Modern Deep Learning Methods for Time Series Analysis

Xavier Stephen Mootoo

A Thesis submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements for the
Master of Science in Computer Science

GRADUATE PROGRAM IN COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

August 2025

Abstract

Deep learning methods for time series analysis have become prominent tools in state-of-the-art predictive modeling, with applications spanning finance, transportation, energy, healthcare, climate science, and numerous other domains. In this thesis, we explore modern deep learning techniques that address two key challenges common across many domains: handling variable-structure inputs and quantifying prediction uncertainty, with the goal of building models that are both robust and adaptable. We demonstrate these advances across two distinct domains: electromagnetic field (EMF) exposure time series forecasting in modern wireless networks and variable-length time series classification (VTSC) in healthcare.

The first contribution introduces EMForecaster, a novel deep learning framework designed for accurate EMF exposure prediction. The architecture employs hierarchical patching to capture temporal patterns at multiple scales, complemented by reversible instance normalization and mixing operations for efficient feature extraction. To enhance reliability, EMForecaster incorporates conformal prediction mechanisms that provide principled uncertainty quantification, enabling trustworthy forecasts with guaranteed coverage rates. A new *Trade-off Score* metric is developed to balance prediction reliability against interval width. Empirical evaluations demonstrate EMForecaster's superior performance across diverse EMF datasets, with improvements

of up to 53.97% over transformer architectures for point forecasts, while maintaining optimal balance between prediction interval coverage and width.

The second contribution presents a Stochastic Sparse Sampling (SSS) framework for variable-length time series classification, a prevalent challenge in healthcare applications. SSS addresses the inherent variability in clinical time series by intelligently sampling fixed windows to compute local predictions, which are then aggregated and calibrated to form robust global classifications. This approach is validated on the task of seizure onset zone (SOZ) localization using intracranial electroencephalography (iEEG) recordings from the Epilepsy iEEG Multicenter Dataset. SSS demonstrates superior performance compared to state-of-the-art methods across most medical centers, particularly excelling in out-of-distribution scenarios with previously unseen data sources. Additionally, SSS provides valuable post-hoc insights by visualizing temporally averaged local predictions throughout the signal.

Together, these methodologies advance the state-of-the-art in time series analysis through innovative deep learning techniques, uncertainty quantification, and interpretability methods, offering significant improvements for both forecasting and classification tasks in real-world wireless network and health-care applications.

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Hina Tabassum, whose expert guidance, constant encouragement, and insightful feedback have been indispensable to this work. I am also especially grateful to Alan A. Díaz-Montiel for his invaluable mentorship and guidance during my time at Krembil Research Institute. Working alongside Alan was a pleasure, and I am incredibly proud of the work we accomplished together.

My appreciation extends to Prof. Luca Chiaraviglio for his insightful contributions; to Dr. Sara Adda (ARPA Piemonte, Italy) and Prof. Cetin Kurnaz (Ondokuz Mayis University, Turkey) for generously sharing the datasets that formed the backbone of this study; and to Anastasios Angelopoulos and Daniele Grattarola for their constructive feedback and comments on my work. I also wish to thank my thesis committee members, Prof. Manos Papagelis and Prof. Divya Sharma, for their thoughtful critiques and guidance. I'm equally grateful as well to Commune AI for generously providing the computational resources that made our experiments possible—special thanks to Luca Vivona and Sal Vivona.

This research was supported by the Discovery, Alliance International and the Canada Graduate Scholarship—Master's (CGS-M) funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada; the Vector Scholarship in

Artificial Intelligence, provided through the Vector Institute, Canada; and the Ontario Graduate Scholarship (OGS) granted by the provincial government of Ontario, Canada.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Time Series Analysis

Time series analysis (TSA) plays a crucial role across a variety of domains, offering powerful tools for analyzing temporal patterns and making predictions from sequential data. In healthcare and neuroscience, TSA enables the interpretation of complex physiological signals such as electroencephalography (EEG) recordings, which can aid in identifying neurological states and providing crucial information on brain dynamics. These applications are particularly vital in epilepsy, where analyzing variable-length electrophysiological recordings aids in pinpointing seizure onset zones (SOZs)—the brain regions where seizures originate—and thus better informs downstream surgical interventions [72].

1.1.1 Time Series Analysis in Healthcare

Artificial Intelligence (AI)-driven time series classification (TSC) has received significant attention in recent years for clinical diagnosis and treatment planning [98]. These methods involve building predictive models that analyze sequences of patient data over

time to identify specific medical conditions such as epileptic events, heart arrhythmias, or monitoring of disease progression. Despite these advancements, application into everyday clinical practices remains limited, largely due to challenges in handling the complexity and variations in medical time series data. TSC methods can be broadly categorized into *finite-context* methods, which operate on fixed-length input sequences, and *infinite-context methods*, which handle variable-length sequences without being restricted to a predetermined window size. Although most medical data consists of variable-length time series due to factors such as unique patient recordings or event durations [4, 32, 123], the majority of medical TSC literature focuses solely on finite-context methods that process fixed-length sequences [54, 84]. To accommodate input variations, finite-context methods necessitate padding, truncation, or interpolation, which often results in distortion of the original medical data. Furthermore, as sequence length increases, so does the number of model parameters, leading to higher computational costs and an increased risk of overfitting.

Epilepsy serves as an important and rising application of AI-driven TSC methods in the medical field. The World Health Organization (WHO) reports epilepsy affects over 50 million people globally, establishing it as one of the most common yet poorly understood neurological disorders [90, 113]. Additionally, one-third of patients do not respond to antiepileptic drugs, making surgery the last resort and accurate SOZ localization essential for effectively planning the operation. The process of SOZ identification involves a two-step procedure: initial implantation of electrodes in areas suspected to contain the SOZ, followed by recording and visual analysis of intracranial electroencephalography (iEEG) signals by medical experts. The task of SOZ localization reduces to classifying individual electrode recordings, representing

different regions within the brain. Effective localization of the SOZ is challenging due to the absence of clinically validated biological markers and the variable-length nature of iEEG signals—consequently, surgical success rates range from 30% to 70% [72, 77].

1.1.2 Time Series Analysis in Wireless Networks

Amid rapid advancements in wireless technology, concerns grow surrounding the potential increase in electromagnetic field (EMF) exposure resulting from new radio transmission frequencies, massive radiating elements, and the dense deployment of network infrastructure in fifth-generation (5G) and beyond (B5G) [26, 103]. EMF exposure can induce thermal effects, such as heating of the exposed tissues and organs, depending on the level of radiation absorbed. Consequently, regulatory authorities such as International Commission on Non-Ionizing Radiation Protection (ICNIRP) define rigorous limits on the maximum radio-frequency EMF exposure between 100 kHz and 300 GHz [52], which are adopted by many nations across the world. ICNIRP guidelines initially published in 1998 have been revised in 2020 to reflect recent advancements in scientific understanding of EMF exposure effects. In addition to the ICNIRP guidelines, several countries have adopted more stringent EMF regulations [114].

As a result of rising public health concerns, EMF monitoring and forecasting is becoming increasingly critical alongside the wireless technological advancements to ensure regulatory compliance. By providing insights into long-term EMF trends, reliable EMF forecasting enables proactive risk management and informed decision-making, thereby balancing wireless network innovation with safety and public trust. EMF forecasting capabilities allow network operators to optimize infrastructure placement and power settings while maintaining EMF levels within regulatory limits, in addition

to enabling public health agencies to properly assess population exposure patterns and implement preventive measures.

1.1.3 Other Applications

Beyond these domains, TSA has been applied extensively in financial markets for forecasting asset returns and risk assessment, climate science for weather forecasting and climate change analysis, industrial settings for predictive maintenance and quality control, as well as numerous other domains [64, 68, 99]. Given the ubiquitous nature of temporal data and its growing importance across sectors, advancing the capabilities of TSA remains a critical research direction with far-reaching implications for both scientific understanding and practical applications.

1.1.4 Classical Time Series Analysis

Classical methods for TSA provide the foundational principles for understanding and forecasting temporal data. These techniques, while often simpler in their mathematical framework than deep learning methods, remain relevant and effective for a variety of applications. They are generally characterized by their reliance on statistical properties of the data, such as trend, seasonality, and autocorrelation. Such methods range from forecasting models like Autoregressive Integrated Moving Average (ARIMA)—a framework that models a future value based on its own past values (autoregression) and past forecast errors (moving average)—to pattern discovery techniques like the Matrix Profile, which efficiently identifies repeated patterns (motifs) and anomalies (discords) [22, 130]. These approaches are highly interpretable, allowing researchers to understand the underlying components of a time series. They are computationally

efficient and can perform well on smaller datasets, providing robust short-term forecasts without the need for extensive computational resources, making them particularly effective when the time series exhibits clear and stable patterns.

However, these traditional approaches also have significant limitations. They often rely on strict statistical assumptions, such as stationarity, a prerequisite for models like ARIMA that is frequently violated in real-world scenarios. In general, classical models also struggle to capture more complex, non-linear relationships within the data. Furthermore, the performance of forecasting models like ARIMA can degrade significantly when it comes to long-term forecasting, as errors tend to accumulate over extended prediction horizons.

1.2 The Rise of Deep Learning for Time Series Analysis

Deep learning has emerged as a transformative tool in TSA, as well as other fields including computer vision, natural language processing, and speech recognition [49, 102, 118]. This surge in popularity can be attributed to several key factors. First, the exponential growth in available data, often referred to as the "big data revolution," has provided the massive datasets necessary for training deep neural networks effectively [63]. The proliferation of sensors, Internet of Things (IoT) devices, and digital systems has generated unprecedented volumes of temporal data, creating an environment where deep learning models can leverage their capacity to learn complex patterns from large-scale datasets. Second, advancements in computational resources, particularly the widespread availability of GPUs and specialized hardware accelerators, have made it feasible to train increasingly sophisticated neural architectures on these extensive datasets [65].

What sets deep learning apart in TSA is its ability to automatically learn hierarchical representations from raw data without requiring extensive feature engineering or prior assumptions about the underlying data generation process [84]. Unlike traditional statistical models that often rely on strict assumptions about stationarity, linearity [22], or specific probability distributions, deep learning models can capture non-linear relationships and adapt to changing patterns in the data. This flexibility, combined with their ability to process high-dimensional multivariate time series, has led to deep learning methods increasingly outperforming classical statistical and machine learning approaches in various forecasting and classification tasks. Moreover, recent architectures such as Transformers and specialized networks including PatchTST have demonstrated remarkable capabilities in capturing both long-term dependencies and local temporal patterns, addressing limitations that have historically challenged traditional time series methods.

1.3 Challenges in Deep Learning for Time Series Analysis

Despite its success, deep learning faces several fundamental challenges in TSA. The irregular nature of real-world temporal data, including missing values, variable-length sequences, and non-uniform sampling rates, poses significant modeling difficulties that require careful architectural design and preprocessing strategies. Moreover, the inherent complexity of deep neural networks often results in "black box" models whose decisions lack interpretability, a critical concern in high-stakes applications such as healthcare and financial forecasting. These interpretability concerns are further compounded by the challenge of quantifying prediction uncertainty, which is essential for reliable decision-making in practical applications.

1.3.1 Challenges in Time Series Classification

In time series classification, a particularly challenging scenario arises when dealing with variable-length sequences, which are common in domains such as healthcare (e.g., patient monitoring) and industrial processes (e.g., manufacturing cycles). Traditional deep learning architectures require fixed-length inputs, necessitating problematic padding or truncation operations that can distort temporal relationships or destroy valuable information. While recurrent architectures such as Long-Short Term Memory (LSTM) networks can naturally handle variable-length inputs, they have been shown to struggle with capturing long-range dependencies and often face training instabilities. Recent approaches leveraging random convolutional kernels have shown promise in addressing these challenges, but the fundamental tradeoff between model flexibility and performance remains a critical challenge. This represents a key limitation in modern deep learning research, as the ability to handle variable-length sequences should be a fundamental capability rather than a specialized feature. The persistence of this challenge, where fixed-length assumptions remain deeply embedded in modern architectures, highlights a significant gap between current deep learning frameworks and real-world applications.

1.3.2 Challenges in Time Series Forecasting

The forecasting landscape presents its own set of distinct challenges for deep learning models. A primary concern is the reliable quantification of prediction uncertainty, which is crucial for risk assessment and decision-making in applications ranging from financial trading to climate modeling. While traditional statistical methods often provide well-calibrated uncertainty estimates through their probabilistic foundations,

deep learning models typically output point predictions that may fail to capture the full range of possible future scenarios with no statistical guarantees. This limitation has spurred research into uncertainty quantification methods such as conformal prediction, which provides distribution-free prediction intervals with finite-sample coverage guarantees [12]. Equally important is the challenge of interpretability in forecasting models, where understanding the reasoning behind predictions is crucial for building trust and enabling practical deployment. While statistical models offer clear insights through their parameter estimates and model structure, deep learning approaches often operate as black boxes, making it difficult to validate their decision-making process or incorporate domain expertise. Recent advances in interpretable machine learning, such as attention visualization and feature attribution methods, have begun to address this gap, but achieving both accurate forecasts and transparent decision processes remains an active area of research.

1.4 Scope of the Thesis

In this thesis, we develop and investigate modern deep learning methods for time series, advancing novel solutions in two high-impact domains. First, we introduce EMForecaster, a novel deep learning framework for EMF exposure prediction from wireless networks that enables regulatory compliance, efficient network management, and monitoring of environmental and health concerns. Second, we develop a modern approach, called Stochastic Sparse Sampling, to train and utilize deep learning models for variable-length time series classification (VTSC), applied to the task of SOZ localization in iEEG recordings. The details are provided as follows:

- With the recent advancements in wireless technologies, forecasting EMF exposure

has become increasingly critical to enable proactive network spectrum and power allocation, as well as network deployment planning. In Chapter 3, we develop a deep learning-empowered time series forecasting framework referred to as *EMForecaster*. The proposed architecture employs patching to process temporal patterns at multiple scales, complemented by reversible instance normalization and mixing operations along both temporal and patch dimensions for efficient feature extraction. We then augment EMForecaster with a conformal prediction mechanism, which is independent of the data distribution, to enhance the trustworthiness of model predictions through uncertainty quantification of forecasts. In particular, the conformal prediction mechanism ensures that the ground truth lies within a prediction interval with target error rate α, where $1 - \alpha$ is referred to as coverage. However, a trade-off exists, as increasing coverage often results in wider prediction intervals. To address this challenge, we propose a new metric referred to as *Trade-off Score*, that balances the trustworthiness of the forecast (i.e., coverage) and the width of prediction interval. Our empirical evaluation demonstrates that EMForecaster achieves superior performance across diverse EMF datasets, spanning both short-term and long-term prediction horizons. In point forecasting tasks, EMForecaster substantially outperforms current state-of-the-art approaches, showing improvements of 53.97% over the Transformer architecture and 38.44% over the average of all baseline models.

- VTSC problems are prevalent in healthcare applications, such as heart rate monitoring and electrophysiological recordings, where sequence length varies among patients and events. VTSC is challenging as finite-context models such as Transformers require padding, truncation, or interpolation, leading to distortion

in the input, higher computational costs, and overfitting, while infinite-context models including recurrent neural networks struggle with overcompression and unstable gradients over long sequences. In Chapter 4, we develop a novel VTSC framework called **S**tochastic **S**parse **S**ampling (SSS) for SOZ localization, a critical VTSC problem requiring identification of seizure-inducing brain regions from variable-length electrophysiological time series. The proposed framework sparsely samples time series windows to compute local predictions, which are then aggregated and calibrated to form a global prediction. We evaluate our method on the Epilepsy iEEG Multicenter Dataset, a heterogeneous collection of iEEG recordings obtained from four independent medical centers. SSS demonstrates superior performance compared to state-of-the-art (SOTA) baselines across most medical centers, and superior performance on all out-of-distribution (OOD) unseen medical centers'. Additionally, SSS naturally provides post-hoc insights into local signal characteristics related to the SOZ, by visualizing temporally averaged local predictions throughout the signal.

A central insight of our work is that the underlying architecture of modern deep learning time series models has become less significant, with many architectures achieving comparable performance. Instead, we argue that research should focus on augmenting these models' capabilities in underexplored yet impactful directions. Our work specifically addresses two such augmentations:

1. **Conformal Prediction** (CP), which we apply to EMF forecasting to quantify uncertainty and enhance forecast trustworthiness while optimizing the balance between prediction interval width and coverage reliability. This approach demonstrates substantial improvements over state-of-the-art methods, showing up to

53.97% improvement over Transformer architectures and introducing a novel Trade-off Score to balance reliability and precision.

2. **Variable-length sequence modeling**, which we implement through our SSS framework (which is model-independent) to address the challenges of heterogeneous time series data in healthcare applications. By sparsely sampling fixed windows to compute local predictions and then aggregating them, we achieve superior performance on both in-distribution and out-of-distribution iEEG data while providing valuable post-hoc insights into local signal characteristics associated with seizure onset zones.

We conduct an in-depth analysis for each augmentation, while comparing against baselines in the current literature.

1.5 Research Outcomes

- X. Mootoo, H. Tabassum, and L. Chiaraviglio, "EMForecaster: A Deep Learning Framework for Time Series Forecasting in Wireless Networks with Distribution-Free Uncertainty Quantification," IEEE Transactions on Network Science and Engineering (IEEE TNSE), to appear, 2025.

- X. Mootoo, A. A. Díaz-Montiel, M. Lankarany, H. Tabassum, "Stochastic Sparse Sampling: A Framework for Local Explainability in Variable-Length Medical Time Series", NeurIPS 2024, Time Series in the Age of Large Models (TSALM) Workshop.

- X. Mootoo, A. A. Díaz-Montiel, M. Lankarany, H. Tabassum, "Stochastic Sparse Sampling: A Variable-Length Time Series Classification Framework for Seizure

Onset Zone Localization", (submitted).

Chapter 2

Preliminaries and Background Work

2.1 Fundamentals of Time Series Analysis

General Definitions

Let $(\mathbf{x}_t)_{t=1}^{T}$ be a time series of total length T. Given a subsequence (or window) $\mathbf{x} = (\mathbf{x}_t)_{t=K}^{K+L}$ with starting index K and sequence length L, the goal of time series forecasting is to predict its continuation $\mathbf{y} = (\mathbf{x}_t)_{t=L+K+1}^{K+L+O}$, where O is the length of the forecast horizon. We will typically view $\mathbf{x} \in \mathbb{R}^L$ and $\mathbf{y} \in \mathbb{R}^O$ as vectors. When preprocessing the initial time series $(\mathbf{x}_t)_{t=1}^{T}$, we use the sliding window technique, which samples windows $(\mathbf{x}_t)_{t=1}^{L}, (\mathbf{x}_t)_{t=2}^{L+1}, \ldots, (\mathbf{x}_t)_{t=L-T}^{T}$ in which we "slide" the starting index K by 1, and sample windows of length L at each index. In general, this forms our dataset, while keeping training, validation, and test sets temporally separated, i.e., they consist of mutually exclusive regions of the sub-sequence. There are not any agreed upon formal definitions, however, *long-term time series forecasting* (LTSF) generally refers to the case of either $O \gg L$, O is relatively large, or both. In contrast, *short-term time series forecasting* (STSF) refers to the case when either $O \ll L$, O is relatively small, or both. The exact meaning of relatively large or small is contextual,

which depends on the sampling rate, and various other characteristics of the time series.

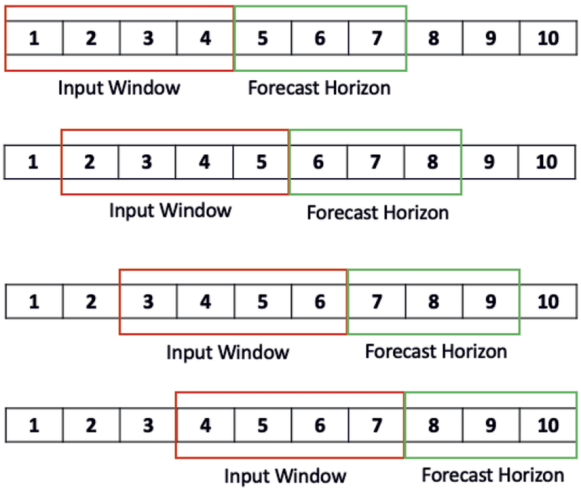


Figure 2.1: A visualization of the sliding window technique used to general input-target pairs $(\mathbf{x}, \mathbf{y})$ in time series forecasting. Taken from [20].

2.1.1 Stationarity of Time Series

Stationarity is an invariant property where statistical properties of a time series remain consistent over time. Stationarity can be classified as: *weak stationarity*, which only considers the covariance of a time series, and *strict stationarity*, which assumes distributions remain invariant over time. While classical models such as ARIMA [22] explicitly require stationarity to maintain their statistical properties, DL has introduced new perspectives on handling non-stationary data [76]. Traditional approaches necessitate explicit transformation of non-stationary data through techniques such as differencing or seasonal decomposition, whereas DL models may learn and adapt to certain types of non-stationarity through hierarchical feature learning capabilities. However, empirical evidence suggests that even for DL models, highly non-stationary distributions can

lead to suboptimal performance when compared to traditional models. Thus, characterizing the stationarity of a time series can improve the forecasting quality [110]. One standard method to test stationarity is the *Augmented Dickey-Fuller (ADF) test* which examines the presence of unit roots in time series using an autoregressive model estimated through ordinary least squares regression [105]:

$$x_t = c + w_1 t + w_2 x_{t-1} + \sum_{i=1}^{p} \phi_i \Delta x_{t-i} + \varepsilon_t \tag{2.1}$$

where x_t is the time series value at time t, c is the estimated drift constant, w_1 is the estimated time trend coefficient capturing deterministic trends, w_2 is the estimated process root determining persistence, ϕ_i are the estimated autoregressive coefficients capturing short-term dynamics, Δx_{t-i} represents lagged differences $(x_{t-i} - x_{t-i-1})$, and ε_t is white noise with zero mean and constant variance. The test evaluates the hypothesis $H_0 : w_2 = 1$ (non-stationary) against $H_a : w_2 < 1$ (stationary) with test statistic [105]:

$$\text{ADF} = \frac{\hat{w}_2 - 1}{\text{SE}(\hat{w}_2)} \tag{2.2}$$

where $\hat{w}_2$ is the estimated value of w_2 and $\text{SE}(\hat{w}_2)$ is its standard error measuring estimation uncertainty. A more negative test statistic provides stronger evidence for stationarity, with the additional terms in the regression controlling for trends and autocorrelation to ensure robust testing. In our analysis provided in Figure 3.7, ADF testing reveals distinct stationarity characteristics among different EMF datasets.

2.1.2 Conformal Prediction and Exchangeability in Time Series

The goal of CP is to provide a *prediction region*, Γ^α, for a given *significance level (or error rate)* $\alpha \in (0, 1)$, ensuring that the ground truth falls within this region with a probability of at least $(1 - \alpha)$. This framework's key advantage lies in its ability to produce valid uncertainty estimates for any black-box predictor, making it particularly valuable in domains where reliable uncertainty quantification is crucial [115, 128]. CP enables constructing prediction intervals with guaranteed finite-sample coverage under minimal assumptions on the data distribution, requiring only exchangeability of the data.

Exchangeability is a key statistical property where the order of time observations does not affect their joint probability—similar to how shuffling a well-mixed deck of cards does not change the probability of drawing any particular sequence. This property underpins CP's theoretical guarantees, but poses challenges for time series analysis. Time series data inherently violates exchangeability since future observations can depend on past states. In wireless networks, for instance, network utilization at time t can depend on its state at $t - 1$. To address this, several adaptations have been proposed, notably in [115], where the strict exchangeability assumption is relaxed through locally exchangeable windows. This approach preserves CP's validity while acknowledging the temporal nature of the data by treating similar historical patterns as locally exchangeable within defined contexts.

2.1.3 Fast-Fourier Transform and Periodicity

The Fast Fourier Transform (FFT) decomposes complex temporal patterns into their constituent frequency components, enabling efficient characterization of cyclical

behaviors within the data [21]. Mathematically, the FFT efficiently computes the Discrete Fourier Transform (DFT), which decomposes a time series $\mathbf{x} = (x_t)_{t=1}^{T}$ into its frequency components by:

$$X_k = \sum_{t=1}^{T} x_t e^{-2\pi i k t/T}, \quad k = 0, \ldots, T-1 \tag{2.3}$$

where the amplitude $|X_k|$ reveals the dominant frequency components (cyclical patterns) in the time series. By transforming time-domain signals into the frequency domain, FFT analysis reveals both the presence and strength of periodic patterns. This spectral decomposition proves particularly valuable in scenarios where multiple overlapping cycles may exist.

2.2 Deep Learning Methods in Time Series

Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is a type of feedforward neural network consisting of an input layer, one or more hidden layers, and an output layer, where each layer is fully connected to the next. The MLP can model complex, non-linear relationships by applying a non-linear activation function σ at each node in the hidden and output layers. Given an input time series $\mathbf{x} \in \mathbb{R}^L$, the transformation at the k-th hidden layer is defined as

$$\mathbf{h}^{(k)} = \sigma\left(\mathbf{W}^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}\right),$$

where each $\mathbf{W}^{(k)}$ is a learnable weight matrix and $\mathbf{b}^{(k)}$ is a learnable bias vector for layer k, with $\mathbf{h}^{(0)} = \mathbf{x}$. The output layer computes the predicted forecast $\hat{\mathbf{y}} \in \mathbb{R}^O$ as:

$$\hat{\mathbf{y}} = \sigma_{\text{out}} \left(\mathbf{W}^{(K+1)} \mathbf{h}^{(K)} + \mathbf{b}^{(K+1)} \right),$$

where K is the number of hidden layers, and σ_{out} is an activation function appropriate for the task, for example, softmax for time series classification or the identity function for time series forecasting. By stacking multiple layers and introducing nonlinearity, MLPs are capable of approximating complex functions, making them widely applicable in various machine learning and time series forecasting contexts.

2.2.1 Convolutional Neural Network (CNN)

A 1D Convolutional Neural Network (CNN) is a type of neural network designed to process sequential data, such as time series or one-dimensional signals, by learning spatial or temporal hierarchies through convolutional operations [15, 38, 69]. In a 1D CNN, the input signal $\mathbf{x} \in \mathbb{R}^L$ is convolved with a set of learnable filters (or kernels), producing feature maps that capture local patterns. The output of the j^{th} filter in layer k is given by:

$$\mathbf{h}_j^{(k)}[n] = \sigma \left(\sum_{m=1}^{M_k} \mathbf{w}_j^{(k)}[m] \mathbf{h}^{(k-1)}[n - m + 1] + b_j^{(k)} \right),$$

where $\mathbf{w}_j^{(k)}$ is the filter of size M_k, $b_j^{(k)}$ is the bias term, σ is a non-linear activation function, and n indexes the position in the sequence. The feature maps from one layer are passed to the next, allowing the network to progressively learn higher-level abstractions.

At the output layer, these learned features can be fed into a fully connected layer or another downstream task. The use of convolutional layers makes 1D CNNs particularly effective at capturing local dependencies in sequential data while reducing the number of parameters compared to fully connected architectures.

2.2.2 Long-Short Term Memory (LSTM) Networks

The *Long Short-Term Memory* (LSTM) network is a specialized recurrent neural network (RNN) designed to model long-term dependencies in sequential data by maintaining a memory cell $\mathbf{c}_t$ at each time step [14, 50, 101]. This structure mitigates vanishing and exploding gradient problems, making LSTMs effective for tasks such as time series forecasting and classification.

At each time step t, the input is a vector $\mathbf{x}_t \in \mathbb{R}^M$, where M is the number of input features or channels. The hidden state $\mathbf{a}_t \in \mathbb{R}^d$ serves as a summary of the information processed by the model up to time t, capturing both short-term patterns and contextual signals needed for immediate predictions. The memory cell $\mathbf{c}_t \in \mathbb{R}^d$, on the other hand, provides a more stable, long-term storage, selectively preserving information that might be important for future time steps. Together, the hidden state and memory cell allow the model to balance between remembering essential past information and responding to new inputs. Each gate (update, forget, and output) produces a vector $\Gamma \in \mathbb{R}^d$ that modulates the flow of information, ensuring that the model learns to focus on relevant parts of the sequence while discarding irrelevant or redundant details. This formulation enables the LSTM to handle both short-term dependencies (through $\mathbf{a}_t$) and long-term dependencies (through $\mathbf{c}_t$) effectively, which is crucial for sequential tasks such as time series forecasting.

