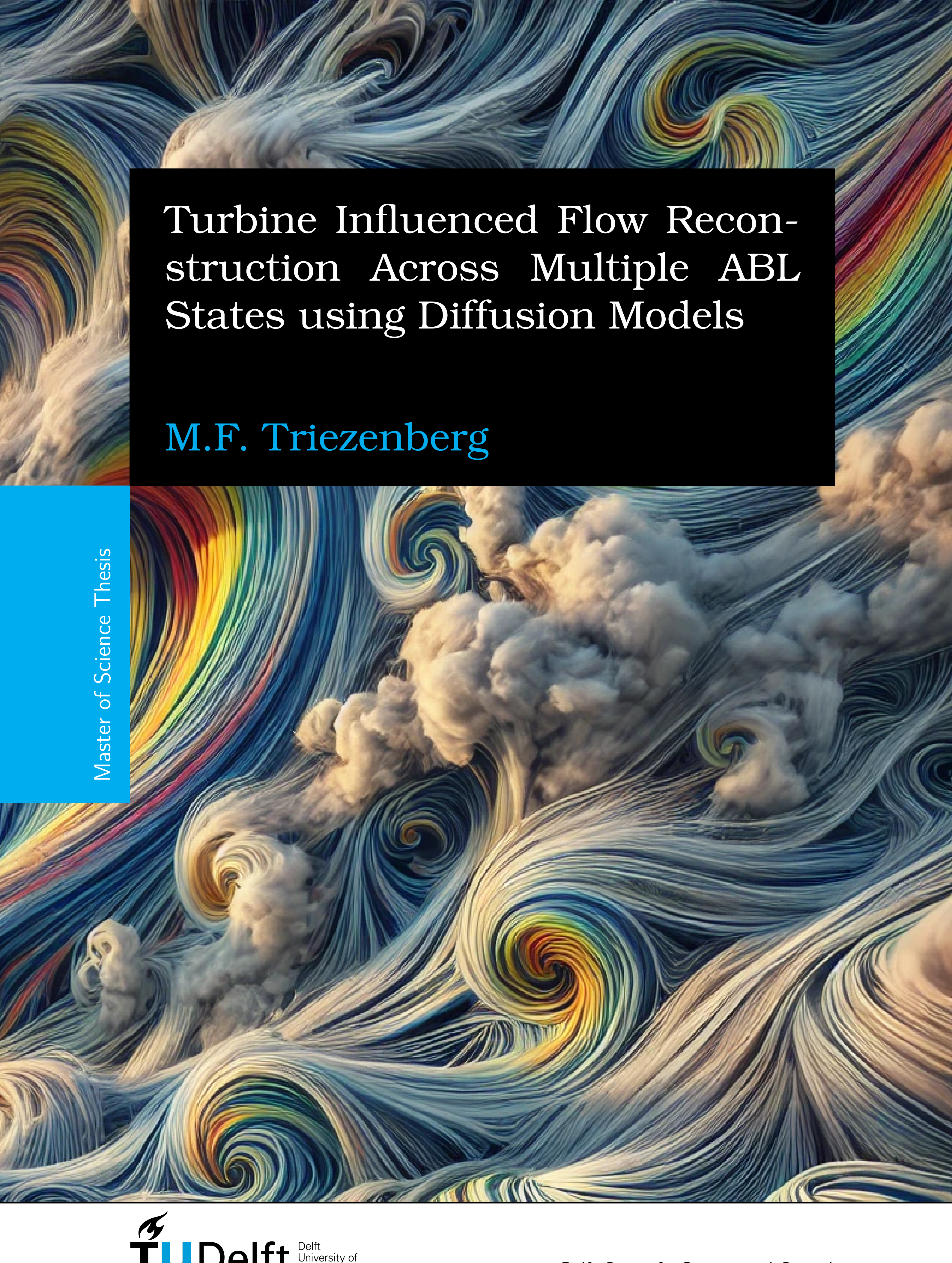

Turbine Influenced Flow Reconstruction Across Multiple ABL States using Diffusion Models

M.F. Triezenberg

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Turbine Influenced Flow Reconstruction Across Multiple ABL States using Diffusion Models

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

M.F. Triezenberg

August 12, 2025

Student number:	5130816	
Project duration:	September 2, 2024 – August 20, 2025	
Thesis supervisor:	Prof.dr.ir. J.W. van Wingerden	TU Delft
	dr. Ir. R.A. Verzijlbergh	TU Delft, Whiffle
Company supervisors:	dr. Ir. J. Maljaars	Whiffle, daily supervisor
External Members:	dr. J. Sun	TU Delft
	dr. Ir. M. Becker	TU Delft

Faculty of Mechanical Engineering (ME) · Delft University of Technology

The work in this thesis was supported by Whiffle BV. Their cooperation is hereby gratefully acknowledged.

Acknowledgements

Over the past year I have learned a lot about physics in the atmospheric boundary layer and generative models, and have thoroughly enjoyed the process. This phase of my studies would not have been the same without the people I worked with me along the way.

Firstly, I would like to extend my sincere gratitude to Jan-Willem van Wingerden for his guidance throughout my thesis. During our meetings he always came with great insights and good questions that helped me look at my research through a control engineering lens. Furthermore, I would like thank Jan-Willem for bringing me in contact with the company Whiffle, the company at which I conducted my thesis.

Secondly, I would like to thank Remco Verzijlbergh for our weekly meetings where he guided me through my thesis by asking critical questions and giving me great insights on the physics-side of my thesis. Furthermore, I would like to thank him for allowing me to do do my research at Whiffle.

Thirdly, I would also like to thank Jakob Maljaars for our bi-weekly one-on-one meetings where he was able to help me out with the planning of my thesis and understanding problems with my model.

I would also like to thank all the employees at Whiffle for their assistance and willingness to help me resolve issues throughout the project.

Finally, I would like to thank my family, friends, housemates and girlfriend for their invaluable support during this thesis.

I would also like to acknowledge the use of artificial intelligence tools to support the writing and editing of this thesis. These tools were used to improve clarity, structure, and formulation of the text under my full supervision and responsibility.

Delft, University of Technology M.F. Triezenberg
August 12, 2025

Abstract

As the energy transition accelerates, improving wind energy efficiency and forecasting becomes increasingly critical. One key challenge lies in reconstructing high-fidelity atmospheric boundary layer (ABL) flow fields from sparse measurements, especially in regions influenced by wind turbines. This thesis explores the use of Latent Diffusion Models (LDMs) to reconstruct physically plausible ABL flow fields from limited spatial data. Where previous work focused on homogeneous, neutral ABL states, this research extends the methodology to more diverse and realistic conditions by including both stable and neutral boundary layers with embedded wind farms. A conditional diffusion model is trained in the latent space of an autoencoder (AE), using both local measurements and global atmospheric labels. The model is evaluated using statistical, physical, and spectral metrics, and its ability to generalize across multiple ABL regimes is analyzed. Results show that the model can generate realistic reconstructions from extremely sparse inputs, including turbine wake structures, and can potentially serve as a tool for initializing Large Eddy Simulations (LES). These findings mark an important step toward integrating generative models into operational wind forecasting and control systems.

Table of Contents

Chapter 1

Introduction

As concerns about climate change continue to intensify, research and investments in renewable energy sources are also on the rise. To achieve net zero emissions by 2050, the International Energy Agency (IEA) [1] estimates that approximately 67% of global energy must come from renewable sources. Among these, wind energy is projected to provide the second largest share. However, the increasing reliance on renewable energy is already presenting challenges for grid stability, primarily due to the unpredictable and intermittent nature of these energy sources.

To increase the total energy production by wind turbines a lot of new wind parks are being built. However, space is a constraint [2][3][4] and there is also a lot to be gained from improving the efficiency of currently existing wind farms. Wind turbines operate in highly turbulent environments, and their performance depends on the ability to adjust to real time wind conditions. Current control methods often rely on limited local sensor data, which does not capture the full complexity of the wind field [5][6]. Getting more realistic wind fields allows for the development of online, high fidelity models that can drive closed-loop control systems. These systems dynamically adjust turbine settings, maximizing energy capture and minimizing mechanical stress. This leads to more efficient wind energy generation and extends the lifespan of turbines, which is not only sustainable but also economically appealing.

Improving grid stability is a crucial component of the energy transition[7]. One effective way to achieve this is by enhancing weather forecasting [8][9], which enables more accurate predictions of wind and solar energy production. High-quality forecasts support better planning for both energy generation and consumption, directly contributing to the shift toward cleaner energy systems. Achieving such forecasts requires a better understanding of the actual state of the atmosphere, particularly the complex and dynamic wind conditions in the Atmospheric Boundary Layer (ABL). By more accurately capturing these real-world flow patterns, forecasts can become more reliable and actionable.

Both these issues require better flow reconstruction to contribute to the energy transition more. A promising new method for doing this is the use of Latent Diffusion Model (LDM), a kind of machine learning used for image generation. Using sparse measurements available LDMs can reconstruct the flow around the sensors for turbulent situations, thus being able to generate realistic flow fields at a wind farm.

1-1 Turbulence and LES in the Atmospheric Boundary Layer

The ABL is the lowest part of the atmosphere, directly interacting with the Earth's surface, and plays a critical role in weather forecasting and wind energy applications. However, the dynamics within the ABL are complex due to turbulence chaotic, irregular flows with a wide range of interacting scales [10]. Turbulence in the ABL is often driven by interactions with terrain and obstacles, forming vortices of various sizes. These vortices undergo an energy cascade, transferring energy from large eddies to smaller ones until dissipated as heat at the Kolmogorov microscale [11].

jTurbulence is governed by the Navier-Stokes equations [12], which describe fluid motion and ensure conservation of mass and momentum. The nonlinear nature of these equations, particularly the convective term, makes them challenging to solve, especially in turbulent regimes where small changes in initial conditions can lead to vastly different outcomes. The wide range of scales involved in turbulent flows from large energy containing eddies to small dissipative ones further complicates their numerical resolution.

To address this, Large Eddy Simulation (LES) is often employed. LES strikes a balance between Direct Numerical Solution (DNS) [13], which fully resolves all turbulent scales but is computationally expensive, and Reynold's Averaged Navier-Stokes (RANS) [14], which models turbulence statistically. LES resolves the larger, energy containing eddies directly, while using a subgrid-scale (SGS) model, such as the *Smagorinsky model* [15], to approximate the effects of smaller eddies. This approach filters out the small-scale turbulence and models it using an eddy viscosity parameter to simulate energy transfer from large to small eddies.

The filtering operation in LES [16] separates large and small scales in the flow field, typically done via convolution, while the Navier-Stokes equations describe the dynamics of the resolved scales. The SGS model accounts for unresolved scales, introducing a turbulent eddy viscosity to represent the dissipative effects. This makes LES computationally efficient and well suited for simulating turbulent flows in the ABL, allowing for accurate predictions of large-scale eddy dynamics without the prohibitive costs of fully resolving all scales. However, as mentioned, LES can benefit from better initial conditions through extrapolating sparse measurements, something that DMs could help with.

1-2 Diffusion Models

Deep generative models have revolutionized image synthesis by generating high quality samples. Two prominent examples, General Adversarial Network (GANS) [17] and Variational AutoEncoder (VAE) [18], have achieved remarkable success. Recently, however, Diffusion Models (DMs) [19] have gathered significant attention, demonstrating performance on par with GANs and VAEs while offering distinct advantages. Unlike GANs, which rely on adversarial training and are often unstable, DMs are more stable due to their likelihood based training process. Similarly, DMs are able to model complex data distributions more effectively than VAEs. These characteristics make DMs an attractive choice for image synthesis tasks.

Diffusion Models, specifically Denoising Diffusion Probabilistic Models (DDPMs), are a class of generative models that transform a simple initial distribution, like a Gaussian, into a complex target distribution using a diffusion process. This process involves gradually adding noise to the input data, and then learning to reverse this noise, step by step, using a neural network. The ability to effectively model this denoising process is key to the success of DMs.

1-3 Flow Reconstruction Using Diffusion Models

Papers by J.Bastek et. al. [20] and D.Shum et. al. [21] first introduced using DMs to do flow reconstruction trained on data generated by LES models. These papers did not focus on flows in the ABL but on a smaller scale. They showed the potential of these models to reconstruct turbulent flows and laid the foundation for further research.

Flow reconstruction in the ABL using DMs was then first studied by A. Rybchuk et al. [22], who employed data from the Rotor Aerodynamics Aeroelastics and Wakes (RAAW) field campaign [23]. This campaign collected wind flow measurements around a wind turbine using a combination of in-situ meteorological tower sensors, ground based lidars, and nacelle mounted lidars. In their approach, a LDM was used to extrapolate sparse measurements to large three-dimensional regions of the ABL. While their method achieved promising results, the training and evaluation were conducted on a highly homogeneous dataset namely, a single ABL state simulation under periodic boundary conditions. In such a setting, flow variability is minimal, and the spatial characteristics of the data are nearly uniform across samples. Consequently, even unconditional sampling tends to produce realistic reconstructions, removing the need of meaningful conditioning.

However, for diffusion based flow reconstruction methods to be viable in real world scenarios, the model must be able to generalize to a much broader range of ABL states, including stable, unstable, and transitional atmospheric conditions. This calls for effective conditioning strategies to guide the generative process toward physically plausible samples that are consistent with sparse observations and global measurements. Furthermore, current works only focused on reconstructing the three velocity components u, v and w while some LES models also require the most-static-energy temperature T_{hl} and specific humidity q_t as initial conditions. So in order to use a DM to do flow reconstruction to use as initial conditions for LES the model has to be able to reconstruct these variables too. This research aims to address these needs by investigating conditioning techniques for both spatially sparse measurements and global state variables while trying to reconstruct the five variables u, v, w, T_{hl} and q_t.

1-4 Research Gap and Research Question

In recent years, DMs have shown significant promise in reconstructing complex flow fields. These models are able to generate high fidelity representations of turbulent flow fields. Rybchuk et al. [22] already showed good performance in reconstructing the ABL. They did this however, without any turbines and only for a Neutral Boundary Layer. Even though their results are promising, there is a lot of improvement needed before real-life application, for e.g. wind farm control or wind resource assessment, is possible. Therefore this research proposes adding a global conditioning to the model enabling it to distinguish between different ABL states. Furthermore, it will also be shown that DMs can reconstruct flow fields that contain wind turbines. These proposals lead to the following research question:

"How can a diffusion model reconstruct turbine influenced flow fields across multiple ABL states?"

To answer this question the model will be fed with different types of global information to inform it about the global state of the ABL. The importance of this additional information will be tested by comparing it to model outputs without this information. Lastly, as has been mentioned, it would be useful to use these reconstructed flow fields as initial conditions for LES. Commonly LES simulations do not only need the velocity components u, v and w but also the moist-static-energy temperature T_{hl} and the specific humidity q_t. This study will also investigate whether a DM can reconstruct these variables too, something that is not yet done in this context.

1-5 Report Structure

The structure of this thesis is as follows: First, Chapter 2 introduces the relevant atmospheric dynamics and provides a theoretical background on LDMs, including AEs, DMs, and guidance mechanisms. Second, Chapter 3 presents the model implementation and training strategy, covering both the AE and the conditional DM. Third, Chapter 4 describes the experimental setup, detailing data generation using LES, preprocessing, measurement design, and hardware specifications. Fourth, Chapter 5 presents the results, including visual and quantitative analyses for various ABL states, and evaluates model performance across different conditions. Finally, Chapter 6 concludes the thesis and offers recommendations for future work.

Chapter 2

Atmospheric Flow Reconstruction using Diffusion Models

Now that the research gap and research question have been established, we turn our attention to maximizing model performance. This involves examining both the fundamentals of atmospheric dynamics and the underlying principles of the model architecture. The goal is to identify ways to optimize the model's design and training procedure to better align with the objectives of this research.

In addition, since in previous work by Rybchuk et al. [22] no explicit guidance was necessary, this chapter also explores alternative guiding mechanisms that may still prove beneficial for more complex or sparse input scenarios.

The structure of this chapter is as follows. Section 2-1 provides a brief overview of relevant ABL dynamics. Section 2-2 introduces the LDM, beginning with its overall structure and followed by foundational explanations of autoencoders and diffusion models. Finally, Section 2-3 discusses various methods of conditioning or guiding the model toward specific outputs.

2-1 Atmospheric Dynamics

As mentioned before, the ABL is the lowest part of the atmosphere that directly interacts with the Earth's surface. The ABL can be in multiple different states that greatly influence the dynamics of the flow in the ABL. These states can be categorized into three main states according to atmospheric stability, namely: Neutral Boundary Layer (NBL), Stable Boundary Layer (SBL) and a Convective Boundary Layer (CBL). These states differ mainly in turbulence characteristics. The SBL occurs mainly at night when surface cooling leads to stratification, which suppresses turbulence. The CBL, on the other hand, occurs during day time when, due to surface heating, there is a lot of buoyant turbulence and vertical mixing. Lastly, the NBL is an intermediate state where there is little to no surface heating or cooling. Figure 2-1 shows the flow in the three different ABL states, clearly indicating the difference

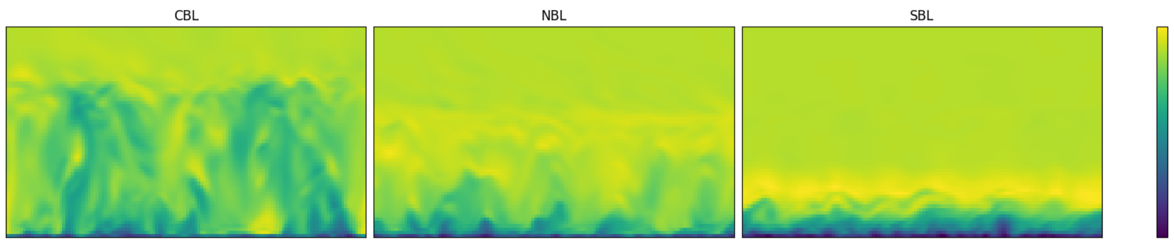

Figure 2-1: Figure showing a side view of the wind in x-direction for the three ABL states: CBL, NBL and SBL.

in turbulence among the different states. The amount of turbulence, and thus to some extent the ABL state, can be quantified using the Bulk Richardson number R_b, which is a measure for the dynamic stability and turbulence in the ABL.

2-1-1 Bulk Richardson number

The Bulk Richardson number is a simplified version of the Richardson number and both are parameters used to quantify the balance between the buoyancy and shear in a flow [24]. The Bulk Richardson number is often used over the Richardson number because the Bulk Richardson number uses differences in flow over a finite distance while the Richardson number requires high-resolution vertical profiles to calculate gradients. Since these high-resolution profiles are often not available due to sparse measurements the Bulk Richardson number is often used. It is a dimensionless number calculated using the following formula:

$$R_b = \frac{g}{\theta_v} \frac{\Delta\theta_v \Delta z}{(\Delta U)^2 + (\Delta V)^2} \tag{2-1}$$

Where:

- g is the gravitational acceleration.
- θ_v is the reference virtual potential temperature.
- $\Delta\theta_v$ is the difference in virtual potential temperature over the layer.
- Δz is the height of the layer.
- ΔU and ΔV are the differences in the wind speeds u and v respectively.

So the R_b is computed by calculating the difference in wind speeds and potential temperature at two different heights. The obtained value of R_b provides information on ABL stability:

- When Rb<0 the ABL is convective.
- When 0<Rb<0.25 the ABL is turbulent and shear driven.
- When 0.25<Rb<1.0 the ABL is intermittently turbulent.
- When Rb>1.0 the ABL is stable

2-1-2 Boundary Layer Height

Another important detail of the ABL is the Boundary Layer Height (BLH) or the height of the inversion layer. As is visible in figure 2-1, there is a clear height at which the turbulent flow in the ABL stops and the flow becomes more laminar. The BLH is the altitude up to which turbulent mixing occurs. As shown in the figure, lower turbulence levels in the SBL lead to a lower BLH compared to the CBL. While the flow beneath the boundary layer is turbulent due to the interactions with the Earth's surface this transitions to more stable, stratified conditions above the boundary layer.

