This is the author's accepted version of the article. The final version published by IEEE is Pablo Fernández Pérez, Claudio Fiandrino, Eloy Pérez Gómez, Hossein Mohammadalizadeh, Marco Fiore and Joerg Widmer, "AIChronoLens: AI/ML Explainability for Time Series AI Forecasting in Mobile Networks," in IEEE Transactions on Mobile Computing, 2025, pp. TBD, doi: TBD.

AIChronoLens: AI/ML Explainability for Time Series Forecasting in Mobile Networks

Pablo Fernández Pérez, *Student Member, IEEE,* Claudio Fiandrino, *Member, IEEE,* Eloy Pérez Gómez, Hossein Mohammadalizadeh, Marco Fiore *Senior Member, IEEE,* and Joerg Widmer, *Fellow, IEEE*

Abstract—Forecasting is increasingly considered a fundamental enabler for the management of next-generation mobile networks. While deep neural networks excel at short- and long-term forecasting, their complexity hinders interpretability, a crucial factor for production deployment. The existing EXplainable Artificial Intelligence (XAI) techniques, primarily designed for computer vision and natural language processing, struggle with time series data due to their lack of understanding of temporal characteristics of the input data. In this paper, we take the research on XAI for time series forecasting one step further by proposing AIChronoLens, a new tool that links legacy XAI explanations with the temporal properties of the input. AIChronoLens allows diving deep into the behavior of time series predictors and spotting, among other aspects, the hidden causes of forecast errors. We show that AIChronoLens's output can be utilized for meta-learning to predict when the original time series forecasting model makes errors and fix them in advance, thereby improving the accuracy of the predictors. Extensive evaluations with real-world mobile traffic traces pinpoint model behaviors that would not be possible to identify otherwise and show how model performance can be improved by $32\,\%$ upon re-training and by up to $39\,\%$ with meta-learning.

Index Terms—Explainable AI, mobile networks, deep learning, time series forecasting, network performance indicators.

1 Introduction

The advent of fifth-generation (5G) mobile networks has considerably changed the landscape of the mobile network ecosystem. The growing availability of higher and faster access to mobile services has contributed to an increase in the demand for mobile traffic which is growing at a staggering pace. Consuming data via 5G is expected to account for 75% of the total mobile traffic in 2029 whereas the same fraction was only 25% at the end of 2023 [1].

The capability to analyze and forecast mobile traffic volumes at the individual level of single Base Station (BS) or the city scale has become key for operators to perform resource management properly. Traffic forecasting makes diverse optimizations possible, such as network deployment planning [2], routing [3], and mobility management [4], resource allocation [5] and network slicing [6], and to reduce the energy consumption footprint [7]. In the context of individual BSs, forecasting traffic volumes has found applicability in anomaly event detection [8], scalable scheduling of pilot signals to improve the quality of channel estimation [9], uplink single-user throughput [10], grant scheduling [11] or buffer status reports [12], and to infer Physical Resource Block (PRB) utilization [13].

Time series forecasting has received significant attention from the Artificial Intelligence (AI) community. Recent state-of-the-art performers like DLinear [14], and PatchTST [15] improved model accuracy significantly over well-known techniques like Long-Short Term Memory (LSTM) or AutoRegressive Integrated Moving Average (ARIMA) in a variety of datasets. In the context of mobile networks, LSTM is still by far the most popular technique for univariate next-step time series forecasting [8], [9], [10], often outperforming other methods like ARIMA [16]. However, transformer-based models like PatchTST learn better long-range dependencies than LSTM which are limited to capturing dependencies in fixed windows of time. Furthermore, the attention mechanisms employed by such models shall provide a competitive advantage over LSTM or ARIMA when the data is noisy or where there are multiple patterns in the data.

The logic governing the above-mentioned models like DLinear, PatchTST, and LSTM is not easily understandable by a human observer, which creates an inherent lack of explainability of the models and hampers their use in production networks. Without a proper understanding of the logic governing such models, network managers are understandably reluctant to blindly trust the output of data-driven models. Moreover, network engineers remain skeptical of the opaque internal operation of these models that make tasks like troubleshooting daunting and that create new surfaces for adversarial attacks [17]: indeed, it has been shown how perturbations to the original input (*e.g.*, added load or jamming) can be crafted to be imperceptible to humans, but sufficient to worsen the accuracy of a predictor [18], [19]. These examples show how explainability is mandatory if those models are to be deployed in production-grade networks. As an interesting counterexample, decision trees [20] have been used in practical scenarios by AT&T for automatic parameter configuration of newly deployed BSs [21]. As explained by the authors of that study, a key element that allowed those models to gain the trust of the operator was their inherent interpretability. Unfortunately, decision trees operate on discrete output variables and are thus very cumbersome to use for mobile traffic forecasting.