Candidate memory cell and gates. The LSTM computes a *candidate memory cell* $\tilde{\mathbf{c}}_t$ as follows:

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_c[\mathbf{a}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c\right) \tag{1}$$

Here, $[\mathbf{a}_{t-1}, \mathbf{x}_t] \in \mathbb{R}^{d+M}$ denotes the concatenation of the previous hidden state $\mathbf{a}_{t-1}$ and the current input $\mathbf{x}_t$. The weight matrix $\mathbf{W}_c \in \mathbb{R}^{d\times(d+M)}$ maps the concatenated vector to the hidden size d, and $\mathbf{b}_c \in \mathbb{R}^d$ is the bias vector.

The gates are computed as:

- **Update Gate:** Controls how much of the candidate memory cell to incorporate into the current memory cell.

$$\Gamma_u = \sigma\left(\mathbf{W}_u[\mathbf{a}_{t-1}, \mathbf{x}_t] + \mathbf{b}_u\right) \tag{2.4}$$

  where $\mathbf{W}_u \in \mathbb{R}^{d\times(d+M)}$ and $\mathbf{b}_u \in \mathbb{R}^h$.

- **Forget Gate:** Determines how much of the previous memory cell to retain.

$$\Gamma_f = \sigma\left(\mathbf{W}_f[\mathbf{a}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f\right) \tag{2.5}$$

  where $\mathbf{W}_f \in \mathbb{R}^{d\times(d+M)}$ and $\mathbf{b}_f \in \mathbb{R}^h$.

- **Output Gate:** Controls the prominence of the new activation.

$$\Gamma_o = \sigma\left(\mathbf{W}_o[\mathbf{a}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\right) \tag{2.6}$$

  where $\mathbf{W}_o \in \mathbb{R}^{d\times(d+M)}$ and $\mathbf{b}_o \in \mathbb{R}^h$.

Memory and activation updates. The memory cell update and new hidden

state (activation) are computed as:

$$\mathbf{c}_t = \Gamma_u \odot \tilde{\mathbf{c}}_t + \Gamma_f \odot \mathbf{c}_{t-1} \tag{2.7}$$

$$\mathbf{a}_t = \Gamma_o \odot \tanh(\mathbf{c}_t) \tag{2.8}$$

Here, $\odot$ denotes element-wise multiplication.

Forecasting with LSTM networks. In time series forecasting, given the time series $\mathbf{x} = (\mathbf{x}_t)_{t=1}^{L}$, we first obtain the entire sequence of hidden states $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_L\}$ recursively. A common approach is to concatenate these hidden states:

$$\mathbf{a}_{\text{final}} = \text{concat}(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_L) \in \mathbb{R}^{d \times L} \tag{2.9}$$

This representation is then passed through a fully connected layer to generate the forecast:

$$\hat{\mathbf{y}} = \mathbf{W}_{\text{out}} \mathbf{a}_{\text{final}} + \mathbf{b}_{\text{out}} \tag{2.10}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{O \times dL}$ and $\mathbf{b}_{\text{out}} \in \mathbb{R}^{O}$ produce an output $\hat{\mathbf{y}} \in \mathbb{R}^{O}$, with O being the length of the forecast horizon.

2.2.3 The Transformer

The vanilla Transformer encoder. The *vanilla Transformer* is a versatile model architecture that uses self-attention mechanisms to capture dependencies across all time steps in a sequence, regardless of their temporal distance [118]. This makes it particularly suitable for time series forecasting, where both short-term and long-term

patterns can be critical. Unlike recurrent models such as LSTMs, the Transformer processes the entire input sequence in parallel, improving efficiency and handling long-range dependencies effectively. In this section, we focus on the *encoder-only* Transformer architecture, which is commonly used for time series forecasting, leaving out the decoder component.

Input and positional encoding. At each time step t, the input is a vector $\mathbf{x}_t \in \mathbb{R}^M$, where M is the number of input features or channels. The input sequence is represented as a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L]^T \in \mathbb{R}^{L \times M}$, where L is the input sequence length. Since the Transformer lacks an inherent notion of temporal order, a *positional encoding* vector $\mathbf{p}_t \in \mathbb{R}^d$ is added to each input vector to provide temporal context:

$$\mathbf{z}_t = \mathbf{W}_{\text{in}}\mathbf{x}_t + \mathbf{p}_t, \tag{2.11}$$

where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{d \times M}$ is a learned projection matrix, and d (or d_{model}) is the model's hidden dimension. This produces an initial sequence $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_L]^T \in \mathbb{R}^{L \times d}$.

The self-attention mechanism. The core of the Transformer encoder is the *self-attention* mechanism, which allows each time step to attend to all other time steps in the sequence. For each input $\mathbf{z}_t$, the model computes three vectors: the query $\mathbf{q}_t$, key $\mathbf{k}_t$, and value $\mathbf{v}_t$:

$$\mathbf{q}_t = \mathbf{W}_Q\mathbf{z}_t, \quad \mathbf{k}_t = \mathbf{W}_K\mathbf{z}_t, \quad \mathbf{v}_t = \mathbf{W}_V\mathbf{z}_t, \tag{2.12}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d}$ are learned projection matrices. The attention score

between time steps t and s is computed as the scaled dot product:

$$\alpha_{ts} = \frac{\mathbf{q}_t^T \mathbf{k}_s}{\sqrt{d}}. \tag{2.13}$$

These scores are normalized across all time steps using the softmax function:

$$\alpha_{t1}, \alpha_{t2}, \ldots, \alpha_{tL} = \text{softmax}\left(\frac{\mathbf{q}_t^T \mathbf{K}^T}{\sqrt{d}}\right), \tag{2.14}$$

where $\mathbf{K} = [\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_L]^T \in \mathbb{R}^{L \times d}$. The output of the attention mechanism for time step t is the weighted sum:

$$\text{Attention}(\mathbf{q}_t, \mathbf{K}, \mathbf{V}) = \sum_{s=1}^{L} \alpha_{ts} \mathbf{v}_s. \tag{2.15}$$

Encoder block design. The output of the self-attention mechanism is passed through a *multi-head attention* layer, which enhances the model's capacity by allowing multiple attention mechanisms to operate in parallel:

$$\text{MultiHead}(\mathbf{Z}) = \text{concat}(\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_h)\mathbf{W}_o, \tag{2.16}$$

where h is the number of attention heads, $\mathbf{H}_i \in \mathbb{R}^{L \times d_h}$ is the output of the i-th head, and $d_h = \frac{d}{h}$ is the dimensionality of each head. The matrix $\mathbf{W}_o \in \mathbb{R}^{d \times d}$ is a learned projection that maps the concatenated outputs of all heads back to the model's hidden dimension d. Each attention head $\mathbf{H}_i$ is computed independently:

$$\mathbf{H}_i = \text{Attention}(\mathbf{Z}\mathbf{W}_Q^{(i)}, \mathbf{Z}\mathbf{W}_K^{(i)}, \mathbf{Z}\mathbf{W}_V^{(i)}), \tag{2.17}$$

where $\mathbf{W}_Q^{(i)}, \mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)} \in \mathbb{R}^{d \times d_h}$ are the learned projection matrices for the i-th head.

Following multi-head attention, a *feedforward network* (FFN) is applied independently at each time step:

$$\text{FFN}(\mathbf{z}_t) = \sigma(\mathbf{W}_1 \mathbf{z}_t + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{2.18}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{ff}} \times d}$, $\mathbf{W}_2 \in \mathbb{R}^{d \times d_{\text{ff}}}$, and d_{ff} is the hidden dimension of the FFN. The Transformer encoder block also includes *layer normalization* and residual connections:

$$\mathbf{Z}' = \text{LayerNorm}(\mathbf{Z} + \text{MultiHead}(\mathbf{Z})), \tag{2.19}$$

$$\mathbf{Z}'' = \text{LayerNorm}(\mathbf{Z}' + \text{FFN}(\mathbf{Z}')). \tag{2.20}$$

Forecasting with Transformers. For time series forecasting, the final hidden representations $\mathbf{Z}_{\text{final}} = [\mathbf{z}_1'', \mathbf{z}_2'', \ldots, \mathbf{z}_L'']^T \in \mathbb{R}^{L \times d}$ from the last encoder block capture both local and global temporal dependencies. In sequence-to-sequence forecasting, each time step in the output sequence is predicted using a linear mapping applied to these representations:

$$\hat{\mathbf{Y}} = \mathbf{Z}_{\text{final}} \mathbf{W}_{\text{out}} + \mathbf{b}_{\text{out}}, \tag{2.21}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{d \times O}$ and $\mathbf{b}_{\text{out}} \in \mathbb{R}^O$. This produces the forecast $\hat{\mathbf{Y}} \in \mathbb{R}^{L \times O}$, where O is the number of features or prediction targets for each time step in the output sequence.

2.2.4 PatchTST

Patching. Time series forecasting attempts to model the relationships between data points from different time steps. A single time point does not equivocate to the level of semantic meaning of words within a sentence, similar to how individual letters do not carry high semantic information by themselves. In contrast, subseries-level patches enhances the model and captures higher semantic information, similar to modeling the relationship of words or subwords in a sentence [33]. However, most Transformer-based methods for time series do not rely on patching and patch-wise attention, and instead treat individual time points as tokens [27,62,73,74,127,137,139]. Instead of using individual time points as tokens, PatchTST treats patches as tokens to be fed into the standard Transformer architecture which greatly improves forecasting performance [88, 119].

Channel Independence. Given a multivariate time series, each variable or channel can either share information within the model (channel mixing), where the tokens input into a Transformer may come from several channels; or each channel can be processed independently, where the input to the model is derived only from a single channel (channel independence). This opens up several different possibilities for the design of time series Transformers, based on the content of the input tokens. In the context of time series, channel independence was proven to be effective with Convolutional Neural Networks (CNNs) [136] and linear models [134], but before the advent of PatchTST, it had not yet been applied to Transformer-based models. Given a time series $x = (x^{(1)}, \ldots, x^{(M)})^T \in \mathbb{R}^{M \times L}$ where M is the number of channels and L is the sequence length, PatchTST is applied to each channel $x^{(i)} \in \mathbb{R}^{1 \times L}$ independently for $1 \leq i \leq L$. Through ablation studies of comparing channel mixing and channel

independence, [88] displays that channel independence is the superior method for LTSF.

Model Architecture. Before discussing the experimental setup and results, we first provide an overview of the PatchTST architecture and the necessary background. Let $x = (x^{(1)}, \ldots, x^{(M)})^T \in \mathbb{R}^{M \times L}$ be a time series window with M channels and L time points. Here, x represents a single example in the batch, where all operations can be generalized by considering the tensor $x \in \mathbb{R}^{B \times M \times L}$ where B is the batch size. With channel independence, one can then view $(B \times M)$ as the batch dimension. For a single channel $1 \leq i \leq M$, the PatchTST pipeline is described below.

1. **RevIN**: $x^{(i)}$ is first normalized across the its channel by:

$$x_r^{(i)} = \texttt{RevIN}(x^{(i)}) = \gamma_i \Big(\frac{x^{(i)} - \mu_i}{\sigma_i} \Big) + \beta_i \tag{2.22}$$

   where $\gamma_i, \beta_i \in \mathbb{R}$ are learnable affine parameters, and μ_i and σ_i are the mean and standard deviation respectively for $x^{(i)}$. In practice, we add a small scalar for numerical stability in the denominator $\sigma_i \coloneqq \sigma_i + \varepsilon$ with $\varepsilon = 10^{-5}$. This method is referred to as reversible instance normalization (RevIN), and has shown to increase performance for time series forecasting [59]. At the final output of the network, we perform the inverse operation (RevOUT) of Equation (2.22), adding back the mean and multiplying the standard deviation into the channel output.

2. **Patching**: Let P be the patch length and S be the stride. Similar to a convolutional kernel, we slide a window of length P across each $x_r^{(i)}$ to create the patches, thus each $x_r^{(i)} \in \mathbb{R}^{1 \times L}$ is transformed into $x_p^{(i)} \in \mathbb{R}^{N \times P}$ where $N = \lfloor \frac{L-P}{2} \rfloor + 2$ is

the number of patches. We pad S repeated values to $x_r^{(i)}$ with the last value of $x_r^{(i)}$ before applying patching.

3. **Positional Encoding**: Let $W_p \in \mathbb{R}^{P \times D}$ and $W_{\text{pos}} \in \mathbb{R}^{N \times D}$ be learnable matrices. We then project $x_p^{(i)}$ with W_p before applying a learnable additive positional encoding W_{pos} by:

$$x_d^{(i)} = x_p^{(i)} W_p + W_{\text{pos}} \tag{2.23}$$

4. **Transformer Encoder**: Each $x_d^{(i)} \in \mathbb{R}^{N \times D}$ is ready to be fed into a Vanilla Transformer backbone independently [119], comprised of a sequence of Transformer blocks with embedding dimension D, treating N as the number tokens. Denote $z^{(i)} \in \mathbb{R}^{N \times D}$ as the output of the Transformer encoder.

5. **Linear Head**: We flatten $z^{(i)}$ to a vector of dimension $1 \times ND$ and apply a linear head $W_h \in \mathbb{R}^{ND \times T}$ where T is the length of the forecast window we wish to predict:

$$\hat{x}^{(i)} = \texttt{flatten}(z^{(i)}) W_h \tag{2.24}$$

where $\hat{x}^{(i)} \in \mathbb{R}^{1 \times T}$.

6. **RevOUT**: We then apply the inverse of Equation (2.22) to “denormalize" the

output, following the reversible instance normalization protocol [59]:

$$\hat{x}_r^{(i)} = \texttt{RevOUT}(\hat{x}^{(i)}) = \sigma_i\Big(\frac{\hat{x}^{(i)} - \beta_i}{\gamma_i}\Big) - \mu_i \tag{2.25}$$

where $\gamma_i, \beta_i, \mu_i, \sigma_i$ are defined in step 1. By construction, it follows that:

$$\texttt{RevOUT} \circ \texttt{RevIN} = \texttt{RevIN} \circ \texttt{RevOUT} = \text{id}$$

where id is the identity, thus we can view the Transformer as learning representations within the "normalized space" before being transformed back to the actual data distribution to form the prediction.

2.2.5 DLinear

DLinear is linear neural network that borrows ideas from statistical time series modelling, and applies seasonal-trend decomposition to the input time series $\mathbf{x}$ before processing it through two distinct linear transformations [134]. While Transformer-based methods have been the most prevalent architecture for time series forecasting, they can often struggle with the temporal information loss due to their permutation-invariant self-attention mechanism. DLinear, however, has demonstrated superior performance over many state-of-the-art Transformer-based models across standardized benchmarks, questioning of the reliability of Transformer-based approaches and highlighting the effectiveness of more simpler deep learning approaches for time series forecasting. The method for DLinear is as follows. Given a time series $\mathbf{x}$, we first pad $\mathbf{x}$ with m values on both sides of the series using its first and last values, extending

its length to $L + 2m$. We then define the moving average $\boldsymbol{\mu}_t$ as:

$$\boldsymbol{\mu}_t = \frac{1}{2m+1} \sum_{k=t-m}^{t+m} \mathbf{x}_k \tag{2.26}$$

for each $0 \leq t \leq L$, where $m \in \mathbb{N}$ is a hyperparameter. The local averaging window m smooths the input time series to form $(\boldsymbol{\mu}_t)_{t=1}^{L}$, which in practice should be tuned to balance trend detection and noise reduction. To obtain the seasonal component we subtract the mean at each timestep, i.e. $\mathbf{x}_{\text{season}} = (\mathbf{x}_t - \boldsymbol{\mu}_t)_{t=1}^{L}$, whereas the trend component is simply the moving average itself $\mathbf{x}_{\text{trend}} = (\boldsymbol{\mu}_t)_{t=1}^{L}$. Viewing both $\mathbf{x}_{\text{season}}, \mathbf{x}_{\text{trend}} \in \mathbb{R}^{L \times M}$ as matrices with L time steps and M channels, we apply the transformation:

$$\hat{\mathbf{y}} = \mathbf{W}_1 \, \mathbf{x}_{\text{trend}} + \mathbf{W}_2 \, \mathbf{x}_{\text{season}} \tag{2.27}$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{O \times L}$ are learnable matrices and $\hat{\mathbf{y}} \in \mathbb{R}^{O \times M}$ is the forecast. Note that operations from Equations (2.26)$-$(2.27) are both linear, thus the transformation $\hat{\mathbf{y}} = f_\theta(\mathbf{x})$ is linear where f_θ is the network, making it a unique case among neural networks.

2.2.6 TimesNet

TimesNet is a temporal convolutional network (TCN), proposed for time series classification, and anomaly detection, and and time series forecasting. It achieves this by first cropping different "views" of the signal, into different period lengths where these periods are determined by information obtained from the Fast Fourier Transform (FFT). After obtaining these views, they are passed through an inception block [55],

and then a dynamic weighting process is applied to each representation of each view, where the weights are obtained from the normalized amplitudes obtained by the FFT.

Denote the input time series as $\mathbf{x} = \mathbf{x}_{1\mathrm{D}} \in \mathbb{R}^{L \times M}$. TimesNet first applies the FFT independently to each column of $\mathbf{x}_{1\mathrm{D}}$ to obtain:

$$\mathbf{x}_{1\mathrm{D}}^{\mathrm{FFT}} = \mathrm{FFT}(\mathbf{x}_{1\mathrm{D}}) \in \mathbb{C}^{T \times C}, \tag{2.28}$$

The amplitudes of each FFT coefficient are obtained by:

$$\mathbf{a}_0 = \mathrm{Amp}(\mathbf{x}_{1\mathrm{D}}^{\mathrm{FFT}}) \in \mathbb{R}^{T \times C}, \tag{2.29}$$

and are then averaged along the channel dimension, giving us:

$$\mathbf{a} = \mathrm{Avg}(\mathbf{a}_0) \in \mathbb{R}^{T}. \tag{2.30}$$

We then select the indices from the Top$-k$ amplitudes in $\mathbf{a}$:

$$f_1, \ldots, f_k = \underset{f_* \in \{1, \ldots, \lceil \frac{L}{2} \rceil\}}{\mathrm{argTop}k}(\mathbf{a}), \tag{2.31}$$

We occlude the latter half of the sequence due to the conjugate symmetry of the coefficients, as they will have identical amplitudes. For each $1 \leq i \leq k$, the corresponding period is given by $p_i = \lceil \frac{L}{f_i} \rceil$. We summarize the process from Equations (2.28)-(2.31) by:

$$\mathbf{a}, \mathcal{F}, \mathcal{P} = \mathrm{Period}(\mathbf{x}_{1\mathrm{D}}) \tag{2.32}$$

where $F = \{f_1, \ldots, f_k\}$ and $P = \{p_1, \ldots, p_k\}$. For each pair (p_i, f_i), we slice the original signal $\mathbf{x}_{1\text{D}}$ along the temporal dimension into subsegments of length p_i, and pad $\mathbf{x}_{1\text{D}}$ appropriately so that p_i divides evenly into its total sequence length L. We then resize the original matrix into a 3D tensor given by:

$$\mathbf{x}_{2\text{D}}^{(i)} = \text{PeriodSlice}_{p_i, f_i}(\mathbf{x}_{1\text{D}}) \in \mathbb{R}^{p_i \times f_i \times M} \tag{2.33}$$

where PeriodSlice$(\cdot)$ denotes the operation of slicing the temporal dimension into subsegments of length p_i, which with padding, gives us a total of f_i segments, and repeat this for each channel independently. Repeating this for $1 \leq i \leq k$ gives us the reshaped signals $\mathbf{x}_{2\text{D}}^{(1)}, \ldots, \mathbf{x}_{2\text{D}}^{(k)}$, which each provide different views of the original signal along its temporal dimension. Thus, a row in $\mathbf{x}_{2\text{D}}^{(i)}$ captures *intraperiod* information, contained in one subsegment, whereas a row in $\mathbf{x}_{2\text{D}}^{(i)}$ captures *interperiod* information that spans over all periods at a given timepoint.

The pipeline of TimesNet is as follows. We first embed $\mathbf{x}_{1\text{D}}$ using:

$$\mathbf{x}_{1\text{D}}^{0} = \text{Embed}(\mathbf{x}_{1\text{D}}) \in \mathbb{R}^{L \times D} \tag{2.34}$$

where D is the model dimension. The l^{th} layer of TimesNet is given by the residual block:

$$\mathbf{x}_{1\text{D}}^{l} = \text{TimesBlock}(\mathbf{x}_{1\text{D}}^{l-1}) + \mathbf{x}_{1D}^{l-1} \tag{2.35}$$

where $\mathbf{x}_{1\text{D}}^{l} \in \mathbb{R}^{L \times D}$ for each $1 \leq l \leq N$, with total layers N. The description of

TimesBlock is given as follows. For each layer $1 \leq l \leq N$

$$\mathbf{a}^{l-1}, \mathcal{F}, \mathcal{P} = \text{Period}(\mathbf{a}^{l}_{\text{1D}}), \tag{2.36}$$

and for each $1 \leq i \leq k$:

$$\mathbf{a}^{l,i}_{2D} = \text{PeriodSlice}_{p_i,f_i}(\mathbf{x}_{\text{1D}}), \tag{2.37}$$

$$\hat{\mathbf{x}}^{l,i}_{\text{2D}} = \text{Inception}(\mathbf{x}^{l,i}_{\text{2D}}), \tag{2.38}$$

$$\hat{\mathbf{x}}^{l,i}_{\text{1D}} = \text{Trunc}(\text{Reshape}_{(p_i f_i, D)}(\hat{\mathbf{x}}^{l,i}_{\text{2D}})), \tag{2.39}$$

where $\mathbf{x}^{l,i}_{\text{2D}} \in \mathbb{R}^{p_i \times f_i \times D}$ is the period-sliced tensor. After the transformation, we feed each tensor into an inception block to obtain $\hat{\mathbf{x}}^{l,i}_{\text{2D}} \in \mathbb{R}^{p_i \times f_i \times D}$, reshape to $\hat{\mathbf{x}}^{l,i}_{\text{2D}} \in \mathbb{R}^{p_i f_i \times D}$ and then truncate to obtain $\hat{\mathbf{x}}^{l,i}_{\text{1D}} \in \mathbb{R}^{T \times D}$ by cutting off any previously padded values. We then apply the adaptive aggregation module defined by:

$$\hat{a}^{l-1}_{f_1}, \ldots, \hat{a}^{l-1}_{f_k} = \text{softmax}(a^{l-1}_{f_1}, \ldots, a^{l-1}_{f_k}) \tag{2.40}$$

where $a^{l-1}_{f_1}, \ldots, a^{l-1}_{f_k}$ are the Top$-k$ amplitudes from Equation (2.31), which are normalized using the softmax$(\cdot)$ function. These normalized amplitudes are used to weight the components $\hat{\mathbf{x}}^{l,1}_{\text{1D}}, \ldots, \hat{\mathbf{x}}^{l,k}_{\text{1D}} \in \mathbb{R}^{L \times D}$ obtained from Equation (2.39), within the weighted sum:

$$\mathbf{x}^{l}_{\text{1D}} = \sum_{i=1}^{k} \hat{a}^{l-1}_{f_i} \hat{\mathbf{x}}^{l,i}_{\text{1D}} \tag{2.41}$$

Amplitudes are often indicative of the relative importance of selected frequencies and

periods, thus, Equation (2.41), weights each component with respect to the importance of the tensor. This mechanism is inspired largely by [127]. By applying the sequence of residual TimesBlocks, we obtain the tensor $\mathbf{x}_{1\mathrm{D}}^{N} \in \mathbb{R}^{L \times D}$ which is then fed through a linear head to compute the prediction $\hat{\mathbf{y}} \in \mathbb{R}^{O \times M}$.

2.3 Background Work

In this section, we provide a review of existing DL models for time series forecasting and summarized their benefits and drawbacks in **Table 1**. The described DL models have been applied for a variety of wireless applications to date including EMF prediction [16, 61, 87, 92], network traffic prediction [30, 34, 47, 56, 89, 96, 112], channel prediction [112, 135], and network quality-of-service (QoS) prediction [28, 48].

Table 2.1: Comparison of Deep Learning Models for Time Series Forecasting in Wireless Networks

Model	Pros	Cons
MLP [16, 34, 61, 87, 89]	Simple architecture; Fewer parameters; Fast training/inference; Low computational cost; Adaptable to different inputs	Limited temporal modeling; Poor with long-range dependencies; Requires feature engineering; Overfits on complex EMF patterns; Cannot capture periodic patterns
CNN [5, 34, 87, 112, 138]	Captures local patterns effectively; Efficient parameter sharing; Detects signal hierarchies; Noise-robust; Parallel processing	Limited receptive field; Issues with irregular sampling; Weak global context modeling; Fixed kernel limitations; Requires careful design
LSTM [28, 30, 34, 56, 87, 92, 135]	Designed for long-term dependencies; Memory cell preserves patterns; Effective gating mechanisms; Handles variable sequences; Models complex dynamics	Sequential processing limits parallelization; Computationally expensive for long sequences; Gradient issues; Many parameters; Difficult interpretation
Transformer [47, 48, 87, 118]	Models dependencies at any distance; Parallel computation; Global context modeling; Multi-head attention; Scales to long sequences	Quadratic complexity; High memory usage; Requires positional encoding; No inherent sequential understanding; Weak with fine-grained patterns
PatchTST [88, 96]	Captures semantic information through patches; Channel independence reduces overfitting; Efficient multi-channel processing; Better noise handling; Superior time series performance	Sensitive to patch parameters; May lose fine-grained details; Parameter tuning challenges; Higher complexity than linear models; More complex implementation
DLinear [96, 134]	Effective seasonal-trend decomposition; Computationally efficient; Outperforms many Transformer models; Resistant to overfitting; Simpler architecture	Limited non-linear modeling; Fixed decomposition constraints; Lower expressiveness; Critical hyperparameter selection; Struggles with non-stationary data
EMForecaster (Proposed)	Patching with spatiotemporal mixing operations; Enhanced feature extraction with reversible normalization; Superior performance across diverse datasets; Balanced prediction intervals for conformal prediction; Effective for both short/long-term	More complex architectural design; Requires deeper understanding of the model; More challenging initial implementation; Requires adequate volume of data to generalize

2.3.1 EMF Analysis and Forecasting

To date, a variety of research studies have focused on EMF exposure modeling and resource optimization in cellular networks using analytical or simulation-based methods. For instance, stochastic geometry tools such as Poisson Point Processes (PPP) and shot-noise processes have been considered to model the spatial distribution of EMF sources and analyze their impact on EMF exposure and coverage in cellular networks [6,23,40–43,86,97,125]. On the other hand, several studies have focused on minimizing EMF exposure in cellular systems [108], [57]. Resource management schemes, including user-scheduling [107], spectrum allocation [108], multi-cell scheduling [109], user association [81], beamforming optimization [124,131], and cross-layer protocols [35,95] have also been considered to minimize EMF exposure. Network-based approaches considered minimizing EMF exposure by optimizing cellular network planning, as discussed in [9,25,53,80,82,91].

Nevertheless, the aforementioned model-based approaches often suffer from inaccuracies due to necessary modeling assumptions, making them less applicable in complex real-world environments. Developing analytical models for such scenarios is not only intractable, but also computationally demanding, as they result in intricate mathematical expressions. Moreover, these methods are constrained by specific wireless channel assumptions, limiting their applicability to dynamic and unpredictable network conditions. Additionally, they lack the ability to learn from historical patterns or trends, preventing them from effectively performing proactive network resource management and planning. As a result, their practicality in real deployments remains a significant challenge.

Recently, deep learning (DL) approaches have gained significant attention in time

series forecasting, an area historically dominated by statistical models and traditional machine learning methods [83]. Traditional techniques, such as Autoregressive Integrated Moving Average (ARIMA) [22] combines differencing, autoregression, and moving average components to model linear relationships in stationary time series data [22], and has long been a standard benchmark in time series modeling, often outperforming more modern methods in standardized testing environments. However, the advent of sophisticated DL architectures, such as transformers, has begun to shift the paradigm [83]. DL methods offer distinct advantages over conventional forecasting methods. These models can learn complex nonlinear relationships without explicit feature engineering and adapt to dynamic changes by modeling hierarchical structures. Unlike traditional methods that require manual pre-processing such as seasonal decomposition, DL models can process raw time series data directly while handling multiple input variables and their interactions.