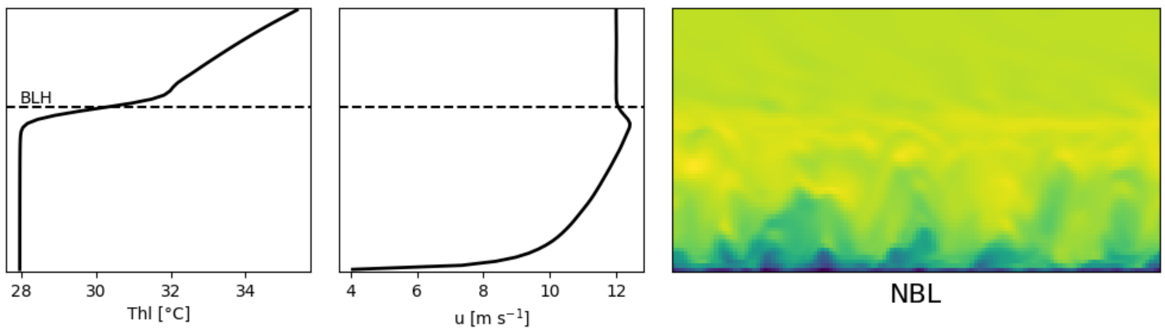

Figure 2-2: Figure showing the horizontally averaged profiles for T_{hl} and u and the Boundary Layer Height in these profiles.

The BLH is an important parameter for understanding to what extent the Earth's surface interactions influences the flow in the ABL. In literature there is no generally accepted way of calcuting the BLH, however G.Rampanelli and D. Zardi [25] proposed using the water liquid potential temperature for this. This is possible because the inversion layer always exists because there is a sudden jump in potential temperature at a certain height. This sudden jump prevents the air from mixing, blocking turbulence and creating the stratified conditions that we mentioned earlier. Figure 2-2 outlines this phenomenon by showing that the height of the BLH is the height at which the vertical profile of T_{hl} has a jump, which is at the same height where the turbulence disappears in the right sub-figure. This figure also shows the vertical profile for the wind speed u.

T_{hl}, also called moist-static-energy temperature [26], is an important variable for some LES simulations because it can function as a state variable for some LES. State variables for LES form the minimal set of variables that define the system's physical state.

$$T_{hl} = T + \frac{gz}{c_p} - \frac{L_v q}{c_p} - \frac{L_i q_i}{c_p} \tag{2-2}$$

Where:

- T is the parcel temperature.
- z is the height above ground level.
- c_p is the specific heat of dry air at constant pressure.

- g is the gravitational acceleration.
- L_v is the latent heat of vaporization.
- q is the water vapor specific humidity.
- L_i is the latent heat of fusion .
- q_i is the specific humidity of ice.

The variable T_{hl} is a conserved variable for moist adiabatic processes such as rising air parcels and cloud formation. It represents the moist static energy per unit heat capacity and due to its conservation it is used as a state variable for some LES simulations.

2-2 Latent Diffusion Models

2-2-1 General Structure

DMs are a state-of-the-art class of generative machine learning models, but they are computationally expensive due to the large number of denoising steps required in the reverse process. This cost becomes particularly significant for high-dimensional 3D domains representing, for example, a wind farm. To reduce the overall computational burden, an LDM is used.

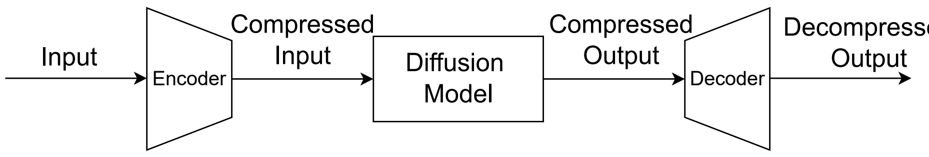

Figure 2-3: The basic structure of the LDM.

An LDM places the diffusion model in the latent space of an AE, a type of Convolutional Neural Network (CNN) that learns to compress input data into a lower-dimensional representation and then reconstructs it. The AE consists of two parts: an encoder that compresses the input, and a decoder that reconstructs it. Because the latent space is a compressed version of the original data, running the DM in this space significantly reduces memory and compute requirements while still capturing important structures in the data. Figure 2-3 shows an overview of the LDM.

Rombach et al.[27] showed that LDMs can achieve high quality results with greatly reduced computational cost. Unlike earlier latent diffusion approaches such as by Vahdat et al.[28], they propose first training the AE separately, followed by the diffusion model. Their experiments found that compressing the data by a factor of 4 to 8 provides a good trade-off between efficiency and reconstruction quality, even improving performance metrics like the Frechét Inception Distance (FID) score [29]. So, following the method proposed by Rombach et al. first the AE will be trained, followed by training the DM with the goal of extrapolating partial measurement into a full flow field. The pipeline for this is shown in figure 2-4. In this figure as input you can see an image that is mostly white except for an area in the lower right corner. This image is called a mask, and the non white values represent measurements done at a certain location. Since the goal of this research is to extrapolate these measurements the model has to be able to get these masks as input and fill in the white area around it, this is called image inpainting. There are different methods for doing this, which will be discussed in section 2-3-1.

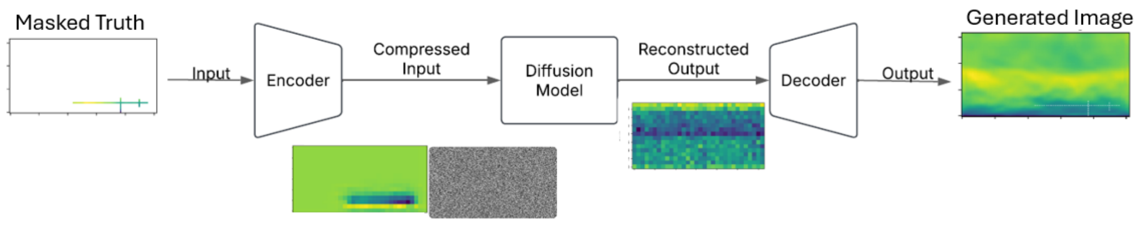

Figure 2-4: Data flow for inference of a Latent Diffusion Model doing flow reconstruction.

As the caption of 2-4 states this is the data flow for a Latent Diffusion Model doing flow reconstruction. This figure shows that such a model can get measurements, for a certain variable, in the form of a mask which are then compressed and fed to the DM. The DM reconstructs the flow for this variable using the measurements, this reconstruction is then decompressed and the final result is obtained. This approach can be applied to multiple variables, which are stored as separate channels within a single 4D data array. For example, when reconstructing five variables, the data would be organized with the shape $[5, X, Y, Z]$, where the first dimension represents the number of variables (channels), and X, Y, and Z correspond to the spatial grid dimensions.

2-2-2 Autoencoder

This section shortly touches on the basics of an AE, explains how residual blocks work and what their use are and mentions possible loss functions for the AE.

AE Basics

AEs are a type of CNN [30] designed to reconstruct their own inputs. An AE consists of two main components: an *encoder* and a *decoder*. The encoder transforms the input data $\mathbf{x}$ into a lower-dimensional latent representation $\mathbf{h}$ via the mapping

$$\mathbf{h} = f(\mathbf{x})$$

The decoder then attempts to reconstruct the original input from the latent code through the reverse mapping

$$\mathbf{r} = g(\mathbf{h})$$

The encoder acts as a feature extractor: it compresses the high-dimensional input into a latent representation that captures the most important structures and patterns. In the context of this work, the encoder maps three-dimensional atmospheric flow fields into a compact and informative latent space that retains key turbulent features. The decoder then reconstructs the input from this representation, ideally preserving all relevant flow information.

If no constraints were placed on the model, the AE could learn to simply invert the encoder through the decoder, achieving perfect reconstructions without learning meaningful structure in the latent space. To prevent this, AEs are designed with architectural or loss based constraints that force the model to learn only the most informative features of the data in order to reconstruct it. This property makes the AE especially valuable for this research, as it ensures that the latent space captures the most essential information from the input flow fields. This compact representation is then used by the diffusion model for guided sample generation.

Residual Block

When building deeper convolutional AEs one can encounter an issue called 'vanishing gradients', making them more unstable during training. Residual blocks were first introduced by He et al. [31], to prevent this phenomenon.

The idea of a residual block is that, instead of letting a layer learn an entire function from scratch, a residual block learns the difference between an input and the desired output. The input is then added back to the output of the block, stabilizing training. A residual block can be defined using the following formula:

$$y = \mathcal{F}(x, \{W_i\}) + x \tag{2-3}$$

where:

- x is the input feature tensor.
- $\mathcal{F}(x, \{W_i\})$ is a learned function, typically a series of convolutional layers with activation functions and possibly normalization layers. W_i stands for the learnable parameters in the layer.
- y is the output of the residual block, representing the sum of the original input and the learned residual.

Loss Function

Choosing the correct loss function is one of the most important parts of effectively training an AE. There are a lot of different losses available that can be combined to create a powerful loss function that optimizes the objective for your AE. This section will discuss multiple losses used in AEs and why they could be a useful addition to the loss function of this particular AE.

Negative Log-Likelihood

The NLL loss or Negative Log-Likelihood loss is a measure of how well the AE predicts the observed data [32]. Minimizing the NLL loss is equal to maximizing the likelihood of observing your data under the assumed probabilistic model. In this case the NLL loss is calculated by taking the reconstruction, done by the AE, as a probabilistic prediction, aiming to maximize the likelihood that your generated samples are similar to data it has been trained on. The NLL loss is calculated as follows:

$$\mathcal{L}_{NLL} = \frac{1}{2}\left[\frac{(\mathbf{x}-\hat{\mathbf{x}})^2}{\sigma^2} + \log(\sigma^2)\right] \tag{2-4}$$

Where:

- $(\mathbf{x}-\hat{\mathbf{x}})^2$ is the absolute reconstruction loss, which is the difference between the input $\mathbf{x}$ and the reconstruction $\hat{\mathbf{x}}$.
- σ^2 is the variance in the reconstruction.
- $\log(\sigma^2)$ is the logarithmic of the predicted variance. It penalizes the model for making σ^2 too large thus controlling its confidence.

Minimizing the NLL loss preserves fine-scale features in the flow field, which could be essential for the reconstruction of turbulence. Furthermore, by modeling uncertainty the model can increase its robustness against noisy or sparse inputs, which are often present in real world applications.

Kullback-Leibler Divergence

The KL loss or Kullback-Leibler divergence loss which is a measure for the distance between two distributions [33]. In the case of an AE the distributions being compared are a predefined latent prior $p(z)$ and the encoder generated latent distribution $q(z|x)$. In this case $p(z)$ is a Gaussian with mean 0 and variance 1 and $q(z|x)$ is the posterior distribution of the latent z given input x. The distance between these two distributions is measured in the following way:

$$D_{\mathrm{KL}}(q(z|x)\,\|\,p(z)) = \int q(z|x)\log\left(\frac{q(z|x)}{p(z)}\right)dz \tag{2-5}$$

However since $p(z)$ used here is a Gaussian this can be rewritten into the following closed form.

$$\mathcal{L}_{\mathrm{KL}} = \frac{1}{2}\sum_{i=1}^{d}\left(\mu_i^2 + \sigma_i^2 - \log\sigma_i^2 - 1\right) \tag{2-6}$$

An AE with a KL loss is called a Variational AE (VAE), first introduced by Kingma and Welling [34], and its purpose is to push the latent representations $q(z|x)$ towards the prior distribution $p(z)$. This prior distribution has a mean μ and variance σ pushing the distribution of the latent space to that of a Gaussian. A well structured latent space is easier to learn for the DM than when another distribution is used. Another reason that this is useful is because it makes the latent representations more continuous allowing for smooth interpolation and thus the generation of realistic new samples. Even though this AE will not be used to generate samples itself this is still a useful feature since a regularized latent space improves a decoder's ability to reconstruct inputs it has not explicitly seen yet. This is a necessary feature because the DM is unlikely to generate samples that the AE have been trained on, using a VAE the model will be able to interpolate between samples it has seen and the ones generated by the DM. Furthermore, the probabilistic nature of a VAE increases robustness against noisy samples, helpful in the case of real-life measurements.

Generator Loss

The Generator loss ($\mathcal{L}_G$) is a loss that is mostly used in GANs. This loss uses a discriminator that aims to improve the reconstruction quality. In this case the decoder is the generator that has to fool the discriminator by producing realistic reconstructions. The loss function for the generator loss looks as follows:

$$\mathcal{L}_G = -\mathbb{E}\left[Discriminator(\hat{x})\right] \tag{2-7}$$

The discriminator is a separate model that is trained on classifying inputs as either real or fake. By learning this it is possible to let the outputs of the AE be classified by the discriminator and try to fool it into thinking they are inputs from the dataset. The loss is negative because the goal is to maximize the discriminator's belief that the fake data is real. By doing this, the model is encouraged to generate physically possible turbulent structures. Furthermore, while just using a NLL loss might lead to blurred reconstruction, the combination with the generator loss promotes sharper output.

2-2-3 Diffusion Models

This section will look into the mathematical background of DMs. this Section will discuss the fundamentals of DMs, followed by an explanation of attention, an essential feature for the model.

DM Basics

Deep generative models, such as General Adversarial Network (GANS) [17] and Variational AutoEncoder (VAE) [18], have achieved notable success in image synthesis. However, Diffusion Model (DM) have recently emerged as a competitive alternative, demonstrating high-quality sample generation. DMs offer several advantages: they are more stable than GANs due to their likelihood based training and do not rely on restrictive Gaussian priors like VAEs, allowing them to model more complex data distributions.

Denosing Diffusion Probabilistic Model (DDPM) [35][19][36][37], a popular class of DMs, use a diffusion process to transform a simple distribution (e.g., Gaussian noise) into a complex data distribution. This process consists of two main phases: a forward process, which adds noise to data over time, and a reverse process, where a neural network learns to denoise the data and generate samples.

Forward Process

In the forward process, the input data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ is gradually corrupted by adding Gaussian noise through a series of steps [19]. Each step $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ adds noise proportional to a variance schedule β_t, resulting in pure noise $\mathbf{x}_T$ at the final step. This forward process can be expressed as a *Markov chain*, and allows for direct sampling at any time step using the following equation:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{2-8}$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2-9}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ and $\alpha_t = 1 - \beta_t$. Since β_t is proportional to the amount of noise added to the image at each step, $\bar{\alpha}_t$ can be seen as the amount of information in the image still left at a certain time step.

The goal of the forward process is turning the original image $\mathbf{x}_0$ into pure noise at $\mathbf{x}_T$ by adding noise through the noise schedule. This noise schedule was originally chosen as a linear schedule but later it was found that a cosine schedule yields better results in most cases. Figure 2-5 shows how $\bar{\alpha}_t$ decays for each noise schedule. Under the linear schedule, $\bar{\alpha}_t$ drops to near zero quickly, so the original image signal is overwhelmed by noise after only a small fraction of the diffusion steps. The cosine schedule, by contrast, keeps $\bar{\alpha}_t$ higher for longer, preserving more of the image content throughout the forward process. Because the model is trained (and later sampled) on inputs that still contain useful structure, it can learn a better denoising function, which translates into higher quality reconstructions at inference time.

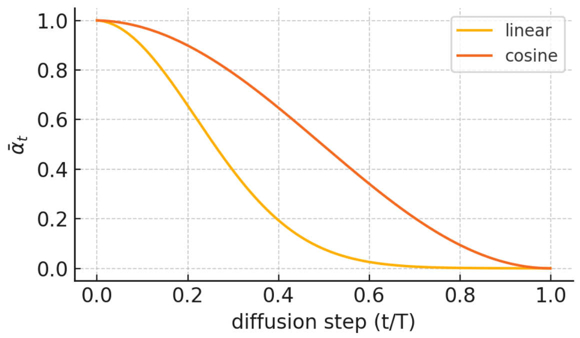

Figure 2-5: Figure showing a linear and cosine variant of the noise schedule.

Reverse Process

The reverse process aims to recover the original data from noise, step by step. Since the true reverse transitions $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ are unknown, a neural network is trained to approximate them. The reverse process is modeled as a Gaussian distribution:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{2-10}$$

where $\mu_\theta(\mathbf{x}_t, t)$ is learned by the neural network, and $\mathbf{\Sigma}_\theta(\mathbf{x}_t, t)$ is often simplified to a fixed schedule [19]. The neural network architecture that is used for this is a U-net [38].

Training

The neural network is trained by minimizing a *variational lower bound (VLB)* on the log-likelihood of the data [39]. This is achieved by minimizing the *Kullback-Leibler (KL) divergence* between the learned reverse distribution, $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, and the true posterior distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$. The KL divergence is a distance measure of how much a distribution differs from another one. Minimizing this distance is done by minimizing the distance between the means of both distributions. Which, in our case leads to the following objective function.

$$L = \mathbb{E}\left[||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] \tag{2-11}$$

Here $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$ is a combination of $\mathbf{x}_0$ and $\mathbf{x}_t$ dependent on the variance β_t. After this, [36] found that training $\mu_\theta(\mathbf{x}_t, t)$ to predict the noise ϵ at any step t yields better results. This leads to simplifying the objective function to [35]:

$$L_{simple}(\theta) = \mathbb{E}_{t,x_0,\epsilon}\left[\left|\left|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right|\right|^2\right] \tag{2-12}$$

In this new objective, the model learns to predict the added noise ϵ by adjusting its parameters so that $\epsilon_\theta(\mathbf{x}_t, t)$ approximates the true noise at each timestep t. When $\epsilon_\theta(\mathbf{x}_t, t)$ has been properly trained $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ can be sampled by computing $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$ where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$.

Attention Mechanism

Since the DM has to be able to extrapolate sparse measurements to the entire domain the DM has to learn the relation between those measurements and the flow in other regions of the domain. Attention, first introduced by Vaswani et al. [40], was made to learn the relations between different words in a sentence. Even though no words are used in this model attention's ability to relate different values in an input to each other is a great feature that can help the DM learn relations between measured and unmeasured areas.

Attention mechanisms take three matrices as input: $\mathbf{Q}$ (queries), $\mathbf{K}$ (keys) and $\mathbf{V}$ (values) each representing different linear transformations of the input features.

- $\mathbf{Q} \in \mathbb{R}^{m \times d_k}$Queries represent the data points for which we want to compute attention scores. In the case of our flow reconstruction this represents locations in the domain, especially where little to no measurements are available. The dimension m corresponds to the number of query positions (typically all spatial locations in the domain), and d_k is the dimensionality of the feature representation.

- $\mathbf{K} \in \mathbb{R}^{n \times d_k}$ Keys also represent different data points in the domain that might contain information relevant to the query locations. Keys enable the model to identify regions with measurements that influence unmeasured areas. Here, n denotes the number of key positions (e.g., the locations where measurements are available), and d_k is again the feature dimension, shared with the queries.

- $\mathbf{V} \in \mathbb{R}^{n \times d_v}$ Values represent the latent-space flow features at the key locations, which will be weighted based on their attention scores to reconstruct missing data. The n entries correspond to the same positions as in $\mathbf{K}$, and d_v is the dimensionality of the value vectors, which are combined to produce the output.

The attention score between each query and key is computed by computing the dot product, scaled by the factor $\sqrt{d_k}$ where d_k is the dimensionality of the keys and queries. This is done to prevent the dot product from exploding, stabilizing gradients during training. The softmax function is a function that maps an input vector to a probability distribution making sure all outputs are between 0 and 1 and sum up to 1. So for this use case the softmax function makes the relation between keys and queries more interpretable by mapping them to a relative importance.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{2-13}$$

The output of an attention layer is a weighted sum of the values, where the weights are derived from the normalized attention scores. A high attention score indicates a strong relation between two points, telling the model that a measurement in one location is strongly correlated to an unmeasured region giving the model essential information on how to inpaint images. This makes attention mechanisms especially useful in models for flow reconstruction using sparse measurements, such as the diffusion models discussed in this work.

2-3 Guidance Mechanisms

The previous sections stated how the overall LDM was built and set up to maximize performance during inference. As has been noted in section 1-3, since the data set used by Rybchuk et al. [22] was so homogeneous no real guidance was needed to end up with a sample that looked like the ground truth. Since the goal of this research is to show that it is also possible to do image inpainting for more difficult, variable datasets it is important to also look into good guidance mechanisms. This section will give an overview of the research that has been done to look into these mechanisms. This research has been split up into two parts, guiding the model by using sparse measurements, which is known as inpainting and will be discussed in section 2-3-1. Furthermore, ways of supplying the model with more global information about, for example, the state of the ABL have been investigated and will be discussed in section 2-3-2. The combination of these two guidance mechanisms should allow the model to infer the right ABL state through conditioning and reconstruct the right wakes through the sparse measurements.

2-3-1 Image Inpainting with Diffusion Models

The goal of this research is to train an LDM that uses sparse, real-life measurements in order to extrapolate them to an entire flow field. The technique of giving partial information to a generative model and making the model fill in the unknown values is called image inpainting. There are different ways of supplying the model with these measurements and they will be discussed below.

