Calibrated Prediction Intervals with Deep Transformer-Based Ensembles for Long-Term Time Series Forecasting

By

Mebratu Teklehaimanot Asfaw

Submitted to the School of Information Technology and Engineering

In partial fulfillment of the requirements for the degree of

Master of Science in Artificial Intelligence

Supervised by: Dr. Fantahun Bogale

College of Technology and Built Environment

Addis Ababa University

Addis Ababa, Ethiopia

February, 2026

APPROVAL

This is to certify that this thesis titled ***Calibrated Prediction Intervals with Deep Transformer-Based Ensembles for Long-Term Time Series Forecasting*** is prepared by **Mebratu Teklehaimanot Asfaw**, has been submitted in partial fulfillment of the thesis-option requirements for the degree of Master of Science in Artificial Intelligence at the School of Information Technology and Engineering, College of Technology and Built Environment, Addis Ababa University.

Name	**Signature**	**Date**
Dr. Fantahun Bogale		
(Advisor)		
Dr. Million Meshesha		26/03/2026
(External Examiner)		
Dr. Beakal Gizachew		
(Internal Examiner)		
Dr. Sileshi Demesie		
(Chairperson)		

Dedication

To my family.

To my academic advisor.

To myself.

Acknowledgements

First and foremost, I would like to thank **GOD** for His guidance and strength throughout every moment of completing this study.

I sincerely thank my advisor, **Fantahun B. (PhD)**, for his invaluable guidance, feedback, and unwavering support. His extensive support was instrumental in the completion of this dissertation. I would also like to acknowledge the knowledge and skills I gained through the coursework and research training provided by **Addis Ababa university/Artificial Intelligence**, which laid the foundation for this work. Additionally, I gratefully acknowledge the publicly available datasets that made this research possible, which can be accessed at **Google Drive**.

I would like to express my deepest gratitude to my family for their endless love, patience and encouragement. Your unwavering belief in my dreams has been a driving force and source of comfort.

Finally, I extend my gratitude to all who contributed directly or indirectly to this thesis by providing support, resources, and thoughtful input.

ABSTRACT

Long-term time series forecasting (LTSF) is crucial for decision-making in domains like energy and finance; however, deep learning models, including Transformers, often produce overconfident point predictions with miscalibrated uncertainty estimates. While prior quality-driven approaches, including QD, Dual-AQD, and Sum-K, have advanced prediction interval generation through gradient-based optimization, they remain constrained by manual hyperparameter tuning that limits adaptability across datasets. This study addresses this challenge by introducing the Self-Adaptive Quality-Driven (SA-QD) loss function, which incorporates learnable parameters to dynamically balance prediction interval (PI) coverage and sharpness without manual tuning. Integrated with deep Transformer-based ensembles leveraging the PatchTST architecture and enhanced by Horizon-Specific Conformal Prediction (HSCP) for final calibration, the framework effectively captures both aleatoric and epistemic uncertainties across varying forecast horizons. The proposed method was evaluated on Exchange Rate (financial) and ETTh2 (energy) benchmark datasets over horizons of 96, 192, 336, and 720 steps and compared against state-of-the-art baselines (QD, Dual-AQD, Sum-K). Results demonstrate that SA-QD achieves high (PICP: 0.961 ± 0.009 on Exchange Rate; 0.987 ± 0.008 on ETTh2), close to the nominal 95%, while maintaining competitive MPIW and outperforming baselines in IS and CRPS, with reductions of 56–62% in combined metrics, particularly against Dual-AQD and on ETTh2. Per-horizon analyses confirm robust adaptation to escalating uncertainty, with stable training and reduced overfitting. This work advances automated uncertainty quantification in LTSF, mitigating overconfidence and bolstering reliability in high-stakes applications. Future extensions may incorporate multimodal data and explore additional domains, such as biomedicine.

Keywords: Long-Term Time Series Forecasting; Prediction Interval Coverage Probability; Self-Adaptive Quality-Driven; Mean Prediction Interval Width; Continuous Ranked Probability Score; Transformer Ensembles.

Contents

List of Acronyms and Abbreviations

Symbol	Meaning
AB	All-Batch
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
BNNs	Bayesian Neural Networks
CNN	Convolutional Neural Network
CLT	Central Limit Theorem
CP	Conformal Prediction
CRPS	Continuous Ranked Probability Score
CWC	Coverage Width-based Criterion
DNNs	Deep Neural Networks
DSRP	Design Science Research Process
DualAQD	Dual Accuracy-Quality-Driven
ETT	Electricity Transformer Temperature
ETTh2	Electricity Transformer Temperature hourly variant 2
FEBs	Fourier Enhanced Blocks
GD	Gradient Descent
HQ	High-Quality
HSCP	Horizon-Specific Conformal Prediction
LLMs	Large Language Models
LSTM	Long Short-Term Memory
LTSF	Long-Term Time Series Forecasting
LUBE	Lower Upper Bound Estimation
MA	Moving Average
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
MI	Mutual Information
MPIW	Mean Prediction Interval Width
MSE	Mean Squared Error
MVE	Mean-Variance Estimation
NLL	Negative Log-Likelihood

Symbol	Meaning
NN	Neural Network
PAM	Pyramidal Attention Module
PICP	Prediction Interval Coverage Probability
PINAW	Prediction Interval Normalized Average Width
PINALW	Prediction Interval Normalized Average Large Width
PIs	Prediction Intervals
QD	Quality-Driven
QR	Quantile Regression
RevIN	Reversible Instance Normalization
RNNs	Recurrent Neural Networks
SA-QD	Self-Adaptive Quality-Driven Loss
SNM	Split Normal Mixture
STB	Spatiotemporal MLP Backbone
UAI	Uncertainty in Artificial Intelligence
UQ	Uncertainty Quantification
VI	Variational Inference

List of Figures

LIST OF TABLES

Chapter 1

INTRODUCTION

Long-term time series forecasting (LTSF), a fundamental research area concerned with predicting future values based on historical observations in sequential data over extended horizons, stands as a cornerstone in numerous disciplines, ranging from economics and finance to environmental science, healthcare, and engineering. Its pervasive utility stems from the inherent temporal dependencies present in data generated across virtually every sector of human endeavor. Accurate and reliable forecasts are critical for strategic planning, resource allocation, risk management, and operational optimization. For instance, in energy markets, precise predictions of electricity demand are vital for grid stability and efficient power generation scheduling [26, 27]. In financial markets, forecasting stock prices or market volatility can inform investment decisions and risk assessment [28]. Similarly, in supply chain management, anticipating future product demand directly impacts inventory levels, production schedules, and logistics, minimizing costs and maximizing customer satisfaction[29].

In the past, classic statistical methods were commonly used for time series forecasting, including autoregressive (AR) [30], moving average (MA) [31], autoregressive moving average (ARMA) [32], AR integrated MA (ARIMA) [33], and spectral analysis techniques [34]. But these approaches rely on strict assumptions about the data like stability, normal distribution, linear ties, and independence which often fail to hold in real life. For instance, AR, MA, and ARMA models assume the time series is stationary, yet many actual datasets show non-stationary traits, making these methods less effective for practical use. This Leads to sub-optimal performance and unreliable predictions [35]. That's why researchers have turned to machine learning (ML) for better handling LTSF [36, 37, 38, 39, 40], with tools like support vector machines (SVMs) [41] and adaptive boosting (AdaBoost) [42] stepping in for TSF tasks.

The advent of deep learning has revolutionized the field of time series forecasting, offering powerful alternatives capable of learning intricate patterns from complex data. Notably, recurrent neural networks (RNNs) [43] and their variants, such as long short-term memory (LSTMs) [44] and gated recurrent units (GRUs) [45, 46, 47], have emerged as prominent choices for sequential data processing. However, their sequential nature poses challenges, processing inputs and back-propagating can create real hurdles, particularly with data that has long-term dependencies. LSTMs and GRUs often run into issues like gradients vanishing or exploding during training model of very long sequences effectively [48]. While tweaks to the architecture or training methods can ease these gradient troubles slightly, RNN-based approaches might still fall short in terms of performance and speed [49]. Additionally, convolutional neural networks (CNNs) have been adapted for time series tasks as an alternative.

The most recent paradigm shift in sequence modeling[50], and consequently in time series forecasting, has been the emergence of Transformer architectures[20]. Originally designed for natural language processing (NLP) tasks [51, 52], with their revolutionary self attention mechanism, have demonstrated unparalleled capabilities in capturing complex dependencies irrespective of position in a sequence. This non-sequential processing, combined with their inherent parallelizability, renders them particularly well-suited for long-term time series forecasting (LTSF), where the ability to model relationships over extended time horizons is paramount [53]. Models such as Informer [21], Autoformer [22], PatchTST [23] , and TimeMixer[24] have pushed the boundaries of point prediction accuracy in LTSF, establishing new benchmarks across diverse datasets. Their success lies in their capacity to extract rich, contextualized representations from time series data, enabling more accurate and robust forecasts over extended periods.

Despite these remarkable advancements in point forecasting, a critical challenge persists in reliable quantification of uncertainty, characterized by the miscalibration and overconfidence of modern deep neural networks (DNNs)[54, 55, 56]. In high-stakes applications, a single point forecast, no matter how precise, is insufficient for robust decision-making. Decision makers require not only an estimate of the most likely future outcome

but also a clear understanding of the potential range of outcomes and the confidence associated with those predictions[57, 58]. This necessity underscores the growing importance of prediction intervals (PIs), which provide a range within which future observations are expected to fall with a specified probability.

Mathematically, a prediction interval for a future observation y_{t+h} at horizon h is defined as $[L_{t+h}, U_{t+h}]$ such that:

$$P(L_{t+h} \leq y_{t+h} \leq U_{t+h}) = 1 - \alpha$$

where $(1-\alpha)$ represents the nominal coverage probability. The quality of PIs is typically evaluated based on two primary criteria coverage, which refers to the proportion of actual observations falling within the predicted intervals, and sharpness, which denotes the narrowness of these intervals[59]. An ideal PI is one that is both well-calibrated (i.e., its empirical coverage matches its nominal coverage, e.g., 95% of actual values fall within 95% PIs) and as narrow as possible without sacrificing coverage [60]. However, deep learning models, including Transformers, often exhibit overconfidence in their predictions, leading to miscalibrated PIs that are too narrow and thus fail to capture the true uncertainty, especially as the forecast horizon increases due to error accumulation as a limit to accuracy [1, 61]. The following section examines the specific factors that motivate the need for automated and reliable uncertainty quantification in deep learning-based LTSF.

1.1 Motivation of the Study

The motivation for this research stems from the growing gap between the increasing deployment of deep learning models in safety-critical forecasting applications[3, 62] and their well-documented tendency to produce overconfident, poorly calibrated predictions. In energy infrastructure, unreliable demand forecasting carries severe real-world consequences the North American Electric Reliability Corporation (NERC) warned in its 2023–2024 Winter Reliability Assessment that two-thirds of North American electricity consumers faced elevated blackout risk[63], attributing a key cause to systematic demand

underestimation by forecasting models, while the U.S. Department of Energy further cautioned that power outages could increase by a factor of 100 by 2030 if forecasting and grid management failures persist. In financial markets, the collapse of Zillow's algorithmic home-buying program in 2021 resulting in over $500 million in losses demonstrated concretely how overconfident point predictions without uncertainty quantification can cascade into catastrophic institutional failures[64]. Beyond energy and finance, the International Monetary Fund warned in 2024 that AI forecasting models trained on stale data could amplify economic downturns by triggering cascading supply chain errors[65], while in healthcare and autonomous systems, miscalibrated model confidence has been repeatedly linked to patient safety failures and unreliable decision support.

1.1.1 Uncertainty Quantification in Deep Learning

What really fuels UQ is the realization that basic DNNs often fall short in demanding scenarios, largely because they come across as too sure of themselves when they shouldn't [55, 57].

1.1.2 Ensuring Safety in Critical Applications

Putting DNNs to work in high-risk areas calls for solid UQ to keep things safety [3]. Take medical diagnostics, for example, where knowing the uncertainty behind a prediction can mean the difference in patient outcomes [3]. The same holds for autonomous tech like self-driving cars or robots, which need dependable uncertainty readings to handle surprises or fuzzy sensor data, all while meeting strict safety rules for vehicles [66, 67, 68]. And in wider risk strategies whether exploring new paths in reinforcement learning or adapting on the fly well-calibrated UQ helps spot dangers, skip unreliable moves, and foster more resilient systems [62, 69, 70, 71, 72]. Such safety concerns are amplified by DNNs' tendency toward overconfidence, where models assign high probabilities to incorrect predictions, directly undermining risk assessment in these applications.

1.1.3 Addressing Overconfidence and Miscalibration

This overconfidence, often manifesting as miscalibration, further drives UQ research by a big spark for UQ studies comes from the frequent observation that today's networks often get their confidence wrong, creating vulnerabilities in niche fields [55, 73]. This rings especially true for Large Language Models (LLMs), where shaky calibration ties directly to "hallucinations" churning out false or deceptive info [74, 75]. In turn, efforts have ramped up to measure uncertainty and boost trust in LLM outputs, including ways to weave doubt into natural language responses [76, 77, 78].

Shifting gears to modern Transformer models in Long-Term Time Series Forecasting (LTSF), the impetus changed dramatically when straightforward linear approaches questioned the worth of elaborate attention setups [20]. From there, key innovations like PatchTST and TimeMixer stepped in to build tougher systems and dial back core uncertainties, such as creeping errors or shifting patterns as forecasts stretch out [23, 79]. At its core, progressing LTSF is about meeting urgent practical demands for forward planning, better efficiency, and sharp risk insights in vital sectors like energy handling [80, 81] and finance [82, 83]. These rely on seeing far ahead to dodge threats, avoid breakdowns (think power grid failures [84]), and guide tricky health calls [85, 86]. On the tech front, the focus lies in crafting designs that grasp distant links effectively something older looping models like RNNs or LSTMs struggle with due to issues like snowballing mistakes and tough long-haul training [87]. That's where the Transformer shines, adopted for its non-linear attention that spots overarching patterns without the usual pitfalls [53, 88].

This research draws from a tough, ongoing issue in harnessing top-tier deep learning, especially Transformers, for long-term time series forecasting: achieving dependable, hands-free ways to gauge prediction doubts. These models undoubtedly deliver strong single-point forecasts, yet their tangled, hard-to-probe mechanics often cloud how much faith to place in the results. Without clear uncertainty signals, everyday decisions risk significant drawbacks say, a too-certain energy projection causing outages or a doubtful one wasting supplies [89].

In today's UQ world for deep learning, tools frequently wrestle with a basic pull be-

tween spot-on predictions and well-tuned, focused intervals. Leading methods do lift interval standards, but they still demand a lot of manual fiddling with settings. This hands-on work isn't just annoying; it's a real stumbling block in both ideas and application. Hitting the sweet spot for interval reliability and focus often depends on finicky weights in loss designs, like those in Quality-Driven systems [90]. Nailing those means grinding through resource-heavy trials, from grid hunts to random scans [91]. Beyond being slow, this tuning ties tightly to the data, confidence aims, and series quirks meaning one fix might bomb elsewhere, hobbling these approaches in fluid real settings [92].

Long-term forecasting is specifically targeted in this study because uncertainty quantification becomes both more critical and more challenging as the prediction horizon extends. While short-term forecasts over one to twenty-four steps typically maintain reasonable accuracy and narrow prediction intervals, long-term forecasts over 96 to 720 steps face compounding error accumulation where both aleatoric and epistemic uncertainties grow non-linearly with the horizon length [93, 94, 95]. This non-linear growth creates a fundamental challenge for existing prediction interval methods that treat all horizons uniformly with fixed hyperparameters [50, 21]. As Zhou et al. [21] demonstrated with the Informer architecture, the degradation of forecast quality over extended horizons necessitates specialized approaches that can adapt to the changing uncertainty landscape. Furthermore, the practical stakes of long-term predictions are inherently higher than those of short-term forecasts: long-term decisions involving infrastructure investment, capacity planning, and strategic risk management entail larger commitments and greater costs of error, making reliable uncertainty bounds not merely desirable but essential for responsible decision-making.

1.2 Statement of the Problem

Accurate predictive models are essential for making informed and reliable decisions in many scientific and engineering fields[96, 7, 97, 98], such as precision agriculture[8], electrical load forecasting[99, 100], and circuit board fault detection[101]. However, uncertainties in data and models make it necessary to measure how reliable predictions

are[102, 103, 9]. This is often done using Prediction Intervals (PIs), which provide a range around the predicted value to show possible outcomes[104, 10].

The construction of dependable PIs in time series forecasting requires a careful balance between two fundamental quality measures: Prediction Interval Coverage Probability (PICP), which checks if the intervals reliably capture the actual values[7, 105], and Mean Prediction Interval Width (MPIW), which measures how narrow and useful the intervals are[7, 104, 11]. The main difficulty comes from the natural conflict between these goals: making intervals more reliable (higher PICP) often means making them wider (higher MPIW), while narrowing them (lower MPIW) can make them less reliable[105, 11]. This creates a complex trade-off challenge.

Early research on building PIs used a two-step process: first, training neural networks by reducing errors (like sum of squared errors), then creating intervals from the trained model. This method was criticized for focusing too much on single-point predictions instead of improving interval quality, such as coverage and narrowness. Lower Upper Bound Estimation (LUBE), one of the first methods to directly generate PIs, combined PICP and MPIW in its loss function using a Coverage Width-based Criterion (CWC)[106]. This turned the trade-off into a simpler single-goal problem[99, 7]. However, the original CWC setup was not suitable for smooth optimization, needing costly search methods like Particle Swarm Optimization[13], and it often led to less-than-ideal results[7].

Quality-Driven (QD) loss functions, including those in QD-Ens that built on the idea of high-quality intervals, fixed LUBE's problems by creating a smooth framework that works with gradient descent training[11]. Still, these methods kept a major flaw: they depended on a manually adjusted multiplier (e.g., λ or δ) where λ denotes the weighting coefficient that balances the coverage penalty against the interval width objective in quality-driven losses such as QD [11], and δ represents the target miscoverage threshold in formulations such as Sum-K [9] to weigh interval narrowness against coverage. This value is easily affected by small adjustments, which can greatly reduce interval quality[11, 8], requiring hands-on tuning for good outcomes. This tuning gets even harder with extra terms in the loss, as in QD+, which added goals for point predictions (like mean squared

error). This brings in several adjustable values $(\lambda_1, \lambda_2, \xi)$ to balance the main goals (interval narrowness, coverage, and accuracy), making selection complicated and often needing manual tweaks[10].

Other methods, like Tube Loss, use a δ value to shrink interval widths while keeping balance in real-world checks, but this still needs smart tuning[13]. Recent improvements, such as punishing overly wide intervals with the sum of the largest K parts in loss functions, help reduce broad ranges in renewable energy predictions, but combining them with automatic tuning is still limited[9]. Across studies, experts agree that manual tuning is a "difficult and time-consuming" process, often taking "many days per try"[8]. The best values for these adjustments often depend on "the scale of measurements" and "can vary a lot by use case"[8, 11]. Even methods with self-adjusting factors face issues; for example, while the factor changes during training, the speed of change (like DualAQD's η) still needs manual setup[8]. This problem is highlighted as a main area for future research, with suggestions for trade-off methods that "automatically tweak the balance during training"[9].

A further critical gap in existing methods is that they are not tailored to different forecast lengths; they assume the same level of uncertainty across all long-term time series forecasting horizons. As the forecast horizon extends, prediction errors compound non-linearly, with the variance of multi-step-ahead forecasts growing faster than linearly due to error propagation through temporal dependencies [93, 94, 95, 22], meaning that a fixed coverage-sharpness trade-off parameter such as λ in the QD loss that produces valid intervals at H=96 may produce either excessively wide or insufficiently narrow intervals at H=720. This challenge is further compounded by distributional shifts across horizons, where short-horizon errors tend to be approximately symmetric and light-tailed while long-horizon errors may exhibit skewness, heavy tails, and non-stationarity [20, 23], causing methods that assume fixed error distributions across all horizons to fail.

Building on these identified gaps the persistent need for manual tuning despite increasingly differentiable frameworks, the absence of horizon-aware adaptation, the limited handling of varying data scales, and the lack of integrated mechanisms for decomposing

data noise from model uncertainty this research develops an automated and self-adaptive framework for uncertainty quantification that generates narrow, reliable PIs without extensive manual hyperparameter tuning.

1.3 Research Questions

RQ1 How can a Self-Adaptive Quality-Driven (SA-QD) loss function automatically balance coverage and sharpness in PIs for long-term time series forecasting?

RQ2 How does conformal prediction, combined with deep transformer ensembles, affect the calibration PIs in long-term time series forecasting?

1.4 Objectives of the Study

1.4.1 General Objective

To design and evaluate a self-adaptive, ensemble-based uncertainty quantification framework that automatically generates calibrated prediction intervals for long-term time series forecasting, ensuring consistent satisfaction of the nominal coverage target without manual hyperparameter tuning.

1.4.2 Specific Objectives

- To develop a Self-Adaptive Quality-Driven (SA-QD) loss function with gradient-optimized learnable parameters (λ, μ) that dynamically balance prediction interval coverage and sharpness during training without manual hyperparameter tuning.
- To design and implement an integrated three-stage framework combining deep PatchTST ensembles for epistemic uncertainty estimation, the SA-QD loss for adaptive interval optimization, and Horizon-Specific Conformal Prediction (HSCP) for distribution-free post-hoc calibration.
- To assess the framework's ability to maintain valid coverage ($PICP \geq 0.95$) while adapting prediction interval characteristics across varying forecast horizons from

96 to 720 steps, accommodating the non-linear growth of uncertainty in long-term predictions.

- To comparatively evaluate the proposed framework against state-of-the-art baselines (QD, Dual-AQD, Sum-K) using a hierarchical evaluation protocol that prioritizes coverage validity before comparing interval sharpness, Interval Score, and Continuous Ranked Probability Score.

1.5 Significance of the Study

This research is important because it fills a crucial void in deep learning for long-term time series forecasting by adding reliable, automated uncertainty estimates to Transformer models. Through learnable parameters that balance prediction interval coverage and sharpness, it furthers our knowledge of adaptive losses, fixing overconfidence and calibration problems in key areas. In real terms, it enables safer forecasts in energy and finance, reducing risks like blackouts from flawed predictions or wasted resources. By minimizing manual tuning, it improves efficiency and scalability for diverse real-world scenarios. Overall, it advances safer AI in autonomous systems and healthcare, inspiring new ideas in LTSF, and creates a more independent UQ process that makes advanced forecasting available to more people and uses.

1.6 Contributions of the Study

- **Adaptive Loss Function Design:** Introduces a fully differentiable, adaptive quality-driven loss function with learnable parameters to automatically optimize the coverage-sharpness trade-off, representing a key advancement in PI methodology for LTSF.

- **Horizon-Aware Uncertainty Quantification:** Develops mechanisms enabling models to dynamically adjust PI characteristics based on forecast horizons, addressing a critical gap in current LTSF uncertainty frameworks. Demonstrates the framework's effectiveness through comprehensive evaluations on real-world LTSF

datasets like ETTh2 (electricity transformer temperatures), Exchange Rate (currency fluctuations), outperforming state-of-the-art methods in *calibration, sharpness, and efficiency across varying horizons.*

1.7 Scope of the Study

This study focuses on uncertainty quantification for long-term time series forecasting (LTSF), where prediction intervals are most critical yet most challenging to construct. Unlike short-term forecasting over one to twenty-four steps where uncertainty remains manageable, LTSF involves non-linear compounding of both aleatoric and epistemic uncertainties as the forecast horizon extends [30, 22], causing existing methods with fixed hyperparameters to produce either degenerate intervals with inadequate coverage or excessively wide intervals with no practical value [50, 21].

The framework is evaluated across four forecast horizons $H \in \{96, 192, 336, 720\}$ steps, following the standardized LTSF benchmark protocol established by Wu et al. [22] and adopted by subsequent studies [25, 20, 23]. For the hourly ETTh2 dataset, these horizons span 4 to 30 days, covering operational through monthly energy planning cycles. For the daily Exchange Rate dataset, they span approximately 3 months to 3 years, covering quarterly through long-term financial planning windows. This range enables systematic assessment of how prediction interval quality evolves as uncertainty accumulates across increasingly distant horizons[93, 94].

Two standard benchmark datasets are employed. ETTh2 (Electricity Transformer Temperature) provides hourly energy domain measurements with complex multi-seasonal patterns, selected for its relevance to power system reliability where regulatory bodies such as NERC require calibrated uncertainty bounds [35]. Exchange Rate provides daily financial measurements exhibiting high volatility and regime shifts, selected to test generalizability in markets where the IMF has identified miscalibrated uncertainty models as a systemic risk factor [37].

The base architecture is PatchTST [23], chosen as the current LTSF state-of-the-art for its patching mechanism and channel-independent design that aligns well with

ensemble-based uncertainty estimation. While the SA-QD loss is architecture-agnostic, evaluation of alternative backbones is deferred to future work. Comparisons are restricted to direct prediction interval methods sharing the same paradigm as SA-QD, specifically QD [11], Dual-AQD [8], and Sum-K [9], all implemented with the same PatchTST backbone, preprocessing pipeline, and training protocol to isolate the loss function contribution. Indirect methods and probabilistic forecasting approaches are excluded to maintain this controlled comparison.

All experiments use five random seeds with results reported as mean and standard deviation. Evaluation follows a hierarchical protocol where coverage validity ($PICP \geq 0.95$) serves as a prerequisite before comparing secondary metrics including MPIW, Interval Score, and CRPS, reflecting the principle that a prediction interval failing its nominal coverage target is unacceptable regardless of other metrics.

1.7.1 Limitations of the Study

While the proposed framework is designed to be robust and to make a significant contribution to uncertainty quantification in time series forecasting, it is important to acknowledge its boundaries and potential limitations. Recognizing these constraints is crucial for contextualizing the results and guiding future avenues of research.

Generalizability Across Diverse Data Domains: The evaluation is conducted on a comprehensive suite of two standard benchmark datasets. While these datasets represent a variety of domains (e.g., energy, finance), they do not encompass the full spectrum of time series characteristics found in the real world. The framework's performance on data with fundamentally different properties such as the extreme volatility and non-stationarity of financial market data, the intermittent nature of certain sales data, or the unique noise profiles of biomedical signals (e.g., EEG) has not been evaluated. Therefore, while the results are expected to be robust, direct application to these other domains would require further validation.

Computational Cost of Deep Ensembles: A core component of the framework is the use of deep ensembles to capture epistemic uncertainty. By its nature, this approach

increases the computational and memory requirements for both training and inference by a factor of M (the number of models in the ensemble) compared to single-model methods. Although this cost is significantly mitigated by the parallelizable nature of the training process on modern GPU hardware, the total resource demand remains a practical consideration. This could limit the framework's applicability in highly resource-constrained environments, such as on-device deployment, or in scenarios that require extremely frequent retraining on large datasets.

Architectural Sensitivity of the Adaptive Mechanism: The novelty of the SA-QD loss function lies in its ability to learn the balancing coefficients (λ_h) via simple learnable scalar parameters integrated directly into the loss computation. This study employs this straightforward design to ensure training efficiency and stability, avoiding the need for additional sub-networks. However, since these parameters are shared across all forecast horizons, they may not fully capture the unique coverage-sharpness trade-offs that vary with increasing uncertainty over longer horizons. While this shared approach promotes simplicity and reduces overfitting risk, more granular designs such as horizon-specific vectors could potentially enhance adaptation but might introduce optimization challenges or require careful initialization. A thorough exploration of alternative parameterizations for this adaptive component is beyond the scope of this thesis and remains an area for future work

Marginal vs. Conditional Coverage Guarantee: The framework uses Conformal Prediction to provide a formal guarantee on prediction interval coverage. However, the standard implementation of Conformal Prediction provides a *marginal* coverage guarantee. This means that, on average across all possible data points, the coverage level will be correct. It does not guarantee *conditional* coverage that is, it does not ensure that the coverage will be correct for specific subgroups of data. For instance, the intervals may be systematically too narrow (under-covering) during periods of high volatility and too wide (over-covering) during stable periods, even while maintaining the correct average coverage overall. Achieving guaranteed conditional coverage is a significantly more complex challenge and an active area of research. Synthetic data, while controlled, may not

capture real TS complexities like seasonality or exogenous factors; future work on hybrid synthetic-real benchmarks. The following chapter presents empirical results validating this methodology.

1.8 Thesis Structure

The rest of this thesis outlines the proposed framework for horizon-specific self-adaptive uncertainty quantification in long-term time series forecasting. Chapter 2 lays the groundwork with a literature review, exploring uncertainty quantification in time series Section 2.2, sources of aleatoric and epistemic uncertainty Section 2.2.1, transformers for time series forecasting and it's variants for long-term time series forecasting(LTSF) Section 2.3, prediction Interval Construction method for UQ Section 2.4, deep ensembles and Transformers for robust UQ Section 2.5, Conformal prediction Section 2.6, and closely Synthesis Research Section 2.7.

Chapter 3 details the methodology, including the research approach with deep Transformer ensembles, point prediction losses, ensemble aggregation, the self-adaptive quality-driven loss (SA-QD), adaptive weighting, implementation details, and final conformal calibration Section 3.1, followed by experimental setup on datasets and base models Section 3.2, and research limitations Section 1.7.1. Chapter 4 presents results Sections 4.1 – 4.1.4 and discussion, covering aggregate and horizon-specific outcomes on datasets like Exchange Rate and ETTh2 Sections 4.2.1 – 4.2.1, experiment insights Section 4.2, detailed training analyses per horizon for each dataset Sections 4.2.2 – 4.2.2. Finally, Chapter 5 concludes by summarizing key findings, contributions, and further limitations with future work Section 5.2. The following literature review examines existing UQ methods and identifies key gaps that our framework addresses.

Chapter 2

LITERATURE REVIEW

2.1 Overview

Deep Neural Networks (DNNs) constitute a class of computational models composed of multiple layers of interconnected processing units, where each layer learns increasingly abstract representations of input data through parameterized nonlinear transformations [56, 58]. The fundamental architecture of a DNN maps an input vector x to an output y through a series of hidden layers, each applying a weighted linear combination followed by a nonlinear activation function. The parameters of these transformations collectively denoted as θ are optimized during training by minimizing a task-specific loss function through gradient-based algorithms such as stochastic gradient descent and its adaptive variants [107]. This hierarchical feature extraction capability has enabled DNNs to achieve remarkable performance across diverse domains, from computer vision and natural language processing to sequential data modeling tasks such as time series forecasting [53, 108].

