University of Oldenburg MEng Engineering Physics

Master's Thesis

Deep Learning-Based Ultra-Short-Term Probabilistic Wind Farm Model for Wake Steering Control

Submitted by: Juan Manuel Boullosa Novo

First supervisor: Prof. Dr. Martin Kühn

Second supervisor: Dr. Sc. Vlaho Petrović

Oldenburg, 18th August 2025

Abstract

The power output of wind farms decreases by 15–25% because of wake effects, which occur when upstream turbines reduce the velocity of air that reaches downstream units. Current industry-standard control strategies operate reactively, using only present wind measurements despite slow turbine yaw actuators (0.3–0.5°/s) requiring 60–180 seconds for realignment. The time difference between wind speed measurements and control decisions requires ultra-short-term wind forecasting (1–5 minutes ahead) for proactive control decisions. The research establishes a complete open-source system which enables the implementation of advanced deep learning models into operational wind farm forecasting systems to connect laboratory prototypes with commercial deployment.

The platform runs a complete end-to-end pipeline which includes OpenOA for data preprocessing quality control, Optuna with Tree-structured Parzen Estimators for automated hyperparameter optimization, PyTorch Lightning for distributed training infrastructure and efficient inference systems for real-time control applications. The modular architecture separates concerns across four repositories, enabling independent development and rapid model substitution. The TACTiS-2 (Transformer-Attentional Copulas for Time Series) model was adapted and integrated as the primary forecasting engine, employing a dual-encoder architecture with two-stage curriculum learning to separately model marginal distributions and dependencies. Validation on the SMARTEOLE dataset (7 turbines, 3 months) achieved reasonable deterministic performance (MAE: 0.633 m/s, RMSE: 0.812 m/s) and preserved spatial correlations (0.879 correlation score). The model faced difficulties with probabilistic calibration because 90% prediction intervals only included 5.15% of actual observations, which exposed basic mismatches between model assumptions and wind farm data features.

The platform shows that deep learning architectures can be used for operational wind forecasting even though uncertainty quantification is challenging. The infrastructure enables the reduction of implementation time from weeks to hours, which supports the development of systematic models and their comparison. All software is released open-source to facilitate reproducibility and future research in forecast-enabled wind farm control.

Acknowledgments

I want to thank all the people who have helped me during my master's thesis work. The completion of this work became possible through their guidance, encouragement and assistance.

I am deeply thankful to my main examiner, Prof. Martin Kühn, for this opportunity, and to my supervisor, Dr. Vlaho Petrović. His mentorship and support were fundamental to my academic journey. Through his efforts I gained the opportunity to conduct my thesis research at the University of Colorado Boulder, which I deeply appreciate.

My sincere appreciation goes to Prof. Lucy Pao for graciously hosting me at the University of Colorado Boulder. Her willingness to welcome me into her research group and provide the necessary resources was essential for the success of this thesis. I must also thank the Europe-Colorado Program that provided the financial support for my research stay.

A special and heartfelt thank you is owed to Aoife Henry. Her brilliant idea formed the very foundation of this project. I deeply appreciate the numerous days we worked together, her patience, and the collaborative spirit which made this research both productive and enjoyable.

Finally, I am grateful for the computing time granted on the High-Performance Computing (HPC) cluster STORM at the University of Oldenburg, which was crucial for the numerical simulations in this work.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Context

The transition from fossil fuels to renewable energy sources represents an essential infrastructure challenge that arises during this century. Wind energy stands as one of the most advanced technologies in its sector and plays a crucial role in achieving the proposed climate objectives. The global wind power infrastructure has surpassed 1200 GW of installed capacity since December 2024 (Global Wind Energy Council, 2024), thus becoming the dominant renewable energy system worldwide and domestically. The economic success of wind energy now depends on its ability to deliver power to the grid consistently as the power system faces various operational requirements. The development of improved wind farm control systems which optimize turbine operations will determine whether wind energy can challenge fossil fuels and nuclear alternatives for market dominance in upcoming years (Boersma et al., 2017).

The increased density of wind farms located in high-wind regions faces multiple efficiency barriers which prevent them from reaching their power potential while meeting grid requirements. The capacity factors of current wind turbines operate at between 25% to 45% because of various limiting factors. The power output decreases by 15 to 25% due to wake losses, turbines operate suboptimally during fast-changing conditions, and they follow conservative control methods that value component longevity and struggle to maintain consistent power delivery because of wind fluctuations.

Multiple control systems for wind farms operate to minimize their respective operational deficiencies. The industry-standard control method known as greedy control functions as the baseline since it operates by having each turbine point its blade directly into the wind stream to produce maximum power without accounting for the effects its wake creates on neighboring turbines (Simley et al., 2020). The implementation of look-up table (LUT) control methods advances farm-level optimization because precomputed tables from steady-state wake models enable wake steering through upstream turbine misalignment to deflect wakes from downstream units (Bossanyi, 2018). The implementation of LUT strategies follows greedy control in complexity since they operate through open-loop systems that react to current wind conditions.

Dynamic models within model predictive control (MPC) strategies allow optimization across multiple future time steps which enables better performance compared to other control approaches. Advanced methods require substantial computation and complex

implementation alongside reliability issues which restrict their use in commercial wind farms. The focus of this thesis is to improve the two dominant industry control methods, which are greedy and LUT control, because of their proven simplicity and reliability.

Both greedy and LUT control systems face a basic restriction because they operate reactively. These control strategies use only present wind measurements to determine their decisions instead of forecasting upcoming weather conditions. The operational time patterns of wind farms create significant challenges because of this issue. Large wind turbine yaw actuators exhibit slow speed behavior with a rate of 0.3–0.5 degrees per second (Simley et al., 2021) even though the underlying wind conditions shift unpredictably and rapidly. Control systems use wind measurements that undergo low-pass filtering for noise reduction, yet this filtering process extends the time needed to make decisions by 30–60 seconds after wind changes become detectable. When control actions are implemented based on filtered wind measurements, the actual wind conditions transform beyond the initial optimization point before slow actuators can respond.

The time difference between control requirements and mechanical yaw actuator response time demands wind condition predictions that extend from 1 to 5 minutes into the future (Sengers et al., 2023) to enable effective control operations. The ability to predict wind directions would enable greedy control systems to make anticipatory yaw adjustments that minimize turbine misalignment time during wind direction changes, thus enhancing energy capture while minimizing unnecessary yawing movements. The implementation of forecasts with LUT control enables operators to choose appropriate yaw offsets that correspond to expected future conditions instead of using current observations, which preserves the effectiveness of wake steering actions as conditions change.

The nature of wind uncertainty demands that predictive models include confidence level measures in their output. Through probabilistic forecasting, operators can make decisions based on risk awareness, leading to more cautious yaw adjustments during high uncertainty periods and stronger control actions during predictions with high confidence levels. The uncertainty quantification method provides essential value for implementing control strategies that keep performance steady regardless of prediction uncertainty. The controller can implement an adaptive 3D LUT method which combines wind speed and direction data with wind direction uncertainty predictions to select appropriate yaw offsets (Simley et al., 2020).

This master thesis presents the TACTiS-2 (Transformer-Attentional Copulas for Time Series) model (Ashok et al., 2024) as an advanced deep learning solution for ultra-short-term probabilistic spatio-temporal wind forecasting. Each turbine's measurements and control actions create a connected system that needs to be modeled as a whole unit because of its spatio-temporal characteristics. The study demonstrates how advanced forecasting models that model complex dependencies between variables can convert existing industrial

control systems into proactive management systems instead of developing new control methods.

1.2 Problem Statement

The development of proactive wind farm control using forecasting techniques demands addressing complex spatio-temporal forecasting problems which differ from regular wind forecasting needs for weather forecasting or energy trading applications. Our control-oriented forecasting requirements differ fundamentally from standard needs because they operate on shorter timescales and face unique limitations which current methods do not address.

The forecasting task for short-term wind control predictions requires historical high-frequency data of vertical and horizontal wind speed vectors and nacelle direction measurements from all turbines in a wind farm to predict future wind conditions at each location for time horizons between 1 to 5 minutes with uncertainty estimates. The required lead time for mechanical wind turbine actuator yaw rate operations determines the duration of this forecasting period for turbine direction orientation based on predicted wind directions. The simple forecasting objective requires multiple advanced complexity layers which make it an extremely difficult machine learning problem.

The problem demands predicting $2N$ interconnected variables for each future timestep in a wind farm with N turbines due to its multivariate nature, which increases the dimensionality of the prediction space. The wind conditions of one turbine depend on wake effects from other turbines, thus creating complex spatial relationships that change with time. The complex spatio-temporal connections between measurements require more than basic univariate methods since these methods would fail to detect essential relationships between wind speed components.

The spatio-temporal connection between turbines creates a distinctive feedback loop because upstream turbine yaw position adjustments modify wake trajectories which then affect the wind conditions of downstream turbines. The model requires prediction of both natural wind evolution and the modifications which specific turbines will make to future wind fields. The dynamic interaction between prediction and control creates a complex system that pushes beyond traditional passive forecasting methods.

Wind direction representation requires precise handling because it exists as a circular variable with a 0 to 360-degree discontinuity. The method required the breakdown of wind velocity into its Cartesian components (u, v) instead of speed and direction. The method ensures continuous prediction for all output variables.

The stochastic and non-stationary characteristics of wind create additional difficulties

when trying to forecast it. The quick changes in wind conditions stem from various factors which include atmospheric instabilities as well as thermal changes, terrain roughness, and other effects. The forecasting model requires the ability to distinguish between predictable patterns, like wind change delays and wake propagation across the farm, and actual random variations.

Real-time operational constraints for operational efficiency are established by the controller integration process. The forecasting model needs to generate exact and timely wind predictions for every turbine before the controller finishes processing and producing new control actions at a rate of 5 seconds, while operating under limited computational resources in wind farm control centers. The implementation of CFD simulations becomes impractical due to computational expense, thus requiring a trade-off between precision and operational efficiency.

The combination of high dimensionality with spatio-temporal coupling and stochastic dynamics along with computational constraints leads to an unresolvable problem for standard forecasting approaches. The ability of traditional statistical methods to model complex non-linear relationships is insufficient, and generic deep learning methods lack the specific requirements needed for wind farm data forecasting. The research into specialized deep learning architectures was motivated by the need to solve these simultaneous challenges.

1.3 Research Objectives and Contributions

The main goal of this thesis is to create and train an optimized deep-learning model that provides ultra-short-term probabilistic wind forecasts from real wind farm SCADA data to support preview-enabled proactive control systems. The TACTiS-2 (Transformer-Attentional Copulas for Time Series) architecture receives adaptation to fulfill this purpose.

Specific objectives include:

1. Developing a comprehensive data preprocessing pipeline to transform the raw SCADA measurements from the wind farm into a format usable for the machine learning (ML) model, addressing data quality issues.

2. Adapting the TACTiS-2 architecture to work with the overarching frameworks and integrating it in the rest of the package.

3. Implementing an effective hyperparameter tuning methodology in order to calibrate the TACTiS-2 model for optimal performance.

4. Validating the forecaster's ability to accurately and precisely predict future wind conditions as well as give good estimations of the probabilistic uncertainty.

1.4 Thesis Organization

The rest of this thesis is organised as follows:

- Chapter 2 provides the necessary background on wind farm control, reviewing related work in wind forecasting and deep learning applied to time series, and introduces the TACTiS-2 architecture.
- Chapter 3 details the methodology, including the system architecture, the data preprocessing pipeline, the TACTiS-2 adaptation strategy, and the training and optimization framework.
- Chapter 4 reports the final experimental results for forecasting accuracy and performance.
- Chapter 5 discusses the implications, limitations and broader impact of this work.
- Finally, Chapter 6 concludes with a summary of all contributions and explores possible directions for further research in the future.

2 Theoretical Foundations

2.1 Wind Farm Dynamics and Control

Renewable energy systems face one of the most complicated fluid mechanics processes in the wind dynamics of wind farms. The process of wind turbine energy extraction from wind also modifies downstream flow patterns, which results in velocity reduction areas with increased turbulence known as 'wakes'. The wake effects spread across the farm, leading to complex time-varying fields that affect power generation and structural stress on all turbines in the farm. Controlling these dynamics plays a crucial role in reaching maximum energy capture potential while satisfying operational safety requirements and extending system lifetime.

2.1.1 Wake Physics and Modeling

The fundamental principles behind wind turbine wakes originated from Betz's 1920 application of momentum theory to wind turbines. Turbine operation extracts wind energy through a pressure discontinuity which creates downstream velocity deficits. The wake area keeps its rotor diameter dimension during the initial phase but expands outward throughout its flow path due to turbulent mixing with free-stream flow. The expanding wake shows a decreasing velocity deficit until the momentum absorption from the surrounding flow becomes complete.

The Jensen model (Jensen, 1983) serves as a basic analytical framework for wake behavior prediction, which continues to function in wind farm control systems since its introduction in 1983. According to the model, the wake expands linearly through space and maintains a constant velocity profile within its wake area. The model earns its "top-hat" designation (Manwell et al., 2010). The wake velocity at position x downstream of a turbine can be described as:

$$u_x = u_\infty \left(1 - \frac{2a}{(1 + k_w x/r_0)^2}\right) \tag{2.1}$$

where u_∞ is the free-stream wind speed, a is the axial induction factor, k_w is the wake decay coefficient, and r_0 is the initial wake radius (normally the rotor radius) (Jensen,

1983). The wake radius expands linearly according to:

$$r_x = r_0 + k_w x \tag{2.2}$$

The wake decay coefficient k_w functions as an essential factor for making accurate predictions, and it varies based on atmospheric environmental factors. Peña et al. (2016) demonstrated that k_w can be related to atmospheric turbulent intensity through $k_w = u_*/u_h$, where u_* is the friction velocity and u_h is the hub-height wind speed. This relationship reveals that typical onshore values range from about 0.0075 to 0.04, which is significantly lower than the often assumed value of 0.075, suggesting that many existing analyses may underestimate wake losses in low-turbulence conditions (Bastankhah and Porté-Agel, 2014).

The Jensen model provides simple real-time control applications through its computational efficiency, but its uniform wake velocity assumption fails to model essential physical processes. Bastankhah and Porté-Agel (Bastankhah and Porté-Agel, 2014, 2016) developed Gaussian wake models to replace traditional velocity distribution models with more accurate Gaussian representations which match experimental findings:

$$\Delta u(x,r) = u_\infty \left(1 - \sqrt{1 - \frac{C_T}{8(\sigma_y/D)(\sigma_z/D)}}\right) \exp\left(-\frac{r^2}{2\sigma^2}\right) \tag{2.3}$$

where C_T is the thrust coefficient, σ_y and σ_z are the wake widths in lateral and vertical directions, D is the rotor diameter, and r is the radial distance from the wake centerline. This formulation captures the smooth velocity transition at wake boundaries and provides more accurate predictions of wake interactions (Niayifar and Porté-Agel, 2016).

Multiple wake interactions present a higher degree of complexity than single wake models. The merging of upstream turbine wakes results in unquantifiable cumulative velocity deficits that cannot be accurately predicted through basic linear combination methods. The control-oriented model developed by King et al. (2021) demonstrates the secondary wake steering effects that result in velocity changes and counter-rotating vortex pairs which impact downstream turbine performance. Research indicates that wake interactions decrease farm power output by 10–40% in specific operational scenarios (Porté-Agel et al., 2020). Additionally, extra losses occur due to turbulence-related fatigue loading.

Wind farm blockage generates an obstacle for flow physics at the scale of farms because it slows down the flow in front of the turbine farm by 2–5%. The effect, often ignored in traditional wake models, has significant implications for both power production estimates and control strategy design.

2.1.2 Greedy Control Strategy

The simplest wind farm control method is the greedy method, which enables individual turbines to find their maximum power output while disregarding the consequences for following turbines. Turbines maintain their yaw positions to face the wind direction they measure locally while using low-pass filtered signals to minimize their response to turbulent or noisy fluctuations.