DL methods are increasingly being adopted for EMF pattern analysis and prediction due to their ability to handle complex, nonlinear relationships in the data. In [61], Kiouvrekis *et al.* employed hierarchical clustering to analyze EMF measurements across 205 schools in Thessaly, Greece. The findings reveal that EMF exposure patterns were independent of urban population density in the 27 MHz–3 GHz range. Bakcan *et al.* integrated static electric field measurements with artificial neural networks (ANN) in [16] to predict EMF exposure at unmeasured locations across a campus setting, validating results against Information and Communication Technology Agency (ICTA) and ICNIRP standards. Focusing on temporal prediction, Pala *et al.* in [92] analyzed EMF dataset comprised of 60 monthly mean values in the range of 1 Hz

- 400 kHz using the Wavecontrol SMP2 device, and demonstrated that long short-term memory (LSTM) outperforms recurrent neural network (RNN) and traditional statistical approaches. However, the model was based on a very limited dataset for both training and testing, raising concerns about overfitting. Moreover, the study did not consider fundamental time series parameters such as *lookback window* (how much historical data to use), *forecast horizon* (how far ahead to predict), and *rolling window stride* (how to slide the prediction window through time). Furthermore, the dataset's frequency range was significantly low to encompass modern wireless networks.

Very recently, Nguyen *et al.* in [87] demonstrated the effectiveness of Transformers [118] over the core DL architectures, including the multilayer perceptron (MLP), LSTM, and convolutional neural networks (CNNs) for short-term EMF forecasting using the data provided in [66]. The training data was collected from real measurements taken in a city of Ordu, Turkey [66], considering multi-step input and output sequences. Building upon the pioneering application of DL to EMF time series forecasting in [87], we note that the framework is restricted to short-term prediction windows (maximum duration of 20 minutes). In addition, although the dataset was carefully collected with 24-hour recordings from 17 sites at 15-second intervals, it is relatively modest by DL standards, where larger datasets typically enable more robust performance evaluation. The limited temporal scope of the data presents challenges in capturing multi-day patterns and long-term exposure trends, as the resulting non-stationary time series limits model predictive capabilities. Furthermore, our analysis in Section V reveals high correlation in the data, suggesting limited diversity in EMF exposure scenarios, which may impact the ability to assess each model's generalizability. Moreover, the work employed classical DL architectures, not specifically designed for temporal modeling.

Finally, although hyperparameter values were documented, the absence of ablation studies and sensitivity analyses makes it difficult to fully understand each model's behavior and validate design choices.

2.3.2 Seizure Onset Zone Localization

As mentioned in Section 1.1.1, proper treatment of variable-length sequences, particularly in SOZ localization is incredibly important for machine learning modeling. To better understand the current landscape of variable-length time series modeling, we provide a formal treatment of finite-context and infinite-context methods, and review state-of-the-art (SOTA) methods within each category.

Definition. *Let $\mathcal{X}$ be a vector space over $\mathbb{R}$ and $f_\theta : \mathcal{X} \to \mathcal{Y}$ be a model with parameters θ and output space $\mathcal{Y}$. We say that f_θ has **finite-context** if $\mathcal{X}$ is finite-dimensional, i.e., there exists some $n \in \mathbb{N}$ such that there is an isomorphism $\mathcal{X} \cong \mathbb{R}^n$ as vector spaces. f_θ is said to have **infinite-context** if $\mathcal{X} = \mathbb{R}^{(\infty)}$ is the space of real number sequences with finite support*[2]*.*

Note that this definition refers to the *native* capabilities of f_θ, without the usage of data manipulation techniques such as padding, truncation, and interpolation. We utilize this formalization to classify our baselines, along with their respective advantages and limitations.

Finite-context methods. Finite-context methods are among the most commonly used approaches for TSC. Transformer-based models have gained significant attention, with variations such as sparse attention, series decomposition, and patching techniques [75, 88, 119, 127]. Several temporal convolutional networks (TCNs) have

[2]Every sequence in $\mathbb{R}^{(\infty)}$ must have finitely many non-zero terms.

also been proposed, to capture temporal dependencies through dilated convolutions and Inception-like architectures [55, 67, 79, 126]. Recently, multilayer perceptrons (MLPs) and simple linear models have demonstrated competitive performance [24,134]. However, finite-context methods struggle with variable-length sequences, requiring padding, truncation, or interpolation for VTSC. Additionally, longer sequences increase the number of model parameters, leading to both higher computational costs and increased risk of overfitting.

Infinite-context methods. The recurrent neural network (RNN) family includes several models capable of ingesting variable-length time series [101]. Long-short term memory (LSTM) networks introduce memory cells and gating mechanisms to better handle long-term dependencies [50]. Gated recurrent units (GRUs) simplify the LSTM architecture, while maintaining similar performance [14]. State space models (SSMs) have gained recent attention, with approaches such as structured state space for sequence modeling (S4), designed to efficiently capture long-range dependencies in sequential data and introducing structured parameterization to enable efficient computation over long sequences [46]. Building on this, recently Mamba introduces a selective SSM that adapts to input dynamics, further improving processing of long-range dependencies in time series data, while maintaining linear time complexity with respect to sequence length [45]. Despite these advancements, RNNs and SSMs struggle with retaining information in extremely long sequences and are prone to vanishing or exploding gradients [106]. In this context, ROCKET offers an alternative, i.e., using random convolutional kernels to convert input into a fixed-length representation for VTSC, but at the cost of interpretability and potentially limited model expressivity [31].

SOZ localization methods. To date, a variety of electrical stimulation methods

that use intracranial electrodes [58, 129] have been considered to enhance localization accuracy through induced responses analyzed by TCNs and logistic regression models. Yet, these approaches require both fixed-length windows and the use of active stimulation.

Several recent proposals have been tailored specifically to SOZ localization using channel (electrode) dependent models [44], while other recent works have explored the notion of *channel independence*. Below we formally define channel dependence and independence.

Definition (Channel Dependence)**.** *For a time series* $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_L)^T \in \mathbb{R}^{L \times M}$*, with* L *time points and* M *channels, a model* $f : \mathbb{R}^L \to \mathcal{Z}$ *is said to be channel independent if for each channel* $\mathbf{x}_i$*,* f *processes it separately to obtain a representation* $\mathbf{z}_i = f(\mathbf{x}_i)$*. This representation then may be used for the relevant task at hand (e.g., fed through a sigmoid function). This is in contrast to a channel dependent model* $f : \mathbb{R}^{L \times M} \to \mathcal{Y}$*, which processes all channels simultaneously, to obtain a single representation* $\mathbf{z} = f(\mathbf{x})$*.*

In the context of SOZ localization, channel independent methods map each $\mathbf{x}_i$ independently to a representation, which is then mapped to the probability of the electrode belonging to the SOZ. However, channel dependent approaches consider all electrodes simultaneously to obtain the final representation. The most common implementation of channel dependence uses graph neural networks (GNNs) which model the relationships between electrodes and the functional connectivity of the brain through message passing mechanisms [120]. A functional connectivity graph is generated by computing pairwise metrics (e.g., correlation) between different electrodes, which creates the initial structure of the graph (e.g., by threshold certain metrics) and optional edge features [37, 44].

Chapter 3

EMForecaster

3.1 Chapter Contributions

None of the existing research has developed a reliable DL-empowered time series forecasting framework for wireless cellular network applications in general, nor specifically for EMF forecasting. The black-box nature of DL models makes it difficult to interpret predictions or understand the underlying decision-making process, raising concerns in applications where transparency and reliability are paramount. To this end, our contributions are listed as follows:

- We develop a novel DL-driven time series forecasting solution, referred to as *EMForecaster*, to forecast EMF patterns considering both long-term and short-term time series, across a variety of locations with a forecast horizon of up to 50 hours. EMForecaster exploits several modern DL techniques such as Reversible Instance Normalization (RevIN), which minimizes the impact of distribution shifts or non-stationarity in the EMF data, in addition to a patching and patch-embedding module. EMForecaster then applies the spatiotemporal backbone (STB) to the patch-embedded data where we employ a mixing operation on the learned patch representations,

enabling learning of hierarchical patterns at multiple scales.

• We augment the *EMForecaster* with a CP framework to ensure that the ground truth lies within a certain prediction interval with *error rate* α. The performance is measured in terms of the independent coverage (IC) of each individual forecast point, the joint coverage (JC) of the forecast points across the forecasted window, and mean prediction interval width (MIW). Unlike traditional uncertainty quantification methods, which often depend on strong distributional assumptions, CP is agnostic to the original data distribution, making it compatible with any DL model and dataset.

• We propose the Trade-off Score (TOS)—a unified metric that evaluates the effectiveness of CP methods by considering two key aspects: (1) how often the true values fall within the predicted ranges (coverage) and (2) how wide these predicted ranges are (interval width). TOS quantifies how well a method balances this fundamental trade-off. While wider prediction ranges are more likely to contain the true values, they provide less precise and thus less useful forecasts. A higher TOS indicates better performance, as the true values fall within the prediction range without requiring excessively wide intervals.

• Our experiments, covering a wide variety of environments and forecast horizons, display EMForecaster's capabilities over the existing DL models. In point forecasting, EMForecaster demonstrates improvements of 53.97% over the Transformer architecture and 38.44% over the average of all baseline models. When compared to the DLinear architecture, EMForecaster maintains 8.01% gains. For conformal forecasting, EMForecaster provides an excellent balance between prediction interval coverage and width, as measured by the TOS. This balance is comparable to DLinear's performance while showing marked improvements of 24.73% over the average baseline and 49.17%

over the Transformer architecture.

• We conduct a comprehensive EMF data analysis by applying spectral decomposition, stationarity testing, and spatial correlation analysis on both short-term and long-term EMF exposure datasets. Stationarity is particularly crucial, which implies that time windows are reasonably independently and identically distributed (i.i.d.), therefore satisfying the exchangeability condition necessary for conformal prediction (see Section II.C for more details).

3.2 Chapter Organization

The remainder of this chapter is organized as follows. Section 3.3 presents our proposed DL model, EMForecaster, followed by conformal prediction for uncertainty quantification in Section 3.4. Section 3.5 details the datasets, experimental setup, and comprehensive data analysis. Section 3.6 presents the baseline models, results, and discussion. Finally, Section 3.7 concludes the chapter.

3.3 Proposed EMForecaster

Given a univariate time series representing EMF exposure over time, we detail the pre-processing and the architecture of the proposed EMForecaster [85], shown in Figure 3.1. EMForecaster incorporates RevIN, hierarchical patching, and a mixing operation to process temporal patterns at multiple scales. By decomposing the input signal into localized patches, the architecture can efficiently capture both fine-grained temporal dynamics and long-range dependencies, enabling effective modeling compared to traditional approaches that process the entire sequence at once. In following subsections, we describe each component of the EMForecaster following the

sequence of the pipeline shown in Figure 3.1.

3.3.1 EMF Time Series Preprocessing

Consider a time series $\mathbf{x} = (x_t)_{t=1}^{T}$, where $x_t \in \mathbb{R}$, $\forall t$. To eliminate outliers, we set a carefully selected threshold $\delta > 0$, where x_t is replaced with a linear interpolation of its neighbors, x_{t-1} and x_{t+1}. We then partition $(x_t)_{t=1}^{T}$ into training, validation, and test datasets. We obtain the mean and standard deviation from the training set, and compute $z-$score to normalize the training, validation, and test datasets. Next, we apply the sliding window method with a stride of 1, to obtain window pairs of size L and O, representing the input and target, respectively. Thus, we obtain our dataset of n input-output pairs, $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$ where $\mathbf{x}_i \in \mathbb{R}^L$ is the lookback window (input), and $\mathbf{y}_i \in \mathbb{R}^O$ is its continuation (target), for any sequence i.

3.3.2 Reversible Instance Normalization (RevIN)

We first apply RevIN to the input time series window $\mathbf{x}$ before feeding it into any temporal modules [59].

3.3.3 Patching and Embedding

Inspired from the PatchTST and Vision Transformer (ViT) methods [36, 88], we consider a patching mechanism along with a transformer encoder backbone. Given the demonstrated effectiveness of patching in time series forecasting compared to non-patching methods [70, 73], we customize this mechanism for our architecture. After passing the input time series $\mathbf{x}$ to obtain $\mathbf{x}^{(r)} = \text{RevIN}(\mathbf{x})$, we apply a patching module to split up the sequence into equally-sized contiguous subsequences. Given a

pre-determined patch dimension P, and patch stride S—similar to a convolutional kernel—we obtain $N = \lfloor \frac{L-P}{S} \rfloor + 1$ patches in total:

$$\mathbf{x}^{(p)} = \text{Patcher}(\mathbf{x}^{(r)}), \tag{3.1}$$

where $\mathbf{x}^{(p)} \in \mathbb{R}^{N \times P}$. Before applying patching, we pad $\mathbf{x}^{(r)}$ with zeros at the end of the time series to ensure there are enough positions to extract all N complete patches of size P. After patching, we perform patch embedding defined by:

$$\begin{aligned} \mathbf{x}^{(d)} &= \text{PatchEmbed}(\mathbf{x}^{(p)}) && (3.2)\\ &= [\mathbf{W}_d(\mathbf{x}_1^{(p)}) \;\; \mathbf{W}_d(\mathbf{x}_2^{(p)}) \cdots \mathbf{W}_d(\mathbf{x}_N^{(p)})]^T && (3.3)\\ &= \mathbf{x}^{(p)}\, \mathbf{W}_d^T, && (3.4) \end{aligned}$$

where $\mathbf{x}_i^{(p)} \in \mathbb{R}^P$ is the i^{th} row of $\mathbf{x}^{(p)}$, $\mathbf{x}^{(d)} \in \mathbb{R}^{N \times D}$ is the resultant embedding, and $\mathbf{W}_d \in \mathbb{R}^{D \times P}$ is a learnable weight matrix which embeds each patch of size P to a (typically larger) dimension of size D, which we refer to as the patch embedding dimension. This is in contrast to several other embedding mechanisms found in other DL time series models, which may apply an embedding along the sequence dimension with $\mathbf{x}^{(d)} = \mathbf{W}_d\, \mathbf{x}^{(p)}$ with $\mathbf{W}_d \in \mathbb{R}^{D \times N}$. The patch-wise embedding approach learns local temporal patterns while maintaining parameter efficiency, as the embedding weights are shared across all patches and independent of the sequence length. Selecting an appropriate patch dimension P and patch embedding dimension D for the task is critical, as P controls the temporal receptive field of local pattern learning, while D determines the richness of the learned representations and the model's capacity to capture complex temporal behavior.

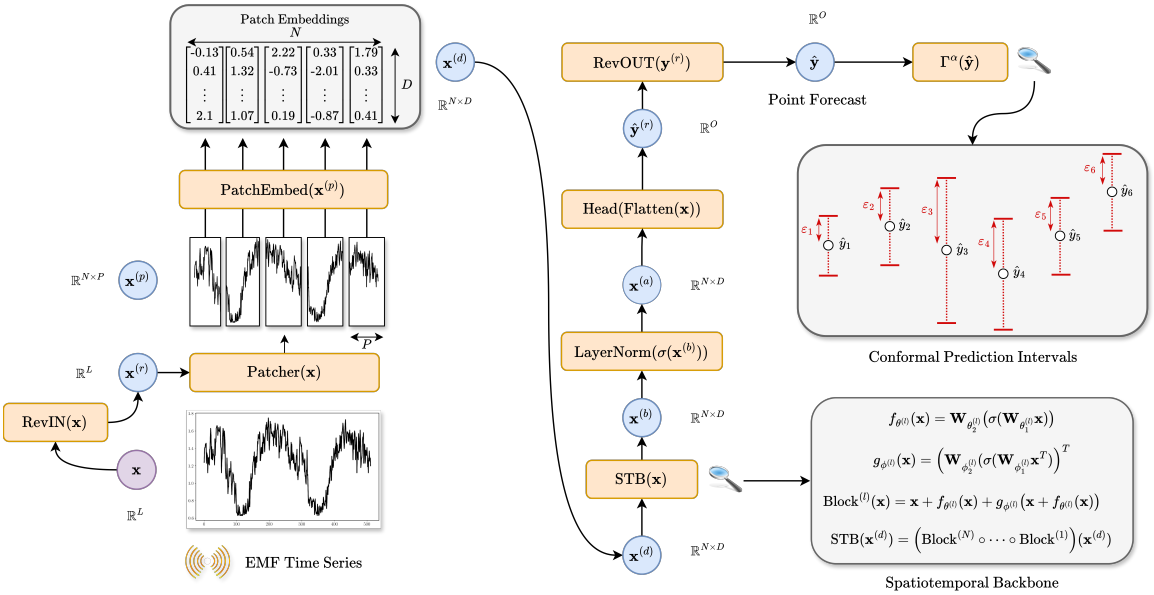


Figure 3.1: The EMForecaster with Conformal Prediction: An EMF input time series $\mathbf{x} \in \mathbb{R}^L$ is processed by Reversible Instance Normalization (RevIN), before being patched to obtain $\mathbf{x}^{(p)} \in \mathbb{R}^{N \times P}$, where each patch of length P is embedded independently to obtain $\mathbf{x}^{(d)} \in \mathbb{R}^{N \times D}$. $\mathbf{x}^{(d)}$ is then fed through the Spatiotemporal Backbone (STB), which applies separate MLPs to the temporal domain and spatial domain of the patch-embedded input. The STB outputs a representation $\mathbf{x}^{(b)} \in \mathbb{R}^{N \times D}$, where we apply a nonlinear activation σ and layer normalization, before flattening the representation to a vector and applying a linear head to obtain the output $\hat{\mathbf{y}}^{(r)} \in \mathbb{R}^O$. The output $\hat{\mathbf{y}}^{(r)}$ is further refined by RevOUT (the inverse of RevIN) which adjusts $\hat{\mathbf{y}}^{(r)}$ to the appropriate distribution to obtain the point forecast $\hat{\mathbf{y}}$. The conformal predictor γ^{α} then outputs a conformal interval for each of the O forecasted points in $\hat{\mathbf{y}} \in \mathbb{R}^O$.

3.3.4 Spatiotemporal Backbone (STB)

MLPMixer, a model proposed initially for computer vision, processes information through two distinct MLP operations: one that mixes information across tokens and another that mixes features within each token [117]. MLPMixer demonstrated that MLP-based architectures could achieve state-of-the-art performance in computer vision, even when compared to several modern Transformer variants [36], challenging the fact that convolutions or self-attention mechanisms are necessary for visual processing.

TSMixer adapts the MLPMixer architecture for time series by utilizing two distinct MLP operations per block: one that mixes information across temporal locations and another that mixes features within each time step. For a layer l, the temporal MLP $f_{\theta^{(l)}} : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$ is defined as:

$$f_{\theta^{(l)}}(\mathbf{u}) = \mathbf{W}_{\theta_2^{(l)}} \big(\sigma(\mathbf{W}_{\theta_1^{(l)}}\, \mathbf{u})\big), \tag{3.5}$$

for any $\mathbf{u} \in \mathbb{R}^{N\times D}$, where $\mathbf{W}_{\theta_1^{(l)}} \in \mathbb{R}^{D_h\times N}$ and $\mathbf{W}_{\theta_2^{(l)}} \in \mathbb{R}^{N\times D_h}$ are learnable matrices with parameters $\theta_1^{(l)}$ and $\theta_2^{(l)}$ respectively, σ is a nonlinearity (e.g., ReLU), and D_h is the hidden dimension of the MLP. Similarly, the patch MLP $g_{\phi^{(l)}} : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$ is defined as:

$$g_{\phi^{(l)}}(\mathbf{u}) = \Big(\mathbf{W}_{\phi_2^{(l)}}(\sigma(\mathbf{W}_{\phi_1^{(l)}}\, \mathbf{u}^T)\Big)^T, \tag{3.6}$$

for any $\mathbf{x} \in \mathbb{R}^{N\times D}$, where $\mathbf{W}_{\phi^{(l)}} \in \mathbb{R}^{D_h\times D}$ and $\mathbf{W}_{\phi_2^{(l)}} \in \mathbb{R}^{D\times D_h}$ are learnable matrices with parameters $\phi_1^{(l)}$ and $\phi_2^{(l)}$, respectively. The hidden dimension D_h and nonlinearity σ are the same as in the temporal MLP.

We provide the patch-embedded time series $\mathbf{x}^{(d)} \in \mathbb{R}^{N\times D}$ as an input to STB composed of K blocks:

$$\mathbf{x}^{(b)} = \mathrm{STB}(\mathbf{x}^{(d)}) = \Big(\mathrm{Block}^{(K)} \circ \cdots \circ \mathrm{Block}^{(1)}\Big)\,(\mathbf{x}^{(d)}). \tag{3.7}$$

where the l^{th} block is modeled as:

$$\mathrm{Block}^{(l)}(\mathbf{u}) = \mathbf{u} + f_{\theta^{(l)}}(\mathbf{u}) + g_{\phi^{(l)}}\big(\mathbf{u} + f_{\theta^{(l)}}(\mathbf{u})\big). \tag{3.8}$$

The above is the explicit form of composing $f_{\theta^{(l)}}$ and $g_{\phi^{(l)}}$ with two residual connections [49], which can be written equivalently as follows:

$$\mathbf{u}' = \mathbf{u} + f_{\theta^{(l)}}(\mathbf{u}), \tag{3.9}$$

$$\text{Block}^{(l)}(\mathbf{x}) = \mathbf{u}' + g_{\phi^{(l)}}(\mathbf{u}'). \tag{3.10}$$

where, $f_{\theta^{(l)}}$ "mixes" the temporal dimension N and $g_{\phi^{(l)}}$ "mixes" the embedding dimension D.

Applying the STB to patch-embedded data represents a novel fusion where the mixer operates on learned patch representations rather than raw temporal slices, enabling the capture of hierarchical patterns at multiple scales. The combination of patch embeddings and mixing operations allows the model to learn relationships between these higher-level temporal abstractions while maintaining the architectural simplicity of MLP-based mixing. We then pass the output of the STB through a nonlinear activation σ and a layer normalization module [71], i.e., $\mathbf{x}^{(a)} = \text{LayerNorm}(\sigma(\mathbf{x}^{(b)}))$, and flatten $\mathbf{x}^{(a)}$ to $\mathbf{x}^{(f)} = \text{Flatten}(\mathbf{x}^{(a)})$, where $\mathbf{x}^{(f)} \in \mathbb{R}^{(ND)}$, in which we obtain the candidate forecast by:

$$\hat{\mathbf{y}}^{(r)} = \text{Head}(\mathbf{x}^{(f)}) = \mathbf{W}_{\text{head}}\, \mathbf{x}^{(f)}, \tag{3.11}$$

$$\text{where} \quad \mathbf{x}^{(f)} = \text{Flatten}(\text{LayerNorm}(\sigma(\mathbf{x}^{(b)}))), \tag{3.12}$$

where $\mathbf{W}_{\text{head}} \in \mathbb{R}^{O \times (ND)}$ is a learnable weight matrix, and $\hat{\mathbf{y}}^{(r)} \in \mathbb{R}^{O}$ is the candidate forecast, which is refined to obtain the final forecast by $\hat{\mathbf{y}} = \text{RevIN}^{-1}(\hat{\mathbf{y}}^{(r)})$ given by (2.25).

3.4 Conformal Time Series Forecasting

CP is a powerful framework for uncertainty quantification that provides rigorous statistical guarantees [12]. The goal of CP is to provide a *prediction region*, Γ^α, for a given *significance level (or error rate)* $\alpha \in (0, 1)$, ensuring that the true outcome falls within this region with a probability of at least $(1 - \alpha)$. Most CP approaches rely on an *inductive approach*, where predictions are generated using a combination of an *underlying model* f and an additional *calibration set*—a technique referred to as *inductive conformal prediction (ICP)* [10,93]. Rather than providing point predictions, such as a single numerical value in regression, models calibrated with ICP generate a continuous interval in which ground truth is theoretically guaranteed to be contained with probability $1 - \alpha$.

In this section, we first describe CP in the context of regression with functions of the form $f : \mathbb{R}^L \to \mathbb{R}$. We then discuss the extension of CP to regression functions of the form $f : \mathbb{R}^L \to \mathbb{R}^O$ to predict multiple future time points simultaneously. Next, we discuss the evaluation metrics and the proposed metric referred to as Trade-off Score (TOS).

3.4.1 CP: Single Time-Step Forecast

While CP has been widely studied, its application to time series forecasting was limited due to the *exchangeability* condition which requires that any reordering of the dataset is equiprobable.

Definition (Exchangeability)**.** *Given a dataset with n observations $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n \subseteq \mathbb{R}^T \times \mathbb{R}$, we say that it is exchangeable if any of its $n!$ permutations are equiprobable. Note that independent and identically distributed (i.i.d.) observations satisfy exchangeability.*

Given the exchangeability, the trustworthiness or reliability of a conformal predictor is described by the conformal coverage guarantee defined below.

Property 1 (Conformal coverage guarantee)**.** *Under the exchangeability assumption, any conformal predictor will return the prediction region* $\Gamma^{\alpha}(\mathbf{x}^{(i)})$ *such that the probability of error* $y^{(l+1)} \notin \Gamma^{\alpha}(\mathbf{x}^{(l+1)})$ *is not greater than* α*, that is:*

$$\mathbb{P}[y^{(l+1)} \in \Gamma^{\alpha}(\mathbf{x}^{(l+1)}) \mid \mathcal{D}] \geq 1 - \alpha. \tag{3.13}$$

Note that the exchangeability assumptions and validity properties are *distribution-free*; that is, there are no required distributional assumptions on the underlying data $\mathcal{D}$, and applies to any underlying predictive model so long as the exchangeability assumption is satisfied on $\mathcal{D}$.

Time series data, however, exhibits temporal dependencies, which inherently violating the exchangeability condition in most scenarios. As a result, directly applying CP to forecast intervals in time series data may not provide theoretical guarantees. Recent advancements have demonstrated that the exchangeability assumption can be relaxed through problem reframing this to the exchangeability of windows, allowing CP to be effectively applied to time series tasks [115, 128].

The inductive variant of CP divides the dataset into two subsets: a proper training set and *calibration set*, such that $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{cal}}$. Denote $n = |\,\mathcal{D}\,|$ and $m = |\,\mathcal{D}_{\text{cal}}\,|$. The training set is used to fit the underlying model f, while the calibration set is utilized to compute the *nonconformity scores* $\eta(\mathcal{D}, (\mathbf{x}^{(i)}, y^{(i)}))$ for all $(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{D}_{\text{cal}}$, which measures the distribution errors on the underlying model. CP guarantees validity for any choice of nonconformity score, even random ones, but in regression

settings, a commonly used score is defined as:

$$R_i = \eta(\mathcal{D}, (\mathbf{x}^{(i)}, y^{(i)})) = d\big(f(\mathbf{x}^{(i)} \mid \mathcal{D}), y^{(i)}\big), \tag{3.14}$$

where $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a distance metric. While the choice of d is flexible, it should reflect the problem's objectives with respect to the dataset and task. The residuals $\{R_i\}_{i=1}^{m}$, computed from the calibration set, form an empirical distribution. The *critical nonconformity score*, $\hat{\epsilon}$, is selected as the $(1-\alpha)$-quantile of this distribution. To account for finite-sample effects, a correction is applied by considering the $\lceil (m+1)(1-\alpha) \rceil$-th smallest residual. For any new input $\mathbf{x}^{(n+1)}$, the prediction interval is defined by:

$$\Gamma^{\alpha}(\mathbf{x}^{(n+1)}) = [\hat{y}^{(n+1)} - \hat{\epsilon}, \hat{y}^{(n+1)} + \hat{\epsilon}], \tag{3.15}$$

where $\hat{y}^{(n+1)} = f(\mathbf{x}^{(n+1)})$ is the model prediction.