In this context, the fundamental objective of EXplainable Artificial Intelligence (XAI) is precisely to provide logical and human-understandable explanations for the black-box behavior of neural networks like LSTMs. Historically, XAI techniques have been conceived and tailored for computer

• *All the authors are with IMDEA Networks Institute, Madrid, Spain. Email: {name.surname}@imdea.org*

Manuscript received MONTH DAY, YEAR; revised MONTH DAY, YEAR.

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

vision and Natural Language Processing (NLP), and not for time series [22]. This is mainly attributed to data characteristics (high-dimensional data like images and video are more intuitive to explain than time series for which pattern identification is more complex) and the surge of interest for computer vision-based applications (medical imagining or security built on object detection and recognition are very popular which has drawn a lot of attention to their interpretation; in contrast, mobile traffic forecasting is not as popular). Prominent XAI techniques like LayeR-wise backPropagation (LRP) [23], SHapely Additive exPlanations (SHAP) [24], Local Interpretable Model-agnostic Explanations (LIME) [25], DeepLIFT [26] have been adapted for time series or are also applicable to time series like DeepAID [27]. However, as we show in §2, they fail to provide useful explanations from a fundamental perspective that goes beyond the simple understanding of input relevance. For example, they are not capable of revealing the hidden causes of model errors that are specific to both *(i)* the model's inner logic, and *(ii)* the currently observed input.

In this paper, we tackle precisely the problem of enhancing the quality of explanations in the context of time series forecasting for mobile networks. Our far-reaching objective is to lower the barrier to the adoption of AI time series forecasting models in production networks. For this, we propose and design AICHRONOLENS, a new tool that addresses the main shortcomings of legacy XAI techniques and provides means to better comprehend LSTM models in action. In essence, AICHRONOLENS resolves the ambiguity of legacy XAI techniques in assigning the same relevance scores to highly diverse input sequences by exploring the Pearson correlation between relevance scores and an enriched expressiveness of the input sequence. We do so by applying an imaging technique called Gramian Angular Field (GAF) [28] that turns an input time series sequence into a 2D representation, making it possible to capture pairwise relationships like local maxima/minima within the input sequence and their spatial distance. Positive or negative correlations between relevance scores and the GAF imply that higher or lower importance is given to relevant samples like local maxima or minima. AICHRONOLENS exploits such added expressiveness to characterize the model behavior.

In practice, AICHRONOLENS can be used either offline or online. The offline use is model inspection: AICHRONOLENS synthesizes tailored explanations on model behavior that can be next used at inference times for monitoring purposes. The online use is for *meta-learning* when the original model makes mistakes and corrects them. Traditionally, meta-learning in the context of time series forecasting indicates the process of automatic knowledge acquisition for either model selection of model hyperparameter fine-tuning [29], [30], [31]. More recently, the concept extended to the prediction of meta-features that indicate if the accuracy of the model is degrading in the presence of unseen data at training time, and thus the model requires re-training [32]. Our work takes meta-learning model errors one step further because we utilize the outputs of AICHRONOLENS as input of the well-known ResNet [33] model to predict and correct errors of the original time series forecasting model.

We perform an extensive evaluation of the strengths of AICHRONOLENS with real-world mobile traffic data for two relevant use cases, *i.e.*, forecasting of traffic loads in an urban area and of the number of connected users to a BS. For the former, we use a measurement dataset collected in a production 4G network serving a major metropolitan region in Europe with minute-level traffic information. For the latter, we use a measurement dataset collected at production BSs with millisecond-level traffic information [34]. We demonstrate that AICHRONOLENS spots model behaviors that cannot be identified otherwise, and partially correct them with the meta-learning architecture.

The key contributions ("C") and findings ("F") of our study are summarized as follows:

C1. We design AICHRONOLENS, a new tool that addresses the inherent shortcomings of prominent XAI tools when applied to AI models for time series forecasting by harnessing liner relationship between relevance scores of XAI tools and temporal characteristics of the input sequences.

C2. We design a meta-learning architecture consisting of an original time series forecasting model and a ResNet model that processes AICHRONOLENS's outputs to forecast and correct the errors in the original time series forecasting model.