Simple greedy control operates independently among field turbines, yet it creates substantial limitations for wind farms. The maximum extraction of power by upstream turbines generates wake effects that diminish downstream energy availability and affect turbine operation. The Fleming et al. (2019) study of a commercial wind farm showed that greedy control limits downstream turbine capacity factors to 50 or 60% of their full freestream potential during aligned wind directions.

2.1.3 LUT-Based Wake Steering

Lookup table (LUT) based wake steering is a more sophisticated control approach that takes into account wake interactions explicitly. The strategy involves intentionally misaligning upstream turbines from the incoming wind direction to deflect their wakes away from the downstream turbines. The lateral wake deflection δ can be approximated as (Jiménez et al., 2010):

$$\delta(x) = \delta_0 + \theta x \left(\frac{C_T \cos^2(\gamma)}{2\alpha_{KA}} \right) \tag{2.4}$$

where δ_0 is the initial wake deflection, θ is a parameter of the yaw-induced vorticity, γ is the yaw misalignment angle, and α_{KA} accounts for the wake expansion.

The power loss of the yawed turbine follows approximately (National Renewable Energy Laboratory (NREL), 2023):

$$P_{\text{yawed}} = P_{\text{aligned}} \cdot \cos^p(\gamma) \tag{2.5}$$

The exponent p generally falls within the range of 1.8 to 3.0 based on turbine design characteristics and environmental conditions. LUT control produces substantial net power increases of 3–15% for the entire farm when wind conditions are favorable.

However, LUT control faces significant practical challenges. The lookup tables derive from steady-state computations, yet the real wind conditions remain highly variable. The optimal yaw positions for power generation during a specific instant become suboptimal as the wind direction alters in the subsequent moments. The turbines experience unintended misalignment because of slow yaw maneuvers, which prevents accurate tracking of rapidly changing setpoints and thus reduces total farm power output.

2.1.4 Timing Constraints and the Need for Forecasting

Both greedy and LUT control strategies suffer from their inability to actively predict future wind conditions. These methods use current or past conditions to generate responses instead of predicting upcoming conditions. The fundamental challenge of wind farm control strategies emerges because of their reactive nature:

- **Wind direction changes:** typically, significant shifts occur on timescales of 30 seconds to 5 minutes.
- **Wake propagation:** wakes travel between turbines in 30–60 seconds (depending on spacing and wind speed).
- **Yaw actuation:** requires 60–180 seconds for realignment (at 0.3–0.5°/s).
- **Wake stabilization:** 2–3 minutes for new wake patterns to stabilize.

The combination of these timescales generates an intricate temporal optimization challenge. Turbine yaw movements that attempt to adjust to wind direction changes become ineffective because the wind conditions have already shifted during the maneuver time. The temporal difference between wind prediction and yaw adjustment becomes crucial for wake steering because it depends on sustained yaw offsets relative to actual wind direction.

Studies from field experiments demonstrate that using preview information about upcoming wind conditions between 1–5 minutes enhances control performance. Simley et al. (2021) reported that controllers employing preview data produced power increases between 2% and 5% when compared to standard LUT control systems while using reduced yaw actuator operation. The study conducted by Sengers et al. (2023) determined the best preview period to be between 60 and 90 seconds because it strikes a balance between accurate forecasting and sufficient control response time.

2.1.5 Implications for Forecasting Requirements

The wind farm dynamics together with control system constraints determine particular requirements for forecasting systems.

Temporal Resolution: Forecasts must be updated every 30–60 seconds to track the quickly changing wind conditions (Simley et al., 2021). These updates must be balanced with the required computational constraints, as the control decisions often require near-instant forecast generation.

Spatial Coverage: Individual turbine-level predictions are essential, as wake effects create high spatial heterogeneity in the farm (Henry et al., 2024). A simple farm-level

forecast is unable to capture the complex flow patterns caused by the individual turbines' wake interactions.

Prediction Horizon: The 1 to 5 minute forecast horizon is the result of the conservative match between control system dynamics and the forecast ability. Shorter horizons give insufficient lead time for control actions, and longer horizons are more computationally expensive and suffer from degrading accuracy (Sengers et al., 2023).

Variables of Interest: Wind speed components (u, v) and direction are the main control variables (Boersma et al., 2017).

Uncertainty Quantification: Probabilistic predictions enable risk-aware control actions, which are particularly important for wake steering where we have both aleatoric and epistemic uncertainty sources (model uncertainty and forecast uncertainty respectively) (Simley et al., 2020).

These requirements differentiate wind farm control forecasting from other applications. Unlike, for example, day-ahead market forecasting, where hourly averages would be sufficient, control-oriented forecasting requires very high temporal resolution and spatial detail. Control applications require accurate predictions of specific events, such as a possible wind direction shift that will arrive 90 seconds in the future.

The successful application of forecast-enabled control techniques heavily depends on forecasting systems that were specifically designed to fulfill these purposes and requirements. Traditional meteorological models do not possess the necessary resolution nor responsiveness for this task (Foley et al., 2012). Statistical methods that rely on historical averages fail to detect the fast non-stationary patterns. Specialized deep learning approaches such as the TACTiS-2 architecture of this thesis aim to bridge this knowledge gap. These approaches both identify complex wind farm-specific spatio-temporal patterns and produce probabilistic forecasts at timescales useful for control systems.

2.2 Deep Learning for Time Series Forecasting

The transformation from traditional statistical techniques to deep learning methods for time series forecasting represents a basic modification in temporal pattern modeling and prediction. The thesis tackles ultra-short-term wind forecasting through multivariate probabilistic predictions at high temporal resolutions and non-linear relationships, which requires this fundamental change. Understanding this evolution and the reasons for the shift to more modern deep learning architectures is important to appreciate why transformer-based models, particularly focused in TACTiS-2 for this work, might offer advantages for wind farm control applications.

2.2.1 Classical Methods vs Neural Networks

The traditional time series forecasting methods of Autoregressive Integrated Moving Average (ARIMA) models (Box et al., 2015) and exponential smoothing methods (Hyndman et al., 2008) provided traditionally robust frameworks for univariate time series analysis, which offer interpretable parameters and well-understood statistical properties. The methods excel at discovering linear dependencies and seasonal patterns when the data generation process remains stable and stationary.

These methods demonstrate clear shortcomings when faced with the advanced requirements of modern forecasting tasks. ARIMA models fail to operate effectively when handling non-linear patterns while needing precise model order selection (Yatiyana et al., 2017). Kalman filters excel at linear state-space models with Gaussian noise but fail to detect complex non-stationary dynamics that usually exist in wind farm environments (de Bézenac et al., 2020). The methods function as univariate systems or need strong correlation assumptions for multivariate setups, which prevents them from handling complex spatio-temporal dependencies found between multiple wind turbines.

Neural networks gained acceptance as a time series forecasting solution despite initial concerns about overfitting (Zhang et al., 1998) before proving themselves through various successful implementations including the M5 competition victory (Makridakis et al., 2022). Neural networks provide three essential benefits: they discover intricate non-linear relationships directly from data, handle complex high-dimensional multivariate inputs without labeling, and generate arbitrary probability distributions instead of Gaussian assumptions.

2.2.2 Recurrent Architectures

Recurrent Neural Networks (RNN) emerged as a natural sequence modelling architecture because they maintain hidden states to capture information from sequences of any length. Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) specifically solved the vanishing gradient issue of basic RNNs through gating mechanisms that enabled gradients to traverse long sequences without exponential loss. These architectures served as the starting point for early deep learning methods in time series prediction.

DeepAR (Salinas et al., 2020) demonstrated the capabilities of LSTM-based probabilistic forecasting through an autoregressive approach which generated parameters for probability distribution outputs at each time step. This model demonstrated superior performance to classical methods on diverse datasets, handled missing values automatically and provided uncertainty estimates. Similar to LSTMs, hybrid architectures which merged Convolutional Neural Networks (CNNs) with LSTMs showed promising results for detecting both local

patterns and long-term dependencies (Shi et al., 2015).

However, RNN architectures maintain fundamental restrictions which create major problems when used for wind farm forecasting needs. RNN training suffers from extended durations because of their sequential processing nature, which prevents efficient parallelization of computations while working with the large SCADA datasets that record turbine measurements at second or minute intervals. The hidden state bottleneck reduces the model's ability to focus on important past observations because it forces all historical data into a set-size vector. RNNs encounter significant challenges in modeling complex inter-variable dependencies because their hidden state must encode temporal patterns of all variables without any method to detect their interactions.

2.2.3 Transformer Neural Networks

The Transformer architecture (Vaswani et al., 2017) introduced a revolution in sequence modeling through self-attention mechanisms which directly capture any sequence position relationship. The main advancement occurs through attention mechanisms that produce relevance scores between all elements so the model can dynamically focus on input information regardless of spatial or temporal distances.

The self-attention mechanism operates through a query-key-value framework, where each element in a sequence is transformed into query (Q), key (K) and value (V) representations. The attention scores are then computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.6}$$

where d_k is the dimension of the key vectors, which serves as a scaling factor to prevent gradient vanishing in the softmax function. This formulation enables each position to attend to all positions in the input sequence with a single operation, which also dramatically improves computational efficiency compared to the sequential processing that is required by RNNs.

Multi-headed attention mechanisms extend this concept further by computing multiple attention functions in parallel, each potentially capturing different types of relationships:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.7}$$

where each $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ operates in a different learned subspace. This mechanism has been shown to be very effective for multivariate time series, as the different heads can specialize in capturing temporal patterns, cross-variable correlations, or specific periodic components.

Transformers offer multiple benefits for time series applications. The parallel processing of time steps through modern GPU hardware leads to substantially shorter training periods. The architecture provides natural multivariate input handling because its attention mechanisms discover intricate spatio-temporal relationships between turbines across large complex wind farm datasets.

2.2.4 Transformer Adaptations for Time Series

Basic transformer models demonstrate good sequence modeling capabilities, but researchers have developed adaptations which focus on time series forecasting requirements. The ProbSparse attention mechanism in Informer (Zhou et al., 2021) enables selective time step attention to decrease self-attention complexity while maintaining prediction quality. The massive wind farm datasets requiring thousands of time steps make this approach highly relevant for wind farm forecasting because it ensures computational efficiency.

Autoformer (Wu et al., 2021) introduces an auto-correlation mechanism to replace self-attention by progressively breaking down time series into trend and seasonal components. The transformer framework demonstrates its adaptability through these architectural innovations that suit particular time series data features. The decomposition process matches wind data periodic characteristics but makes assumptions about stationarity which might not apply during fast weather changes.

The Temporal Fusion Transformer (Lim et al., 2021) combines local temporal processing from LSTMs with self-attention to capture long-range dependencies. Figure 2.1 illustrates the general encoder-decoder transformer architecture for time series forecasting.

However, the majority of existing methods concentrate on point forecasting and basic parametric distributions without providing enough flexibility to model the multiple modes found in wind data.

2.2.5 Probabilistic Forecasting and Uncertainty Quantification

The inability of point forecasts to show uncertainty prevents risk assessment for control decisions even when their accuracy is high. The limitation of point forecasts is addressed by probabilistic forecasting which predicts the complete conditional distribution of future values from historical observations.

Multiple deep learning forecasting approaches exist for creating probabilistic predictions. Quantile regression approaches determine particular predictive distribution percentiles, yet they fail to produce a complete probability distribution across all quantiles (Bazionis and Georgilakis, 2021). Parametric methods that output distribution parameters face two main limitations since they require specific distributional assumptions and often fail to

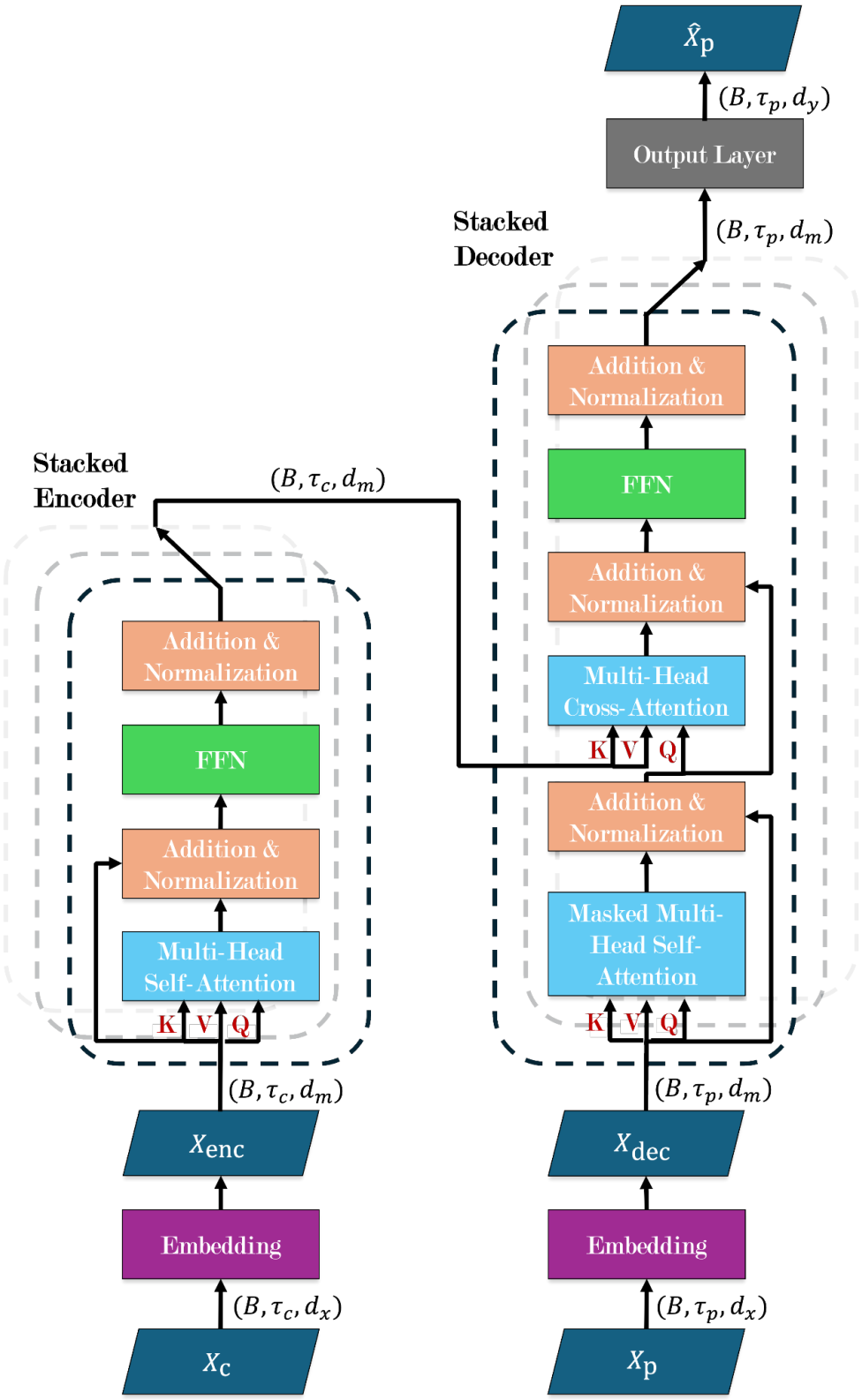

Figure 2.1: Transformer architecture for time series forecasting showing the dual-encoder design. The stacked encoder processes historical time series data (X_{enc}) through multi-head self-attention and feed-forward networks (FFN), while the stacked decoder generates future predictions (X_p) using both self-attention and cross-attention mechanisms. The embedding layers transform input features into high-dimensional representations, and addition & normalization layers ensure stable training dynamics. Adapted with permission from Henry, A. (2025).

represent the diverse multimodal patterns found in wind data.

Rasul et al. (2021b) describe normalizing flows as a method that uses invertible transformations in a sequence to convert a simple base distribution into a complex target distribution. TimeGrad (Rasul et al., 2021a) uses RNN encodings together with denoising diffusion models to generate predictions from the predictive distribution. The methods enable distribution modeling without parametric constraints, yet their inference process demands multiple sampling steps, thus making them unsuitable for real-time control systems with limited latency tolerance.