The Inpainting Task

In image inpainting a DM has to learn the relationship between known and unknown areas. These known areas can range from small patches to larger regions. Inputs that only contain partial information are called masks, masks consist of zeros for unknown pixels (masked pixels) and nonzero for the known pixels. In general diffusion based inpainting, the generative process is guided to only reconstruct the masked parts, while the unmasked parts are superimposed. Formally, given an image $\mathbf{x}$ and a binary mask $\mathbf{m}$, where $m_{i,j} = 1$ indicates observed pixels and $m_{i,j} = 0$ indicates missing ones, the goal is to sample $\tilde{\mathbf{x}} \sim p(\mathbf{x} \mid \mathbf{x} \odot \mathbf{m})$.

In the literature, two major methods exist for diffusion based inpainting: *preconditioned* and *postconditioned* inpainting.

Preconditioned Inpainting involves training a conditional diffusion model to learn a distribution p(x|y), where y represents conditioning information such as observed pixels or a mask. During training, the model is explicitly exposed to various conditional scenarios, allowing for fast inference later. However, this method requires significant computational resources and time during training to generalize across all possible mask configurations [27].

Postconditioned Inpainting, on the other hand, does not require a trained conditional model. Instead, an *unconditional* diffusion model is used as a prior during inference. One approach is *RePaint* [41], which works by taking a jump back in time every few steps to end up with a better result. It does this in order to remove weird artifacts that appear when just superimposing the mask every step.

How it works is when you start with an image that is full noise at timestep T, the masked values are superimposed and denoising happens for a few steps until you arrive at time t, where t < T. Then the sample generated at time t gets a little bit of noise added to it again so it has noise equal to timestep t+3 and then you start denoising the image again. The idea behind this method is that by taking some extra steps the model incorporates the masks better without the artifacts.

Alternative Methods such as the propagation based approach, proposed by Corneanu et al. [42], aim to reduce both training and inference costs. Their method uses a small trainable function that learns to share information between observed and unobserved regions. This function learns to propagate information from known to unknown regions, allowing an unconditional model to be used for inpainting. Although it avoids repeated sampling, the authors report that their inference time is still approximately 5 times longer than that of a preconditioned model.

Method Selection While all these methods achieve comparable reconstruction quality, their computational characteristics differ. For practical applications where fast inference is critical, such as real-time flow field estimation, preconditioned models are preferred. Although training is more expensive, it is a one time cost, whereas slow inference can be a significant bottleneck in operational settings. Based on this trade-off, the preconditioned approach is adopted in this research.

2-3-2 Mask-Guided Conditioning

The goal of adding a mask is to guide the model towards the state of the ABL that was measured in real-life. This has to be done by supplying the model with essential information in the best way possible. In this section the different ways of supplying masks to the model will be discussed as they have been mentioned in literature.

Input Concatenation One of the simplest approaches is to concatenate the masks to the input tensor as additional channels. This is done by concatenating the masked inputs as well as the binary mask, providing the model with spatial information about which regions are known and which are missing. This method is computationally inexpensive and widely used in earlier inpainting frameworks such as [43].

Noise Resampling Another strategy, often used in DDPM based inpainting, is to apply noise only to the masked regions during training and sampling, while keeping the unmasked parts unchanged. This encourages the model to focus denoising efforts solely on the unknown regions. This approach was demonstrated in works like [44]. This method can be applied to both pre- and postconditioned models but mostly used in postconditioned models since in preconditioned models the mask information should already be learned during inference.

Cross-Attention Conditioning More advanced models employ cross-attention mechanisms to incorporate mask information. Here, the mask (or a learned embedding of the observed data) serves as context that the model attends to during the denoising process. This strategy is often used in text-to-image generation (e.g., [27]) and has been adapted for spatially aware inpainting and flow reconstruction tasks (e.g., [45, 46]).

Because in their study Rybchuk et. al. [47] have found that hard constraining the mask, as is done for noise resampling, leads to artifacts in the reconstructed field this option will not be chosen. Furthermore, Cross-Atttention requires another model that can map the masks into an embedding which increases model complexity. Because of this the conditioning method used in this research will be concatenation of masks.

Class-Conditional Diffusion

In class-conditional diffusion models a label is provided as input to the model in order to guide the generative process toward a certain class. Labels are numbers that represent a certain class in the data on which the model is trained. These numbers can represent categorical classes such as 'cat' and 'dog' or have a more continuous and physical meaning such as time. These categorical labels are then given numbers like 1 or 2, but feeding these numbers to the model is not done in practice. This is avoided because models struggle to relate data to such labels because it is looking for a mathematical relation between the labels, so label 2 is twice label 1. However, in the real world the labels stand for cat and dog, and cat is not twice dog, so the model would have a hard time learning this relationship. In contrast, continuous labels do exhibit numerical relationships (2 minutes is twice as much as one minute) so using them can be appropriate. To tackle the issue for categorical labels, label embedding is used, which is a way of breaking a label, a single number, down into a vector which contains multiple numbers. This vector is then concatenated to the input data to condition the generation steps on the label's information.

There are multiple ways of embedding a label, but one often used is sinusoidal embedding, as done by [48] which called it Positional Encoding (PE). In a sinusoidal embedding, sinusoidal functions are used at different frequencies to map an integer to a higher dimensional space. As mentioned before, the integer is mapped to a vector, which is done by using a sine and cosine function for the even and odd positions in the vector respectively.

$$PE(i)_{2k} = \sin(\frac{i}{10000^{2k/d}}), PE(i)_{2k+1} = \cos(\frac{i}{10000^{2k/d}}) \tag{2-14}$$

The equation above shows how the embedded vector is filled. In this formula, d is the dimension of the vector, k is the index of the vector going from 0 to $\frac{d}{2} - 1$, i is the integer to be embedded and 10000 is an arbitrary but intentionally chosen large variable. the dimension d can be chosen by the user which is a trade-off between memory and expressiveness of the label. The choice for a combination of sine and cosine has been made because they change continuously with the input meaning that a slight change in inputs shifts the embedding in a predictable way, helping the model extra- or interpolate to values not seen during training. Looking at the argument of both functions it can be seen that as k increases the denominator increases, and thus the argument becomes smaller leading to a lower frequency. This change in frequency is important for multiple reasons, as is shown in [49]. Firstly, because the big range of frequencies ensures uniqueness across all inputs. Secondly, high frequencies help the model distinguish between inputs that are close to each other, while the low frequencies help the model grasp the difference between inputs that are far apart. The need for this big range of frequencies is also the reason a large number such as 10000 is often chosen.

Chapter 3

Model Implementation and Training Strategy

This chapter outlines the practical implementation of the deep learning models used in this study, focusing on both architectural design and training strategy. Building on the theoretical foundations laid out in earlier chapters, we describe how the AE and DM were constructed and optimized to enable high fidelity flow field reconstruction from sparse measurements.

Section 3-1 details the architecture and training of the autoencoder, which serves as a crucial component for compressing the high-dimensional flow data into a lower-dimensional latent representation. This dimensionality reduction enables efficient training of the more computationally intensive diffusion model.

In the section 3-2, we present the design of the diffusion model, which operates in the latent space produced by the autoencoder. The DM is conditioned on sparse input measurements and is responsible for reconstructing the full flow field. Attention is given to architectural trade-offs, loss functions, and optimization strategies used to ensure stable and effective training.

Together, these components form the core of the end-to-end system used for flow reconstruction, balancing interpretability, computational cost, and model accuracy.

3-1 Autoencoder

This section outlines the choices made for the AE's architecture and training strategy, based on the theoretical background provided in Section 2-2-2. Section 3-1-1 describes the final network architecture, while Section 3-1-2 presents the corresponding training configuration.

3-1-1 Architecture

As mentioned in chapter 2-2-1 the reason for using the AE is compressing input data to lower the computational cost of the DM. From section 2-2-2 we know that AEs are perfect for this since they are built to extract the most important details from input data into the latent space. In order to do this as effectively as possible, using the information from section 2-2-2, architectural choices were made which will be discussed in this section.

Input, Output and Latent Space

The most important factors for choosing the architecture of a model are the desired dimensions of the input, output and latent space. As mentioned in section 1-3, we want to reconstruct 5 variables, which will be done in a 3D domain with spatial dimensions [128,128,64] leading to input and output dimensions of [5,128,128,64] per sample. Next the dimensions of the latent space have to be chosen. As section 2-2-1 mentioned, Rombach et al. [27] found that a compression ratio of four lead to the best results so this will also be adopted here. However, for interpretability of the latent space it has been chosen that the 5 variables will be kept intact and thus only the spatial dimensions will be compressed, leading to latent space dimensions of [5,32,32,16]. Figure 3-1 shows what the data flow for the AE looks like.

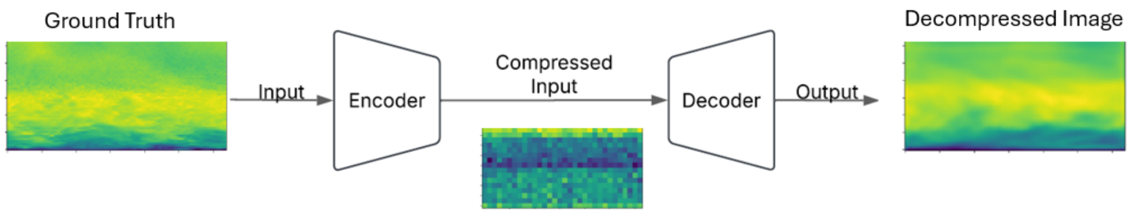

Figure 3-1: Data flow for training the AE.

Network Structure

Figure 3-2 shows the structure of the AE that was used to ensure the dimensions of the input, output and latent space were correctly implemented. This image shows how the dimensions of the inputs change throughout the process of down- and upsampling, while also showing the use of 4 residual blocks per layer to prevent vanishing gradients, as mentioned in section 2-2-2.

3-1-2 Training Setup

Aside from the architecture used in the AE, another important part of optimizing performance is choosing the right loss function. The total loss function, used during training, is shown in equation 3-1.

$$\mathcal{L}_{\text{AE}} = \mathcal{L}_{\text{NLL}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_g \cdot \gamma_g \cdot \mathcal{L}_G \tag{3-1}$$

Where:

- $\mathcal{L}_{NLL}$ is the Negative-Log Likelihood loss as shown in equation 2-4.

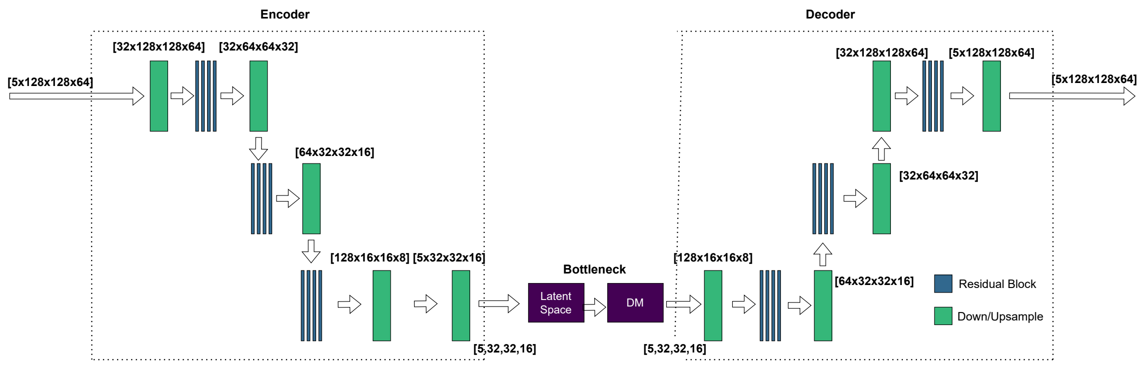

Figure 3-2: The architecture of the AE used to do flow reconstruction. DM denotes the actual DM

- λ_{KL} is a scaling factor for the KL divergence loss which is $1e^{-6}$ in our case, found through hyperparameter tuning.
- $\mathcal{L}_{KL}$ is the KL divergence loss as explained in equations 2-5 and 2-6.
- λ_g is a scaling factor for the Generator loss which is 0.5 in our case, found through hyperparameter tuning.
- γ_g is a weight that changes from 0 to 1 after a certain amount of steps, giving the discriminator time to learn what images are real. In our case γ_g changes from 0 to 1 after 5000 steps.
- $\mathcal{L}_G$ is the generator loss as explained in equation 2-7

This equation shows that the loss function used consists of a combination of all losses discussed in section 2-2-2. Furthermore, since a KL loss is used for the AE it is actually a VAE.

Another important feature of the training setup is the right optimizer used to optimize the loss function. Two optimizers widely used in Machine Learning are Adam [50] and AdamW [51] because of their ability to handle sparse gradients and their adaptive learning rate. The difference between Adam and AdamW are that AdamW takes care of regularization, meaning it prevents overfitting to the training data. However, since our AE is a VAE and we use KL divergence, which both already regularize the model, the extra regularization from the optimizer is not needed and thus Adam has been chosen as optimizer with a learning rate of $1e^{-5}$. Furthermore, due to the high memory use of the model, a batch size of 2 has been adopted.

3-2 Diffusion Model

This section outlines the architectural and training choices made for the diffusion model, guided by the theoretical background discussed in 2-2-3. Section 3-2-1 describes the final network architecture used for conditional generation in latent space, while Section 3-2-2 details the training setup, including loss formulation, optimizer configuration, and learning rate scheduling.

3-2-1 Architecture

This section will discuss the final architecture used for the DM. In this decision process considerations such as computational power and memory usage played important roles. Because training a DM is an expensive process a trade-off has to be made between model complexity and computational efficiency.

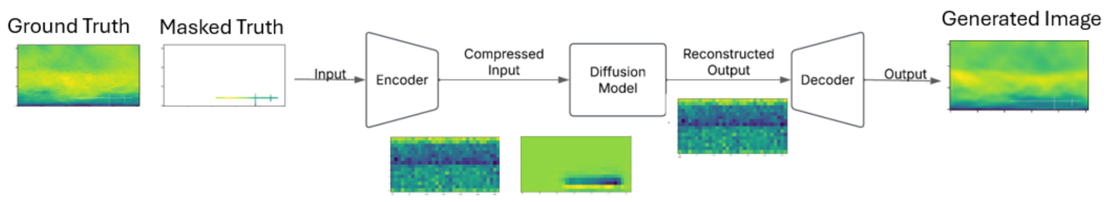

Figure 3-3: Data flow for training the Diffusion Model.

Input and Output

Unlike for the AE, the dimensions of the latent space are not that important for the DM since the it is not explicitly used. Furthermore, as figure 3-3 shows for training our conditional DM the input is not only the 5 variables that we want to reconstruct but also the masks used to mimic their measurements. This increases the input channels from 5 to 10 since each variable gets measurements in our case. Furthermore, to supply the model with information of where measurements are done a binary mask is also concatenated to the input. This mask contains ones for places where measurements are done and zeroes in the other places. Concatenating this mask as well, results in 11 channels. As mentioned in the previous section the AE compressed the data down to spatial dimensions [32,32,16] leading to final input and output dimensions of [11,32,32,16] for the DM.

Network Structure

Figure 3-4 shows the network structure of the DM. This structure shows that the input of dimensions [11,32,32,16] is sampled down in three layers to a dimension of [512,4,4,2] in the bottleneck, only keeping the most important details. This figure clearly shows the use of skip connections, which are characteristic for a U-net. Furthermore, it can be seen that in some layers attention blocks are used, for which the motivation was given in section 2-2-3. The use of attention is quite expensive which is why it is only used in downsampled layers, since this is less computationally expensive. In both the decoder and the encoder 5 residual blocks are used per layer while attention is not used for the first layer. Lastly, in the bottleneck 2 residual blocks and one attention block are used.

3-2-2 Training Setup

The loss function that will be used to optimize the DM during training is the one shown in equation 2-12. AdamW will be used as optimizer to help regularize the model during training.

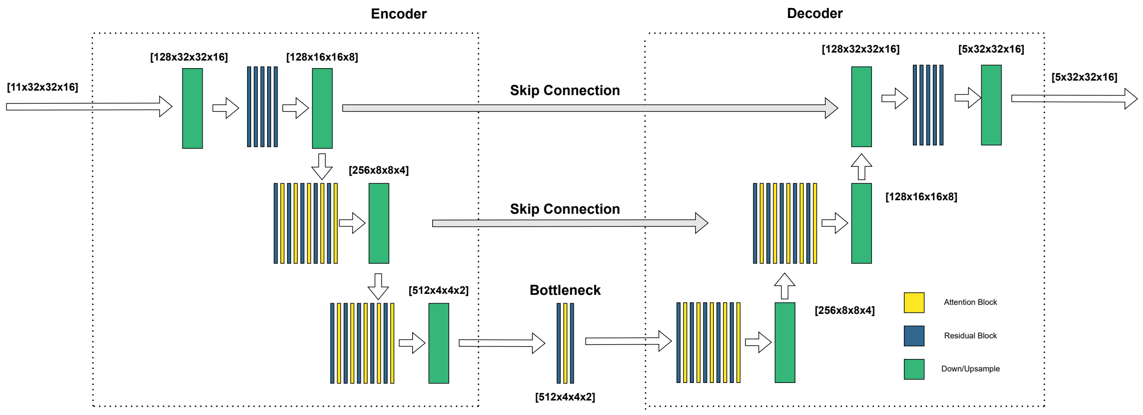

Figure 3-4: The architecture of the DM used to do flow reconstruction.

The learning rate that will be used follows a linear schedule instead of just being a constant value. The model uses a base learning rate of $5e^{-5}$ which is multiplied by a schedule that starts at $1e^{-6}$ and grows to one, linearly over 10000 steps. This has been done because without this schedule the model the training process could explode when the model sees image with a lot of noise resulting in exploding gradients. Lastly, due to memory constraints the batch size used is 1.

Chapter 4

Experimental Setup

The goal of this research is to reconstruct physically plausible atmospheric flow fields from sparse, on-site measurements using an LDM. Achieving this requires a realistic and diverse training dataset, accurate modeling of sparse observations, and careful selection of relevant atmospheric labels. This chapter outlines the experimental setup used to create such a foundation, covering data generation, preprocessing, measurement design, and hardware resources.

Section 4-1 details how training data was generated, including simulation setups for different ABL states and wind farm configurations. It also describes the data sampling strategy and the preprocessing pipeline used to prepare the data for model training.

Section 4-2 discusses how sparse observations and global labels are constructed to mimic real-world measurements. This includes the spatial masks that emulate LiDARs and meteorological masts, as well as discrete labels like BLH and R_b, which are used to improve flow reconstruction.

Section 4-3 provides an overview of how the experiments are set up in the following chapter, offering context for how the results were obtained.

Finally, Section 4-4 provides a brief overview of the hardware used for training and inference. The combination of high-fidelity LES data, realistic observational constraints, and appropriate computational resources ensures a robust and practically relevant setup for training the LDM.

4-1 Data Collection and Preprocessing

To reconstruct realistic atmospheric flow fields from sparse observations, we require a model capable of learning the complex turbulent wind data. Ideally, such a model would be trained on dense, high-resolution measurements of the flow across the entire domain. However, acquiring complete observational data in the atmospheric boundary layer is practically infeasible due to cost, accessibility, and the limitations of current sensing technologies. As mentioned in section 1-1, LES is a trade-off between fidelity and efficiency and will thus be used to generate realistic datasets.

4-1-1 LES Model: ASPIRE

The LES model used for this research is the model Atmospheric Simulation Platform for Innovation Research and Education (ASPIRE) developed and operated by the company Whiffle. This model runs almost entirely on Graphical Processing Units (GPUs) and originates from DALES: Dutch Atmospheric Large Eddy Simulation [52]. While DALES was built to study turbulence in the ABL, Aspire uses GPUs to increase its computational efficiency in order for it to be used for weather forecasting. Furthermore, the use of GPUs also allows the user to visualize the LES domain during runtime.

LES Boundary conditions

Appropriate boundary conditions are critical for any LES model. ASPIRE offers several boundary condition strategies, each designed for different assumptions about how flow behaves at the edges of the computational domain.

Periodic Boundary Conditions

Under periodic lateral boundary conditions, ASPIRE effectively "loops" the flow field across the domain edges. This means that all flow that leaves the domain on one side enters again on the other side. Periodic boundary conditions are applicable for real life situations when the part of the ABL you are interested in is relatively homogeneous; this could, for example, be when temperature gradients or terrain variations are small. These homogeneous conditions result in a more stable simulation and are generally cheaper to run than the other methods, this makes them a good starting point for testing the diffusion model. Because of the homogeneity of the simulation there is no new momentum or energy injected into the domain, this is why in periodic simulations external forcing mechanisms are used. An example of this is applying a constant geostrophic wind, which acts as a large-scale pressure gradient force that drives flow across the domain.