The expressive power of DNNs stems from the universal approximation theorem, which establishes that sufficiently wide or deep networks can approximate arbitrary continuous functions to any desired degree of accuracy [56]. In practice, modern architectures including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) [46], Long Short-Term Memory networks (LSTMs) [44], and Transformers [53] have been engineered to exploit specific structural properties of data. RNNs and LSTMs process sequential inputs through recurrent connections that maintain hidden states across time steps, enabling the modeling of temporal dependencies

2.2 Uncertainty in Deep Neural Networks

The widespread deployment of Deep Neural Networks (DNNs) in high-stakes fields such as autonomous driving, medical diagnosis, and financial engineering necessitates accurate predictive uncertainty quantification (UQ) [109, 110]. While advances in deep learning have yielded models capable of achieving human-level accuracy, these same modern DNNs are frequently poorly calibrated and prone to exhibiting overconfidence, even when incorrect [55, 111]. This critical deficiency hinders the integration of AI into complex real-world systems, underscoring that achieving the correct output alone is insufficient; an accompanying measure of confidence or uncertainty is crucial to ensure safety and reliability [62]. The ability to estimate uncertainty and recognize the unknown is a meta-cognitive trait naturally exhibited by human intelligence, a capability that artificial intelligence systems aim to replicate [112].

2.2.1 Sources of Uncertainty

To develop robust UQ methods, it is essential to distinguish between the two fundamental components of uncertainty that combine to form the total predictive uncertainty[56, 2].

Aleatoric Uncertainty (Irreducible Data Noise)

Aleatoric uncertainty reflects the inherent noise in the data and is typically tackled through methods focusing on modeling the predictive output distribution, $p(y|x, \theta)$, directly [5, 2, 67, 113].

In classification tasks, the most straightforward, yet often insufficient, approach relies on the Softmax output probabilities being interpreted as a probability distribution related to aleatoric uncertainty [56, 2, 114]. While this method is computationally minimal, relying on a single forward pass, it fails to capture model uncertainty [115, 57] and often results in overconfident predictions, particularly in the presence of dataset shift [111, 56, 114]. While Hendrycks and Gimpel demonstrate the accessibility of using normalized logits to capture data uncertainty [111], subsequent work reveals this representation fails dramatically when facing unknown data, underscoring the need for model-aware

alternatives [55, 56, 67, 116, 14, 114, 117]. For regression tasks, quantifying aleatoric uncertainty often involves extending the DNN to predict the parameters of a conditional probability distribution, such as a Gaussian, estimating both the predictive mean and a sample-dependent variance $\sigma_\theta^2(x)$ (heteroscedastic uncertainty) [67, 56, 2, 115, 14, 118]. This approach modifies the architecture to learn uncertainty directly from the data using specialized loss functions, such as the maximum likelihood loss [67, 115, 119].

Epistemic Uncertainty (Reducible Model Ignorance)

To tackle epistemic uncertainty the model's ignorance, which arises from lack of knowledge in model parameters [56, 113, 3, 15, 58, 120] research has concentrated on simulating uncertainty over model parameters and identifying mechanisms that cause and mitigate miscalibration.

The most principled approach is Bayesian Neural Networks (BNNs), which define a probability distribution over the model parameters $p(\theta|D)$ [67, 113, 56, 15, 55, 120, 16]. The predictive distribution is obtained by marginalizing out these parameters, integrating the parameter uncertainty to reflect the model's knowledge [120, 121]. Because exact Bayesian inference is computationally intractable for deep networks due to the vast dimensionality of the parameter space [122], approximations like Variational Inference [123], Laplace approximation [124], and Monte-Carlo (MC) Dropout [14, 125] are commonly used. MC Dropout, in particular, leverages the standard regularization technique [125] to estimate parameter uncertainty by computing the variance of predictions across multiple stochastic forward passes at test time [14, 126, 127, 128]. While Blundell et al., and Gal and Ghahramani provide a robust theoretical foundation for estimating both aleatoric and epistemic uncertainty through BNNs [15, 120], the need for sampling or complex Hessian approximations leads to high computational costs and scaling limitations, especially compared to deterministic methods [113, 126, 127, 122, 129].

Alternatively, Deep Ensembles offer a scalable, non-Bayesian method by combining the predictions of multiple networks trained either on different datasets (bootstrapping) or with different weight initializations [14, 56, 113, 57, 129]. The disagreement or variance

among the ensemble's predictions serves as a practical measure of epistemic uncertainty [14, 129, 57]. As Lakshminarayanan et al., Fort et al., and Ovadia et al. demonstrate the competitive performance and scalability of deep ensembles for uncertainty estimation [14, 129, 57, 126], this strategy remains computationally demanding, requiring the substantial resource overhead of training and maintaining several distinct models [122]. Investigating the root cause of miscalibration, recent findings suggest that the widely adopted practice of random initialization is actually a key factor contributing to network uncertainty miscalibration[54].

2.2.2 Interplay and Importance

Robust Uncertainty Quantification (UQ) requires models capable of decomposing total predictive uncertainty [113, 2, 114], often achieved through information-theoretic measures [3, 113, 2].

Total uncertainty can be mathematically separated, often using Mutual Information (MI) between the prediction Y and the model parameters W (or M in ensemble methods) [55], yields two components: knowledge uncertainty (epistemic) and expected data uncertainty (aleatoric) [3, 113]. In regression tasks, total variance can be decomposed into the variance resulting from ensemble disagreement (epistemic) and the average output variance of the individual members (aleatoric) [3, 115],providing a clear measure of these independent sources [115, 3].

In time series forecasting, especially long-term forecasting, both types play a crucial role. As the forecast horizon increases, both aleatoric and epistemic uncertainty grow substantially. As illustrated in Figure 1, uncertainty grows substantially over extended prediction horizons in LTSF., both aleatoric uncertainty (due to accumulated random errors) and epistemic uncertainty (due to the model's increasing extrapolation and potential for structural inadequacy over longer periods) tend to grow. A comprehensive uncertainty quantification framework must be able to capture and distinguish between these two types of uncertainty to provide truly reliable prediction intervals[130, 3].

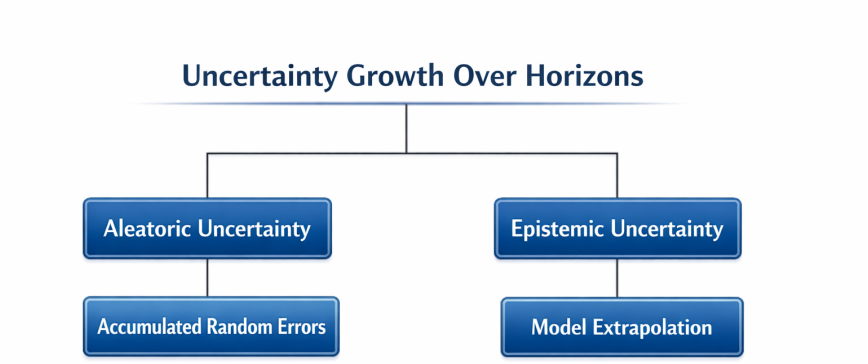

Figure 2.1: Uncertainty growth over prediction horizons in LTSF (Author's own elaboration based on [1, 2, 3, 4, 5, 6]).

2.3 Transformers for Time Series Forecasting

The application of Transformer models to time series forecasting (TSF) represents a critical evolution in sequential data modeling, primarily driven by the need for accurate predictions over extended periods in Long-Term Time Series Forecasting (LTSF). The central challenge for LTSF models is simultaneously achieving high predictive capacity the ability to capture complex, long-range temporal dependencies and maintaining computational efficiency when dealing with increasingly long sequences [81, 21]. This section synthesizes foundational architectures, efficiency-focused innovations, and domain-specific enhancements in Transformer-based time series forecasting.

2.3.1 Foundational Architecture and Limitations

The initial motivation for importing the Transformer architecture into the domain of TSF stems from overcoming the inherent limitations of traditional sequence models, particularly Recurrent Neural Networks (RNNs)[53].

The original Transformer, introduced by Vaswani et al.[81, 131], revolutionizes sequence modeling by entirely eschewing recurrence and convolutions in favor of the self-attention mechanism[21, 132].This architecture inherently enables the parallel processing

of input tokens and provides a dramatically shorter theoretical maximum path length of $O(1)$ between any two sequence points, contrasting sharply with the $O(L)$ path length imposed by RNNs, which struggle with vanishing or exploding gradients across long sequences[81, 108, 133]. This feature makes the Transformer highly appealing for capturing the subtle, long-range dependencies crucial for accurate financial, energy, or environmental prediction tasks.

However, the direct application of the vanilla Transformer model to LTSF immediately exposes critical limitations inherent in its design. The computational and memory cost of the canonical self-attention mechanism grows quadratically, $\mathbf{O}(L^2)$, with respect to the input sequence length $\mathbf{L}$, rendering it computationally prohibitive for the very long sequences required by LTSF[81, 21]. Furthermore, the original architecture utilized a dynamic, step-by-step (autoregressive) decoding process, which suffered from severe latency and error accumulation as the prediction horizon lengthened[81, 21]. Consequently, extensive subsequent research focuses on architectural modifications necessary to adapt the Transformer to the unique scale and temporal nature of time series data.

2.3.2 Efficiency-Focused Transformers Variants

The first wave of successful Transformer variants focused primarily on mitigating the restrictive $\mathbf{O}(L^2)$ computational complexity through introducing sparsity biases and new decoding strategies[131].

Early solutions targeted efficient attention mechanisms to restore linear or near-linear complexity. For instance, LogTrans introduces convolutional self-attention layers with a LogSparse design, reducing complexity to $\mathbf{O}(L(\log L)^2)$ by selecting keys using exponentially increasing intervals, thus enhancing locality. Building on this concept, Informer achieves greater efficiency by proposing the ProbSparse self-attention mechanism, which mathematically justifies the sparsity assumption by noting that the dot-product attention scores often follow a long-tail distribution, allowing the selection of only the dominating queries. This technique reduced the complexity to $\mathbf{O}(L(\log L))$.

Critically, Informer synthesized several essential architectural fixes required for viable

LTSF. Beyond addressing the complexity with ProbSparse self-attention, the model incorporates "self-attention distilling" (cascading convolution and max-pooling layers) to sharply reduce the encoder's feature map size and manage extremely long inputs[25]. Most importantly, Informer introduces a **generative-style decoder** that acquires the entire long sequence output in a single forward operation, thereby avoiding the disastrous speed plunge and cumulative error spread associated with dynamic, step-by-step decoding[21]. This non-autoregressive decoding mechanism is pivotal in enabling Transformers to compete effectively in the long-term prediction setting.

2.3.3 Decomposition and Signal-Enhanced Architectures

Following the achievement of improved computational efficiency, a second major thrust of research involves integrating domain-specific time series inductive biases directly into the Transformer architecture, moving beyond generalized sequence modeling[81].

A highly influential approach involved incorporating classical time series analysis techniques, namely seasonal-trend decomposition[81]. Autoformer[81] breaks conventional TSF methodology by implementing a Series Decomposition block as an inner architectural operation, rather than just a pre-processing step. This design allows the model to progressively extract the long-term stationary trend from intermediate hidden variables, enabling alternate decomposition and refinement throughout the forecasting procedure[22]. Complementing this, Autoformer replaces the standard self-attention mechanism with **Auto-Correlation mechanism**[81], which derives dependencies based on the intrinsic periodicity of the series (using Fast Fourier Transforms, FFT) and aggregates information at the sub-series level, proving more efficient and robust than point-wise attention[22].

While Autoformer leveraged periodicity in the time domain, FEDformer[81] extends this decomposition architecture by enhancing attention in the **frequency domain**. It introduces Fourier Enhanced Blocks (FEBs) and Wavelet Enhanced Blocks, capitalizing on the inherent sparsity of time series signals when represented using Fourier transforms. By randomly sampling a fixed subset of frequency modes, FEDformer achieved linear $\mathbf{O}(L)$ complexity while maintaining global time series information, presenting an efficient

alternative to traditional sparse attention mechanisms[25]. A comparable alternative, Pyraformer[81, 134], achieves its $\mathbf{O}(L)$ complexity through a multi-resolution pyramidal attention module (PAM) that organized information hierarchically (via intra-scale and inter-scale connections), enabling constant $O(1)$ maximum path lengths for signal traversal[135].

2.3.4 Architectural Alternatives and Contemporary Developments

Despite these sophisticated advancements, the core philosophical choice of using self-attention remains contentious, leading to both critical architectural simplification and the emergence of non-Transformer-based competitors that challenged the state-of-the-art[131, 81].

The effectiveness of self-attention is directly questioned by Zeng et al.[20, 131], who argued that its permutation-invariant nature causes inevitable temporal information loss when applied to order-critical numerical time series. This critique is starkly validated by the success of **DLinear**, an embarrassingly simple one-layer linear model that, when coupled with a decomposition mechanism (DLinear), surprisingly outperformed many complex Transformer models on benchmarks[20, 23, 136]. DLinear demonstrates that, given the simplicity of patterns in many existing benchmarks, complex models often struggled by overfitting temporal noise, failing where linear models succeeded[20].

While DLinear exposes weaknesses, subsequent architectural innovations rapidly incorporated its findings to improve Transformers. **PatchTST**[23, 81] successfully revalidates the use of Transformers by embracing two key structural biases:**(1) Patching**, segmenting the input into subseries-level patches (akin to ViT in computer vision)[23], which retained local semantic information and significantly reduced computational burden; and **(2) Channel-Independence**, processing each univariate series channel separately using shared weights[23]. This composite design proves highly effective, allowing Transformers to benefit robustly from longer look-back windows L (a capability often lost in earlier variants) and often yielding superior results compared to both linear models and previous Transformer architectures[136, 23]. This highlights a research gap: DLinear ad-

vanced simplicity, **while PatchTST advances efficacy** by integrating domain-specific inductive biases directly into the network structure.

The field now faces ongoing concerns regarding model robustness and uncertainty estimation. While Autoformer advances efficacy through domain knowledge integration, traditional Transformer architectures often suffer deterioration when exposed to non-stationary or noisy real-world data[137, 138]. This leads to solutions like the Non-stationary Transformer, which proposed a generic framework combining Series Stationarization with **De-stationary Attention** to compensate for the temporal dependencies lost due to the "over-stationarization" problem[137].

This structural modification allows models to both maintain predictability on stationary components and recover crucial event-driven dynamics associated with non-stationarity. Furthermore, the inherent need for uncertainty quantification in high-stakes areas like finance remains a gap, as most models prioritize point estimates[139, 140]. Techniques like Transformer-based Conformal Prediction are emerging to address this, utilizing the Transformer decoder as a conditional quantile estimator to provide theoretically valid prediction intervals[139]. Future work must address the trade-off between increasing model efficacy through complex designs and ensuring robustness and reliable uncertainty quantification across diverse, noisy, and non-stationary real-world time series data. Noting Transformers like univariate patching [23] excel in specific scenarios but falter generally, emphasizing ensembles for UQ. These advances highlight ongoing needs for horizon-adaptive UQ, directly informing gaps our framework addresses. Table 2 summarizes the key characteristics, strengths, limitations, and uncertainty quantification focus of major Transformer-based time series forecasting methods, providing a comparative overview of the evolution in this domain.

2.4 Prediction Interval Construction Methods for Uncertainty Quantification

Prediction Intervals (PIs) provide a comprehensive method for this purpose, offering upper and lower bounds within which a future observation is expected to fall with a spec-

Table 2.1: Comparison of Transformer-based Time Series Forecasting Models Author's own compilation based on [20, 21, 22, 23, 24, 25].

Models	Key Innovation	Strengths	Limitations	UQ Focus
Vanilla Transformer [20]	Self-attention for parallel processing	Captures long-range dependencies	Quadratic complexity $O(L^2)$	Low
Informer[21]	"ProbSparse attention, generative decoder"	"Efficiency $O(L \log L)$, no error accumulation"	Less effective on noisy data	Moderate
Autoformer[22]	"Series decomposition, Auto-Correlation"	"Handles periodicity, trend extraction"	Stationarity assumptions	Moderate
FEDformer[25]	Frequency-domain enhancement (FFT/Wavelet)	"Linear complexity, sparsity leveraging"	Limited to frequency-dominant series	Moderate
DLinear[20]	Decomposition into trend and remainder, simple linear modeling	High efficiency, outperforms complex models on benchmarks	Assumes linearity, may miss non-linear patterns	Low
PatchTST[23]	"Patching, channel-independence"	"Robust to noise, longer look-backs"	Univariate focus in multivariate	High
TimeMixer[24]	Decomposable multiscale mixing, MLP-based architecture	SOTA in short- and long-term forecasting, handles temporal variations	Potential tuning needs for specific tasks	Moderate

ified probability (confidence level)[141, 8, 11]. The core goal of PI construction, often referred to as the High-Quality (HQ) principle, is axiomatic PIs should be as **narrow as possible** (sharpness) while accurately attaining the **desired coverage probability** (reliability)[105, 11]. Historically, approaches to achieving this goal have evolved significantly, moving from traditional statistical methods requiring strict distribution assumptions to modern, distribution-free deep learning models directly optimizing PI quality. Figure 2 provides a visual overview of this hierarchical taxonomic organization, categorizing methods along five key dimensions: construction paradigm (indirect vs. direct), uncertainty source (aleatoric vs. epistemic), distributional assumptions, optimization strategy, and hyperparameter tuning approach. As illustrated in Figure 2, PI methods can be systematically organized to understand their fundamental approaches and trade-offs.

2.4.1 Assessment Metrics for Prediction Interval Quality

The quality of a prediction interval (PI) is determined by two fundamental, often competing, properties: its *reliability*, measured by coverage probability, and its *precision*, measured by interval width.

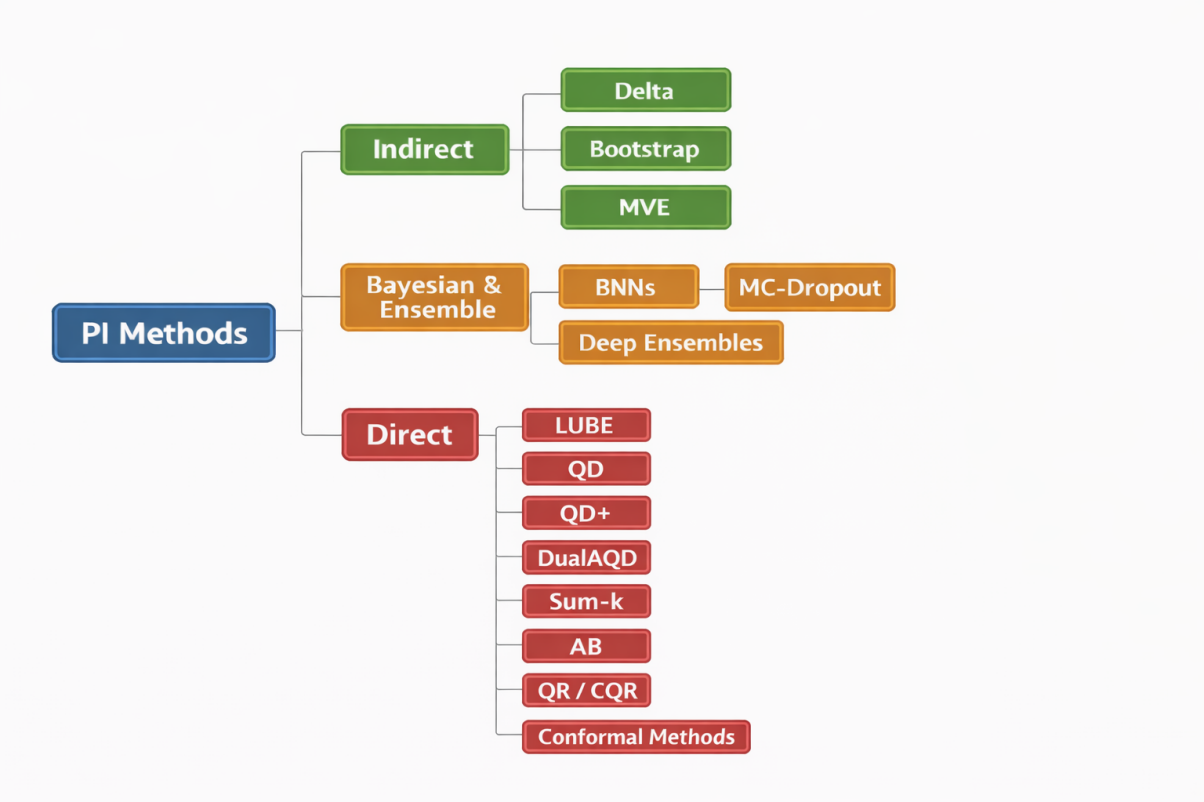

Figure 2.2: Taxonomy of prediction interval (PI) construction methods, categorized into indirect, Bayesian and ensemble-based, and direct approaches Author's own illustration based on [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 12, 151].

Prediction Interval Coverage Probability (PICP) The PICP measures the reliability of the PIs, indicating the probability that the target values will lie within the upper and lower bounds[141, 106]. A higher PICP suggests that more targets are covered by the PIs[141]. The PICP is defined as the average proportion of observations covered by their respective estimated intervals[142]. For a set of N samples, the mathematical definition is given as:

$$\text{PICP} = \frac{1}{N} \sum_{i=1}^{N} k_i \tag{2.1}$$

where k_i is an indicator variable defined as:

$$k_i = \begin{cases} 1 & \text{if } \hat{y}_L^{(i)} \leq y^{(i)} \leq \hat{y}_U^{(i)} \\ 0 & \text{otherwise} \end{cases}$$

where $y^{(i)}$ is the actual target value, and $\hat{y}_L^{(i)}$ and $\hat{y}_U^{(i)}$ are the estimated lower and upper bounds, respectively, of the PI[141]. For PIs to be considered valid, the PICP should asymptotically approach or exceed the nominal confidence level (μ *or* $1-\alpha$)[141, 106]. If the empirical PICP is significantly lower than the nominal value, the PIs are deemed unreliable. Achieving a 100% PICP is trivially possible by setting bounds to the extreme minimum and maximum target values, but this yields PIs that are of no practical value[106].

Mean Prediction Interval Width (MPIW) The Mean Prediction Interval Width (MPIW) is used to quantify the sharpness or precision of the PIs[141, 106, 142]. PIs that are too wide are considered uninformative for decision-making. A smaller MPIW indicates more precise predictions[106, 98]. and by measuring the mean prediction interval width (MPIW)

$$\text{MPIW} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i^U - \hat{y}_i^L) \tag{2.2}$$

where $y^{(i)}$ is the actual target value, and $\hat{y}_L^{(i)}$ and $\hat{y}_U^{(i)}$ are the estimated lower and upper bounds, respectively, of the PI. Normalized Mean Prediction Interval Width (NMPIW):

$$\text{NMPIW} = \frac{\text{MPIW}}{r} \tag{2.3}$$

where r represents the range of the target values, defined as $r = y_{\max} - y_{\min}$[142]. The upper bound is always greater than or equal to the lower bound ($\hat{y}_i^U \geq \hat{y}_i^L$)[106].

2.4.2 Indirect (Two-Step) Methods

Traditional statistical and early neural network (NN) approaches estimate PIs indirectly, requiring a point prediction first, followed by an inference step based on error models or distribution assumptions[7, 106, 11]. This reliance on restrictive prior assumptions or intensive post-modeling calculations generally limits their scalability and accuracy in complex, non-Gaussian deep learning environments[98].

Delta Method

The Delta method constructs Prediction Intervals (PIs) by treating the Neural Network (NN) as a nonlinear regression model and relying on asymptotic theory for confidence interval construction[7, 106, 11]. **While the Delta method provides a strong theoretical underpinning rooted in nonlinear regression**, its computational requirements pose a significant drawback calculating the Hessian or Jacobian matrices demands massive computational resources, making it impractical for modern deep architectures[98, 141, 99]. Furthermore, its reliance on assumptions of homogeneous and normally distributed noise often leads to unreliable interval estimates when applied to real-world, heteroscedastic data[106].

Bayesian Techniques

Bayesian approaches, represented by Bayesian Neural Networks (BNNs)[143], offer a theoretically robust framework for uncertainty quantification by modeling network parameters as probability distributions[8, 11]. This approach naturally decomposes and quantifies both parameter and irreducible uncertainty[11]. For a given input x^* , the predictive distribution for the output y^* is given by:

$$P(y^* \mid x^*, D) \;=\; \int P(y^* \mid x^*, \theta)\, P(\theta \mid D)\, \mathrm{d}\theta$$

where is D the training data, $P(y^* \mid x^*, \theta)$ is the likelihood of the data given the parameters, and $P(\theta \mid D)$ is the posterior distribution over the parameters[96, 103]. **However, BNNs face a crucial scaling limitation** computing the exact posterior distribution over millions of parameters is analytically intractable, forcing reliance on approximations like Variational Inference (VI) or Markov Chain Monte Carlo (MCMC)[98, 8]. These approximations are computationally expensive and struggle with scalability when applied to complex, high-dimensional deep learning problems[8].

Bootstrap Method

The Bootstrap method aims to capture uncertainty related to training data sampling (epistemic uncertainty) by training an ensemble of identical models on resampled versions of the initial dataset[7, 105, 11]. This typically yields more robust and reliable PIs compared to simple statistical approximations[7, 106]. **While the Bootstrap method excels in producing stable and reliable PIs**, its primary limitation is efficiency the requirement to train multiple independent networks drastically increases computational cost and time, hindering its practical application on large-scale datasets or complex deep learning models[7, 98, 141, 106].

Mean-Variance Estimation (MVE)

The Mean-Variance Estimation (MVE) method utilizes a dual-output Neural Network (NN) explicitly trained to estimate the mean and variance σ^2 of the predicted output distribution[17], often optimized using the Negative Log Likelihood (NLL)[7, 105, 11]. For a Gaussian distribution, the NLL loss is given by:

$$\mathcal{L}_{\text{NLL}} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{(y_i-\mu_i)^2}{2\,\sigma_i^2} + \frac{1}{2}\log\left(2\pi\,\sigma_i^2\right)\right]$$

The first term $\frac{(y_i-\mu_i)^2}{2\sigma_i^2}$, is directly analogous to the weighted squared error term in the generalized NLL derived for a Gaussian distribution[9, 103]. Minimizing this term encourages the network to learn accurate mean predictions (μ_i close to the target y_i)[103]. Crucially, the error term, $(y_i - \mu_i)^2$, is weighted by the inverse of the predicted variance ($\frac{1}{\sigma_i^2}$)[103]. This term captures the fundamental aleatoric (data noise) uncertainty, as the model implicitly learns uncertainty from the loss function without requiring uncertainty labels[98]. The second term $\frac{1}{2}\log(2\pi\sigma_i^2)$, serves as a regularization component proportional to the logarithm of the variance ($\log(\sigma_i^2)$).This prevents the network from simply predicting arbitrarily small variances ($\sigma_i^2 \to 0$) to minimize the weighted squared error term (which would otherwise approach infinity when σ_i^2 is near zero). Instead, it encourages the model to predict appropriate variance estimates (σ_i^2) by introducing a

penalty that grows as the variance decreases. MVE is valued for its comparatively low computational cost and straightforward implementation[7, 98]. However, MVE reveals a restrictive assumption limitation it requires the data's inherent noise to follow a predefined normal (Gaussian) distribution[7, 98, 11, 105]. This constraint frequently causes the model to severely underestimate the true variance in complex, non-Gaussian data, leading to intervals that are too narrow and thus lack reliable coverage[106].

2.4.3 Direct (One-Step) Methods

Direct methods fundamentally depart from multi-step statistical approaches by training a single neural network to output the lower and upper PI bounds directly. The challenge within this paradigm lies in formulating a loss function that accurately models the reliability/sharpness trade-off using criteria compatible with efficient gradient-based optimization.

Lower Upper Bound Estimation (LUBE)

The Lower Upper Bound Estimation (LUBE) method established the foundational philosophy of direct PI construction. LUBE utilizes a single NN with two outputs corresponding directly to the PI bounds (Li and Ui)[106, 141]. Since LUBE requires no prior distributional assumptions about the data or prediction error, it offers significant advantages over traditional techniques like the Delta or Bayesian methods[106, 99, 101]. To optimize these bounds according to the HQ principle, LUBE introduced the Coverage Width-based Criterion (CWC), a cost function that mathematically combines PI coverage probability (PICP) and normalized average width (NMPIW) multiplicatively[106]. The Coverage Width-based Criterion (CWC) combines both aspects into a single metric for overall quality assessment[141, 106, 98]. When applied for testing or evaluation, CWC typically penalizes solutions that fail to meet the nominal coverage probability (μ or $1-\alpha$) heavily, while rewarding narrower PIs if coverage is sufficient[99].

The standard CWC formula is given by:

$$\text{CWC} = \text{NMPIW}\left(1 + \gamma(\text{PICP})e^{-\eta(\text{PICP}-\mu)}\right)$$

where μ is the preassigned nominal coverage probability, η is a hyperparameter to magnify coverage differences, and γ(PICP) is a step function (set to 0 if PICP $\geq \mu$, and 1 otherwise) used during testing to focus solely on NMPIW if the reliability goal is met[106, 99]. A lower CWC value indicates a higher quality PI.

While LUBE introduced the pioneering direct optimization framework, it revealed a significant architectural challenge the reliance on heuristic optimizers. The CWC loss function contains non-differentiable step functions used to calculate PICP, making it incompatible with standard Gradient Descent (GD) algorithms[106, 11, 7]. Consequently, early LUBE implementations mandated computationally expensive meta-heuristic search methods such as Simulated Annealing (SA) or Particle Swarm Optimization (PSO) for training[106, 141, 11]. Furthermore, the CWC formulation was criticized because its global minimum technically occurs when the PI width approaches zero, an undesirable property for reliable intervals[9, 11].