Theorem 1 (Vovk, Gammerman, and Saunders [122])**.** *Suppose* Γ^{α} *is defined as in Equation (3.15), then* Γ^{α} *is a conformal predictor and Property (1) is satisfied.*

3.4.2 CP: Multiple Time-Step Forecast

In univariate time series forecasting with horizon $O > 1$ and underlying model $f : \mathbb{R}^{L} \to \mathbb{R}^{O}$, our goal is to generate prediction intervals for each future time point. Specifically, for a model that produces predictions $f(\mathbf{x}^{(n+1)}) = \hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_O)^T$, we must construct intervals as follows:

$$[\hat{y}_t^{(n+1)} - \hat{\varepsilon}_t, \;\; \hat{y}_t^{(n+1)} + \hat{\varepsilon}_t], \;\; 1 \leq t \leq O.$$

For any new observation $(\mathbf{x}^{(n+1)}, \mathbf{y}^{(n+1)})$ where $\mathbf{y}^{(n+1)} = (y_1^{(n+1)}, \ldots, y_O^{(n+1)})^T$, and any time step $1 \leq t \leq O$, these intervals should satisfy the following coverage guarantee similar to Property (1):

$$\mathbb{P}\big[y_t^{(n+1)} \in [\hat{y}_t^{(n+1)} - \hat{\varepsilon}_t, \hat{y}_t^{(n+1)} + \hat{\varepsilon}_t]\big] \geq 1 - \alpha \tag{3.16}$$

For any example $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, the nonconformity score can be generalized to:

$$R_i = \left[|y_1^{(i)} - \hat{y}_1^{(i)}|, \ldots, |y_O^{(i)} - \hat{y}_O^{(i)}|\right] \tag{3.17}$$

In [115], the authors indicate that the only restriction needed is that the underlying model must map to all O predictions simultaneously, rather than recursively, such as in Bayesian forecasting methods. Given this constraint, as all O predicted values are obtained from the same representation, we employ a Bonferroni correction when obtaining the critical nonconformity scores, which are obtained by considering the $\lceil (m+1)(1-\alpha/O) \rceil$-th smallest residuals $\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_O$ from each score distribution. By defining:

$$\Gamma_t^{\alpha}(\hat{y}_t) = [\hat{y}_t - \hat{\varepsilon}_t, \hat{y}_t + \hat{\varepsilon}_t] \tag{3.18}$$

for all $1 \leq t \leq O$, we obtain our valid conformal predictor.

Theorem 2. *Let $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$ be the dataset of exchangeable time series windows, where $\mathbf{y}^{(i)} \in \mathbb{R}^O$ is the continuation of time series window $\mathbf{x}^{(i)} \in \mathbb{R}^L$, for every $1 \leq i \leq n$, generated from the same underlying distribution. Let $f : \mathbb{R}^L \to \mathbb{R}^O$ be a forecasting model which maps to all forecasted values simulateneously. Then for*

any $\alpha \in (0, 1)$*, using the method in Equation (3.18) we have that:*

$$\mathbb{P}[y_t \in \Gamma_t^\alpha(\hat{y}_t), \text{ for } 1 \leq t \leq O] \geq 1 - \alpha \tag{3.19}$$

Evaluation Metrics

For a set of predictions $\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_n$ with corresponding target sequences $\mathbf{y}_1, \ldots, \mathbf{y}_n$, the *independent coverage* (IC) and *joint coverage* (JC) are defined by:

$$\text{IC} = \frac{1}{nO} \sum_{i=1}^{n} \sum_{t=1}^{O} \mathbb{1}[y_t^{(i)} \in \Gamma_t^\alpha(\hat{y}_t^{(i)})] \tag{3.20}$$

$$\text{JC} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1} \left\{ \bigwedge_{t=1}^{O} y_t^{(i)} \in \Gamma_t^\alpha(\hat{y}_t^{(i)}) \right\} \tag{3.21}$$

where $\mathbb{1}$ is the indicator function and $\bigwedge$ denotes the logical `AND` operation. IC measures the proportion of examples where the true value falls within the prediction interval at each horizon time t separately, while JC measures the proportion of examples where the true values fall within their respective prediction intervals across the full horizon time points simultaneously. While the empirical IC and JC typically increase with lower significance levels α, the *mean interval width* (MIW) of the prediction given below:

$$\text{MIW} = \frac{1}{O} \sum_{t=1}^{O} 2\hat{\varepsilon}_t, \tag{3.22}$$

typically increases as well, which provides the average measure of uncertainty for model predictions. Therefore, a fundamental trade-off exists between coverage and interval width: models achieving higher JC often do so at the cost of wider prediction

intervals.

The core challenge is therefore to minimize the MIW while maximizing JC and IC across a chosen significance level α.

3.4.3 The Trade-off Score (TOS)

Given the fundamental trade-off between achieving optimal coverage while maintaining efficient prediction intervals, we propose a unified scoring metric that synthesizes JC, IC, and MIW into a single measure bounded in [0,1] referred to as weighted-average coverage (WAC) defined below:

$$\text{WAC} = \frac{\beta\text{JC} + (1-\beta)\text{IC}}{2} \tag{3.23}$$

where $\beta \in [0,1]$ is a hyperparameter controlling the relative importance of joint coverage versus independent coverage. While the choice of β should reflect the practitioner's objectives, we recommend $\beta > \frac{1}{2}$ as joint coverage typically presents a more important and stricter criterion in CP. To incorporate the width of prediction regions, consider a collection of k conformal predictors derived from k models with corresponding MIWs $m_1, \ldots, m_k \in \mathbb{R}^+$. We standardize these measures using z-score normalization $z_i = \frac{\mu - m_i}{\sigma}$, where μ and σ denote the sample mean and standard deviation of $\{m_1, \ldots, m_k\}$, respectively. Finally, we introduce our proposed trade-off score (TOS), a metric that combines both coverage validity and prediction width. For each conformal predictor i, we define:

$$\text{TOS}(i) = \lambda\text{WAC} + (1-\lambda)\frac{1}{1+e^{z_i}} \tag{3.24}$$

where $\lambda \in [0, 1]$ controls the balance between WAC and normalized MIW. The reflected sigmoid transformation of the z-scores ensures the second term remains bounded in [0,1], resulting in TOS(i) $\in [0, 1]$ for all conformal predictors. This unified metric acknowledges that an ideal conformal predictor should not only achieve the desired coverage levels, but do so with reasonably tight prediction intervals. Without such a combined metric, we might favor methods that achieve perfect coverage at the cost of excessively wide intervals, or methods that produce deceivingly narrow intervals but fail to maintain reliable coverage. The TOS enables practitioners to make these trade-offs explicit through the tuning parameters β and λ, while ensuring fair comparison across different methods through appropriate normalization. For meaningful comparisons, it is recommended to evaluate against a diverse set of methods that span different architectural families and methodological approaches. This helps ensure robust normalization and provides comprehensive performance context.

3.5 Experiments

In this section, we describe each of the selected datasets along with our experimental configuration for point forecasting and conformal forecasting. Our benchmarks include both *short-term* and *long-term* EMF exposure series. We define short-term to be any range within 24 hours, and long-term to exceed 24 hours. We analyze the characteristics of EMF exposure in several datasets through spectral decomposition, stationarity testing, and spatial correlation analysis.

3.5.1 EMF Datasets: Long-term EMF Series (Italy)

Adda *et al.* [3] presented long-term EMF measurements from four locations within Rome and Turin, Italy. Table 3.1 describes all measurement sites and their respective dataset specifications such as the year of recording, total length of each time series (i.e. duration of measurement), and reported sampling rate (Δt). Measurement sites include the Department of Electronic Engineering (DEE) at the University of Rome Tor Vergata, a University Hospital (UH) in Rome, Polytechnic University of Turin (PT), and the Porta Nuova Train Station (TS).

Table 3.1: Long-term EMF datasets from Italy with reported the duration of measurement, year of recording, and reported sampling rate (Δt).

City	**Location**	**Duration**	**Year**	$\boldsymbol{\Delta t}$
Rome	Dept. of Electronic Engineering (DEE 22')	190 days	2022	6 min
Rome	Dept. of Electronic Engineering (DEE 23')	491 days	2023	6 min
Rome	University Hospital (UH 22')	139 days	2022	6 min
Rome	University Hospital (UH 23')	592 days	2023	6 min
Turin	Polytechnic University of Turin (PT)	168 days	2020	6 min
Turin	Train Station (TS)	29 days	2022	6 min

Although each location exhibits different characteristics in their EMF distribution, each site displays similar daily exposure patterns with cyclical behaviour, as shown in Figure 3.2. Note that Δt represents the 6 min sampling interval (pre-processed from the data provider), where the true sampling rate of the device is 3 sec. For each location within Rome, narrow-band monitoring was performed using an Anritsu MS27102A spectrum monitor (9 kHz – 6 GHz) connected to a Keysight N6850A omnidirectional antenna (20 MHz – 6 GHz) and the required EMF aggregated over all frequencies is then obtained by computing the root sum squared values of the EMF

measured at narrow-band frequencies. In Turin, a Narda 8059 wideband monitor with an electric field sensor (100 kHz – 7 GHz) was used. For more information regarding each dataset, we refer the readers to [3].

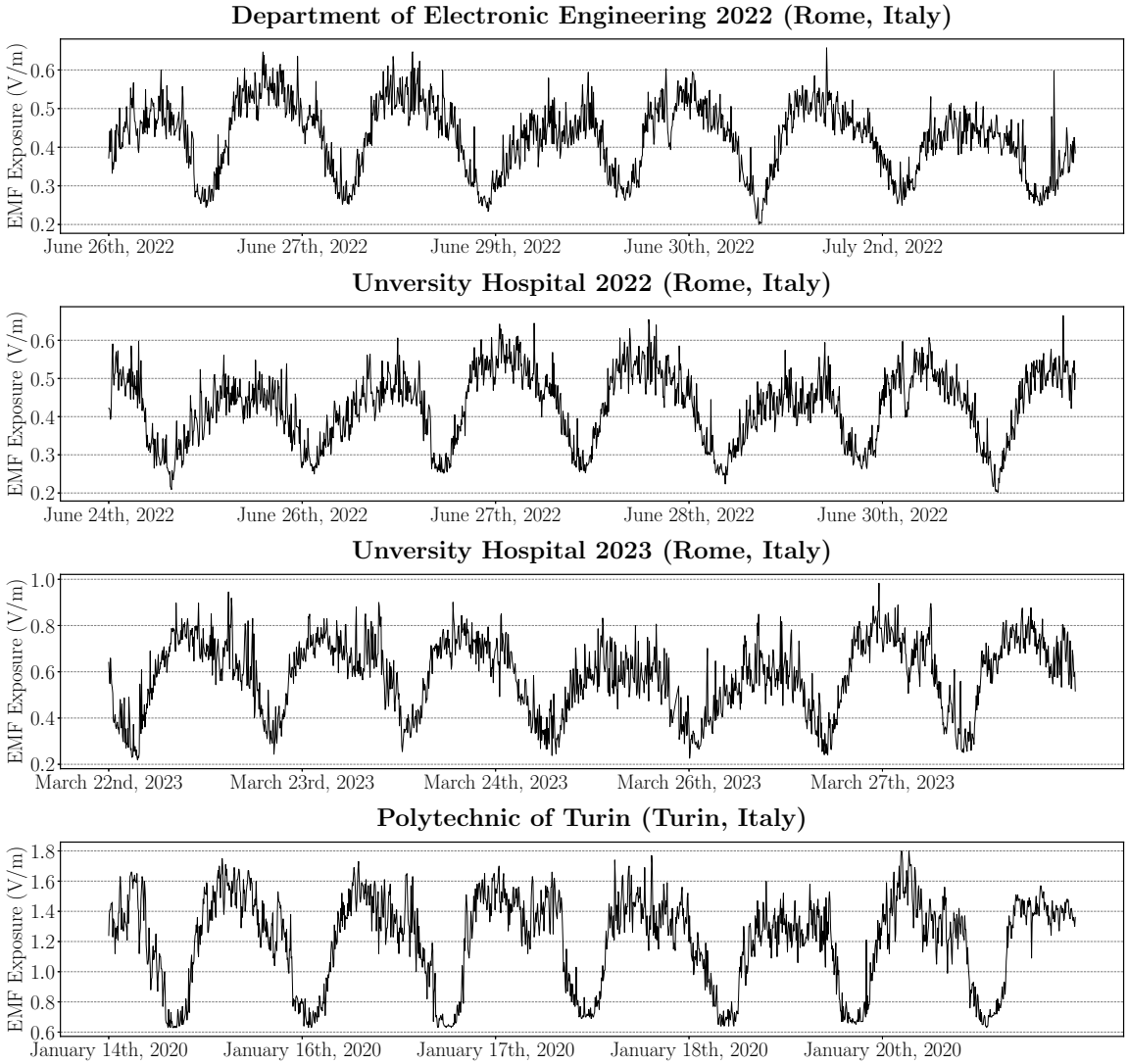


Figure 3.2: Visualization of the EMF exposure (V/m) over time from five locations in the Italy (DEE 22', UH 22', UH 23', PT, TS).

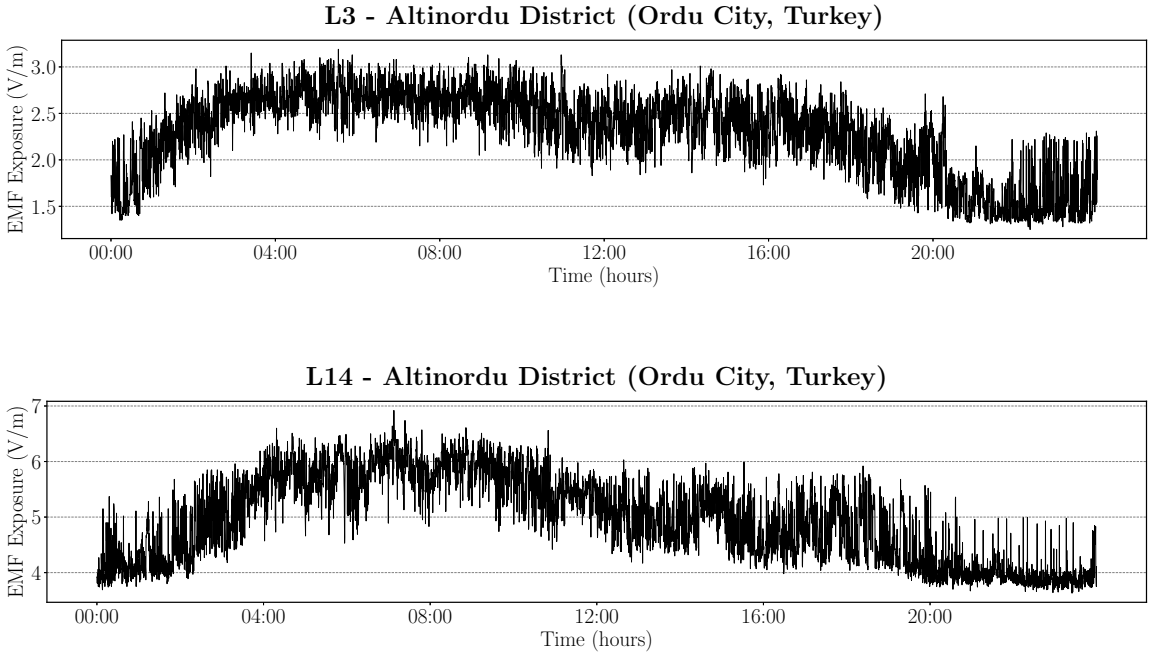


Figure 3.3: Visualization of the EMF exposure (V/m) over time from two locations in the Turkey dataset (L3, L14).

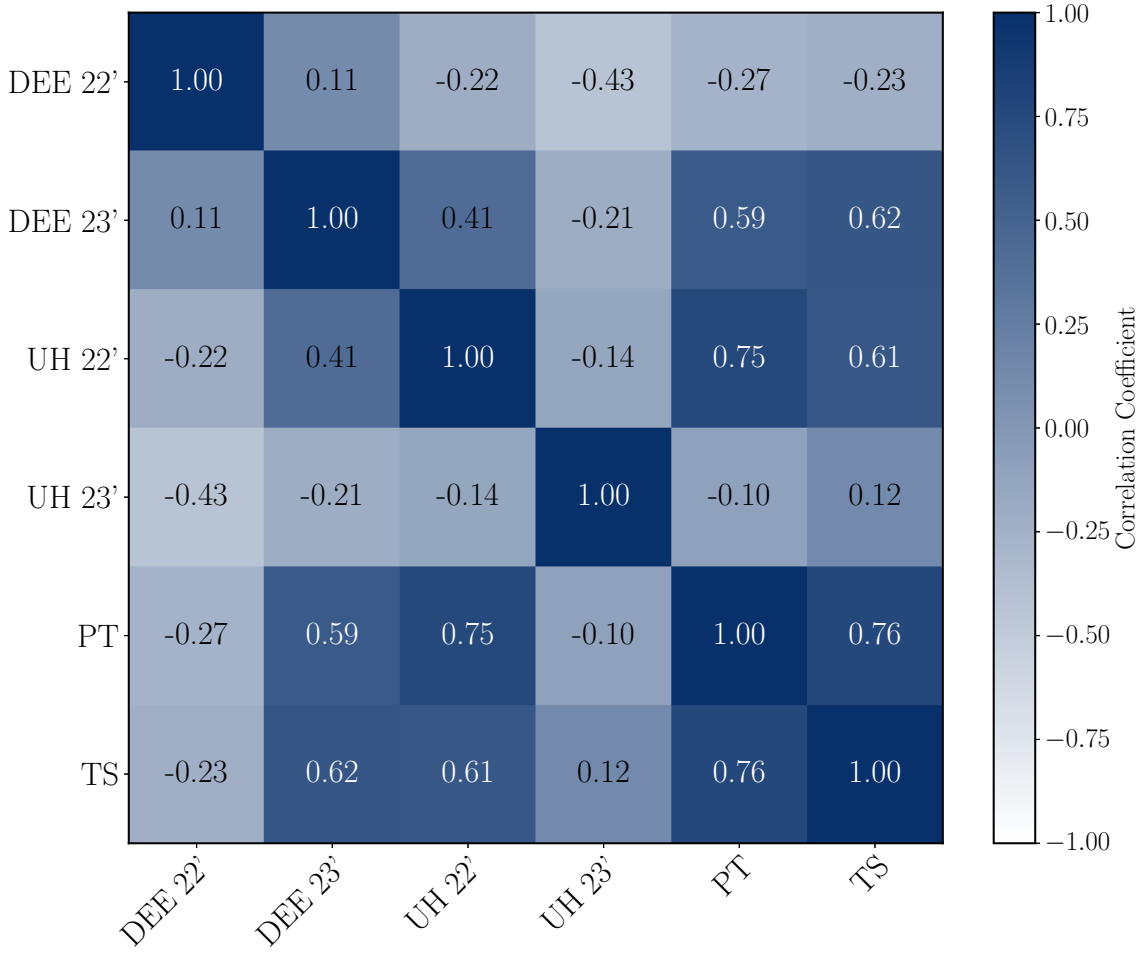


Figure 3.4: Correlation heatmap of EMF exposure over a one week period for all 4 locations in Italy.

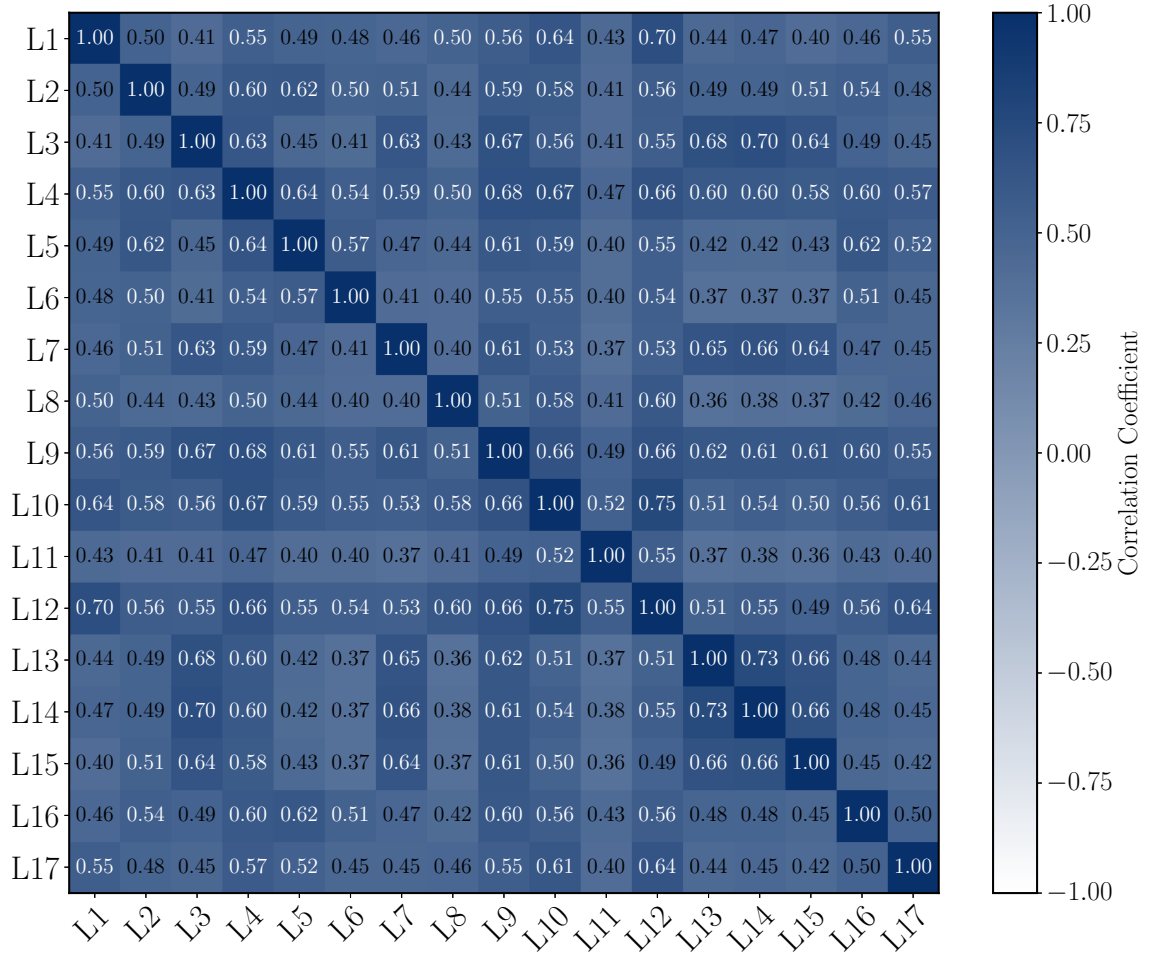
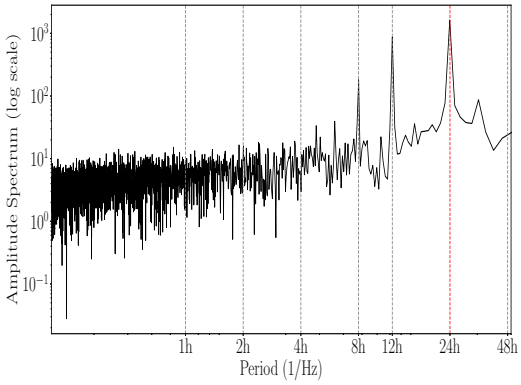
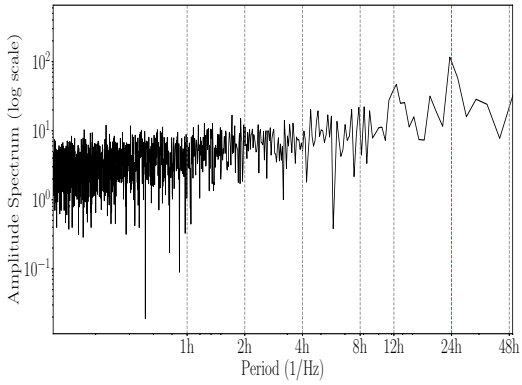


Figure 3.5: Correlation heatmap of EMF over a 24 hour (total) period for all 17 locations within the dataset from Turkey.



(a) FFT magnitude (DEE 22') with respect to the period (1/Hz).



(b) FFT magnitude (UH 22') with respect to the period (1/Hz).

Figure 3.6: FFT magnitude plots of DEE 22' and UH 22'.

3.5.2 EMF Datasets: Short-term EMF Series (Turkey)

In [66], Kurnaz *et al.* collected EMF exposure data in the Altınordu District of Ordu City, Turkey. Short-term ($\leq$ 24hr) broadband EMF measurements were conducted at 17 locations along the main streets of Altınordu, covering a frequency range of 100 kHz to 3 GHz, with a sampling rate of 15 seconds and duration of 24 hours. The authors used a PMM-8053 EMF meter equipped with an EP-330 electric field isotropic probe for the wideband measurements.

3.5.3 Long-Term vs. Short-Term Forecasting

The datasets from Italy and Turkey exhibit fundamental differences in their exposure measurements that significantly impact our analysis and potential model forecasting capabilities. The datasets from Italy contain extensive recordings spanning up to 592 days across six different locations, showing clear and consistent daily cyclical patterns throughout the extended monitoring period. On the other hand, the dataset from Turkey covers 24-hours periods for each of the 17 locations, but comes with more frequent measurements at 15-sec intervals compared to Italy's 6 min reported sampling rate. These contrasting characteristics indicate that our forecasting approach for the dataset from Italy can effectively model both the prominent daily cycles and their deviations, whereas the dataset from Turkey presents us with a more challenging prediction scenario due to its shorter observation window and less apparent periodicity, despite its higher temporal resolution. Our decision to maintain the 15-sec sampling rate for the dataset from Turkey rather than downsampling to match the 6 min intervals of datasets from Italy was driven by data quantity considerations. Averaging to 6 min intervals would have drastically reduced our already limited amount of

data available from the 24-hours recordings, which would be insufficient for a fair comparison between our proposed DL model and baselines.

3.5.4 Correlation of Measurement Sites

Figures 3.4 and 3.5 illustrate the correlation between measurement sites for Italy and Turkey through heatmap visualizations, respectively. While datasets from Italy exhibit selective high correlations between specific measurement pairs, such as PT and TS, the majority of sites demonstrate weak inter-site correlations. In contrast, Turkey reveals consistent positive correlations across all measurement site pairs. This uniform correlation pattern in the Turkish data may be attributed to the environmental conditions during data collection and geographical proximity, in addition to the the limited 24-hours sampling period, which could potentially inflate the observed correlation coefficients due to the shorter temporal window.

3.5.5 Cyclical Pattern Analysis

We observe that all time series within the dataset from Italy exhibit cyclical patterns, as shown in Figure 3.2. Using the FFT, we analyze these periodicities in the frequency domain, as illustrated in Figures 3.6a and 3.6b, which display the magnitude of Fourier coefficients as a function of period (1/Hz) rather than frequency (Hz). This representation, where a red vertical line marks the 24-hours period, clearly reveals dominant peaks at both 12-hours and 24-hours duration, demonstrating strong daily periodic components in the EMF exposure signals across both DEE '22 and UH '22 environments. Similar observations were noted for other datasets from Italy.