C3. We perform an extensive evaluation of AICHRONOLENS with real-world datasets and several time series forecasting models to demonstrate that it provides detailed explanations of model behavior that are useful to monitor and verify the model's robustness over time.

F1. We find that, unlike legacy XAI tools, AICHRONOLENS is capable of pinpointing differences in hyperparameter settings at training times of different models applied to the same test data. For example, higher learning rates translate into stronger correlations between the relevance scores and the time series inputs while lower learning rates exhibit weak or non-linear correlation.

F2. We find that the correlation coefficients obtained as outcomes of AICHRONOLENS show geometrical properties that can be related to the model errors. Further, we show the root causes of this issue, *e.g.*, poor model design or data that is inherently hard to predict.

F3. We find that AICHRONOLENS can be used to improve model accuracy in several ways: either by refining the training process with informative feedback on missing patterns in the original training data or via meta-learning.

For the sake of reproducibility and to further stimulate the research in the field, we released the artifacts of our initial study [35] (the trained LSTM models and the code of AICHRONOLENS) at: https://git2.networks.imdea.org/wng/aichronolens. Upon acceptance of this version of the manuscript, we will update the repository with PatchTST and DLinear models, as well as the meta-learning architecture for error correction.

2 Background and Motivation

2.1 Background

Time Series Forecasting: Problem Formulation. The objective of Machine Learning (ML) models that tackle the problem of time series traffic forecasting is to predict a sequence of future values at a certain time step t, having observed

a sequence of past values. Values can be traffic volumes, number of users or PRBs measured over time. Formally, let $\mathcal{X}_\mathcal{T} = \{x_1, x_2, \ldots, x_T\}$ be the whole sequence of values at time $t = \{1, 2, \ldots, T\}$. Let $X_t \in \mathcal{X}_\mathcal{T}$ be the set of historical n past values at time t: $X_t = \{x_{t-n+1}, x_{t-n+2}, \ldots, x_t\}$. n is known as *history*, *lookback* or *input sequence*, with $n \ll T$. Then, the forecast $\hat{Y}_t = \{\hat{y}_{t+1}, \hat{y}_{t+2}, ..., \hat{y}_{t+H}\}$ at time t is:

$$\hat{Y}_t = F(X_t), \tag{1}$$

where H is the prediction length or horizon: for single-step predictions $H = 1$ while in multi-step predictions $H > 1$. F is a generic prediction function and the ML model design phase is all about defining a proper F for the problem under analysis. F is trained by evaluating at each iteration a loss function $Z_\theta(Y_t, \hat{Y}_t)$, where $Y_t = \{y_{t+1}, y_{t+2}, ..., y_{t+H}\}$ is the ground truth, and updating the parameters θ (*e.g.*, the weights) to fulfill a specific objective, *e.g.*, minimizing the Mean Absolute Error (MAE) or Mean Square Error (MSE) between Y_t and $\hat{Y}_t$.

A Primer on XAI. Promoting trustworthiness in AI has experienced a surge of interest over the last few years [36], [37], also in the context of mobile networks [38], [39]. While *interpretability* focuses on contextualizing the model outputs about its design, *explainability* goes beyond and provides customized knowledge that describes how and why a model comes to achieve a given output [40]. *Intrinsic* or *transparent* XAI techniques foster interpretability, while *post-hoc* XAI techniques apply after training and concern explainability [41]. AICHRONOLENS synthesizes *post-hoc* explanations.

XAI for Time Series. Although XAI was conceived and tailored for computer vision and NLP, there exists some applicability to time series [22], especially in the context of time series classification [42], [43]. The techniques that apply to forecasting are often tailored to multi-variate time series [44]. Many mobile network problems instead require tools for univariate time series, which we present next.

XAI Techniques. There exists model-agnostic and model-specific techniques. SHAP [24], LIME [25] and Eli5 [45] belong to the first category and provide explanations by perturbing the inputs of the models to determine how relevant the features are for the prediction. These techniques differ in the way they compute the relevance scores. Conversely, LRP [23] is model-specific. LRP provides explanations by evaluating which neurons are relevant to a prediction given the input data, making it thus possible to highlight which part of the input data influences the prediction the most.

We now provide the reader with the necessary background on the XAI techniques used in the rest of the paper:

• *LRP* assigns a score to all the inputs of a predictor and this score indicates the extent of their contribution to the predictor. The relevance scores are computed by tracking back from the output the individual activation of each neuron and its weight in subsequent layers of the model. LRP follows a conservation principle for which the total amount of relevance distributed in layer p remains unaltered in layer q. When the backpropagation reaches the input layer, the relevance score is distributed to the input sequence.