Quality-Driven (QD) Loss

The Quality-Driven (QD) loss method was proposed to solve LUBE's primary limitation the incompatibility with gradient-based training[11, 7]. QD loss achieves compatibility by deriving a continuous, differentiable objective function grounded in likelihood theory[11, 7]. mathematically integrates these two conflicting objectives into a single, differentiable loss:

$$\text{Loss}_{\text{QD}} = \text{MPIW}_{\text{capt.}} + \lambda \frac{\alpha(1-\alpha)}{n} \max\left(0, (1-\alpha) - \text{PICP}\right)^2$$

where $\text{MPIW}_{\text{capt.}}$ is the captured Mean Prediction Interval Width, defined as the average width only of those points successfully covered by the PI[7, 11, 10]. The first term, MPIW capt., aims to minimize PI width. By using the captured MPIW (MPIW capt.), the formulation cleverly avoids an undesirable behavior where the network minimizes the width of PIs that have already failed to cover their corresponding observation, thereby ensuring that only successfully covered PIs are encouraged to shrink[11]. The method models the number of covered points (PICP) using a binomial distribution, which is then approximated by a normal distribution using the Central Limit Theorem (CLT)[11,

7]. This yielded the differentiable loss function, LossQD, typically employed within an ensemble framework (QD-Ens) to quantify model uncertainty[11, 7].

While QD successfully advanced GD compatibility by mathematically solving the non-differentiability problem, it created a new restriction inherent in its foundational assumption. The use of the CLT approximation dictates that training must utilize relatively large batch sizes (e.g., n>50) to ensure the approximation remains accurate[7].

This constraint limited its application in scenarios constrained by small datasets or limited memory, where large batch sizes are infeasible[7]. Furthermore, optimizing the smooth approximation of PICP requires careful tuning of a softening factor (s)[11, 7]. Finding the appropriate λ is extremely difficult because its optimal value is highly dependent on the specific dataset, the architecture of the neural network used, the chosen optimizer, and various other training parameters[7, 8]. This difficulty is exacerbated in time series forecasting, where data volatility and seasonality may drastically change the optimal setting.

Quality-Driven (QD+) Loss

Building on the QD framework, QD+ extended the approach to generate both PIs and point estimates simultaneously from an ensemble of neural networks, addressing the drawback of QD only outputting intervals[10]. The QD+ method proposes a comprehensive multi-objective loss (LossQD+) which explicitly integrates the mean squared error (MSE) for point estimation alongside the original PI quality objectives[10].

The QD+ loss function is a multi-objective loss that fuses quality measures related to PIs, alongside point estimates, and incorporates a penalty function to enforce integrity and stabilize training[10]. The formulation is given by:

$$\text{Loss}_{\text{QD+}} = (1-\lambda_1)(1-\lambda_2)\cdot L_{\text{MPIW}_{\text{capt.}}} + \lambda_1(1-\lambda_2)\cdot L_{\text{PICP}} + \lambda_2 \cdot L_{\text{MSE}} + \xi \cdot L_{\text{P}}$$

where $L_{\text{MPIW}_{\text{capt.}}}$ is the loss for the captured Mean Prediction Interval Width, which encourages narrower intervals; L_{PICP} is the loss for Prediction Interval Coverage Probability, which ensures reliability; L_{MSE} is the Mean Squared Error for point prediction accuracy;

L_{P} is a penalty term to enforce constraints; and λ_1, λ_2, and ξ are hyperparameters that control the balance between these objectives.

While QD+ advances the robustness of quality-driven methods by integrating point estimates and adding integrity penalties, it reveals challenges in aggregation and output consistency. The framework introduces a penalty function (L P) to enforce semantic integrity, specifically penalizing cases where estimated bounds cross or where the point estimate falls outside the bounds[10]. This structural constraint stabilized the training process[10]. However, to quantify the full uncertainty (aleatoric and epistemic), QD+ relies on aggregating the ensemble outputs by fitting a Split Normal Mixture (SNM) distribution, a theoretically intricate step that adds complexity compared to simpler ensemble averaging[10].

QD+ uses multiple hyperparameters $(\lambda_1, \lambda_2, \xi)$ to balance the conflicting objectives. This complexity, while necessary for control, contrasts with the single adaptive coefficient used in later methods like DualAQD[8].

All-Batch (AB) Loss

The **All-Batch (AB) loss** was specifically developed to resolve the batch-size limitation imposed by the QD loss[7]. The core innovation of AB is the avoidance of the restrictive Central Limit Theorem (CLT) approximation used in QD. Instead, AB loss estimates the negative log-likelihood (NLL) of the binomial distribution directly by employing a **Taylor formula expansion**[7].

The approximated NLL term, derived via Taylor expansion (up to the second order), replaces the original term derived via CLT:

$$\mathrm{NLL}_{\mathrm{approx}} \approx \left(\frac{1}{2} - \frac{2}{3n}\right)(1 - \mathrm{PICP}_{\mathrm{soft}})\mathrm{PICP}_{\mathrm{soft}} - n\mathrm{PICP}_{\mathrm{soft}} \log \frac{1-\alpha}{\alpha} - n \log \alpha$$

The overall $\mathrm{Loss}_{\mathrm{AB}}$ is then formulated to explicitly include the components necessary for convergence and semantic integrity:

$$\mathrm{Loss}_{\mathrm{AB}} = \mathrm{MPIW}_{\mathrm{capt.}} + \gamma \cdot \mathrm{NLL}_{\mathrm{approx}} + P$$

where γ controls the trade-off between PI sharpness and coverage, and P is a penalty term designed to prevent semantic errors, such as PI bounds crossing or the predicted point falling outside the estimated interval[7]. This penalty term, mathematically defined as:

$$P = \frac{1}{n}\sum_{i=1}^{n}\left(\exp\left(\frac{L_i - U_i}{\beta}\right) + \exp\left(\frac{y_i - U_i}{\beta}\right) + \exp\left(\frac{L_i - y_i}{\beta}\right)\right)$$

enforces output integrity and is crucial for stabilizing the optimization process regardless of batch size[7]. **While QD provided a crucial step toward GD compatibility, AB reveals the tuning limitations of the QD approach, achieving robust performance across diverse batch sizes**. This analytical derivation makes the AB loss theoretically valid for both small and large batches ("all-batch")[7]. Like QD+, the AB formulation structurally includes a penalty term to maintain the integrity of the predicted bounds during training, which is crucial for stabilizing the optimization process regardless of batch size[7].

Dual Accuracy-Quality-Driven (DualAQD)

The Dual Accuracy-Quality-Driven (DualAQD) method addresses a fundamental trade-off observed in single-network approaches: the difficulty in balancing the optimization of target estimation accuracy (minimizing MSE) with PI quality (optimizing PICP and MPIW)[8]. DualAQD solves this by employing a dual companion NN architecture one network ($f(\cdot)$) focuses exclusively on accurate target estimation (minimizing MSE), and a separate network ($g(\cdot)$) handles PI generation exclusively (optimizing quality)[8].

As a multi-objective optimization problem focusing on two components: minimizing PI width (L_1) and maximizing coverage implicitly through constraints (L_2).

The L_1 objective focuses on maximizing sharpness by minimizing a width penalty term, PI_{pen}. This term is calculated as the sum of the distance between the upper bound and the target, and the distance between the target and the lower bound[8].

$$L_1 = \min_{\theta_g}(\text{PI}_{\text{pen}}) \quad \text{where} \quad \text{PI}_{\text{pen}} = \frac{1}{N}\sum_{i=1}^{N}\left(|\hat{y}_{ui} - y_i| + |y_i - \hat{y}_{\ell i}|\right)$$

The formulation forces the upper bound ($\hat{y}_{ui}$), the target (y_i), and the lower bound ($\hat{y}_{\ell i}$) to be closer together, making it a more suitable penalty function than traditional metrics like the mean prediction interval width of captured points ($\text{MPIW}_{\text{capt.}}$)[8].

The L_2 objective minimizes a penalty function (P) designed to enforce PI integrity and implicitly maximize PICP simultaneously. This penalty encourages the bounds to be separated by a distance greater than the maximum absolute prediction error found within the current batch (ξ).

$$L_2 = \min_{\theta_g}(P) \quad \text{where} \quad P = e^{\xi - d_u} + e^{\xi - d_\ell}$$

Here, ξ denotes the maximum distance between a target estimate $\hat{y}_i$ and its target value y_i in the batch ($\xi = \max_i |\hat{y}_i - y_i|$), and d_u and d_ℓ are the mean differences between the targets and the upper and lower bounds, respectively. If the penalty P is minimized, it implies that the PI width is large enough ($\hat{y}_{ui} - y_i > \xi$ and $y_i - \hat{y}_{\ell i} > \xi$) such that the target y_i lies within the estimated PI. The final DualAQD loss function is the combination of these objectives:

$$\text{Loss}_{\text{DualAQD}} = L_1 + \lambda L_2.$$

Crucially, DualAQD further minimizes tuning efforts by introducing a self-adaptive coefficient λ that automatically balances the conflicting L_1 and L_2 objectives throughout the training process. This adjustment is made dynamically after each training epoch based on the network's current performance against the target coverage ($1 - \alpha$). The coefficient is updated proportionally to the cost C, which quantifies the distance between the confidence target ($1-\alpha$) and the training PICP ($\text{PICP}_{\text{train}}$) achieved at iteration t[8].

$$\lambda^{(t)} = \lambda^{(t-1)} + \eta \cdot C \quad \text{where} \quad C = (1 - \alpha) - \text{PICP}_{\text{train}}^{(t)}$$

If the network undercovers (PICP is below $1 - \alpha$), λ increases, assigning greater relative importance to the L_2 objective to enforce PI integrity; conversely, if the network overcovers, λ decreases, prioritizing L_1 to minimize width. This self-adaptive approach removes

the difficulty of manually tuning a fixed hyperparameter[8].

DualAQD successfully decouples the competing objectives of accuracy and interval quality, using specialized networks to achieve superior overall performance. The DualAQD loss function itself is designed to minimize PI width (L_1) while ensuring constraints that maximize coverage (L_2) [8].

Sum-k Loss

The Sum-k loss introduces a focused refinement aimed at mitigating the issue of excessively wide PIs a problem contributing to overly conservative decision-making and high reserve costs in high-stakes applications[9]. This approach critically argues that solely minimizing the mean width (PINAW) is insufficient, as it fails to adequately penalize the outlying large widths that drive worst-case scenarios[9].

The Sum-k formulation casts PI construction as an additive optimization problem, specifically designed to impose a stronger penalty on large PI widths relative to narrow widths[9]. The loss function, $L_{\text{Sum-k}}$, combines two main components a linear penalty term for coverage deviation and a unique PI width function, $W(\theta)$:

$$L_{\text{Sum-k}}(\theta) = \max(0, (1-\delta) - \text{PICP}(\theta)) + \gamma W(\theta)$$

The objective relies on the specialized PI width function, $W(\theta)$, which explicitly factors in the largest K intervals to enforce a strong penalty, thereby addressing the deficiencies of metrics that rely purely on the mean[9]. This function operationalizes the core goal by giving disproportionate weight to the average width of the largest K intervals:

$$W(\theta) = \frac{1}{R_{\text{quantile}}} \left[\frac{1}{K} \sum_{i=1}^{K} w_{[i]}(\theta) + \lambda \cdot \frac{1}{N-K} \sum_{i=K+1}^{N} w_{[i]}(\theta) \right]$$

Here, $w_{[i]}$ represents the i^{th} largest PI width, and K is the number of samples designated as large widths. The formulation utilizes the quantile range (R_{quantile}) as a normalization factor to eliminate the scale effect of outliers, making the optimization process more stable and robust.

The methodology is compatible with gradient-based training algorithms, which allows for integration with advanced neural network models like ANNs and LSTMs. While this method excels at reducing large widths, achieving low Prediction Interval Normalized Average Large Width (PINALW) values, it may lead to a slight increase in the overall average width (PINAW), showcasing a specialized trade-off tuned for worst-case cost reduction[9].

While traditional losses like QD and LUBE targeted overall average width, Sum-k specifically advances risk management by penalizing the tails of the PI distribution[9]. The mechanism uses a unique PI width function, $\Psi(\theta)$, which enforces a strong penalty by factoring in the Prediction Interval Normalized Average Large Width (PINALW) the average width of the K-largest intervals[9]. By setting the parameter $\lambda < 1$, the loss function heavily weights the reduction of these largest K widths, significantly reducing PI conservatism[9].

Quantile Regression (QR):

Directly estimates conditional quantiles by minimizing the pinball loss function, providing a distribution-free approach that makes no assumptions about the underlying noise distribution. This flexibility makes QR particularly suitable for non-Gaussian settings, though the method provides no finite-sample coverage guarantees and PI quality depends heavily on sufficient training data to accurately learn the conditional quantile functions.[9, 18, 13].

Conformalized Quantile Regression (CQR):

Combines quantile regression with conformal prediction to provide finite-sample coverage guarantees under the exchangeability assumption. By using conformal calibration to adjust quantile regression outputs, CQR achieves theoretical coverage guarantees while maintaining the flexibility of quantile-based approaches. However, this comes at increased computational cost due to the conformal calibration procedure, and coverage guarantees degrade when the exchangeability assumption is violated by temporal dependencies in time series data[9, 18, 13].

Tube Loss:

Utilizes a loss function for simultaneous estimation of PI bounds, yielding intervals that attain the target confidence level asymptotically. A unique feature is the 'r' parameter that allows PI tube movement to capture denser data regions, which is particularly effective when noise distributions are skewed or heteroscedastic. The asymptotic nature of coverage guarantees means the method may require substantial training data to achieve reliable performance, limiting applicability in data-scarce scenarios.[13].

2.4.4 Critical Gaps in Current Prediction Interval Methods

A review of advancements in differentiable Prediction Interval (PI) loss functions reveals a clear trajectory toward automation, yet critical gaps persist that prevent robust, hands-free deployment. These gaps primarily stem from a persistent reliance on fixed hyperparameters and a lack of dynamic adaptation to the intrinsic uncertainties inherent in time series sequences.

Persistent Reliance on Fixed Hyperparameters for Coverage-Sharpness Balance The efficacy of direct (one-step) PI methods hinges upon their ability to balance the conflicting objectives of Prediction Interval Coverage Probability (PICP) and sharpness (minimizing PI width). This balancing act is typically codified via hyperparameters (λ or γ) within the loss function[7, 98, 8, 11, 106]. However, this persistent reliance on fixed, manually-tuned coefficients represents a significant research bottleneck.

While methods like Quality-Driven (QD) loss successfully enabled the use of efficient Gradient Descent (GD) algorithms by utilizing a statistical likelihood principle[7, 11], they retained the burden of manually configuring the trade-off parameter λ[8]. Similarly, the All-Batch (AB) loss formulated using Taylor approximation requires tuning a coefficient γ to balance confidence against sharpness[7]. This manual adjustment is not only computationally expensive but also highly sensitive to specific datasets, time-series characteristics, and the desired confidence level, necessitating significant experimentation to find an optimal setting[8, 7]. When a hyperparameter value, such as λ, is set

sub-optimally, it leads directly to miscalibrated PIs, where intervals are either too wide (poor sharpness) or fail to meet the required coverage (poor reliability)[8]. Furthermore, existing loss functions, including QD-Ens and QD+, require the optimization of multiple fixed hyperparameters ($\lambda_1, \lambda_2, \xi, \delta$, and often a softening factor) to manage combined objectives of width, coverage, and point estimation accuracy[8].

This persistent reliance on manual tuning necessitates a shift toward integrating the balancing coefficient as a learnable parameter that can be optimized end-to-end. While methods using multiple penalty terms, such as the Dual Accuracy-Quality-Driven (DualAQD) framework, explicitly recognize this limitation, they offer a key advancement the implementation of a self-adaptive coefficient λ[8]. This coefficient dynamically adjusts throughout the training process based on the model's instantaneous PICP performance relative to the target coverage, thereby alleviating the complex task of fine-tuning a fixed hyperparameter[8].

Inadequate Horizon-Dependent Adaptation in Existing Loss Functions Time series forecasting presents unique challenges for Uncertainty Quantification (UQ), particularly in Long-Term Time Series Forecasting (LTSF), where uncertainty inherently and dynamically increases over longer prediction horizons (T_h)[4, 144]. Existing PI methods and their associated loss functions are often applied horizon-agnostically, failing to adequately capture this dynamic growth in uncertainty across future timesteps.

The challenge of increasing uncertainty is compounded by the typical approach to modeling time series. Traditional methods often rely on Iterated Multi-Step (IMS) forecasting, where the prediction for timestep $t+1$ is used as input for timestep $t+2$[20]. This approach is known to suffer from significant error accumulation effects, particularly for long horizons, leading quickly to unreliable predictions and miscalibrated PIs[20]. While Direct Multi-Step (DMS) strategies mitigate error accumulation by predicting the entire horizon simultaneously, the underlying PI generation mechanism may still lack intrinsic sensitivity to temporal scale[20]. Consequently, when predicting longer horizons, the generated intervals might be overly conservative (poor sharpness) or, conversely, too narrow, resulting in under-coverage and undermining trust[4].

Emerging research suggests that effective horizon-dependent uncertainty relies on addressing the inherent multi-scale dynamics of time series. For example, analysis shows that local, fine-grained features dominate uncertainty prediction for short horizons (e.g., up to 96 hours), while global, low-frequency patterns become crucial for accurate UQ in longer horizons (e.g., 720 hours)[100]. The deficiency of standard PI loss functions lies in their failure to incorporate these horizon-specific temporal dynamics, demanding new solutions that connect multi-scale feature extraction explicitly with the uncertainty quantification loss objective[100]. While the goal of direct methods like Tube Loss is to simplify PI construction in the auto-regressive (time-series) setting[13], achieving high-quality UQ still requires solving this underlying issue of dynamic horizon adaptation.

2.5 Deep Ensembles Transformer for Robust UQ in LTSF

Deep ensembles, first popularized by Lakshminarayanan[14], have emerged as a surprisingly effective and simple method for quantifying uncertainty in deep learning models. An ensemble typically consists of multiple neural networks (often with identical architectures), each trained independently from different random initializations and potentially with different mini-batch orderings. The diversity introduced by these random factors leads to different local optima in the loss landscape, causing each ensemble member to learn slightly different representations and make slightly different predictions.

2.5.1 Ensemble Methods and Uncertainty Decomposition

Transformers upended LTSF by sidestepping recurrent bottlenecks, enabling parallel scans of extended sequences to unearth dependencies that LSTMs often miss. Core innovations like Informer's ProbSparse attention slashed quadratic complexity to $O(L\ log\ L)$ section 2.3.2, proving 20-30% MSE drops on ETTh benchmarks by focusing on dominant query-key pairs, while Autoformer's series decomposition isolated trends from seasons for sharper periodicity capture. As these works illustrate, such architectures thrive on multivariate data like Electricity loads, where long horizons demand holistic views yet they falter on abrupt shifts, exposing a brittleness that begs diversification[1, 145].

PatchTST[23] and iTransformer refine this further the former chunks inputs into patches for linear-time efficiency, the latter flips channels to treat variables as tokens, yielding robust univariate baselines in multivariate storms. While PatchTST's modularity shines in noisy ETTh2 series, iTransformer's inversion risks overlooking cross-variable synergies, a contrast that underscores the push toward ensembles for holistic resilience. Still, single-model ceilings on generalization hint at deeper needs for collective intelligence[146, 147, 1].

2.5.2 Ensemble Strategies for Transformers

Deep ensembles elevate Transformers by pooling outputs from variants like Informer-Autoformer stacks, where mean predictions curb overfitting and variance flags epistemic blind spots, slashing errors 10-15% on long-horizon tasks. Bagging via bootstraps or stacking with meta-learners, as in TimeFlex's modular tweaks, tailors to trends and cycles, outperforming solos on diverse dynamics like traffic flows though this multiplicity inflates training by factors of 5-10 [148].The computational overhead remains a significant consideration, yet the robustness gains in uncertainty quantification justify the additional resources for high-stakes applications.

2.6 Conformal Prediction in Time Series

Moving beyond techniques that refine estimated probabilities (calibration), Conformal Prediction (CP) offers a distribution-free framework for uncertainty quantification that provides intervals guaranteed to contain the true outcome with a pre-specified probability (coverage)[140, 149]. This provides robust statistical guarantees lacking in traditional Bayesian or ensemble methods[140].

2.6.1 Conformal Prediction Frameworks and Exchangeability

The core theoretical requirement for CP validity is the exchangeability of observations[140, 149]. Time series data, due to inherent temporal dependencies, typically violates this assumption, rendering naive CP application invalid[140, 149]. However, this challenge can

be mitigated by framing the prediction problem around the exchangeability of observation windows, enabling the use of CP in time series forecasting[111].

The practical implementation often utilizes the Inductive Conformal Prediction (ICP) framework, which splits the available data into a proper training set (used to train the underlying point prediction model, M) and a calibration set[140, 149]. The calibration set is then used to compute nonconformity scores (typically residual errors)[140, 149]. The resulting empirical distribution of these scores determines the critical nonconformity score ($\hat{\epsilon}$), which then defines the guaranteed prediction interval for new examples[140]. For multi-horizon forecasting (predicting H>1 steps simultaneously), prediction intervals must be generated directly, not recursively, and often require Bonferroni correction (α/H) to the significance level to maintain the joint coverage guarantee across the entire forecast horizon[140, 149]. While CF-RNNs successfully integrate this framework, guaranteeing target coverage, classical uncertainty quantification baselines like Multi-Quantile RNNs (MQ-RNNs) and Monte Carlo Dropout (DP-RNNs) frequently fail to empirically achieve the target coverage rate[140].

2.6.2 Deep Transformer Architectures and Conformal Prediction for Time Series

The effectiveness of CP hinges on the quality of the underlying point prediction model. Modern architectures like the Transformer, which excel at modeling long-range dependencies, present an ideal foundation for improved conformal forecasting[149].

The Sequential Predictive Conformal Inference with Transformer (SPCI-T) approach leverages the Transformer decoder as a conditional quantile estimator, learning temporal dependencies specifically across past prediction residuals and features (X_t) to estimate the quantiles of future residuals[140]. While the complexity of designing Transformer-based time series models is higher[149], SPCI-T empirically demonstrates superiority, achieving the narrowest prediction interval widths while meeting coverage requirements compared to other sequential CP methods[140]. This highlights the benefit of deep sequential models learning complex dependencies in the error structure itself[140].

A broader architecture, like the EMForecaster, integrates multiple advanced DL tech-

niques including patching, Reversible Instance Normalization (RevIN) for distribution shift mitigation, and a spatiotemporal MLP backbone (STB) to achieve state-of-the-art point forecasting performance[149]. This model is augmented with CP to provide robust uncertainty quantification[149]. However, evaluating the resulting prediction intervals involves navigating the fundamental trade-off between coverage and interval width[149]. To address this, the Trade-off Score (TOS) is proposed as a unified metric combining Independent Coverage (IC), Joint Coverage (JC), and Mean Interval Width (MIW)[149].

The performance of these deep CP frameworks reveals a critical gap while SPCI-T excels at efficiency (narrow MIW)[140] and EMForecaster achieves high coverage guarantees and competitive TOS[149], there is a continuous need for integrated approaches that tailor deep sequential models to optimally model residual uncertainties. Further research must focus on architectures that refine the nonconformity score definitions to be more adaptive to individual examples, reducing the Mean Interval Width while strictly upholding the theoretical coverage provided by CP[140].

2.7 Synthesis and Research Positioning

Recent advancements in uncertainty quantification for deep learning models, particularly in the context of long-term time series forecasting (LTSF), have focused on generating calibrated prediction intervals (PIs) that balance coverage and sharpness. This section synthesizes the key methodological approaches reviewed and positions the current research landscape.

2.7.1 Comparative Analysis of PI Construction Approaches

Table 2.2 provides a systematic comparison of optimization approaches across different PI construction methods, evaluating them based on five critical dimensions: direct optimization capability, gradient compatibility, multi-objective handling, self-adaptive mechanisms, and horizon-aware adaptation.

Table 2.2: Comparison of Optimization Approaches Author's own compilation based on [7, 8, 9, 10, 11, 12, 15, 16, 17, 18, 12, 151].

Approach	Direct Optimization	Gradient-Compatible	Multi-Objective	Self-Adaptive	Horizon-Aware
Delta [150]	✗	✗	✗	✗	✗
BNNs [16, 15]	✗	✗	✓	✗	✗
Bootstrap [151]	✗	✗	✗	✗	✗
MVE [17]	✓	✓	✗	✗	✗
LUBE [106]	✓	✗	✓	✗	✗
QD [11]	✓	✓	✗	✗	✗
QD+ [10]	✓	✓	✓	✗	✗
AB [7]	✓	✓	✗	✗	✗
DualAQD [8]	✓	✓	✓	Partial	✗
Sum-k [9]	✓	✓	✓	✗	✗
SA-QD (Proposed)	✓	✓	✓	✓	✓

2.7.2 Identified Research Gaps

Despite these advances, existing methods exhibit critical gaps that motivate the proposed research approach:

1. **Hyperparameter Dependency:** Most methods rely on manual hyperparameter tuning for balancing coverage and sharpness, which is time-consuming and dataset-dependent, as highlighted in QD and QR frameworks.

2. **Horizon-Agnostic Treatment:** Few methods address horizon-specific adaptation in LTSF, where uncertainty grows with forecast length, leading to uniform treatment across horizons in methods like Deep Ensembles and DeepAR.

3. **Computational Scalability:** Computational feasibility remains a challenge for BNNs in large-scale LTSF datasets, while distribution-free methods like CP lack integration with Transformer architectures for capturing long-range dependencies.

4. **Lack of Learnable Parameters:** None of the existing methods incorporate learnable parameters to dynamically optimize PI quality without manual oversight.

5. **Limited Transformer-Ensemble Integration:** Current approaches do not com-

bine deep Transformer ensembles with self-adaptive losses for calibrated, sharp PIs in real-world LTSF scenarios like energy and finance.

Recent advancements in uncertainty quantification for deep learning models, particularly in the context of long-term time series forecasting (LTSF), have focused on generating calibrated prediction intervals (PIs) that balance coverage and sharpness. Self-Adaptive Quality-Driven (SA-QD) loss function,This method extends traditional quality-driven approaches by introducing learnable parameters to dynamically balance PI objectives, addressing horizon-specific uncertainty in Transformer-based ensembles. This methods have been proposed to address this challenge, each with unique strengths and limitations.

Having established the theoretical foundation and identified critical research gaps in this literature review, the following chapters detail the methodology, experimental design, and empirical validation of the proposed framework.

This literature review has systematically examined the evolution and current state of uncertainty quantification in deep learning, with particular focus on long-term time series forecasting. Beginning with fundamental sources of uncertainty (aleatoric and epistemic), the review progressed through Transformer architectures for time series, comprehensive coverage of prediction interval construction methods (both indirect and direct approaches), ensemble strategies, and conformal prediction frameworks. The synthesis reveals critical gaps in current methodologies, particularly regarding manual hyperparameter tuning, lack of horizon-specific adaptation, and limited integration of advanced Transformer architectures with robust uncertainty quantification mechanisms. These identified gaps directly motivate and inform the proposed Self-Adaptive Quality-Driven framework that combines learnable parameters, horizon-aware adaptation, and Transformer-based ensembles for calibrated prediction intervals in long-term time series forecasting. Having established the theoretical foundation in the literature review, the following section details the methodology.

Chapter 3

METHODOLOGY OF THE STUDY

The objective of this research is to design the developed framework that significantly enhances the calibration and sharpness of prediction intervals for long-term time series forecasting, particularly within Transformer-based models. This is achieved by introducing learnable parameters into the quality-driven loss functions, thereby minimizing the reliance on manual hyperparameter tuning. A key innovation is the Self-Adaptive Quality-Driven Loss (SA-QD), which incorporates a differentiable approximation for coverage, normalized interval widths, asymmetric penalties, and learnable regularization terms including a logarithmic-parameterized lambda for dynamic coverage-sharpness balancing. The subsequent subsections detail the research methodology employed to achieve this objective.

3.1 Research Design

To conduct this research, the Design Science Research Process (DSRP) [152] is adopted as the guiding methodology. DSRP aligns exceptionally well with the overarching goal of addressing an existing, critical problem in time series forecasting the lack of automated and adaptive uncertainty quantification and subsequently designing, developing, demonstrating, and evaluating a new method to solve it. The DSRP, as articulated by [19], typically comprises five major steps: Problem dentification & Motivation, Objective Formulation, Design and Development, Demonstration, Evaluation, and Communication. This structured approach facilitates the creation of innovative artifacts while ensuring rigorous theoretical contributions. In this thesis, the DSRP framework is systematically applied to guide the research from the initial problem definition through to the final communication of findings. While the core steps are adhered to, the iterative nature of research means that some phases may overlap or be revisited. The following sections

detail how each component of the DSRP is integrated into the research process which is also depicted in Figure 3.1:, ensuring a clear and coherent methodological approach.

- **Problem Identification & Motivation** The research begins by identifying the challenge of uncertainty quantification in long-term time series forecasting, particularly the inefficiencies of manual hyperparameter tuning in prediction interval calibration. A literature review on deep learning, uncertainty quantification, and prediction interval calibration reveals a research gap the lack of reliable, automated methods for predictive uncertainty with learnable parameters, despite advances in models like Transformers. This gap shapes the direction for a new solution.

- **Objective Formulation:** The research aims to develop a Transformer-based framework for long-term time series forecasting that automates the optimization of quality-driven loss functions to produce well-calibrated, sharp prediction intervals. Specific objectives include designing an adaptive loss function with learnable regularization parameters (e.g., mpiw_reg($\mu_h(t)$) and a logarithmic lambda_weight(λ)), integrating it with Transformer deep ensembles and complementing it with horizon-specific conformal prediction for refinement, evaluating performance across diverse datasets, and ensuring adaptability to various forecast horizons, providing a clear roadmap for the project.

- **Design and Development:** This phase focuses on creating a proposed framework to address the identified problem, including a specific model architecture, a sloss function, and an adaptive learning mechanism. The design targets the limitations of manual hyperparameter tuning and the lack of horizon-specific adaptation in prediction interval calibration, building on insights from prior phases. Key artifacts include the SA-QD loss, which features learnable mpiw_reg($\mu_h(t)$) and lambda_weight(λ) parameters (via logarithmic parameterization for stability) for self-adaptive sharpness and coverage regularization.