NonPeriodic Boundary Conditions

Unlike periodic boundary conditions, nonperiodic boundary conditions rely on explicit inflow/outflow conditions that can either come from data, through a Numerical Weather Prediction (NWP) or self built boundary conditions. These conditions assume that the flow is known upstream and propagates downstream, requiring external input from observational data in combination with coarser large-scale models leading to real-weather LES. Nonperiodic boundary conditions are essential when the flow is inhomogeneous, such as in the presence of obstacles, complex terrain, or varying atmospheric conditions. Furthermore, nonperiodic simulations are also necessary when the simulated domain becomes so big that spatial gradients in weather conditions will occur, this is something that is not modeled by a periodic simulation.

While using nonperiodic boundary conditions results in more realistic simulations, this also leads to a dataset that is much more complex than a periodic one. This is due to the fact that a nonperiodic simulation is much more dynamical than a periodic one which results in e.g. more changes in boundary layer height and shear. Even though reconstructing data gathered from a nonperiodic simulation would thus bring us closer to reconstructing real world weather this is a big step since there is little known about a DM's ability to reconstruct flow fields. This is why, for this research, the model will be reconstructing data gathered by running periodic simulations in order to get a better understanding of a DM's behavior in reconstructing different ABL states with different measurements being available.

4-1-2 Datasets and Simulation Setup

This work is trying to extend the works by Rybchuk et al. [22] by creating a more realistic dataset. This will be done by first using a dataset similar to that in [22] and then expanding it in small steps. In order to do this the following datasets will be used:

1. A NBL with a single BLH and no turbines
2. A NBL with two different BLHs containing a 6x6 wind farm
3. A SBL containing a 6x6 wind farm
4. A combination of datasets 2 and 3.

These datasets have been picked to show that our model is able to reconstruct similar data to that of Rybchuk et al. After that we want to show that our model is able to reconstruct turbine influenced flow fields and also flow fields with varying BLHs. Then we want to show that our model can also reconstruct an SBL, also containing wind turbines. Finally, by combining the two datasets we want to show that our model is able to distinguish different ABL states from each other in a dataset containing wind turbines and varying BLHs for the NBLs. Furthermore, these datasets will contain the variables u, v, w, T_{hl} and q_t for each snapshot, to make sure the generated samples could be used as initial conditions for LES.

Simulation Setups

Now that it is known what datasets we want to use we can start setting up the simulations that will generate the data. This will be done by first generating the dataset that only contains an NBL with a single BLH and no turbines, and later altering this dataset by making small changes to the simulation. The periodic LES simulation has to be initialized using an input profile, which is depicted in table 4-1. This setup was inspired by a paper by D.Allearts and J.Meyers [53] who simulated an NBL and an SBL. Values taken from their simulations are the roughness lengths, geostrophic wind and the vertical potential temperature profile.

A NBL with a single BLH and no turbines

As mentioned all other simulations will have small changes compared to the input profile in table 4-1. The table shows the setup for the NBL with a single BLH without turbines.

NBL with two different BLHs containing a 6x6 wind farm

Next, from the NBL with one BLH without turbines we go to an NBL with two different BLHs and a 6x6 wind farm, which was created by using the same input profile as shown in table 4-1 but then a 6x6 wind farm was added. The 6x6 by wind farm consists of 5MW turbines with a diameter of 126 meters. The wind turbines are evenly spaced from each other 1008 meters away and the wind farm is centered in the middle of the domain.

Table 4-1: Input profiles and boundary conditions for the NBL simulation with an inversion layer at 1000 meters.

Quantity	**Symbol**	**Values**	**Notes**
Heights	z	0, 900, 1000, 1100, 1500, 2000 m	Height levels for profiles
Moist-static-energy temperature	T_{hl}	300, 300, 304, 304, 305.6, 307.6 K	Isothermal up to 1 km, stronger inversion above
Specific humidity	q_t	0.0001 (constant)	Dry profile
Horizontal wind	u	12 m s^{-1} (constant)	No vertical shear
Horizontal wind	v	0 m s^{-1} (constant)	No cross-wind component
Geostrophic wind	u_g	12 m s^{-1} (constant)	Large-scale forcing
Geostrophic wind	v_g	0 m s^{-1} (constant)	Large-scale forcing
Surface pressure	p_s	101300 Pa	Typical sea-level pressure
Roughness length	z_{0m}	0.1 m	Momentum roughness
Roughness length	z_{0h}	0.1 m	Heat roughness
Surface moisture flux	wq_t^s	0.0	No evaporation at the surface
Surface heat flux	$w\theta_l^s$	0.0 K m s^{-1}	Neutral: no surface heating/cooling

Another addition to the dataset is another BLH, which was created by using the same input profile but than with different potential temperatures at different heights to create an inversion layer at a different height. The new input profile for the potential temperature is shown in table 4-2 while all other inputs are kept the same, for this simulation the same 6x6 wind farm was added.

Table 4-2: Input profiles and boundary conditions for the NBL simulation with a inversion layer at 1400 meters.

Quantity	**Symbol**	**Values**	**Notes**
Heights	z	0, 1400, 1500, 1600, 1800, 2000 m	Height levels for profiles
Moist-static-energy temperature	T_{hl}	300, 300, 304, 306, 308, 310 K	Isothermal up to 1.4 km, inversion above

SBL containing a 6x6 wind farm

To obtain the dataset containing an SBL that is influenced by wind turbines only one value had to be changed with respect to the previous dataset. Both input profiles from the previous section were used but now the surface heat flux $w\theta_l^s$, is changed from 0.0 to -0.01 Kms^{-1}. This has been done to mimic the surface cooling that is present in an SBL at for example night time.

NBL and SBL both containing a 6x6 wind farm

The final dataset that was used was a combination of the previous two, leading to a dataset containing turbine influenced flows, in two different ABL states where the NBL contained two different BLHs. Using this final dataset should show that the model used is able to distinguish different BLHs from each other while also being able to differentiate between different ABL states, increasing the realism of the problem compared to that of Rybchuk et al.

Parent-Child Simulation

Because the boundary conditions used for these simulations are periodic, it is not realistic to just place a wind farm inside them. This is due to the nature of a periodic run, where flow leaving the domain on one side enters again on the other side. This means that when a wind farm is placed inside such a simulation, flow, that just passed a wind farm and has slowed down, when leaving the domain flows back into the domain and passes through the wind farm again, essentially creating an infinite wind farm, which is not realistic. In order to tackle this issue a parent-child domain configuration is used to generate physically consistent inflow conditions for a wind farm simulation. To address this, the simulation is split into two domains.

A parent domain is initialized with periodic lateral boundaries and the same ABL state as the corresponding child simulation. This parent domain spans a much larger area and does not contain any turbines. Its primary purpose is to allow the wakes generated in the child domain, influenced by wind turbines, to recover before potentially re-entering the wind farm. This necessitates the use of a large area. The domain configuration is summarized in Table 4-3 [1]. While the resolution is relatively coarse, this is intentional: since detailed flow structures are not required in the parent domain, a lower resolution significantly reduces computational cost without affecting its effectiveness for wake recovery.

Table 4-3: Domain configuration for the parent simulation.

Parameter	**Value**	**Description**
N_x	128	Number of grid points in the x-direction
N_y	128	Number of grid points in the y-direction
N_z	64	Number of grid points in the z-direction
L_x	25,192 m	Domain size in the x-direction
L_y	25,192 m	Domain size in the y-direction
L_z	2,048 m	Domain height
dz_{in}	16 m	Initial vertical grid spacing

Secondly, a child domain is defined within the larger parent domain and represents the region of primary interest. This smaller domain receives its boundary conditions from the parent domain to ensure that the inflow is not already influenced by turbine-induced disturbances. This setup allows for a more accurate representation of the undisturbed atmospheric boundary layer entering the simulation domain. The configuration details of the child simulation are provided in Table 4-4.

These boundary conditions are satisfied by using "nudge factor" and "nudge extent" variables. Nudge factor stands for how aggressively the flow in the child domain is forced towards that of the parent domain. In this research a nudge factor of 0.2 has been chosen which means that 20% of the discrepancy between the child and parent flow is corrected. This value has

[1]While not the recommended simulation setup, this configuration produced results considered good enough for the purposes of this study.

Table 4-4: Domain configuration for the child simulation.

Parameter	Value	Description
N_x	128	Number of grid points in the x-direction
N_y	128	Number of grid points in the y-direction
N_z	64	Number of grid points in the z-direction
L_x	8192 m	Domain size in x-direction
L_y	8192 m	Domain size in y-direction
L_z	2048 m	Domain height
dz_{in}	16 m	Initial vertical grid spacing

been picked because it is a good trade-off between forcing the flow to look like that of the parent while still keeping it realistic. The nudge extent defines how far into the child domain the nudging is applied. The value chosen for this is 15 which means that nudging happens at 15 grid boxes from the boundaries. This value has also been picked because of its trade-off between realism and making sure the parent flow is sufficiently adopted.

In summary, this nested domain setup enables the realistic simulation of wind farm wake dynamics in an idealized boundary layer, while still preserving the advantages of periodic LES for turbulence generation. Enabling us to create ABL states as we see fit to ultimately test our model.

4-1-3 Data Sampling and Preprocessing

This section will mention how long the simulations were run and how often the data was sampled from the simulations. After that it will be discussed how the obtained data was stored and preprocessed to use during training of the model, this also includes the train, test and validation splits used.

Simulation Durations and Sampling Frequency

The data was acquired by running four different simulations for a day of which the first six hours were spin-up. To ensure more variability each simulation was based on a different, randomly chosen seed. ASPIRE outputs flow statistics at fixed intervals. Specifically, three-dimensional snapshots of the following variables were stored every 120 seconds:

- Horizontal wind components: u, v
- Vertical wind component: w
- Liquid water potential temperature: T_{hl}
- Total specific humidity: q_t

These variables were saved on a full 3D grid of shape $(128 \times 128 \times 64)$. Furthermore, the global variable R_b was also extracted every 120 seconds.

Data Format and Preprocessing

The simulation output was stored in a format compatible with Python based tooling. For use in training, the following preprocessing steps were applied.

The data is outputted as one big .nc dataset and converted to a .lmdb dataset to ensure efficient loading during training.

All fields were normalized per channel between the values -1 and 1 using a min max normalization. This has been done to prevent the model prioritizing a certain channel because the values in that channel are simply higher.

In order to prevent overfitting the dataset will be split into three groups: Training, Validation and Testing. The Training set is there to increase the model's performance on the data by changing its parameters to improve the outputs. The Validation set is there to validate that the model is not overfitting to the Training set at the end of each epoch. Finally, when training is done the model will be used on the Test set to inspect how the model performs on data not seen during training at all. These three groups are partitioned as follows:

- **Training set:** 80% of all available snapshots, balanced across ABL types (SBL and NBL).
- **Validation set:** 10% of the data used to validate the improvement of the model on the training data. After each epoch the current model is scored on the validation data and saved if it gets the best score on this data.
- **Test set:** 10% of the data used to evaluate generalization performance after training is done. This set is used for testing the model's performance after training is done, to see how the model handles unseen data.

4-2 Used Observations

The goal of this research is to build a LDM that, when given measurements, can reconstruct a flow field that is closely related to the real flow field. These measurements can, for example, be LiDARs measuring the wind speed, temperature sensors and anenometers measuring the wind direction. These measurements will be passed to the LDM which will use them to reconstruct the flow field. How this is done will be explained in the following sections.

4-2-1 Spatial Observations

In order for the diffusion model to reconstruct the correct flow from sparse on-site measurements, it must learn the relationship between the observed measurements and the unobserved regions of the domain. This is achieved by training a conditional diffusion model for inpainting, as described in Section 2-3-1.

Specifically, during training, the sparse measurement mask is concatenated with the full flow field, allowing the model to learn how local observations relate to the global flow structure.

However, during inference, only the masked input is provided, and the model is expected to extrapolate and reconstruct the full flow field from this limited information.

To find out whether the model will be able to use real-life measurements, it is important to mimic these measurements through the use of a realistic mask. To do this, the mask in figure 4-1 has been used. In this figure purple means no information and yellow means there is information in this place. This figure consists of a top and side view of the mask, where the top view is from a horizontal slice at a height z = 3 which has been picked since this is at hub height for the wind turbines in our simulations. The disks that are visibile in this slice represent spinning LiDARs. The shape of this mask has been chosen to mimic measurements done by a LiDAR measuring the wakes of the turbines. It is desired that the known values contain the wakes of the turbines, in real-life this happens because nacelle mounted LiDARs rotate with the turbine when the wind direction changes. Because this does not happen with these masks a circular shape has been chosen for all turbines to make sure that there is always a wake in the known regions of the mask. In the side view the horizontal lines that are shown are the same LiDAR disks as visible in the top view and the vertical line represents a meteor mast that does measurements from heights z = 0 to z = 3. This is important to measure information such as the temperature gradient and the wind profile, which, as mentioned in section 2-1, give a lot of information about the state the ABL is in. Such a meteorological mast is also often at a wind farm so this is a realistic addition to the mask.

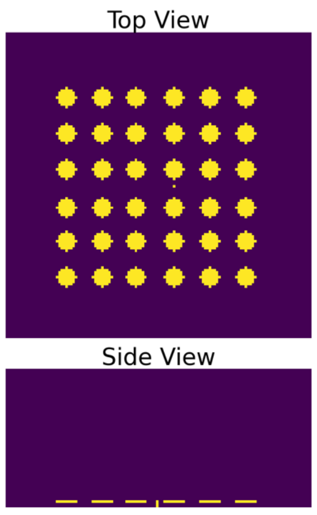

Figure 4-1: The mask used for mimicking on-site measurements, showing a top view at height z = 3 and a side view at y = 70.

4-2-2 Discrete Observations

One of the proposals of this research is to add global conditioning labels to the model to feed it information about the flow field that it is not able to extract from just the spatial measurements in the masks. Two labels that come to mind that could be important additions with respect to Rybchuk's research are the BLH and R_b.

The addition of the BLH is necessary because most measurements are done below 300 meters while the inversion layer can grow to even bigger heights than 2000 meters. The flow for a BLH of 1000 meters can be exactly the same as that for a BLH of 1500 meters so to correctly reconstruct flow fields with the right BLH the model has to get this information from somewhere else than the mask, which it can get from a label.

Another useful label for the dataset we are using is the R_b, since we are now combining two different ABL states. By just using the mask the model might not be able to distinguish the different states from each other and, as mentioned in section 2-1, the R_b gives a clear differentiation between these states. So by feeding these labels to the model it should be able to tell in which state the ABL is, helping it with generating the correct flow field. As equation 2-1 shows, the R_b is computed by calculating the difference in wind speeds and potential temperature between two different heights. However, it is important to note that Rb calculations are sensitive to the heights selected to measure differences in virtual potential temperature and wind speed components. For instance, when the upper measurement height is above the actual boundary layer in stable conditions, the calculated R_b value might not accurately reflect the true stability condition. Since we will be investigating an SBL which has a low BLH the upper measurement is chosen to be at 200 meters, while the lower one is at 0 meters.

Adding these two labels to the model could be a powerful addition for helping the model reconstruct the right flow fields, but how are these labels obtained? These labels can be obtained from re-analysis data such as ERA5 [54] or coarse-scale models such as Whiffle's Mesoscale [55] model which is used as a coupling between observational data and ASPIRE to give approximations of the state of the ABL.

4-3 Experiments

To evaluate the model's performance on the datasets introduced in Section 4-1-2, four experiments are conducted, as outlined in Table 4-5. Each experiment represents a specific combination of ABL regime, turbine presence, and label availability. While the training data for each experiment remains fixed, samples and labels passed to the model during inference may change to test the importance of labels. So for example, even though experiment 2 is always trained on both BLHs, during inference it is possible to only give samples with a high BLH to the model or let the model infer samples without a label.

To distinguish between these variations, a compact notation is adopted. For example, `Exp2-∅,M` refers to Experiment 2 performed without any labels for the mixed set meaning both high and low BLHs, while `Exp2-B,M` indicates the same experiment with the BLH label included. Similarly, `Exp4-BR,M` denotes Experiment 4 conducted with both BLH and R_b labels for the SBL and NBL combined. Furthermore, experiment 4 inferred on only the SBL samples is denoted by `Exp4-BR,S`, since the NBL in experiment 4 still exists of both high and low BLHs a high BLH is noted as follows, `Exp4-BR,NH` where NH stands for NBL High respectively.

This coding scheme is used throughout the remainder of the report to clearly indicate the experimental setup and inference conditions associated with each result.

Table 4-5: Overview of the four experiments performed, indicating ABL regime, turbine presence, and label configuration. An "X" in the Inference column means there are no different inference subsets. The term "Mixed" refers to: both high and low BLH samples in Experiment 2, and both SBL and NBL samples in Experiment 4.

Experiment	**ABL State**	**Turbines**	**Labels**	**Inference**
Experiment 1	NBL	No	None	X
Experiment 2	NBL	Yes	BLH, Ø	High, Low, Mixed
Experiment 3	SBL	Yes	None	X
Experiment 4	Mixed	Yes	BLH, R_b, Ø	NBL, SBL, Mixed

4-4 Used Hardware

All model training was performed on a workstation equipped with an NVIDIA GeForce RTX 3090 GPU, which features 24 GB of memory and 10496 CUDA cores. This high memory GPU enabled the training of 3D convolutional networks with large spatial domains and deep architectures using residual and attention blocks. To prevent data loading bottlenecks and improve overall training speed, the dataset was stored on a high speed Solid State Drive (SSD), significantly reducing input/output (I/O) latency compared to traditional hard drives. The combination of GPU compute power and fast storage ensured that model training proceeded efficiently.

In summary, the chosen LES setup, domain configuration, and sparse measurement masks provide a physically consistent and computationally tractable foundation for training the LDM. Together with carefully selected labels such as the Bulk Richardson number, this experimental design enables a robust investigation into the model's ability to reconstruct diverse ABL states from sparse, realistic measurements

Chapter 5

Analysis of the Results

This chapter outlines the methodology used to evaluate the performance of the conditional diffusion model. While the generated samples may appear visually realistic, a good analysis is needed to assess whether they are statistically consistent with the training data, physically plausible, and how they behave under varying conditioning inputs

5-1 NBL with one BLH

5-1-1 Visual Inspection

Figure 5-1 shows a visual representation of the model's outputs for the variable u, while conditioned on the measurements shown in the dotted, red lines in the input. This figure shows that the model generates images that look like a possible realization of the measurements it was conditioned on, while not being exactly the same. This is because of the DM's probabilistic behavior and the fact that the mask is not hard constraint but given as additional information. This figure also shows that the model is able to reconstruct realistic turbulence. Visualizations for the other variables are included in Appendix A.

5-1-2 Quantitative Analysis

Statistical Analysis

Firstly, we compare the probability-density functions (PDFs) of the sampled data with those of the test set. Because a diffusion model is explicitly trained to approximate the data distribution (Section 2-2-3), PDFs provide a natural first quantitative check. Figure 5-2 shows that the sampler reproduces the distribution of each variable: the blue (test) and orange (sampled) curves peak at identical locations for u, v, w, T_{hl} and q_t. Nevertheless, the sampled peaks are consistently lower and broader than the test peaks. We interpret this as residual additive noise: small perturbations introduced (or not fully removed) during the

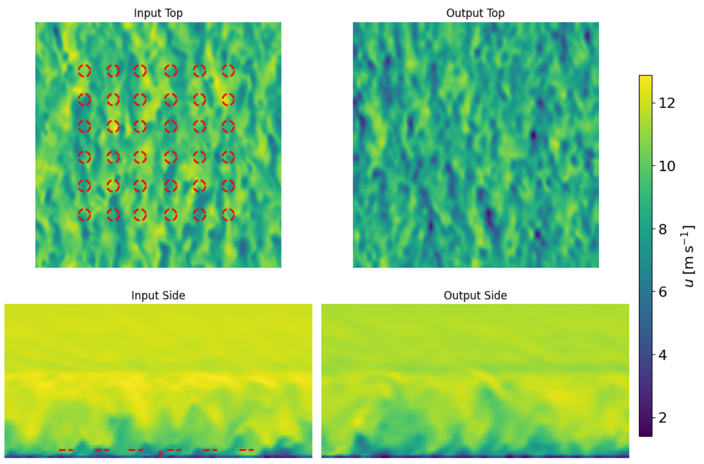

Figure 5-1: A top and side view of the variable u showing the input and the model's output for the NBL case. Areas inside the red dotted lines represent measurements given to the model.

reverse diffusion process spread probability mass from the central bin into neighboring bins. Matching the PDFs is therefore largely successful, but the slight broadening suggests room for further sharpening.