• *SHAP* provides feature-based explanations by approximating the Shapley values of a prediction. These are obtained by examining the effect of removing one feature at a time under the combination of the presence/absence of all other features. SHAP generates global and local explanations in the form of log-odds, which can be turned into a probability distribution with the `softmax` operation.

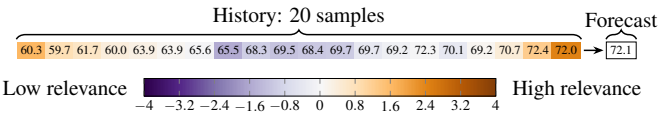


Fig. 1. Example of LRP scores for an input sequence of traffic load

LRP appears to be more suitable for the specific case of time series prediction, as it provides high-quality fidelity in the explanations vis-a-vis the SHAP and LIME counterparts [46]. Nevertheless, for the sake of completeness, we will use both LRP and SHAP.

2.2 Motivation

We now elaborate on the need for AICHRONOLENS by showing the limitations of LRP and SHAP techniques. For the specific problem of time series classification, a recent work [47] shows that different methods lead to different post-hoc explanations for the same model on the same dataset. In this work, we go further and show that when applied to univariate time series the *same* XAI method provides ambiguous explanations with no relation to the input sequence.

For this purpose, we train a model composed of an LSTM layer with 200 neurons followed by a fully connected output layer with a single hidden unit. The model applies to a dataset that contains traffic volumes from a production network with a 3 minute granularity where 28 541 samples are used for training and 7 121 for testing. §4.1 provides more details on the datasets and all the other models trained for validation purposes from which we derived the model currently under consideration. We apply both LRP and SHAP on the test set. Fig. 1 portrays an example of an input sequence of 20 values, the forecast, and the LRP scores. Next, we perform an extensive clustering analysis utilizing Dynamic Time Warping (DTW) Barycenter Averaging (DBA) [48] and Soft-DTW [49] k-means. For each technique, we run DBA and Soft-DTW for several cluster sizes (*i.e.*, $\kappa = [3 : 10]$) and compute the silhouette score to identify the optimal number of clusters [50]. Fig. 2 portrays an example of the obtained results for LRP where the optimal number of clusters is $\kappa = 4$ for both DBA and Soft-DTW. It takes nearly 16 hours to execute on an Intel® Core™ i9-11900K processor operating at 3.5 GHz and equipped with a Nvidia RTX 3090 GPU. On the top of the figure, we show the LRP scores and, on the bottom, the corresponding input sequence that produced a prediction explained by the generated scores. From Fig. 2 we observe that, for each cluster, there is no unique relationship that bonds the LRP scores with input sequences. We verify that the same behavior holds for SHAP too. The lack of such a relationship suggests that the XAI techniques are either not effectively capturing the salient characteristic of the model or that the model itself is not adequate for the job.

3 AICHRONOLENS

In light of the motivation presented in §2.2, this section presents AICHRONOLENS, a new technique that enhances the depth of explanations of legacy explainability tools. We first delve into its design principles (§3.1), next we