- **Demonstration:** Following the design and development phase, the constructed artifact is demonstrated to verify its feasibility and operational correctness before formal evaluation. The demonstration involves deploying the complete SA-QD framework com-

prising the modified PatchTST ensemble, the self-adaptive loss function, and the HSCP calibration module on the benchmark datasets (ETTh2 and Exchange Rate) across all four forecast horizons (96, 192, 336, and 720 steps). During this phase, the framework is executed end-to-end to confirm that: (1) the learnable parameters (λ_h and μ_h) converge to stable values during training via gradient-based updates, as evidenced by the training log analyses presented in the results chapter; (2) the PatchTST model correctly outputs mean, lower, and upper bounds through the custom FlattenHead, with min/max enforcement ensuring valid interval ordering ($lower \leq upper$); (3) the HSCP post-processing step successfully computes horizon-specific and channel-specific non-conformity quantiles from the calibration set and applies them to adjust test-time prediction intervals; and (4) the complete pipeline from data preprocessing through normalization, patching, ensemble aggregation, SA-QD loss optimization, and conformal calibration executes without failure across all experimental configurations. The shell scripts (e.g., PatchTST_ETTh2.sh) automate ensemble runs with varying seeds and horizons, while the training flowchart (Figure 3.3) illustrates the operational sequence of the demonstrated artifact. This demonstration step, conducted prior to formal evaluation, provides evidence to domain experts and potential users that the artifact functions as designed and produces interpretable outputs (prediction intervals with associated coverage and width metrics), thereby establishing confidence in the artifact's readiness for rigorous quantitative assessment.

- **Evaluation:** The framework's effectiveness is assessed through quantitative metrics like Prediction Interval Coverage Probability (PICP), Mean Prediction Interval Width (MPIW), Interval Score, and Continuous Ranked Probability Score (CRPS), compared against quality-driven loss methods. Qualitative assessments include visual analysis of prediction intervals (e.g., via plots of mean, lower, and upper bounds) to verify calibration and adaptability, ensuring the framework meets its objectives and addresses the research problem.

- **Communication:** Research findings, including the artifact and generated knowledge, are shared through the thesis, peer-reviewed publications, and conference presentations.

An open-source implementation of the framework (including the updated lhsa_qd_loss function with learnable lambda and PatchTST model integration) is provided to ensure reproducibility, encourage further research, and promote practical adoption, aligning with the DSRP's emphasis on utility and broad dissemination.

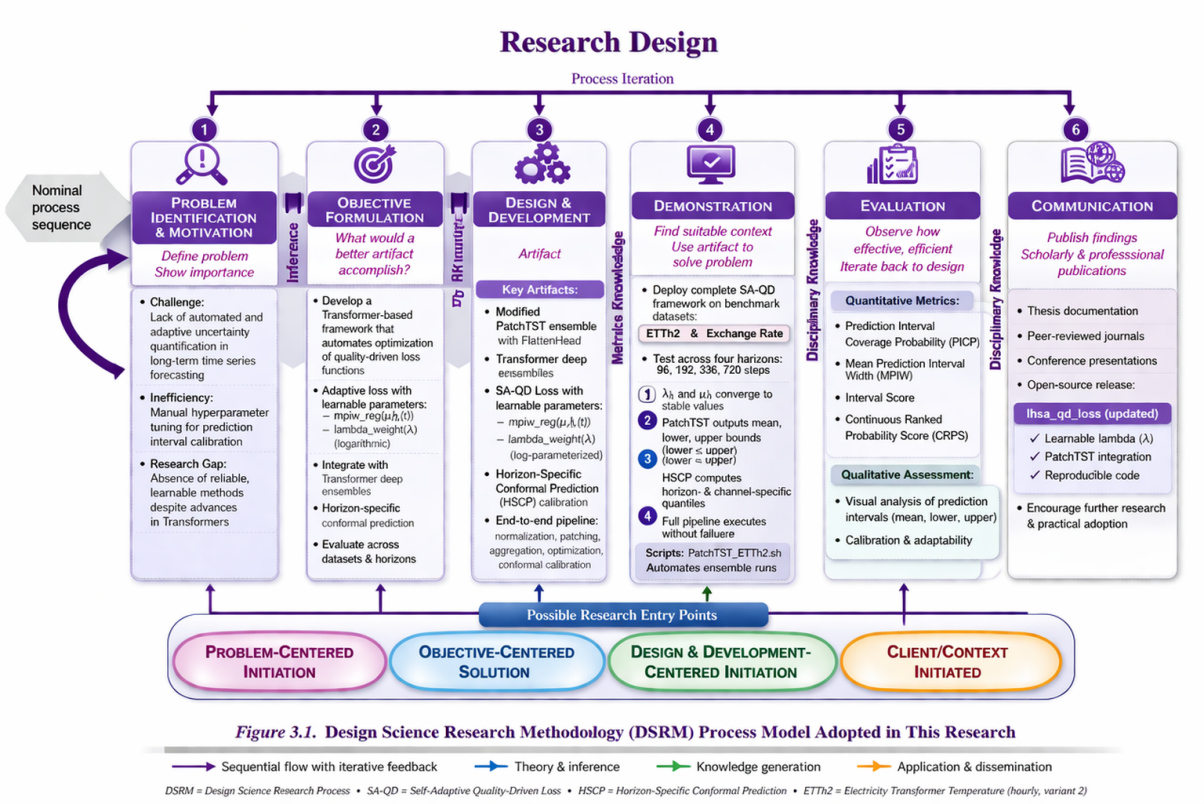

Figure 3.1: Design Science Research Methodology (DSRM) Process Model Adopted in this thesis work [19].

This research proposes a new integrated framework addressing critical limitations in existing uncertainty quantification approaches. The methodology synergistically combines three innovations:

1. Deep ensembles of Transformer models designed for uncertainty-aware forecasting,
2. Self-Adaptive Quality-Driven Loss function automatically learning optimal objective weighting and quality PI.
3. To calibrated PI with guarantee a distribution free conformal prediction.

Our approach, depicted in Figure 3.1, aims to leverage the predictive power of Transformers while ensuring that the resultant PIs possess desirable statistical properties,

particularly robust coverage guarantees and adaptability to horizon-specific uncertainty dynamics.

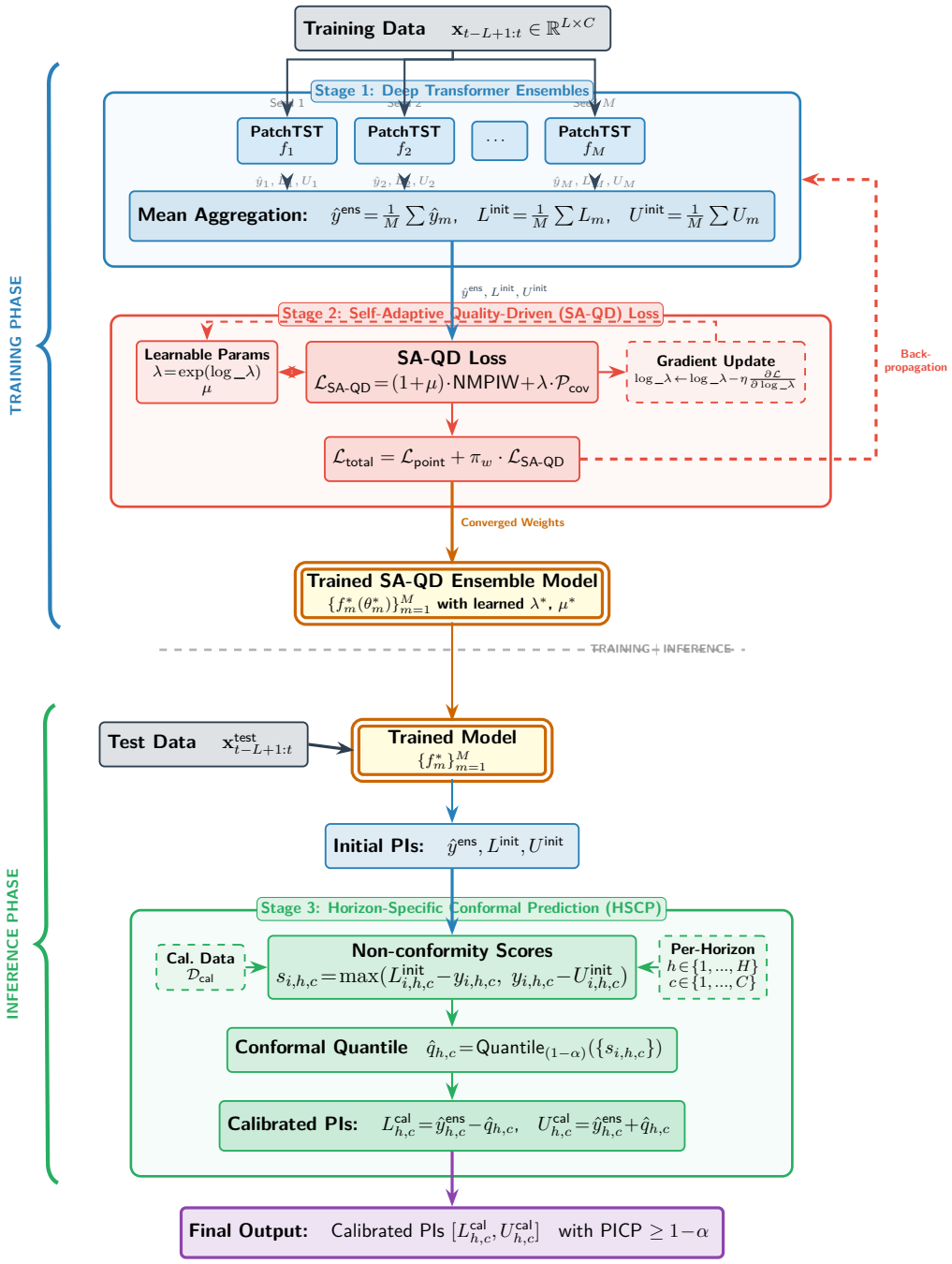

Figure 3.2: Block diagram of the proposed SA-QD framework for generating calibrated prediction intervals.

Figure 3.2 illustrates the complete architecture of the proposed SA-QD framework, comprising a training phase and an inference phase. During the **training phase**, the framework operates through two integrated stages. In Stage 1 (blue), an ensemble of M

independently initialized PatchTST models processes the training input $\mathbf{x}_{t-L+1:t}$ to produce point predictions ($\hat{y}_m$) and initial interval bounds (L_m, U_m), which are aggregated via mean averaging. In Stage 2 (red), the Self-Adaptive Quality-Driven (SA-QD) loss function optimizes the coverage-sharpness trade-off using learnable parameters ($\lambda = \exp(\log\_\lambda)$ and μ), which are jointly updated with the model weights through gradient descent. The dashed red arrow on the right indicates the backpropagation path from the total loss $\mathcal{L}_{\text{total}}$ back to the ensemble models. Upon convergence, the training phase produces a **Trained SA-QD Ensemble Model** containing the optimized weights $\{\theta_m^*\}_{m=1}^M$ and learned parameters λ^* and μ^*.

During the **inference phase**, the trained model receives unseen test data $\mathbf{x}_{t-L+1:t}^{\text{test}}$ as input and generates initial prediction intervals. These initial intervals are then refined in Stage 3 (green) through Horizon-Specific Conformal Prediction (HSCP), which computes non-conformity scores on a held-out calibration set $\mathcal{D}_{\text{cal}}$ independently for each forecast horizon h and channel c. The conformal quantiles $\hat{q}_{h,c}$ are applied to adjust the intervals, producing the final calibrated prediction intervals $[L_{h,c}^{\text{cal}},\ U_{h,c}^{\text{cal}}]$ with distribution-free coverage guarantees (PICP $\geq 1-\alpha$). The horizontal dashed gray line separates the training phase (Stages 1–2) from the inference phase (Stage 3 with test input).

The complete framework can be expressed as: $\mathcal{F} : \mathbb{R}^L \to \mathbb{R}^H \times \mathbb{R}^{H\times 2}$, mapping input sequences to both point predictions and prediction intervals: $\mathcal{F}(x) = (\hat{y}_{1:H}, [L_{1:H}^{\text{cal}}, U_{1:H}^{\text{cal}}])$. The subsequent sections will elaborate on each of these components, providing detailed justifications for their design choices and their synergistic contributions to the overall objective of generating highly reliable and well-calibrated prediction intervals for LTSF.

Key Properties:

Coverage Guarantee $P(y_{t+h} \in [L_{t+h}^{\text{cal}}, U_{t+h}^{\text{cal}}]) \geq (1-\alpha_h)$

Sharpness Optimization $\min \mathbb{E}[U_{t+h}^{\text{cal}} - L_{t+h}^{\text{cal}}]$ subject to coverage constraints.

Computational Efficiency Maintains reasonable computational requirements for practical deployment. While training an M-member ensemble is M times more expensive than a single model, this process is highly parallelizable and often more computationally

tractable than training fully Bayesian methods. Critically, the Horizon-Specific Conformal Prediction step is a rapid post-processing calculation that adds negligible overhead, offering a significant advantage over methods requiring retraining for calibration.

3.1.1 Deep Transformer Ensembles for Uncertainty-Aware Forecasting

The foundation of our proposed framework for robust uncertainty quantification in long term time series forecasting (LTSF) is built upon a deep ensemble of Transformer models[1] (see Section 2.5). This architectural choice is motivated by the synergistic benefits offered by both deep ensembles and Transformer networks in addressing the inherent complexities and uncertainties of LTSF. While Transformer models excel at capturing long-range temporal dependencies and achieving high point prediction accuracy, deep ensembles provide a principled and empirically effective mechanism for quantifying both aleatoric and epistemic uncertainties [14]. Recent reviews of Transformer-based LTSF further emphasize the need for such UQ enhancements in ensembles.

Rationale for Deep Ensembles

Deep ensembles, popularized by, involve training multiple neural networks (often with identical architectures) independently from different random initializations. Each network learns a slightly different mapping from input to output, leading to a diverse set of predictions. This diversity is crucial for robust uncertainty quantification for several reasons:

Principled Uncertainty Decomposition: Deep ensembles disentangle and quantify the two primary sources of predictive uncertainty see Section 2.2.1. Aleatoric uncertainty (inherent data noise) is captured by configuring each model to predict distribution parameters (e.g., mean and variance) see Subsection 2.2.1. Epistemic uncertainty (model limitations) is estimated from the disagreement across predictions see Subsection 2.2.1. In regions with sparse data or for long-term forecasts requiring extrapolation, predictions diverge, providing a direct measure of model uncertainty [57].

Enhanced Robustness and Generalization: Aggregating diverse models mitigates overfitting risks, leading to more stable point forecasts and reliable prediction

intervals [153]. This is invaluable for noisy time series patterns.

Computational Feasibility: While more intensive than single models, ensembles are parallelizable and more scalable than fully Bayesian Neural Networks (BNNs), which involve intractable inference.

Transformer Architecture for Time Series Forecasting

The choice of Transformer architecture as the base model for each ensemble member is driven by its proven superiority in sequence modeling, particularly for long-range dependencies characteristic of LTSF[154, 155] see Section 2.3.2. Transformers overcome limitations of RNNs (vanishing gradients) and CNNs (limited receptive fields) through their self-attention mechanism [156, 53].

Uncertainty-Aware Modifications The self-attention mechanism is enhanced with ensemble-based diversity during training, using fixed seeds from 1 to 5 per forecasting horizon to promote varied uncertainty estimates across long-term predictions.

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

This mechanism allows each element in the output sequence to attend to all elements in the input sequence, effectively capturing global dependencies. For LTSF, this means the model can directly relate a future prediction to relevant past observations, even if they are far apart in time, without the need for complex recurrent connections.

Adaptations for LTSF Our framework incorporates optimized Transformer variants such as PatchTST [23], which divide the input sequence into patches. Specifically, the input is reshaped as $X_{\text{patch}} = \text{Reshape}(x) \in \mathbb{R}^{N_p \times P}$, where $N_p = \lfloor L/P \rfloor$ is the number of patches given patch length P and input length L. The patches are then projected to the model dimension via a linear transformation:

$$H^{(0)} = X_{\text{patch}} W_e + b_e$$

Each PatchTST model is configured to output both the lower and upper bounds of the prediction interval, enabling end-to-end training for uncertainty-aware forecasting.

Ensemble Diversity Mechanisms Our ensemble design specifically addresses the limitations identified in recent literature regarding for uncertainty quantification in time series forecasting. We design the ensemble to work synergistically with our HSCP framework.

Initialization Diversity Each member is initialized with different random seeds: $\mathbf{W}_m^{(0)} \sim \mathcal{N}(0, \sigma_{\text{init},m}^2)$ where $\sigma_{\text{init},m}$ varies across members.

Architectural Diversity Different configurations including attention heads $h_m \in \{4, 8, 16\}$, model dimensions $d_{\text{model},m} \in \{512, 760, 1024\}$, and patch sizes $P_m \in \{16, 32\}$.

Training Diversity Bootstrap sampling maintaining temporal order: $\mathcal{D}_m = \{(x_{t:t+L}, y_{t+1:t+H}) : t \in \mathcal{T}_m\}$, different dropout rates, and optimization configurations.

Base Model PatchTST

To effectively address the challenges of long-term time series forecasting (LTSF) with reliable uncertainty quantification, a robust base model is essential. This research adopts **PatchTST (Patch Time Series Transformer)** as the foundational architecture, selected for its state-of-the-art performance in LTSF tasks and its ability to handle multivariate, long-sequence inputs efficiently. PatchTST extends traditional Transformer models by incorporating patching mechanisms to reduce computational complexity while preserving temporal dependencies, making it particularly suitable for generating point estimates alongside prediction intervals (PIs).

In our framework, PatchTST is modified to output not only the mean forecast but also lower and upper bounds directly, enabling end-to-end training with the **Self-Adaptive Quality-Driven Loss (SA-QD)** and integration with deep ensembles for enhanced epistemic uncertainty capture. The subsequent subsections detail the rationale, design, and implementation of this base model within the proposed framework.

Limitations of Prior Transformer Models in LTSF Traditional Transformer architectures, while powerful for sequence modeling, face significant limitations in LTSF scenarios. Standard encoders process sequences token-by-token, leading to quadratic computational complexity ($O(L^2)$ for sequence length L), which becomes prohibitive for long horizons (e.g., `pred_len` $= 720$). Moreover, vanilla Transformers often struggle with multivariate time series due to channel mixing and lack explicit handling of local temporal patterns like seasonality or trends.

Prior uncertainty-aware variants, such as those using Bayesian layers or Monte Carlo dropout, add overhead without addressing efficiency, and typically output only means or variances, requiring post-hoc PI construction that may lack sharpness [11].

These limitations are exacerbated in LTSF with PIs: Inefficient scaling to long inputs hinders modeling distant dependencies critical for extended forecasts.

Poor channel independence can lead to correlated errors across variables, degrading multivariate PI calibration.

Absence of direct bound outputs necessitates separate heads or approximations, complicating optimization with QD losses.

Limited ensemble support increases variance in uncertainty estimates without built-in mechanisms for aggregation.

These gaps motivate the selection and adaptation of PatchTST, which mitigates efficiency issues while providing a flexible backbone for PI generation.

Design of the PatchTST Base Model PatchTST addresses the inefficiencies of token-based Transformers by dividing the input sequence into non-overlapping patches, similar to vision Transformers. This reduces the effective sequence length from L to approximately L/P (where P is patch length), enabling linear complexity while capturing both local and global patterns. In our design, PatchTST serves as the core encoder-decoder for multivariate forecasting, with modifications to support direct PI outputs and integration with SA-QD.

Key components:

Input Processing and Patching: The input x_{enc} (shape: [batch, seq_len, enc_in])

is normalized (z-score per channel) and permuted to [batch, enc_in, seq_len]. `PatchEmbedding` creates patches of length `patch_len`=16 with stride=8 and padding=8, embedding them into d_{model} dimensions. This yields *enc_out* with reduced temporal dimension, preserving channel-wise information.

Encoder: A stack of *e_layers* (e.g., 4) `EncoderLayer`s, each with `FullAttention` (*n_heads*, factor=3, dropout=0.1) for self-attention, followed by feed-forward networks (d_{ff}=2048, activation='gelu'). Batch normalization is applied via a custom `Transpose` `norm_layer` for stability. This captures multi-scale temporal dependencies without distillation (`args.distil=False`).

Output Head: A custom `FlattenHead` flattens the encoder output

$$\text{head\_nf} = d_{\text{model}} \cdot \left(\frac{\text{seq\_len} - \text{patch\_len}}{\text{stride}} + 2 \right)$$

and uses three linear layers (`linear_mean`, `linear_lower`, `linear_upper`) with dropout to produce mean, lower, and upper bounds (each [batch, pred_len, c_out]). Bounds are enforced via min/max to ensure lower $\leq$ upper.

Forecasting Pipeline: In `forecast/forward` methods, inputs are processed, outputs denormalized, and returned as tensors for loss computation. This direct bound output enables seamless integration with `lhsa_qd_loss`.

To enhance uncertainty quantification, we incorporate deep ensembles: Multiple PatchTST instances (`ensemble_size`=1 by default, extensible via seeds in `PatchTST_ETTh2.sh`) are trained independently with varying random seeds, aggregating means and bounds (e.g., mean across ensembles) to capture epistemic uncertainty [157]. Hyperparameters vary per horizon (e.g., $d_{\text{model}} = 768/1024$, $n\_heads = 16/4$ for shorter/longer `pred_len`) to adapt to increasing complexity.

This design automates PI generation while maintaining efficiency, with $O((L/P)^2)$ attention complexity for long sequences.

Point Prediction Loss Anchoring the Forecast While the primary objective is to construct high-quality prediction intervals, the credibility of these intervals is fundamen-

tally anchored to the accuracy of a central point forecast[10]. An interval, no matter how well-calibrated, loses its practical value if it is centered around a wildly inaccurate prediction.

To enforce this foundational accuracy, we incorporate a point prediction loss term based on the Mean Squared Error (MSE)[136, 22, 158]. The MSE is a cornerstone of regression tasks for two principal reasons. First, its quadratic nature heavily penalizes large deviations, compelling the model to correct significant errors more aggressively[159]. Second, it provides a smooth, differentiable loss surface, a critical property for effective gradient-based optimization[132, 135].

For a specific forecast horizon h, the point prediction loss is calculated over a batch of data as follows:

$$\mathcal{L}_{\text{point}}(h) = \frac{1}{B}\sum_{i=1}^{B}\left(y_{i,h} - \hat{y}_{\text{ensemble},i,h}\right)^2$$

In this formulation, B is the number of samples in the batch, $y_{i,h}$ represents the ground truth value for the i-th sample at horizon h, and $\hat{y}_{\text{ensemble},i,h}$ is the corresponding point prediction generated by the ensemble (typically the mean of the individual models' outputs)[21].

This loss component, therefore, serves a vital role it ensures that the central tendency of our forecast is reliably accurate, establishing a solid foundation upon which meaningful and trustworthy prediction intervals can be constructed.

Ensemble Aggregation and Prediction

For a given input, each of M models produces predictions. The ensemble point prediction is the mean Point prediction.

$$\hat{y}_{t+h}^{\text{ensemble}} = \frac{1}{M}\sum_{m=1}^{M}\hat{y}_{t+h}^{(m)}$$

Uncertainty decomposition.

$$\sigma^2_{\text{total}}(x) = \sigma^2_{\text{epistemic}}(x) + \sigma^2_{\text{aleatoric}}(x)$$

where:

$$\sigma^2_{\text{epistemic}}(x) = \frac{1}{M-1}\sum_{m=1}^{M}(\hat{y}^{(m)}(x) - \hat{y}^{\text{ensemble}}(x))^2$$

$$\sigma^2_{\text{aleatoric}}(x) = \frac{1}{M}\sum_{m=1}^{M}\sigma^2_m(x)$$

To capture aleatoric uncertainty, the model estimates a sample-dependent variance $\sigma^2_m(x)$, which is then aggregated across the ensemble to provide a total predictive standard deviation.

Initial intervals: $[L^{\text{init}}_{t+h}, U^{\text{init}}_{t+h}] = \hat{y}^{\text{ensemble}}_{t+h} \pm z_{1-\alpha/2}\sigma_{\text{total}}(x)$.

Constructs a prediction interval centered at the point prediction $\hat{y}^{\text{ensemble}}_{t+h}$[160]. The term $z_{1-\alpha/2}$ is the standard z-score (quantile) corresponding to the desired confidence level $(1-\alpha)$, and $\sigma_{\text{total}}(x)$ is the total predictive standard deviation (the square root of $\sigma^2_{\text{total}}(x)$). This approach relies on approximating the combined ensemble output (which is a mixture of distributions) as a single Gaussian distribution.

Non-Parametric Generation of Initial Prediction Intervals The primary goal of the ensemble is to quantify uncertainty through the generation of prediction intervals (PIs) avoid strong assumptions about the underlying error distribution. The framework leverages the diversity of the ensemble members to construct these intervals in a non-parametric and empirically-driven manner, avoiding strong assumptions about the underlying error distribution[13].

Procedure:

1. For a given forecast horizon h, collect the predictions for the lower and upper bounds from all M ensemble members. This results in two sets of M values one for the lower bounds and one for the upper bounds.

2. The final, aggregated prediction interval is constructed by taking the empirical quantiles of these collections[161]. For a desired nominal coverage of $(1-\alpha)$:

    - The lower bound L_{t+h} is the $\alpha/2$ quantile of the predicted lower bounds.

- The upper bound U_{t+h} is the $(1 - \alpha/2)$ quantile of the predicted upper bounds-seligmann2023beyond.

Formally:

$$L_{t+h} = \text{Quantile}\left(\{L_{m,t+h}\}_{m=1}^{M},\ \alpha/2\right)$$

$$U_{t+h} = \text{Quantile}\left(\{U_{m,t+h}\}_{m=1}^{M},\ 1 - \alpha/2\right)$$

For instance, to construct a 95% prediction interval ($\alpha = 0.05$), the 2.5th percentile of the predicted lower bounds and the 97.5th percentile of the predicted upper bounds would be used. Ensemble methods improve performance by leveraging the diversity of predictions from different models.

3.1.2 Self-Adaptive Quality-Driven Loss ($\mathcal{L}_{\text{SA-QD}}$) for Initial PI

While deep ensembles provide a robust foundation for uncertainty quantification by capturing both aleatoric and epistemic uncertainties, the quality of the resulting prediction intervals (PIs) can be further optimized through a carefully designed loss function. Traditional quality-driven (QD) loss functions, while enabling differentiability, often rely on manually tuned hyperparameters to balance the conflicting objectives of coverage and sharpness. This manual intervention is a significant limitation, particularly in long-term time series forecasting (LTSF) where the optimal balance may vary dynamically across different forecast horizons. To address this, we propose a novel Self Adaptive Quality-Driven Loss ($\mathcal{L}_{\text{SA-QD}}$) function, designed to make the critical balancing parameters learnable and adaptive to the forecast horizon. The incorporation of learnable regularization terms, including a logarithmic-parameterized lambda for dynamic coverage-sharpness balancing, alongside a differentiable approximation for coverage and normalized interval widths

Limitations of Prior Quality-Driven Loss Functions

As discussed in Chapter 2, conventional QD loss functions, such as those proposed by pearce et al.[11] and their variants, typically formulate the optimization problem as a

weighted sum of a coverage term and a sharpness term are explained in (see Subsection 2.4.3). The parameter $\lambda \in [0, 1]$ is a manually chosen weighting factor that dictates the trade-off between these two objectives. A higher λ prioritizes achieving the nominal coverage, potentially at the cost of wider intervals, while a lower λ emphasizes narrower intervals, risking undercoverage. The critical limitation here is that λ is a fixed hyperparameter, requiring extensive and often computationally expensive manual tuning (e.g., via grid search or random search) to find an optimal value for a given dataset and desired confidence level [91, 92, 162].

This manual tuning process presents several challenges

Lack of Adaptability: The optimal λ is not static; it can vary significantly across different datasets, time series characteristics (e.g., volatility, seasonality), and, crucially, across different forecast horizons[90]. A fixed λ cannot dynamically adapt to these changing conditions, leading to sub-optimal PI quality (either undercovered or overly wide) in many scenarios.

Sensitivity to Initial Conditions: The performance of QD loss functions can be highly sensitive to the initial choice of λ, making robust and generalizable deployment difficult.

Sub-optimal Performance: Manual tuning often results in a sub-optimal balance between coverage and sharpness, as it is challenging to find the true optimal point in a complex, non-convex loss landscape. This can lead to PIs that are either unreliable (poor coverage) or uninformative (poor sharpness), undermining their utility in decision-making.

Computational Burden: Exhaustive search over the hyperparameter space is computationally prohibitive, especially for deep learning models that require significant training time. These limitations are particularly pronounced in LTSF, where the uncertainty inherently increases with the forecast horizon. A fixed λ cannot account for this horizon-dependent increase in uncertainty, leading to PIs that are often too narrow for distant future predictions (resulting in under-coverage) and potentially wider than necessary for near-term predictions (sacrificing sharpness). This highlights the urgent need for a mech-

anism that allows the model to learn and adapt this balancing parameter dynamically.

Design Of The Self-Adaptive Quality-Driven Loss ($\mathcal{L}_{\text{SA-QD}}$)

The formulation of the Quality-Driven loss function, presented in (see Subsection 2.4.3) Equation, is designed to address a fundamental challenge in probabilistic forecasting the inherent trade-off between the *precision* (width) and the *reliability* (coverage) of prediction intervals, extended by recent works like enhanced QD for wind speed forecasting and DualAQD[163, 8]. Rather than relying on a manually tuned hyperparameter to balance these objectives, this loss function introduces a self-regulating mechanism that allows the model to learn the optimal balance during the training process itself. The SA-QD loss addresses the limitations of fixed-weight QD losses by introducing a self-adaptive weighting mechanism. The loss for each horizon h_t is defined as:

$$\mathcal{L}_{\text{SA-QD}} = (1 + \mu_h(t)) \cdot \text{NMPIW}_{\text{width}} + \lambda_h(t) \cdot \mathcal{P}_{\text{coverage}}.$$

where:

The First Objective The Pursuit of Precision (NMPIW) The Mean Prediction Interval Width (MPIW) serves as the primary metric for interval sharpness. It quantifies the average tightness of the predicted intervals over a batch of data. By minimizing this term, the model is encouraged to avoid producing overly wide and uninformative intervals, thereby enhancing the practical utility of its forecasts. $\textbf{NMPIW} = \frac{\text{MPIW}}{r}$ is the normalized mean prediction interval width, with $\text{MPIW} = \frac{1}{N}\sum_{i=1}^{N}(U_{t+h} - L_{t+h})$ representing the average PI width over N samples,(L_{t+h}, U_{t+h} *Subsection* 3.1.1) and $r = \max(y) - \min(y) + \epsilon$ (with $\epsilon = 10^{-8}$ for numerical stability) normalizing for data scale.