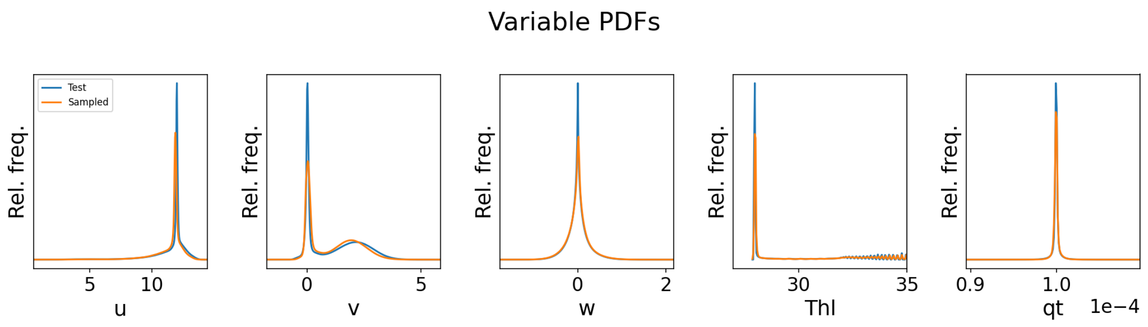

Figure 5-2: Figure showing the PDFs for all five variables of the test and sampled data for the NBL case without turbines, `Exp1`.

Next, we will have a look at the horizontally averaged vertical profiles for all five variables. This will show whether the model has correctly captured the characteristic profiles per variable for a NBL. Looking at figure 5-3 it can be seen that for u, v, w and q_t of the sampled data follows that of the test data closely, except for a small negative bias in the lower half of the profile for v, however this bias still falls well with in the standard deviation. The sampled variable T_{hl} also looks similar to that of the test data, however there are some small errors. As we can see T_{hl} above the inversion layer, at approximately 1000 meters, the sampled data follows the test data well, however, below that the sampled data has a positive bias, that even falls outside the standard deviation, and looks a bit noisy compared to the straight line from the test data. This is particularly remarkable given that the upper part of the temperature profile, which the model reconstructs well, exhibits stronger nonlinear variation, whereas the lower section is nearly constant in height and should, in principle, be easier to reconstruct.

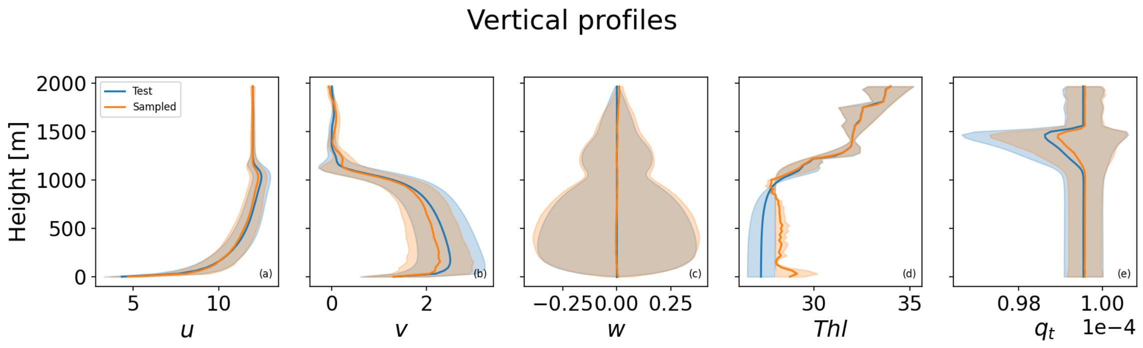

Figure 5-3: Figure showing the horizontally averaged vertical profiles for all five variables of the test and the sampled data for the NBL case without turbines, the shaded areas represent the standard deviation from the mean, `Exp1`.

Physical Analysis

To assess whether the generated flow fields adhere to physical principles, we analyze two key metrics: mass conservation and the turbulent energy spectrum.

We evaluate the turbulent energy spectrum. According to Kolmogorov's theory, fully developed turbulence should exhibit an inertial range with a spectral slope of $-5/3$. Figure 5-4 compares the energy spectrum of the test and sampled fields at a representative height ($z \approx 90$m). The test data follows a slope slightly flatter than $-5/3$ between $k_x \approx 0.002\ m^{-1}$ and $0.01\ m^{-1}$, as expected, before dropping off due to resolution limits below the subgrid scale. The sampled data matches this behavior in the inertial range but diverges at higher wavenumbers, where it flattens rather than decaying. This excess energy at small scales likely reflects high frequency noise that the model was unable to fully suppress during sampling. While this discrepancy does not strongly affect large-scale flow features, it suggests that some form of spectral filtering or physics aware loss function may be necessary for enforcing realistic small-scale behavior.

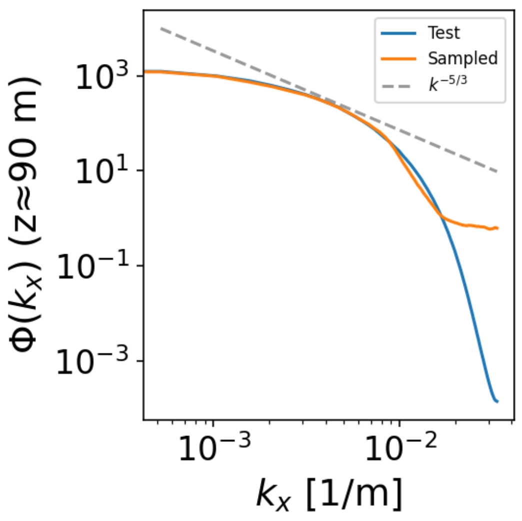

Figure 5-4: This figure shows the turbulence energy spectrum of the test and sampled data compared to the energy cascade with a spectral slope of -5/3 for the NBL case without turbines, `Exp1`.

In incompressible flows, the divergence of the velocity field should be zero at every point in space, reflecting conservation of mass. The divergence is the sum of the velocity components in all three spatial directions. Figure 5-5 compares the distribution of velocity divergence in the test and sampled data. For both, the distributions are centered around zero, but the test data exhibits a sharper, more pronounced peak. In contrast, the sampled data has a lower and broader peak, suggesting that perfect incompressibility is not preserved as well. This degradation is likely due to residual noise introduced during the reverse diffusion process, which the model may not fully eliminate. Another contributing factor could be that the model was not explicitly trained to enforce incompressibility, and therefore does not prioritize this physical constraint.

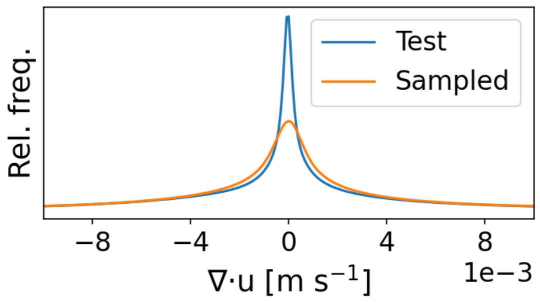

Figure 5-5: This figure shows how the test and sampled data adhere to the conservation of mass, total conservation of mass would lead to one vertical line at $\nabla u = 0$, `Exp1`.

While the results in this section are promising and demonstrate the model's ability to reconstruct realistic flow fields from sparse measurements it is important to note that, as mentioned in section 1-3, this dataset is far from real-world conditions. These results form a baseline for further investigating the model's shortcomings and limitations.

5-2 Turbine Influenced NBL with 2 BLHs

We will now look at the model's ability to reconstruct flow fields for a more complicated dataset where we add turbines to the NBL and we will have two different BLHs. One of the additions to the model for this case is a label that contains the BLH, this analysis will show that the model is able to reconstruct turbine influenced flow fields with different BLHs when using a label. The need for this label will be mentioned shortly in the statistical analysis of this section and will be further discussed in another section.

5-2-1 Visual Inspection

Figure 5-6 shows the model's reconstruction for the horizontal wind component u, compared to the masked input. The output retains realistic turbulent features and shows accurate reconstruction of turbine wakes. The reconstruction for the other variables are shown in Appendix A.

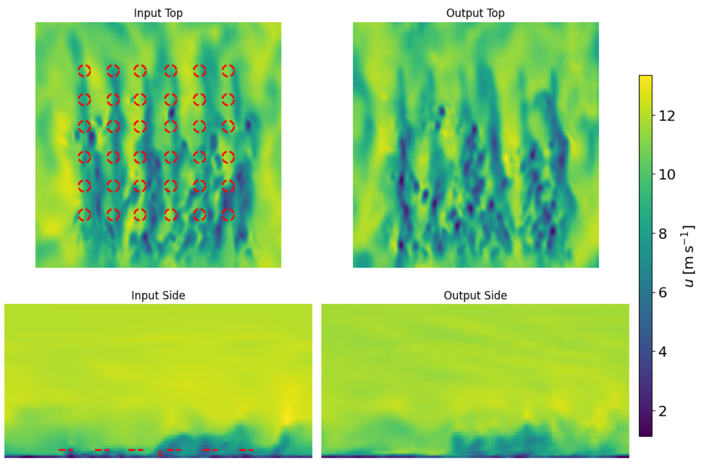

Figure 5-6: A top and side view of the variable u showing the input and the model's output for the turbine influenced NBL case with two BLH. Areas inside the red dotted lines represent measurements given to the model. The side view is a slice taken along the wind direction.

5-2-2 Quantitative Analysis

Now that we have shown that the model is able to reconstruct images that look qualitatively similar it is also important to check whether the samples generated by the model are quantitatively similar to that of the data. This will first be done by visually comparing the vertical profiles of the test data to that of the sampled data, after that these profiles will also be compared numerically.

Statistical Analysis

Figure 5-7 shows the vertical profiles for a high and a low BLH from the test data compared to the model output without a label. The model has only been given samples with a high BLH here. In this figure most profiles look similar except for T_{hl} where you can clearly see that the temperature profiles are different for the two different BLHs. This plot also clearly shows that without the label the model's output hovers somewhere in between the profiles of the low and high BLH, while it should only be reconstructing high BLH. This indicates that there is indeed need for a label informing the model about the BLH.

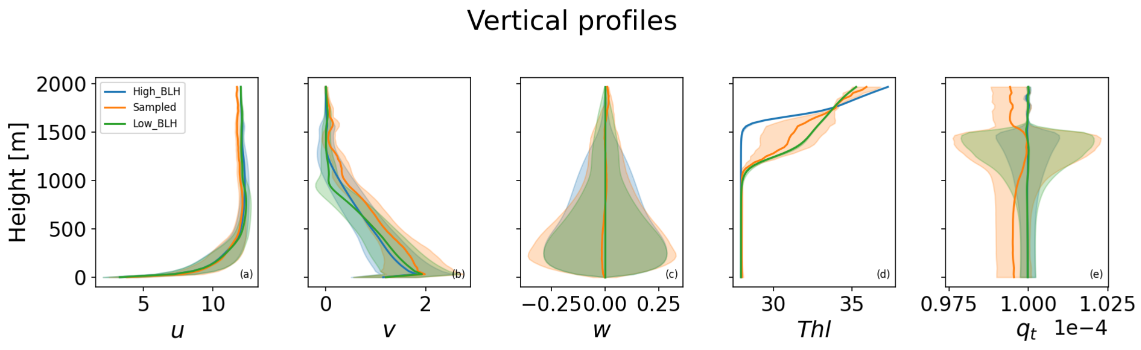

Figure 5-7: Figure showing the horizontally averaged vertical profiles for all five variables of the test and the sampled data for the turbine influenced NBL case for two BLHs without a label, the shaded areas represent the standard deviation from the mean, `Exp2-∅,NH`.

Figure 5-8 shows the vertical profiles for the same dataset but then with a label, showing that when the label is given the model only reconstructs high BLH. The sampled profiles for u, w, and T_{hl} follow the test data closely. The variable v shows a slight positive bias of approximately $0.2m/s$ between 500–1700 m, while q_t is overestimated by roughly 1×10^{-6} kg/kg. Both deviations remain within reasonable bounds. This overestimation may be attributed to the narrow dynamic range, with the majority of values concentrated within just 6% of the full min–max range. As a result, the model encounters small gradients during training, which may limit its ability to accurately optimize this variable. Furthermore, the vertical profile for q_t from the test data does not look realistic. Normally the specific humidity would decrease above the inversion layer but that is not the case here. This unrealistic profile from the test data could be another reason the model is not reconstructing it well because it has problems learning relations between the data because of the unrealistic humidity levels.

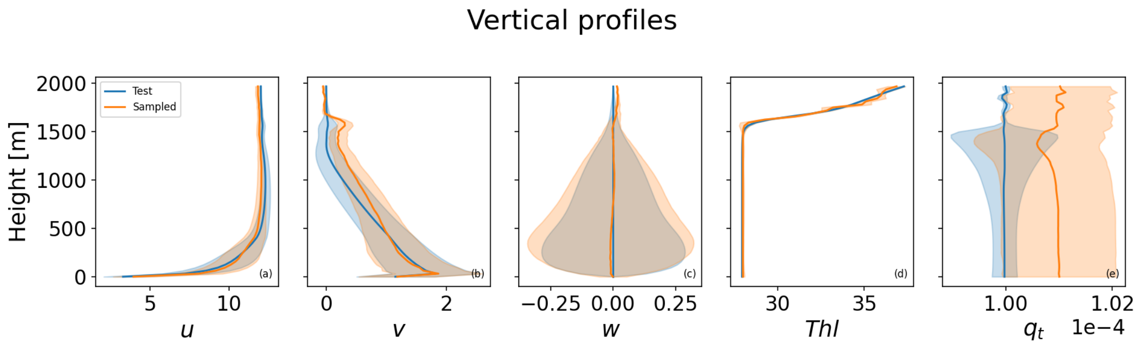

Figure 5-8: Figure showing the horizontally averaged vertical profiles for all five variables of the test and the sampled data for the turbine influenced NBL case for a high BLH, the shaded areas represent the standard deviation from the mean, `Exp2-B,NH`.

Lastly, figure 5-9 shows that the model is also able to reconstruct the low BLH as well when it is informed with a label.

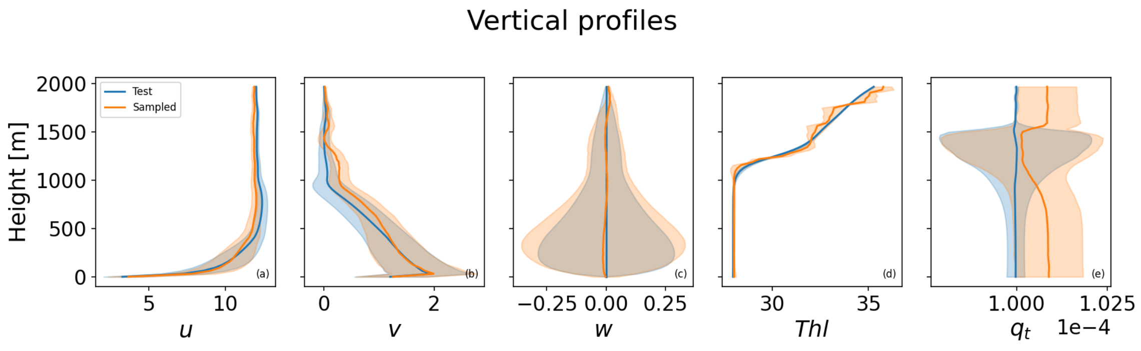

Figure 5-9: Figure showing the horizontally averaged vertical profiles for all five variables of the test and the sampled data for the turbine influenced NBL case for a low BLH, the shaded areas represent the standard deviation from the mean,`Exp2-B,NL`.

5-2-3 Vertical–profile error metric

To better understand the model's improvements when additional labels or data are introduced, the errors between the sampled vertical profiles and those from the test data will be computed. This enables a straightforward comparison of model performance across different configurations. Before error calculation, all variables are normalized to prevent those with larger magnitudes from influencing the error values more than those with small magnitudes.

For each variable the script evaluates the vertical L^1 distance between the two normalized profiles resulting in the overall integral between the two profiles.

$$I(x) = \int_{z_{\text{surf}}}^{z_{\text{top}}} \left| \bar{\mathbf{x}}_{\text{Sampled}}(z) - \bar{\mathbf{x}}_{\text{Test}}(z) \right| \mathrm{d}z, \tag{5-1}$$

The results of this analysis are presented in Table 5-1, showing that the model conditioned on labels consistently yields lower errors across all variables except for q_t. Interestingly, for q_t, the model without labels performs significantly better. This discrepancy may be attributed to the fact that neither model was able to learn this variable particularly well. Overall, the mean error for the model incorporating BLH information via a label is approximately half that of the model without any additional conditioning. More plots for this dataset such as the PDFs, Conservation of mass and turbulence energy spectrum can be found in Appendix A

Table 5-1: Vertical profile errors for the turbine influenced NBL with and without labels.

Model	u	v	w	T_{hl}	q_t	Mean
NBL without label	0.85	1.10	0.070	5.46	0.065	1.63
NBL with label	0.80	0.83	0.062	0.72	1.70	0.82

Wake Reconstruction

Lastly, we also want to investigate whether the model is correctly reconstructing the wakes that are present in the data. We will do this by looking at the velocity deficit in the test data compared to that of the sampled data.

To assess the wake structure, an upstream reference velocity U_∞ was computed as the average wind speed over several grid points upstream of the turbines. The Velocity Deficit (VD) was then calculated along turbines centered planes comparing the local wind speed to U_∞ as follows:

$$\mathrm{VD}(x,y,z) = 100 \cdot \frac{1}{N} \sum_{1}^{n=6} \frac{U_\infty(x_n,z) - U(x_n,y,z)}{U_\infty(x_n,z)} \, [\%]. \tag{5-2}$$

Where the summation is done for the six turbine rows present in the simulation.

To reduce small-scale variability, VD is averaged across a narrow strip centered at the turbine row, yielding a 2D curtain of mean velocity deficit in the height–downwind plane. To focus on the general wake structure rather than sample specific turbulence, the velocity deficit curtains are averaged over many individual flow fields. This ensemble averaging highlights whether the model captures the typical shape and magnitude of turbine wakes across a range of conditions.

Figure 5-10 compares the mean velocity deficit from the test data and the generated samples for this NBL case. The sampled flow fields clearly exhibit wake structures in the lower right part of the domain, indicating that the model is capable of reconstructing turbine induced velocity deficits. However, wake development in the test data begins earlier, around 2000 m downwind, while the model does around 3000 m. This discrepancy may stem from the lower magnitude deficits in that region, which the model might interpret as background turbulence rather than wakes, which could possibly be circumvented by adding information about turbine locations. Another reason for this loss of fine-scale details could be the AE that loses this information during compression. Furthermore, the magnitude of the velocity deficit is systematically underestimated in the sampled data, as indicated by the darker (lower VD) regions compared to the brighter wake in the test data.

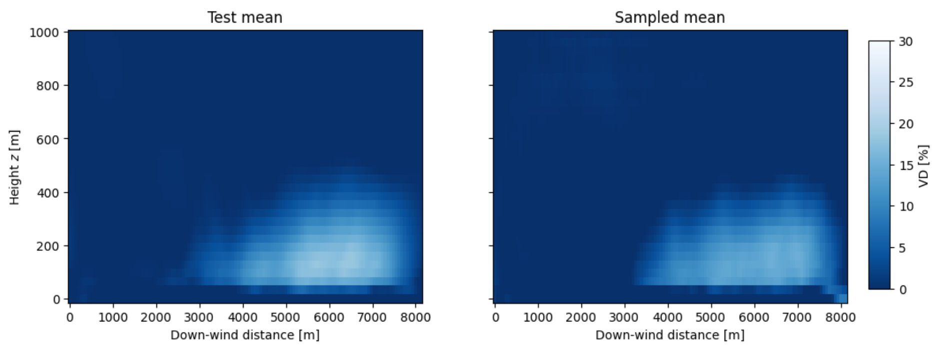

Figure 5-10: Comparison of the mean velocity deficit (VD) curtain for test data and generated samples under NBL conditions.

In this section we have shown that the model has trouble inferring the right BLH when just using measurements. However, this shortcoming can be overcome by adding a label containing information about the BLH. Furthermore, the VD figure has shown that even though the model is able to capture the general structure and locations of the wakes, it does not fully capture the right magnitude and small scale structures.

5-3 Turbine Influenced SBL

Now that we have shown that the model is able to reconstruct a turbine influenced NBL with different BLHs we want to show that it is also able to reconstruct a SBL.

5-3-1 Visual Inspection

Figure 5-11 presents a representative reconstruction of the horizontal wind component u for the SBL case. Compared to the more turbulent NBL, the flow field here appears more stratified, with reduced mixing and clearly defined turbine wakes. The model successfully captures these characteristics, including the lower boundary layer height, and produces output that is visually consistent with the expected flow features. Reconstructions for the other variables can be found in Appendix A.