Copula-based methods, on the other hand, provide an elegant solution by separating the modeling of marginal distributions from their dependency structure (Salinas et al., 2019). This decomposition is very well-suited for wind forecasting, where the individual turbine measurements may have different marginal behaviors due to individual characteristics (e.g. local terrain effects), while their dependencies are influenced by the atmospheric dynamics and wake interactions with the other turbines in the field.

The TACTiS architecture (Drouin et al., 2022) uses this approach with transformer attention mechanisms to learn non-parametric copulas which capture arbitrary dependency structures at a reasonable computational cost.

2.2.6 Implications for Wind Farm Forecasting

Deep learning models became relevant for wind farm data analysis because they can learn complex multivariate relationships between multiple turbine measurements without imposing restrictive assumptions. The models need to adapt to non-stationary dynamics, which arise from changing atmospheric conditions, wake effects and turbine operations, without human involvement.

The necessity of uncertainty quantification in control systems requires probabilistic forecasting methods. Controllers need to evaluate performance benefits against potential negative outcomes so they select cautious control actions, which demand predictive distributions with no parametric assumptions along with confidence intervals. Real-time control systems operate within 5–10 second update cycles that require models to produce accurate predictions efficiently.

The transformer architecture together with the TACTiS-2 model fulfills these requirements because it uses attention-based dependency modeling and flexible copula-based probabilistic approaches with efficient computation, which makes it suitable for real-time control systems. The developed platform in this thesis utilizes these capabilities to support both training and deployment of advanced transformer models for wind farm forecasting operations. This research aims to show how modern deep learning and AI approaches can improve renewable energy systems' efficiency along with reliability.

2.3 TACTiS-2: Transformer-Attentional Copulas

2.3.1 The Copula Framework for Wind Farm Dependencies

Copula theory provides an elegant mathematical framework that can naturally decompose the multivariate problem of capturing the complex dependencies between multiple turbines in a wind farm into manageable components, separating the modeling of the individual turbine behaviors from their complex interdependencies. This decomposition is valuable for wind farms where the wake effects create spatial and temporal correlations and each turbine shows its own local wind characteristics influenced by many uncertain elements.

Mathematical Foundations of Copulas

A copula is a multivariate cumulative distribution function (CDF) defined on the unit hypercube $[0, 1]^d$ with uniform marginal distributions (Nelsen, 2006). Formally, for a d-dimensional random vector $U = [U_1, \ldots, U_d]$ where each $U_i \sim \mathcal{U}[0, 1]$, the copula C is defined as:

$$C(u_1, \ldots, u_d) = P(U_1 \leq u_1, \ldots, U_d \leq u_d) \tag{2.8}$$

The importance of copulas in multivariate modeling comes from Sklar's theorem (Sklar, 1959), which states that any joint CDF of a random vector $X = [X_1, \ldots, X_d]$ can be uniquely decomposed as:

$$F(x_1, \ldots, x_d) = C\Big(F_1(x_1), \ldots, F_d(x_d)\Big) \tag{2.9}$$

where $F_i(x_i) = P(X_i \leq x_i)$ represents the marginal CDF of variable X_i. When the marginal distributions are continuous, this decomposition is unique, and the corresponding probability density function can be expressed as:

$$f(x_1, \ldots, x_d) = c\Big(F_1(x_1), \ldots, F_d(x_d)\Big) \cdot \prod_{i=1}^{d} f_i(x_i) \tag{2.10}$$

where c denotes the copula density and f_i represents the marginal density of X_i. This factorization manages to separate the dependency structure (from the copula) from the marginals (captured by the individual distributions), which allows for modular modelling approaches that can adapt to the complex nature of wind farm data.

Natural Alignment with Wind Farm Structures

The copula framework shows promise to be quite well-suited for wind farm applications due to the characteristics of the problem. In a wind farm with N turbines, each contributing

u and v wind components, we have a $2N$-dimensional prediction problem where we have both marginal behaviors and complex non-linear dependencies. The copula decomposition allows us to model these aspects separately.

Each turbine in a wind farm experiences unique local conditions that are determined by its location, surrounding terrain and wake exposure compared to other turbines. As previously mentioned, upstream turbines operate in free-stream conditions with distributions more influenced by atmospheric turbulence, while downstream turbines experience the wake-influenced flows with generally increased variance and reduced mean velocities (Bastankhah and Porté-Agel, 2014). The marginal distribution F_i can capture these turbine-specific characteristics individually, without making generalistic assumptions. For example, a turbine that is frequently exposed to wake fields might show a bimodal wind speed distribution, while a front-row turbine might show a more conventional Weibull distribution.

The dependency structure between turbines, which is encoded in the copula C, naturally represents the wake propagation dynamics. When the wind flows from turbine i to turbine j with a propagation delay τ_{ij}, the copula can in theory capture the lagged correlation structure without requiring explicit temporal modeling in the marginals. This separation is valuable because wake-induced correlations depend on the relative positions and wind directions, while the marginal distributions depend on the turbine's local conditions. As Henry et al. (2024) demonstrated, these spatial correlations can exceed 0.8 for aligned turbine pairs but be under 0.3 for perpendicular configurations, which indicates the need for flexible dependency models.

Limitations of Parametric Copulas

While parametric copulas such as Gaussian and Student-T copulas have been successfully applied to wind power forecasting (Gilbert et al., 2020; Bessa, 2016), they impose restrictive assumptions that limit their effectiveness for capturing the full complexity of wind farm dependencies.

The Gaussian copula, defined through the inverse of the standard normal CDF Φ, assumes that dependencies can be fully characterized by a correlation matrix Σ:

$$C_{\text{Gauss}}(u_1, \ldots, u_d; \Sigma) = \Phi_\Sigma\Big(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d)\Big) \tag{2.11}$$

This formulation cannot capture tail dependencies, which is the tendency for extreme events to occur simultaneously. In wind farm contexts, this is important for handling rapid weather changes where correlations intensify at extreme wind speeds. Theuer et al. (2022) made the observation that these correlation structures during high-wind events are quite distinct from normal conditions, with wake effects often becoming less pronounced

as turbulence intensity increased.

The Student-T copula introduces tail dependence; however, it enforces symmetry, assuming equal correlation intensification in both upper and lower tails. Wind farm data often shows asymmetric patterns characterized by strong correlations during low-wind wake conditions (lower tail) but weaker correlations during high-wind conditions when the wakes dissipate quickly (upper tail). This asymmetry inherently cannot be captured by the Student-T copulas, potentially leading to suboptimal uncertainty quantification.

Both Gaussian and Student-T copulas make these assumptions about static dependencies that cannot adapt to changing conditions. Wake propagation, however, can vary significantly with different weather patterns. A correlation matrix estimated during stable conditions might misrepresent the dependencies during convective conditions. This is why the model for this thesis was chosen for its ability to dynamically adjust its dependency structures based on varying conditions – something that parametric copulas cannot perform.

Non-Parametric Approaches

These limitations of parametric copulas for wind farm applications motivate the search for a different approach that can adapt to the complex non-stationary nature of wake-induced dependencies without putting restrictive distribution assumptions in place. The search for suitable architectures for the forecasting platform in this thesis required looking for models that could capture high-dimensional multivariate dependencies, provide full probabilistic distributions for uncertainty-aware control and maintain good computational efficiency suitable for real-time control, while handling the irregular data availability typical of SCADA systems.

The transformers' attention mechanism ability to learn dynamic relationships between variables, as explained in Section 2.2.3, gives a solid foundation for non-parametric copula modelling. The attention mechanism can allow the model to discover which historical observations and turbine measurements are the most relevant to predict the current dependencies, automatically adapting to changing atmospheric conditions.

Among the options, the TACTiS model showed itself to be particularly well suited due to the unique combination of copula-based probabilistic modelling and transformer-based flexibility. Given the limited time available for the master thesis and the complexity of the requirements, leveraging this state-of-the-art model that had already demonstrated strong performance in some standard multivariate time series benchmarks seemed like the sensible option, and allowed shifting the focus onto the challenges of creating the wind forecasting open-source platform for integrating different forecasting models – tuning, training and validating them – instead of developing a whole new model from scratch.

2.3.2 TACTiS: Attentional Copulas

The development of TACTiS (Transformer-Attentional Copulas for Time Series) by Drouin et al. (2022) introduced a novel approach that uses attention mechanisms to learn flexible dependency structures directly from the input data. This shows potential in overcoming the limitations of parametric copulas, which would be limiting for the characteristics of wind farm operation.

The main advantage of TACTiS is its addition of 'attentional copulas', which are a new class of non-parametric copulas that leverage neural networks trained to mimic valid copula functions without making parametric assumptions. Unlike the Gaussian or Student-T copulas, attentional copulas can learn any arbitrary dependency patterns directly from the observed data, adapting their structure to match the empirical relationships between variables.

The attentional copula factorizes the joint copula density autoregressively according to the permutation $\pi = [\pi_1, \ldots, \pi_d]$ of the variable indices:

$$c_{\phi_c^\pi}(u_1, \ldots, u_d) = c_{\phi_{c1}^\pi}(u_{\pi_1}) \times c_{\phi_{c2}^\pi}(u_{\pi_2} \mid u_{\pi_1}) \times \cdots \times c_{\phi_{cd}^\pi}(u_{\pi_d} \mid u_{\pi_1}, \ldots, u_{\pi_{d-1}}) \tag{2.12}$$

where each conditional density is parameterized using an attention mechanism. The first term is fixed as a uniform distribution $\mathcal{U}[0, 1]$, ensuring proper copula marginals.

The attention-based conditioner produces parameters for each conditional distribution by performing multi-head attention over a memory composed of the representation of all observed and preceding missing tokens and their CDF-transformed values. Then, for each conditional distribution, the attention mechanism computes:

$$q_k = \text{Query}_{\theta_C}\left(z_{\pi_k}^{(m)}\right), \qquad K = \text{Key}_{\theta_C}(z, u), \qquad V = \text{Value}_{\theta_C}(z, u) \tag{2.13}$$

The attention weights effectively learn this way as a data-driven similarity metric between conditions, automatically identifying which historical observations are most relevant for predicting current dependencies.

Ensuring that the learned function satisfies the mathematical definition of a valid copula, according to Theorem 1 of Drouin et al. (2022), requires the model to be 'permutation invariant', which means producing the same joint density regardless of the ordering of variables in the autoregressive factorization.

To achieve this permutation invariance, TACTiS optimizes the following objective across all possible permutations:

$$\theta^* = \arg\min_\theta \; \mathbb{E}_{\pi \sim \Pi} \, \mathbb{E}_{x \sim \mathcal{D}} \left[-\log g_{\phi^\pi}\left(x_1^{(m)}, \ldots, x_d^{(m)}\right) \right] \tag{2.14}$$

where Π represents the set of all $d!$ permutations of the d variables, and g_{ϕ^π} is the density estimator using permutation π.

This means that by averaging over all permutations, the model is forced to learn a dependency structure that remains consistent regardless of the factorization order, which ensures that the marginal distribution of each variable, when integrated over all others, gives the required uniform distribution on $[0, 1]$.

However, this comes at a severe computational cost. For a wind farm with 7 turbines, each giving u and v wind speed components ($d = 14$), the number of possible permutations is $14! = 87{,}178{,}291{,}200$. Even evaluating a single gradient step across all permutations would require over 87 billion forward passes through the model, which makes it infeasible for practical training.

To address this issue, Drouin et al. (2022) propose a Monte Carlo approximation where only a small random subset of permutations is sampled during each training batch:

$$\hat{\theta} = \arg\min_{\theta} \; \mathbb{E}_{x \sim \mathcal{D}} \left[-\frac{1}{|\Pi_{\text{batch}}|} \sum_{\pi \in \Pi_{\text{batch}}} \log g_{\phi^\pi}\left(x_1^{(m)}, \ldots, x_d^{(m)}\right) \right] \tag{2.15}$$

where $|\Pi_{\text{batch}}| \ll d!$ represents a small number of randomly sampled permutations per batch. However, this reduction of the permutation space would require a very large number of training iterations to achieve convergence, which still becomes extremely computationally inefficient for large databases like wind farm SCADA data. Additionally, the sparse sampling of the permutations potentially affects the learning of multivariate dependencies in a negative way. When modeling wake propagation from turbine a to turbine b, the model would need to observe permutations where a precedes b in the factorization. With random sampling, this rarely occurs, which might lead to poor capture of physical dependencies.

These fundamental limitations of the original TACTiS motivated the usage of TACTiS-2, which maintains the flexibility of attentional copulas while eliminating the permutation-based training, making this model a much better alternative for wind farm forecasting.

2.3.3 TACTiS-2: Architectural Innovations

The previously mentioned limitations of the original TACTiS architecture motivated the development of TACTiS-2, which introduced the ability to learn the attentional copulas through a much more efficient architecture that maintains theoretical validity while achieving computational tractability (Ashok et al., 2024). The transformation from $O(d!)$ to $O(d)$ complexity unlocked by this model allows it to be technically feasible for its implementation in wind farm control applications such as the one in this thesis.

Linear Complexity Solution

The key concept behind TACTiS-2's efficiency is its departure from the permutation-invariant density estimator that defined its predecessor. Where TACTiS required averaging over all possible factorizations of the joint distribution (around 87 billion permutations for a 14-variable wind farm system), TACTiS-2 follows a fixed autoregressive factorization strategy that manages to keep the validity of the learned copula while dramatically reducing the computational requirements.

This is achieved through a two-stage decomposition of the learning problem. Instead of trying to simultaneously optimize over all the distribution parameters, TACTiS-2 separates the learning of the marginal distributions from that of their dependency structure. Mathematically, this translates to this sequential two-phased problem:

Phase 1:

$$\theta_M = \arg\min_{\theta_M} \ \mathbb{E}_{x\sim\mathcal{D}}\Big[-\log \prod_i f_{\varphi_i}\Big(x_i^{(m)}\Big)\Big] \tag{2.16}$$

Phase 2:

$$\theta_C = \arg\min_{\theta_C} \ \mathbb{E}_{x\sim\mathcal{D}}\Big[-\log c_{\varphi_c}\Big(F_{\varphi_1}(x_1^{(m)}), \ldots, F_{\varphi_d}(x_d^{(m)})\Big)\Big] \tag{2.17}$$

This decomposition is theoretically sound with Sklar's theorem (Sklar, 1959), which guarantees that any multivariate distribution can be uniquely represented as the product of its marginal distributions and a copula function capturing its dependencies. Also, using a fixed arbitrary ordering of variables during training eliminates the need to consider all permutations while still learning a valid copula through its autoregressive factorization:

$$c(u_1, \ldots, u_d) = \prod_{i=1}^{d} c_{\varphi_{c,i}}(u_i \mid u_1, \ldots, u_{i-1}) \tag{2.18}$$

where each conditional $c_{\varphi_{c,i}}$ is parameterized as a histogram distribution with typically 20 bins, providing a flexible non-parametric representation.

Two-Stage Training Curriculum

In the first stage, only the marginal encoder parameters θ_M are optimized, while the copula components are bypassed entirely. The model learns to accurately capture the univariate distribution of each of the variables through modified Deep Sigmoidal Flows (DSF) (Huang et al., 2018), constrained to output values in the $[0, 1]$ range, to ensure valid probability integral transforms:

$$u_i = \mathrm{DSF}_{\varphi_i}(x_i), \qquad \varphi_i = \mathrm{HN}_{\theta_M}\Big(z_i^M\Big) \tag{2.19}$$

where HN_{θ_M} is a hypernetwork that produces DSF parameters based on the marginal encoder's representations z_i^M. For wind farm data, this stage captures location-specific wind characteristics such as Weibull-like distributions of the typical wind speeds at each turbine as well as any systematic biases, temporal and seasonal patterns in wind variability that are specific to each turbine location.

The constraint set on the DSF outputs to be in the $[0, 1]$ range is fundamental for maintaining the necessary mathematical validity that ensures that the transformed variables u_i follow a uniform distribution, which is a requirement of copula theory (Nelsen, 2006).