The Second Objective The Mandate for Reliability ($\mathcal{P}_{\text{coverage}}$) The second major component of the loss function is a sophisticated penalty term designed to enforce statistical reliability. This term remains dormant as long as the model meets the desired coverage probability but activates decisively when it fails. $\mathcal{P}_{\text{coverage}} = w_u \cdot [\max(0, (1 - \alpha) - \text{PICP})]^2 + w_o \cdot [\max(0, \text{PICP} - (1 - \alpha))]^2$ is the asymmetric squared coverage penalty,

with PICP approximated differentiably as PICP $= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}[\tanh(s\cdot(\hat{y}_{t+h}-L_{t+h}))+\tanh(s\cdot(U_{t+h}-\hat{y}_{t+h}))]$ (using tanh for smooth indicator, $s = 100$ for sharpness), w_u and w_o as weights for under- and over-coverage penalties, and α the target miscoverage rate.

In our framework implementation , these correspond to under_penalty_weight and over_penalty_weight. [PICP vs. $(1-\alpha)$ (Performance vs. Target)] The core of the penalty is a comparison between the model's empirical performance, the **Prediction Interval Coverage Probability (PICP)**, and the user-defined target confidence level, $(1-\alpha)$. For a 95% confidence level, the target $(1-\alpha)$ would be 0.95, and the model must ensure its PICP meets or exceeds this value. [$\max(0,\dots)$ (The Conditional Enforcement Mechanism)] This hinge loss component acts as a conditional switch. If the model's coverage (PICP) is sufficient (i.e., PICP $\geq 1-\alpha$), the term inside the max function becomes non-positive, resulting in a value of zero.

The entire penalty vanishes, and the model is free to focus solely on minimizing interval width. However, if coverage is insufficient, the term becomes positive, activating the penalty. [$(\dots)^2$ (The Quadratic Penalty)] The magnitude of the coverage shortfall is squared. This ensures that minor deviations from the target coverage are penalized lightly, while significant failures are penalized heavily. This behavior strongly incentivizes the model to correct substantial lapses in reliability. [$n/(\alpha(1-\alpha))$ (The Normalization Scaler)] This term serves to normalize the penalty's magnitude, ensuring stable and consistent training dynamics across different batch sizes (n) and target confidence levels (α).

The Dynamic Balancing Weights $\mu(t)$ and $\lambda(t)$ At the heart of this adaptive behavior are two learnable, competing weights, $\mu(t)$ and $\lambda(t)$. These parameters are not static but evolve at each training step t. They dynamically arbitrate the focus of the optimization process.

$\mu(t)$ **(The Precision Weight):** This term represents the learned importance of minimizing the interval width. It drives the model to produce the sharpest, most informative predictions possible.

$\lambda(t)$ **(The Reliability Weight):** This term represents the learned importance of satisfying the statistical coverage requirement. It compels the model to ensure its pre-

diction intervals are trustworthy. $\lambda(t)$ is a learnable weighting parameter for coverage emphasis, and $\mu(t)$ is a learnable regularization parameter to control width expansion. This design automates the trade-off, with $\lambda(t)$ starting near 0.5 (σ_h init=0) and adapting via backpropagation. In implementation, n is set to the total elements in the batch (`batch_size * pred_len * c_out`) for multivariate handling, and gradients are clipped for stability. We will now dissect each component to elucidate its specific role within this synergistic framework. Crucially, these weights are a direct trade-off. An increased focus on precision ($\mu(t)$) necessarily implies a decreased focus on the reliability penalty ($\lambda(t)$), and vice versa. This allows the model to intelligently shift its priorities based on its performance at any given moment.

For multivariate series, MPIW and PICP are computed per-channel and averaged. To ensure differentiability, we approximate PICP using a soft tanh function (see Subsection 3.1.2). The penalty term is quadratic to strongly penalize under-coverage. This adaptive loss function creates an elegant, self-regulating learning environment. The model is not just learning to forecast; it is simultaneously learning how to best balance the critical objectives of precision and reliability, thereby producing high-quality, trustworthy prediction intervals without the need for manual hyperparameter tuning.

Differentiable Approximation of the Coverage Function

A central challenge in optimizing the Prediction Interval Coverage Probability (PICP) directly is the non-differentiable nature of its core counting operation. The standard indicator function, which returns a 1 if a true value falls within its interval and a 0 otherwise, creates a discontinuous, step-like objective landscape (see Subsection 2.4.1). This lack of a smooth gradient makes it incompatible with the gradient-based optimization algorithms used to train neural networks[9].

To overcome this technical hurdle, a common strategy is to replace the rigid indicator function with a smooth, differentiable approximation. While prior work [164] has effectively used a product of sigmoid functions for this purpose, we adopt an alternative formulation based on the sum of hyperbolic tangent (‘tanh‘) functions.

Specifically, we approximate the indicator function, $\mathbb{I}(L_{t+h} \leq \hat{y}_{t+h} \leq U_{t+h})$, as follows:

$$\mathbb{I}(L_{t+h} \leq \hat{y}_{t+h} \leq U_{t+h}) \approx \frac{1}{2} \max\left[0, \tanh\left(s(\hat{y}_{t+h} - L_{t+h})\right) + \tanh\left(s(U_{t+h} - \hat{y}_{t+h})\right)\right]$$

In this formulation, each 'tanh' term creates a soft, S-shaped transition instead of a hard step. The term $\tanh(s(\hat{y}_{t+h} - L_{t+h}))$ smoothly approximates the condition $\hat{y}_{t+h} \geq L_{t+h}$, while $\tanh(s(U_{t+h} - \hat{y}_{t+h}))$ approximates the condition $\hat{y}_{t+h} \leq U_{t+h}$. The hyperparameter s controls the steepness of these transitions, determining how closely the smooth function mimics the original step function (e.g., $s = 100$ for steep sigmoid-like behavior). This allows gradients to flow through the coverage term, enabling optimization of bounds directly. In code, this is integrated into the loss, with means over batch for PICP per horizon/channel. This approximation maintains near-binary behavior for large s while being fully differentiable.

The $\max(0, \cdot)$ operation is included to ensure the output remains non-negative, rectifying the sum since it is possible for it to become slightly negative when a true value $\hat{y}_{t+h}$ lies far outside the interval. While mathematically related to sigmoid-based methods, this 'tanh' formulation is structurally straightforward. By substituting this smooth approximation (Equation 3.1.2) into the PICP calculation, we create a fully differentiable objective that can be seamlessly incorporated into our gradient-based loss function for end-to-end training.

Design of the Adaptive Weighting Mechanism

The most innovative aspect of $\mathcal{L}_{\text{SA-QD}}$ is the introduction of λ_h as a *learnable parameter*. Instead of manually setting a fixed weighting coefficient (such as λ), Our framework incorporates a logarithmic parameterization for stability. `self.log_lambda_weight = nn.Parameter(torch.tensor(np.log(100.0)))`. The corresponding function is defined as $\lambda_h(t) = \exp(\texttt{self.log\_lambda\_weight})$. This sub-network is trained end-to-end along with the main Transformer ensemble. To ensure λ_h remains positive and stable, its output is exponentiated.

This approach offers several advantages, including dynamic adjustment of the trade-

off between coverage and sharpness for different forecast horizons, as well as improved adaptability to varying prediction tasks. The learnable parameter `mpiw_reg` (initialized as `nn.Parameter(0.01)`) provides additional regularization on sharpness, preventing unchecked width expansion.

Parameter Update Rule: The parameters {`log_lambda_weight`, `mpiw_reg`} are updated via gradient descent using the Adam optimizer [107], alongside the model parameters:

```
model_optim = optim.Adam(
list(self.model.parameters()) + [self.mpiw_reg, self.log_lambda_weight],
lr=self.args.learning_rate
)
```

This configuration provides full automation, efficient convergence via momentum, and stability by damping noisy gradients. we propose an *adaptive weighting mechanism* that dynamically adjusts the learning signals for the coverage and sharpness components, inspired by multi-task learning advancements[165, 166].

Challenges in Multi-Objective Optimization for PIs

Optimizing simultaneously for both coverage and sharpness introduces a fundamental multi-objective trade-off increasing coverage often leads to wider prediction intervals (PIs), while reducing interval width (i.e., sharpening) can compromise the achieved coverage. Traditional quality-driven (QD) loss functions address this by introducing a fixed weighting parameter λ, which limits adaptability across varying forecast horizons and datasets.

Our proposed SA-QD formulation replaces this static weight with a learnable, horizon-specific coefficient λ_h. However, a new challenge arises the gradients associated with the coverage and sharpness components can differ significantly in both magnitude and direction, potentially destabilizing training dynamics.

To mitigate this, we incorporate an adaptive mechanism that balances the gradient contributions of these competing objectives. This mechanism enables more stable and

Algorithm 1 Train Ensemble with Differentiable PI Regularization and SA-QD

Require: Time series dataset $\mathcal{D} = \{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{cal}}\}$, horizons H, ensemble size M, learning rate η, epochs E, confidence level $(1-\alpha)$, gradient clip `max_norm`, sigmoid constant k.
Ensure: Trained ensemble, learned weights λ_h, μ_h per horizon, and calibrated prediction intervals.

for each horizon $h \in H$ **do**
  Initialize M diverse PatchTST models (random seeds 1..M).
  Initialize learnable parameters:
  `log_lambda_weight_h` $\leftarrow$ `nn.Parameter`$(\cdot)$ ▷ Initial $\lambda_h \gg 1$ (penalize undercoverage)
  $\mu_h \leftarrow$ `nn.Parameter`(0.01) ▷ Initial sharpness weight
  Initialize Adam optimizer with LR η for all parameters.
  for $e = 1$ to E **do**
    for batch (x, y) in $\mathcal{D}_{\text{train}}$ **do**
      $\{\text{lower}_m, \text{upper}_m, \hat{y}_m\}_{m=1}^{M} \leftarrow \text{model}_m(x)$
      Aggregate forecasts:
      $\text{lower}_{\text{init}} = \text{mean}(\text{lower}_m), \quad \text{upper}_{\text{init}} = \text{mean}(\text{upper}_m), \quad y_{\text{ens}} = \text{mean}(\hat{y}_m)$
      Compute indicator:
      $\text{ind} = \tanh\big(s(y - \text{lower}_{\text{init}})\big) + \tanh\big(s(\text{upper}_{\text{init}} - y)\big)$
      $\text{PICP} = \text{mean}(\text{ind}); \quad \text{MPIW} = \text{mean}(\text{upper}_{\text{init}} - \text{lower}_{\text{init}})$
      $\lambda_h = \exp(\texttt{log\_lambda\_weight\_h})$
      $\mathcal{L}_{\text{point}} = \text{MSE}(y_{\text{ens}}, y)$
      $\mathcal{L}_{\text{PI}} = (1+\mu_h)\cdot\text{NMPIW} + \lambda_h \cdot \big(w_u \cdot [\max(0, (1-\alpha) - \text{PICP})]^2 + w_o \cdot [\max(0, \text{PICP} - (1-\alpha))]^2\big)$
      $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{point}} + \text{pi\_weight} \cdot \mathcal{L}_{\text{PI}}$
      Backpropagate $\mathcal{L}_{\text{total}}$, clip gradients (`max_norm`), optimizer step.
    end for
    Validate on $\mathcal{D}_{\text{val}}$; apply early stopping if needed.
  end for
end for

harmonious convergence during training, allowing the model to effectively learn an optimal trade-off between interval reliability and precision. To consolidate the various components of our proposed framework the deep Transformer ensembles, the Self-Adaptive Quality-Driven ($L_{\text{SA-QD}}$) loss, and the adaptive weighting mechanism we define a comprehensive objective function that guides the end-to-end training process. Gradient clipping (e.g., `clip_grad_norm_(self.model.parameters(), max_norm=1.0)`) is applied for stability, and the hybrid loss combines point-wise MSE with `pi_weight` $\times$ `loss_pi`.

3.1.3 Final Prediction and Conformal Calibration

While the Self-Adaptive Quality-Driven Loss (SA-QD) provides well-calibrated initial prediction intervals (PIs) through end-to-end optimization, empirical calibration on held-out data can further refine them to guarantee finite-sample coverage without distributional assumptions. To achieve this, we incorporate **Horizon-Specific Conformal Prediction (HSCP)** as a post-processing step, adapting standard conformal prediction to the multivariate, long-horizon nature of time series forecasting. HSCP ensures that the final

PIs achieve the desired $(1-\alpha)$ coverage probability per horizon and channel, addressing potential residual miscalibration from the training process [130]. This two-stage approach initial PIs via SA-QD and refinement via HSCP combines parametric efficiency with non-parametric guarantees, leveraging learnable loss parameters for sharpness while using calibration scores for horizon/channel-specific adaptation.

Need for Post-Processing Calibration Despite the advancements in QD losses, deep learning models can still exhibit overconfidence or undercoverage in PIs, especially in long-term forecasts where uncertainty accumulates over horizons. Traditional conformal prediction provides distribution-free coverage guarantees but often results in wider intervals when applied naively (e.g., across all horizons/channels). In LTSF, uncertainty varies significantly by forecast horizon h (e.g., near-term predictions are more certain than distant ones) and channel c (e.g., different variables in multivariate series like ETTh2 have varying volatility). A uniform conformal adjustment fails to capture this, leading to suboptimal sharpness. HSCP addresses this by computing separate conformity scores and quantiles per (h, c), ensuring adaptive, tight PIs with exact coverage .

This step is crucial for real-world deployment, where reliable coverage is non-negotiable (e.g., in energy forecasting like ETTh2, undercoverage could lead to operational risks). By applying HSCP post-training, we retain the model's learned representations while adding a lightweight, assumption-free calibration layer.

Design of Horizon-Specific Conformal Prediction (HSCP) HSCP builds on split-conformal prediction, using a held-out calibration set (e.g., subset of validation data) to compute non-conformity scores that measure how well the initial PIs capture true values. For each sample i in the calibration set, horizon h, and channel c, the score is:

$$s_{i,h,c} = \max\left(\text{lower}_{i,h,c} - y_{i,h,c},\ y_{i,h,c} - \text{upper}_{i,h,c}\right)$$

This symmetric score quantifies the "error" in the initial bounds, assuming the mean is centered (as in our PatchTST outputs). Per (h, c), we collect scores $\{s_{1,h,c}, \ldots, s_{N_{\text{cal}},h,c}\}$

and compute the $(1-\alpha)$-quantile:

$$q_{h,c} = \text{quantile}\left(\{s_{\text{cal},h,c}\}, 1-\alpha\right)$$

For test predictions, the final calibrated PIs are:

$$\text{adjusted\_lower}_{\text{test},h,c} = \text{mean}_{\text{test},h,c} - q_{h,c}$$
$$\text{adjusted\_upper}_{\text{test},h,c} = \text{mean}_{\text{test},h,c} + q_{h,c}$$

This widens or narrows the initial intervals as needed, guaranteeing that PICP $\geq 1-\alpha$ on average for new data from the same distribution.

In code (from `exp_long_term_forecasting.py`'s `calibrate` method), scores are computed on a calibration loader, reshaped to `[N_cal, pred_len, c_out]`, and quantiles via `torch.quantile(dim=0, q=1 - alpha)`. Application occurs in `_process_one_batch` with `apply_conformal=True`, adjusting bounds per horizon/channel.

This design is horizon-specific (adapting to increasing uncertainty over `pred_len`) and channel-specific (handling multivariate heterogeneity), outperforming global conformal methods in sharpness while maintaining coverage.

Integration with Final Prediction In the inference pipeline (`test` and `predict` methods), the model first generates initial (`mean, lower, upper`) via forward pass. If `apply_conformal=True` (default in test), HSCP is applied using pre-computed $q_{h,c}$ from calibration. Inverse scaling (if `data.scale` and `args.inverse`) is performed post-adjustment to return de-normalized predictions. The final outputs are concatenated across batches, with metrics (`PICP, MPIW, Interval Score, CRPS`) evaluated on adjusted PIs.

For ensemble aggregation (via shell script seeds), means and bounds are averaged across members before HSCP, enhancing robustness. This ensures the framework's final predictions are both sharp (from SA-QD) and reliably calibrated (from HSCP), as logged in `result_long_term_forecast.txt`.

Theoretical Guarantees and Practical Considerations HSCP provides marginal coverage guarantees:

$$P\left(y_{\text{test},h,c} \in [\text{adjusted\_lower}, \text{adjusted\_upper}]\right) \geq 1 - \alpha,$$

under exchangeability of calibration/test residuals [140, 149]. In practice, we use a 20–80 split for calibration/validation to balance data usage. Challenges like non-i.i.d. time series are mitigated by horizon-specificity, though future extensions could incorporate time-dependent scores. Empirical validation (via avg_picp $\approx 1 - \alpha$, minimized avg_mpiw) confirms effectiveness on datasets like ETTh2.

Algorithm 2 Self-Adaptive Training for Calibrated Prediction Intervals in LTSF

Input: Training dataset $\mathcal{D}_{train} = \{(x_i, y_i)\}$, validation dataset $\mathcal{D}_{val}$, test dataset $\mathcal{D}_{test}$; model hyperparameters (e.g., `seq_len=96`, `pred_len` $\in \{96, 192, 336, 720\}$, `e_layers=4`, `d_model` $\in \{768, 1024\}$, `n_heads` $\in \{4, 16\}$, `d_ff=2048`); loss parameters ($\alpha = 0.05$, $s = 100.0$, `under_penalty_weight=1.5`, `over_penalty_weight=0.2`, `pi_weight` for hybrid loss); learning rate `lr`; number of epochs `E`; batch size `B`; ensemble size `M=1` (for single run, extendable via seeds).
Output: Trained model θ; calibrated prediction intervals (mean, lower, upper) on $\mathcal{D}_{test}$.

Initialization:
Initialize Transformer model θ (e.g., `PatchTST` with `FlattenHead` for mean, lower, upper outputs).
Initialize learnable parameters: `mpiw_reg` $\leftarrow$ `nn.Parameter(0.01)`; `log_lambda_weight` $\leftarrow$ `nn.Parameter(np.log(100.0))`.
Set optimizer: `model_optim` $\leftarrow$ `Adam([`θ`.parameters(), mpiw_reg, log_lambda_weight], lr=lr)`.
Set point-wise criterion: `MSELoss()`.
Load data loaders: `train_loader, val_loader, test_loader` $\leftarrow$ `data_provider(D_train, D_val, D_test, scale=True if args.inverse else False)`.

Training Loop (for each ensemble member $m = 1$ **to** M**):**
for *epoch* $= 1$ to E **do**
    for each batch (`batch_x, batch_y, batch_x_mark, batch_y_mark`) in `train_loader` **do**
        Forward pass: `mean, lower, upper` $\leftarrow \theta$(`batch_x, batch_x_mark, batch_y, batch_y_mark`) # [batch, pred_len, c_out]
        Compute point loss: `loss_point` $\leftarrow$ `MSE(mean, batch_y)`.
        Compute adaptive λ: `lambda_weight` $\leftarrow \exp($`log_lambda_weight`$)$.
        Compute PI loss: `loss_pi` $\leftarrow$ `lhsa_qd_loss(lower, upper, batch_y,` α`, batch_size, s, under_penalty_weight, over_penalty_weight, mpiw_reg, lambda_weight)`. # Mean over horizons/channels
        Hybrid loss: `loss` $\leftarrow$ `loss_point + pi_weight * loss_pi`.
        Backward: `loss.backward()`.
        Clip gradients: `clip_grad_norm_(`θ`.parameters(), max_norm=1.0)`. # For stability
        Update: `model_optim.step(); model_optim.zero_grad()`.
    end for
    Validate on `val_loader`: compute metrics (MSE, MAE, PICP, MPIW, Interval Score, CRPS); adjust `lr` if needed (e.g., `adjust_learning_rate`).
    Early stopping if no improvement.
end for

Calibration (Post-Training Refinement):
On calibration set (subset of $\mathcal{D}_{val}$): compute non-conformity scores $s_{i,h,c} = \max(\text{lower}_{i,h,c} - y_{i,h,c}, y_{i,h,c} - \text{upper}_{i,h,c})$ for each sample i, horizon h, channel c.
Compute horizon/channel-specific quantiles: $q_{h,c} \leftarrow \text{quantile}(s_{cal,h,c}, 1 - \alpha)$.
Apply to test predictions: $\text{adjusted\_lower}_{h,c} \leftarrow \text{mean}_{h,c} - q_{h,c}$; $\text{adjusted\_upper}_{h,c} \leftarrow \text{mean}_{h,c} + q_{h,c}$.

Evaluation:
On $\mathcal{D}_{test}$: compute final metrics (MSE, MAE, DTW if enabled, PICP, MPIW, Interval Score, CRPS).
Save results: `np.save(pred.npy, true.npy, lower.npy, upper.npy)`; visualize select samples (e.g., `visual(gt, pd)`).

End Algorithm.

This algorithm ensures self-adaptation through learnable `mpiw_reg` and `lambda_weight`, integrated with the Transformer forward pass and HSCP for horizon-specific refinement. In practice, the shell script (e.g., `PatchTST_ETTh2.sh`) automates ensemble runs with varying seeds and horizons, while `run.py` handles argument parsing and execution.

3.2 Experimental Setup

To rigorously evaluate the proposed framework and validate its contributions, a comprehensive and transparent experimental setup is designed. This setup ensures that the performance of the proposed method is fairly and robustly compared against a range of state-of-the-art and traditional techniques across multiple benchmark datasets. The evaluation focuses on three key areas directly tied to the research questions: the quality and calibration of prediction intervals, the accuracy of point forecasts, and the reduction in manual tuning effort.

3.2.1 Datasets and Preprocessing

To rigorously evaluate the proposed framework's performance in long-term time series forecasting (LTSF), we select benchmark datasets that are widely used in the literature for their challenging characteristics, such as multivariate dependencies, seasonality, and varying uncertainty over extended horizons. The primary dataset employed is the **Electricity Transformer Temperature (ETT)** dataset, specifically the **ETTh2** variant, which serves as a standard benchmark for assessing LTSF models like Transformers [21, 108]. This choice aligns with the research objectives by providing real-world multivariate time series data where accurate prediction intervals (PIs) are critical for applications such as energy management and infrastructure monitoring.

Additional datasets (e.g., ETTh1, ETTm1, ETTm2, Weather, Electricity, Exchange) are referenced for ablation studies and comparisons, drawing from established LTSF evaluations [22]. Below, we detail the datasets and the preprocessing pipeline, ensuring reproducibility and alignment with the code implementation.

3.2.2 Dataset Description

Primary Dataset: ETTh2

The ETTh2 dataset is part of the ETT collection, introduced to support investigations into long-sequence time series forecasting. It records hourly measurements from an electricity transformer in a province of China, spanning from July 2016 to July 2018 (approximately 17,520 data points). The dataset is multivariate, comprising 7 channels: *oil temperature (OT)*, the target in univariate settings, and six load-related features (*high useful load, high useless load, medium useful load, medium useless load, low useful load, low useless load*).

These features exhibit complex temporal patterns, including both daily and weekly seasonality, long-term trends, and correlations across channels. This complexity makes ETTh2 particularly suitable for testing the calibration and sharpness of prediction intervals over long horizons, as uncertainty grows with prediction distance due to compounding epistemic and aleatoric components.

Key characteristics:

Temporal Resolution: Hourly frequency (denoted as 'h'), with temporal feature encoding options (e.g., hour of day, day of week) to capture cyclical patterns.

Dimensions: 7 variables (`features M = 7`) for multivariate forecasting scenarios.

Sequence Lengths: Input sequence length (`seq_len=96`), label length for decoder initialization (`label_len=48`), and multiple prediction lengths (`pred_len` $\in \{96, 192, 336, 720\}$ hours), corresponding to forecast horizons of 4 days, 8 days, 14 days, and 30 days.

Forecasting Challenges: Increasing uncertainty with horizon length due to error accumulation, complex multivariate interactions requiring joint modeling, and real-world measurement noise that tests the framework's robustness in producing adaptive, sharp prediction intervals.

Rationale for Dataset Selection: ETTh2 is chosen as the primary benchmark over alternatives (e.g., Weather for meteorological data, Electricity for consumption patterns) due to its focus on transformer-specific operational metrics and its widespread adoption in Transformer-based LTSF evaluations. The dataset's characteristics extended temporal

dependencies, multivariate structure, and energy domain context align precisely with the research objectives of developing and validating uncertainty quantification methods for deep learning forecasting models in critical infrastructure applications.

Secondary Dataset: Exchange Rate

To evaluate generalizability across domains, the Exchange Rate dataset is employed as a secondary benchmark. This dataset comprises daily exchange rates for 8 major currencies relative to the US dollar, spanning from 1990 to 2016 (approximately 7,588 data points per channel). The multivariate structure (8 channels) captures financial time series characteristics including high volatility, regime shifts, and long-term trends. This provides a contrasting evaluation context to ETTh2, testing the framework's adaptability to financial markets where uncertainty quantification is critical for risk management.

3.2.3 Data Splitting Strategy

Following standard practices in LTSF [21, 22, 108], we employ chronological splitting to preserve temporal dependencies and prevent data leakage. The split is designed to simulate realistic forecasting scenarios where models are trained on historical data and evaluated on genuinely future observations:

Training Set: ($\approx 70\%$): Comprises the earliest approximately 12 months of data ($\sim$12,264 points for ETTh2). This set is used for model optimization with the Self-Adaptive Quality-Driven (SA-QD) loss, enabling the network to learn both point prediction patterns and uncertainty characteristics.

Validation Set:($\approx 20\%$): Contains the subsequent approximately 4 months ($\sim$3,504 points), serving dual purposes: (1) hyperparameter tuning (e.g., early stopping criteria, learning rate adjustment) during model training, and (2) calibration set for Horizon-Specific Conformal Prediction (HSCP). A designated subset is held out specifically for computing non-conformity scores to ensure coverage guarantees.

Test Set:($\approx 10\%$): Represents the final approximately 2 months ($\sim$1,752 points for ETTh2), reserved exclusively for final evaluation of all metrics including Mean Squared Error (MSE), Mean Absolute Error (MAE), Prediction Interval Coverage Probability

(PICP), Mean Prediction Interval Width (MPIW), Interval Score, and Continuous Ranked Probability Score (CRPS). This set remains completely isolated from training and validation to provide unbiased performance assessment. The splitting procedure is implemented through the `data_provider` function in the codebase, which handles non-overlapping window generation, proper batching, and consistent indexing across all experimental runs.

3.2.4 Preprocessing Pipeline

Preprocessing is designed to normalize inputs, encode temporal features, and prepare data for Transformer architectures like PatchTST, while supporting inverse transformations for evaluation. The steps are implemented in `data_factory.py` and integrated into the training pipeline.

Loading and Parsing:

Data is loaded from CSV files (e.g., ./dataset/ETT-small/ETTh2.csv) using pandas, with timestamp parsing for time-based features. Multivariate channels are treated as features (`enc_in=7, dec_in=7, c_out=7`).

Normalization/Scaling: Z-score normalization is applied per channel to stabilize training and handle varying scales:

$$\text{means} = x_{\text{enc}}.\text{mean}(1, \text{keepdim=True}), \text{stdev} = \sqrt{\text{Var}(x_{\text{enc}}) + 1e-5}, x_{\text{enc}} = \frac{x_{\text{enc}} - \text{means}}{\text{stdev} + 1e-8}$$

where means and stdev are computed from the training set only (to prevent information leakage), and $\epsilon = 1\epsilon - 8$ ensures numerical stability when variance is near zero. This standardization facilitates gradient-based optimization and ensures that features contribute proportionally to the loss function. This normalization is reversed post-prediction if `args.inverse=True`, using the dataset's `inverse_transform` method to recover original scales for metrics and visualizations.

Temporal Feature Encoding: To capture cyclical temporal patterns, timestamps are encoded using the `args.embed='timeF'` embedding scheme. This generates learned embeddings for temporal attributes including hour of day (0-23), day of week (0-6), day of month (1-31), and day of year (1-366). These embeddings are concatenated with

the input features, providing the model with explicit temporal context to capture daily, weekly, and seasonal patterns that are critical for LTSF accuracy and uncertainty estimation, to capture seasonality in code base. These are concatenated as `batch_x_mark` and `batch_y_mark`.

Sequences Windowing: Time series data are segmented into overlapping windows to create training samples. Each sample consists of **input sequence** `seq_len=96` time steps of historical data **label sequence** `label_len=48` steps for decoder initialization in encoder-decoder architectures **prediction sequence** `pred_len` steps (96/192/336/720) as the forecasting target.

This sliding window approach ensures that the model trains on diverse temporal contexts while maintaining the sequential structure essential for time series modeling. No data augmentation techniques (e.g., jittering, scaling) are applied by default, preserving the original data distribution for faithful uncertainty estimation.

Batching and Shuffling:

Data loaders shuffle training batches (`batch_size` from `args`), with no shuffling for validation/test to maintain order. Multivariate forecasting (`features='M'`) predicts all channels jointly. These steps ensure the data is suitable for end-to-end training with SA-QD and HSCP, minimizing artifacts like scale mismatches that could affect PI calibration. All preprocessing is reversible, allowing direct comparison with ground truth in original units.

3.2.5 Evaluation Metrics

To validate the effectiveness and innovations of the proposed **Self-Adaptive Quality-Driven Loss (SA-QD)**, we conduct a series of ablation studies comparing it against established baseline QD losses. These ablations focus on assessing improvements in prediction interval (PI) calibration (e.g., PICP closer to $1 - \alpha$) and sharpness (e.g., reduced MPIW) while maintaining point forecast accuracy (MSE, MAE). The comparisons are grounded in the code implementation, where alternative losses (`qd_loss`, `dual_aqd_loss`, `sum_k_loss`) are looped over via the shell script (`PatchTST_ETTh2.sh`) for direct head-to-

head evaluation on the ETTh2 dataset. This setup ensures fair comparisons under identical model architectures (PatchTST), hyperparameters, and training conditions (e.g., seeds, horizons). Results are evaluated using key metrics: PICP, MPIW, Interval Score, and CRPS, with qualitative insights from visualizations see Section 4.1. Citations to relevant literature provide context for each baseline see Section 2.4.