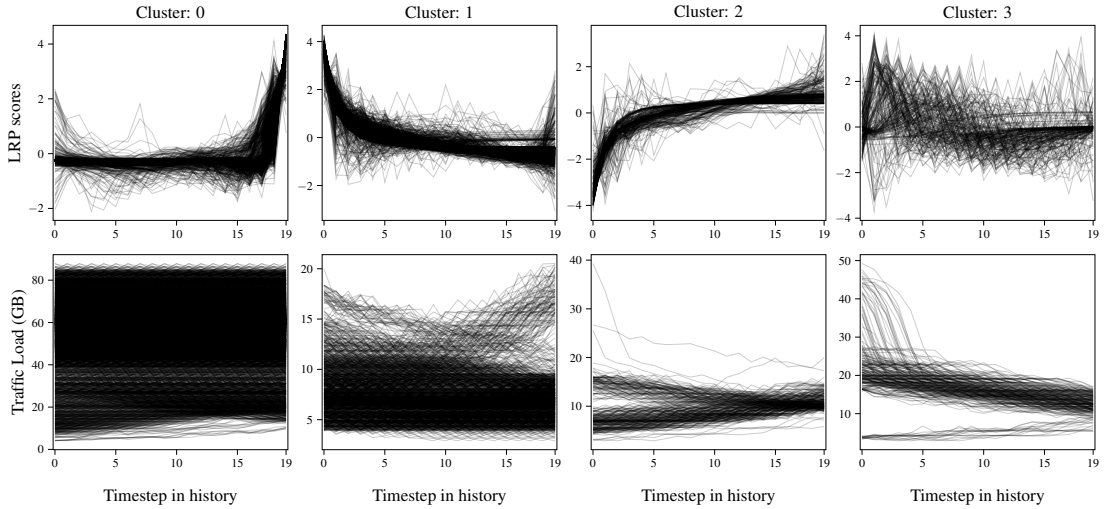


Fig. 2. Main shortcoming of prominent XAI methods: explanation profiles can originate from highly diverse input sequences

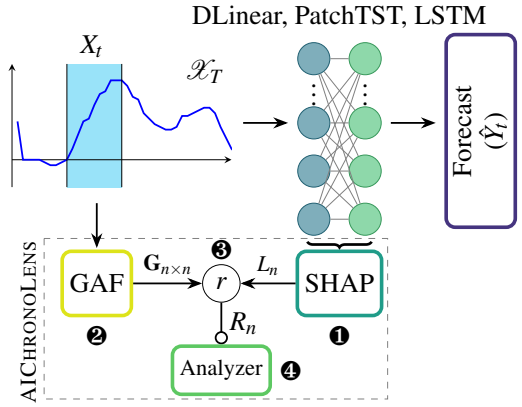


Fig. 3. AICHRONOLENS architecture

present its architectural design (§3.2), its computational complexity (§3.3), and, finally, we discuss the architecture for meta-learning (§3.4).

3.1 Overview and Design Principles

Fig. 3 outlines the high-level design of AICHRONOLENS. In a nutshell, AICHRONOLENS extracts through XAI techniques like SHAP or LRP relevance scores (L_n) that define the contribution of each element of the input sequence X_t to one element of the forecast $\hat{Y}_t$ (module ❶ in the figure). To resolve the ambiguity highlighted in §2.2, AICHRONOLENS uses an imaging technique, the GAF, on X_t to reveal patterns within the input sequence (module ❷). The GAF encodes pairwise angular relationships by representing each point of X_t as a cosine-based angular coordinate. The operation transforms X_t into a more expressive representation that we harness to generate explanations. Next, AICHRONOLENS probes for linear correlation with the Pearson's coefficient between the relevance scores L_n and the GAF representation $G_{n\times n}$ (module ❸). We specifically probe for linear correlation to understand whether the model provides higher or lower importance to relevant samples in the input sequence like local maxima or minima. Finally, the "Analyzer" module monitors when this relation holds true (alignment between relevance scores and input sequences as series of correlation vectors R_n computed over time) or not and exploits transitions between the two cases as base information to synthesize more profound explanations (module ❹).

We design AICHRONOLENS with the following *design principles* (*DP*) in mind:

• DP_1: *XAI Generality.* We allow for any of the existing XAI tools to be plugged into AICHRONOLENS, which makes AICHRONOLENS highly general. At the same time, this allows us to compare the explanations that the different XAI tools provide when applied to the same trained LSTM model on the same dataset.

• DP_2: *Univariate Time Series Specificity.* While being general regarding the pluggable XAI techniques and AI models for time series forecasting, in this paper we restrain the scope of action to univariate time series tasks. We leave to future work the adaptation of AICHRONOLENS for models dealing with either multi-variate or spatio-temporal inputs.

3.2 AICHRONOLENS Design

As introduced above, AICHRONOLENS consists of four main modules: module ❶ exploits an XAI technique for each prediction, module ❷ applies the GAF operation, module ❸ computes the Pearson's correlation coefficient on both relevance scores and GAF matrix, and module ❹ synthesizes explanations from the obtained correlation coefficients.

Relevance Scores from XAI (❶). In computer vision, XAI techniques indicate the relevance of each pixel of an image at each point in time t. In analogy, by taking into account that $\hat{Y}_t$ depends on the past, or input sequence X_t, then XAI techniques provide relevance scores L_n to each element of the input sequence $x_i \in X_t, \forall i = 1, 2, \ldots, n$.

According to DP_1, we now show how to obtain the relevance scores L_n by considering the two most prominent XAI techniques for each category of methods, *i.e.*, LRP (backpropagation) and SHAP (perturbation).