Once the marginal distributions converge, typically after 20–30 epochs based on validation loss plateau patterns, the marginal parameters θ_M are frozen and training focuses entirely on the copula encoder parameters θ_C. This stage optimizes:

$$\mathcal{L}_{\text{copula}} = -\mathbb{E}\Big[\log c_{\varphi_c}(u_1, \ldots, u_d)\Big], \qquad u_i = F_{\varphi_i}(x_i) \tag{2.20}$$

where $u_i = F_{\varphi_i}(x_i)$ are the probability integral transforms using the frozen marginal CDFs. By holding marginals fixed, the model can concentrate on learning complex dependency structures without unintendedly altering the marginal distributions. This has the potential to capture wake propagation patterns that involve lagged correlations between the upstream and downstream turbines, atmospheric coupling effects that induce variations across the farm, and other non-linear dependencies.

Dual Encoder Architecture

The main architectural innovation that enabled the two-stage approach is the introduction of a dual encoder design, which replaces the original TACTiS single encoder with two specialized transformer encoders.

First, the Marginal CDF Encoder produces representations z^M for representing marginal cumulative distribution functions. It processes both observed and missing tokens using masked attention:

$$z_i^M = \text{Enc}_{\theta_M}(x_i \cdot m_i,\, c_i,\, m_i,\, p_i) \tag{2.21}$$

where $x_i \cdot m_i$ represents the masked input (zero for missing values), c_i contains covariates, m_i is the binary mask indicator and p_i provides sinusoidal positional encoding.

Secondly, the Attentional Copula Encoder generates representations z^C dedicated to modeling the dependency structures. While it receives the same inputs as the marginal encoder, its parameters are optimized specifically to capture inter-variable relationships rather than individual distributions:

$$z_i^C = \text{Enc}_{\theta_C}(x_i \cdot m_i,\, c_i,\, m_i,\, p_i) \tag{2.22}$$

Another important feature of both encoders is their ability to handle missing data natively through the masking mechanism, which makes TACTiS-2 very suitable for wind farm SCADA systems, and gives another layer of safety after the preprocessing steps to make sure the input data can be used for forecasting purposes.

Objective Function

The complete TACTiS-2 objective function can be expressed as a two-phased optimization problem. It is stated as:

Phase 1 Objective (Marginal Learning):

$$\mathcal{L}_1(\theta_M) = -\mathbb{E}_{x\sim\mathcal{D}}\Big[\sum_{i=1}^{d} \log f_{\varphi_i}\Big(x_i^{(m)}\Big)\Big] \tag{2.23}$$

where f_{φ_i} are the marginal densities parameterized by Deep Sigmoidal Flows with parameters $\varphi_i = \mathrm{HN}_{\theta_M}(z_i^M)$.

Phase 2 Objective (Copula Learning):

$$\mathcal{L}_2(\theta_C) = -\mathbb{E}_{x\sim\mathcal{D}}\Big[\log c_{\varphi_c}\Big(F_{\varphi_1^*}(x_1^{(m)}), \ldots, F_{\varphi_d^*}(x_d^{(m)})\Big)\Big] \tag{2.24}$$

where $F_{\varphi_i^*}$ are the fixed marginal CDFs from Phase 1, and c_{φ_c} is the attentional copula density parameterized using causal attention mechanisms:

$$\begin{aligned} K_i &= \mathrm{Key}_{\theta_C}\Big(z^{C(o)},\, z_{1:i-1}^{C(m)},\, u^{(o)},\, u_{1:i-1}^{(m)}\Big), \\ V_i &= \mathrm{Value}_{\theta_C}\Big(z^{C(o)},\, z_{1:i-1}^{C(m)},\, u^{(o)},\, u_{1:i-1}^{(m)}\Big), \\ q_i &= \mathrm{Query}_{\theta_C}\Big(z_i^{C(m)}\Big), \\ \varphi_{c,i} &= \mathrm{Attn}_{\theta_C}(q_i,\, K_i,\, V_i) \end{aligned} \tag{2.25}$$

The attentional mechanism allows each missing variable to attend to all observed variables and previously predicted missing variables in the autoregressive ordering, creating a flexible non-parametric copula that can capture arbitrary dependency structures. Each conditional distribution $c_{\varphi_{c,i}}(u_i \mid u_{1:i-1})$ is represented as a histogram with B bins (typically $B = 20$), where the attention mechanism outputs the parameters $\varphi_{c,i}$ that define the bin probabilities.

Sample-Based Probabilistic Forecasting

Rather than producing only point predictions or parametric distributions, TACTiS-2 has the ability to generate forecasts by sampling from the learned joint distribution.

This is done by first performing the marginal transformations of the observed values to uniform space:

$$u_i^{(o)} = F_{\varphi_i}\left(x_i^{(o)}\right) \qquad \text{for all observed variables} \tag{2.26}$$

Then, autoregressively sampling the missing variables we want to predict (future wind speeds) from the histogram distribution, and transforming them back to the original space by applying the inverse marginal CDFs:

$$x_i^{(m)} = F_{\varphi_i}^{-1}\left(u_i^{(m)}\right) \qquad \text{for all sampled variables} \tag{2.27}$$

This process can be repeated N times to generate an entire empirical distribution of forecasts. Each repetition produces a different but plausible future scenario of the wind speed. For wind farm control, this means the controller would receive not just a single forecast but a distribution of possible future wind direction trajectories, which would enable more robust decision-making that accounts for forecast uncertainty.

These features make TACTiS-2 theoretically sound and well suited for operational wind farm forecasting where both accuracy and computational efficiency are critical. The linear scaling with the number of variables instead of factorial allows the model to handle wind farms with a large number of turbines and features, and the two-stage curriculum ensures stable training conditions – reasons why this model was selected as promising for the purposes of this thesis.

3 Methodology

3.1 System Architecture and Design

This section presents the comprehensive system architecture developed for ultra-short-term probabilistic wind forecasting in operational wind farms. The platform serves as a complete end-to-end solution which converts high-frequency Supervisory Control and Data Acquisition (SCADA) measurements into usable probabilistic predictions that can be used for wind farm control purposes. The architecture includes essential design elements of modularity along with scalability features and operational efficiency capabilities which support the development and validation of forecast-enabled control strategies.

3.1.1 Platform Overview

The forecasting platform includes a complete data processing system that transforms SCADA measurements into probabilistic forecasts for wind farm control needs. The forecasting system differs from conventional meteorological prediction systems because it emphasizes fast computation when delivering short-term predictions for 1–5 minute intervals (Sengers et al., 2023). The system framework tackles the main issue of combining state-of-the-art deep learning approaches into an operational system which needs both high dependability and quick computations.

The system architecture consists of four main operational stages that work together as a single system. The *Preprocessing Pipeline* cleans and structures raw SCADA data for deep learning analysis while solving wind farm data issues such as sensor faults, curtailment events and wake-induced relationships. The *Model Optimization Engine* applies distributed computing resources to perform automatic hyperparameter tuning by using Bayesian optimization for efficient navigation of the high-dimensional parameter space. The *Training Infrastructure* uses PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019) to perform model training efficiently through automatic mixed precision and distributed data parallelism. The *Inference System* produces probabilistic forecasts while maintaining the required computational speed for real-time applications.

The system operates through three operational modes, which include tune, train and inference modes that fulfill different computational and temporal needs. The tuning mode performs distributed hyperparameter optimization through the Optuna framework (Akiba et al., 2019) that employs Tree-structured Parzen Estimators (Bergstra et al., 2011)

and Hyperband pruning (Li et al., 2017) for efficient exploration of the parameter space. The training mode executes the two-stage curriculum learning method which TACTiS-2 requires by first learning individual distributions before using attentional copulas to model dependencies (Ashok et al., 2024). The inference mode produces probabilistic forecasts while using computational methods that are appropriate for real-time applications.

3.1.2 Modular Repository Architecture

The platform adopts a modular design philosophy that separates concerns across multiple interconnected repositories, facilitating independent development, testing, and deployment of components. This ensures that modifications to one component do not cascade through the entire system, enabling parallel development streams and reducing integration complexity.

The primary `wind-forecasting` repository serves as the orchestration layer, implementing the core pipeline logic and coordinating interactions between specialized components. Built on PyTorch Lightning, it provides automated experiment tracking through integration with Weights & Biases (Biewald, 2020), enabling comprehensive logging of metrics, hyperparameters, and model artifacts. The repository implements distributed training coordination across multiple GPUs using PyTorch's DistributedDataParallel (DDP) strategy, which has been shown to achieve near-linear scaling efficiency up to 32 GPUs for transformer-based models (Rasley et al., 2020). Checkpoint management ensures training resumption after interruptions, while standardized YAML configuration files enable reproducible experiments across different computing environments.

Model implementations reside in the `pytorch-transformer-ts` repository, which houses the adapted TACTiS-2 architecture alongside comparative baseline models. The TACTiS-2 implementation follows the architectural specifications detailed in Ashok et al. (2024), incorporating the dual-encoder design that separates marginal distribution learning from dependency modeling. The repository also includes implementations of Informer (Zhou et al., 2021), which employs ProbSparse attention to reduce computational complexity from $O(L^2)$ to $O(L \log L)$; Autoformer (Wu et al., 2021), which introduces auto-correlation mechanisms for capturing periodic patterns; and Spacetimeformer (Grigsby et al., 2021), which explicitly models spatial relationships through graph attention networks.

The framework leverages a fork of Amazon's GluonTS library (Alexandrov et al., 2020) to provide essential time series functionality including data transformations, sliding window generation, and evaluation metrics. The library enables efficient handling of multivariate time series data and implements probabilistic evaluation metrics such as the Continuous Ranked Probability Score (CRPS) for assessing forecast quality (Gneiting and Raftery, 2007), alongside traditional deterministic metrics like MAE and RMSE, as

well as implementing the custom samplers such as `ExpectedNumInstanceSampler` and `SequentialSampler` used for distinct purposes during the training procedure.

Control integration is facilitated through the `Wind-Hybrid-Open-Controller` (WHOC) framework, though the controller implementation remains under active development and is not the primary focus of this thesis. The framework provides standardized interfaces between forecasting outputs and control algorithms, supporting both greedy control strategies that maximize individual turbine power and wake steering approaches based on lookup tables (LUT). To determine the optimal prediction horizons for these control strategies, a comprehensive grid search was conducted across different forecast horizons, as detailed in Figure 3.1.

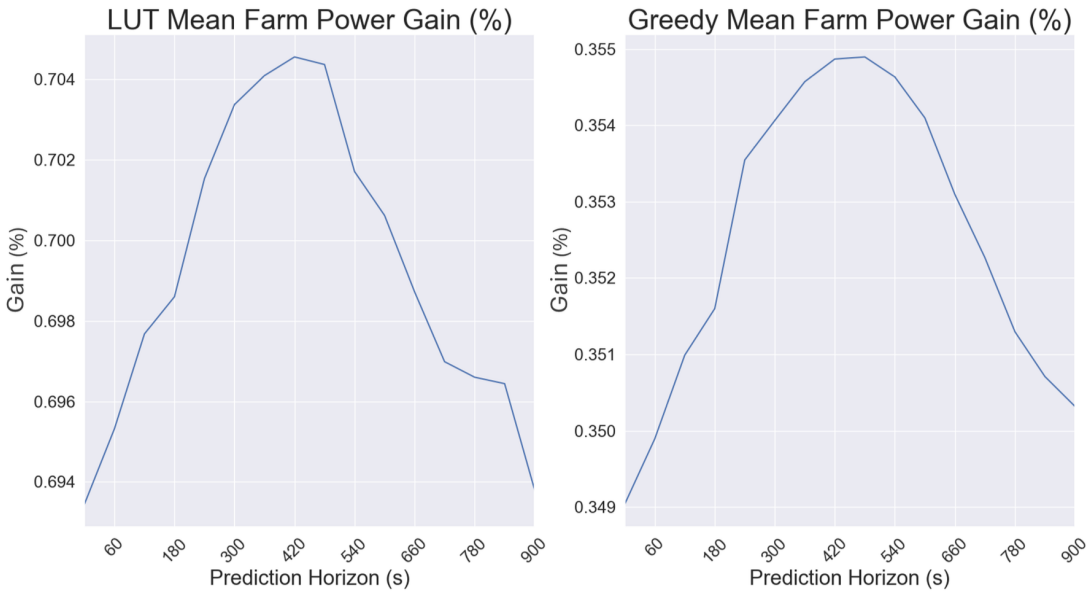

Figure 3.1: Wind farm power gains grid search for dynamic prediction horizons under LUT and greedy controllers.

The grid search evaluated prediction horizons ranging from 0 seconds to 900 seconds at 60-second intervals, identifying the configurations that maximized forecast accuracy while maintaining computational feasibility. The results indicated optimal horizons of approximately 480 seconds for greedy control and 420 seconds for LUT-based control.

3.1.3 Data Flow and Integration

The integration framework manages the data flow between system components, ensuring efficient processing from raw SCADA data through to probabilistic forecasts. Data flows through the system following a carefully orchestrated pipeline that balances computational efficiency with prediction accuracy.

The system utilizes the SMARTEOLE experiment dataset (Porté-Agel et al., 2020)

from the Sole du Moulin Vieux wind farm in northern France. This dataset comprises seven Senvion MM82 turbines, each rated at 2.05 MW with a hub height of 80 meters and rotor diameter of 82 meters, arranged in a north-south configuration with approximately 5 rotor diameter spacing. The relatively flat terrain and regular spacing create ideal conditions for studying wake interactions, with downstream turbines experiencing power deficits of 20–40% under aligned wind conditions (Bastankhah and Porté-Agel, 2014).

Raw SCADA measurements arrive at 1-minute frequency, providing horizontal (u) and vertical (v) wind velocity components derived from nacelle-mounted anemometers, nacelle position indicating turbine yaw angle, active power generation in kilowatts, and operational status flags indicating turbine availability and control mode. The preprocessing pipeline, detailed in Section 3.2, transforms these raw measurements through quality control procedures that identify and filter frozen sensors, anomalous power readings, and curtailment periods. Feature engineering creates derived variables including wind speed magnitude $WS = \sqrt{u^2 + v^2}$, normalized spatial coordinates encoding turbine positions, and cyclically encoded temporal features using sine and cosine transformations to capture diurnal and seasonal patterns.

The processed data feeds into the TACTiS-2 model, which employs a dual-encoder transformer architecture specifically adapted for multivariate time series forecasting. The model incorporates spatial positional encodings based on normalized turbine coordinates, enabling the attention mechanism to learn physical relationships governing wake propagation. The model implicitly learns these physical relationships through its attention patterns, with analysis revealing that attention weights correlate strongly with geometric turbine alignments under prevailing wind conditions.

The model generates probabilistic forecasts by sampling from the learned joint distribution, producing N_{samples} trajectory realizations that maintain realistic spatio-temporal correlations. For a forecast horizon of T timesteps and D variables (14 for our 7-turbine system with u, v components), the model outputs a tensor of shape $[N_{\text{samples}}, T, D]$ representing the empirical predictive distribution. This sample-based approach preserves complex multimodal distributions that arise from atmospheric instabilities and wake interactions, which parametric approaches would oversimplify.

Integration with wind farm control systems represents a potential application of the forecasting platform, though controller validation is beyond the scope of this thesis. The probabilistic forecasts generated by the model can be translated into formats suitable for different control strategies. For greedy control, point estimates of future wind conditions could enable anticipatory yaw adjustments. For LUT-based wake steering, uncertainty estimates could inform risk-aware control decisions. The FLOw Redirection and Induction in Steady State (FLORIS) simulator (National Renewable Energy Laboratory (NREL), 2023) provides a framework for physics-based validation of control performance, imple-

menting engineering wake models such as the Gaussian wake model of Bastankhah and Porté-Agel (2016).

3.1.4 Operational Modes

The platform implements three distinct operational modes, each with specific computational requirements and optimization objectives. The separation of these modes enables efficient resource utilization while maintaining clear boundaries between development and production operations.