3.5.6 Stationarity of EMF Exposure

We apply the ADF test to assess stationarity, as shown in Figure 3.7, which displays the ADF test statistics and statistical significance (p) for the datasets considering two scenarios: the original raw data and first-order differenced data (indicated by dark borders). The results reveal that all time series from Italy are stationary ($p < 0.05$), while many measurement sites in Turkey exhibit non-stationary behavior. This distinction can be attributed to the limited 24-hours recording period in Turkey, which captures only a single daily cycle and thus appears as a non-stationary time series. In contrast, the Italy measurements, as demonstrated in the previous section, display cyclical patterns centered around a mean value without persistent trends, confirming their stationary nature. Upon applying first-order differencing, we observe improved (more negative) ADF statistics across both datasets, with all time series from Italy becoming increasingly stationary with significantly reduced ADF statistics.[1]

3.5.7 Training Configuration for DL Models

We partition $(x_t)_{t=1}^{T}$ into training, validation, and test datasets, with proportions of 70%, 10%, and 20%, respectively. For all considered DL models, including EMForecaster and baselines, we conduct training using the Adam optimizer across 100 epochs with a batch size of 2048, employing early stopping with a patience of 20 epochs on the validation Mean Square Error (MSE) [60]. We use the MSE as our objective function

[1]The stationarity of a time series, i.e., maintaining consistent statistical properties over time, suggests that windows drawn from it are more likely to be i.i.d., and thus exchangeable. This connection to exchangeability is particularly relevant for CP-based forecasting. This observation aligns with our empirical results, where we achieved state-of-the-art performance in both point forecasting and CP tasks on the stationary Italy datasets, while experiencing modest results on the non-stationary Turkey dataset.

and main evaluation metric on the test set, given by:

$$\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{O}\sum_{i=1}^{O}(y_i - \hat{y}_i)^2, \tag{3.25}$$

for a prediction $\hat{\mathbf{y}} \in \mathbb{R}^O$ and label $\mathbf{y} \in \mathbb{R}^O$. We performed comprehensive hyperparameter tuning through grid search, optimizing separately for model, dataset, and prediction length O. Model training was executed on a single NVIDIA RTX 6000 Ada Generation GPU with 48GB of memory. The reported performance metrics on the test set represent averages across five random seeds, using the model configuration that achieved the lowest validation MSE (for the specific dataset and prediction length).

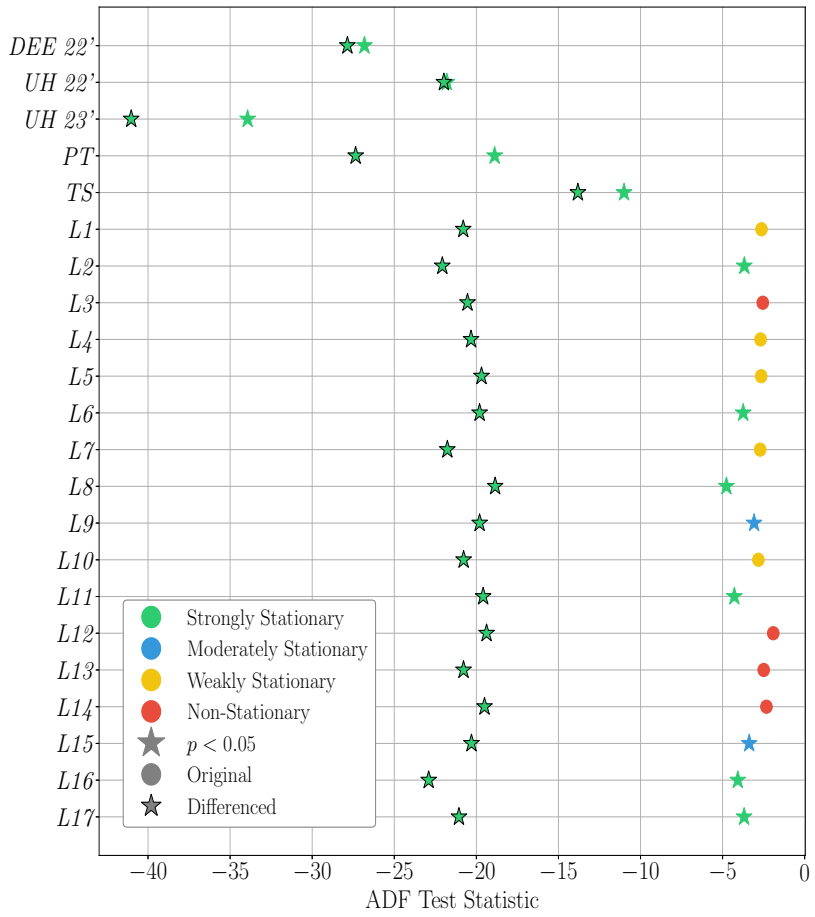


Figure 3.7: ADF Test to characterize stationarity of the dataset from Turkey (17 locations) and Italy (4 locations). ADF test statistics and their statistical significance (p) are shown.

Table 3.2: EMF point forecasting in terms of MSE with lookback window $L = 336$ and prediction lengths $O \in \{96, 192, 336, 512\}$.

		EMForecaster	DLinear	TSMixer	PatchTST	Transformer	LSTM	CNN	MLP	ARIMA
	O	(2025)	(2023)	(2023)	(2023)	(2017)	(1997)	(1988)	(1986)	(1970)
DEE 22'	96	**0.3019**	0.3488	0.3119	0.3710	0.7744	0.3409	0.3217	0.3217	1.2371
	192	**0.3461**	0.4055	0.3561	0.4151	1.1312	0.4033	0.3794	0.3794	1.4484
	336	**0.3811**	0.4791	0.3911	0.4664	1.1417	0.4426	0.4314	0.4314	1.4193
	512	**0.4436**	0.5420	0.4536	0.5336	1.3673	0.4854	0.4806	0.4806	1.4333
UH 22'	96	**0.2404**	0.2841	0.2685	0.2992	1.2874	0.3540	0.3599	0.4059	1.6604
	192	**0.2726**	0.3159	0.3026	0.3345	1.2291	0.4863	0.4185	0.4175	1.9352
	336	**0.3148**	0.3735	0.3580	0.3801	0.8394	0.5407	0.4733	0.4733	1.9328
	512	**0.3535**	0.4165	0.4045	0.3973	0.6675	0.6049	0.5463	0.5463	1.9635
UH 23'	96	**0.2999**	0.3251	0.3003	0.3490	1.0483	0.3180	0.3133	0.3133	1.0896
	192	**0.3158**	0.3417	**0.3158**	0.4017	0.7665	0.3453	0.3391	0.3393	1.1964
	336	0.3346	0.3716	**0.3326**	0.4083	0.6530	0.3517	0.3600	0.3600	1.2197
	512	0.3434	0.3945	**0.3429**	0.4422	0.7776	0.3670	0.3787	0.3787	1.2328
PT	96	**0.1383**	0.1423	0.1481	0.1460	0.3441	0.2027	0.1554	0.1554	0.2151
	192	**0.1476**	0.1483	0.1564	0.1509	0.3970	0.2536	0.1616	0.1616	0.2931
	336	**0.1571**	0.1592	0.1697	0.1632	0.3977	0.2546	0.1723	0.1723	0.3348
	512	**0.1656**	0.1670	0.1756	0.1675	0.3314	0.2571	0.1787	0.1787	0.3574
Turkey	96	0.4174	**0.4054**	0.4396	0.4822	0.5349	0.7714	0.7660	0.8014	0.8014
	192	0.4564	**0.4422**	0.4793	0.5348	0.5804	0.8221	0.7758	0.8143	0.8143
	336	0.4885	**0.4818**	0.5106	0.6050	0.6007	0.7917	0.7827	0.8296	0.8296
	512	0.5230	**0.5101**	0.5463	0.6612	0.6241	0.8018	0.7848	0.8356	0.8356

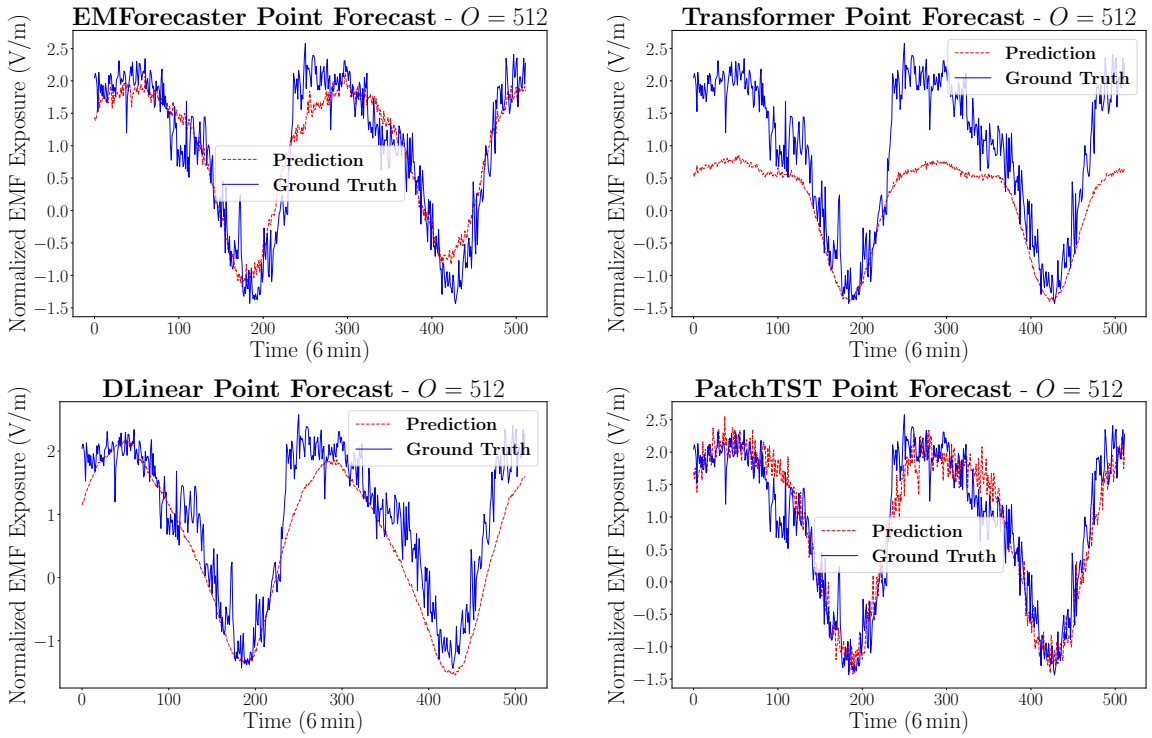


Figure 3.8: Point forecasting plot of model predictions (red) versus ground truth labels (blue) with lookback $L = 336$ and prediction length $O = 512$ on the UH 22' dataset.

3.6 Results

In this section, we describe the considered baselines and analyze the performance of the proposed EMForecaster for both *point forecasting*, which provides deterministic predictions, and *conformal forecasting*, which generates prediction intervals with statistical guarantees.

3.6.1 Baselines

The considered baselines are listed as follows:

• **MLP** processes the input time series through multiple fully-connected layers, capturing non-linear relationships between historical and future values through a direct

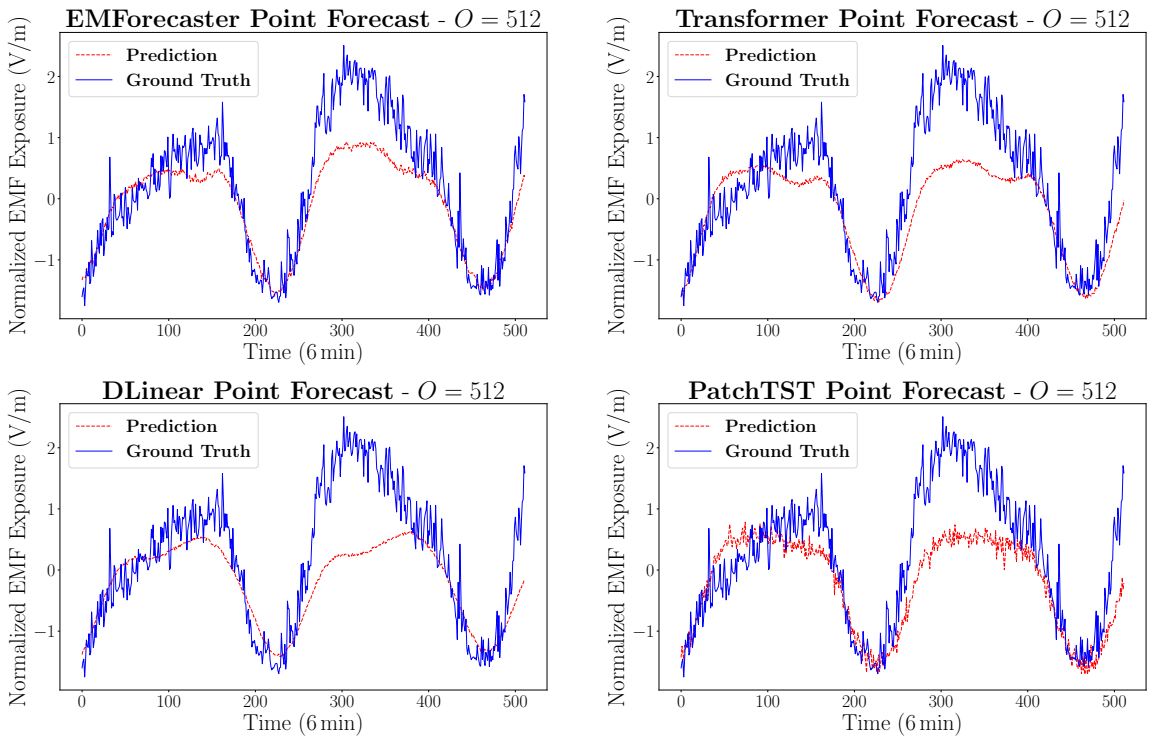


Figure 3.9: Point forecasting plot of model predictions (red) versus ground truth labels (blue) with lookback $L = 336$ and prediction length $O = 512$ on the DET 22' dataset.

mapping approach.

• **LSTM** uses gating mechanisms and recurrent connections to model temporal dependencies, enabling selective retention of relevant historical information across the series [50].

• **CNNs** utilize 1D convolutions sliding over the input sequence to extract local temporal patterns, effectively capturing hierarchical features through shared parameters.

• **Transformer** employs self-attention mechanisms to model relationships among all time steps simultaneously, allowing it to capture both long-range dependencies and local patterns. The parallel processing nature of attention makes it particularly effective at modeling complex temporal interactions [118]. In this work, we use the

standard encoder-only architecture.

• **DLinear** is a modern neural network that has consistently outperformed many contemporary Transformer-based forecasting models. This is done by explicitly decomposing each input time series into trend and seasonal components, processing them through separate linear projection layers. The trend extractor uses a moving-average filter and linear layer to capture long-term dependencies and the seasonal extractor passes the high-frequency residual through another linear layer to model periodic patterns—and then recombining these outputs to form an interpretable final forecast [134].

• **TSMixer** is an MLP-based time series model, inspired by MLP-mixer [24, 117]. TSMixer processes time series through parallel MLPs that separately handle temporal patterns (across time steps) and feature interactions (across measurements).

• **PatchTST** is a patched time series Transformer that segments each series into fixed-length patches, feeds them into a standard Transformer encoder to capture local dynamics via patch embeddings and global dependencies via self-attention. PatchTST has empirically outperformed leading variants across multiple benchmarks while gaining broad adoption in diverse domains [88].

• **ARIMA** is a traditional statistical learning model, which combines differencing, autoregression, and moving average components to model linear relationships in stationary time series data [22].

3.6.2 Results and Discussions: Point Forecasting

Comparative Analysis of EMForecaster with Baselines

We evaluate the performance of point forecasting, i.e., forecasting individual future time points in the traditional sense on all datasets. Window sampling is performed on all 17 time series from Turkey, before taking the union of all windows to produce a full, multi-measurement site dataset. In contrast, we considered each dataset from Italy separately (DEE 22', UH 22', UH 23', PT) due to their distributional differences and weaker correlation compared to Turkey, as shown in Fig. 3.5.

For each dataset, we evaluate four prediction lengths $O \in \{96, 192, 336, 512\}$ independently, with a fixed lookback window of $L = 336$. The reported sampling rate is 6 min for Italy, thus the forecast ranges from 9.6 hrs to 51.2 hrs with a lookback window of 33.6 hrs. For the dataset from Turkey, the lookback window is fixed at 84 min, with prediction lengths ranging from 24 min to 128 min. Therefore, although the units for L and O are identical, the forecasting lengths differ as the reported sampling rate is different in the two datsets.

Point Forecasting Plots

Figure 3.8 compares point forecasting results within the UH 22' dataset for EMForecaster alongside the Transformer, DLinear, and PatchTST with a lookback window $L = 336$ and prediction length $O = 512$. EMForecaster strikes a desirable middle ground: it captures the key daily EMF exposure cycle and its finer microstructure without capturing spurious noise. In contrast, DLinear produces an overly smooth trajectory that closely follows the dominant cycle, but systematically lags and ignores short-term fluctuations, leading to substantial phase errors at the beginning and end

of the forecast window. PatchTST, while highly responsive to transient perturbations, overfits fine-scale noise and yields excessive variance around the true signal. The standard Transformer underfits the cycle amplitude, consistently underestimating trough depths and failing to align with the highest peaks. Overall, EMForecaster maintains stable adherence to the underlying exposure pattern, while reproducing its microstructure—avoiding both the under- and overfitting exhibited by the other models.

Table 3.2 displays point forecasting performance across all baselines. EMForecaster achieves consistently superior MSE performance compared to modern DL baselines (PatchTST, TSMixer, DLinear), while traditional approaches (Transformer, LSTM, CNN, MLP, and ARIMA) show higher MSE on average. While DLinear and TSMixer achieve comparable performance on the PT and Turkish datasets, their performance degrades significantly on other datasets, particularly with increasing forecast horizons O. EMForecaster demonstrates robust performance across the Italian datasets, though it shows slightly lower performance on the Turkish dataset due to inherent non-stationarity and limited data availability. These results indicate that in stationary environments, EMForecaster exhibits strong generalization capabilities across diverse settings and forecast horizons.

Analysis of Patch Dimension and Patch Embedding Dimension

Figures 3.10 (MSE) and 3.11 (Runtime) together illustrate three key trade-offs in EMForecaster:

- **Patch size** (P): As P grows from $8 \rightarrow 36$, MSE increases from $0.32 \rightarrow 0.39$, while runtime drops from $104\text{s} \rightarrow 62\text{s}$ indicating that smaller patches capture

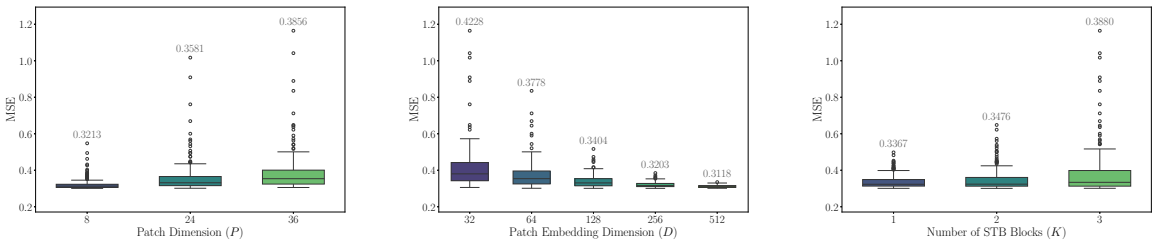


Figure 3.10: MSE boxplots for the (a) patch dimension P, (b) patch embedding dimension D, and (c) the number of STB blocks K, over multiple hyperparameter configurations on the DEE 22' dataset with lookback window $L = 336$ and prediction length $O = 96$. Average MSE is reported at the top of each boxplot.

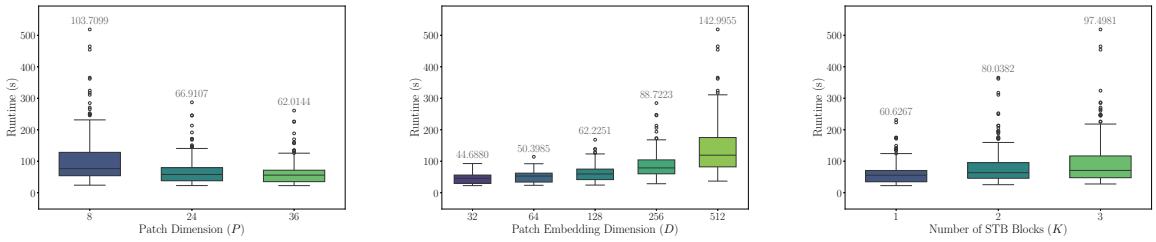


Figure 3.11: Runtime boxplots for the patch dimension P, patch embedding dimension D, and the number of STB blocks K, over multiple hyperparameter configurations on the DEE 22' dataset with lookback window $L = 336$ and prediction length $O = 96$. Average Runtime is reported at the top of each boxplot.

fine-grained dynamics more accurately at the expense of longer computation.

- **Embedding dimension** (D): Increasing D from $32 \rightarrow 512$ reduces MSE from $0.42 \rightarrow 0.31$, but raises runtime from $45\text{s} \rightarrow 143\text{s}$ showing that richer embeddings enhance representational power (lower error) but incur higher computing time.

- **Number of STB blocks** (K): Moving from $1 \rightarrow 3$ blocks drives MSE up from $0.34 \rightarrow 0.38$ and runtime from $61\text{s} \rightarrow 97\text{s}$ revealing that deeper stacks begin to overfit and slow down inference rather than improve generalization.

The inverse relationship between patch size (P) and performance, together with the

strong gains from larger embedding dimensions (D), highlights a synergistic interaction. That is, using smaller patches concentrates the model's attention on fine-grained, localized signal fluctuations, and then projecting each patch into a high-dimensional embedding space amplifies its representational richness. In practice, a small P ensures that each embedding encodes tightly scoped temporal features, while a large D provides sufficient capacity for the patch MLP and time MLP to disentangle and recombine these features into expressive, global patterns. Consequently, configurations with $D \gg P$ maximize the information extracted per patch—validating our design choice to pair fine-resolution inputs with high-capacity embeddings.

3.6.3 Results and Discussions: Conformal Forecasting

We conduct our conformal forecasting experiments on the same datasets used for point forecasting for the prediction length $O = 96$ with lookback window $L = 336$ using the framework in section 3.4.2 on EMForecaster and the baselines.

Table 3.3: Performance evaluation of EMForecaster, examining the effect of sampling rates Δt across several datasets, with lookback window $L = 336$ and prediction length $O = 96$.

	Δt	MSE	JC	IC	MIW
DEE 22'	6min	0.3008	82.89	98.98	4.99
	30min	0.2104	84.96	99.49	6.93
UH 22'	6min	0.2467	73.70	97.95	4.11
	30min	0.2133	51.19	96.07	5.94
PT	6min	0.1372	62.41	98.58	2.73
	30min	0.1181	65.04	98.81	3.09

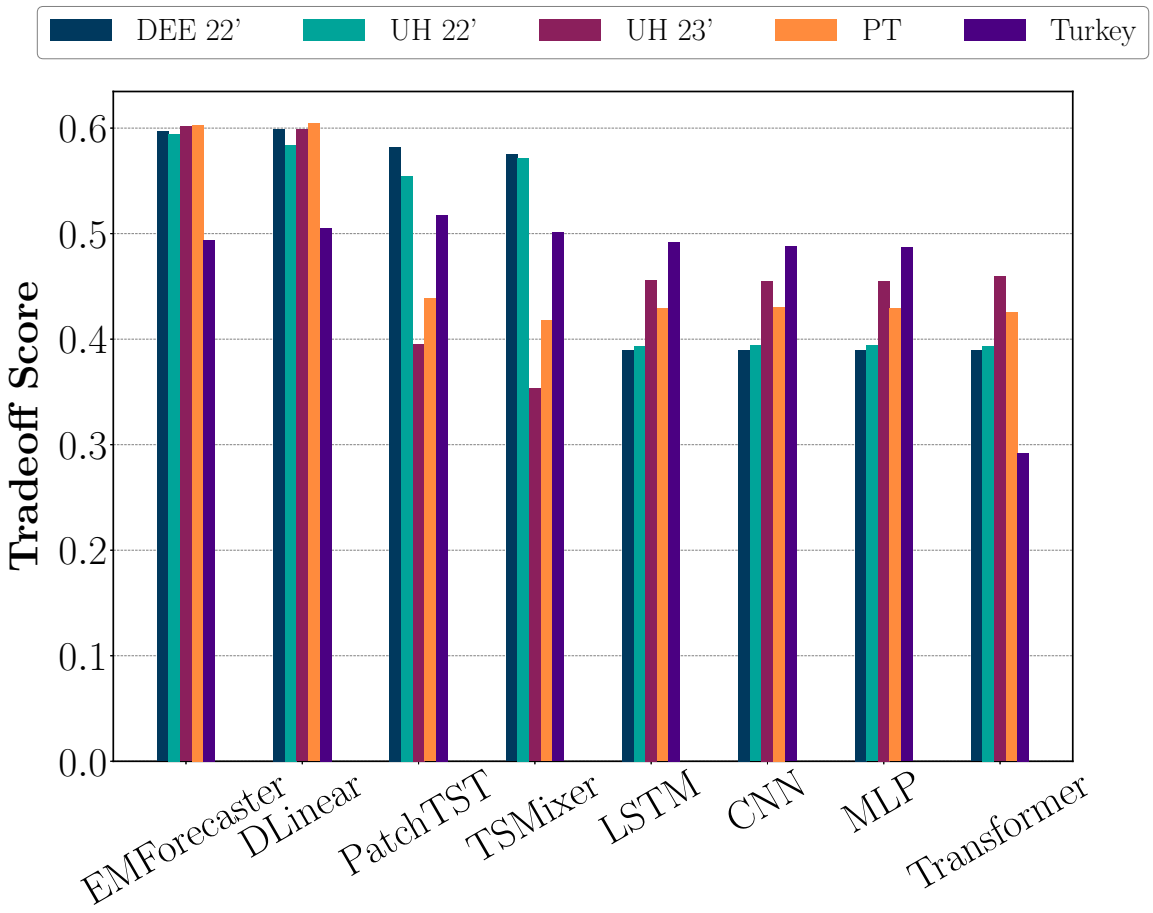


Figure 3.12: Performance comparison of different forecasting models across datasets using the Tradeoff Score (TOS) metric using $\alpha = 0.01$, $\beta = \frac{2}{3}$, and $\lambda = \frac{1}{2}$.

Comparative Analysis of EMForecaster with CP

Fig. 3.12 displays the results across all datasets for each model with respect to the TOS. We see that EMForecaster achieves comparable performance to DLinear while achieving superior performance compared to all other baselines. For this experiment, we used $\beta = \frac{2}{3}$ and $\lambda = \frac{1}{2}$, as suggested in the discussion from section 3.4.3. $\beta = \frac{2}{3}$ skews the WAC to the joint coverage as it is a harder metric to optimize, while still optimizing partly for the IC, while $\lambda = \frac{1}{2}$ provides an equal importance between

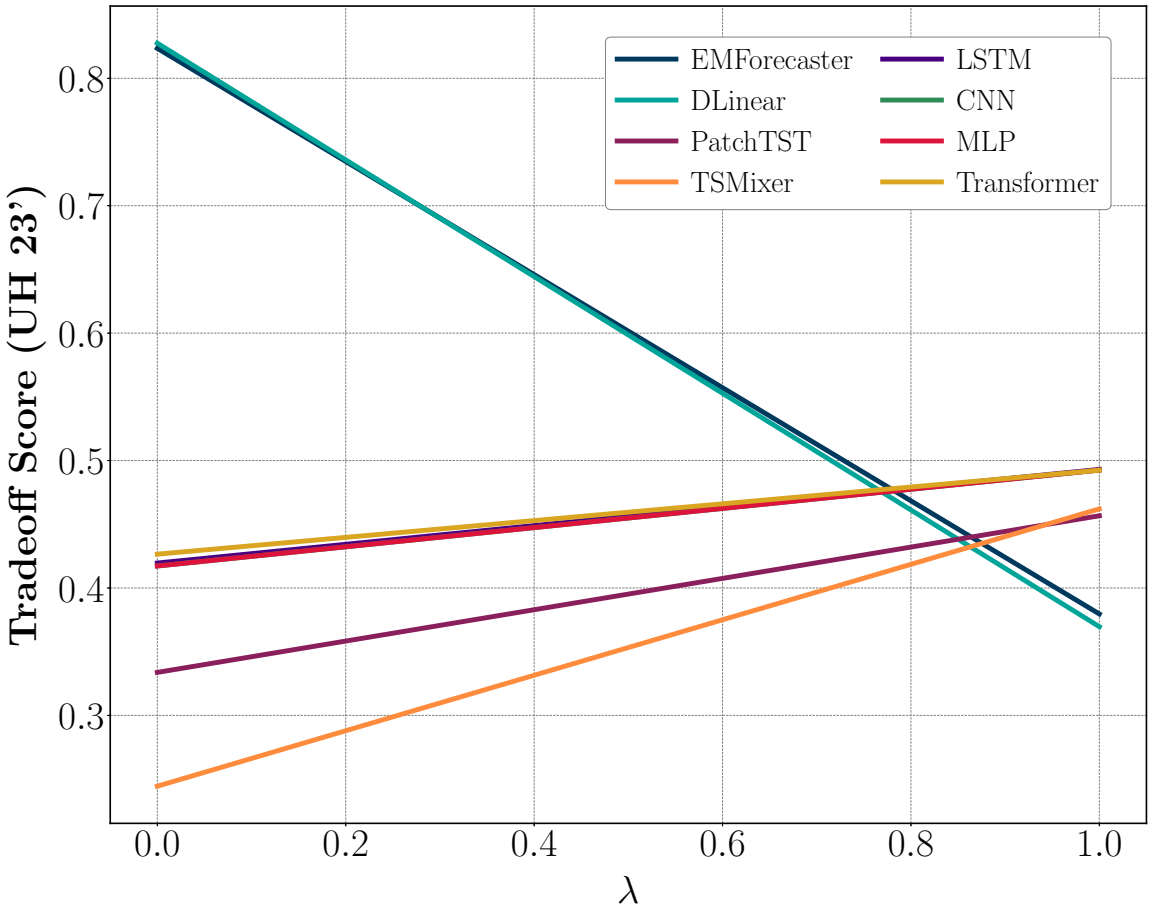


Figure 3.13: Tradeoff Score (TOS) as a function of λ with $\alpha = 0.01$ and $\beta = \frac{2}{3}$ for UH 23' dataset.

coverage and prediction interval width when computing the TOS.