LRP computes relevance scores $L^q_{i\leftarrow j}$ that represent the relevance score propagated from neuron j in layer p back to neuron i in layer q:

$$L^q_{i\leftarrow j} = L^p_j \cdot \frac{a_i w_{i,j}}{\sum_k a_k w_{k,j}}, \tag{2}$$

where the term L^p_j indicates the relevance score at neuron j in layer p, $w_{i,j}$ is the weight connecting neuron i in layer

q to neuron j in layer p, while the index k refers to the set of neurons contributing to the activation of neuron j. The process iterates from the prediction output $\hat{Y}_t$ to the input X_t where the relevance scores L_n are attributed to the n elements of X_t. Note that the process is independent of the number of neurons of the model, *i.e.*, regardless of the depth of the neural network LRP always produces a total of n relevance scores.

• SHAP is an XAI framework based on Shapley values from game theory. Shapley values measure the contribution (positive or negative) of each feature in the input to predict the output. Note that a negative contribution means a negative deviation from the expected value of the model, *i.e.*, the empirical mean of the predictions over a specific set of observations. Formally, the definition of a Shapley value is the following: Given a set of players (features if extrapolated to ML problems) $M = \{1, 2, \ldots, m\}$ and a characteristic function (or ML model) $v : 2^M \to \mathbb{R}$, the Shapley value $L_i(v)$ for player i is defined as:

$$L_i(v) = \sum_{S \subseteq M \setminus \{i\}} \frac{|S|!(m - |S| - 1)!}{m!} \left(v(S \cup \{i\}) - v(S)\right)$$

where $S \subseteq M \setminus \{i\}$ denotes any subsets of M that do not contain player i, $|S|$ is the number of players in subset S, m is the total number of players, $v(S)$ is the value of subset S.

Imaging via GAF (❷). Given the inherent ability of ML in dealing with images, there exists several attempts of transforming time series into images [51]. Recurrence Plots (RP), GAF, and Markov Transition Field (MTF) are popular imaging techniques for time series [28]. In a nutshell, all of them turn a time series of length m into an image of $m \times m$ pixels. The difference between these techniques lies in how they define the image. RPs compute the Euclidean distance for each value $j \in m$ of the time series. RPs are not capable of dealing with time series of variable length and different scales, and can not effectively represent upward and downward trends [52]. GAF represents a time series using polar rather than Cartesian coordinates by constructing a Gram matrix where each element is the cosine of the sum of 2 angles. Finally, MTF constructs a Markov matrix of quantile bins on the time series values and encodes into a quasi-Gramian matrix the dynamic transition probability of each element $j \in m$. Although the MTF technique preserves temporal dependencies like GAF, it does not allow reconstructing the original time series as GAF does. The difference lies in the fact that GAF operates on time series values directly, while MTF operates on transition probabilities of quantiles. Hence, we use GAF for AICHRONOLENS.

To obtain a GAF, the original elements of time series $x_i \in X_t$, with $i = 1, \ldots, n$, undergo a set of transformations. First, we rescale them in the range $[-1, 1]$:

$$\widetilde{x}_i = \frac{(x_i - \max(X_t)) + (x_i - \min(X_t))}{(\max(X_t) - \min(X_t))}. \tag{3}$$

Next, we represent $\widetilde{X}_n$ in polar coordinates by encoding the value as the angular cosine and the time step as the radius:

$$\begin{cases} \phi_i = \arccos(\widetilde{x}_i), -1 \leq \widetilde{x}_i \leq 1, \widetilde{x}_i \in \widetilde{X}; \\ r_i = \dfrac{i}{n}, i \in \mathbb{N}. \end{cases} \tag{4}$$

In this equation, r maps the progression over time of the input sequence as radial distance in concentric circles and the lookback size n is used for normalization so that the progression of time and, in turn, the size of the concentric circles, is proportional to the length of the series. The variables ϕ and r in the context of Gramian Angular Fields (GAF) representations do not have subscript i because they describe a global transformation of the entire input sequence X_t rather than individual components indexed by i. In other words, they represent the result of a transformation applied to each $\widetilde{x}_i$ and i and depend on the entire X_t because of the rescaling operation.