The hyperparameter tuning mode employs distributed Bayesian optimization to identify optimal model configurations. The search space encompasses architectural parameters including the number of encoder layers (2–8), attention heads (4–16), and hidden dimensions (128–512); training hyperparameters such as learning rates (10^{-5} to 10^{-3}), batch sizes (16–128), and warmup steps (0–2000); and model-specific parameters including the number of copula bins (20–100) and Deep Sigmoidal Flow layers (2–6). A total list of hyperparameters for TACTiS-2 can be seen in Appendix A. The Tree-structured Parzen Estimator (TPE) algorithm models the distribution of good and bad hyperparameter configurations separately, sampling new configurations by maximizing the ratio $l(x)/g(x)$, where $l(x)$ represents the density of good configurations and $g(x)$ the density of poor configurations (Bergstra et al., 2011). Hyperband pruning accelerates the search by allocating resources adaptively, terminating poorly performing trials early based on intermediate validation metrics.

The training mode implements the two-stage curriculum learning approach required by TACTiS-2. Stage 1 focuses on marginal distribution learning, optimizing $\theta_M = \arg\min \mathbb{E}\Big[-\log \prod_i f_i(x_i)\Big]$, where f_i represents the marginal density of variable i. This stage typically starts to converge within 20–30 epochs, with validation loss plateauing as the model captures location-specific wind characteristics. Stage 2 freezes marginal parameters and optimizes the copula encoder, learning the dependency structure through $\theta_C = \arg\min \mathbb{E}\Big[-\log c\Big(F_1(x_1), \ldots, F_d(x_d)\Big)\Big]$, where c represents the copula density and F_i the learned marginal CDFs. The attention-based copula learns non-parametric dependencies that capture wake propagation patterns and atmospheric coupling effects.

The inference mode prioritizes computational efficiency while maintaining prediction quality. Context windows of length $5 \times$ prediction_horizon are maintained in memory, with new observations appended and old observations discarded in a rolling fashion. Batch processing enables simultaneous prediction for all turbines, leveraging GPU parallelism to achieve sub-second inference times.

3.1.5 Computational Infrastructure

The platform is designed to leverage modern high-performance computing infrastructure for development and training while maintaining the potential for deployment on resource-constrained edge devices. This dual requirement influenced several architectural decisions, balancing computational power during model development with efficiency considerations for future operational deployment.

Training operations utilize distributed computing across multiple NVIDIA H100 GPUs of the STORM cluster of the University of Oldenburg, coordinated through PyTorch's DistributedDataParallel framework. The two-stage training curriculum of TACTiS-2 particularly benefits from this parallelization, with Stage 1 distributing marginal distribution estimation across GPUs where each processes a subset of variables, and Stage 2 leveraging data parallelism to accelerate attention mechanism computation.

Hyperparameter optimization employs a distributed trial execution strategy coordinated through a PostgreSQL database backend. Each trial operates independently, enabling embarrassingly parallel execution across multiple compute nodes. The Optuna framework manages trial scheduling, result aggregation, and pruning decisions, with the database ensuring consistency across distributed workers. A typical hyperparameter search evaluates 100–300 configurations, with Hyperband pruning significantly reducing the search space to about a third.

Inference operations are optimized for computational efficiency, with the platform designed to generate forecasts within seconds to support potential control applications.

3.1.6 Design Principles

The platform architecture evolved through iterative refinement based on operational experience and performance analysis. Several key design principles emerged that guide ongoing development and ensure long-term maintainability.

The principle of separation of concerns ensures that each system component has a single, well-defined responsibility. The forecasting model remains agnostic to control strategies, accepting standardized inputs and producing probabilistic outputs without knowledge of downstream usage. Controllers operate independently of forecasting methodology, consuming predictions through abstract interfaces that hide implementation details. This separation enables independent optimization of each component and simplifies system validation through unit testing.

All experiments are fully reproducible through comprehensive configuration management, with YAML files capturing model architecture, training parameters, and data processing steps. Git-based version control tracks code changes, while model checkpoints

and configurations are systematically archived. The PostgreSQL database backend used for hyperparameter optimization maintains a complete history of all tuning experiments, enabling longitudinal performance analysis.

These principles demonstrate that sophisticated probabilistic forecasting methods can be developed and validated for ultra-short-term wind prediction, with the potential for future deployment in real-time control environments. The subsequent sections detail how this architecture is leveraged through the data preprocessing pipeline, model adaptation, and training methodology to achieve state-of-the-art forecasting performance suitable for wind farm applications.

3.2 Data Preprocessing Pipeline

Wind forecasting through deep learning requires converting SCADA measurements into appropriate formats, which needs a specific processing pipeline to address wind farm data uniqueness. The following section explains the SMARTEOLE dataset preprocessing methodology, which utilizes the OpenOA (Open Operational Assessment) toolkit from the National Renewable Energy Laboratory (NREL) for quality control procedures alongside data processing strategies through the Polars DataFrame library. The preprocessing pipeline guarantees that the forecasting model (TACTiS-2) receives clean, properly formatted inputs along with the spatio-temporal relationships needed for accurate wind farm forecasting.

3.2.1 SMARTEOLE Dataset

The SMARTEOLE experiment delivers complete operational data from the 'Sole du Moulin Vieux' wind farm in northern France, which serves as an ideal validation source for ultra-short-term forecasting techniques (Porté-Agel et al., 2020). Seven Senvion MM82 wind turbines at the wind farm generate 2.05 MW of power each from their 80-meter high hubs and 82-meter wide rotors. The turbines form a primarily linear north-south pattern with an average spacing of 330 meters (4.0 rotor diameters) that enables strong wake interactions when winds align.

The dataset spans approximately three months, from February 17 to May 24, 2020, capturing diverse operational conditions including varied wind patterns and turbulence intensities characteristic of the spring season in northern France. Raw SCADA measurements are recorded at 1-minute resolution, capturing the following key variables for each turbine:

- **Wind velocity components:** horizontal wind components u (west-east, positive

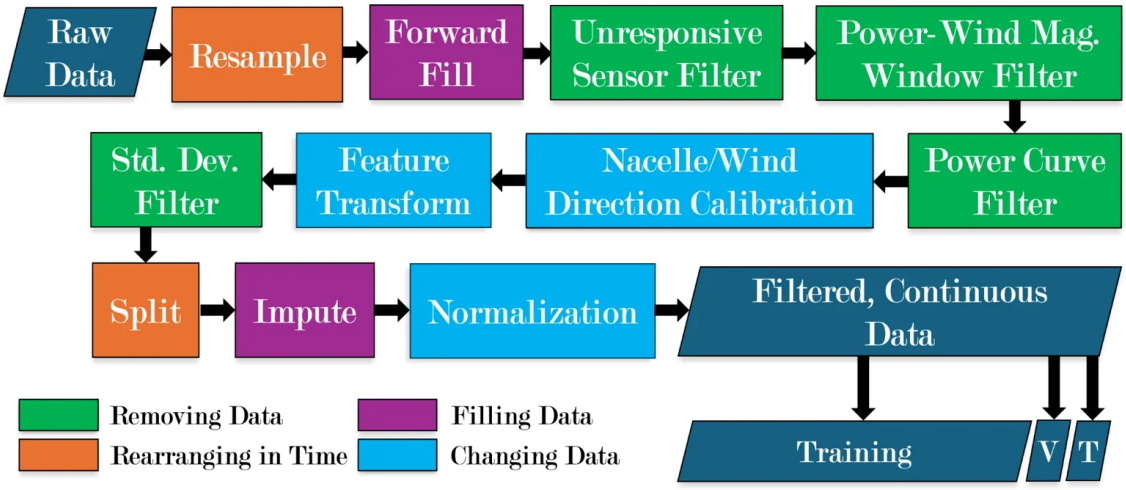

Figure 3.2: Data preprocessing pipeline. The pipeline transforms raw SCADA measurements through multiple quality control and feature engineering stages. Green blocks indicate data removal operations (unresponsive sensor filter, power curve filter, standard deviation filter), orange blocks represent temporal rearrangement (resample, split), purple blocks show data filling procedures (forward fill, impute), and blue blocks denote data transformation steps (nacelle/wind direction calibration, feature transform, normalization). The pipeline ensures high-quality, continuous data suitable for deep learning model training and validation. Adapted with permission from Henry, A. (2025).

eastward) and v (south-north, positive northward) in m/s, derived from nacelle-mounted anemometers.

- **Nacelle position:** yaw angles in degrees relative to true north (0° north, 90° east), indicating turbine orientation.
- **Power output:** active power generation in kW.

The unprocessed data contains about 943,000 points for each variable across 7 turbines during 97 days of measurement (134,661 time steps × 7 turbines), thus requiring advanced preprocessing methods and memory-efficient processing methods.

3.2.2 Data Quality Control

Wind farm SCADA data inherently contains various artifacts and quality issues that must be addressed before model training. The quality control procedure implemented by the OpenOA (Open Operational Assessment) toolkit from NREL employs multiple filtering steps for wind plant operational assessment by applying validated industry-standard filtering methods (Perr-Sauer et al., 2021). The preprocessing pipeline uses a configuration-driven architecture to execute filters, which allows both reproducibility and parameter tuning without code modification. Figure 3.2 illustrates the complete preprocessing workflow.

The quality control framework operates through three distinct processing stages that can be executed independently or sequentially:

- **Data loading** (`--reload_data`): initial ingestion of raw CSV data and conversion to Parquet format.
- **Filter regeneration** (`--regenerate_filters`): recalculation of all quality control masks.
- **Preprocessing application** (`--preprocess_data`): application of filters and transformations.

This modular approach enables efficient reprocessing of specific components without repeating the entire pipeline, particularly valuable during hyperparameter optimization and experimentation.

Sensor Responsiveness Detection: The first quality control stage identifies periods when sensors become unresponsive or 'frozen'. The detection algorithm examines 20 consecutive measurements (20 minutes at 1-minute resolution) for each variable, flagging sequences where values remain exactly constant. The threshold of $\Delta T_u = 1200$ seconds is configurable through the preprocessing configuration file. Detected frozen periods are marked as missing data rather than being treated as valid measurements, preventing the model from learning spurious patterns from sensor malfunctions.

Power Curve Filtering: To ensure the forecasting model learns from normal operational conditions, periods corresponding to non-standard operation or curtailment are filtered using OpenOA's `bin_filter` method (Perr-Sauer et al., 2021). This approach bins power output in 100 kW intervals and identifies outliers based on wind speed deviations exceeding $\Delta|w| = 1.5\,\mathrm{m/s}$ from each bin's median value. The filter operates on power values between 1% and 95% of rated power ($[\underline{p}, \bar{p}] = [0.01p_r, 0.95p_r]$, where $p_r = 2.05\,\mathrm{MW}$), effectively removing periods of curtailment, partial operation, or sensor errors that would otherwise introduce artificial patterns not representative of natural wind-wake dynamics. The use of median-based statistics provides robustness against outliers while preserving legitimate operational variations.

Statistical Range Filtering: The preprocessing pipeline implements multiple range-based filters to identify and remove measurements falling outside physically plausible or operationally relevant bounds. OpenOA's `range_flag` filter enforces absolute bounds of $[|w|_{\min}, |w|_{\max}] = [3, 25]\,\mathrm{m/s}$ for wind speeds, corresponding to typical operational ranges between cut-in and cut-out speeds. Additionally, the `window_range_flag` validates power output within 1%–102% of rated power, accounting for minor measurement uncertainties while removing clearly erroneous readings. These filters ensure that only physically realistic measurements within operational ranges are retained for model training.

Data Availability and Continuity: The preprocessing pipeline addresses data continuity through handling of missing data segments. Extended gaps exceeding $\Delta T_s = 1200$ seconds (20 minutes) trigger the creation of separate continuity groups, ensuring that training examples do not span discontinuous periods that could introduce temporal inconsistencies. Each continuity group represents a contiguous segment of valid data, with groups shorter than the minimum required duration (1200 seconds) being excluded to ensure sufficient temporal context for forecasting. This approach, while not explicitly configured as a separate filter, effectively achieves similar goals to traditional availability filtering by maintaining temporal coherence in the training data.

3.2.3 Feature Engineering

The transformation of raw SCADA measurements into informative features suitable for neural network training requires careful consideration of both mathematical properties and physical interpretations. The feature engineering process balances the need for rich, informative inputs while maintaining computational efficiency and avoiding redundancy.

Cartesian Wind Decomposition: To address the circular discontinuity inherent in directional measurements (the 0°/360° boundary problem), wind velocity vectors are decomposed into Cartesian components. The transformation implemented in the preprocessing pipeline follows:

$$u_i = |w|_i \times \sin\Big((\angle w_i + 180) \times \pi/180\Big), \qquad v_i = |w|_i \times \cos\Big((\angle w_i + 180) \times \pi/180\Big) \quad (3.1)$$

where $|w|_i$ represents wind speed in m/s and $\angle w_i$ represents wind direction in degrees from true north for turbine i. This decomposition ensures continuity in all input variables, essential for gradient-based optimization during neural network training. The horizontal component (`ws_horz`) captures the east-west wind velocity component, while the vertical component (`ws_vert`) represents the north-south component. The 180-degree offset in the implementation aligns with meteorological conventions for wind direction reporting.

Nacelle Direction Encoding: Similar to wind direction, nacelle position measurements are transformed using trigonometric functions to ensure continuity:

$$\gamma_i^s = \sin(\gamma_i \times \pi/180), \qquad \gamma_i^c = \cos(\gamma_i \times \pi/180) \quad (3.2)$$

where γ_i represents nacelle direction in degrees. This dual encoding preserves the circular nature of yaw angles while providing continuous representations suitable for neural network processing (Jammalamadaka and SenGupta, 2001).

Feature Set Composition: The final feature set comprises four primary measurements per turbine:

- **Wind speed** (m/s) – capturing the available kinetic energy in the wind field.
- **Wind direction** (degrees) – determining wake propagation paths and turbine alignment.
- **Power output** (kW) – reflecting turbine operational state and energy extraction.
- **Nacelle position** (degrees) – indicating turbine yaw orientation and control response.

These features are selected based on their direct relevance to wind forecasting and wake modeling dynamics. Through the Cartesian decomposition described above, the wind measurements are transformed into continuous variables (`ws_horz`, `ws_vert`, `nd_cos`, `nd_sin`), resulting in a total of 28-dimensional feature vectors at each timestep (4 primary features $\times$ 7 turbines).

3.2.4 Missing Data Imputation

Despite comprehensive quality control measures, operational wind farm datasets inevitably contain missing data due to sensor failures, communication interruptions, or maintenance periods. The preprocessing pipeline implements a correlation-based imputation strategy leveraging spatial redundancy in wind farm measurements.

The imputation process applies OpenOA's `impute_all_assets_by_correlation` function to execute spatial interpolation between turbines based on their measurement correlation data (Perr-Sauer et al., 2021). The method demands at least a correlation coefficient of $R^2 \geq 0.7$ between turbines to obtain reliable gap-filling results. The method uses measurements from neighboring turbines to estimate missing values when the threshold is achieved, while adjusting for systematic offsets identified during preprocessing. This method makes effective use of wind farm data spatial correlations because nearby turbines share similar wind patterns which are modified by wake effects.

For gaps where spatial interpolation is not feasible (due to insufficient correlation or simultaneous missing data across multiple turbines), the affected time periods are excluded from the dataset. Extended missing data periods exceeding 60 minutes are removed entirely to avoid introducing artificial patterns that could compromise model training. This conservative approach prioritizes data quality over quantity, ensuring that the model learns from reliable measurements rather than potentially biased imputations.

3.2.5 Data Normalization and Standardization

The final preprocessing step involves normalizing feature distributions to facilitate stable and efficient neural network training. The pipeline implements min-max scaling to transform all features to the range $[-1, 1]$, ensuring uniform scale across different measurement

types while preserving the relative relationships within each feature (Goodfellow et al., 2016).