Table 3.4 results reveal EMForecaster achieving strong JC while maintaining narrowest MIW, striking the best balance between JC and MIW among the three methods as seen in the results of the Tradeoff Score. Although DLinear runs a very close tradeoff between MIW and JC, EMForecaster slightly outperforms it on both the JC/MIW front and in point-forecast accuracy (lower MSE). The Transformer attains the highest JC, but does so with excessively wide MIW, rendering its output to be far less practical. This delicate balance highlights EMForecaster's ability to deliver

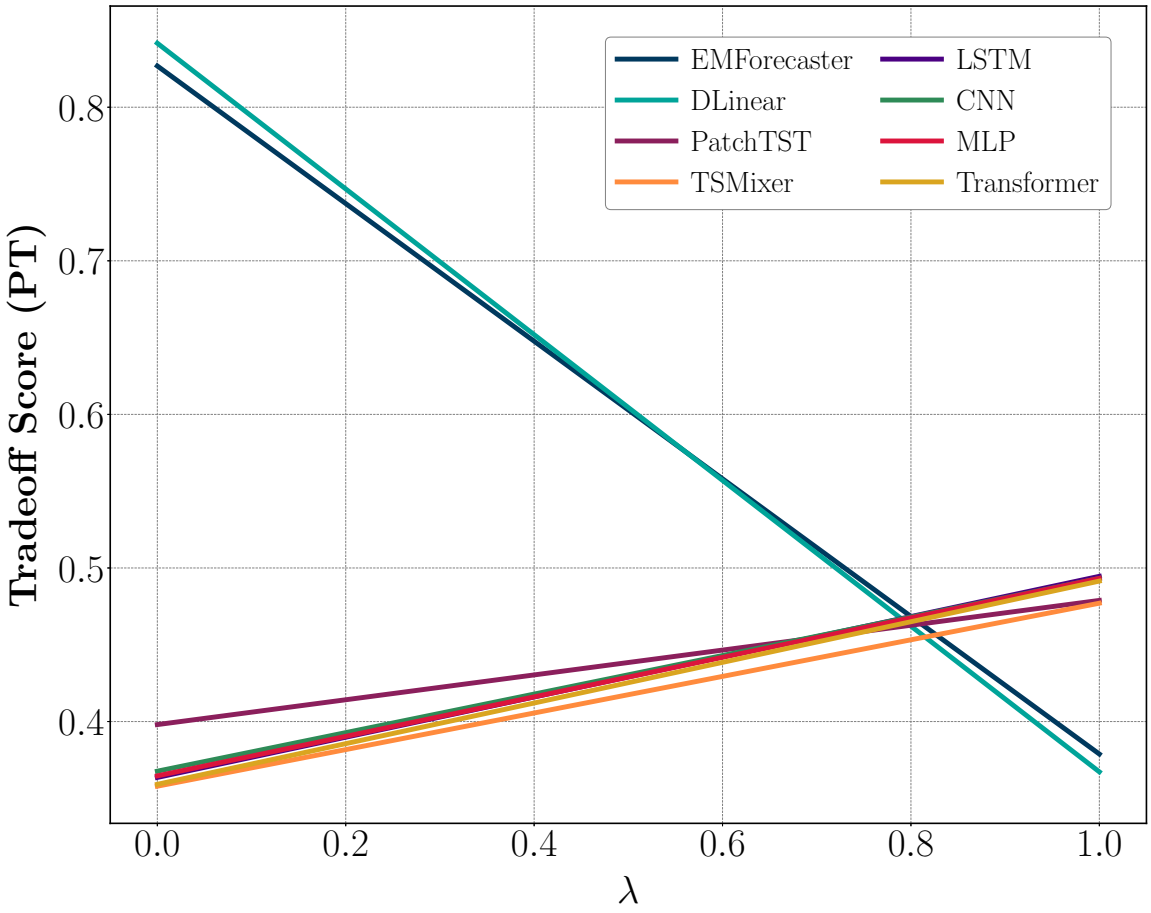


Figure 3.14: Tradeoff Score (TOS) as a function of λ with $\alpha = 0.01$ and $\beta = \frac{2}{3}$ for PT dataset.

reliable conformal intervals without sacrificing predictive precision.

Choice of λ in Conformal Forecasting

Fig. 3.13 and Fig. 3.14 examine the impact of λ on the TOS for the UH 23' and PT datasets respectively. Both EMForecaster and DLinear outperform all other baselines up to $\lambda \approx 0.8$. Beyond this point, the TOS becomes heavily influenced by coverage. Thus, while many baselines can achieve good coverage, it requires a very large λ to achieve an optimal TOS, indicating a poor balance between obtaining coverage and

interval widths for most baselines. In contrast, EMForecaster and DLinear strike an optimal balance, avoiding excessively wide intervals while maintaining strong coverage.

3.6.4 Analysis of Differencing and Temporal Resolution

Table 3.3 presents EMForecaster's performance across different temporal resolutions. We compare results at the original sampling rate of $\Delta t = 6$ min and a down-sampled rate of $\Delta t = 30$ min, where the latter is obtained by averaging five consecutive 6 min measurements. While independent coverage remains consistent across both sampling rates, we observe a notable decrease in joint coverage for the UH '22 dataset at $\Delta t = 30$ min. The coarser temporal resolution ($\Delta t = 30$ min) generally yields lower MSE due to the smoothing effect that reduces high-frequency noise. Interestingly, despite higher MSE values, the $\Delta t = 6$ min resolution achieves narrower mean prediction intervals, suggesting superior conformal forecasting performance. This demonstrates a trade-off between point forecast accuracy and the precision of uncertainty quantification across different temporal resolutions.

3.6.5 Computational Complexity

For DET 22' with lookback window $L = 336$, forecast horizon $O = 96$, and $\alpha = 0.01$, the combined point forecasting and conformal forecasting training and evaluation durations were 38.51 sec for EMForecaster, 27.19 sec for DLinear, 10.98 min for TSMixer, 10.87 min for PatchTST, 1.25 min for the Transformer, 1.04 min for the LSTM, 52.68 sec for the MLP, and 54.99 sec for the CNN. Our computational results demonstrate that EMForecaster is the second fastest model, with only DLinear achieving marginally faster execution times. Modern architectures such as TSMixer and PatchTST exhibit

significantly longer training times, approximately 17 times that of EMForecaster, indicating less efficient parameter utilization. In addition, EMForecaster outperforms traditional DL architectures including LSTM and Transformer in terms of computational efficiency, executing 1.6-2x faster, which makes it particularly suitable for applications requiring rapid training or deployment under computational constraints.

Table 3.4: Conformal time series forecasting performance for EMForecaster and select baselines on the UH 23', PT and Turkey datasets (lookback $L = 336$, horizon $O = 96$).

	UH 23'				PT				Turkey			
	IC	JC	MIW	MSE	IC	JC	MIW	MSE	IC	JC	MIW	MSE
EMForecaster	98.11	64.83	4.645	0.3005	98.98	67.23	3.123	0.1202	99.20	61.92	4.978	0.4174
DLinear	98.22	61.77	4.620	0.3689	98.50	60.93	2.680	0.1420	99.31	63.47	5.026	0.4080
Transformer	99.97	97.74	6.343	1.0020	99.97	97.46	4.416	0.3539	100.0	100.0	8.477	0.7714

3.7 Conclusion

In this work, we have introduced EMForecaster, a DL-empowered time series forecasting framework designed to predict EMF trends across diverse locations and varying forecast horizons beyond 50 hours. To improve the reliability of EMF predictions, we integrated a distribution-free uncertainty quantification framework, using CP, ensuring the ground truth falls within a specified prediction interval with a user-specified error rate. To enhance the evaluation of conformal forecasting, we introduce a novel TOS evaluation metric which balances prediction interval width and empirical coverage, allowing for objective comparisons across different conformal predictors. Our extensive experiments demonstrate that EMForecaster significantly outperforms existing DL models in both point forecasting and conformal forecasting. Future research directions include extending EMForecaster to Beyond 5G (B5G) data and expanding the forecast horizon to multiple years.

Chapter 4

Stochastic Sparse Sampling

4.1 Chapter Contributions

In this chapter we develop a variable-length time series classification (VTSC) framework, overcoming the limitations of both finite-context and infinite-context methods, with applications to the highly critical task of *seizure onset zone (SOZ) localization*—the task of identifying brain regions from which seizures emanate [17].

Although the channel dependent approaches enable explicit modeling of the brain's functional connectivity, they encounter several challenges. Mapping the resulting representation $\mathbf{z}$ to all M probabilities (where M is the number of iEEG channels) is a complex task with significantly higher sample complexity, while simultaneously offering fewer training examples (M fewer compared to channel independent approaches). This challenge is highlighted in Table 4.1, where a channel dependent method would only have $n = 35$ examples (needing to predict $M \geq 80$ channels per example), a channel independent would have $n = 7611$ examples (needing to only predict 1 channel per example), when comparing the number of patients to the number of total time series. Data scarcity is commonplace in medical contexts, particularly due to the

prohibitive costs of data acquisition and the specialized expertise required (in this case, neurosurgical specialists and epileptologists). Additionally, channel dependent approaches typically sacrifice detailed local signal analysis and interpretability in favor of relational learning, limiting their ability to extract meaningful information from individual electrode signals, which is crucial in a clinical setting where interpretability is required. Lastly, due high variance of epileptic dynamics, generalization beyond a single patient is extremely difficult for such methods [29, 44].

Given the aforementioned limitations of channel dependent methods, in this chapter, we investigate the feasibility of channel independent methods for two key reasons, **(1)** they are resilient to inter-patient variability and are unaffected by factors like channel count, thus can be applied in multiple hospital settings with different hardware, and **(2)** they generalize better to domains beyond electrophysiological data, as they learn local signal characteristics rather than explicitly modeling functional connectivity between electrode sites.

To this end, we propose **S**tochastic **S**parse **S**ampling (SSS), a training and inference framework for VTSC evaluated on the task of SOZ localization. For each patient iEEG recording, the objective of SOZ localization is to precisely identify the channels (electrodes) belonging to the SOZ. This effectively reduces the task to univariate TSC. While SSS focuses on SOZ localization, this framework can be applied for a variety of medical time series, where variation among sequence length is high and access to clinical data is low. Our main contributions can be summarized as follows:

- **Robustness to long and variable-length sequences**. Rather than processing entire sequences, SSS samples fixed-length windows, feeds each independently through a single model, and then aggregates their predictions, thereby eliminating

the need for padding, truncation, or interpolation used by finite-context methods and preventing context overload seen in infinite-context methods over long sequences. SSS delivers +**11.4**% F1-score, +**9.4**% Area Under the Curve (AUC), and +**8.3**% accuracy gains over the average baseline while avoiding context overload, outperforming every baseline on the 'All' medical center evaluation.

- **Generalization to unseen patient populations.** SSS demonstrates strong out-of-distribution performance on data from unseen medical centers, outperforming modern baselines by up to +**26.8**% accuracy on specific centers and achieving average gains of +**15.37**% F1-score, +**16.27**% AUC, and +**8.66**% accuracy on the average center. This generalization is not only crucial for clinical deployment across diverse healthcare settings with varying patient demographics and recording equipment, but also provides compelling evidence that local signal features implicitly contain information about the SOZ.

- **Computationally efficient training**. By relying on a single local model, the SSS outperforms finite-context methods with fewer parameters, thus reducing computational demands and risk of overfitting. By partitioning time series into windows, memory complexity is only a function of window and batch size, instead of total sequence length. This enables efficient processing of even extremely long sequences on standard hardware, with our ablation studies confirming that performance remains consistent regardless of batch size.

- **Explainability through local predictions.** SSS provides interpretability by assigning probability scores to individual windows that contribute to the final prediction—a crucial feature for SOZ localization where clinical decisions

typically rely on visual analysis. Given the significant risks of surgical brain tissue removal and the absence of universal epilepsy biomarkers, this window-level explainability facilitates integration into existing clinical workflows while advancing the understanding of epilepsy.

- **Compatibility with modern and classical backbones.** SSS can integrate with any time series backbone which ensures that our approach leverages well-established frameworks, allowing for adaptability across a diverse array of contexts.

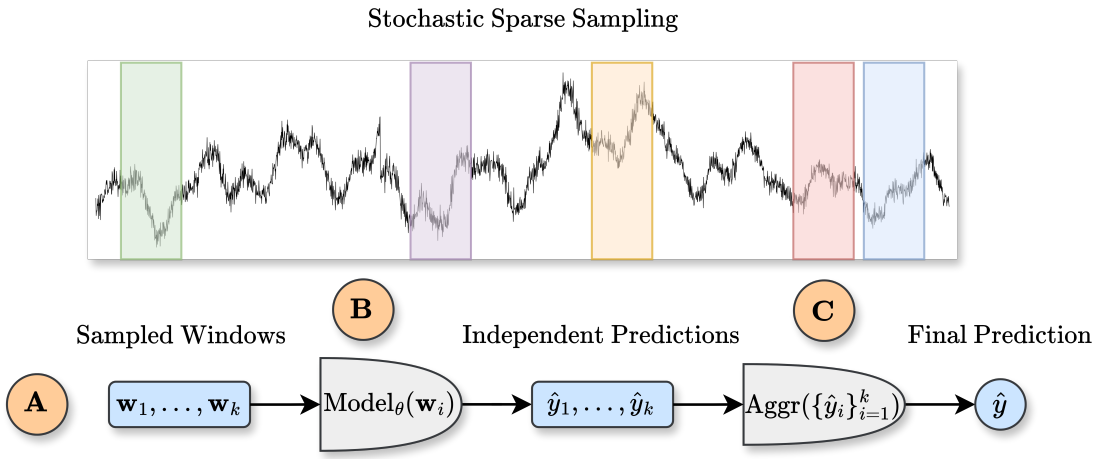


Figure 4.1: An overview of Stochastic Sparse Sampling (SSS) training procedure. (**A**) For a given time series, we sample windows of fixed-length at random throughout the signal. (**B**) Each window is processed independently by a local model with parameters θ, outputting the local predictions $\hat{y}_1, \ldots, \hat{y}_k$. (**C**) Local predictions are then fed through an aggregation function to form the final prediction $\hat{y}$.

4.2 Chapter Organization

Section 4.3 introduces our SSS framework, detailing both the training algorithm and the calibrated inference procedure. Section 4.4 then outlines the Epilepsy iEEG

Multicenter Dataset, highlights its key characteristics, and explains the experimental design used to compare finite- and infinite-context VTSC methods. Finally, Section 4.5 presents the findings: in-distribution performance (Table 4.3), out-of-distribution robustness across multiple medical centers (Table 4.5), and ablation studies that examine the effects of batch size, window length, and calibration model choice. Section 4.6 concludes with the main findings of this chapter.

4.3 Proposed SSS Framework for SOZ Localization

In this section, we describe the proposed SSS framework which includes two key steps (1) model training with sparsely sampled time series windows, and (2) calibrated inference. Figure 4.1 provides a graphical illustration of the framework.

4.3.1 Stochastic Sparse Sampling

Consider a collection of time series $X = \left\{(\mathbf{x}_t^{(1)})_{t=1}^{T_1}, \cdots, (\mathbf{x}_t^{(n)})_{t=1}^{T_n}\right\}$ with labels $Y = \{y^{(1)}, \cdots, y^{(n)}\}$, where each series i has a sequence length of $T_i \in \mathbb{N}$, and for each time point t, the vector $\mathbf{x}_t^{(i)} \in \mathbb{R}^M$ has $M \in \mathbb{N}$ channels. The goal of VTSC is to learn a classifier f_θ which maps each series $(\mathbf{x}_t^{(i)})_{t=1}^{T_i}$ to its corresponding class in $\{1, \ldots, K\}$ for $K \in \mathbb{N}$ classes. We require that f_θ can handle sequences of any length—that is, it has infinite context—since we assume that each T_i can be arbitrarily large at inference time. Otherwise, we must adjust a finite-context classifier using padding, truncation, or interpolation.

During each training epoch, SSS performs a sampling procedure *without replacement* to create each batch. This method allows us to process sequences of arbitrarily length, while maintaining constant memory complexity (equivalent to batch size) for efficient

training. We first construct the set of all windows $\mathcal{W}$ by performing the slicing window method over each individual time series in X. For a given window size $L \in \mathbb{N}$, with window stride S, and a time series i with sequence length T_i, we obtain $A_i = \lfloor \frac{T_i - L}{S} \rfloor + 1$ windows. Note that $A_i \propto T_i$ for each i. During training, we convert $\mathcal{W}$ to a PyTorch dataset and use the native PyTorch dataloader with batch size B. When a batch is sampled, windows are drawn uniformly from $\mathcal{W}$, and thus for each i, the probability of observing a window from series i is given by:

$$p_i = A_i / (\sum_{j=1}^{n} A_j) \approx T_i / (\sum_{j=1}^{n} T_j).$$

If N_i represents the number of windows in the batch from series i, then we achieve the sampling property of $N_i \sim \text{Binomial}(B, p_i)$, where $p_i \approx T_i / (\sum_{j=1}^{n} T_j)$, i.e., the probability of a given window in the batch being from series i is proportional to its relative sequence length. From a memory complexity perspective, the core issue of variable-length modeling is processing information from long sequences, while minimizing the memory footprint from computing gradients with standard methods such as Adam [60]. By sparse training, we bypass this constraint by utilizing a local model to process each window independently, thus the memory complexity of SSS is simply a function of batch size and window size (fixed parameters), not total sequence length. Proportional sampling also ensures fair representation of each series based on its length, allowing longer sequences—which contain more information—to contribute more samples. By sampling only a subset of windows, SSS introduces sparsity into the training process, reducing computational cost found in finite-context methods, and reduces the likelihood of context overload seen in infinite-context methods. Also note that by sampling without replacement, the model observes each window exactly

once during a single training epoch.

4.3.2 Stochastic Sparse Sampling Training Algorithm

After sampling a batch of windows $\mathcal{W}_0 = \{\mathbf{w}_1, \ldots, \mathbf{w}_B\}$, each $\mathbf{w}_b \in \mathcal{W}_0$ is processed independently by a local model f_θ to obtain a window prediction (probability) $\hat{y}_b = f_\theta(\mathbf{w}_b) \in [0,1]^K$, representing our probability distribution over $K \in \mathbb{N}$ classes. The choice of f_θ can be any time series backbone, in our experiments we select PatchTST [88]. For each $1 \leq i \leq n$, we define the subset of all windows from series i as:

$$\mathcal{W}_i = \{\mathbf{w} \in \mathcal{W}_0 \mid \mathbf{w} \text{ is from series } i\}, \tag{4.1}$$

and let:

$$\mathcal{Y}_i = \{f_\theta(\mathbf{w}) \mid \mathbf{w} \in \mathcal{W}_i\}, \tag{4.2}$$

be the set of multiset of window probabilities. To obtain the global prediction (probability) for time series i, we aggregate all window probabilities from all examples that are originating from series i, i.e.,

$$\hat{y}^{(i)} = \text{Aggregate}(\mathcal{Y}_i) = \sum_{\hat{y} \in \mathcal{Y}_i} \alpha_{\hat{y}} \hat{y}, \tag{4.3}$$

where each $\alpha_{\hat{y}} \in [0,1]$ is a weight for the window probability $\hat{y}$ given to the final output, with the constraint that $\sum_{\hat{y} \in \mathcal{Y}_i} \alpha_{\hat{y}} = 1$; that is, Aggregate$(\cdot)$ produces a *convex combination* over all window probabilities in $\mathcal{Y}_i$. In our experiments, we use mean aggregation, i.e., $\alpha(\hat{y}) = \frac{1}{|\mathcal{Y}_i|}$ for all $\hat{y} \in \mathcal{Y}_i$, due to its simplicity and effectiveness for

our current objectives. However, any convex aggregator guarantees that $\hat{y}^{(i)}$ remains a valid probability distribution over K classes, and allows for potential of non-uniform aggregation functions, enabling weighting of window predictions based on factors such as prediction uncertainty, or frequency characteristics. Theorem 3 shows that under convex aggregation, for binary or multiclass classification, a valid probability distribution is always guaranteed for $\hat{y}^{(i)}$.

Theorem 3 (Probability Distribution Guarantee)**.** *Fix $K, n \in \mathbb{N}$. Suppose $\alpha_1, \ldots, \alpha_n \geq 0$ satisfies $\sum_{i=1}^{n} \alpha_i = 1$, and $\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_n \in [0,1]^K$ each satisfy $\sum_{j=1}^{K} \hat{y}_{ik} = 1$ for $1 \leq i \leq n$. That is, each $\hat{\mathbf{y}}_i = (\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{iK})^T$ represents a valid discrete probability distribution over K classes (e.g., softmax output). Then the convex combination:*

$$\hat{\mathbf{y}}^{(c)} = \sum_{i=1}^{n} \alpha_i \hat{\mathbf{y}}_i,$$

also represents a valid discrete probability distribution, where $\hat{\mathbf{y}}^{(c)} = (\hat{y}_1^{(c)}, \hat{y}_2^{(c)}, \ldots, \hat{y}_K^{(c)})^T$ represents the channel (total time series) probability, and thus satisfies $\sum_{j=1}^{K} y_j = 1$.

Proof. By construction, $\hat{y}_j^{(c)} = \sum_{i=1}^{n} \alpha_i \hat{y}_{ij}$ for each entry $1 \leq j \leq K$. Then $\hat{y}_j^{(c)} \geq 0$, since for each $1 \leq i \leq n$ and $1 \leq j \leq K$ we are given that $\alpha_i \geq 0$ and $\hat{y}_{ij} \geq 0$. Furthermore, we can write:

$$\begin{aligned}
\sum_{j=1}^{n} \hat{y}_j^{(c)} &= \sum_{j=1}^{K} \sum_{i=1}^{n} \alpha_i \hat{y}_{ij} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{K} \alpha_i \hat{y}_{ij} && \text{(Swap summation order)} \\
&= \sum_{i=1}^{n} \alpha_i \sum_{j=1}^{K} \hat{y}_{ij}
\end{aligned}$$

$$= \sum_{i=1}^{n} \alpha_i \qquad \left(\textstyle\sum_{j=1}^{K} \hat{y}_{ij} = 1 \text{ for all } i\right)$$

$$= 1 \qquad \left(\textstyle\sum_{i=1}^{n} \alpha_i = 1\right)$$

It follows that since each $y_j \geq 0$ and $\sum_{j=1}^{K} y_j = 1$, then $\hat{\mathbf{y}}^{(c)}$ is a valid discrete probability distribution over K classes. □

Note that in our experiments, we consider only binary classification for $K = 2$, however, the above formulation shows that SSS extends naturally to multiclass classification as well with any convex aggregator.

Algorithm 1 SSS Training Algorithm (Single Epoch)

Input: Time series $X = \left\{(\mathbf{x}_t^{(1)})_{t=1}^{T_1}, \ldots, (\mathbf{x}_t^{(n)})_{t=1}^{T_n}\right\}$, labels $Y = \{y^{(1)}, \ldots, y^{(n)}\}$, model f_θ with parameters θ, batch size B, loss function $\mathcal{L}$
Output: Updated model parameters θ
$\mathcal{W} \leftarrow$ Set of all windows from each series in X
while $\mathcal{W} \neq \emptyset$ **do**
  ▷ Sample B windows with probability $T_i / \left(\sum_j T_j\right)$ from series i, for all i
  $\mathcal{W}_0 \leftarrow \textsc{Sample}(\mathcal{W}, B)$
  for $i = 1, \ldots, n$ **do**
    $\mathcal{W}_i \leftarrow \{\mathbf{w} \in \mathcal{W}_0 \mid \mathbf{w} \text{ is from series } i\}$ ▷ Windows from series i
    $\mathcal{Y}_i \leftarrow \{f_\theta(\mathbf{w}) \mid \mathbf{w} \in \mathcal{W}_i\}$ ▷ Window probabilities for series i
    $\hat{y}^{(i)} \leftarrow \textsc{Aggregate}(\mathcal{Y}_i)$ ▷ Final probability for series i
  end for
  $\mathcal{L}_{\text{batch}} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$ ▷ Loss over the batch
  $\theta \leftarrow \textsc{Update}(\theta, \mathcal{L}_{\text{batch}})$ ▷ Update local model parameters
  $\mathcal{W} \leftarrow \mathcal{W} \setminus \mathcal{W}_0$ ▷ Remove sampled windows from the pool
end while
return θ

4.3.3 Theoretical Considerations of Stochastic Sparse Sampling

Our prediction for a given univariate time series is formulated as a Monte Carlo estimate, a technique in which the true average over all possible windows is efficiently approximated by the mean of a smaller, randomly drawn sample. From the set of all available windows, we draw a random sample of size n, denoted $\{W_1, \ldots, W_n\}$, where each W_i is a random variable representing a sampled window. These are transformed by our neural network f_θ to produce output probabilities, also treated as random variables, where $Y_i = f_\theta(W_i)$. In this framework, the uppercase W_i and Y_i represent the stochastic process of sampling and prediction, distinct from their specific realized values used in computation (i.e., a data vector $\mathbf{w}_i$ and its scalar output $\hat{y}_i$). The final prediction, $Y = \frac{1}{n}\sum_{i=1}^{n} Y_i$, is the sample mean of these random variables. Because n is small, Y serves as a noisy but computationally efficient and unbiased estimator of the true mean prediction. This inherent stochasticity acts as a form of implicit regularization, discouraging f_θ from overfitting to any single window's features. While the temporal nature of the data introduces autocorrelation that partially violates the strict i.i.d. assumption, the Central Limit Theorem provides theoretical motivation, suggesting that the distribution of our estimator Y approaches normality as n increases, thereby stabilizing the learning process.

4.3.4 Alternative Sampling Approaches

Beyond uniform random sampling, alternative strategies can structure the information presented to the network to improve learning dynamics. For instance, *stratified sampling* could be employed by first identifying distinct operational regimes within the time series and then sampling windows proportionally from each stratum. This

ensures that every mini-batch provides a balanced and representative gradient signal to f_θ, forcing the model to learn from a diverse set of patterns in every update step.

To perform stratified sampling, the entire population of windows $\mathcal{W}$ is partitioned into K mutually exclusive strata $\{S_1, S_2, \ldots, S_K\}$. This partitioning is based on a meaningful characteristic, such as the window's statistical properties or its temporal proximity to a labeled event. In seizure onset zone localization, it may be beneficial in future studies to form strata based on medically relevant phases: *pre-ictal* (before a seizure), *ictal* (during a seizure), and *post-ictal* (after a seizure).

For each time series, to form a training subset of size n, we draw a specific number of windows, n_k, from each stratum S_k, such that $\sum_{k=1}^{K} n_k = n$. The number of samples n_k is typically chosen to be proportional to the relative size of its stratum, $|S_k|$. The n_k windows from a given stratum are then sampled uniformly at random from within that stratum. This approach may be particularly powerful for imbalanced datasets, as it guarantees the model is consistently exposed to rare but critical patterns (e.g., the ictal phase), preventing the learning process from being biased towards the far more prevalent inter-ictal state and thus promoting a more robust final model.