5-3-2 Quantitative Analysis

The vertical profiles in Figure 5-12 confirm strong reconstruction performance for u, v, and w, with near perfect alignment between sampled and test data. For T_{hl}, the model follows the general trend but deviates slightly above 1000 m. The specific humidity q_t is systematically underestimated by approximately 5×10^{-6} kg/kg, though the overall shape is preserved.

Wake Reconstruction

Figure 5-13 shows the VD curtains for test data and generated samples under SBL conditions. Compared to the NBL case, the wakes in the SBL are more coherent and persistent due to

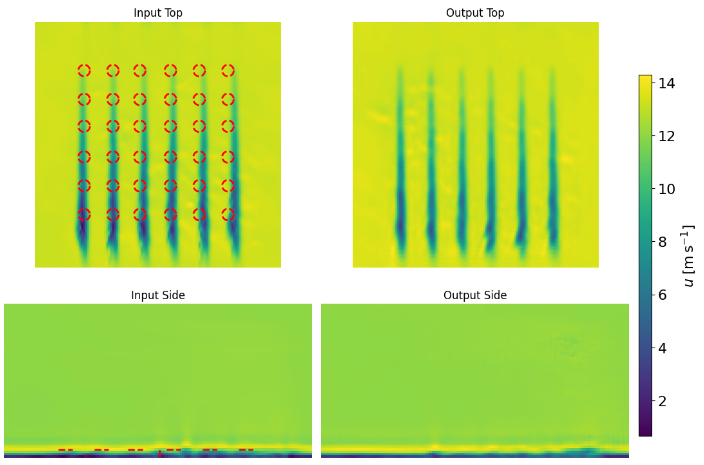

Figure 5-11: Figure showing the output for the variable u of the model when conditioned on the input for the turbine influenced SBl case. The red areas in the left images represent the mask that is given to the model during inference. The side view is a slice taken along the wind direction.

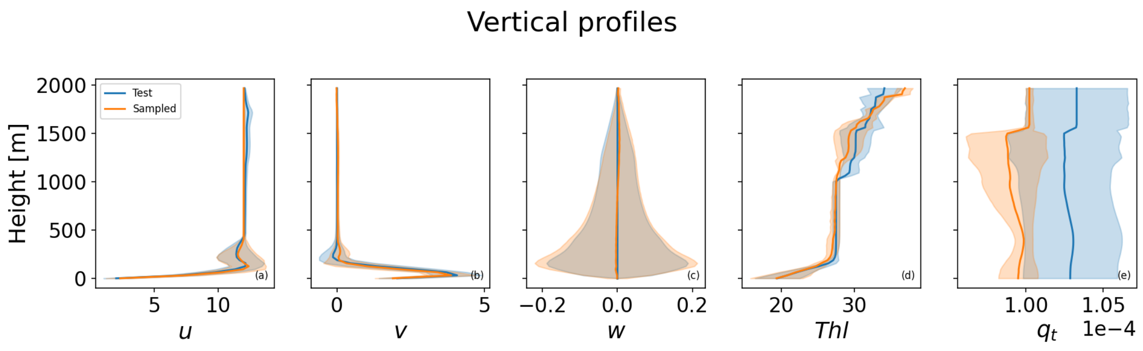

Figure 5-12: Figure showing the horizontally averaged vertical profiles for all five variables of the test and the sampled data for the turbine influenced SBL case, the shaded areas represent the standard deviation from the mean, `Exp3`.

reduced turbulence and mixing. The sampled data closely matches the test data, capturing the onset and extent of the wake from approximately 2000 m to 8000 m. This suggests that the model performs better in reproducing SBL wake structures, likely due to the more stable and less chaotic flow regime. Unlike the NBL case, the magnitudes of the sampled data's VD are closer to that of the test data here, which can be attributed to the fact that in this stable case it is easier for the model to distinguish wakes and turbulence from each other.

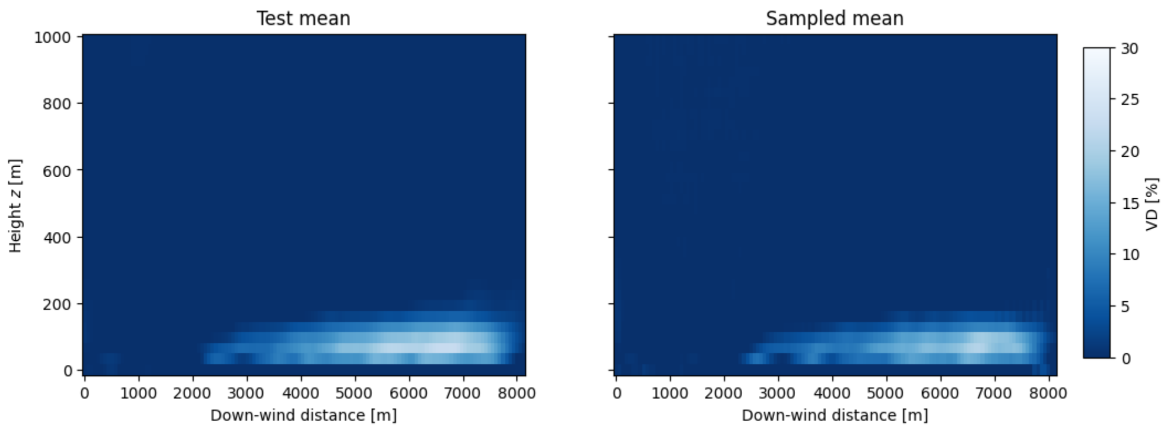

Figure 5-13: Comparison of the mean velocity deficit (VD) curtain for test data and generated samples under SBL conditions.

The results in this section show that the model is also capable of reconstructing turbine influenced flow fields in a SBL, extending its applicability beyond the neutral cases analyzed previously. The generated samples accurately reflect the expected low turbulence environment, capture turbine wake structures, and align well with the vertical profiles of the test data. This demonstrates that the model is able to adapt to fundamentally different ABL regimes when presented in isolation. However, for realistic deployment, the model must also be able to distinguish and reconstruct both NBL and SBL cases when they are present within the same dataset. The ability to resolve regime specific structures without explicit separation is essential for generalization, and will be evaluated in the next section. Other plots for this dataset can be found in Appendix A

5-4 Turbine Influenced Dataset containing an NBL and SBL

The previous results have shown that our model is able to reconstruct a turbine influenced flow field for both a SBL and NBL seperately. However, to ever be applicable for real-life use the model should not only be able to reconstruct these seperate ABL states but also identify in which state the ABL is and reconstruct the correct flow field. Whether the model is able to will be investigated here. Since previous work has already demonstrated that the model can reconstruct flow fields that are visually similar to the test data when trained and tested on separate datasets, this will not be repeated here, but the plots are shown in Appendix A. This section will thus focus on showing that the model is able to distinguish the two ABL states from each other using the R_b and will show that this final model performs better on this dataset than the models from the previous sections.

5-4-1 Quantitative Inspection

We will start of by qualitatively showing that the model can not distinguish the two ABL states from each other when not being fed with R_b label. This is done in figure 5-14 where we can see a vertical profile and the blue lines represent the SBL, the green lines the NBL, and the orange lines the sampled data without labels. The vertical profiles of the sampled data fall inbetween that of the NBL and SBL states showing that the model can not distinguish the two states from each other which is why the sampled data's profile lies in between those states.

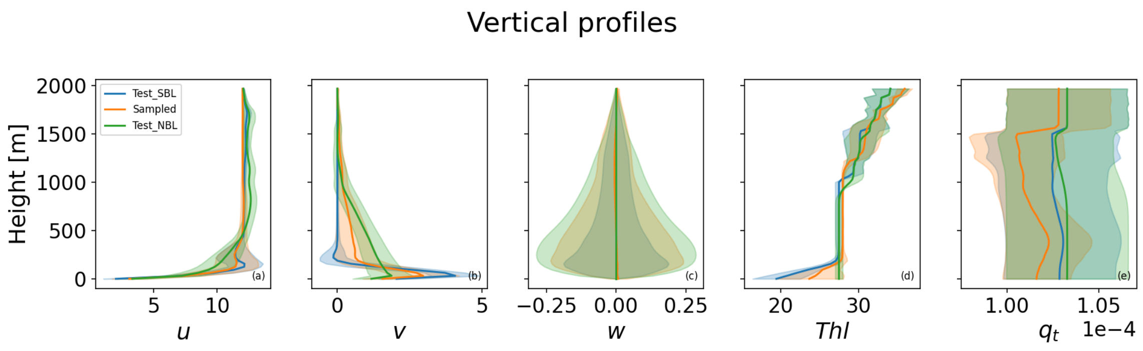

Figure 5-14: Figure showing the horizontally averaged vertical profiles for all five variables of the test and the sampled data for the mixed case without labels, the shaded areas represent the standard deviation from the mean, `Exp4-∅,S`.

Figures 5-15 and 5-16 show the vertical profiles for the SBL and NBL states respectively when the model is informed about the ABL state through the R_b number, clearly showing improvements.

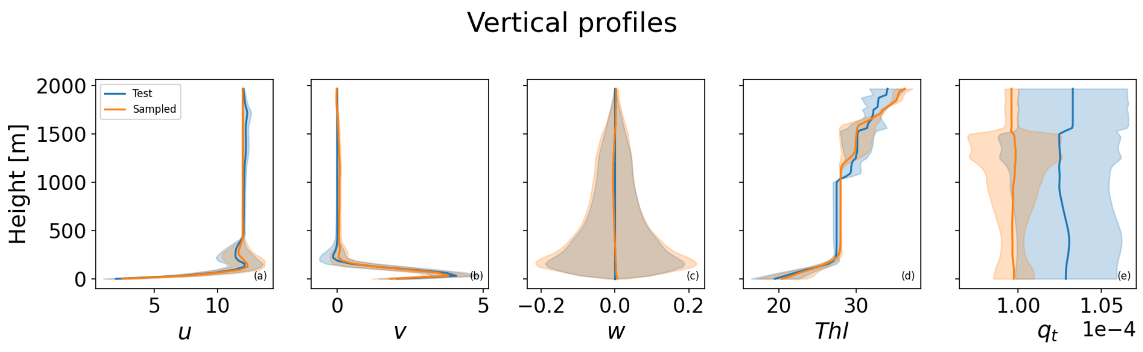

Figure 5-15: Figure showing the horizontally averaged vertical profiles for all five variables of the test and the sampled data in the mixed dataset for the SBL case, the shaded areas represent the standard deviation from the mean, `Exp4-BR,S`.

To quantitatively assess the reconstruction performance, Table 5-2 reports the integrated profile error (Equation 5-1) between test and sampled data for each variable. The results show that the model conditioned on the label achieves lower errors across all variables except for q_t. The increase in q_t error suggests that the model still struggles to accurately reconstruct this variable. Despite this, the overall mean error remains slightly lower with label conditioning, indicating that the added label provides an improvement in reconstruction accuracy for most flow variables.

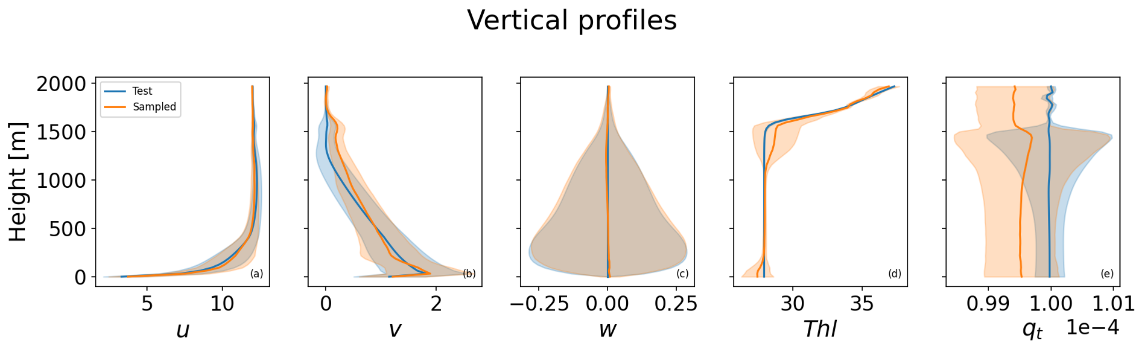

Figure 5-16: Figure showing the horizontally averaged vertical profiles for all five variables of the test and the sampled data in the mixed dataset for the NBL case, the shaded areas represent the standard deviation from the mean, `Exp4-BR,NH`.

Table 5-2: Vertical profile errors for the turbine influenced NBL with and without labels.

Model	u	v	w	T_{hl}	q_t	Mean
Mixed without label	0.55	1.5	0.035	1.60	1.80	1.10
Mixed with label	0.52	0.52	0.034	1.10	2.87	1.01

5-4-2 Physical Analysis

For this dataset we would like to investigate if the model's outputs adhere to the two laws of physics also discussed in section 5-1-2.

Figure 5-17 shows the turbulence energy spectrum for the test and sampled data. Both follow the expected $-5/3$ slope approximately between $k_x = 0.001$ and $k_x = 0.01$, indicating that the model reproduces the correct spectral energy distribution across those scales. Beyond $k_x = 0.015$, the test spectrum continues to decay while the sampled spectrum flattens, suggesting that high-frequency noise or insufficient dissipation may remain in the generated fields.

Figure 5-18 assesses mass conservation by plotting the divergence distribution. Ideally, incompressible flow would yield a sharp peak at $\nabla \cdot \mathbf{u} = 0$. The sampled data shows a broader and lower peak compared to the test data, indicating that although the model approximates divergence free behavior, some residual noise or imperfect reconstruction causes slight deviations from strict mass conservation.

In summary, this section demonstrated that conditioning the model on the R_b enables it to distinguish between different ABL states specifically the NBL and SBL. Without this label, the model produces flow reconstructions that lie between the two regimes, indicating an inability to separate them. With R_b provided, however, the model generates more accurate, state specific reconstructions, as reflected in improved vertical profiles and lower reconstruction errors for most variables. This capability marks an important step toward generalizing to realistic, turbine influenced flow fields across varying ABL conditions. While challenges remain particularly in capturing scalar fields like q_t and achieving full physical consistency this approach moves us closer to robust real-world applicability.

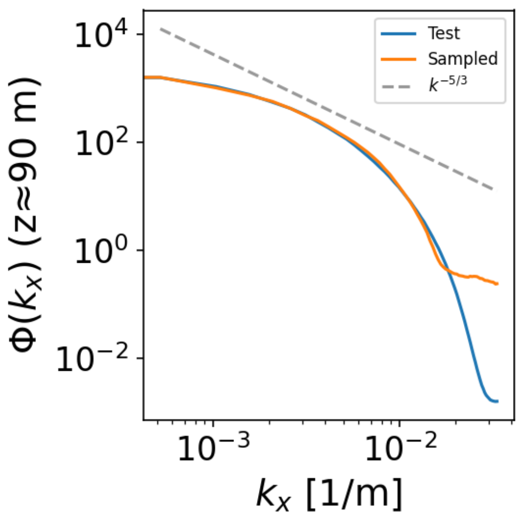

Figure 5-17: This figure shows the turbulence energy spectrum of the test and sampled data compared to the energy cascade with a spectral slope of -5/3 for the mixed case, `Exp4-BR,M`.

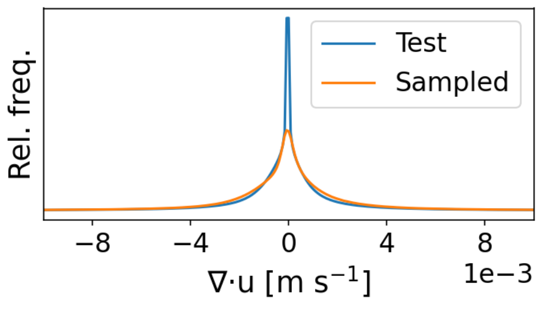

Figure 5-18: This figure shows how the test and sampled data adhere to the conservation of mass, total conservation of mass would lead to one vertical line at $\nabla u = 0$, `Exp4-BR,M`.

5-5 Model Comparisons

To conclude this chapter, Table 5-3 presents a direct comparison of the profile integral errors across all developed models when evaluated on the mixed, turbine influenced dataset. The results highlight a clear trend: each addition of complexity whether through inclusion of turbine effects, atmospheric variability, or conditioning on physical labels leads to a notable reduction in error. The final model, trained on both NBL and SBL states and conditioned on R_b and BLH, achieves the lowest mean error across all key variables. These findings confirm that the integration of physical labels and diverse boundary layer regimes enables the model to generalize better and produce more physically accurate reconstructions. While full real-world applicability still requires further improvement, this final model represents a meaningful step toward to robust, label aware flow field reconstruction in realistic ABL scenarios.

Table 5-3: Profile-integral errors for the different models used.

Model	u	v	w	T_{hl}	q_t	Mean
NBL without turbines	1.36	5.13	0.032	7.49	3.56	3.51
NBL with turbines	1.29	2.18	0.060	4.41	3.78	2.34
SBL with turbines	1.31	1.53	0.034	4.31	5.81	2.60
Mixed	0.52	0.52	0.034	1.10	2.87	1.01

Chapter 6

Conclusion and Recommendation

6-1 Conclusions

This master's thesis has investigated the use of LDMs for reconstructing physically plausible flow fields in the ABL using sparse measurements. This work was motivated by the need for improved flow reconstruction around wind farms to better support wind energy applications and a more precise LES initialization. While previous works had already reconstructed the ABL in a NBL state, this research focused on extending the model used to be able to distinguish multiple ABL states, that were also influenced by wind turbines, in the best way possible. This led to the research question: *"How can a diffusion model reconstruct turbine influenced flow fields across multiple ABL states?"*

In order to get a better understanding of the model's behavior under different conditioning circumstances, an extra analysis was done where the variance between samples conditioned on the same input was studied. This analysis and the research question led to the following conclusions.

Firstly, it can be concluded that DMs can be used to reconstruct turbine influenced ABLs. This work noticed that even though other research had reconstructed flow fields in the ABL, it had not been done with turbine influenced flows yet. For these models to be used for wind energy applications it is important, ofcourse, that the model can actually reconstruct flows with wind turbines inside them. The model showed good performance doing this, reconstructing the wakes of the turbines well in most cases. This was most apparent in the SBL state because of the low turbulent nature of this state which allowed the wakes to be analyzed well.

Secondly, it can be concluded that DMs can, other than the wind velocity components u, v and w, also reconstruct the moist-static-energy temperature T_{hl}. Previous works had shown their ability to reconstruct all three wind components. However, T_{hl} and q_t had never been reconstructed in the ABL using a DM before, even though this is of importance for one of the interests of this research, which is to see whether DMs can reconstruct flow fields in order to later use as initial conditions for LES. Commonly, LES simulations do require these two variables as initial conditions, so to be sure that the flows reconstructed by the model could, at some point, be used as initial conditions. This research has found that DMs are able to reconstruct T_{hl} well, but has more difficulty with reconstructing q_t. However, this difficulty could be attributed to the fact that the input fields for q_t are unrealistic and not necessarily the model's inability to learn this variable.

Thirdly, it can be concluded that the DM can distinguish different ABL states and BLHs from each other using labels such as the Rib and the BLH. This work introduced adding global information to the model, in the form of a label, to supply it with additional information. This is an essential feature of the proposed model that could maybe be extended to other labels to supply the model with information not available in the local, spatial measurements.

Finally, it can be concluded that the model is able to reconstruct turbine influenced flows across different ABL states to some extent. The evaluation of the model's results were promising, showing good performance in the reconstruction of key structural features of the ABL, including wind turbine wakes and inversion layers. Statistically, the generated samples matched the vertical profiles and PDFs of the train and test data reasonably well, although an error for q_t was noticed. Physical realism was assessed via the turbulence spectrum and mass conservation. While the energy spectrum followed the expected cascade at large scales, excess energy at small scales indicated residual numerical noise. Similarly, the divergence of the velocity field was not zero, but this mirrored the non conservative nature of the training data itself. These results were accomplished by applying structured guidance to the model. Lastly, analyses into the VDs of the flow fields showed that the model is able to capture the general structures in wakes while missing some fine-scale details in the NBL case. This thesis has shown that with the right conditioning DMs are able to reconstruct turbine influenced flows across multiple ABL states, even though there is still a lot of improvement possible.

6-1-1 Future Potential

While this thesis focused on evaluating the LDM across specific experimental conditions, the broader implications extend beyond the immediate findings. The model's ability to reconstruct realistic ABL flow fields under varying atmospheric conditions demonstrates strong potential for scaling the approach toward real-world forecasting applications. As more LES data becomes available capturing diverse ABL regimes and terrain types the conditional diffusion model can be trained on increasingly realistic scenarios. This opens the door to using such models for short-term wind energy forecasting. Ultimately, this research supports the long-term vision of integrating generative models into the wind energy pipeline, where they can provide fast, physics consistent reconstructions from sparse measurements.