With time increase, the values of the time series shift between angular positions, while the radius increases at a steady rate. The polar representation for the visualization of the time series brings two important properties. First, it is bijective, as $\cos(\phi)$ is monotonic when $\phi \in [0, \pi]$ which makes it possible to recover the original time series. Second, it preserves absolute temporal relations as opposed to Cartesian coordinates. In Cartesian coordinates, the area under a curve between two points depends only on the time interval between the two points. However, in polar coordinates, the area also depends on the absolute values of the time series at those points. This means that even if two-time intervals have the same length, the corresponding areas in polar coordinates can be different if the time series values at the endpoints are different. Armed with such representation, we can define the GAF as $\mathbf{G}_{n \times n}$ for each $t \in T$:

$$\mathbf{G}_{n \times n} = \begin{bmatrix} \cos(\phi_1 + \phi_1) & \cdots & \cos(\phi_1 + \phi_n) \\ \cos(\phi_2 + \phi_1) & \cdots & \cos(\phi_2 + \phi_n) \\ \vdots & \ddots & \vdots \\ \cos(\phi_n + \phi_1) & \cdots & \cos(\phi_n + \phi_n) \end{bmatrix}. \tag{5}$$

By defining the inner product as follows:

$$\langle v, z \rangle = v \cdot z - \sqrt{1 - v^2} \cdot \sqrt{1 - z^2}. \tag{6}$$

a Gram matrix [53] $\mathbf{G}_{n \times n}$ can be rewritten as:

$$\mathbf{G}_{n \times n} = \begin{bmatrix} \langle \widetilde{x}_1, \widetilde{x}_1 \rangle & \cdots & \langle \widetilde{x}_1, \widetilde{x}_n \rangle \\ \langle \widetilde{x}_2, \widetilde{x}_1 \rangle & \cdots & \langle \widetilde{x}_2, \widetilde{x}_n \rangle \\ \vdots & \ddots & \vdots \\ \langle \widetilde{x}_n, \widetilde{x}_1 \rangle & \cdots & \langle \widetilde{x}_n, \widetilde{x}_n \rangle \end{bmatrix}. \tag{7}$$

A Gram matrix of a set of vectors $\boldsymbol{v_1}, ..., \boldsymbol{v_n}$ in an inner product space is the Hermitian matrix of the inner product (a matrix $\boldsymbol{B}$ is Hermitian if and only if each element fulfills $b_{ij} = \overline{b_{ji}}$, that is, the matrix is equal to its conjugate transpose). The GAF representation provides several features. First, it preserves temporal dependency, because the time increases as we move from top left to bottom right. It contains temporal correlations, since $\mathbf{G}_{(i,j||i-j|=t)}$ corresponds to the relative correlations of the directions that lie in the time step t. The main diagonal of $\mathbf{G}_{n \times n}$ is a special case containing the original time series values. In a GAF, high values (close to 1) are those where local maxima or minima in the original time series correlate either with themselves or other maxima or minima respectively. The values close to 0 are the result of a correlation between local maxima or minima with points of intermediate values in the original time series. Finally, negative values (close to -1) originate from the correlation

between a point with a local maxima or minima with another point in the original time series with a local minima or maxima respectively.

Defining Correlations (❸). Armed with relevance scores L_n and $\mathbf{G}_{n\times n}$, we seek correlation between these two quantities. In essence, we aim to understand if there is a linear relation between the relevance scores (*i.e.*, L_n) and elements of the input time series (*i.e.*, $\mathbf{G}_{n\times n}$). Specifically, by construction, each row of $\mathbf{G}_{n\times n}$ characterizes inner relationships between samples of the input time series. Denote the ith row of this matrix as G_i, a $1\times n$ vector defined in (7). Given that also L_n is a $1\times n$ vector, we can compute the Pearson's correlation coefficient between these two quantities. Specifically:

$$R_n = \frac{\text{cov}(G, L)}{\sigma_G \sigma_L} = \begin{bmatrix} \rho_0 \\ \rho_1 \\ \vdots \\ \rho_n \end{bmatrix}. \tag{8}$$

where σ is the standard deviation and $\text{cov}(\cdot,\cdot)$ is the covariance. By computing (8) for each time step $t = 1, \ldots, T$, we obtain a correlation matrix $\mathbf{C}$ with dimensions $n \times T$:

$$\mathbf{C} = \begin{bmatrix} \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,T} \\ \rho_{2,1} & \rho_{2,2} & \cdots & \rho_{2,T} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{n,1} & \rho_{n,2} & \cdots & \rho_{n,T} \end{bmatrix}_{n\times T}, \tag{9}$$

which can be written in compact form as

$$\mathbf{C} = \begin{bmatrix} R_{n,1} & R_{n,2} & \ldots & R_{n,T} \end{bmatrix}. \tag{10}$$