4.3.5 Calibrated Inference

To derive the prediction for a time series $(\mathbf{x}_t^{(i)})_{t=1}^{T_i}$ at inference time, we utilize *all* windows from the selected time series, to form its final prediction $\hat{y}^{(i)}$. Since there are little computational constraints during inference (e.g., no gradient computations), sparsity is no longer required for large sequences, and thus we can utilize all information to make final predictions. Let $\mathcal{W}_i$ be the collection of all windows from series i. We pass each window through the local model to obtain the multiset of all window

probabilities $\mathcal{Y}_i$ as shown in Equation (4.2). Before the aggregation step, we utilize a calibrator $g_\phi : [0,1]^K \to [0,1]^K$, which adjusts each individual window probability to reduce the presence of noise, and define:

$$\tilde{\mathcal{Y}}_i = \{g_\phi(\hat{y}) \mid \hat{y} \in \mathcal{Y}_i\}, \tag{4.4}$$

which is then fed into the final prediction during the aggregation step $\hat{y}^{(i)} = \text{Aggregate}(\tilde{\mathcal{Y}}_i)$. By calibrating the window probabilities before aggregation, we correct for biases or misestimations in the predicted probabilities, which occur when the output probabilities of the local models do not accurately reflect the true likelihood of the event. We consider isotonic regression and Venn-Abers predictors for our calibration method [111, 121]. It is important to note, that these calibration techniques do not alter underlying structure of f_θ and do not utilize input features from the time series; rather, they adjust the output probabilities to mitigate the effect of noise during the aggregation step.

4.3.6 Probability Calibration Methods

Let $\hat{y}_1, \ldots, \hat{y}_n \in [0,1]$ be the uncalibrated window probabilities, each with a corresponding binary label $y_1, \ldots, y_n \in \{0,1\}$ derived from the label of the time series; that is, if $\hat{y}_i = f_\theta(\mathbf{w}_i)$ for a window $\mathbf{w}_i$, then the window label y_i is inherited from the global time series it was sampled from. The goal of probability calibration is to transform each $\hat{y}_i$ into $\tilde{y}_i = g_\phi(\hat{y}_i)$, such that $\tilde{y}_i$ represents true likelihood of a class. Within this context, probability calibration can help mitigate the impact of temporal fluctuations and local anomalies by adjusting probabilities for each individual windows. Note that while calibration may yield more refined probability estimates with reduced noise, it

is an integral intermediate step rather than a global optimization procedure on the final probabilities. For each calibration we considered, we provide a short description below.

Isotonic Regression-based Calibration

is a nonparametric method that fits a weighted least-squares model subject to monotonicity constraints [111]. Formally, this can be stated as a quadratic program (QP) given by:

$$\begin{aligned} &\min_{g} \sum_{i=1}^{n} w_i(\tilde{y}_i - y_i)^2 \\ &\text{subject to } \tilde{y}_i \leq \tilde{y}_j, \quad \forall i, j \text{ where } \hat{y}_i \leq \hat{y}_j. \end{aligned} \tag{4.5}$$

where $\tilde{y}_i = g(\hat{y}_i)$ and $w_i \geq 0$ are weights assigned to each datapoint, which are often each set to $w_i = 1$ to provide equal importance over all inputs. The monotonicity constraint ensures that uncalibrated probabilities will always map to equal or higher calibrated probabilities.

Venn-Abers Predictors-based Calibration

is based on the concept of isotonic regression but extends it to ensure validity within the framework of conformal prediction, which provides uncertainty estimates with distribution-free theoretical guarantees [12, 121]. For an uncalibrated probability $\hat{y}$, two isotonic calibrators are trained:

$$p_0 = g_0(\hat{y}) \text{ and } p_1 = g_1(\hat{y}) \tag{4.6}$$

where g_0 and g_1 are isotonic functions derived from augmented sets. These sets include $(\hat{y}, 0)$ and $(\hat{y}, 1)$ respectively, alongside all other uncalibrated probabilities and their respective labels. The values p_0 and p_1 represent likelihoods for class 0 and class 1, while the interval $[p_0, p_1]$ provides an uncertainty estimate of where the true probability resides. The final calibrated probability is then given by:

$$\tilde{y} = \frac{p_1}{1 - p_0 + p_1}. \tag{4.7}$$

Venn-Abers predictors provide guaranteed validity in terms of calibration, meaning the predicted probabilities closely match empirical frequencies. In comparsion to isotonic regression, Venn-Abers can be more computationally intensive, as it requires fitting two isotonic functions simultaneously.

4.4 Datasets, Baselines, and Data Pre-processing

In this section, we describe the details of the considered datasets and benchmarks. We also present the data pre-processing required for finite and infinite context methods.

4.4.1 iEEG Datasets

The Epilepsy iEEG Multicenter Dataset[1] consists of iEEG signals with SOZ clinical annotations from four medical centers including the Johns Hopkins Hospital (JHH), the National Institute of Health (NIH), University of Maryland Medical Center (UMMC), and University of Miami Jackson Memorial Hospital (UMH). Since UMH contained only a single patient with clinical SOZ annotations, we do not consider it in our main evaluations. However, we use UMH within the multi-center evaluation in Table 4.3

[1] https://openneuro.org/datasets/ds003029/versions/1.0.7

and the training set for OOD experiments for SOZ localization on unseen medical centers in Table 4.5. We select the F1-score, Area Under the Receiver Operator Curve (AUC), and accuracy for our evaluation metrics.

Table 4.1: iEEG Multicenter Dataset Summary: For each medical center, we report the total number of patients recorded (n), the number of patients with SOZ annotations (n_{SOZ}), the number of time series recordings (n_{ts}), the percentage of time series labeled as SOZ (p_{SOZ}), the type of iEEG method used (e.g., electrocorticography, ECoG), the sampling frequency (Hz) (some recordings may vary), and post-operative patient outcomes following SOZ surgical resection.

Medical Center	n	n_{SOZ}	n_{ts}	p_{SOZ}	**iEEG Type**	**Frequency (Hz)**	**Patient Outcomes**
JHH	7	3	1458	7.48%	ECoG	1000	No
NIH	14	11	3057	12.23%	ECoG	1000	Yes
UMMC	9	9	2967	5.56%	ECoG	250-1000	Yes
UMF	5	1	129	25.58%	ECoG	1000	No

Table 4.1 provides an overview of the iEEG Multicenter Dataset. For each center, we filter out patients (n) who have SOZ annotations (n_{SOZ}). All channels for all patients are grouped together into one dataset per medical center, where n_{ts} indicates the number of examples. However, due to the heavy imbalance between SOZ-labeled and non-SOZ labeled time series, the number of examples used for training and validation decreases significantly. We employ class balancing, resulting in $\lfloor 2 \cdot p_{\text{SOZ}} \cdot n_{\text{ts}} \rfloor$ examples for each medical center, which is then split into training, validation, and testing.

4.4.2 Finite and Infinite Context Baselines

For finite-context baselines, we include a variety of modern time series backbones including PatchTST, which uses sub-windows as tokens in combination with the traditional Transformer architecture [88, 119]. TimesNet uses TCN architecture that models both inter-period and intra-period dynamics by leveraging Fast Fourier Transform (FFT) features to slice the signal into multiple views, processed later

through inception blocks [55, 126]. ModernTCN is a recently proposed TCN which decouples temporal and channel information processing by using separate DWConv and ConvFFN modules for more efficient representation learning [79]. DLinear is a linear neural network, which outperforms several modern Transformer-based architectures, utilizing traditional seasonal-trend decomposition techniques [134].

For our infinite-context baselines, we utilize ROCKET, which applies randomly initialized, fixed convolutional kernels to the input sequence. ROCKET compresses the resulting convolutional outputs to the maximum value and the proportion of positive values (PPV), where these features then fed to a linear classifier [31]. We also consider GRUs and an LSTM network, both of which are popular RNN frameworks designed to capture long-term dependencies in sequential data [14, 50]. Additionally, we use the recent SSM architecture, Mamba, which applies selective state updates for efficient long-range dependency modeling [45].

For each baseline, we perform comprehensive grid search over multiple hyper-parameters and optimize with respect to best accuracy score on the validation set for all medical centers. Each experiment is conducted over 5 random seeds, with the average result reported for the best configuration. In all experiments, we train using the Adam optimizer [60], for 50 epochs, with cosine learning rate annealing (one cycle with 50 epochs in length) which adjusts the learning rate down by two orders of magnitude (e.g., 10^{-4} to 10^{-6}) by the last epoch. We also implement early stopping with a patience of 15, and apply reversible instance normalization [59] for each input. For most of the baselines we use a dropout rate of $0.2 - 0.3$ and weight decay to $10^{-4} - 10^{-5}$, but do not explicitly tune these parameters in our grid search. For finite-context methods, we set the batch size to the entire dataset (596 individual

univariate time series for all medical centers), whereas infinite-context methods use a batch size of 1 due to process each variable-length sequence in its entirety. Our search space on the PatchTST backbone used in SSS is provided in Table 4.2.

Table 4.2: Hyperparameter search space for PatchTST backbone used in SSS. Best configuration is highlighted in red.

Parameter	**Search Values**
d_{model}	$\{16, 32, 64\}$
d_{ff}	$\{32, 64, 128\}$
`num_heads`	$\{2, 4, 8\}$
`num_enc_layers`	$\{1, 2, 3\}$
`lr`	$\{10^{-4}, 10^{-5}\}$
L	$\{1000, 3000, 5000, 10000\}$

4.4.3 Data Pre-processing

Each patient recording contains multiple channels corresponding to individual electrodes from the iEEG device. During pre-processing, for all patients, we extract all channels and balance the dataset to have an equal number of SOZ and non-SOZ channels. After, we partition this dataset into train, validation, and test channels with a percentage of $70\%, 10\%, \text{and} 20\%$ split, respectively, and ensure that during the window sampling phase of SSS there is no temporal leakage from the test set. Each channel, or univariate time series, is $z-$score normalized to have zero mean and unit standard deviation. Due to the extremely long sequence length of several channels, we downsample to fit the dataset into memory (250GB RAM). To achieve this, for each channel, we applied a 1D average pooling layer with `kernel_size`=24 and `kernel_stride`=12 before feeding it to the baseline model or before performing the window sampling procedure for SSS at train-time. This downsampling was applied

to SSS and all baselines for fair comparison.

For each finite-context method, we perform a combination of padding and truncation according to the chosen window size L. If the sequence length of the original time series exceeds L, it was truncated to obtain the first L time points, otherwise if it was less than L, it was padded with zeros at the end of the sequence to ensure a sequence length of L.

4.5 Numerical Results and Discussions

In this section, we present numerical results, ablations, and their relevant discussions. Our experiments were conducted on 4x NVIDIA RTX 6000 Ada Generation GPUs using the PyTorch framework with CUDA [94].

SOZ Localization

Table 4.3 displays experimental results for SOZ localization across individual and combined medical centers. SSS outperforms all baselines on every evaluation metric in the multicenter evaluation and shows strong performance for JHH and NIH centers, with comparable results at UMMC. The variation in UMMC performance likely stems from its inconsistent sampling frequencies (250-1000 Hz) compared to the uniform 1000 Hz at JHH and NIH. Notably, finite-context methods generally outperform infinite-context approaches for in-distribution univariate VTSC, with SSS and other finite-context methods consistently rank among the top performers.

We attribute SSS' performance increase due to its ability to learn local signal representations, learning invariant characteristics across various signal conditions. This capability is especially valuable for SOZ localization, which presents significant

Table 4.3: SOZ localization. F1 score, AUC, and Accuracy are reported for each medical center, averaged over 5 seeds. For each center, we train and evaluate a separate model; the first column represents training and evaluation on all centers. **Bolded** values with * and † denote the best and second-best results, respectively.

	All			JHH			NIH			UMMC		
Model	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC	Acc.
SSS (Ours)	**0.7629***	**0.7999***	**72.35***	**0.8187***	**0.8851***	**81.37***	**0.6716***	0.6853	**64.22†**	**0.7978†**	**0.8279†**	76.06
PatchTST [88]	**0.7097†**	**0.7852†**	66.83	**0.7419†**	**0.8045†**	71.82	0.6402	**0.7036†**	62.11	**0.8015***	0.8121	77.58
TimesNet [126]	0.6897	0.7174	65.98	0.6891	0.8029	**73.64†**	0.5950	0.6806	**66.00***	0.7821	0.8099	**77.06†**
ModernTCN [79]	0.6938	0.7305	68.42	0.6710	0.7508	67.73	0.5055	**0.7220***	64.00	0.6371	0.8203	71.76
DLinear [134]	0.6916	0.7044	68.41	0.6873	0.7395	66.36	0.6055	0.6405	59.50	0.7658	0.7729	77.05
ROCKET [31]	0.6847	0.7481	69.27	0.6753	0.7752	69.09	**0.6520†**	0.6546	62.63	0.7686	0.7900	74.55
Mamba [45]	0.6452	0.7134	64.39	0.6456	0.6764	62.27	0.5974	0.6050	58.95	0.7900	**0.8424***	76.36
GRUs [14]	0.6948	0.7340	65.85	0.6140	0.6959	63.18	0.6171	0.6283	62.63	0.7920	0.8211	**77.27***
LSTM [50]	0.6709	0.7144	65.43	0.6571	0.6190	59.09	0.5657	0.5909	54.74	0.7604	0.8060	73.64

challenges due to both intra-patient and inter-patient variability. Our evaluation, which incorporates three heterogeneous datasets, provides a rigorous testbed for assessing SSS's generalizability in learning local signal characteristics within medical time series. The results depict that sparse training is sufficient for achieving SOTA performance.

Out-of-Distribution SOZ Localization

Table 4.5 reports our results for SOZ localization in the OOD setting. At training time, from the collection of four medical center datasets, we omit one and train on the remaining three. On inference time, we test solely on the omitted medical center to gauge how well each method performs OOD. For iEEG signals from epilepsy patients, inter-patient variability can be significant due to differences in placement of electrodes in the brain, and the inherent heterogeneity of epileptogenic networks

Table 4.4: SOZ localization. 95% confidence intervals for F1 score, AUC, and Accuracy are reported for each medical center over 5 seeds. For each center, we train and evaluate a separate model; the first column represents training and evaluation on all centers. **Bolded** values with * and † denote the best and second-best results, respectively, for the mean scores from Table 4.3.

	All			JHH			NIH			UMMC		
Model	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC	Acc.
SSS (Ours)	[.72, .79]	[.76, .87]	[63, 78]	[.74, .9]	[.76, .97]	[71, 90]	[.6, .74]	[.66, .76]	[57, 72]	[.75, .84]	[.72, .9]	[70, 82]
PatchTST	[.57, .73]	[.61, .8]	[58, 73]	[.7, .79]	[.73, .88]	[65, 79]	[.54, .76]	[.66, .79]	[53, 75]	[.7, .87]	[.71, .93]	[69, 86]
TimesNet	[.6, .78]	[.66, .78]	[62, 70]	[.56, .82]	[.72, .88]	[67, 80]	[.38, .81]	[.48, .88]	[51, 81]	[.72, .85]	[.76, .86]	[71, 83]
ModernTCN	[.63, .76]	[.68, .78]	[63, 74]	[.61, .74]	[.64, .87]	[58, 77]	[.14, .95]	[.63, .82]	[50, 78]	[.40, .88]	[.74, .9]	[59, 85]
DLinear	[.62, .76]	[.6, .81]	[.62, .75]	[.62, .75]	[.63, .84]	[58, 74]	[.43, .78]	[.55, .73]	[47, 72]	[.69, .84]	[.67, .88]	[70, 84]
ROCKET	[.56, .81]	[.69, .8]	[64, 75]	[.58, .77]	[.66, .89]	[60, 79]	[.57, .73]	[.57, .74]	[55, 70]	[.69, .85]	[.7, .88]	[66, 83]
Mamba	[.54, .75]	[.65, .77]	[59, 70]	[.51, .78]	[.6, .76]	[54, 71]	[.47, .73]	[.42, .79]	[46, 72]	[.74, .84]	[.76, .93]	[71, 82]
GRUs	[.64, .75]	[.69, .78]	[60, 72]	[.47, .75]	[.57, .83]	[49, 77]	[.48, .76]	[.57, .69]	[52, 73]	[.73, .86]	[.74, .9]	[71, 84]
LSTM	[.61, .73]	[.65, .78]	[62, 69]	[.61, .70]	[.52, .72]	[50, 68]	[.48, .65]	[.49, .70]	[49, 61]	0.7604	0.8060	73.64

across individuals. Thus, even among medical time series, this can be one of the most challenging tasks to perform OOD.

Table 4.5: Out-of-Distribution SOZ localization. For each center, we train on all other centers and evaluate on the selected center. **Bold** values with * and † denote the best and second-best results, respectively.

	JHH			NIH			UMMC		
Model	F1	AUC	Acc.	F1	AUC	Acc.	F1	AUC	Acc.
SSS (Ours)	**0.6981***	**0.6590***	**57.80***	**0.6492***	**0.6092***	**54.73†**	**0.7243***	**0.8048***	**72.42***
PatchTST [88]	0.6175	0.5267	50.46	0.5986	0.4829	48.17	0.5067	0.5274	57.63
TimesNet [126]	0.5261	0.4501	47.00	0.4461	0.4407	45.85	0.3177	0.3108	46.14
ModernTCN [79]	0.4934	0.4970	49.54	0.4019	0.4651	48.71	0.3804	0.4474	50.55
DLinear [134]	0.4205	0.4775	47.25	0.5090	0.4945	50.54	0.5602	0.5236	56.00
ROCKET [31]	0.5784	0.5777	**56.71†**	0.5051	0.5522	52.91	0.5608	0.5941	58.36
Mamba [45]	0.5790	**0.5835†**	55.68	**0.6183†**	**0.5767†**	**55.69***	0.5715	0.5953	55.76
GRUs [14]	0.5779	0.4868	48.80	0.5824	0.5588	53.66	**0.6689†**	**0.7645†**	**69.30†**
LSTM [50]	**0.6362†**	0.5165	50.92	0.5774	0.5678	53.87	0.6581	0.6616	62.36

SSS demonstrates superior performance across all medical centers in the OOD setting, across all medical centers, with significant performance gaps between several baselines. For example, on the UMMC dataset, there's a substantial difference between model performance, with TimesNet achieving only 46.14% accuracy while SSS reaches 72.42%. This contrasts to the in-distribution performance on UMMC, where results are much closer (the lowest performing model achieves 71.76% while SSS reaches 76.06%). This pattern isn't an isolated case and can be seen as a general trend, especially among finite-context methods where performance consistently drops in the OOD setting. While Table 4.3 shows that top-performing baselines for in-distribution classification are largely finite-context methods, Table 4.5 reveals that infinite-context methods (including SSS) exclusively occupy the Top 2 positions for OOD SOZ Localization. We hypothesize that finite-context methods experience performance degradation primarily due to challenges with long sequence lengths. Since these methods use parameters proportional to sequence length, they tend to overfit the training data, leading to the performance discrepancies observed in our results. In contrast, SSS exhibits robust performance in both settings—maintaining strong results in-distribution (where finite-context methods typically excel) and out-of-distribution (where infinite-context methods demonstrate superior performance).

Our experiments from Table 4.3 and Table 4.5 suggest that SSS may be learning local signal characteristics present in different patient populations, leading to an advantage over finite-context and infinite-context methods. Our visualizations of SSS's predictions OOD in Figure 4.3 supports this hypothesis, as there exist clear qualitative differences in locally averaged window probabilities with respect to anomalous signal characteristics, such as spikes or increases in amplitude or frequency. Figure 4.2

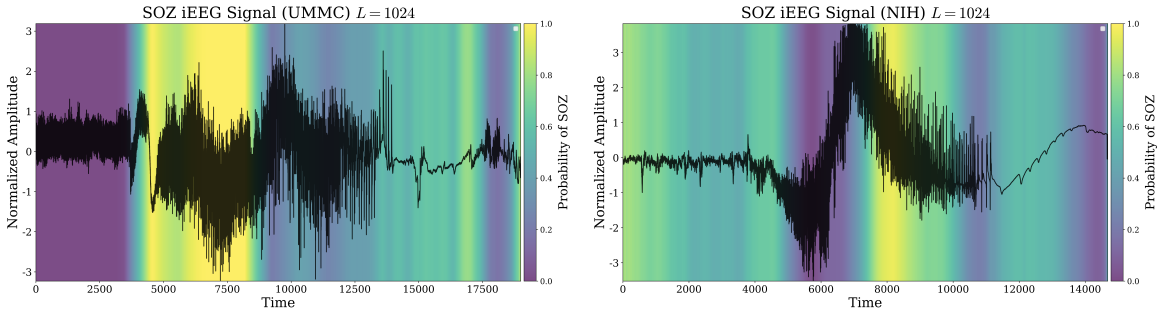


Figure 4.2: Visualization of locally averaged window probabilities with the SSS framework for in-distribution iEEG channels with $L = 1024$

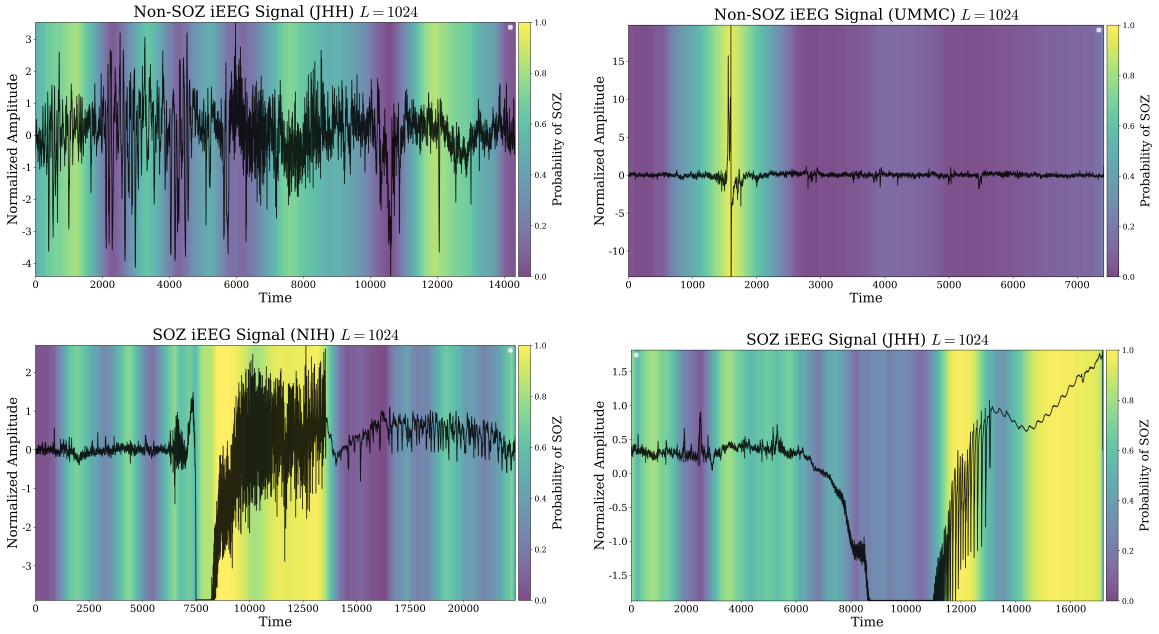


Figure 4.3: Visualization of locally averaged window probabilities with the SSS framework for OOD iEEG channels (patients completely unseen during training) with $L = 1024$.

shows SSS's predictions in-distribution which also suggest a form of implicit semantic segmentation for anomalous local regions of the signal with respect to the SOZ probability. More analysis is needed to further solidify our understanding of the interaction between qualitative signal characteristics and SOZ probabilities, where this would benefit from a rigorous explainability study and consulting with medical

experts for future studies.

Impact of Batch Size

While the aggregation function in Eq (4.3) may benefit from a higher number of samples within the batch (due to mean approximation), we observe that in Table 4.6, SSS' performance remains relatively constant across various batch sizes.

Table 4.6: Performance of SSS over various batch sizes.

Batch Size	**F1 Score**	**AUC**	**Acc. (%)**
128	0.7563 ± 0.02717	0.8139 ± 0.04302	70.79 ± 2.323
512	0.7498 ± 0.02354	0.7965 ± 0.03526	69.46 ± 3.109
2048	0.7651 ± 0.03568	0.8194 ± 0.05166	71.40 ± 6.078
4096	0.7441 ± 0.02987	0.7979 ± 0.06818	68.09 ± 4.848
8192	0.7629 ± 0.02829	0.7999 ± 0.05331	72.35 ± 4.965

Impact of Window Size

Table 4.7 examines the effect of window size (L) on SSS' performance. While the performance of $L = 512$ and $L = 1024$ remain relatively close, we notice that all evaluation metrics drop significantly for $L = 2048$. This suggests that a large receptive field is not always advantageous, and that SSS benefits from balanced receptive field and a relatively small window size.

Table 4.7: Performance of SSS over various window sizes L.

L	**F1 Score**	**AUC**	**Acc. (%)**
512	0.7567 ± 0.01075	0.8141 ± 0.03054	70.62 ± 2.165
1024	0.7629 ± 0.02829	0.7999 ± 0.05331	72.35 ± 4.965
2048	0.7334 ± 0.03003	0.7719 ± 0.04762	68.52 ± 3.353

Impact of Calibration

We observe that while the performance of Venn-Abers Predictors and isotonic regression remain relatively close, Table 4.8 displays that removing calibration significantly degrades model performance, indicating calibration effectively reduces noise among window probabilities.

Table 4.8: Performance of SSS with different calibration methods.

Calibration Method	**F1 Score**	**AUC**	**Acc. (%)**
Isotonic Regression	0.7629 ± 0.02829	0.7999 ± 0.05331	72.35 ± 4.965
Venn-ABERS	0.7637 ± 0.02704	0.8003 ± 0.05308	72.47 ± 4.773
No Calibration	0.7291 ± 0.06909	0.7830 ± 0.03844	69.93 ± 4.635

4.6 Conclusion

This work introduces novel VTSC framework, **S**tochastic **S**parse **S**ampling (SSS), tailored for medical time series applications with variable-length sequences. SSS blends the best of both worlds between finite-context methods (enabling usage of modern finite-context backbones) while allowing sampling of the entire signal in a computationally efficient manner, that is less prone to context overload from infinite-context methods, and less likely to overfit compared to finite-context methods. SSS learns local signal characteristics and aggregates this information to provide a global prediction, which provides inherent interpretability and superior performance to the SOTA in-distribution and OOD. For future work, we plan to extend this work by: (1) benchmarking SSS across a wider variety of variable-length medical time series to validate its generalizability, (2) providing further rigorous post-hoc insights into the window probability distribution learned by SSS, and (3) incorporating uncertainty

estimates within the aggregation function to highlight anomalous regions of the signal, potentially enhancing clinical decision support.

Chapter 5

Conclusions & Future Directions

5.1 Conclusion

The first study introduces Stochastic Sparse Sampling (SSS), a framework for classifying variable-length medical time series. Instead of forcing every signal into a fixed format, SSS randomly draws short windows from each recording, feeds them through a single local model, and mean-aggregates the resulting probabilities. This simple routine lets modern finite-context backbones handle arbitrarily long sequences without padding or truncation, while avoiding the "context overload" that plagues infinite-context models. Across four epilepsy centres, SSS consistently outperformed all contemporary baselines both *in-distribution* and, more importantly *out-of-distribution* generalization, where recordings came from unseen hospitals with different hardware and sampling rates. The window-level probabilities double as heat-maps that highlight likely seizure-onset regions, providing clinicians with interpretable cues that dovetail with their visual workflow and reducing the risk of mis-localizing epileptogenic tissue. Because training cost scales only with window and batch size, the method remains GPU-friendly even on hour-long intracranial EEG, suggesting a practical path toward multi-centre clinical

deployment.