It is important to note, however, that while the LDM produces flow fields that are statistically and physically plausible, each output is one of many possible realizations consistent with the input data. In practice, multiple samples may reproduce the same large-scale wake structure but differ in small-scale turbulent features. This limits the immediate applicability of the model for real-time turbine control, where precise knowledge of inflow turbulence is critical for wake steering and load mitigation. Therefore, while the model shows clear promise for forecasting, resource assessment, and LES initialization, further refinement would be required before it can be reliably used for high-resolution control purposes.

6-2 Recommendations

While the results from this thesis are a significant improvement of previous works looking into flow reconstruction in the ABL, they also highlight areas for future research. Recommendations are split into 3 categories: Improvement of the current configuration, making the setup more realistic and future applications.

6-2-1 Configuration Improvements

The focus of this subsection is possible improvements to the dataset used, the way data was processed and the conditioning of the model to improve overall performance.

Firstly, we recommend recreating the dataset in order to obtain a realistic q_t. As has been mentioned before the model was not able to reconstruct the vertical profiles for the variable q_t well, which could be due to the fact that the input data's profiles for this variable are not realistic. To find out whether the problem lies with the model or this unrealistic behavior we thus recommend creating a dataset with a realistic profile.

Secondly, we recommend further investigating the roll of the mask in guiding the model's outputs. As mentioned, there are multiple ways to incorporate mask information into the model, such as hard constraining the output at masked locations or concatenating the mask to the input. This work adopts the latter approach, which leads to outputs that are consistent with the measurements but not guaranteed to exactly match them. For applications where accurate reconstruction at the measurement locations is critical,such as wake control, it would be worthwhile to explore alternative strategies for integrating mask information more directly.

6-2-2 Setup Realism

This subsection focuses on making the setup used to mimic real-life data assimilation using sparse measurements at a wind farm more realistic.

Firstly, we recommend extending the dataset to a wider range of states to research whether the model is still able to distinguish certain states from each other. Right now the used dataset consists of a SBL and NBL where, except for the mean wind direction, the flow does not really change within each state. To study whether the model is able to reconstruct the flow in a more realistic setup this dataset could be extended. This can be done in multiple ways, firstly by adding a CBL to the dataset. When this state is added the three main ABL states are present in the dataset. This is important to check whether the model is able to distinguish all three ABL states from each other. Furthermore, the current dataset consists of two different ABL states, but in the ABL state there is not much change. It would be interesting to see if the model can still cope when, for example, there are NBLs with multiple boundary layer heights, wind speeds and wind directions, which it could be informed off through adding more labels. If the model is still able to reconstruct the flow in that setup a good final step of checking whether the model can reconstruct real-life weather is using nonperiodic LES data with ERA5 boundary conditions as mentioned in section 4-1-1. This data is a lot more variable and covers all different states the ABL has, if the mode is able to reconstruct data generated this way we can say it is ready to do it in real-life.

Secondly, we recommend using different masks to improve the realism of the setup with respect to the measurements available at a wind farm. Currently the masked being used for the final experiments is shown in figure 4-1. Even though this mask only has information on about 0.15% of the domain, which is little, this is still more than is present at a lot of wind farms in real-life. A lot of wind farms only have anenometers on each turbine nacelle, which for the set up we used would be equal to only knowing the wind speeds in 36 points throughout the entire domain. The currently used mask has about 1500 points at which it measures the variables. Seeing how the model behaves when conditioned on sparser masks is thus an interesting study. Furthemore, the mask currently used also measures q_t and T_{hl} in every point it measures the wind speed, this is often not possible in real-life so it would be interesting to see how the model reconstructs these variables when they are only given in, for example, the meteorological mast present in the mask.

6-2-3 Future Applications

This subsection states an interesting way the reconstructed flow fields can be used.

It is recommended to look into the use of the reconstructed flow fields for an Ensemble Kalman-Filter (EnKF). When there is enough confidence in the model to reconstruct realistic weather conditions with realistic measurements the reconstructed flow fields can be used as observations in an EnKF, where the underlying model is an LES model. In this setup the DM's output can be used as observations used to guide an ensemble of LES runs. This approach allows the EnKF to correct the LES state towards the model's inpainted measurements, combining the physical realism of the LES with the real-life measurements of the DM. If implemented correctly this method would give realistic simulation of the flow happening at a wind farm providing essential information for either turbine control or wind resource assessment.

6-3 Final Words

In conclusion, this thesis addresses shortcomings in the literature by showing a DM's ability to reconstruct turbine influenced flow fields across multiple ABL states. An essential feature for the future applicability of DMs in data assimilation and wind farms. Furthermore, this the framework proved that a DM can also reconstruct the liquid potential temperature, while having issues with the specific humidity and showed the power of adding global labels the model in order to further guide it towards a specific flow field.

Appendix A

Additional Analysis Plots

This appendix contains plots that were not essential for the normal thesis but could still give better insights on the performance of the model. This chapter has been split up into the four different cases as discussed in chapter 5.

A-1 NBL

Figure A-1 shows the visual inspection of the model comparing the model outputs with its input.

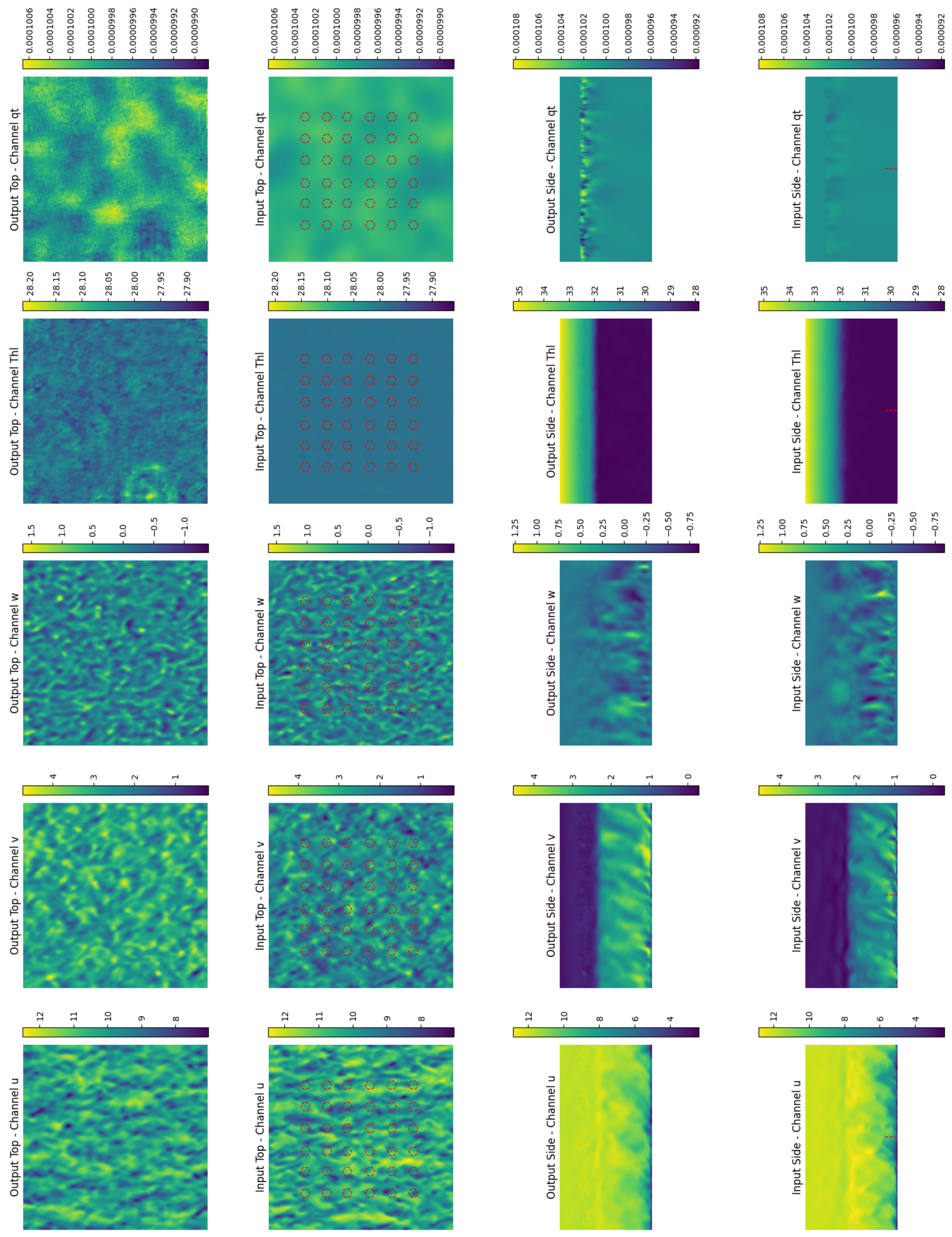

Figure A-1: Figure showing the visual inspection of the output compared to the input in the NBL case

A-2 Turbine Influenced NBL across 2 BLHs

In chapter 5 some analyses for turbine influenced NBl case were left out and will be shown in this section. Firstly, figure A-2 contains the visual inspection for all five variables of the output compared to that of the input.

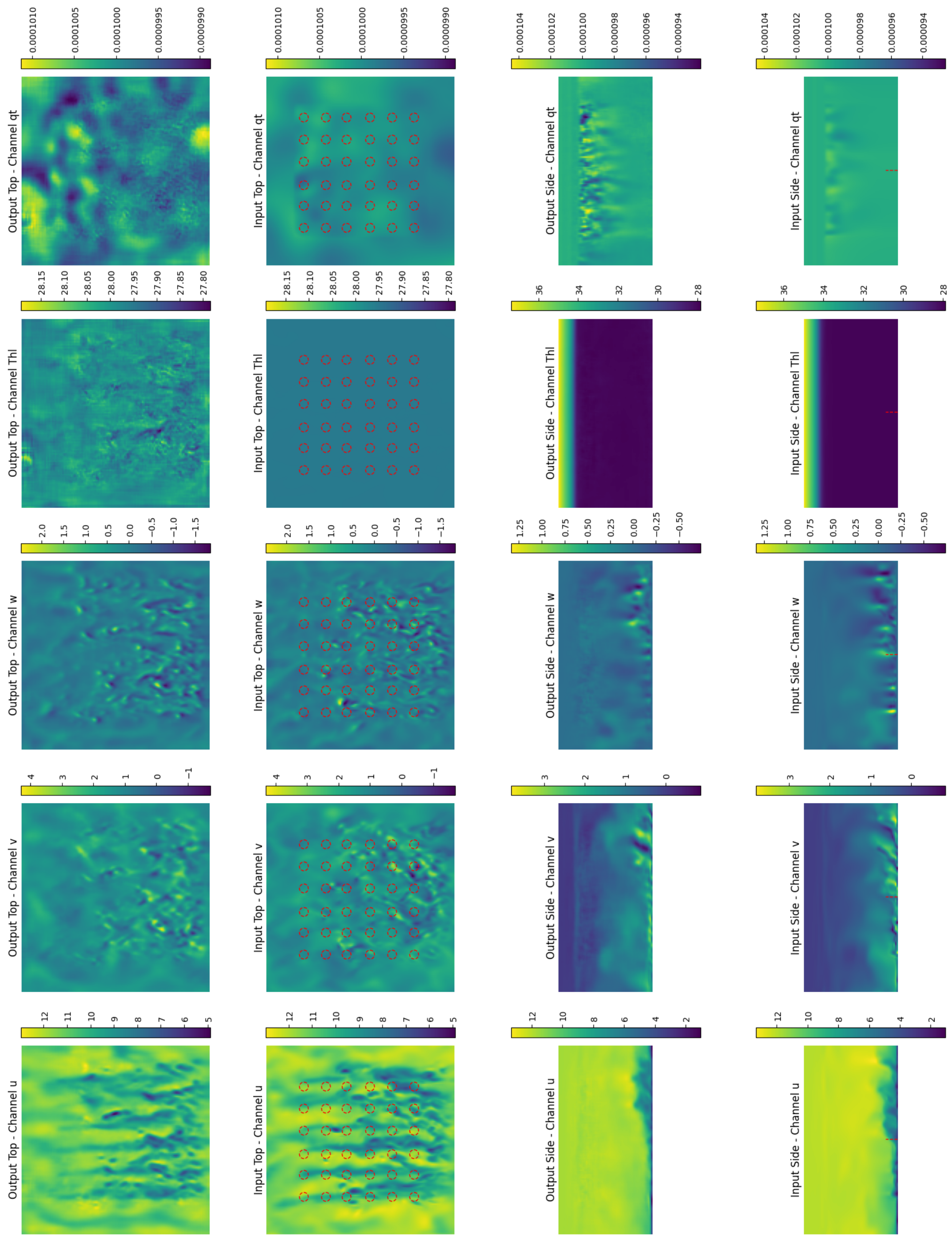

Figure A-2: Figure showing the visual inspection of the output compared to the input in the NBL case

Secondly, figure A-3 contains the PDFs of the test data compared to that of the sampled data.

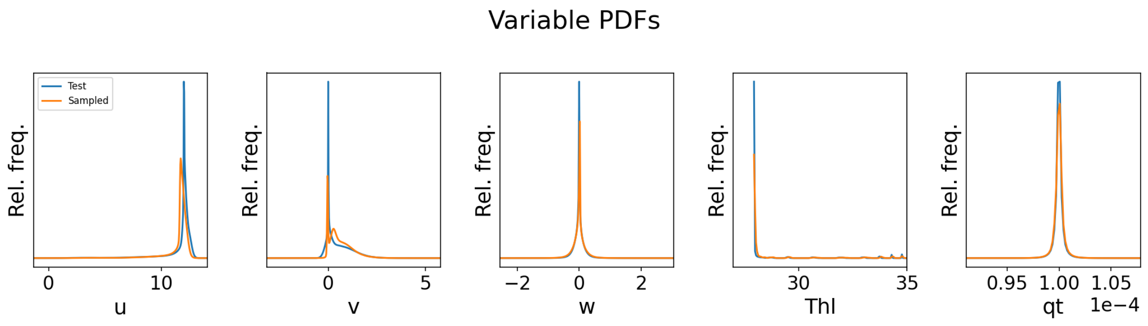

Figure A-3: Figure showing the PDFs for all five variables of the test and sampled data for the turbine influenced NBL case.

Lastly, figures A-4 and A-5 show the turbulence energy spectrum and the conservation of mass respectively.

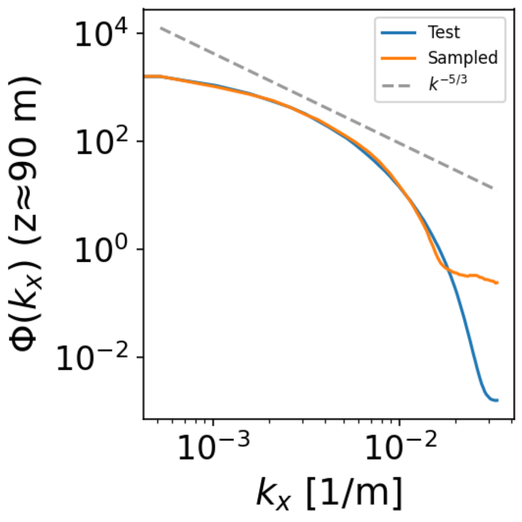

Figure A-4: This figure shows the turbulence energy spectrum of the test and sampled data compared to the energy cascade with a spectral slope of -5/3 for the turbine influenced NBL case.

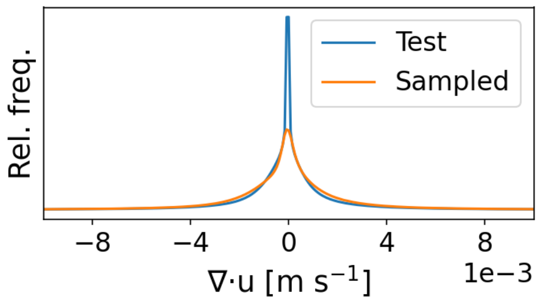

Figure A-5: This figure shows how the test and sampled data adhere to the conservation of mass, total conservation of mass would lead to one vertical line at $\nabla u = 0$.

A-3 Turbine Influenced SBL

In chapter 5 some analyses for turbine influenced SBl case were left out and will be shown in this section. Firstly, figure A-6 contains the visual inspection for all five variables of the output compared to that of the input.

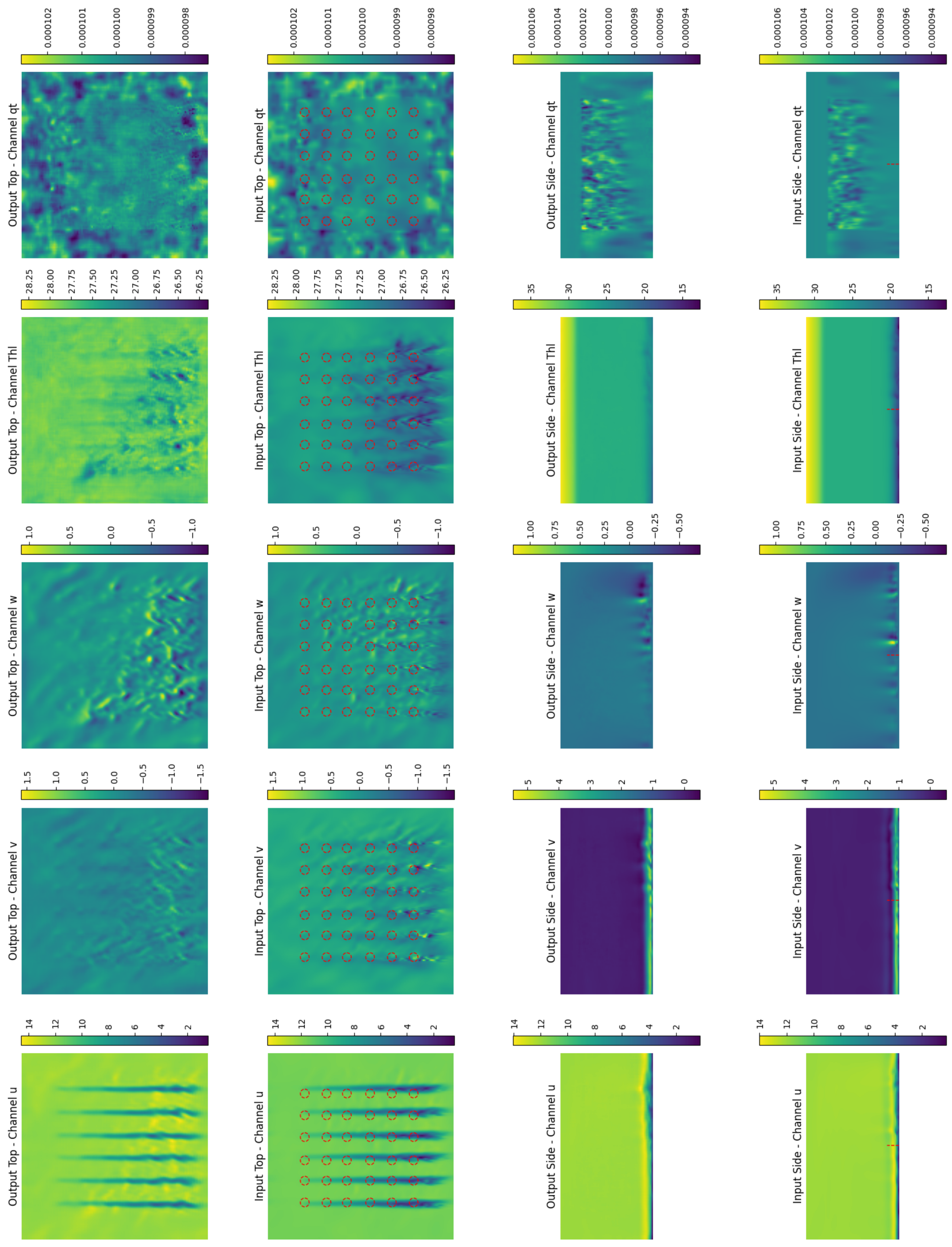

Figure A-6: Figure showing the visual inspection of the output compared to the input in the NBL case

Secondly, figure A-7 contains the PDFs of the test data compared to that of the sampled data.