The normalization transformation is defined as:

$$x_{\text{normalized}} = 2 \times \frac{x - x_{\min}}{x_{\max} - x_{\min}} - 1 \tag{3.3}$$

where $x_{\min}$ and $x_{\max}$ represent the minimum and maximum values computed from the training set for each feature type. This scaling approach ensures all features are bounded within the same range, preventing any single feature from dominating the learning process due to scale differences. The normalization parameters are computed exclusively from the training set to prevent data leakage and are applied consistently across training, validation, and test sets (Hastie et al., 2009).

An important implementation detail specific to the TACTiS-2 model: due to its internal batch-wise standardization mechanism, the model requires denormalized data for inference. Therefore, the preprocessing pipeline maintains both normalized data for standard model training and a separate denormalized Parquet file specifically for TACTiS-2 inference, ensuring compatibility with the model's architectural requirements (Ashok et al., 2024).

3.2.6 Efficient Data Processing

The preprocessing pipeline leverages modern data processing technologies to handle the computational demands of large-scale wind farm data efficiently. The implementation utilizes the Polars DataFrame library, which provides several advantages over traditional pandas-based processing (Vink, 2023), such as lazy evaluation, columnar data storage, native parallelisation and native error management.

The pipeline supports configurable multiprocessing through the `--multiprocessor` flag, enabling parallel computation of filters across turbines and concurrent processing of different feature types. Memory usage is managed through a configurable RAM limit (default 75% of available memory) and the use of temporary directories for intermediate results, preventing memory exhaustion during processing of large datasets.

For distributed training environments, the pipeline implements rank-based synchronization to ensure consistency across multiple processes. Only the rank 0 process creates the train/validation/test splits, while other processes wait for completion before loading the preprocessed data. This synchronization mechanism prevents race conditions and ensures all training processes work with identical data splits. Additionally, cache files are named with context-dependent information (e.g. `*_ctx68_train.pkl`), including the context length in the filename to prevent cache collisions when different model configurations are trained simultaneously.

3.2.7 Data Structuring and Splitting

The final preprocessed data is structured into supervised learning examples through a sliding window approach, where each example comprises a context window of historical measurements and corresponding future values to be predicted. The context length is configured proportionally to the prediction horizon based on hyperparameter optimization results (Sengers et al., 2023). For the 210-second prediction horizon (3.5 minutes, used with greedy control strategies), the model utilizes approximately 17.5 minutes of historical context. For the 510-second prediction horizon (8.5 minutes, used with lookup table control), the model employs approximately 42.5 minutes of historical context. This ratio ensures the model has sufficient temporal context to capture relevant wind dynamics, wake propagation patterns, and control responses.

The temporal data splitting follows a chronological scheme to ensure realistic evaluation that mirrors real-world deployment conditions:

- **Training set:** first 70% of the time series.
- **Validation set:** next 10% of the time series.
- **Test set:** final 20% of the time series.

This chronological splitting strategy ensures the model is evaluated on genuinely future data, avoiding the optimistic bias that would result from random splitting in time series applications (Bergmeir et al., 2018). The validation set serves dual purposes: guiding hyperparameter tuning through Optuna optimization (Akiba et al., 2019) and determining early stopping points during training. The test set provides an unbiased estimate of real-world performance on completely unseen future data.

The preprocessing pipeline handles discontinuous data segments through the continuity group mechanism described earlier, ensuring that each training example contains only continuous measurements without gaps exceeding specified thresholds. This approach maintains the temporal integrity of the data while maximizing the utilization of available measurements.

For storage efficiency and fast data access during training, the processed dataset is stored in Parquet format with columnar compression. This format provides significant storage savings compared to raw CSV files while maintaining rapid random access patterns required for mini-batch training (Apache Software Foundation, 2023). The combination of efficient storage, structured preprocessing, and careful data organization enables effective training of deep learning models on the comprehensive spatio-temporal wind farm dataset.

3.2.8 Configuration Management

All preprocessing parameters are externalized in YAML configuration files, enabling complete reproducibility of the preprocessing pipeline and facilitating systematic experimentation with different parameter settings. The configuration system supports dataset-specific profiles (e.g. `preprocessing_inputs_flasc_STORM.yaml`), allowing the same preprocessing codebase to handle different wind farms and data sources through configuration changes alone. This design philosophy ensures that preprocessing decisions are transparent, documented, and reproducible – essential requirements for scientific research and operational deployment.

3.3 TACTiS-2 Model Adaptation

The adaptation of the TACTiS-2 model for ultra-short-term wind farm forecasting required transforming the general-purpose implementation into a production-ready system. While the theoretical foundations are established in Section 2.3, this section documents the engineering efforts to bridge the gap between the research prototype and operational deployment requirements.

3.3.1 Adaptation Requirements and Challenges

The reference implementation of TACTiS-2 (Ashok et al., 2024) utilized the PyTorch Time Series framework with basic training loops that lacked distributed training support, checkpoint management, and comprehensive monitoring capabilities required for production systems. The implementation assumed normalized, complete data without robust mechanisms for handling irregular sampling and missing values characteristic of SCADA systems. Furthermore, the single-optimizer training approach proved unstable for the 28-dimensional wind farm data (7 turbines $\times$ 4 features).

The original two-stage training procedure lacked implementation details for stage transitions, particularly regarding optimizer state management and learning rate scheduling. This necessitated developing a robust transition mechanism to maintain training stability while ensuring convergence of both marginal and copula components. Additionally, real-time control applications require sub-second inference to meet 5–10 second control update cycles, demanding optimization of the inference pipeline without sacrificing prediction accuracy.

3.3.2 PyTorch Lightning Integration

The core architectural adaptation encapsulates TACTiS-2 within a custom PyTorch Lightning module (Falcon and The PyTorch Lightning team, 2019), providing distributed training coordination, automatic mixed precision, and standardized training infrastructure. The integration employs manual optimization control (`self.automatic_optimization = False`) to enable sophisticated stage transition management, as the two-stage curriculum requires replacing the optimizer completely when transitioning from marginal to copula learning.

This manual optimization enables mid-training optimizer replacement, necessary to handle the stage transitions with different loss metrics correctly; stage-specific gradient clipping; fine-grained control over learning rate scheduling through cosine annealing techniques; and direct gradient access for monitoring and debugging.

3.3.3 Two-Stage Training Infrastructure

Stage Transition Mechanism

The implementation performs atomic operations at the epoch boundary when transitioning from Stage 1 (marginal learning) to Stage 2 (copula learning):

1. Update model stage indicator.
2. Freeze marginal encoder and DSF parameters.
3. Create fresh AdamW optimizer with only copula parameters.
4. Initialize new learning rate scheduler for Stage 2.

The fresh optimizer initialization with clean momentum buffers prevents optimization artifacts from Stage 1 affecting copula learning dynamics.

Learning Rate Scheduling

Each stage employs combined linear warmup and cosine annealing:

$$\eta(t) = \begin{cases} \eta_{\text{base}} \cdot \dfrac{t}{T_{\text{warmup}}} & \text{if } t < T_{\text{warmup}} \\ \eta_{\text{base}} \cdot \dfrac{1}{2}\left(1 + \cos\left(\pi \cdot \dfrac{t - T_{\text{warmup}}}{T_{\text{total}} - T_{\text{warmup}}}\right)\right) & \text{otherwise} \end{cases} \tag{3.4}$$

3.3.4 GluonTS Compatibility Layer

Integration with GluonTS (Alexandrov et al., 2020) required addressing tensor format discrepancies. GluonTS expects time series as `[batch, time, series]`, while TACTiS-2 operates on `[batch, series, time]`. The compatibility layer performs:

- Tensor transposition between frameworks.
- Batch-wise standardization using GluonTS scalers.
- Missing value indicator transformation for masked attention.

This ensures seamless data flow while maintaining TACTiS-2's ability to handle missing data through its attention mechanism.

3.3.5 Data Pipeline Adaptations

The wind farm data pipeline implements sliding window generation with configurable context and prediction lengths. For the 210-second prediction horizon (greedy control), the model uses 1050 seconds of context (5:1 ratio). For the 510-second horizon (LUT control), it employs 2550 seconds of context.

TACTiS-2's internal batch-wise standardization requires providing denormalized data, unlike other models that expect pre-normalized inputs. The pipeline maintains separate normalized and denormalized datasets, selecting appropriately based on model configuration.

3.3.6 Implementation Decisions

Several design choices emerged from the adaptation process, including introducing a fresh optimizer manually for Stage 2, which ensures clean momentum initialization and prevents Stage 1 artifacts from affecting the copula learning. This was required to train the copula with a distinct loss metric separated from the marginals learning.

3.4 Training and Optimization

Following the adaptation of the TACTiS-2 model for wind farm forecasting described in Section 3.3, this section presents the training and optimization methodology developed to identify the optimal model configurations and ensure convergence during learning. The implementation is based on the theoretical background of the two-stage curriculum

explained in Section 2.3.3, focusing on the practical implementation of the hyperparameter optimization, distributed training in the HPC, and convergence monitoring required.

3.4.1 Hyperparameter Optimization Framework

The high-dimensional hyperparameter space of transformer-based architectures necessitates intelligent optimization strategies, as exhaustive search becomes computationally intractable with dozens of interdependent parameters, which can reach trillions of possible permutations. The framework employs Optuna (Akiba et al., 2019), implementing a study-trial paradigm where the overall optimization task (study) coordinates evaluation of individual parameter configurations (trials) across distributed computing resources.

Optuna Integration Architecture

The optimization framework, implemented in the `wind_forecasting/tuning/core.py` module, orchestrates the hyperparameter search through a PostgreSQL-backed distributed architecture. This design allows for recovery from failures, asynchronous continuation of studies, and true distributed optimization across multiple compute nodes, which is essential given the lengthy training runs required by TACTiS-2. The studies stored there maintain complete trial histories including intermediate validation metrics, which enable advanced pruning decisions and posterior analysis and refinement of the optimization strategies.

The primary sampling strategy employs the Tree-structured Parzen Estimator (TPE) algorithm (Bergstra et al., 2011), which models $p(x|y)$ rather than the conventional $p(y|x)$, where x represents hyperparameters and y the objective value. As detailed by Bergstra and Bengio (2012), TPE maintains separate probability distributions for well-performing ($l(x)$) and poorly-performing ($g(x)$) configurations, divided by a quantile threshold γ. The implementation uses $\gamma = 0.15$, meaning the top 15% of trials define the "good" distribution, with new configurations selected by maximizing the ratio $l(x)/g(x)$ as an approximation to Expected Improvement.

Hyperband Pruning Algorithm

Computational efficiency is enhanced through the Hyperband algorithm (Li et al., 2017), which frames hyperparameter optimization as a multi-armed bandit problem. The algorithm performs successive halving with a reduction factor $\eta = 2$, progressively eliminating the bottom half of configurations at geometrically increasing resource checkpoints. The implementation configures minimum resources at 2 epochs to allow initial learning dynamics to stabilize, with maximum resources set to the full training budget. This approach

has been shown to reduce optimization time by up to 50× compared to random search while maintaining solution quality (Li et al., 2017).

Distributed Optimization Infrastructure

The distributed optimization uses SLURM job scheduling scripts to submit jobs to the University of Oldenburg STORM HPC cluster. The rank-0 worker creates the Optuna study with specified sampler and pruner configurations, while non-zero ranks implement a retry mechanism with exponential backoff to handle study loading.

3.4.2 Two-Stage Training Implementation

While the theoretical basis for two-stage training is established in Section 2.3.3, the practical implementation requires careful orchestration of optimizer transitions, parameter freezing, and learning rate scheduling. The PyTorch Lightning module employs manual optimization control (`self.automatic_optimization = False`) with TACTiS-2 specifically in order to enable the sophisticated state management required for stage transitions.

Stage Transition Mechanism

The transition from marginal learning (Stage 1) to copula learning (Stage 2) occurs at a configurable epoch boundary, typically epoch 20 based on empirical convergence patterns. The implementation performs atomic operations to ensure training stability:

Listing 3.1: Pseudocode for stage transition.

```
# Pseudocode for stage transition
if current_epoch == stage2_start_epoch:
    # 1. Update model stage indicator
    model.set_stage(2)

    # 2. Freeze marginal parameters
    for param in marginal_parameters:
        param.requires_grad = False

    # 3. Create fresh optimizer for copula parameters
    optimizer_stage2 = AdamW(copula_parameters, lr=lr_stage2)

    # 4. Initialize new scheduler
    scheduler_stage2 = create_scheduler(optimizer_stage2)
```

The fresh optimizer initialization with clean momentum buffers prevents optimization artifacts from Stage 1 affecting copula learning dynamics, addressing stability issues observed in preliminary experiments.

Parameter Group Management

The implementation maintains explicit parameter groups for each stage through careful tracking of module names. Marginal parameters encompass `flow_series_encoder`, `flow_time_encoding`, `flow_input_encoder`, `flow_encoder`, and `decoder.marginal` modules. Copula parameters include `copula_series_encoder`, `copula_input_encoder`, `copula_encoder`, and `decoder.copula` components. This explicit separation ensures correct gradient flow and enables stage-specific hyperparameter tuning.

Learning Rate Scheduling

Each stage uses a compound schedule combining an initial linear warmup with cosine annealing (Equation 3.4), following recommendations from Loshchilov and Hutter (2017) for transformer training. The warmup phase, though configurable, was set to span 10% of total stage steps, with the learning rate linearly increasing from 0 to the base rate. Subsequently, cosine annealing smoothly decays the rate to 10% of the initial value, providing stable convergence while avoiding sharp local minima.

Stage-specific learning rates were implemented to reflect the different optimization landscapes. Stage 1 uses higher learning rates (5×10^{-4} to 5×10^{-3}) for rapid marginal distribution learning, while Stage 2 employs lower rates (1×10^{-4} to 2×10^{-3}) for fine-grained copula optimization.

3.4.3 Dynamic Training Configuration

Batch Size Scaling

To maintain consistent data coverage across different batch size configurations during hyperparameter optimization, the framework implements dynamic scaling of training steps. For a trial with batch size b relative to baseline b_0, the scaling factor $s = b_0/b$ adjusts both `limit_train_batches` and validation frequency. This ensures that models with different batch sizes observe approximately equivalent amounts of data per epoch, enabling fair performance comparison across the hyperparameter space.

Stage-Aware Pruning

The `StageAwarePruner` wrapper for Hyperband was developed to address expected performance degradation during stage transition and avoid early pruning.

Listing 3.2: Pseudocode for stage-aware pruning.

```
# Pseudocode for stage-aware pruning
class StageAwarePruner:
    def prune(self, study, trial):
        current_epoch = len(trial.intermediate_values)

        # Disable pruning during transition buffer
        if in_transition_buffer(current_epoch):
            return False

        # Filter trials to same stage for comparison
        current_stage = get_current_stage(current_epoch)
        comparable_trials = filter_by_stage(study.trials,
                                            current_stage)

        # Apply base pruner on filtered trials
        return base_pruner.prune(comparable_trials, trial)
```

The transition buffer spans 5 epochs after stage change, allowing the copula optimizer to stabilize before resuming pruning decisions. This prevents premature termination of promising configurations experiencing natural transition dynamics.

3.4.4 Hyperparameter Search Space

The search space encompasses 35 hyperparameters, the values of which can be seen in Appendix A. Key architectural parameters include encoder depth (for marginals and copula), embedding dimensions, attention heads and decoder architecture.

Optimization parameters are tuned separately for each stage, including learning rates (log-scale), batch sizes (categorical: 512, 1024, 2048), warmup steps (as fraction or absolute), and weight decay (0 to 0.01, log-scale). The complete search space specification is provided in Appendix A.

3.4.5 Training Protocols

Evaluation Metrics

The primary optimization metric is validation negative log-likelihood (NLL), directly measuring probabilistic prediction quality as established by Gneiting and Raftery (2007). Secondary metrics for inference include the Continuous Ranked Probability Score (CRPS) for calibration assessment, Mean Absolute Error (MAE) for point forecast accuracy using median predictions, and 90% prediction interval coverage (PICP) with normalized width (PINAW) for uncertainty quantification quality.