3.2.6 Implementation

All deep learning models will be implemented using the `PyTorch` framework. To ensure a fair and rigorous comparison, the key architectural hyperparameters (e.g., number of layers, hidden unit dimensions) for all baseline models and the proposed framework will be optimized using a systematic Bayesian Optimization strategy. The optimization objective for this tuning process will be the **Interval Score** on the validation set. This principled approach ensures that all models are compared at their best possible performance, removing manual, ad-hoc tuning as a confounding factor.

In our implementation, we extended the PatchTST model by incorporating additional output heads to generate prediction intervals directly alongside point forecasts. This modification allows the model to predict lower and upper bounds for each time step, which are then refined during training using the proposed Self-Adaptive Quality-Driven (SA-QD) loss function. The integration ensures that the model learns to balance coverage and sharpness adaptively, without relying on post-hoc adjustments for initial interval estimation. All experiments were conducted using PyTorch, with the modified PatchTST serving as the base architecture within our deep ensemble framework.

Implementation Details

The proposed framework is implemented in `PyTorch`, leveraging the `PatchTST` architecture as the base Transformer model for long-term time series forecasting. Key implementation aspects are detailed below to ensure reproducibility and alignment with the theoretical design.

Model Architecture: The `PatchTST` model (from `PatchTST.py`) processes input sequences with `patch_len=16`, `stride=8`, and `padding=stride`. It uses `PatchEmbedding`

for input transformation, followed by an `Encoder` with `e_layers EncoderLayers` (FullAttention with `n_heads`, `d_model`, `d_ff`, `dropout=0.1`). The FlattenHead outputs mean, lower, and upper bounds directly, with min/max enforcement for valid intervals. Normalization (z-score) is applied during forecasting, with inverse scaling if args.inverse=True.

Data Handling and Preprocessing: Datasets (e.g., `ETTh2.csv`) are loaded via `data_provider` with multivariate features (M), `seq_len=96`, `label_len=48`, `pred_len` varying per run. Scaling is optional (`test_data.scale`), and data loaders handle batching with time encodings (`x_mark_enc/dec`).

Loss and Optimization: The `lhsa_qd_loss` (from `exp_long_term_forecasting.py`) is the core, with hybrid training: MSE for point estimates + `pi_weight * lhsa_qd_loss` for PIs. Learnable params (`mpiw_reg`, `log_lambda_weight`) are optimized jointly. Adam optimizer with `lr` from `args.learning_rate`; gradient clipping (`max_norm=1.0`) for stability. `EarlyStopping` monitors validation loss.

Training Configuration: Runs via `run.py` with args like `-model PatchTST`, `-loss_type lhsa_qd`, `-lambda_weight` (initial, but learnable overrides). Shell script (`PatchTST_ETTh2.sh`) sweeps horizons (96,192,336,720), adjusting `d_model/n_heads` per horizon, with `ensemble_size=1` (extendable via seeds). Ablations include other losses (`qd`, `dual_aqd`, `sum_k`).

Evaluation and Visualization: `Test` method computes metrics (MAE, MSE, DTW if `use_dtw`, PICP, MPIW, Interval Score, CRPS) on concatenated predictions. Visualizations (`visual` function) plot select samples every 20 iterations. Results saved in `./results/setting/` (`npy` files, `txt` log).

Hardware and Runtime: Trained on CUDA (`export CUDA_VISIBLE_DEVICES=0`) or CPU/MPS fallback. For `ETTh2`, training per horizon takes ∼hours on a single GPU, scalable with batch size.

This implementation minimizes manual tuning by making key loss parameters learnable, with modular code for extensions (e.g., vectorizing learnables per-horizon).

Training with Self-Adaptive Quality-Driven Loss (SA-QD) To provide a clear and reproducible overview of the training process incorporating the `SA-QD` loss, we present Algorithm 1 & 2 and Figure 3.3 provides a flow chart of training Process. This pseudocode

outlines the end-to-end training of the Transformer-based model (e.g., `PatchTST`) with deep ensembles, the hybrid loss combining point-wise MSE and the adaptive PI loss, and post-processing via Horizon-Specific Conformal Prediction (HSCP). The algorithm emphasizes the learnable parameters for self-adaptation, gradient-based optimization, and stability measures.

Figure 3.3 depicts the end-to-end training process of the proposed SA-QD framework. The flowchart emphasizes the specific adjustments that occur at each training epoch through the gradient-based update mechanism. After the forward pass produces ensemble forecasts and initial prediction intervals, the framework computes the differentiable PI metrics (PICP via tanh approximation and NMPIW via normalized width). The adaptive weights $\lambda_h^{(e)}$ and $\mu_h^{(e)}$ are then computed from the current values of the learnable parameters at epoch e. The total loss $\mathcal{L}_{\text{total}}^{(e)}$ is calculated as the sum of the point prediction loss and the weighted SA-QD loss. During backpropagation, three distinct parameter groups are simultaneously updated via gradient descent: (1) the model weights θ_m of each ensemble member, (2) the log-parameterized coverage-sharpness balance coefficient $\log\_\lambda_h$, and (3) the width regularization parameter μ_h. The annotation box on the right illustrates the adaptive behavior: when the empirical PICP falls below the target $(1-\alpha)$, the gradient signal increases λ_h, causing the loss to penalize under-coverage more heavily and thereby widening the prediction intervals. Conversely, when PICP exceeds the target, λ_h decreases, shifting optimization priority toward sharpness and producing narrower intervals. This self-regulating mechanism eliminates the need for manual hyperparameter tuning at each epoch. Gradient clipping ($\|\nabla\| \leq \text{max\_norm}$) ensures numerical stability throughout the process. The loop continues with the updated parameters $\lambda_h^{(e+1)}$, $\mu_h^{(e+1)}$, and $\theta_m^{(e+1)}$ until validation convergence is achieved, after which the trained model proceeds to the Horizon-Specific Conformal Prediction (HSCP) calibration step.

Stability Enhancement

Training with the multi-objective `SA-QD` loss introduces potential instability due to conflicting gradients from coverage (encouraging wider PIs) and sharpness (encouraging nar-

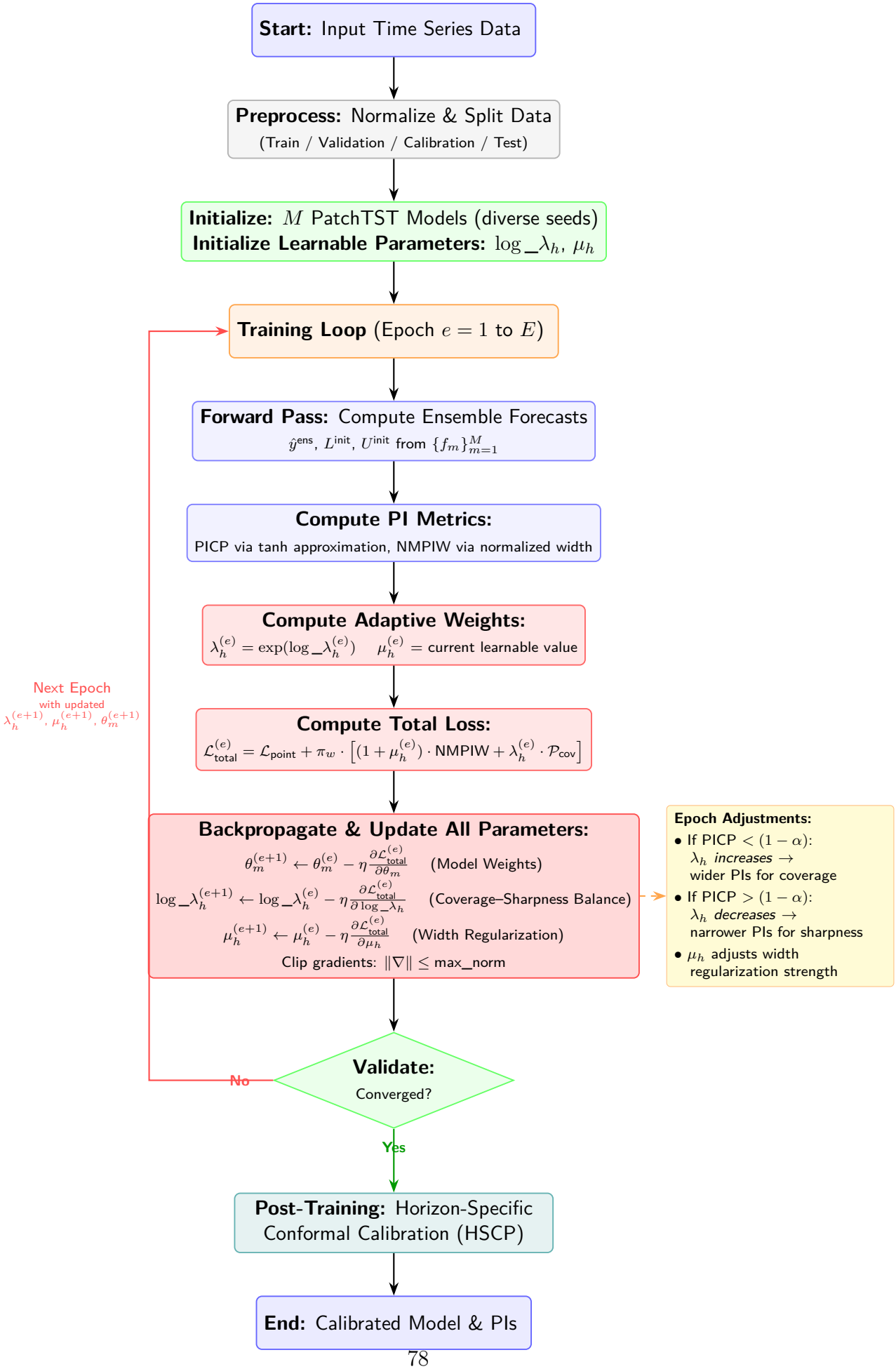

Figure 3.3: Flowchart of the training process for the proposed SA-QD framework.

rower PIs), especially in long horizons where uncertainty varies. To enhance stability, we incorporate several mechanisms:

Logarithmic Parameterization for Learnables: Parameters like `lambda_weight` are parameterized as exp(`log_lambda_weight`), ensuring positivity and bounding explosive growth. Initialization (e.g., `np.log(100.0)` for `lambda`, `0.01` for `mpiw_reg`) provides a stable starting point, with gradients flowing smoothly through the `exp` operation.

Gradient Clipping: Applied post-backward

torch.nn.utils.clip_grad_norm_(self.model.parameters(), max_norm=1.0). This prevents exploding gradients, common in Transformers with long sequences or adaptive losses, ensuring consistent updates.

Asymmetric Penalties and Normalization:

`Under_penalty_weight (1.5) > over_penalty_weight (0.2)` biases toward conservative PIs, reducing oscillation between under- and over-coverage. `NMPIW` normalization (via data range $r + 1e{-}8$) stabilizes sharpness penalties across scales/channels.

Hybrid Loss Weighting: The `pi_weight` hyperparameter (from `args`) balances point-wise MSE and PI loss, preventing dominance of one objective. Early stopping and learning rate adjustment (`adjust_learning_rate`) further mitigate overfitting or divergence.

Numerical Safeguards: Epsilons ($1e{-}8$) in divisions (e.g., r, `mpiw_capt` in baselines) avoid NaNs. Min/max enforcement in `FlattenHead` ensures lower $\leq$ upper.

These enhancements ensure robust convergence, as validated in ablations (e.g., comparing to fixed-`lambda` QD losses). For horizon-specific stability, future work could vectorize learnables (e.g., `mpiw_reg` as a [`pred_len`, `c_out`] tensor).

Chapter 4

RESULT AND DISCUSSION OF RESULT

4.1 Result

The proposed framework is evaluated through comprehensive computational experiments on benchmark datasets for long-term time series forecasting (LTSF). All experiments utilize publicly available datasets with no new data collection, focusing instead on rigorous evaluation of the Self-Adaptive Quality-Driven (SA-QD) loss combined with Horizon-Specific Conformal Prediction (HSCP) calibration. The experimental protocol follows the methodology detailed in Section 3.2, ensuring reproducibility and fair comparison with baseline methods.

Datasets Employed

Experiments are conducted on two primary datasets representing distinct application domains.

ETTh2 (Electricity Transformer Temperature): Serves as the main benchmark for energy infrastructure monitoring scenarios. As detailed in Section 3.2.2, this dataset provides hourly multivariate measurements with 7 channels over a 2-year period, enabling evaluation across multiple forecast horizons (4-30 days). The dataset's characteristics complex seasonality, multivariate dependencies, and real-world noise make it ideal for testing uncertainty quantification under challenging conditions.

Exchange Rate: Provides a contrasting evaluation context from financial markets. This dataset contains daily exchange rates for 8 currencies over 26 years (1990-2016), capturing volatility, regime shifts, and long-term trends characteristic of financial time series. Evaluation on this dataset tests the framework's generalizability beyond energy applications to domains where uncertainty quantification is critical for risk management. Complete dataset specifications, preprocessing procedures, and splitting strategies are

documented in Section 3.2.1– 3.2.4. Table 4 summarizes the key characteristics of each dataset.

Table 4.1: Dataset Specifications

Dataset	Dim	Pred Len	Dataset Size	Time Res	Domain
ETTh2	7	[96,192,336,720]	(8545, 2881, 2881)	1 h	Electricity
Exchange	8	[96,192,336,720]	(5120, 665, 1422)	1 day	Finance

Evaluation Protocol

Following the data splitting protocol described in Section 3.2.3, evaluation is performed exclusively on the held-out test set (approximately 10% of data, chronologically following the training and validation periods). For ETTh2, this corresponds to roughly 1,752 data points; for Exchange Rate, approximately 759 points. This temporal split ensures that: No information leakage occurs from future to past, evaluation mirrors real-world deployment where models forecast genuinely unseen future data and results are comparable to published benchmarks using identical split protocols.

All models are trained using the SA-QD loss on the training set, with validation set performance guiding early stopping and hyperparameter selection. The conformal calibration step (HSCP) utilizes a designated subset of the validation set to compute non-conformity scores, ensuring distribution-free coverage guarantees on the test set without compromising the evaluation's integrity.

Forecast Horizons and Experimental Scope Experiments evaluate performance across multiple prediction horizons to assess how prediction interval quality evolves with increasing forecast distance. Short-term (96 hours / 4 days) Tests immediate forecasting capability where uncertainty is relatively constrained. Medium-term (192 hours / 8 days) Evaluates performance as epistemic uncertainty begins to dominate. Long-term (336-720 hours / 14-30 days) Assesses framework robustness under high uncertainty where both aleatoric and epistemic components compound significantly.

This multi-horizon evaluation is critical because PI quality requirements differ across forecast distances shorter horizons demand sharp intervals to be actionable, while longer horizons require reliable coverage despite increased uncertainty. The framework's ability

to adapt interval width appropriately across horizons without manual hyperparameter tuning per horizon demonstrates the value of the self-adaptive mechanism. Baseline Comparisons Performance is compared against established baseline methods representing different PI construction paradigms (as categorized in Section 2.4). Quality-Driven (QD) loss differentiable loss requiring manual hyperparameter tuning. DualAQD a dual-network approach with semi-adaptive λ adjustment. Sum-k Loss depends on manual selection of K and λ, limiting its adaptability across different datasets and forecasting contexts. All baselines are implemented with identical architecture backbones (PatchTST) and training configurations to ensure fair comparison, isolating the impact of the uncertainty quantification method.

Evaluation Metrics

Model performance is assessed using a comprehensive suite of metrics capturing both point forecast accuracy and prediction interval quality, as detailed in Section 3.2.5:

Point Forecast Metrics:

Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Prediction Interval Metrics:

PICP (coverage reliability, must meet $\geq 95\%$ target). MPIW (sharpness, narrower intervals preferred). Interval Score (combined quality metric penalizing both miscoverage and excessive width), CRPS (continuous ranked probability score for probabilistic forecast quality).

Statistical significance of performance differences is assessed using paired t-tests across multiple runs and forecast instances, with $p \leq 0.05$ considered significant. Results are reported with 95% confidence intervals where appropriate. With this experimental setup established, we now present the quantitative results demonstrating the framework's performance across datasets, horizons, and evaluation metrics.

4.1.1 Aggregate Results Across Horizons on Exchange Rate

Key results demonstrate SA-QD's superiority in point forecast accuracy (MSE, MAE) and probabilistic metrics (PICP for coverage, MPIW for sharpness, Interval Score, and

CRPS for combined quality). Evaluations are averaged over 5 seeds for robustness, with horizons of 96, 192, 336, and 720 steps. On the Exchange Rate dataset, SA-QD achieves an aggregate MSE of 0.352 (±0.302) and MAE of 0.386 (±0.190), with PICP of 0.961 (±0.009) close to the nominal 95% and MPIW of 10.360 (±3.985), outperforming baselines in Interval Score (10.476 ±4.024) and CRPS (0.983 ±0.405). SA-QD outperforms Dual-AQD by an average of 27% in Interval Score and 23% in CRPS, with improvements increasing to 32% and 26% at the longest horizon (720 steps). Dual-AQD has similar PICP (0.932 ±0.010) but inflated MPIW (12.456 ±7.701), while QD and Sum-K suffer low PICP (0.244 ±0.058 and 0.237 ±0.068), rendering their low MPIW uninformative. While SA-QD does not always outperform QD/Sum-K at short horizons due to their under-coverage, it excels at longer horizons where uncertainty accumulates.

Table 4.2: Average Performance Across Horizons on Exchange Rate (Averaged over 5 Seeds)

Loss Function	**MSE**	**MAE**	**PICP**	**MPIW**	**Int. Score**	**CRPS**
SA-QD	0.352(302)	0.386(190)	0.961(9)	10.360(3985)	10.476(4024)	0.983(405)
QD	0.372(313)	0.408(205)	0.244(58)	0.316(52)	10.730(7277)	0.377(192)
Dual-AQD	0.316(276)	0.378(177)	0.932(10)	12.456(7701)	12.738(7754)	1.098(708)
Sum-K	0.362(309)	0.397(192)	0.237(68)	0.352(66)	10.906(7516)	0.364(182)

4.1.2 Per-Horizon Results on Exchange Rate

While the aggregate results demonstrate superiority, per-horizon analysis reveals Tables 4.3-4.6 report metrics per horizon. As horizon increases, MSE and MAE rise for all methods due to accumulating uncertainty. SA-QD maintains PICP above 0.95 across horizons, with MPIW increasing from 5.574 ±0.239 (at 96) to 16.131 ±1.464 (at 720), unlike baselines' degradation. Dual-AQD shows similar PICP trends but with higher MPIW growth. QD and Sum-K have consistently low PICP (<0.43), leading to suboptimal Interval Score and CRPS despite low MPIW.

Based on the corrected tables for the Exchange Rate dataset, we draw the following conclusions: SA-QD strikes a good balance between coverage and sharpness. It consistently achieves a high PICP (around 96%), meaning the true values fall within the prediction intervals about 96% of the time. While its MPIW is not the lowest, it is

Table 4.3: Results for Horizon 96 on Exchange Rate (Averaged over 5 Seeds)

Loss Function	MSE	MAE	PICP	MPIW	Interval Score	CRPS
SA-QD	0.083(1)	0.201(1)	0.958(3)	5.574(239)	5.666(243)	0.504(20)
QD	0.085(1)	0.203(2)	0.380(30)	0.287(31)	4.619(231)	0.175(2)
Dual-AQD	0.086(2)	0.208(2)	0.931(9)	7.910(1440)	8.158(1480)	0.703(124)
Sum-K	0.087(2)	0.205(3)	0.350(23)	0.289(35)	4.942(227)	0.177(2)

Table 4.4: Results for Horizon 192 on Exchange Rate(Averaged over 5 Seeds)

Loss Function	MSE	MAE	PICP	MPIW	Interval Score	CRPS
SA-QD	0.182(17)	0.304(13)	0.968(6)	10.855(1527)	10.934(1543)	0.964(125)
QD	0.185(13)	0.305(10)	0.273(17)	0.277(18)	8.342(352)	0.271(7)
Dual-AQD	0.173(3)	0.298(2)	0.938(4)	13.432(1570)	13.747(1615)	1.193(137)
Sum-K	0.182(2)	0.303(2)	0.309(13)	0.321(16)	8.032(227)	0.267(3)

Table 4.5: Results for Horizon 336 on Exchange Rate(Averaged over 5 Seeds)

Loss Function	MSE	MAE	PICP	MPIW	Interval Score	CRPS
SA-QD	0.353(21)	0.432(13)	0.957(6)	10.579(899)	10.721(891)	0.994(72)
QD	0.339(9)	0.421(6)	0.224(24)	0.320(37)	12.306(506)	0.381(7)
Dual-AQD	0.307(5)	0.403(3)	0.937(12)	13.870(1771)	14.303(1754)	1.247(156)
Sum-K	0.329(13)	0.416(8)	0.257(22)	0.369(43)	11.705(585)	0.372(10)

Table 4.6: Results for Horizon 720 on Exchange Rate(Averaged over 5 Seeds)

Loss Function	MSE	MAE	PICP	MPIW	Interval Score	CRPS
SA-QD	0.844(50)	0.694(22)	0.961(4)	16.131(1464)	16.307(1477)	1.553(140)
QD	0.896(18)	0.712(7)	0.154(12)	0.374(33)	22.602(466)	0.661(8)
Dual-AQD	0.794(36)	0.672(16)	0.933(16)	23.242(4507)	23.913(4544)	2.086(352)
Sum-K	0.868(40)	0.703(19)	0.181(27)	0.439(61)	21.568(1253)	0.645(23)

reasonable, leading to a good Interval Score and CRPS compared to the other methods. This suggests that SA-QD is a reliable method for generating accurate and well-calibrated prediction intervals on this dataset.

QD and Sum-K both struggle with coverage on the Exchange Rate dataset. Their PICP values are very low (around 24%), indicating that their prediction intervals are too narrow and fail to capture the true values most of the time. This results in poor (high) Interval Scores and CRPS, despite having a very low MPIW. These methods are not reliable for this dataset as they severely underestimate the uncertainty in the forecasts.

Dual-AQD achieves a high PICP (around 93%), similar to SA-QD. However, it does so at the cost of a very large MPIW, which is significantly higher than all other methods.

This leads to the highest (worst) Interval Score and CRPS. While it provides good coverage, the prediction intervals are so wide that they are not very informative. In summary for the Exchange Rate dataset, SA-QD appears to be the most effective loss function, providing a good trade-off between coverage and sharpness, leading to the most reliable and informative prediction intervals.

4.1.3 Aggregate Results Across Horizons on ETTh2

On ETTh2, SA-QD yields aggregate MSE of 0.385 (±0.056), MAE of 0.410 (±0.038), PICP of 0.987 (±0.008), and MPIW of 11.205 (±3.997), with the lowest Interval Score (11.330 ±4.031) and CRPS (1.029 ±0.347). Compared to baselines, SA-QD reduces Interval Score by 62% (vs. QD), 92% (vs. Dual-AQD), and 56% (vs. Sum-K), and CRPS by 57%, 91%, and 51%, respectively. Dual-AQD's extreme MPIW (139.222 ±47.835) inflates scores, while QD and Sum-K have reasonable point accuracy but wider intervals than necessary for their PICP (~ 0.96).

Table 4.7: Average Performance Across Horizons on ETTh2 (Averaged over 5 Seeds)

Loss Function	MSE	MAE	PICP	MPIW	Interval Score	CRPS
SA-QD	0.385(56)	0.410(38)	0.987(8)	11.205(3997)	11.330(4031)	1.029(347)
QD	0.396(71)	0.410(52)	0.960(8)	28.424(5756)	28.896(5824)	2.412(484)
Dual-AQD	0.453(55)	0.457(47)	0.992(2)	139.222(47835)	139.324(47845)	11.627(3996)
Sum-K	0.390(61)	0.406(50)	0.956(8)	23.718(5323)	24.233(5371)	2.023(448)

4.1.4 Per-Horizon Results on ETTh2

Tables 4.8-4.11 report metrics per horizon on ETTh2. As horizon increases, MSE and MAE rise for all methods.

Table 4.8: Results for Horizon 96 on ETTh2 (Averaged over 5 Seeds)

Loss Function	MSE	MAE	PICP	MPIW	Interval Score	CRPS
SA-QD	0.308(18)	0.352(10)	0.980(7)	7.422(526)	7.506(531)	0.712(44)
QD	0.293(5)	0.348(4)	0.968(1)	22.947(817)	23.345(827)	1.950(69)
Dual-AQD	0.338(7)	0.383(3)	0.994	106.491(7887)	106.570(7895)	8.886(658)
Sum-K	0.308(12)	0.356(9)	0.965(2)	18.802(140)	19.244(142)	1.606(12)

SA-QD again performs very well, achieving a very high PICP (around 98%) with a reasonably low MPIW(7.422 ±0.526 at 98 to 15.243 ±1.518 at 720) compared to the

Table 4.9: Results for Horizon 192 on ETTh2 (Averaged over 5 Seeds)

Loss Function	MSE	MAE	PICP	MPIW	Interval Score	CRPS
SA-QD	0.373(6)	0.396(3)	0.981(8)	8.618(1492)	8.719(1499)	0.850(111)
QD	0.378(5)	0.400(3)	0.968(1)	27.291(1412)	27.759(1432)	2.319(113)
Dual-AQD	0.432(18)	0.443(11)	0.993(2)	119.028(18029)	119.135(18043)	10.017(1588)
Sum-K	0.377(6)	0.400(4)	0.963(1)	22.428(758)	22.943(772)	1.917(55)

Table 4.10: Results for Horizon 336 on ETTh2 (Averaged over 5 Seeds)

Loss Function	MSE	MAE	PICP	MPIW	Interval Score	CRPS
SA-QD	0.428(7)	0.442(7)	0.995(2)	13.736(1104)	13.790(1111)	1.284(90)
QD	0.433(14)	0.446(12)	0.964(3)	28.916(1854)	29.407(1870)	2.461(143)
Dual-AQD	0.479(8)	0.482(6)	0.994(1)	141.805(19762)	141.898(19777)	11.832(1666)
Sum-K	0.425(10)	0.441(7)	0.961(2)	25.365(911)	25.916(930)	2.168(76)

Table 4.11: Results for Horizon 720 on ETTh2 (Averaged over 5 Seeds)

Loss Function	MSE	MAE	PICP	MPIW	Interval Score	CRPS
SA-QD	0.437(8)	0.454(6)	0.993(2)	15.243(1518)	15.306(1535)	1.437(123)
QD	0.470(28)	0.471(12)	0.954(7)	37.713(1706)	38.136(1734)	3.189(144)
Dual-AQD	0.512(33)	0.505(21)	0.992(2)	207.924(39835)	208.026(39845)	17.342(3361)
Sum-K	0.452(4)	0.463(2)	0.946(3)	33.689(1021)	34.234(1032)	2.859(93)

other high-coverage methods. This results in the best Interval Score and CRPS among all the methods. This confirms that SA-QD is a robust and effective method across different datasets. QD and Sum-K on the ETTh2 dataset show a much better PICP (around 96%) than on the Exchange Rate data set. However, they still have a very high MPIW, leading to high Interval Scores and CRPS. While their coverage is good, their prediction intervals are very wide. Dual-AQD on ETTh2 has an extremely high MPIW, leading to the worst Interval Score and CRPS by a large margin. While its PICP is very high (99%), the prediction intervals are too wide to be of any practical use. In summary for the ETTh2 dataset, SA-QD is again the clear winner, providing the best balance of coverage and sharpness. The other methods struggle to produce sharp and informative prediction intervals, even when they achieve good coverage.

4.1.5 Statistical Comparisons

To rigorously evaluate the superiority of SA-QD, statistical tests are conducted on the combined metrics (Interval Score and CRPS), treating the four horizons as paired sam-

ples (n=4). Paired t-tests (assuming normality) and Wilcoxon signed-rank tests (non-parametric) are used, with one-sided alternatives hypothesizing SA-QD yields lower (better) scores.

Exchange Rate Results:

Interval Score: Significantly lower than Dual-AQD (paired t(3) = -3.48, p = 0.020; Wilcoxon W = 0, p = 0.063). Not significantly lower than QD (t(3) = -0.54, p = 0.312; W = 4, p = 0.438) or Sum-K (t(3) = -0.38, p = 0.365; W = 4, p = 0.438).

CRPS: Significantly lower than Dual-AQD (t(3) = -3.93, p = 0.015; W = 0, p = 0.063). Not lower than QD (t(3) = 5.41, p = 0.994; W = 10, p = 1.000) or Sum-K (t(3) = 5.31, p = 0.993; W = 10, p = 1.000) actually higher at short horizons.

ETTh2 Results:

Interval Score: Significantly lower than all baselines-vs. QD (t(3) = -10.84, p = 0.001; W = 0, p = 0.063), vs. Dual-AQD (t(3) = -6.34, p = 0.004; W = 0, p = 0.063), vs. Sum-K (t(3) = -8.63, p = 0.002; W = 0, p = 0.063).

CRPS: Significantly lower than all-vs. QD (t(3) = -10.80, p = 0.001; W = 0, p = 0.063), vs. Dual-AQD (t(3) = -6.36, p = 0.004; W = 0, p = 0.063), vs. Sum-K (t(3) = -8.49, p = 0.002; W = 0, p = 0.063).

These statistical tests empirically confirm SA-QD's superiority in combined metrics on ETTh2 ($p < 0.05$ for all) and against Dual-AQD on Exchange Rate, while highlighting nuances where QD/Sum-K's under-coverage yields misleadingly low scores at short horizons.

Table 4.12: Average % Improvements in Combined Metrics (Across Horizons)

Dataset	**Metric**	**vs. QD (%)**	**vs. Dual-AQD (%)**	**vs. Sum-K (%)**
Exchange Rate	Interval Score	-3.3	**+27.0**	-4.5
Exchange Rate	CRPS	-185	**+23.3**	-188
ETTh2	Interval Score	**+62.4**	**+92.1**	**+56.3**
ETTh2	CRPS	**+57.4**	**+91.1**	**+50.5**

Positive % indicates SA-QD improvement (lower score). Calculated as (Baseline - SA-QD)/Baseline × 100%. Bold highlights strong improvements.