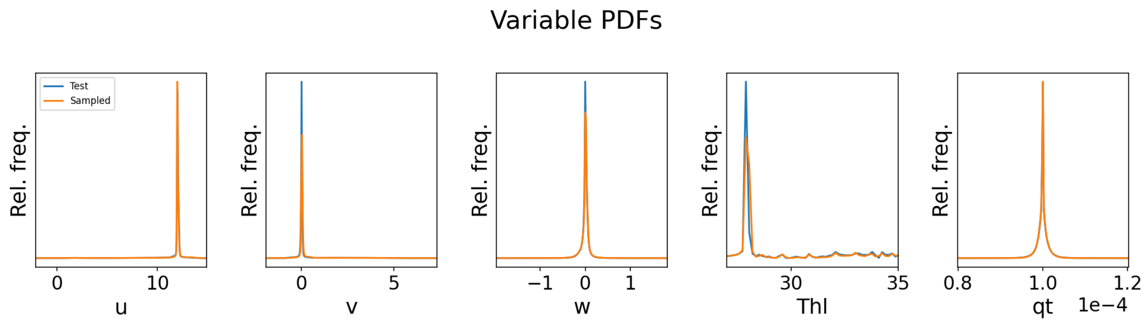

Figure A-7: Figure showing the PDFs for all five variables of the test and sampled data for the turbine influenced SBL case.

Lastly, figures A-8 and A-9 show the turbulence energy spectrum and the conservation of mass respectively.

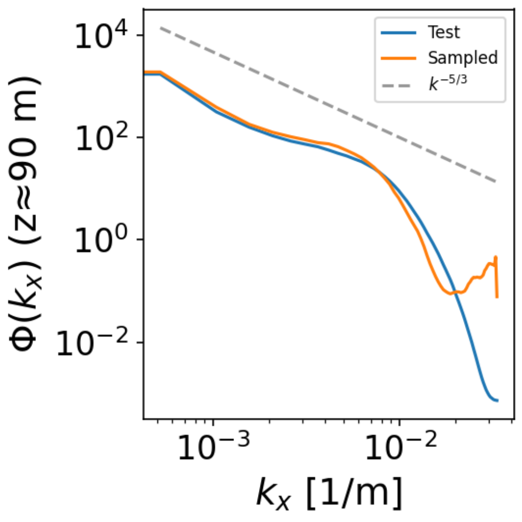

Figure A-8: This figure shows the turbulence energy spectrum of the test and sampled data compared to the energy cascade with a spectral slope of -5/3 for the turbine influenced NBL case.

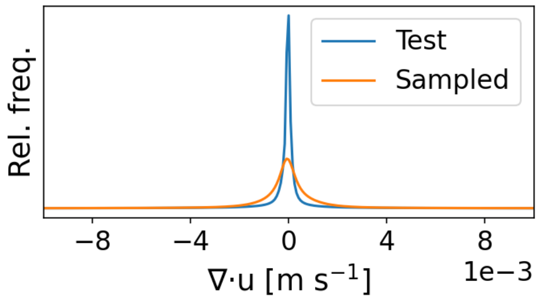

Figure A-9: This figure shows how the test and sampled data adhere to the conservation of mass, total conservation of mass would lead to one vertical line at $\nabla u = 0$.

A-4 Turbine Influenced Mixed Dataset Containing SBL and NBL

In chapter 5 some analyses for the turbine influenced mixed dataset were left out and will be shown in this section. Firstly, figure A-10 contains the visual inspection for all five variables of the output compared to that of the input for the SBL case.

Secondly figure A-11 contains the visual inspection for all five variables of the output compared to that of the input for the NBL case.

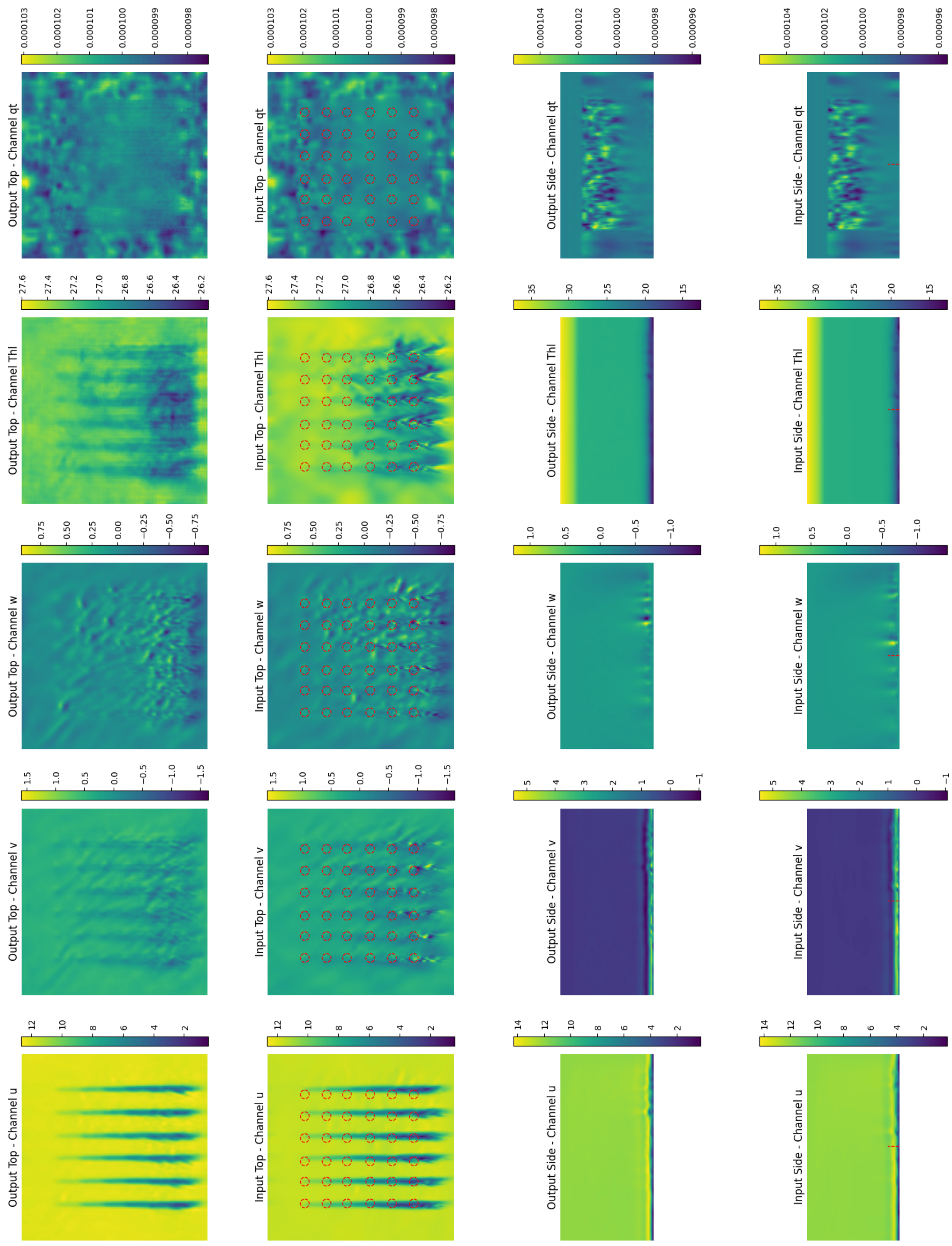

Figure A-10: Figure showing the visual inspection of the output compared to the input in the NBL case

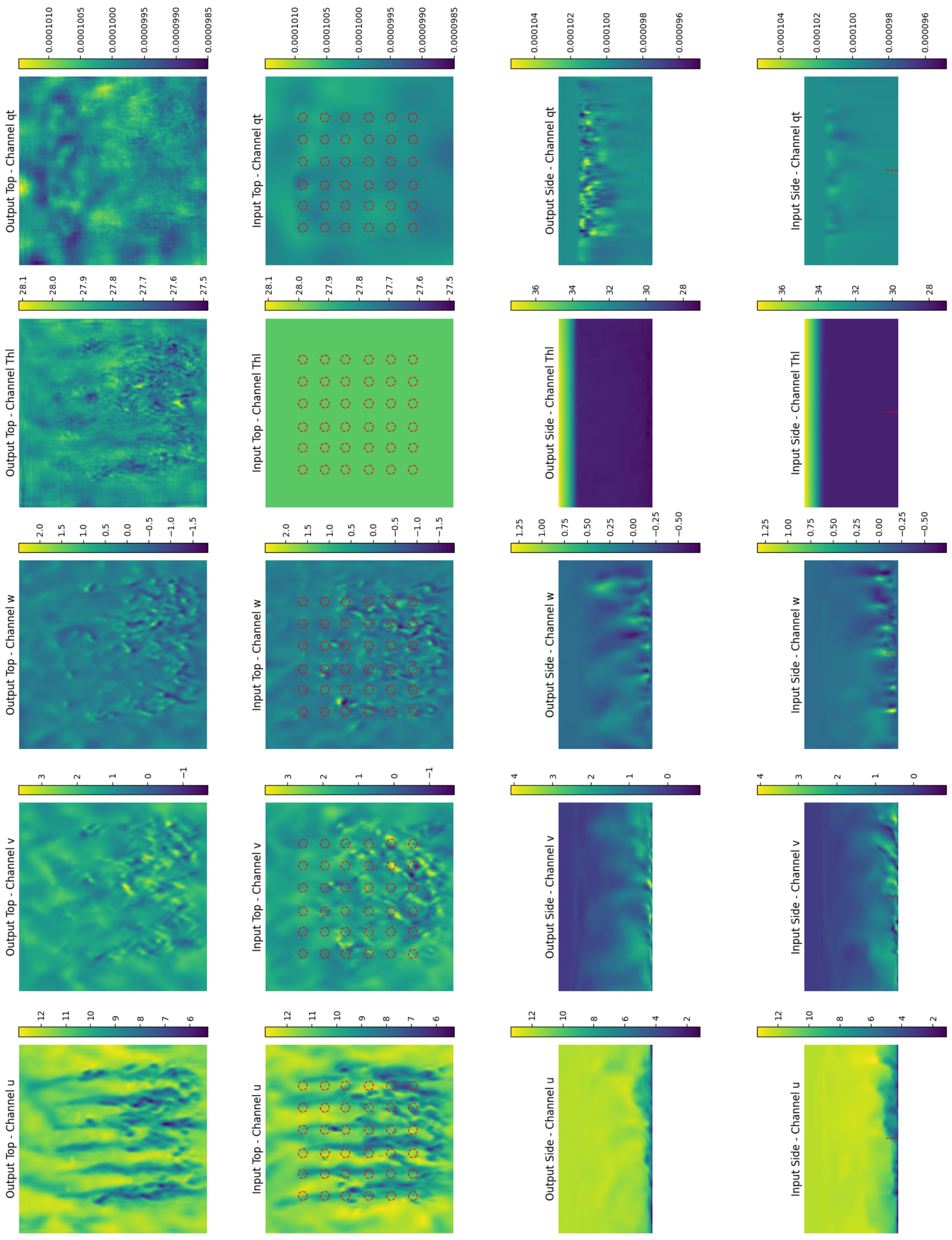

Figure A-11: Figure showing the visual inspection of the output compared to the input in the NBL case

Figures A-12 and A-13 show the PDFs for the SBL and NBL respectively.

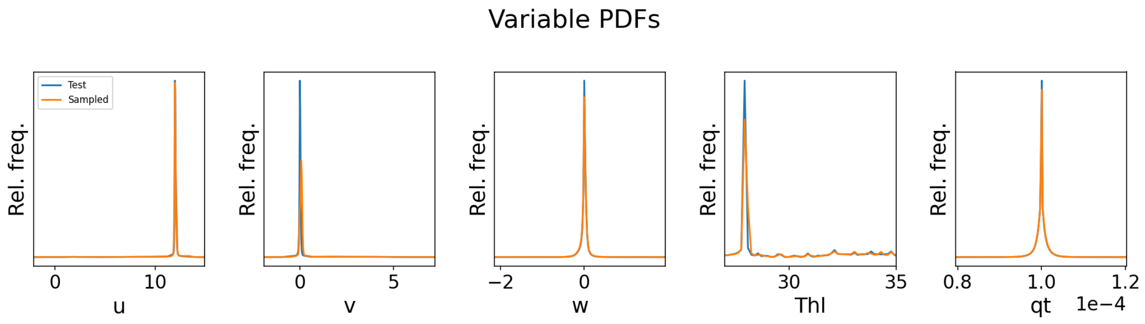

Figure A-12: Figure showing the PDFs for all five variables of the test and sampled data for the turbine influenced mixed dataset's SBL case.

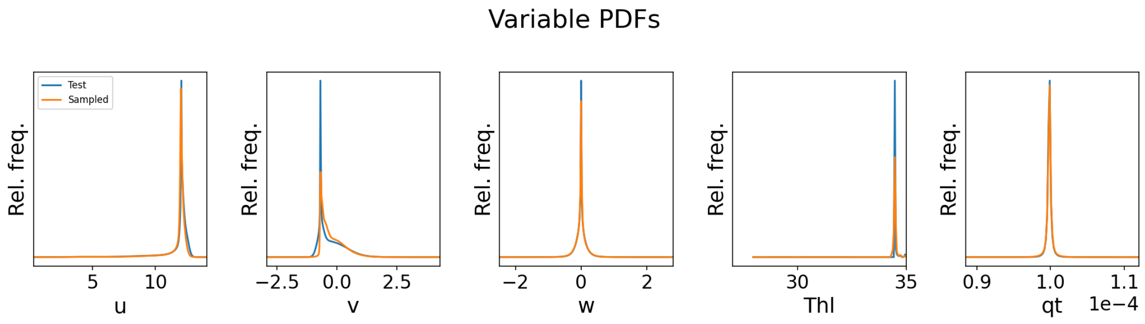

Figure A-13: Figure showing the PDFs for all five variables of the test and sampled data for the turbine influenced mixed dataset's NBL case.

Bibliography

[1] International Energy Agency (IEA). hart: Renewable power generation by technology in the net zero scenario, 2010- 2030., 2022.

[2] Piotr Tryjanowski and Andrzej Pawlewicz. Constraints on the development of wind energy in rural areas: The case of poland. *PLOS ONE*, 18(11):e0294563, 2023.

[3] Anna-Kaisa Kosenius, Matleena Kniivilä, and Markku Ollikainen. Forest fragmentation and biodiversity: Land-use effects of wind power in finland. *Environment, Development and Sustainability*, 26:8192–8212, 2024.

[4] Nicolai G. Nygaard and Anna C. Newcombe. Wake effects from large offshore wind farms observed using scanning lidars. *Proceedings of the National Academy of Sciences*, 118(36):e2103875118, 2021.

[5] Johan Meyers, Wim Munters, and Claire VerHulst. Wind farm control: prospects and challenges. *Wind Energy Science*, 7(5):2271–2294, 2022.

[6] Tobias Horlbeck, Joachim Peinke, and Michael Heisel. Data-driven wind farm flow control and challenges towards field implementation. *Renewable and Sustainable Energy Reviews*, 184:113618, 2025.

[7] The Times. Britain's power grid was on the brink due to poor wind forecasts, national grid admits, 2025. Accessed July 2025.

[8] Lin Sun, Wei Zhang, and Yuting Li. A comprehensive review of forecasting methods for renewable energy integration. *Discover Energy*, 5(1):25, 2025.

[9] Akash Sharma and Min Liu. Short-term forecasting for load and renewable energy balancing in smart grids, 2025. arXiv preprint.

[10] Roland B. Stull. *An Introduction to Boundary Layer Meteorology.* Kluwer Academic Publishers, Dordrecht, 1988.

[11] N.Katopodes. *Free-surface Flow.* Butterworth-Heinemann, 2019.

[12] G.Lukaszewicz and P.Kalita. *Navier-Stokes Equations*. Springer, 2016.

[13] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[14] David C. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, 3rd edition, 2006.

[15] J.Smagorinsky. General circulation experiments with primitive equations. *Monthly Weather Review*, 91(3), 1963.

[16] U.Piomelli. Large-eddy simulation: achievements and challenges. *Progress in Aerospace Sciences*, (35), 1999.

[17] I.Goodfellow et al. Generative adversarial networks. *Communications of the ACM*, 2020.

[18] A.Razavi et al. Generating diverse high-fidelity images with vq-vae-2, 2019.

[19] J.Sohl-Dickstein et al. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[20] W.Sun JH.Bastek, D.Kochmann. Physics-informed diffusion model, 2023.

[21] A.Farimani D.Shum, Z.Li. A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 2023.

[22] A.Rybchuk, M.Hassanaly, N.Hamilton, P.Doubrawa, M.Fulton, and L.Martinez-Tossas. Ensemble flow reconstruction in the atmospheric boundary layer from spatially limited measurements through latent diffusion models. *Physics of Fluids*, 2023.

[23] P. Doubrawa C. L.Kelley and J. W. Naughton. Rotor, aeroelastics, aerodynamics, and wake (raaw) project, 2023.

[24] H. Richardson, S. Basu, and A. A. M. Holtslag. Improving stable boundary-layer height estimation using a stability-dependent critical bulk richardson number. *Boundary-Layer Meteorology*, 148:93–109, 2013.

[25] G. Rampanelli and D. Zardi. A method to determine the capping inversion of the convective boundary layer. *Journal of Applied Meteorology*, 43(6):925–933, 2004.

[26] S. H. Derbyshire, I. Beau, P. Bechtold, J. Y. Grandpeix, J. M. Piriou, J. L. Redelsperger, and P. M. M. Soares. Sensitivity of moist convection to environmental humidity. *Quarterly Journal of the Royal Meteorological Society*, 130(604):3055–3079, 2004.

[27] R.Rombach, A.Blattman, D.Lorenz, P.Esser, and B.Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[28] J.Kautz A.Vahdat, K.Kreis. Score-based generative modeling in latent spac, 2021.

[29] M.Heusel et al. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017.

[30] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[32] Hengshuai Yao, Dong-lai Zhu, Bei Jiang, and Peng Yu. Negative log likelihood ratio loss for deep neural network classification. In *Proceedings of the Future Technologies Conference (FTC 2019)*, pages 276–282. Springer, October 2019.

[33] Solomon Kullback and Richard A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[34] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[35] j.Ho, A.Jain, and P.Abbeel. Denosing diffusion probabilistic models, 2020.

[36] A.Nichol and P.Dhariwal. Improved denosing diffusion probabilistic models, 2021.

[37] A.Nichol and P.Dhariwal. Diffusion models beat gans on image synthesis, 2021.

[38] N.Siddique et al. U-net and its variants for medial image segmentation: A review of theory and applications, 2021.

[39] D.P.Kingma and M.Welling. Auto-encoding variational bayes, 2022.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30. Curran Associates, Inc., 2017.

[41] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11451–11461. IEEE, 2022.

[42] Ciprian Corneanu, Raghudeep Gadde, and Aleix M. Martinez. Latentpaint: Image inpainting in latent space with diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4334–4343, January 2024.

[43] J.Song, C.Meng, and S.Ermon. Denoising diffusion implicit models. *ICLR*, 2021.

[44] S.Ermon Y.Song. Improved techniques for training score-based generative models, 2020.

[45] X.Zhao J.Zhang. Three-dimensional spatiotemporal wind field reconstruction based on physics-informed deep learning. *Applied Energy*, 2021.

[46] X.Zhao J.Zhang. Spatiotemporal wind field prediction based on physics-informed deep learning and lidar measurements. *Applied Energy*, 2021.

[47] A.Rybchuk et al. Ensemble flow reconstruction in the atmospheric boundary layer from spatially limited measurements through latent diffusion models. *Physics of Fluids*, 2023.

[48] A.Vaswani et al. Attention is all you need, 2017.

[49] B.Mildenhall et al. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[50] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[51] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[52] A. P. Siebesma, A. A. Wyszogrodzki, A. Brast, H. J. J. Jonker, A. Chlond, F. Couvreux, and J. Teixeira. The dales model: A large-eddy simulation approach to study atmospheric boundary layer dynamics. *Geoscientific Model Development*, 3(2):415–444, 2010.

[53] Dries Allaerts and Johan Meyers. Gravity waves and wind-farm efficiency in neutral and stable conditions. *Boundary-Layer Meteorology*, 166(2):269–299, 2018. Published online 13 October 2017; Open access.

[54] Hersbach, H., Bell, B., Berrisford, P., and et al. Era5 reanalysis data. https://www.ecmwf.int/en/forecasts/dataset/ecmwf-reanalysis-v5, 2018. Copernicus Climate Change Service (C3S) Data. Accessed: 2025-05-08.

[55] Whiffle B.V. Advanced topics: Meso simulation. https://docs.whiffle.cloud/advanced-topics/meso-simulation, 2024. Accessed: 2025-05-08.

Glossary

List of Acronyms

ABL	Atmospheric Boundary Layer
LDM	Latent Diffusion Model
DM	Diffusion Model
LES	Large Eddy Simulation
VAE	Variational AutoEncoder
GANS	General Adversarial Network
RANS	Reynold's Averaged Navier-Stokes
DNS	Direct Numerical Solution
DDPM	Denosing Diffusion Probabilistic Model
CNN	Convolutional Neural Network
VD	Velocity Deficit
FID	Frechét Inception Distance