Convergence Criteria

Multiple criteria determine training termination: maximum epochs (100 for full training, 40 for tuning trials), early stopping on validation loss plateau, and loss plateau detection (change < 0.001 for 5 epochs). These allow balancing computational efficiency with convergence, which is important to better balance exploration versus exploitation during hyperparameter tuning.

3.4.6 Implementation Infrastructure

DataModule Integration

The GluonTS DataModule handles data loading, transformation and batching with intelligent caching. Preprocessed splits are stored in Parquet format with context-dependent naming (e.g. `*_ctx68_train.pkl`) in order to avoid confusion between different model configurations. The rank-aware creation ensures that the splits are performed consistently across distributed processors in the HPC, with rank-0 creating the splits while the others wait for synchronization.

Checkpoint Management

The hierarchical checkpoint organization maintains clear separation between experiments, studies, and trials:

Listing 3.3: Checkpoint directory layout.

```
logs/<project>/<run_id>/checkpoints/<suffix>/trial_<id>/
    |-- epoch=N-val_loss=X.XXX.ckpt  # Best model
    `-- last.ckpt                    # Final state
```

Checkpoints include complete model state, optimizer state, training configuration, and random number generator states, enabling full reproducibility and failure recovery. The checkpoint callback monitors validation loss with `save_top_k=1` to preserve only the best model, reducing storage requirements during large-scale hyperparameter searches.

3.4.7 Experimental Protocol

The complete experimental protocol involves three consecutive phases: hyperparameter optimization (100–300 trials ideally), final model training with the optimal configuration extracted from winning study trials, and final evaluation.

The optimized final hyperparameters are presented in Appendix A.

3.5 Code Availability and Reproducibility

All software developed for this thesis is available as open-source code to ensure reproducibility and facilitate future research. The implementation is organized across four main repositories:

1. **Wind Forecasting Framework:**
   `https://github.com/achenry/wind-forecasting`
2. **PyTorch Transformer Timeseries Models:**
   `https://github.com/boujuan/pytorch-transformer-ts`
3. **Modified Forked GluonTS Library:**
   `https://github.com/achenry/gluonts`
4. **Wind Hybrid Open Controller:**
   `https://github.com/achenry/wind-hybrid-open-controller`

These repositories include comprehensive documentation, configuration files for all experiments, and detailed instructions for reproducing the results presented in this thesis. The modular architecture enables researchers to extend individual components or adapt the framework for different wind farms and forecasting models.

4 Experimental Results and Analysis

This chapter presents the experimental validation of the developed wind forecasting platform through the implementation and evaluation of the TACTiS-2 model on the SMARTEOLE dataset. While the primary focus of this thesis is the development of a comprehensive framework for integrating and deploying state-of-the-art deep learning models for wind farm forecasting, the experimental results provide valuable insights into both the capabilities of the platform and the challenges inherent in probabilistic wind forecasting.

4.1 Experimental Setup and Configuration

The experimental validation was conducted using the complete forecasting pipeline described in Chapter 3, demonstrating the platform's capability to handle the full workflow from raw SCADA data to probabilistic predictions. The TACTiS-2 model was trained following the optimized hyperparameters identified through the Optuna-based tuning framework (Trial 59), with the configuration detailed in Appendix A.

4.1.1 Training Configuration

The model was trained using the two-stage curriculum learning approach on the STORM HPC cluster at the University of Oldenburg, utilizing NVIDIA H100 GPUs with 32 GB memory allocation. The training process successfully completed 41 epochs, with Stage 1 (marginal learning) spanning epochs 0–19 and Stage 2 (copula learning) covering epochs 20–40. The stage transition was automatically managed by the PyTorch Lightning integration, demonstrating the robustness of the implemented training infrastructure.

Training utilized a batch size of 64 with a context window of 360 timesteps (30 minutes at 5-second resolution) and a prediction horizon of 72 timesteps (6 minutes). The small batch size was chosen due to increased convergence during the hyperparameter tuning and not due to memory limitations.

4.1.2 Validation Dataset

The validation was performed on the preprocessed SMARTEOLE test set, from the period February 19–20, 2020. Each initialization generated 200 probabilistic forecast sample

trajectories across the 7 turbines of the wind farm.

4.2 Deterministic Forecasting Performance

The deterministic metrics, computed using the ensemble mean predictions, demonstrate that the platform successfully integrated and trained the TACTiS-2 model to capture the fundamental wind dynamics at the wind farm scale.

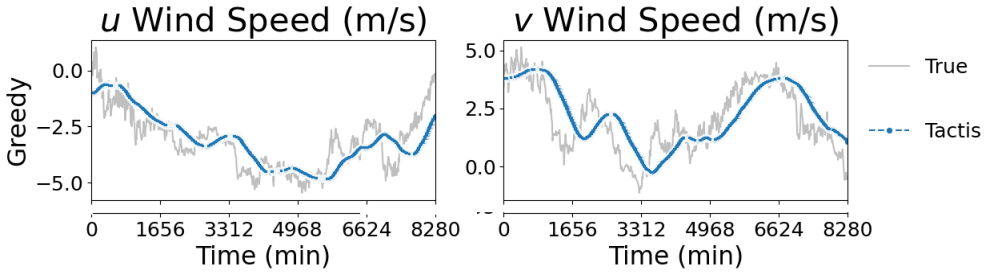

Figure 4.1: Detail of a trajectory of the horizontal (u) and vertical (v) wind speeds: ground truth (grey) versus the forecasted mean (blue).

Figure 4.1 highlights the detail of a trajectory of the horizontal (u) and vertical (v) wind speeds' ground truth versus the forecasted mean.

4.2.1 Point Forecast Accuracy

The model achieved a mean absolute error (MAE) of 0.633 m/s and root mean square error (RMSE) of 0.812 m/s across all turbines and forecast horizons. The mean absolute percentage error (MAPE) of 13.29% indicates reasonable relative accuracy, particularly considering the complexity of wake-affected conditions in the wind farm.

4.2.2 Turbine-Specific Performance

Performance analysis across individual turbines reveals expected variations based on their positions within the wind farm array. Table 4.1 summarizes these findings, and Figure 4.2 presents the aggregated per-turbine errors.

The higher errors observed for Turbine 5 (MAE of 0.898 m/s for the horizontal component) likely reflect its position experiencing complex wake interactions, demonstrating the model's sensitivity to the spatial heterogeneity inherent in wind farm flows. Conversely, Turbine 4's superior performance (MAE of 0.488 m/s) suggests more predictable flow conditions at its location.

Table 4.1: Per-turbine deterministic performance of the TACTiS-2 model on the SMARTEOLE test set.

Turbine	`ws_horz` MAE (m/s)	`ws_vert` MAE (m/s)	`ws_horz` Correlation	`ws_vert` Correlation
1	0.551	0.490	0.495	0.501
2	0.556	0.556	0.516	0.355
3	0.503	0.544	0.616	0.435
4	0.488	0.649	0.499	0.507
5	0.898	1.043	0.463	0.577
6	0.808	0.645	0.303	0.299
7	0.509	0.626	0.415	0.506

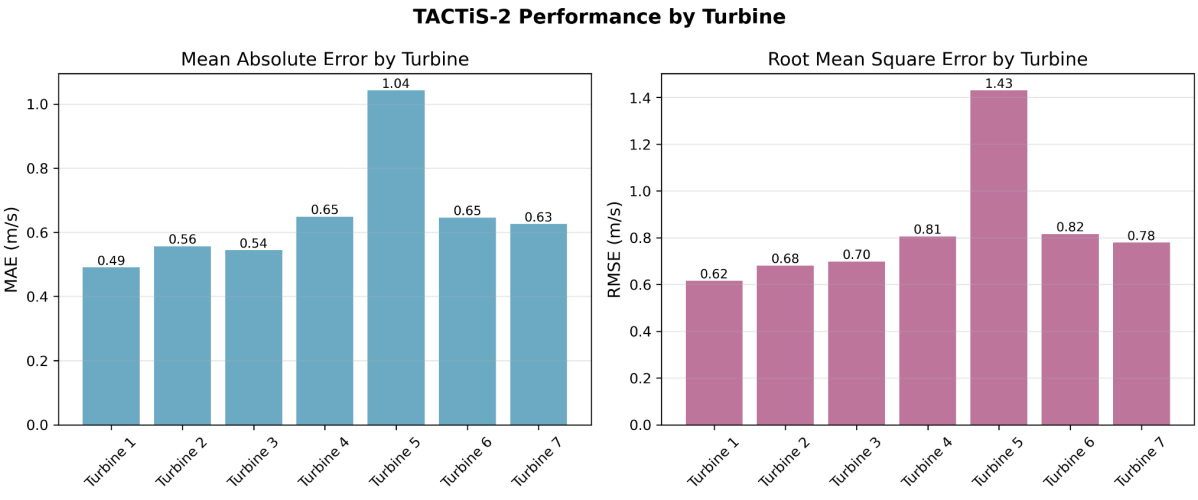

Figure 4.2: TACTiS-2 performance by turbine: mean absolute error (left) and root mean square error (right) aggregated over both wind speed components.

4.3 Probabilistic Calibration Analysis

While the deterministic forecast shows successful training and integration, the probabilistic calibration reveals significant challenges.

4.3.1 Prediction Interval Coverage

The prediction interval coverage probability (PICP) at the 90% confidence level achieved only 5.15%. This severe miscalibration extends across all confidence levels, with even the 95% prediction intervals capturing only approximately 5.15% of observations.

The prediction interval normalized average width (PINAW) values of 0.094 at 90% confidence indicate that the forecast intervals are extremely narrow, confirming that the model has essentially collapsed to near-deterministic predictions despite generating 200 samples.

4.3.2 Technical Analysis of Calibration Failure

The probabilistic calibration failure can be attributed to several factors:

Stage 2 Training Dynamics: Analysis of the training logs reveals that the copula learning phase (Stage 2) exhibited minimal loss improvement after epoch 25, suggesting that the attentional copula mechanism struggled to learn meaningful dependency structures from the wind farm data.

Sample Diversity: Despite generating 200 samples, the extremely low variance suggests that the model defaulted to a near-deterministic mode, potentially due to the dominance of the mean prediction in the loss function.

4.4 Spatial Correlation

Despite the probabilistic calibration challenges, the model demonstrated partial success in capturing spatial dependencies within the wind farm.

4.4.1 Inter-Turbine Correlations

The correlation preservation score of 0.879 indicates that the model maintains much of the spatial structure present in the observations, though with some systematic biases. The mean correlation error of 0.121 suggests room for improvement in capturing the precise

magnitude of wake-induced dependencies. Figure 4.3 shows the correlation score heat maps between turbines.

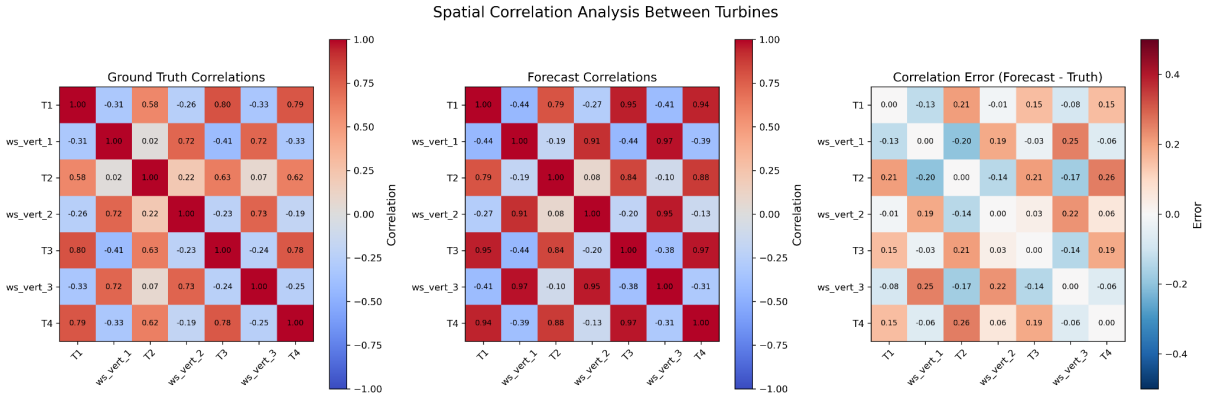

Figure 4.3: Spatial correlation analysis between turbines: ground truth correlations (left), forecast correlations (center), and correlation error (forecast minus truth, right).

4.5 Platform Validation

Beyond the forecasting metrics, the experimental validation successfully demonstrated the platform's operational capabilities and integration features.

The complete training process, including both stages of the curriculum learning, required approximately 8 hours on a single H100 GPU. This is quite good in computational efficiency, considering the model processes 14 variables (7 turbines $\times$ 2 components) with complex attention mechanisms.

The platform's checkpoint management system successfully handled the stage transition at epoch 20, automatically freezing marginal parameters and initializing the copula optimizer without manual intervention. This automation is crucial for the reproducibility and scalability of the framework.

The inference engine managed to validate the dataset with 2160 timesteps on a single H100 GPU in 1.5 hours, averaging approximately 2.5 seconds per 200-sample ensemble.

These metrics preliminarily indicate that the platform would potentially be suitable for real-time control applications on performance alone, given the 5–10 second window for control update cycles.

4.5.1 Platform Integration Features

The experimental validation exercised all major platform components, including:

1. **Data Pipeline:** successfully processed 3 months of SCADA data through the quality control and feature engineering pipeline.

2. **Hyperparameter Optimization:** Optuna integration evaluated 59 trial configurations before identifying optimal parameters.

3. **Distributed Training:** demonstrated compatibility with SLURM-based HPC environments.

4. **Monitoring Integration:** Weights & Biases tracking captured all training metrics and model artifacts.

5. **Checkpoint Management:** automatic model saving and recovery functioned correctly throughout training.

4.6 Summary of Results

The experimental results showed both the successes and challenges encountered in developing an operational wind forecasting platform for deep learning models.

The primary achievement demonstrated by these experiments is the successful integration of a complex, state-of-the-art deep learning architecture into a production-ready forecasting pipeline. The platform handled all aspects of the machine learning workflow, from data preprocessing through model training to inference, without manual intervention. This end-to-end automation is essential for the practical deployment of deep learning models in operational wind farm environments.

The reasonable deterministic performance (MAE of 0.633 m/s) validates that the platform correctly implemented the TACTiS-2 architecture and training procedures. The model learned meaningful representations of wind dynamics, as evidenced by the correlation scores and spatial structure preservation.

The severe miscalibration of probabilistic forecasts, while disappointing from a forecasting perspective, provides valuable insights for future research. The diagnostic revealed that the issue likely lies in the fundamental mismatch between the model's assumptions and the characteristics of wind farm data.

This finding underscores the importance of modular platform design. Alternative probabilistic models can be readily integrated and evaluated using the same infrastructure, enabling systematic comparison of different uncertainty quantification approaches without reimplementing the entire pipeline. In fact, other models such as Informer, Autoformer and Spacetimeformer have already been implemented and are ready for deployment.

5 Discussion

5.1 Interpretation of Results

The experimental validation reveals both achievements and challenges in developing an operational wind forecasting platform. While TACTiS-2's probabilistic calibration failed (5.15% coverage versus 90% expected), the platform successfully integrated and diagnosed this complex deep learning model, demonstrating its value for wind energy applications.

The developed platform automates the complete forecasting workflow, reducing implementation time decisively. Achievements include the successful two-stage training of a 14-variable transformer model, stable operation with automated stage transitions and parameter management, and a modular architecture enabling rapid model substitution.

The MAE of 0.633 m/s validates the platform's core functionality and is competitive with specialized wind forecasting models. The correlation preservation score of 0.879 shows the model learned spatial relationships despite failing at uncertainty quantification. Turbine-specific errors (0.488–0.898 m/s) reflect physical wake effects, confirming the model captures meaningful patterns.

5.2 Scientific Contributions

The validation revealed important findings:

- **Two-stage training challenges:** Stage 2 copula learning requires careful loss function design.
- **Diagnostic importance:** metrics alone provide insufficient insight into model behavior.
- **Infrastructure value:** robust platforms are equally important as a powerful model.