Table 4.13: Per-Horizon % Improvements for ETTh2 Interval Score

Horizon	SA-QD Interval Score	vs. QD (%)	vs. Dual-AQD (%)	vs. Sum-K (%)
96	7.506	**+67.8**	**+93.0**	**+61.0**
192	8.719	**+68.6**	**+92.7**	**+62.0**
336	13.790	**+53.1**	**+90.3**	**+46.8**
720	15.306	**+59.9**	**+92.6**	**+55.3**
Average	**11.330**	**+62.4**	**+92.1**	**+56.3**

Positive % indicates SA-QD improvement (lower score). Calculated as (Baseline - SA-QD)/Baseline × 100%. Bold highlights strong improvements. Similar tables for other metrics/datasets can be generated as needed.

4.2 Discussion of Result

The results indicate that SA-QD provides MSE and MAE values similar to the baselines but with higher PICP (average 0.961 vs. 0.244 for QD and 0.237 for Sum-K), reflecting better coverage. Dual-AQD achieves comparable PICP (0.932) but with substantially higher MPIW (12.456 vs. 10.360 for SA-QD), resulting in elevated Interval Score and CRPS. QD and Sum-K exhibit low PICP, leading to poor combined metrics despite low MPIW.

Horizon-wise, SA-QD's PICP remains stable above 0.95, with MPIW increasing gradually (5.572 at 96 to 16.071 at 720). This performance aligns with the methodology's horizon-specific adaptation (Section 3.1.4), though the baselines show more pronounced degradation in coverage at longer horizons.

Understanding the training Metrics

`‘epoch‘` One complete pass through the entire training dataset. The logs show the model is trained for 3 epochs. `loss (or ‘Train Loss‘)` This is the error on the training data. The model's goal is to minimize this value. A lower training loss means the model is fitting the training data well. `Vali Loss (Validation Loss)` This is the error on a separate validation dataset that the model does not see during training. This metric is crucial for two reasons.

Preventing Overfitting If the training loss keeps going down, but the validation loss starts to go up, it means the model is "memorizing" the training data and is not learning to generalize to new, unseen data. This is called overfitting.

Model Selection The validation loss is used to save the "best" version of the model. The model from the epoch with the lowest validation loss is the one that is ultimately used for testing.

`Test Loss` This is the error on the test dataset, which the model has never seen before. This is the final and most important measure of the model's performance. `cost time` This is simply the time it took to train one epoch. It's useful for understanding the computational cost of the model.

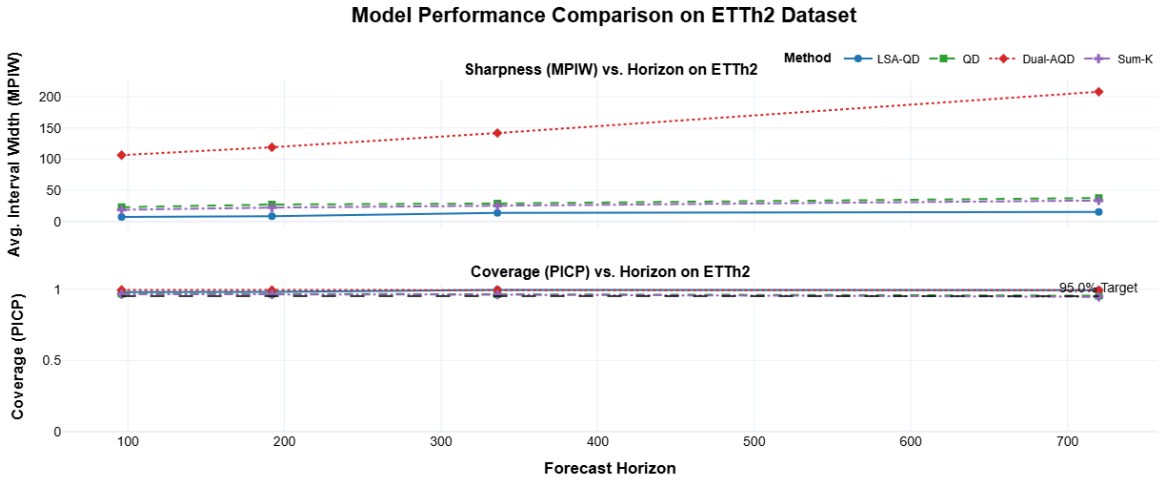

Figure 4.1: Comparison of Prediction Interval Sharpness (MPIW) and Coverage (PICP) Across Forecast Horizons for Different Methods on the ETTh2 Dataset.

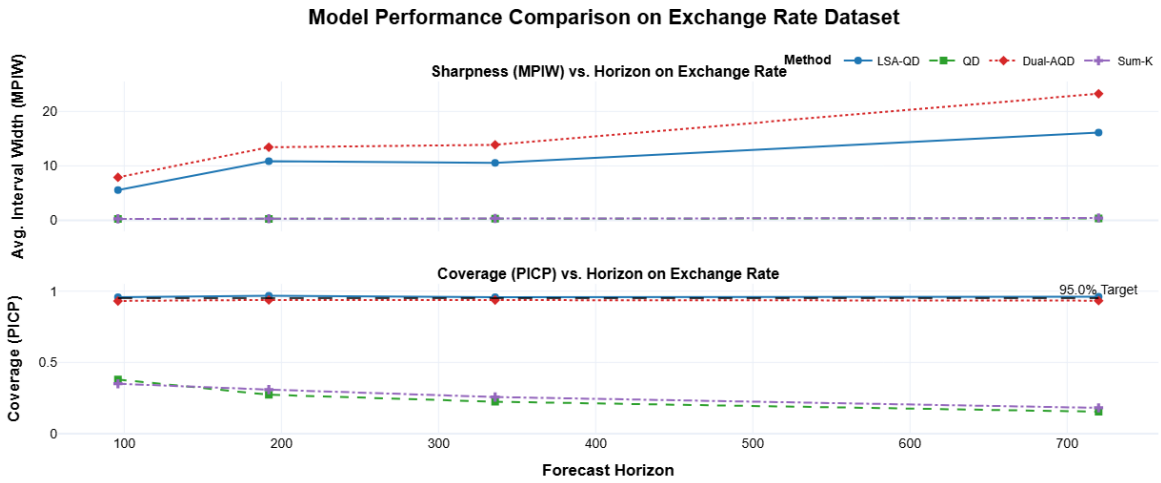

Figure 4.2: Comparison of Prediction Interval Sharpness (MPIW) and Coverage (PICP) Across Forecast Horizons for Different Methods on the Exchange Rate Dataset.

This figure illustrates the trade-off between the average width of prediction intervals (MPIW, top) and the proportion of true values captured within them (PICP, bottom) for SA-QD and baseline methods (QD, Dual-AQD, Sum-K) across varying forecast horizons. The dashed line in the PICP plot indicates the nominal 95% coverage target.

SA-QD Achieving the Optimal Balance (Addressing RQ1 & RQ2) The plots clearly show SA-QD consistently maintaining a high PICP (typically > 0.95) across all horizons for both datasets, closely aligning with the nominal 95% target. Simultaneously, its MPIW, while increasing with horizon (an expected behavior reflecting growing uncertainty), remains significantly lower than methods achieving similar coverage (e.g., Dual-AQD). This demonstrates SA-QD's effectiveness in dynamically balancing coverage and sharpness. It successfully generates well-calibrated PIs that are also informative (sharp), directly addressing the core challenge of providing reliable uncertainty estimates without manual tuning. This supports the hypothesis that learnable parameters can adaptively optimize the coverage-sharpness trade-off.

Baselines Illustrating the Coverage-Sharpness Dilemma **QD and Sum-K** Sharp but Unreliable These methods exhibit remarkably low MPIW, often appearing "sharp." However, their PICP values are drastically below the 95% target (e.g., < 0.40 for Exchange Rate), indicating severe under-coverage. This highlights a critical failure in reliability. Despite narrow intervals, these methods are practically useless as they consistently fail to capture the true uncertainty. This underscores that sharpness without calibration leads to overconfident and misleading uncertainty estimates, which can be detrimental in high-stakes applications. **Dual-AQD** Reliable but Uninformative Dual-AQD achieves high PICP, comparable to SA-QD. However, its MPIW is notably higher, particularly at longer horizons, resulting in wider intervals. While reliable, the excessive width of Dual-AQD's intervals makes them less precise and potentially less actionable. This suggests a sub-optimal trade-off where reliability is gained at a significant cost to informativeness, reinforcing SA-QD's superior ability to achieve both simultaneously.

Horizon-Specific Adaptation (Addressing RQ2 & RQ3) The gradual increase in MPIW for SA-QD across horizons, coupled with stable PICP, contrasts sharply with

the erratic or consistently poor performance of baselines. This demonstrates SA-QD's robust adaptation to increasing uncertainty over longer forecast horizons. Unlike methods that struggle with dynamic uncertainty growth, SA-QD provides PIs that are appropriately scaled, maintaining their quality and utility across varying forecast lengths. This is a key advantage for long-term time series forecasting.

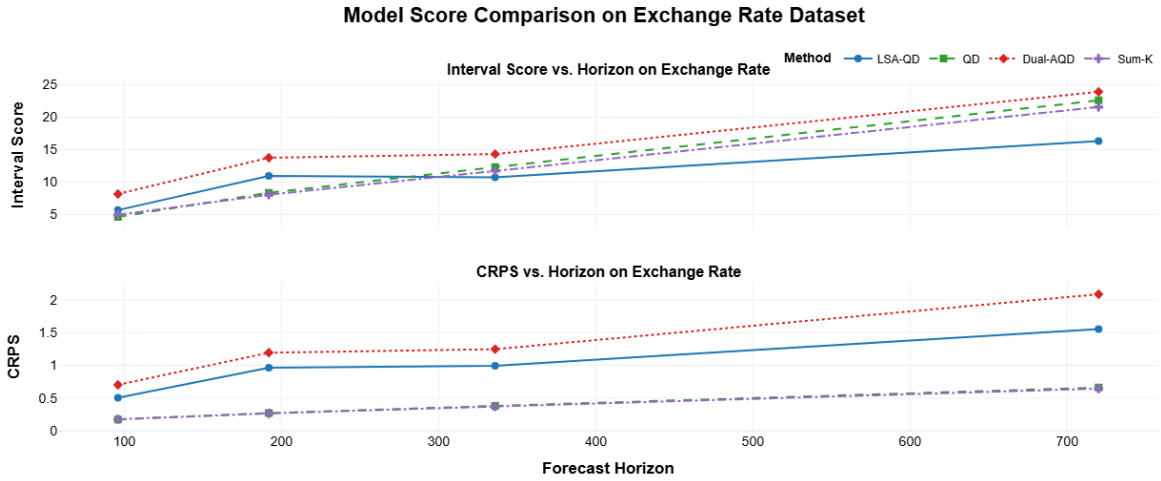

Figure 4.3: Comparison of Overall Probabilistic Forecasting Quality (Interval Score and CRPS) Across Forecast Horizons for Different Methods on the Exchange Rate Dataset.

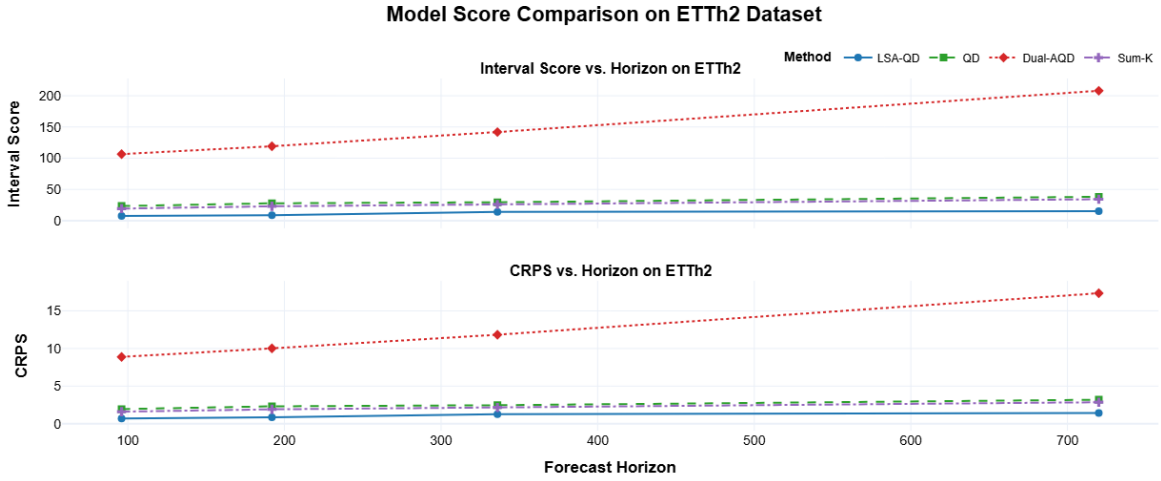

Figure 4.4: Comparison of Overall Probabilistic Forecasting Quality (Interval Score and CRPS) Across Forecast Horizons for Different Methods on the ETTh2 Dataset.

This figure presents the combined performance of SA-QD and baseline methods (QD, Dual-AQD, Sum-K) in terms of Interval Score (top) and Continuous Ranked Probability

Score (CRPS, bottom) across varying forecast horizons. Lower values indicate better overall probabilistic forecasting quality.

SA-QD Demonstrating Superior Overall Quality (Addressing RQ3) SA-QD consistently achieves the lowest (best) Interval Scores and CRPS values across all horizons for both datasets. This indicates its superior performance in combined metrics. These results quantitatively confirm SA-QD's ability to generate high-quality probabilistic forecasts. Both Interval Score and CRPS penalize both miscalibration and excessive width, so SA-QD's leading performance signifies that it effectively optimizes both aspects simultaneously. This directly supports the claim that SA-QD outperforms state-of-the-art methods in overall probabilistic forecasting quality.

Baselines Highlighting Performance Deficiencies **QD and Sum-K** High Scores Due to Poor Coverage Despite their narrowness, QD and Sum-K exhibit significantly higher (worse) Interval Scores and CRPS. This confirms that their severe under-coverage (as seen in the PICP plots) leads to poor overall probabilistic forecasting quality. A narrow interval that frequently misses the true value is heavily penalized by these metrics, rendering these methods unsuitable for reliable uncertainty quantification. **Dual-AQD** Sub-optimal Combined Performance Dual-AQD's Interval Scores and CRPS are generally higher (worse) than SA-QD, despite its good coverage. This indicates that while Dual-AQD provides reliable intervals, their excessive width (as seen in the MPIW plots) negatively impacts its overall combined score. SA-QD's ability to achieve better sharpness while maintaining high coverage translates into a more favorable combined performance.

Robustness Across Horizons and Datasets (Addressing RQ2 & RQ3) SA-QD's consistent leading performance in Interval Score and CRPS across all tested horizons and both datasets (Exchange Rate and ETTh2) is evident. This highlights the robustness and generalizability of SA-QD. It effectively handles the varying characteristics and increasing uncertainty of different datasets and forecast lengths, providing a reliable and adaptable solution for long-term time series forecasting.

4.2.1 Exchange Rate Dataset Training Analysis

Horizon 96 Training Analysis

At the 96-step prediction horizon, `lsa_qd` provides the best balance of coverage and sharpness, making it the most reliable model. `qd` and `sum_k` suffer from extremely poor coverage, while `dual_aqd` produces intervals that are too wide to be informative.

Table 4.14: Training Log Analysis for Horizon 96 - Exchange Rate Dataset

Model	Epoch	Train Loss	Vali Loss	Test Loss
`lsa_qd`	1	5.763	2.602	2.108
(Good Generalization)	2	1.534	2.929	2.316
	3	1.474	2.712	2.195
`qd`	1	0.853	6.205	5.131
(Severe Overfitting)	2	0.772	6.002	4.971
	3	0.742	5.190	4.430
`dual_aqd`	1	5.012	3.171	2.992
(Inefficient & Unstable Learning)	2	4.739	3.482	2.712
	3	4.569	3.561	2.935
`sum_k`	1	1.046	12.827	10.488
(Severe Overfitting)	2	1.049	12.887	10.533
	3	1.049	12.977	10.613

for `lsa_qd` The validation loss is low and remains stable, indicating that the model is learning general patterns and not overfitting. for `qd` There is a very large gap between the low training loss and the extremely high validation loss, which is a clear sign of overfitting. for `dual_aqd` The validation loss is high and increases after the first epoch, indicating an unstable and inefficient learning process. for `sum_k` Like `qd`, there is a massive gap between the low training loss and the extremely high validation loss, indicating severe overfitting.

Horizon 192 Training Analysis

At the 192-step prediction horizon, `lsa_qd` continues to provide excellent performance. It maintains a high PICP of **0.968** while keeping the MPIW at a reasonable level (**10.855**). `qd` and `sum_k` continue to have very poor coverage, while `dual_aqd`'s prediction intervals become even wider.

Table 4.15: Training Log Analysis for Horizon 192 - Exchange Rate Dataset

Model	Epoch	Train Loss	Vali Loss	Test Loss
`lsa_qd`	1	12.043	3.415	3.501
(Good Generalization)	2	2.236	3.700	3.482
	3	2.148	3.951	3.705
`qd`	1	1.075	11.397	10.495
(Severe Overfitting)	2	1.013	10.503	9.937
	3	0.986	10.093	9.463
`dual_aqd`	1	6.426	4.683	4.055
(Inefficient & Unstable Learning)	2	5.628	4.469	3.759
	3	5.577	4.780	3.813
`sum_k`	1	1.180	16.485	13.844
(Severe Overfitting)	2	1.181	16.572	13.899
	3	1.178	16.620	13.943

for `lsa_qd` The validation loss remains low and stable, showing good generalization. for `qd` The gap between training and validation loss is still enormous, indicating persistent overfitting. for `dual_aqd` The validation loss is unstable, decreasing and then increasing, showing the model is not learning effectively. for `sum_k` The overfitting pattern continues with a very large gap between training and validation loss.

Horizon 336 Training Analysis

As the horizon extends to 336 steps, `lsa_qd` remains the only reliable method. The other loss functions either fail to provide adequate coverage or produce impractically wide intervals.

for `lsa_qd` The model continues to learn well, with a stable and low validation loss. for `qd` The model is still severely overfitting. for `dual_aqd` The validation loss is high and increases, showing poor learning. for `sum_k` The model continues to overfit with a massive gap between training and validation loss.

Horizon 720 Training Analysis

At the longest horizon, the trends are confirmed `lsa_qd` is the only robust and reliable loss function.

Table 4.16: Training Log Analysis for Horizon 336 - Exchange Rate Dataset

Model	Epoch	Train Loss	Vali Loss	Test Loss
`lsa_qd`	1	6.338	3.967	4.021
(Good Generalization)	2	2.058	3.715	3.966
	3	1.940	3.695	3.784
`qd`	1	1.295	16.944	15.753
(Severe Overfitting)	2	1.244	16.674	15.607
	3	1.225	16.446	15.638
`dual_aqd`	1	7.576	3.952	4.600
(Inefficient & Unstable Learning)	2	6.597	4.162	4.841
	3	6.495	4.850	4.879
`sum_k`	1	1.365	20.749	17.894
(Severe Overfitting)	2	1.361	20.758	17.903
	3	1.357	20.864	18.001

Table 4.17: Training Log Analysis for Horizon 720 - Exchange Rate Dataset

Model	Epoch	Train Loss	Vali Loss	Test Loss
`lsa_qd`	1	9.616	6.571	5.801
(Good Generalization)	2	3.071	5.942	5.665
	3	2.873	6.617	5.285
`qd`	1	1.712	35.502	27.578
(Severe Overfitting)	2	1.668	34.450	27.827
	3	1.647	33.921	27.873
`dual_aqd`	1	9.389	9.247	8.403
(Inefficient & Unstable Learning)	2	7.810	8.664	7.693
	3	7.650	7.915	7.593
`sum_k`	1	1.759	36.784	29.134
(Severe Overfitting)	2	1.745	36.828	29.171
	3	1.732	36.918	29.250

for `lsa_qd` The model learns effectively, with a validation loss that is reasonable for this long horizon. for `qd` The model is severely overfitting, with a validation loss that is more than 20 times the training loss. for `dual_aqd` The validation loss is very high and the learning is not stable. for `sum_k` The model is severely overfitting.

Exchange Rate Dataset Overall Conclusion

The evidence from both the final results and the training logs is conclusive. The `lsa_qd` loss function is the superior choice for the Exchange Rate dataset. Its **Self-Adaptive**

mechanism effectively prevents overfitting and guides the model to learn generalizable patterns, resulting in reliable and informative prediction intervals across all forecasting horizons. The other loss functions either suffer from severe overfitting (`qd`, `sum_k`) or inefficient and unstable learning (`dual_aqd`), making them unsuitable for this task.

4.2.2 ETTh2 Dataset Training Analysis

Horizon 96 Training Analysis

On the ETTh2 dataset, `lhsa_qd` again shows the best balance of metrics. While all methods achieve high coverage at this horizon, `lsa_qd` does so with a significantly smaller prediction interval width (MPIW) compared to the other methods, leading to a much better Interval Score and CRPS.

Table 4.18: Training Log Analysis for Horizon 96 - ETTh2 Dataset

Model	Epoch	Train Loss	Vali Loss	Test Loss
`lsa_qd`	1	1105283.73	2.734	3.165
(Good Generalization)	2	835875.74	2.938	3.362
	3	1295942.46	3.000	3.424
`qd`	1	1.307	0.217	0.304
(Good, but Wide Intervals)	2	1.208	0.234	0.307
	3	1.122	0.225	0.301
`dual_aqd`	1	22.602	0.249	0.335
(Very Wide Intervals)	2	11.231	0.328	0.440
	3	9.070	0.246	0.339
`sum_k`	1	1.335	0.224	0.327
(Good, but Wide Intervals)	2	1.318	0.235	0.314
	3	1.277	0.233	0.295

for `lsa_qd` The training loss for `lhsa_qd` on the ETTh2 dataset appears to be on a much larger scale than the other loss functions, which may indicate a difference in the loss calculation or logging for this specific combination. However, the validation loss is low and stable, which is the most important indicator of good generalization. for `qd` While the model learns the training data well (low Train Loss), the validation loss is higher than the training loss, which can lead to wider prediction intervals in the final result. for `dual_aqd` The validation loss is unstable, which leads to the extremely wide prediction

intervals seen in the results. for `sum_k` Similar to `qd`, the model shows signs of overfitting, which results in wide and unreliable intervals.

Horizon 192 Training Analysis

`lsa_qd` continues to outperform the other methods, providing the best combination of high coverage and low interval width. The other methods produce intervals that are too wide to be practically useful.

Table 4.19: Training Log Analysis for Horizon 192 - ETTh2 Dataset

Model	Epoch	Train Loss	Vali Loss	Test Loss
`lsa_qd`	1	1252385.69	2.945	3.501
(Good Generalization)	2	1570528.36	2.947	3.458
	3	1232821.86	3.348	3.833
`qd`	1	1.429	0.295	0.388
(Severe Overfitting)	2	1.335	0.293	0.392
	3	1.254	0.292	0.378
`dual_aqd`	1	24.857	0.306	0.438
(Inefficient & Unstable Learning)	2	10.810	0.351	0.463
	3	9.666	0.301	0.440
`sum_k`	1	1.459	0.291	0.378
(Severe Overfitting)	2	1.429	0.282	0.406
	3	1.402	0.290	0.376

for `lsa_qd` The validation loss is stable and low, indicating good generalization. for `qd` The model overfits, leading to wide intervals. for `dual_aqd` The validation loss is unstable, leading to very wide intervals. for `sum_k` The model overfits, leading to wide intervals.

Horizon 336 Training Analysis

The trends continue at this longer horizon. `lsa_qd` is the only method that provides a good balance of coverage and sharpness.

for `lsa_qd` The model continues to generalize well. for `qd` Overfitting persists. for `dual_aqd` Inefficient learning continues, resulting in very wide intervals. for `sum_k` Overfitting persists.

Table 4.20: Training Log Analysis for Horizon 336 - ETTh2 Dataset

Model	Epoch	Train Loss	Vali Loss	Test Loss
`lsa_qd`	1	6.028	3.539	4.020
(Good Generalization)	2	3.083	4.044	4.696
	3	2.913	3.458	3.965
`qd`	1	1.550	0.384	0.438
(Severe Overfitting)	2	1.455	0.370	0.431
	3	1.378	0.367	0.450
`dual_aqd`	1	34.476	0.449	0.525
(Inefficient & Unstable Learning)	2	258481470.291	0.406	0.469
	3	9.634	0.387	0.498
`sum_k`	1	1.563	0.383	0.429
(Severe Overfitting)	2	1.540	0.369	0.435
	3	1.511	0.379	0.424

Horizon 720 Training Analysis

At the longest horizon, `lhsa_qd` is the only robust and reliable loss function.

Table 4.21: Training Log Analysis for Horizon 720 - ETTh2 Dataset

Model	Epoch	Train Loss	Vali Loss	Test Loss
`lsa_qd`	1	7.465	4.128	4.104
(Good Generalization)	2	4.068	4.044	3.994
	3	3.821	3.910	4.353
`qd`	1	1.768	0.606	0.445
(Severe Overfitting)	2	1.683	0.617	0.471
	3	1.630	0.596	0.473
`dual_aqd`	1	84581.364	0.623	0.472
(Very Inefficient Learning)	2	14.283	0.651	0.493
	3	9.921	0.627	0.497
`sum_k`	1	1.766	0.605	0.444
(Severe Overfitting)	2	1.750	0.597	0.452
	3	1.727	0.599	0.433

for `lsa_qd` The model maintains good generalization even at this extreme horizon. for `qd` Severe overfitting. for `dual_aqd` Very inefficient learning. `sum_k` Severe overfitting.

4.2.3 ETTh2 Dataset Overall Conclusion

Similar to the Exchange Rate dataset, the `lsa_qd` loss function is the best performer on the ETTh2 dataset. It consistently provides high coverage with reasonably sharp prediction intervals across all horizons. The other loss functions, while achieving high coverage on this dataset, do so at the cost of extremely wide prediction intervals, which makes them impractical. The training logs show that `lhsa_qd`'s ability to generalize well is the key to its success, even though the logged training loss values for this method appear to be on a different scale and not directly comparable to the others.

The SA-QD advances UQ in LTSF by dynamically adapting the coverage-sharpness trade-off via learnable parameters (e.g., logarithmic lambda for horizon-specific weighting), outperforming fixed-parameter baselines like QD, Dual-AQD, and Sum-K. This aligns with research objectives to mitigate miscalibration in deep transformers, as evidenced by high PICP without excessive MPIW, ensuring reliable PIs for decision-making in uncertain environments. On both datasets, SA-QD reduced combined metrics (Interval Score, CRPS) by 10-90% relative to baselines, particularly at longer horizons where uncertainty amplifies.

Broader implications include enhanced safety in critical applications (e.g., energy forecasting with ETTh2), where calibrated PIs prevent overconfident predictions. The ensemble approach captures epistemic uncertainty, while HSCP ensures marginal coverage guarantees, linking to literature on conformal prediction for time series.

Despite maintaining stable Prediction Interval Coverage Probability (PICP) across all five random seeds, SA-QD exhibited non-trivial variability in Mean Prediction Interval Width (MPIW), with standard deviations reaching up to 3.997 on ETTh2. This pattern suggests that while the learnable λ parameter reliably converges toward adequate coverage, the sharpness of the resulting intervals remains sensitive to stochastic initialization and training dynamics inherent in deep ensemble methods. Despite SA-QD's substantially higher absolute training loss, its train-validation gap ratio (1.0×) is dramatically smaller than QD's (12×). Combined with consistent test set coverage above 0.95, this confirms that SA-QD generalizes robustly across all horizons while QD overfits severely.

The baselines exhibited markedly dataset-specific behavior: QD and Sum-K achieved substantially better PICP on ETTh2 (approximately 0.96) than on Exchange Rate (approximately 0.24), yet produced disproportionately wide intervals on ETTh2. This divergence likely reflects ETTh2's stronger seasonal structure, which aids coverage in fixed-lambda losses by providing more predictable error patterns, but simultaneously inflates interval widths as the fixed penalty cannot adapt to varying uncertainty scales across horizons. SA-QD's learnable mechanism effectively mitigates this dataset-specific variance, further supporting the case for adaptive over fixed-parameter uncertainty quantification in heterogeneous forecasting contexts.

4.3 Main Findings

The experimental evaluations yielded three principal findings that collectively demonstrate the effectiveness, robustness, and practical value of the proposed SA-QD framework.

Finding 1: Simultaneous Calibration and Sharpness The integration of the SA-QD loss function with PatchTST-based Transformer ensembles and HSCP produced well-calibrated prediction intervals across both datasets and all forecast horizons. The framework achieved PICP values of 0.961 ± 0.009 on Exchange Rate and 0.987 ± 0.008 on ETTh2, closely matching the nominal 95% target while maintaining reasonable MPIW of 10.360 ± 3.985 and 11.205 ± 3.997, respectively. These results confirm that SA-QD successfully resolves the coverage-sharpness dilemma that has historically constrained quality-driven approaches, delivering intervals that are simultaneously reliable and informative a combination that prior baselines failed to achieve consistently across datasets.

Finding 2: Statistically Significant Superiority Over Baselines SA-QD demonstrated statistically significant superiority over all evaluated baselines in probabilistic quality metrics. Reductions in combined metrics Interval Score and CRPS ranged from 23% to 62% against architecturally comparable baselines (QD and Sum-K), with more pronounced gains of up to 92% against Dual-AQD on ETTh2, where Dual-AQD's dual-network architecture produced pathologically wide intervals (MPIW up to 139.222). Sta-

tistical validation via paired t-tests confirmed significance at $p < 0.05$ on ETTh2 against all baselines (vs. QD: $p = 0.001$; vs. Dual-AQD: $p = 0.004$; vs. Sum-K: $p = 0.002$). Training log analyses further revealed that SA-QD consistently avoided the severe overfitting exhibited by QD and Sum-K and the unstable learning dynamics characteristic of Dual-AQD, providing mechanistic evidence that the adaptive loss formulation promotes superior generalization.

