stationary atmospheric boundary layer through generative artificial intelligence. *arXiv preprint arXiv:2410.14024*, 2024.

[43] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):4713–4726, 2022.

[44] Vinothkumar Sekar, Qinghua Jiang, Chang Shu, and Boo Cheong Khoo. Fast flow field prediction over airfoils using deep learning approach. *Physics of Fluids*, 31(5), 2019.

[45] Dule Shu, Zijie Li, and Amir Barati Farimani. A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 478:111972, 2023.

[46] Joseph Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, 91(3):99–164, 1963.

[47] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[48] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252. PMLR, 2023.

[49] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021.

[50] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[51] Karen Stengel, Andrew Glaws, Dylan Hettinger, and Ryan N King. Adversarial super-resolution of climatological wind and solar data. *Proceedings of the National Academy of Sciences*, 117(29):16805–16815, 2020.

[52] Jian Sun, Zongben Xu, and Heung-Yeung Shum. Image super-resolution using gradient profile prior. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[53] Torsten Tritscher, Raanan Raz, Yoav Levi, Ilan Levy, David M Broday, et al. Emissions vs. turbulence and atmospheric stability: A study of their relative importance in determining air pollutant concentrations. *Science of the Total Environment*, 733:139300, 2020.

[54] Eva M Urbano, Konstantinos Kampouropoulos, and Luis Romeral. Energy crisis in europe: The european union' s objectives and countries' policy trends—new transition paths? *Energies*, 16(16):5957, 2023.

[55] Zhihao Wang, Jian Chen, and Steven CH Hoi. Deep learning for image super-resolution: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3365–3387, 2020.

[56] Zhuo Wang, Kun Luo, Dong Li, Junhua Tan, and Jianren Fan. Investigations of data-driven closure for subgrid-scale stress in large-eddy simulation. *Physics of Fluids*, 30(12), 2018.

[57] Jonathan A Weyn, Dale R Durran, and Rich Caruana. Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, 12(9):e2020MS002109, 2020.

[58] You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018.

[59] Wei Xu, Sen Jia, Zhuo-Xu Cui, Qingyong Zhu, Xin Liu, Dong Liang, and Jing Cheng. Joint image reconstruction and super-resolution for accelerated magnetic resonance imaging. *Bioengineering*, 10(9):1107, 2023.

[60] Xuan Xu, Yanfang Ye, and Xin Li. Joint demosaicing and super-resolution (jdsr): Network

design and perceptual optimization. *IEEE Transactions on Computational Imaging*, 6:968–980, 2020.

[61] Mustafa Z Yousif, Linqi Yu, and Hee-Chang Lim. High-fidelity reconstruction of turbulent flow from spatially limited data using enhanced super-resolution generative adversarial network. *Physics of Fluids*, 33(12), 2021.

[62] Liujie Zhang, Qiang Wang, Kun Luo, Xuanxuan Ming, and Jianren Fan. A novel residual-pyramid-attention super resolution model for mesoscale meteorological forecasting spatial downscaling. *International Journal of Green Energy*, 21(15):3458–3469, 2024.