The complete open-source release and reproducible workflows enable verification and extension of results. Configuration files and automated workflows ensure all experiments can be recreated.

5.3 Limitations

Validation used only three months of SMARTEOLE data from a single wind farm. Broader validation across diverse sites and longer periods is needed.

The selection and correct integration of a suitable model for wind forecasting is still a critical point that requires domain expertise, despite the platform's capabilities.

6 Conclusions and Future Work

6.1 Contributions Summary

This thesis developed a production-ready platform for integrating deep learning models into wind farm forecasting systems. The primary contribution is the comprehensive infrastructure that automates the entire workflow from raw SCADA data preprocessing through model training to probabilistic forecast generation. The platform successfully integrated the TACTiS-2 transformer model, demonstrating that complex deep learning architectures can be deployed for operational wind forecasting despite encountering significant challenges in uncertainty quantification.

The platform reduces the time required to implement and evaluate new forecasting models from weeks to hours through its modular architecture and automated pipelines. By integrating industry-standard tools including OpenOA for data quality control, Optuna for hyperparameter optimization, and PyTorch Lightning for distributed training, the system provides a robust foundation for systematic model development and comparison. The comprehensive diagnostic capabilities enabled detailed analysis of model behavior, revealing that TACTiS-2's probabilistic calibration failure stemmed from fundamental architectural incompatibilities rather than implementation issues.

6.2 Future Work

Immediate research priorities should focus on integrating alternative uncertainty quantification methods that avoid distributional assumptions. Dropout-based approaches and quantile regression could provide more robust uncertainty estimates for the non-Gaussian wind data. Physics-informed architectures that incorporate conservation laws and wake dynamics through specialized layers or loss functions represent another promising direction. Model compression techniques including pruning and quantization could reduce computational requirements while maintaining performance, making the platform accessible to smaller operators.

Long-term development should explore cloud-native architectures for improved scalability and accessibility. Integration with real-time streaming platforms would enable continuous forecast updates as new SCADA data arrives. Direct coupling with wind farm controllers would close the loop from forecasting to optimization, though this requires

careful attention to safety and stability constraints. The platform's modular design facilitates extension to related applications including solar forecasting, hybrid renewable systems, and offshore wind farms with different atmospheric dynamics.

6.3 Closing Remarks

This thesis has demonstrated that advancing wind energy forecasting requires equal attention to infrastructure development and algorithmic innovation. The comprehensive platform developed for this research is a significant contribution that improves the accessibility of wind forecasting transformer models and their integration in the future. Being open source, the platform will allow researchers to focus on the fundamental forecasting challenges rather than implementation and infrastructural details.

As the global energy system transitions toward renewable sources, the infrastructure developed in this thesis positions the wind energy community to effectively leverage advances in machine learning for improving wind farm performance and supporting climate goals.

References

Aas, K., Czado, C., Frigessi, A., and Bakken, H. (2009). Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics*, 44(2):182–198. `https://doi.org/10.1016/j.insmatheco.2007.02.001`.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*.

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., and Wang, Y. (2020). GluonTS: Probabilistic and neural time series modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6.

Apache Software Foundation (2023). Apache Parquet. `https://parquet.apache.org`.

Ashok, A., Marcotte, É., Zantedeschi, V., Chapados, N., and Drouin, A. (2024). TACTiS-2: Better, faster, simpler attentional copulas for multivariate time series. In *International Conference on Learning Representations (ICLR)*.

Bastankhah, M. and Porté-Agel, F. (2014). A new analytical model for wind-turbine wakes. *Renewable Energy*, 70:116–123.

Bastankhah, M. and Porté-Agel, F. (2016). Experimental and theoretical study of wind turbine wakes in yawed conditions. *Journal of Fluid Mechanics*, 806:506–541.

Bazionis, I. K. and Georgilakis, P. S. (2021). Review of deterministic and probabilistic wind power forecasting: Models, methods, and future research. *Electricity*, 2(1):13–47.

Bergmeir, C., Hyndman, R. J., and Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70–83.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Bessa, R. J. (2016). On the quality of the Gaussian copula for multi-temporal decision-making problems. In *2016 Power Systems Computation Conference (PSCC)*, pages 1–7. IEEE. https://doi.org/10.1109/PSCC.2016.7541001.

Biewald, L. (2020). Experiment tracking with Weights and Biases. Software available from https://www.wandb.com.

Boersma, S., Doekemeijer, B., Gebraad, P., Fleming, P., Annoni, J., Scholbrock, A., Frederik, J. A., and van Wingerden, J.-W. (2017). A tutorial on control-oriented modeling and control of wind farms. In *Proceedings of the American Control Conference (ACC)*.

Bossanyi, E. (2018). Combining induction control and wake steering for wind farm energy and fatigue loads optimisation. *Journal of Physics: Conference Series*, 1037:032011.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 5th edition.

de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., Hasson, H., Gallinari, P., and Januschowski, T. (2020). Normalizing Kalman filters for multivariate time series analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Drouin, A., Marcotte, É., and Chapados, N. (2022). TACTiS: Transformer-attentional copulas for time series. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 5447–5493.

Falcon, W. and The PyTorch Lightning team (2019). PyTorch Lightning. https://github.com/Lightning-AI/pytorch-lightning.

Fleming, P., King, J., Dykes, K., Simley, E., Roadman, J., Scholbrock, A., Murphy, P., Lundquist, J. K., Moriarty, P., Fleming, K., et al. (2019). Initial results from a field campaign of wake steering applied at a commercial wind farm – Part 1. *Wind Energy Science*, 4(2):273–285.

Foley, A. M., Leahy, P. G., Marvuglia, A., and McKeogh, E. J. (2012). Current methods and advances in forecasting of wind power generation. *Renewable Energy*, 37(1):1–8.

Gilbert, C., Browell, J., and McMillan, D. (2020). Leveraging turbine-level data for improved probabilistic wind power forecasting. *IEEE Transactions on Sustainable Energy*, 11(3):1152–1160. https://doi.org/10.1109/TSTE.2019.2920085.

Global Wind Energy Council (2024). Global wind report 2024. Technical report, GWEC, Brussels, Belgium.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Grigsby, J., Wang, Z., and Qi, Y. (2021). Long-range transformers for dynamic spatiotemporal forecasting. arXiv preprint arXiv:2109.12218.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition.

Henry, A., Simley, E., Pao, L. Y., and Fleming, P. (2024). Deep learning for spatio-temporal wind direction prediction with uncertainty quantification. *Wind Energy Science*. In preparation.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. (2018). Neural autoregressive flows. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Hyndman, R. J., Koehler, A. B., Ord, J. K., and Snyder, R. D. (2008). *Forecasting with Exponential Smoothing: The State Space Approach*. Springer.

Jammalamadaka, S. R. and SenGupta, A. (2001). *Topics in Circular Statistics*. World Scientific.

Jensen, N. O. (1983). A note on wind generator interaction. Technical report, Risø National Laboratory.

Jiménez, Á., Crespo, A., and Migoya, E. (2010). Application of a LES technique to characterize the wake deflection of a wind turbine in yaw. *Wind Energy*, 13(6):559–572.

King, J., Fleming, P., King, R., Martínez-Tossas, L. A., Bay, C. J., Mudafort, R., and Simley, E. (2021). Control-oriented model for secondary effects of wake steering. *Wind Energy Science*, 6(3):701–714.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52.

Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764.

Loshchilov, I. and Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364.

Manwell, J. F., McGowan, J. G., and Rogers, A. L. (2010). *Wind Energy Explained: Theory, Design and Application*. John Wiley & Sons, 2nd edition.

National Renewable Energy Laboratory (NREL) (2023). FLORIS: FLOw Redirection and Induction in Steady State. Version 3.4. `https://github.com/NREL/floris`.

Nelsen, R. B. (2006). *An Introduction to Copulas*. Springer Science & Business Media, 2nd edition. ISBN: 978-0-387-28659-4.

Niayifar, A. and Porté-Agel, F. (2016). Analytical modeling of wind farms: A new approach for power prediction. *Energies*, 9(9):741.

Peña, A., Réthoré, P.-E., and van der Laan, M. P. (2016). On the application of the Jensen wake model using a turbulence-dependent wake decay coefficient: the Sexbierum case. *Wind Energy*, 19(4):763–776.

Perr-Sauer, J., Optis, M., Fields, M. J., Bodini, N., Lee, J. C. Y., Todd, A., Simley, E., Hammond, R., Phillips, C., Lunacek, M., Kemper, T., Williams, L., Craig, A., Agarwal, N., Sheng, S., and Meissner, J. (2021). OpenOA: An open-source codebase for operational analysis of wind farms. *Journal of Open Source Software*, 6(58):2171.

Porté-Agel, F., Bastankhah, M., and Shamsoddin, S. (2020). Wind-turbine and wind-farm flows: A review. *Boundary-Layer Meteorology*, 174:1–59.

Rasley, J., Rajbhandari, S., Ruwase, O., and He, Y. (2020). DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*.

Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. (2021a). Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U., and Vollgraf, R. (2021b). Multivariate probabilistic time series forecasting via conditioned normalizing flows. In *International Conference on Learning Representations (ICLR)*.

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. (2019). High-dimensional multivariate forecasting with low-rank Gaussian copula processes. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191.

Sengers, B. A. M., Rott, A., Simley, E., Sinner, M., Steinfeld, G., and Kühn, M. (2023). Increased power gains from wake steering control using preview wind direction information. *Wind Energy Science*, 8(11):1693–1710.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Simley, E., Fleming, P., and King, J. (2020). Design and analysis of a wake steering controller with wind direction variability. *Wind Energy Science*, 5(2):451–468.

Simley, E., Fleming, P., King, J., and Sinner, M. (2021). Wake steering wind farm control with preview wind direction information. In *Proceedings of the American Control Conference (ACC)*, pages 1783–1789.

Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8:229–231.

Theuer, F., Rott, A., Schneemann, J., von Bremen, L., and Kühn, M. (2022). Observer-based power forecast of individual and aggregated offshore wind turbines. *Wind Energy Science*, 7(5):2099–2116. `https://doi.org/10.5194/wes-7-2099-2022`.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Vink, R. (2023). Polars: Blazingly fast DataFrames in Rust and Python. `https://pola.rs`.

Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yatiyana, E., Rajakaruna, S., and Ghosh, A. (2017). Wind speed and direction forecasting for wind power generation using ARIMA model. In *2017 Australasian Universities Power Engineering Conference (AUPEC)*.

Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1):35–62.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

A TACTiS-2 Hyperparameters

This appendix lists the complete hyperparameter search space explored during the Optuna-based optimization together with the values chosen for the final model (Trial 59).

Table A.1: Training and optimization parameters.

Hyperparameter	Search Range	Chosen Value	Description
`lr_stage1`	$[2 \times 10^{-6}, 1 \times 10^{-5}]$ (log)	3.22×10^{-6}	Learning rate for Stage 1
`lr_stage2`	$[1 \times 10^{-5}, 2 \times 10^{-3}]$ (log)	1.60×10^{-4}	Learning rate for Stage 2
`weight_decay_stage1`	$\{0.0, 10^{-6}, 10^{-7}\}$	1.00×10^{-6}	Weight decay for Stage 1 optimizer
`weight_decay_stage2`	$\{0.0, 2 \times 10^{-5}, 10^{-5}, 5 \times 10^{-6}, 10^{-6}\}$	5.00×10^{-6}	Weight decay for Stage 2 optimizer
`stage2_start_epoch`	Fixed	40	Epoch at which Stage 2 training begins
`eta_min_fraction_s1`	$[10^{-3}, 0.05]$ (log)	0.0124	Fraction of initial LR for Stage 1 cosine annealing $\eta_{\min}$
`eta_min_fraction_s2`	$[10^{-5}, 0.01]$ (log)	1.43×10^{-5}	Fraction of initial LR for Stage 2 cosine annealing $\eta_{\min}$
`gradient_clip_val_stage1`	$\{0, 1.0, 3.0, 5.0\}$	1.0	Gradient clipping norm for Stage 1
`gradient_clip_val_stage2`	$\{0, 1.0, 3.0, 5.0, 10.0\}$	0	Gradient clipping norm for Stage 2 (0 = no clipping)

Table A.2: Marginal CDF encoder architecture.

Hyperparameter	Search Range	Chosen Value	Description
`marginal_embedding_ dim_per_head`	$\{16, 32, 64, 128, 256, 512\}$	128	Embedding dimension per attention head for marginal encoder

Table A.2: Marginal CDF encoder architecture (continued).

Hyperparameter	Search Range	Chosen Value	Description
`marginal_num_heads`	$[2, 6]$	6	Number of attention heads in marginal encoder
`marginal_num_layers`	$[3, 5]$	3	Number of transformer layers in marginal encoder
`flow_input_encoder_ layers`	$[3, 5]$	4	Number of input encoding layers for flow network
`flow_series_embedding_ dim`	{5, 8, 16, 32, 64, 128, 256}	8	Time series embedding dimension for flow network
`stage1_activation_function`	{relu, gelu}	relu	Activation function for Stage 1 components

Table A.3: Attentional copula encoder architecture.

Hyperparameter	Search Range	Chosen Value	Description
`copula_embedding_dim_ per_head`	{8, 16, 32, 64, 128, 256}	16	Embedding dimension per attention head for copula encoder
`copula_num_heads`	$[2, 6]$	2	Number of attention heads in copula encoder
`copula_num_layers`	$[1, 3]$	3	Number of transformer layers in copula encoder
`copula_input_encoder_ layers`	$[2, 4]$	4	Number of input encoding layers for copula network
`copula_series_ embedding_dim`	{16, 32, 48, 64, 128, 256}	64	Time series embedding dimension for copula network
`ac_mlp_num_layers`	$[3, 6]$	5	Number of layers in attentional copula MLP

Table A.3: Attentional copula encoder architecture (continued).

Hyperparameter	Search Range	Chosen Value	Description
`ac_mlp_dim`	{32, 64, 128, 256}	32	Hidden dimension of attentional copula MLP layers
`stage2_activation_function`	{relu, gelu}	relu	Activation function for Stage 2

Table A.4: Decoder architecture.

Hyperparameter	Search Range	Chosen Value	Description
`decoder_dsf_num_layers`	$[1, 4]$	3	Number of Deep Sigmoidal Flow layers
`decoder_dsf_hidden_dim`	{48, 64, 128, 256, 512}	128	Hidden dimension of DSF layers
`decoder_mlp_num_layers`	$[2, 5]$	3	Number of MLP layers in decoder
`decoder_mlp_hidden_dim`	{8, 16, 32, 48, 64, 128, 256}	8	Hidden dimension of decoder MLP
`decoder_transformer_ num_layers`	$[2, 5]$	5	Number of transformer layers in decoder
`decoder_transformer_ embedding_dim_per_head`	{32, 64, 128}	128	Embedding dimension per head in decoder transformer
`decoder_transformer_ num_heads`	$[3, 5]$	4	Number of attention heads in decoder transformer
`decoder_num_bins`	{50, 100, 200, 300}	50	Number of bins for copula discretization (resolution)

Table A.5: Regularization and general configuration.

Hyperparameter	Search Range	Chosen Value	Description
`dropout_rate`	{0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.015}	0.008	Dropout rate for regularization
`encoder_type`	{standard, temporal}	temporal	Type of encoder architecture
`batch_size`	{64, 128, 256, 512}	64	Training batch size
`context_length_factor`	Fixed	5	Factor to determine context window (context_length = factor × prediction_length)

Table A.6: Fixed training configuration.

Hyperparameter	Search Range	Chosen Value	Description
`max_epochs`	Fixed	100	Maximum number of training epochs
`val_check_interval`	Fixed	1.0	Validation frequency (1.0 = every epoch)
`trainer.gradient_clip_val`	Fixed	1.0	Global gradient clipping value
`seed`	Fixed	420	Random seed for reproducibility
`dataset.sampler`	Fixed	random	Data sampling strategy
`prediction_length`	Fixed	72	Number of future time steps to predict (360 s / 5 s = 72)
`context_length`	Fixed	360	Number of past time steps used as context (5× 72 = 360)