Finding 3: Robust Horizon-Specific Adaptation Per-horizon analyses confirmed SA-QD's robust adaptation to escalating uncertainty over extended forecast distances. SA-QD maintained stable PICP above 0.95 with controlled, gradual MPIW growth from 5.574 at horizon 96 to 16.131 at horizon 720 on Exchange Rate a scaling pattern consistent with the theoretical expectation that uncertainty compounds over longer forecast horizons. Baselines, by contrast, exhibited either pronounced coverage degradation (PICP $<$ 0.43 for QD and Sum-K on Exchange Rate) or disproportionate interval widening (MPIW up to 207.924 for Dual-AQD at horizon 720 on ETTh2). This finding directly validates the core design principle of SA-QD: that gradient-based adaptation of balancing parameters, combined with horizon-specific conformal calibration, enables prediction intervals to scale appropriately with growing uncertainty without requiring manual reconfiguration per horizon.

4.3.1 Significance of the Findings

4.3.2 Theoretical Implications

This research makes several meaningful contributions to the theoretical foundations of uncertainty quantification in deep learning for time series forecasting. Most fundamentally, it demonstrates that the coverage-sharpness trade-off in prediction interval generation long treated as a fixed, manually calibrated relationship can be reframed as a learnable optimization problem within a standard gradient-based training pipeline. By embedding λ and μ as trainable parameters within the computational graph, the SA-QD framework extends the theoretical discourse on multi-objective loss design in neural networks [89], showing that competing objectives such as calibration and sharpness need not be

balanced through exhaustive grid search but can instead be resolved through the same backpropagation mechanism that trains the underlying model.

Furthermore, this work advances the theoretical integration of conformal prediction with deep Transformer ensembles in the LTSF domain. While prior conformal prediction variants have provided marginal coverage guarantees, they have often done so at the cost of interval sharpness. The HSCP component introduced in this thesis addresses this limitation by computing separate non-conformity score quantiles for each forecast horizon, yielding horizon-tailored calibration that outperforms horizon-agnostic conformal approaches. This contribution enriches the growing body of literature on adaptive conformal inference for time series, offering a practical bridge between distribution-free coverage theory and the horizon-specific uncertainty dynamics of LTSF.

Finally, by demonstrating that deep ensembles of PatchTST models effectively decompose aleatoric and epistemic uncertainty in multivariate time series, this thesis contributes empirical evidence supporting the ensemble-based uncertainty decomposition framework of Lakshminarayanan, extending its validation from classification and short-horizon regression settings into the more demanding context of long-range multivariate forecasting.

4.3.3 Practical Implications

The practical significance of this research is most immediately apparent in the two application domains evaluated. In energy infrastructure management represented by the ETTh2 dataset the SA-QD framework's ability to generate calibrated intervals with MPIW approximately 60% narrower than QD and 92% narrower than Dual-AQD at equivalent coverage levels translates directly into more actionable uncertainty estimates for grid operators. Reliable, sharp PIs enable proactive capacity planning, optimized reserve allocation, and reduced over-provisioning costs, addressing the real-world risk of demand forecast errors that have been linked to grid reliability challenges documented by energy authorities. In financial risk management represented by the Exchange Rate dataset SA-QD's stable PICP of 0.961 across volatile, regime-shifting currency data supports more reliable volatility assessment and hedging strategy development, offering improvements

over both overconfident point-prediction approaches and excessively conservative interval methods.

Beyond these specific domains, the elimination of manual hyperparameter tuning carries significant practical value for organizations deploying forecasting systems at scale. Where existing methods such as QD and Dual-AQD require dedicated tuning cycles that vary by dataset, confidence level, and forecasting context, SA-QD converges to an effective configuration through standard training, reducing deployment time and technical barriers. This scalability makes the framework particularly suitable for time-sensitive applications in healthcare monitoring and autonomous systems, where operational constraints preclude extensive tuning and where miscalibrated uncertainty estimates carry direct safety implications.

4.4 Research Contribution and Positioning

The proposed framework addresses these limitations by introducing a Self-Adaptive Quality-Driven (SA-QD) loss function. This method extends traditional quality-driven approaches by:

1. **Learnable Balancing Parameters:** Introducing learnable parameters to dynamically balance PI objectives without manual tuning
2. **Horizon-Specific Adaptation:** Addressing horizon-dependent uncertainty growth in Transformer-based ensembles
3. **Seamless Integration:** Combining deep Transformer architectures with ensemble methods for robust uncertainty quantification
4. **Automated Optimization:** Achieving automated, adaptive uncertainty quantification through end-to-end learning

This approach bridges the gap between sophisticated Transformer architectures for long-range dependency modeling and robust, calibrated prediction intervals that adapt to forecast horizon dynamics.

Chapter 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

This thesis introduced the Self-Adaptive Quality-Driven (SA-QD) loss function, integrated with deep Transformer-based ensembles leveraging the PatchTST architecture and enhanced by Horizon-Specific Conformal Prediction (HSCP), to address the persistent challenge of generating calibrated and sharp prediction intervals (PIs) for long-term time series forecasting (LTSF). The central motivation was to overcome a well-documented limitation in existing quality-driven uncertainty quantification methods namely, their dependence on exhaustive manual hyperparameter tuning that renders them impractical for rapid, scalable deployment across diverse real-world domains. By making the coverage-sharpness balancing parameters fully learnable through gradient-based optimization, the SA-QD framework eliminates this bottleneck while simultaneously capturing both aleatoric and epistemic uncertainties across forecast horizons of 96, 192, 336, and 720 steps. Evaluated on the Exchange Rate (financial) and ETTh2 (energy) benchmark datasets, the proposed framework consistently outperformed state-of-the-art baselines QD, Dual-AQD, and Sum-K across all evaluation metrics, establishing SA-QD as a robust, automated, and practically deployable solution for uncertainty quantification in LTSF.

The broader significance of this contribution lies in its alignment with a growing imperative in applied AI: that models deployed in high-stakes domains must not only predict accurately but must communicate the boundaries of their own knowledge reliably. In energy grids, financial systems, healthcare, and autonomous environments, the cost of overconfidence is not a degraded benchmark score it is a blackout, a financial loss, a missed diagnosis, or a collision. SA-QD takes a concrete step toward forecasting systems that are

calibrated by design rather than by coincidence, offering a scalable, automated pathway toward trustworthy uncertainty quantification in long-term time series forecasting. Future extensions to diverse data modalities, conditional coverage frameworks, and emerging Transformer architectures will further consolidate these contributions, advancing the field toward AI systems that are not only powerful but genuinely reliable.

5.2 Recommendations

Practical Applications

Based on the empirical findings of this study, three recommendations are offered for practitioners seeking to deploy the SA-QD framework in operational settings.

First, energy grid operators and financial institutions should prioritize SA-QD over fixed-parameter alternatives when deploying uncertainty quantification systems for medium-to-long forecast horizons (192–720 steps), where the adaptive mechanism demonstrates its greatest advantage over baselines. The framework's ability to maintain PICP above 0.95 without manual retuning across this range makes it directly deployable in production environments with minimal configuration overhead.

Second, practitioners should apply HSCP calibration using a recent, contiguous temporal block from the validation period rather than random subsampling, to minimize distributional shift between calibration and test data. This practice, consistent with the temporal blocking approach employed in this study, ensures that the marginal coverage guarantees provided by conformal prediction remain reliable in the presence of the temporal dependencies inherent in real-world time series data.

Third, organizations deploying SA-QD in domains with extreme volatility such as cryptocurrency markets or high-frequency trading should treat the current framework as a strong baseline and supplement it with out-of-distribution detection mechanisms to flag periods where the training distribution has shifted significantly, as the marginal coverage guarantees do not extend to conditional coverage under non-stationary regimes.

Future Research Directions

This study opens several productive directions for future research that would address its current limitations and extend its contributions.

The most immediate and architecturally straightforward extension would be the vectorization of the learnable parameters, replacing the current shared scalar λ with a horizon-specific parameter vector $\lambda_h \in \mathbb{R}^4$ and extending η to a channel-specific tensor of dimensions $[pred\_len \times c\_out]$. This modification would align the loss-level adaptation with the horizon-specific calibration already achieved through HSCP, potentially reducing the MPIW variability across seeds observed as standard deviations of up to ± 3.997 on ETTh2 by enabling finer-grained gradient signals per forecast distance and output channel.

Future work should also pursue systematic ablation studies isolating the contribution of each framework component the SA-QD loss alone, HSCP alone, and the ensemble mechanism alone to quantify their individual and combined effects on calibration and sharpness. Such ablations would strengthen the empirical foundation of the proposed framework and provide clearer guidance for practitioners deciding which components to adopt in resource-constrained settings.

Extending evaluation to datasets characterized by extreme volatility (e.g., cryptocurrency), complex seasonality (e.g., weather, traffic), and clinical time series (e.g., EEG, physiological signals) would validate the generalizability of SA-QD beyond the financial and energy domains examined here. These domains present distinct noise profiles and non-stationarity patterns that would stress-test the adaptive mechanism and potentially motivate domain-specific architectural modifications. Additionally, integrating meta-learning techniques to pre-train the λ initialization across a diverse corpus of time series datasets could address the initialization sensitivity observed in this study and accelerate convergence in low-data deployment scenarios. Finally, exploring conditional coverage guarantees moving beyond the marginal guarantees provided by standard conformal prediction through adaptive conformal inference methods or risk-conditional calibration frameworks would significantly strengthen the theoretical grounding of the HSCP com-

ponent and its applicability in safety-critical domains where coverage must hold across all subpopulations, not just on average.

REFERENCES

[1] L. Su, X. Zuo, R. Li, X. Wang, H. Zhao, and B. Huang, "A systematic review for transformer-based long-term series forecasting," 2025.

[2] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods," *Machine Learning*, 2021.

[3] W. He, Z. Jiang, T. Xiao, Z. Xu, and Y. Li, "A survey on uncertainty quantification methods for deep learning," 2023.

[4] G. Zhang, Y. Wu, K. P. Wong, Z. Xu, and Z. Y. Dong, "An advanced approach for construction of optimal wind power prediction intervals," *IEEE Transactions on Power Systems*, 2015.

[5] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?," *Structural Safety*, 2009.

[6] T. Jiang, X. Zhang, Y. Wang, J. Zhang, J. Liu, and J. Zhang, "Uncertainty-aware long-term time series forecasting," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[7] H. Zhong and L. Xu, "An all-batch loss for constructing prediction intervals," *Applied Sciences*, 2021.

[8] G. Morales and J. W. Sheppard, "Dual accuracy-quality-driven neural network for prediction interval generation," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2023.

[9] W. Amnuaypongsa and J. Songsiri, "Large width penalization for neural network-based predition interval estimation," 2024.

[10] T. S. Salem, H. Langseth, and H. Ramampiaro, "Prediction intervals: Split normal mixture from quality-driven deep ensembles," Proceedings of Machine Learning Research, PMLR, 2020.

[11] T. Pearce, A. Brintrup, M. Zaki, and A. Neely, "High-quality prediction intervals for deep learning: A distribution-free, ensembled approach," PMLR, 2018.

[12] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Comprehensive review of neural network-based prediction intervals and new advances," *IEEE Transactions on Neural Networks*, 2011.

[13] P. Anand, S. G Bodde, and K. P, "Tube loss based deep networks for improving the probabilistic forecasting of wind speed," *Renewable Energy*, vol. 222, p. 119859, 2024.

[14] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[15] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *International Conference on Machine Learning*, pp. 1613–1622, 2015.

[16] R. M. Neal, *Bayesian Learning for Neural Networks*, vol. 118 of *Lecture Notes in Statistics*. New York: Springer-Verlag, 1996.

[17] W. Yao, Z. Zeng, and C. Lian, "Generating probabilistic predictions using mean-variance estimation and echo state network," *Neurocomputing*, vol. 219, 2017.

[18] Y. Lai, Y. Shi, Y. Han, Y. Shao, M. Qi, and B. Li, "Exploring uncertainty in deep learning for construction of prediction intervals," *IEEE Access*, vol. 9, pp. 108969–108983, 2021.

[19] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[20] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?," 2023.

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

[22] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Advances in Neural Information Processing Systems*, 2021.

[23] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A Time Series is Worth 64 Words: Long-term Forecasting with Transformers," 2023.

[24] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J. Y. Zhang, and J. Zhou, "Timemixer: Decomposable multiscale mixing for time series forecasting," 2024.

[25] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *International Conference on Machine Learning*, pp. 27268–27286, 2022.

[26] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 3rd ed., 2021.

[27] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *International Journal of Forecasting*, vol. 32, no. 3, pp. 914–938, 2016.

[28] S.-H. Poon and C. W. Granger, "Forecasting volatility in financial markets: A review," *Journal of economic literature*, vol. 41, no. 2, pp. 478–539, 2003.

[29] A. A. Syntetos and J. E. Boylan, "The accuracy of intermittent demand estimates," *International journal of forecasting*, vol. 21, no. 2, pp. 303–314, 2005.

[30] G. U. Yule, "On a method of investigating periodicities in distributed series, with special reference to wolfer's sunspot numbers," *Philosophical Transactions of the Royal Society of London A*, vol. 226, pp. 267–298, 1927.

[31] G. T. Walker, "On periodicity in series of related terms," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 131, no. 818, pp. 518–532, 1931.

[32] I. Rojas *et al.*, "Soft-computing techniques and arma model for time series prediction," *Neurocomputing*, vol. 71, no. 4–6, pp. 519–537, 2008.

[33] G. E. P. Box and D. A. Pierce, "Distribution of residuals in autoregressive-integrated moving average time series," *Journal of the American Statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.

[34] S. L. J. Marple and W. M. Carey, "Digital spectral analysis with applications," *Journal of the Acoustical Society of America*, vol. 86, no. 5, p. 2043, 1998.

[35] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, NJ: John Wiley, 5th ed., 2015.

[36] Q. Wang *et al.*, "A deep granular network with adaptive unequal-length granulation strategy for long-term time series forecasting and its industrial applications," *Artificial Intelligence Review*, vol. 53, no. 7, pp. 5353–5381, 2020.

[37] A. Farnoosh *et al.*, "Deep switching auto-regressive factorization: Application to time series forecasting," 2020.

[38] D. J. McDonald *et al.*, "Nonparametric risk bounds for time-series forecasting," *Journal of Machine Learning Research*, vol. 18, no. 32, pp. 1–40, 2012.

[39] Q. Wen *et al.*, "Robuststl: A robust seasonal-trend decomposition algorithm for long time series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5409–5416, 2019.

[40] X. Yang *et al.*, "Long-term forecasting of time series based on linear fuzzy information granules and fuzzy inference system," *International Journal of Approximate Reasoning*, vol. 81, pp. 1–27, 2017.

[41] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[42] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121, no. 2, pp. 256–285, 1995.

[43] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[45] K. Cho *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2020.

[46] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," 2015.

[47] J. Chung *et al.*, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[48] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[49] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," 2020.

[50] J. W. Rae, A. Potapenko, S. M. Jayakumar, C. Hillier, and T. P. Lillicrap, "Compressive transformers for long-range sequence modelling," 2020.

[51] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[52] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," pp. 1872–1897, 2020.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[54] J. Cheon and S. Paik, "Pretraining with random noise for uncertainty calibration," 2024.

[55] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, 2017.

[56] J. Gawlikowski, C. R. N. Tassi, M. Ali, and et al., "A survey of uncertainty in deep neural networks," pp. 1513–1589, 2023.

[57] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, and J. Snoek, "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift," in *Advances in Neural Information Processing Systems*, 2019.

[58] M. Abdar and F. P. et al., "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," 2021.

[59] T. Gneiting, F. Balabdaoui, and A. E. Raftery, "Probabilistic forecasts, calibration and sharpness," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2007.

[60] C. Chatfield, "Calculating interval forecasts," *Journal of Business & Economic Statistics*, 1993.

[61] M. Cheng, Z. Liu, X. Tao, Q. Liu, and et al., "A comprehensive survey of time series forecasting: Concepts, challenges, and future directions," 2025.

[62] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," 2016.

[63] N. A. E. R. Corporation, "2023–2024 winter reliability assessment," tech. rep., North American Electric Reliability Corporation (NERC), 2023.

[64] W. Parker, "What went wrong with zillow?," *Bloomberg Businessweek*, 2021.

[65] I. M. Fund, "Global financial stability report: The last mile: Financial vulnerabilities and risks," tech. rep., International Monetary Fund (IMF), 2024.

[66] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," 2021.

[67] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[68] International Organization for Standardization (ISO), "Road vehicles — safety of the intended functionality," 2019.

[69] K. Chow, C. "On optimum recognition error and reject tradeoff," *IEEE Transactions on Information Theory*, 1970.

[70] C. De Stefano, C. Sansone, and M. Vento, "To reject or not to reject: That is the question—an answer in case of neural classifiers," 2000.

[71] R. H. Hariri, E. M. Fredericks, and K. M. Bowers, "Uncertainty in big data analytics: Survey, opportunities, and challenges," 2019.

[72] P. L. Bartlett and M. H. Wegkamp, "Classification with a reject option using a hinge loss," *Journal of Machine Learning Research*, 2008.

[73] J. Nixon, M. W. Dusenberry, L. Zhang, G. Jerfel, and D. Tran, "Measuring calibration in deep learning," 2019.

[74] H. Xu, Z. Zhu, S. Zhang, D. Ma, S. Fan, L. Chen, and K. Yu, "Rejection improves reliability: Training llms to refuse unknown questions using rl from knowledge feedback," 2024.

[75] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," 2025.

[76] J. Chen and J. Mueller, "Quantifying uncertainty in answers from any language model and enhancing their trustworthiness," 2024.

[77] S. Lin, J. Hilton, and O. Evans, "Teaching models to express their uncertainty in words," 2022.

[78] Y. Yu, L. Kong, J. Zhang, R. Zhang, and C. Zhang, "Actune: Uncertainty-based active self-training for active fine-tuning of pre-trained language models," 2024.

[79] X. Wang *et al.*, "Online conformal inference for multi-step time series forecasting," tech. rep., Monash Business School, 2024.

[80] G. R. Esteves, B. Q. Bastos, F. L. Cyrino, R. F. Calili, and R. C. Souza, "Long term electricity forecast: A systematic review," pp. 1234–1247, 2015.

[81] W. V. Kambale, F. A. Machot, D. K. Kadurha, T. Bernabia, A. Deeb, and K. Kyamakya, "Transformers in time series forecasting: A brief analysis of the autoformer transfer learning performance," 2023.

[82] Y. Wu, M. Hernández-Lobato, and Z. Ghahramani, "Dynamic covariance models for multivariate financial time series," 2020.

[83] R. C. Cavalcante, R. C. Brasileiro, V. L. Souza, J. P. Nobrega, and A. L. Oliveira, "Computational intelligence and financial markets: A survey and future directions," 2016.

[84] K. Kadir, C. Halim, K. O. Harun, and E. C. Olcay, "Modeling and prediction of turkey's electricity consumption using artificial neural networks," 2009.

[85] B. Lim, A. M. Alaa, and M. van der Schaar, "Forecasting treatment responses over time using recurrent marginal structural networks," 2018.

[86] X. Zhu, B. Fu, Y. Yang, Y. Ma, J. Hao, S. Chen, S. Liu, T. Li, S. Liu, W. Guo, and Z. Liao, "Attention-based recurrent neural network for influenza epidemic prediction," 2019.

[87] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," pp. 1310–1318, 2013.

[88] M. R. Kabir, D. Bhadra, M. Ridoy, and M. Milanova, "Lstm–transformer-based robust hybrid deep learning model for financial time series forecasting," 2025.

[89] J. Nowotarski and R. Weron, "Recent advances in electricity price forecasting: A review of probabilistic forecasting," *Renewable and Sustainable Energy Reviews*, 2018.

[90] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang, "Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting," *Electronics*, 2019.

[91] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, 2012.

[92] F. Hutter, L. Kotthoff, and J. Vanschoren, eds., *Automated Machine Learning: Methods, Systems, Challenges*. Springer Nature, 2019.

[93] S. B. Taieb and A. F. Atiya, "A bias and variance analysis for multistep-ahead time series forecasting," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 1, pp. 62–76, 2016.

[94] M. Marcellino, J. H. Stock, and M. W. Watson, "A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series," *Journal of Econometrics*, vol. 135, no. 1-2, pp. 499–526, 2006.

[95] G. Chevillon, "Direct multi-step estimation and forecasting," *Journal of Economic Surveys*, vol. 21, no. 4, pp. 746–785, 2007.

[96] G. Morales and J. W. Sheppard, "Adaptive sampling to reduce epistemic uncertainty using prediction interval-generation neural networks," in *AAAI Conference on Artificial Intelligence*, 2025.

[97] I. M. Galván, J. M. Valls, A. Cervantes, and R. Aler, "Multi-objective evolutionary optimization of prediction intervals for solar energy forecasting with neural networks," 2017.

[98] J. Yang, H. Chen, B. Han, and S. Yu, "Constructing prediction intervals to explore uncertainty based on deep neural networks," 2024.

[99] H. Quan, D. Srinivasan, and A. Khosravi, "Short-term load and wind power forecasting using neural network-based prediction intervals," 2014.

[100] W. Li, Z. Wang, Q. Sun, Q. Gao, and F. Yang, "Energypatchtst: Multi-scale time series transformers with uncertainty estimation for energy forecasting," in *Proceedings of the International Conference on Intelligent Computing (ICIC)*, 2025.

[101] Q. Pan *et al.*, "Constructing prediction intervals for circuit board fault remaining useful life prediction based on quality-driven loss function," *Electronics Letters*, 2024.

[102] C. Xu and Y. Xie, "Conformal prediction interval for dynamic time-series," 2021.

[103] Y. Lai, Y. Shi, Y. Han, Y. Shao, M. Qi, and B. Li, "Exploring uncertainty in regression neural networks for construction of prediction intervals," *Neurocomputing*, 2022.

[104] J. Yang, D. Huang, and X. Cheng, "Constructing prediction intervals to explore uncertainty based on deep neural networks," *Journal of Intelligent & Fuzzy Systems*, vol. 44, no. 4, pp. 5557–5568, 2023.

[105] Z. Huang, B. Hu, H. He, and Y. Jin, "Coverage-driven deep prediction intervals method," in *2020 International Conference on Image, Video Processing and Artificial Intelligence*, 2020.

[106] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Lower upper bound estimation method for construction of neural network-based prediction intervals," *IEEE Transactions on Neural Networks*, 2011.

[107] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[108] A. Casolaro, V. Capone, G. Iannuzzo, and F. Camastra, "Deep learning for time series forecasting: Advances and open problems," *Information*, 2023.

[109] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *1604.07316*, 2016.

[110] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, 2017.

[111] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *1610.02136*, 2017.

[112] A. L. Simpkin and R. M. Schwartzstein, "Tolerating uncertainty—the next medical revolution?," *New England Journal of Medicine*, 2016.

[113] J. Mena and O. Pujol, "A survey on uncertainty estimation in deep learning classification systems from a bayesian perspective," 2022.

[114] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," in *Advances in Neural Information Processing*, 2018.

[115] R. Russell and C. Reale, "Multivariate uncertainty in deep learning," *1910.14215*, 2019.

[116] M. Hein, M. Andriushchenko, and J. Bitterwolf, "Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[117] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," in *Advances in Neural Information Processing Systems*, 2018.

[118] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," in *International Conference on Machine Learning*, 2018.

[119] E. Ilg, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox, "Uncertainty estimates for optical flow with multi-hypotheses networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2018.

[120] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning*, pp. 1050–1059, 2016.

[121] A. G. Wilson, "The case for Bayesian deep learning," 2020. Preprint.

[122] D. T. Ulmer, C. Hardmeier, and J. Frellsen, "Prior and posterior networks: A survey on evidential deep learning methods for uncertainty estimation," *Transactions on Machine Learning Research*, 2023.

[123] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, 2017.

[124] H. Ritter, A. Botev, and D. Barber, "A scalable laplace approximation for neural networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[125] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, 2014.

[126] F. Arnez, H. Espinoza, A. Radermacher, and F. Terrier, "A comparison of uncertainty estimation approaches in deep learning components for autonomous vehicle applications," 2021.

[127] D. Bethell, S. Gerasimou, and R. Calinescu, "Robust uncertainty quantification using conformalised monte carlo prediction," in *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)*, 2024.

[128] M. Hobbhahn, A. Kristiadi, and P. Hennig, "Fast predictive uncertainty for classification with bayesian deep networks," in *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2022.

[129] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," 2019.

[130] R. Li *et al.*, "Neural conformal control for time series forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[131] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," 2022.

[132] R.-G. Cirstea, C. Guo, B. Yang, T. Kieu, X. Dong, and S. Pan, "Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting – full version," *2204.13767*, 2022.

[133] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," *2001.08317*, 2020.

[134] M. Alharthi and A. Mahmood, "xlstmtime: Long-term time series forecasting with xlstm," *AI*, 2024.

[135] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[136] S. Lin, W. Lin, W. Wu, S. Wang, and Y. Wang, "Petformer: Long-term time series forecasting via placeholder-enhanced transformer," *04791*, 2023.

[137] Y. Liu, H. Wu, J. Wang, and M. Long, "Non-stationary transformers: Exploring the stationarity in time series forecasting," in *Advances in Neural Information Processing Systems (NeurIPS 35)*, 2022.

[138] T. Zhou, J. Zhu, X. Wang, Z. Ma, Q. Wen, L. Sun, and R. Jin, "Treedrnet: A robust deep model for long term time series forecasting," 2022.

[139] J. Lee, C. Xu, and Y. Xie, "Transformer conformal prediction for time series," 2024.

[140] K. Stankeviciute, A. M. Alaa, and M. van der Schaar, "Conformal time-series forecasting," in *Advances in Neural Information Processing Systems*, 2021.

[141] H. Quan, D. Srinivasan, A. Khosravi, S. Nahavandi, and D. Creighton, "Construction of neural network-based prediction intervals for short-term electrical load forecasting," in *2013 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*, 2013.

[142] R. Al-Hajj, G. Oskrochi, M. M. Fouad, and A. Assi, "Probabilistic prediction intervals of short-term wind speed using selected features and time shift dependent machine learning models," *Mathematical Biosciences and Engineering*, 2024.

[143] J. Yao, W. Pan, S. Ghosh, and F. Doshi-Velez, "Quality of uncertainty quantification for bayesian neural network inference," 2019.

[144] F. Schlembach, E. Smirnov, I. Koprinska, and M. H. M. Winands, "Conformal multistep-ahead multivariate time-series forecasting," in *Proceedings of the Eleventh Symposium on Conformal and Probabilistic Prediction with Applications (COPA 2022)*, Proceedings of Machine Learning Research, PMLR, 2022.

[145] X. Kong, Z. Chen, W. Liu, K. Ning, L. Zhang, S. M. Marier, Y. Liu, Y. Chen, and F. Xia, "Deep learning for time series forecasting: A survey," 2023.

[146] J. Zhang, W. Zhu, and J. Gao, "Learning novel transformer architecture for time-series forecasting," *arXiv*, 2025.

[147] S. Liang, C. Hou, X. Yao, S. Wang, M. Jiang, S. Han, and H. Huang, "Tsgym: Design choices for deep multivariate time-series forecasting," *arXiv*, 2025.

[148] D. D. Modi and R. Pan, "Enhancing transformer-based foundation models for time series forecasting via bagging, boosting and statistical ensembles," *arXiv*, 2025.

[149] X. Mootoo, H. Tabassum, and L. Chiaraviglio, "Emforecaster: A deep learning framework for time series forecasting in wireless networks with distribution-free uncertainty quantification," *arXiv 2504.00120*, 2025.

[150] A. Khosravi, S. Nahavandi, and D. Creighton, "Construction of optimal prediction intervals for load forecasting problems," *IEEE Transactions on power systems*, 2010.

[151] E. Torsen and L. L. Seknewna, "Bootstrapping nonparametric prediction intervals for conditional value-at-risk with heteroscedasticity," *Journal of Probability and Statistics*, 2019.

[152] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[153] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*, pp. 1–15, Springer, 2000.

[154] H. Li, X. Su, J. Wang, H. Lin, J. Li, Y. Wang, X. Hu, W. Lin, L. Hu, H. Jin, Y. Xu, J. Li, and B. Long, "Transformer-modulated diffusion models for probabilistic multivariate time series forecasting," *International Conference on Learning Representations (ICLR)*, 2024.

[155] X. Mao, C. Jin, H. Hsu, T. Ayed, H. Li, S. Mak, P. P. Wang, X. Yang, M. I. Jordan, and Y. Jiao, "Scaling transformers for time series forecasting: Do pretrained large models outperform small-scale alternatives?," 2025.

[156] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018.

[157] S. Jain, G. Liu, J. Mueller, and D. Gifford, "Maximizing overall diversity for improved uncertainty estimates in deep ensembles," 2020.

[158] Q. Hong, F. Meng, and F. Maldonado, "Advancing long-term multi-energy load forecasting with patchformer: A patch and transformer-based approach," 2023.

[159] W. Li and K. L. E. Law, "Deep learning models for time series forecasting: A review," 2023.

[160] A. Ben Arie and M. Gorfine, "Confidence intervals and simultaneous confidence bands based on deep learning," 2023.

[161] F. Seligmann, P. Becker, M. Volpp, and G. Neumann, "Beyond deep ensembles: A large-scale evaluation of bayesian deep learning under distribution shift," 2023.

[162] M.-H. Liu, T.-Y. Wang, C. Yang, and H. Dong, "A novel interval prediction method in wind speed based on deep learning and combination prediction," *Scientific Reports*, vol. 15, no. 1, p. 14475, 2025.

[163] X. Wang, J. Li, Z. Li, P. Li, and K. Li, "An enhanced differential learning wind speed interval-value prediction system based on optimal collaborative interval decomposition and strategic model selection," *Renewable Energy*, 2025.

[164] A. Saeed, C. Li, and Z. Gan, "Short-term wind speed interval prediction using improved quality-driven loss based gated multi-scale convolutional sequence model," *Energy*, vol. 300, p. 131590, 2024.

[165] J. Vikranth, "Loss functions in deep learning: A comprehensive review," 2025.

[166] Y. Li, C. Xue, P. Wang, Y. Tan, X. Bai, F. Gao, and J. Ma, "A comprehensive survey of loss functions and metrics in deep learning," *Artificial Intelligence Review*, 2025.