The companion work, *EMForecaster*, applies the same local-window philosophy to wireless-network telemetry. A patch-embedding backbone captures multi-scale electromagnetic-field (EMF) patterns, reversible instance normalization handles distribution shifts, and a distribution-free *conformal prediction* layer wraps every point forecast with a statistically valid uncertainty band. Experiments on diverse short- and long-term EMF datasets show that EMForecaster decisively outperforms state-of-the-art deep-learning baselines in both point accuracy and interval quality, striking a better balance between coverage and width according to the proposed *Trade-off Score*. For network operators, this means they can plan power, spectrum, and siting decisions days in advance with quantified risk; for regulators and public-health agencies, it provides an auditable tool for ensuring exposure remains within safety limits even as 5G and beyond networks densify.

Together, the two studies demonstrate how task-specific sampling and uncertainty-aware forecasting can generalize across domains—yielding robust, interpretable models that translate directly into better clinical and engineering decisions.

5.2 Future Directions

Below we discuss future directions in the domain of health care and wireless networks.

5.2.1 Health Care Applications

Pathology–Signal Linkage through Local Signal Analysis

A long-standing open question in epilepsy surgery is whether the electrophysiology of the SOZ contains recurrent signal characteristics that are *pathognomonic* for the

underlying tissue abnormality [72]. The next iteration of SSS should therefore couple its window-level probabilities with *local attribution tools*, e.g. SHAP values or gradient-based saliency maps, and then cluster the highest-importance fragments to recover candidate motifs [78, 116]. Today, epileptologists still rely heavily on expert visual inspection of intracranial EEG to decide whether a given electrode resides in the SOZ; these judgments guide resection margins and, ultimately, surgical outcomes. Automating "where to look" with an interpretable SSS pipeline would (i) surface the same salient windows that a specialist might scan for in their existing pipeline, (ii) quantify how strongly each window supports an SOZ hypothesis, and (iii) expose previously overlooked micro-patterns that correlate with poor post-operative seizure control. After extracting high-importance motifs, one can cross-reference them with post-operative outcomes and SOZ labels to test a simple hypothesis: *do motifs flagged by SSS differ systematically across histopathology classes, or do they instead represent generic markers of the SOZ?*

Causal-Inference Based Estimation of the True SOZ

Because the *true* SOZ is never directly observable, the only information we can reasonably extract are the (i) post-operative seizure outcomes and (ii) clinicians' *predicted* SOZ labels, which are the SOZ labels used in nearly all modern SOZ localization studies. However, given the low post-operative success rate (i.e. elpilepsy symptoms recede), these proxies known to contain high estimation error, and are not necessarily an accurate representation of the true SOZ sites. A causal-inference layer could explicitly treat that estimated label as an imperfect proxy for a latent "true SOZ," asking whether the presence of a given motif systematically shifts the odds of

belonging to that latent zone and, in turn, alters surgical success rates. Even basic covariate-matched comparisons between patients (or electrodes) with versus without the motif can start to untangle this mediated chain of influence, while instrument-based or counterfactual extensions remain on the table as data volume and clinical consensus grow. This approach replicates a clinician's workflow by providing rapid, interpretable suggestions, while simultaneously flagging probable misidentifications and guiding experts toward the latent true SOZ, thereby improving post-operative outcomes when resecting seizure onset zones.

5.2.2 Forecasting for Wireless Networks

Wireless Channel State Information Prediction

The upcoming generation of wireless networks will see a mix of low and high frequencies ranging from sub-6GHz to millimeter and sub-THz frequencies, referred to as multi-band networks [1, 51, 104]. Subsequently, channel sensing [18] and network traffic forecasting in multi-band networks will become critical. Furthermore, due to higher transmission frequencies, extremely-large antenna arrays can be supported on APs, thus the machine learning-enabled channel forecasting solutions should be scalable and capable of incorporating a variety of blockage and mobility constraints [2,7,8]. In order to further enhance the recognition accuracy in large-scale multi-band networks, edge learning and federated learning solutions with antenna selection and beamforming will be necessary [13, 133].

Temporally Adaptive Conformal Prediction

Adaptive conformal prediction (ACP) recalibrates its quantile at every forecast step, replacing the single fixed-width band of standard inductive conformal prediction with intervals that contract in stable periods and widen when residuals become volatile. Widely used implementations include Conformalized Quantile Regression (CQR), Jackknife+ predictive inference, and Adaptive Conformal Inference (ACI), all of which preserve the finite-sample coverage guarantees of conformal methods while allowing the interval width to vary with changing uncertainty [19, 39, 100].

When forecasting autocorrelated (serially dependent) time series data, the exchangeability assumption underpinning these guarantees no longer holds, because neighboring residuals are correlated rather than i.i.d. Temporal ACP variants therefore modify the calibration step to restore validity: (i) block conformal prediction forms non-overlapping residual blocks large enough that dependence across blocks is negligible (such as in the case of Conformal Time Series Forecasting); (ii) conformal PID control and other control-theoretic schemes adjust the interval width dynamically in response to forecast-error "feedback" [11]; and (iii) aggregation-based ACI extensions weight multiple adaptive experts, each tuned to a different volatility regime, to maintain efficient coverage under drift [132]. These strategies ensure that conformal intervals remain reliable even when temporal dependence or regime shifts would otherwise invalidate the fixed-width approach.

For wireless-network operators, volatility-sensitive ribbons translate into more confident power-cap decisions and fewer false alarms than the static alternative, thereby improving both compliance and spectrum efficiency in evolving RF environments.

References

[1] Sylvester Aboagye, Mohammad Amin Saeidi, Hina Tabassum, Yamin Tayyar, Ekram Hossain, Hong-Chuan Yang, and Mohamed-Slim Alouini. Multi-band wireless communication networks: Fundamentals, challenges, and resource allocation. *IEEE Trans. on Commun.*, 72(7):4333–4383, 2024.

[2] Sylvester Aboagye and Hina Tabassum. Thz network placement and mobility-aware resource allocation for indoor hybrid thz/vlc wireless networks. *IEEE Transactions on Communications*, 73(4):2554–2569, 2025.

[3] Sara Adda, Luca Chiaraviglio, Daniele Franci, Chiara Lodovisi, Nicola Pasquino, Settimio Pavoncello, Chiara Pedroli, and Renata Pelosini. High noon for mobile networks: Short-time emf measurements to capture daily exposure. *IEEE Trans. on Instrumentation and Measurement*, 72:1–10, 2023.

[4] Elena Agliari, Adriano Barra, Orazio Antonio Barra, Alberto Fachechi, Lorenzo Franceschi Vento, and Luciano Moretti. Detecting cardiac pathologies via machine learning on heart-rate variability time series and related markers. *Scientific Reports*, 10(1):8845, 2020.

[5] Elif Ak and Berk Canberk. Forecasting quality of service for next-generation data-driven wifi6 campus networks. *IEEE Transactions on Network and Service Management*, 18(4):4744–4755, 2021.

[6] Maarouf Al Hajj, Shanshan Wang, Lam Thanh Tu, Soumaya Azzi, and Joe Wiart. A statistical estimation of 5G massive MIMO networks' exposure using stochastic geometry in mmwave bands. *Applied Sciences*, 10(23), 2020.

[7] Mehrazin Alizadeh, Xavier Mootoo, Omer Waqar, and Hina Tabassum. Qos-aware deep unsupervised learning for star-ris assisted networks: A novel differentiable projection framework. *IEEE Wireless Communications Letters*, 2024.

[8] Mehrazin Alizadeh and Hina Tabassum. Power control with qos guarantees: A differentiable projection-based unsupervised learning framework. *IEEE Transactions on Communications*, 71(8):4605–4619, 2023.

[9] Edoardo Amaldi, Antonio Capone, and Federico Malucelli. Planning UMTS base station location: Optimization models with power control and algorithms. *IEEE Trans. on Wireless Commun.*, 2(5):939–952, 2003.

[10] Anastasios Angelopoulos, Stephen Bates, Jitendra Malik, and Michael I Jordan. Uncertainty sets for image classifiers using conformal prediction. *arXiv preprint arXiv:2009.14193*, 2020.

[11] Anastasios Angelopoulos, Emmanuel Candes, and Ryan J Tibshirani. Conformal pid control for time series prediction. *Advances in neural information processing systems*, 36:23047–23074, 2023.

[12] Anastasios N Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[13] Saba Asaad, Hina Tabassum, Chongjun Ouyang, and Ping Wang. Joint antenna selection and beamforming for massive mimo-enabled over-the-air federated learning. *IEEE Transactions on Wireless Communications*, pages 1–1, 2024.

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[15] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[16] M Rafet Bakcan et al. Measurement and prediction of electromagnetic radiation exposure level in a university campus. *Tehnivcki vjesnik*, 29(2):449–455, 2022.

[17] Sai Sanjay Balaji and Keshab K Parhi. Seizure onset zone identification from ieeg: A review. *IEEE Access*, 10:62535–62547, 2022.

[18] Borna Barahimi, Hakam Singh, Hina Tabassum, Omer Waqar, and Mohammad Omer. Rscnet: Dynamic csi compression for cloud-based wifi sensing. *arXiv preprint arXiv:2402.04888*, 2024.

[19] Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. Predictive inference with the jackknife+. *The Annals of Statistics*, 49(1):486–507, 2021.

[20] B Benson, WD Pan, A Prasad, GA Gary, and Q Hu. Forecasting solar cycle 25 using deep neural networks. *Solar Physics*, 295(5):65, 2020.

[21] Peter Bloomfield. *Fourier analysis of time series: an introduction.* John Wiley & Sons, 2004.

[22] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[23] Lin Chen, Ahmed Elzanaty, Mustafa A. Kishk, Luca Chiaraviglio, and Mohamed-Slim Alouini. Joint uplink and downlink EMF exposure: Performance analysis and design insights. *IEEE Trans. on Wireless Commun*, 22(10):6474–6488, 2023.

[24] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*, 2023.

[25] Luca Chiaraviglio, Angela Sara Cacciapuoti, Gerardo Di Martino, Marco Fiore, Mauro Montesano, Damiano Trucchi, and Nicola Blefari Melazzi. Planning 5G networks under EMF constraints: State of the art and vision. *IEEE Access*, 6:51021–51037, 2018.

[26] Luca Chiaraviglio, Ahmed Elzanaty, and Mohamed-Slim Alouini. Health risks associated with 5G exposure: A view from the communications engineering perspective. *IEEE Open Jrnl. of the Commun. Soc.*, 2:2131–2179, 2021.

[27] Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting. In *IJCAI*, 2022.

[28] Alexander GB Colpitts and Brent R Petersen. Short-term multivariate KPI forecasting in rural fixed wireless LTE networks. *IEEE Networking Letters*, 5(1):11–15, 2023.

[29] Jeff Craley, Christophe Jouny, Emily Johnson, David Hsu, Raheel Ahmed, and Archana Venkataraman. Automated seizure activity tracking and onset zone localization from scalp eeg using deep neural networks. *PloS one*, 17(2):e0264537, 2022.

[30] Anestis Dalgkitsis, Malamati Louta, and George T Karetsos. Traffic forecasting in cellular networks using the LSTM RNN. In *Proc.of the 22nd Pan-Hellenic Conf. on informatics*, pages 28–33, 2018.

[31] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

[32] T Deutsch, ED Lehmann, ER Carson, AV Roudsari, KD Hopkins, and PH Sönksen. Time series analysis and control of blood glucose levels in diabetic patients. *Computer Methods and Programs in Biomedicine*, 41(3-4):167–182, 1994.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[34] Mario Di Mauro, Giovanni Galatro, Fabio Postiglione, Wei Song, and Antonio Liotta. Multivariate time series characterization and forecasting of VoIP traffic

in real mobile networks. *IEEE Trans. on Network and Service Management*, 21(1):851–865, 2023.

[35] Luis Diez, Ramon Aguero, and Joel Penhoat. Reducing the electromagnetic exposure over LTE networks by means of an adaptive retransmission scheme: A use case based on a video service. In *IEEE 81st Veh. Tech. Conf. (VTC Spring)*, pages 1–5, 2015.

[36] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[37] Chunying Fang, Xingyu Li, Meng Na, Wenhao Jiang, Yuankun He, Aowei Wei, Jie Huang, and Ming Zhou. Epilepsy lesion localization method based on brain function network. *Frontiers in Human Neuroscience*, 18:1431153, 2024.

[38] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[39] Isaac Gibbs and Emmanuel Candes. Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34:1660–1672, 2021.

[40] Quentin Gontier, Lucas Petrillo, François Rottenberg, François Horlin, Joe Wiart, Claude Oestges, and Philippe De Doncker. A stochastic geometry approach to EMF exposure modeling. *IEEE Access*, 9:91777–91787, 2021.

[41] Quentin Gontier, Charles Wiame, Shanshan Wang, Marco Di Renzo, Joe Wiart, François Horlin, Christo Tsigros, Claude Oestges, and Philippe De Doncker. Joint metrics for EMF exposure and coverage in real-world homogeneous and inhomogeneous cellular networks. *IEEE Trans. on Wireless Commun*, 23(10):13267–13284, 2024.

[42] Quentin Gontier, Charles Wiame, Joe Wiart, François Horlin, Christo Tsigros, Claude Oestges, and Philippe De Doncker. On the impact of dynamic beamforming on EMF exposure and network coverage: A stochastic geometry perspective, 2024.

[43] Quentin Gontier, Charles Wiame, Joe Wiart, François Horlin, Christo Tsigros, Claude Oestges, and Philippe De Doncker. On the uplink and downlink EMF exposure and coverage in dense cellular networks: A stochastic geometry approach, 2024.

[44] Daniele Grattarola, Lorenzo Livi, Cesare Alippi, Richard Wennberg, and Taufik A Valiante. Seizure localisation with attention-based graph neural networks. *Expert systems with applications*, 203:117330, 2022.

[45] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[46] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[47] Md Arafat Habib, Pedro Enrique Iturria Rivera, Yigit Ozcan, Medhat Elsayed, Majid Bavand, Raimundus Gaigalas, and Melike Erol-Kantarci. Transformer-based wireless traffic prediction and network optimization in O-RAN. In *2024 IEEE Intl. Conf. on Commun. Workshops (ICC Workshops)*, pages 1–6. IEEE, 2024.

[48] Aroosa Hameed, John Violos, Aris Leivadeas, Nina Santi, Rémy Grünblatt, and Nathalie Mitton. Toward QoS prediction based on temporal transformers for IoT applications. *IEEE Trans. on Network and Service Management*, 19(4):4010–4027, 2022.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[50] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[51] Md Tanvir Hossan and Hina Tabassum. Mobility-aware performance in hybrid rf and terahertz wireless networks. *IEEE Transactions on Communications*, 70(2):1376–1390, 2022.

[52] (ICNIRP). Guidelines for limiting exposure to electromagnetic fields (100 kHz to 300 GHz). *Health Physics*, 2020.

[53] International Telecommunication Union. Guidance on complying with limits for human exposure to electromagnetic fields. `https://www.itu.int/rec/T-REC-K.Sup14-201909-I`, 2019. Accessed: 2025-01-02.

[54] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[55] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

[56] Deva Priya Isravel, Salaja Silas, Jaspher W Kathrine, Elijah Blessing Rajsingh, and J Andrew. Multivariate forecasting of network traffic in SDN based ubiquitous healthcare system. *IEEE Open Journal of the Communications Society*, 2024.

[57] Hanyu Jiang, Li You, Ahmed Elzanaty, Jue Wang, Wenjin Wang, Xiqi Gao, and Mohamed-Slim Alouini. Rate-splitting multiple access for uplink massive MIMO with electromagnetic exposure constraints. *IEEE Jrnl. on Sel. areas in Commun.*, 41(5):1383–1397, 2023.

[58] Graham W Johnson, Leon Y Cai, Derek J Doss, Jasmine W Jiang, Aarushi S Negi, Saramati Narasimhan, Danika L Paulo, Hernán FJ González, Shawniqua Williams Roberson, Sarah K Bick, et al. Localizing seizure onset zones in surgical epilepsy with neurostimulation deep learning. *Journal of neurosurgery*, 138(4):1002–1007, 2022.

[59] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021.

[60] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[61] Yiannis Kiouvrekis, Ioannis Givisis, Theodor Panagiotakopoulos, Ioannis Tsilikas, Agapi Ploussi, Ellas Spyratou, and Efstathios P Efstathopoulos. A comparative analysis of explainable artificial intelligence models for electric field strength prediction over eight european cities. *Sensors*, 25(1):53, 2024.

[62] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[63] Rob Kitchin. *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage, 2014.

[64] Petter N Kolm and Nicholas Westray. Improving deep learning of alpha term structures from the order book. *Available at SSRN*, 2024.

[65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[66] Cetin Kurnaz and Mustafa Mutlu. Comprehensive radiofrequency electromagnetic field measurements and assessments: a city center example. *Environmental Monitoring and Assessment*, 192:1–14, 2020.

[67] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[68] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Graphcast: Learning skillful medium-range global weather forecasting, 2023.

[69] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[70] Seunghan Lee, Taeyoung Park, and Kibok Lee. Learning to embed time series patches independently. In *The Twelfth International Conference on Learning Representations*, 2024.

[71] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *ArXiv e-prints*, pages arXiv–1607, 2016.

[72] Adam Li, Chester Huynh, Zachary Fitzgerald, Iahn Cajigas, Damian Brusko, Jonathan Jagid, Angel O Claudio, Andres M Kanner, Jennifer Hopp, Stephanie Chen, et al. Neural fragility as an eeg marker of the seizure onset zone. *Nature neuroscience*, 24(10):1465–1474, 2021.

[73] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[74] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022.

[75] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024.

[76] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, 2022.

[77] Wolfgang Löscher, Heidrun Potschka, Sanjay M Sisodiya, and Annamaria Vezzani. Drug resistance in epilepsy: clinical impact, potential mechanisms, and new innovative treatment options. *Pharmacological reviews*, 72(3):606–638, 2020.

[78] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[79] Donghao Luo and Xue Wang. Moderntcn: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024.

[80] Michel Matalatala, Margot Deruyck, Sergei Shikhantsov, Emmeric Tanghe, David Plets, Sotirios Goudos, Kostas E Psannis, Luc Martens, and Wout Joseph. Multi-objective optimization of massive MIMO 5G wireless networks towards power consumption, uplink and downlink exposure. *Applied Sciences*, 9(22):4974, 2019.

[81] Michel Matalatala, Margot Deruyck, Emmeric Tanghe, Sotirios Goudos, Luc Martens, and Wout Joseph. Joint optimization towards power consumption and electromagnetic exposure for massive MIMO 5G networks. In *IEEE 29th Annual Intl. Symposium on Personal, Indoor and Mobile Radio Commun. (PIMRC)*, pages 1208–1214, 2018.

[82] Michel Matalatala, Margot Deruyck, Emmeric Tanghe, Luc Martens, and Wout Joseph. Optimal low-power design of a multicell multiuser massive MIMO system at 3.7 GHz for 5G wireless networks. *Wireless Commun. and Mobile Computing*, 2018(1):9796784, 2018.

[83] John A. Miller, Mohammed Aldosari, Farah Saeed, Nasid Habib Barna, Subas Rana, I. Budak Arpinar, and Ninghao Liu. A survey of deep learning and foundation models for time series forecasting, 2024.

[84] Navid Mohammadi Foumani, Lynn Miller, Chang Wei Tan, Geoffrey I Webb, Germain Forestier, and Mahsa Salehi. Deep learning for time series classification and extrinsic regression: A current survey. *ACM Computing Surveys*, 56(9):1–45, 2024.

[85] Xavier Mootoo, Hina Tabassum, and Luca Chiaraviglio. Emforecaster: A deep learning framework for time series forecasting in wireless networks with

distribution-free uncertainty quantification. *IEEE Transactions on Network Science and Engineering*, pages 1–19, 2025.

[86] Nor Aishah Muhammad, Norhudah Seman, Nur Ilyana Anwar Apandi, Chua Tien Han, Yonghui Li, and Olakunle Elijah. Stochastic geometry analysis of electromagnetic field exposure in coexisting sub-6 GHz and millimeter wave networks. *IEEE Access*, 9:112780–112791, 2021.

[87] Chi Nguyen, Adnan Ahmad Cheema, Cetin Kurnaz, Ardavan Rahimian, Conor Brennan, and Trung Q Duong. Deep learning models for time-series forecasting of rf-emf in wireless networks. *IEEE Open Jrnl. of the Commun. Soc.*, 2024.

[88] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[89] Ali Yadavar Nikravesh, Samuel A Ajila, Chung-Horng Lung, and Wayne Ding. Mobile network traffic prediction using MLP, MLPWD, and SVM. In *IEEE Intl. congress on big data (BigData Congress)*, pages 402–409, 2016.

[90] World Health Organization et al. *Epilepsy: a public health imperative*. World Health Organization, 2019.

[91] Edward J Oughton, Konstantinos Katsaros, Fariborz Entezami, Dritan Kaleshi, and Jon Crowcroft. An open-source techno-economic assessment framework for 5G deployment. *IEEE Access*, 7:155930–155940, 2019.

[92] Zeydin Pala. Examining emf time series using prediction algorithms with R. *IEEE Canadian Jrnl. of Electrical and Computer Engineering*, 44(2):223–227, 2021.

[93] Harris Papadopoulos. Inductive conformal prediction: Theory and application to neural networks. In *Tools in artificial intelligence*. Citeseer, 2008.

[94] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[95] Joël Penhoat, Ramón Agüero, Fabien Heliot, and Milos Tesanovic. Enabling low electromagnetic exposure multimedia sessions on an LTE network with an IP multimedia subsystem control plane. In *Mobile Networks and Management: 6th Intl. Conf., MONAMI*, pages 207–216. Springer, 2015.

[96] Pablo Fernandez Perez, Claudio Fiandrino, Marco Fiore, and Joerg Widmer. Dissecting advanced time series forecasting models with aichronolens. In *IEEE INFOCOM 2024-IEEE Conf. on Computer Commun. Workshops (INFOCOM WKSHPS)*, pages 01–02. IEEE, 2024.

[97] Yujie Qin, Mustafa A. Kishk, Ahmed Elzanaty, Luca Chiaraviglio, and Mohamed-Slim Alouini. Unveiling passive and active EMF exposure in large-scale cellular networks. *IEEE Open Jrnl. of the Commun Society*, 5:2991–3006, 2024.

[98] Pranav Rajpurkar, Emma Chen, Oishi Banerjee, and Eric J Topol. Ai in health and medicine. *Nature medicine*, 28(1):31–38, 2022.

[99] Mohan Raparthy and Babu Dodda. Predictive maintenance in iot devices using time series analysis and deep learning. *Dandao Xuebao/Journal of Ballistics*, 35:01–10.

[100] Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized quantile regression. *Advances in neural information processing systems*, 32, 2019.

[101] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986. *Biometrika*, 71(599-607):6, 1986.

[102] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[103] Mohammad Amin Saeidi, Hina Tabassum, and Mehrazin Alizadeh. Molecular absorption-aware user assignment, spectrum, and power allocation in dense thz networks with multi-connectivity. *IEEE Trans. on Wireless Commun.*, pages 1–1, 2024.

[104] Mohammad Amin Saeidi, Hina Tabassum, and Mohamed-Slim Alouini. Multi-band wireless networks: Architectures, challenges, and comparative analysis. *IEEE Communications Magazine*, 2023.

[105] Said E Said and David A Dickey. Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 71(3):599–607, 1984.

[106] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

[107] Yusuf A Sambo, Fabien Héliot, and Muhammad Ali Imran. A user scheduling scheme for reducing electromagnetic emission in the uplink of mobile communication systems. In *IEEE Online Conf. on Green Commun. (OnlineGreenComm)*, pages 1–5, 2014.

[108] Yusuf Abdulrahman Sambo, Mohammed Al-Imari, Fabien Héliot, and Muhammad Ali Imran. Electromagnetic emission-aware schedulers for the uplink of OFDM wireless communication systems. *IEEE Trans. on Veh. Tech.*, 66(2):1313–1323, 2016.

[109] Yusuf Abdulrahman Sambo, Fabien Heliot, and Muhammad Ali Imran. Electromagnetic emission-aware scheduling for the uplink of multicell OFDM wireless systems. *IEEE Trans. on Veh. Tech.*, 66(9):8212–8222, 2017.

[110] Domenico Santoro, Tiziana Ciano, and Massimiliano Ferrara. A comparison between machine and deep learning models on high stationarity data. *Scientific Reports*, 14(1):19409, 2024.

[111] Mervyn J Silvapulle and Pranab Kumar Sen. *Constrained statistical inference: Order, inequality, and shape constraints*. John Wiley & Sons, 2011.

[112] Su P Sone, Janne J Lehtomäki, and Zaheer Khan. Wireless traffic usage forecasting using real enterprise network data: Analysis and methods. *IEEE Open Journal of the Communications Society*, 1:777–797, 2020.

[113] Carl E Stafstrom and Lionel Carmant. Seizures and epilepsy: an overview for neuroscientists. *Cold Spring Harbor perspectives in medicine*, 5(6):a022426, 2015.

[114] Rianne Stam. Comparison of international policies on electromagnetic fields:(power frequency and radiofrequency fields). 2018.

[115] Kamile Stankeviciute, Ahmed M Alaa, and Mihaela van der Schaar. Conformal time-series forecasting. *Advances in neural information processing systems*, 34:6216–6228, 2021.

[116] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[117] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

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

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

[120] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[121] Vladimir Vovk and Ivan Petej. Venn-abers predictors. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pages 829–838, 2014.

[122] Volodya Vovk, Alexander Gammerman, and Craig Saunders. Machine-learning applications of algorithmic randomness. *PMLR*, 1999.

[123] Dominik Walther, Johannes Viehweg, Jens Haueisen, and Patrick Mäder. A systematic comparison of deep learning methods for eeg time series analysis. *Frontiers in Neuroinformatics*, 17:1067095, 2023.

[124] Minshen Wang, Li Lin, Ji Chen, David Jackson, Wolfgang Kainz, Yihong Qi, and Perry Jarmuszewski. Evaluation and optimization of the specific absorption rate for multi-antenna systems. *IEEE Trans. on electromagnetic compatibility*, 53(3):628–637, 2011.

[125] Charles Wiame, Simon Demey, Luc Vandendorpe, Philippe De Doncker, and Claude Oestges. Joint data rate and EMF exposure analysis in manhattan environments: Stochastic geometry and ray tracing approaches. *IEEE Trans. on Veh. Tech.*, 73(1):894–908, 2024.

[126] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

[127] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

[128] Chen Xu and Yao Xie. Conformal prediction interval for dynamic time-series. In *Intl. Conf. on Machine Learning*, pages 11559–11569. PMLR, 2021.

[129] Bowen Yang, Baotian Zhao, Chao Li, Jiajie Mo, Zhihao Guo, Zilin Li, Yuan Yao, Xiuliang Fan, Du Cai, Lin Sang, et al. Localizing seizure onset zone by a cortico-cortical evoked potentials-based machine learning approach in focal epilepsy. *Clinical Neurophysiology*, 158:103–113, 2024.

[130] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1317–1322. Ieee, 2016.

[131] Dawei Ying, David J Love, and Bertrand M Hochwald. Beamformer optimization with a constraint on user electromagnetic radiation exposure. In *47th Annual Conf. on Information Sciences and Sys.(CISS)*, pages 1–6, 2013.

[132] Margaux Zaffran, Olivier Féron, Yannig Goude, Julie Josse, and Aymeric Dieuleveut. Adaptive conformal predictions for time series. In *International Conference on Machine Learning*, pages 25834–25866. PMLR, 2022.

[133] Sheyda Zarandi and Hina Tabassum. Federated double deep q-learning for joint delay and energy minimization in iot networks. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2021.

[134] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? 2023.

[135] Yunyang Zhang, Yulun Wu, Aijun Liu, Xiaochen Xia, Ting Pan, and Xian Liu. Deep learning-based channel prediction for leo satellite massive mimo communication system. *IEEE Wireless Commun. Letters*, 10(8):1835–1839, 2021.

[136] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International conference on web-age information management*, pages 298–310. Springer, 2014.

[137] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[138] Tao Zhou, Haitong Zhang, Bo Ai, Chen Xue, and Liu Liu. Deep-learning-based spatial–temporal channel prediction for smart high-speed railway communication networks. *IEEE Transactions on Wireless Communications*, 21(7):5333–5345, 2022.

[139] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series

forecasting. In *Proc. 39th International Conference on Machine Learning (ICML 2022)*, 2022.