Analyzing Correlations (❹). At the heart of AICHRONOLENS, the "Analyzer" module exploits $\mathbf{C}$ to synthesize explanations. For the sake of illustration, let us consider the case of observing $W = 3$ time steps (*i.e.*, t, $t+1$, and $t+2$) of correlation vectors with a history of $n = 6$ samples. With the increase in time, a correlation coefficient ages and vanishes once the presence in the history of the sample of the time series that contributed to its generation is over. To observe the evolution of the Pearson's coefficients of each sample over time, we create a new matrix $\mathbf{S}$ by storing all the secondary diagonals of length T in rows:

$$\mathbf{C}_{6\times 3} = \begin{bmatrix} \rho_{1,1} & \rho_{1,2} & \rho_{1,3} \\ \rho_{2,1} & \rho_{2,2} & \rho_{2,3} \\ \rho_{3,1} & \rho_{3,2} & \rho_{3,3} \\ \rho_{4,1} & \rho_{4,2} & \rho_{4,3} \\ \rho_{5,1} & \rho_{5,2} & \rho_{5,3} \\ \rho_{6,1} & \rho_{6,2} & \rho_{6,3} \end{bmatrix}, \mathbf{S}_{4\times 3} = \begin{bmatrix} \rho_{3,1} & \rho_{2,2} & \rho_{1,3} \\ \rho_{4,1} & \rho_{3,2} & \rho_{2,3} \\ \rho_{5,1} & \rho_{4,2} & \rho_{3,3} \\ \rho_{6,1} & \rho_{5,2} & \rho_{4,3} \end{bmatrix}. \tag{11}$$

$t \quad t+1 \quad t+2$

For practical use, $\mathbf{S}^T$ is more convenient than $\mathbf{S}$.

All the explanations of AICHRONOLENS rely on the analysis of $\mathbf{C}$ or $\mathbf{S}^T$ over windows $w \leq T$. AICHRONOLENS's outputs are directly linked to the temporal consistency between the time series input and the relevance scores of legacy XAI techniques. This consistency materializes in the form of patterns within $\mathbf{C}$ or $\mathbf{S}^T$. Specifically, depending on their value, either positive or negative, the correlation values generate patterns in w. For instance, from left to right, bottom to top: $\rho_{6,1}$, $\rho_{6,2}$, $\rho_{6,3}$, $\rho_{5,2}$, $\rho_{5,3}$, and $\rho_{4,3}$ would form a triangle of negative correlation if all the coefficients are in the range $[-1, 0]$. Our explanations are based on what we call points of interest that arise when these patterns in w break or alter because these define the response of the model to the time evolution of the original time series. The response of the model could be either correct if the model has successfully learned such a pattern or erratic if the model did not learn such a pattern during training.

Figure 4 exemplifies the above concept and shows at a high level the conceptual advantage of AICHRONOLENS over the use of SHAP alone. Specifically, we show the response of the model in the presence of a spike in the original time series and we highlight the spike with a box around time step 16. The top subplot illustrates the forecasts of the predictor and ground truth. Note that while the prediction may be multi-horizon (predicting multiple time steps at once), AICHRONOLENS can only explain one item of the prediction at a time, $\hat{y}_{t+h} \in \hat{Y}_t$ with $h = \{1, 2, ..., H\}$. In this example, we focus on forecasting the immediate next time step, i.e., $h = 1$. The second subplot is a visual representation of the ground truth, *i.e.*, the original time series. For each time step t on the x-axis, the y-axis displays a window X_t containing the normalized (with standard scaler) lookback utilized for the prediction in time step $t+1$. Since the offset between consecutive windows is exactly one value, each new element of the time series will enter a window X_t in position 0 and will shift by one value in consecutive windows reaching the position 300. An oracle would learn exactly these patterns. The third subplot shows the SHAP values that measure the contribution of each value in X_t for predicting the output $\hat{y}_{t+h}$. The SHAP values mimic the patterns visible in the ground truth in an ambiguous manner. Finally, the bottom subplot shows AICHRONOLENS's outputs and it becomes clear how AICHRONOLENS preserves the time series patterns of the ground truth much better than SHAP. These patterns directly relate to the temporal consistency of the model response to the evolution of the time series. When this temporal consistency breaks, as seen with the spike in the figure, it is recorded in AICHRONOLENS's outputs, allowing the detection of these points of interest. These points of interest are sensitive and are points where the model might make errors if the training process is not sufficiently accurate.

To summarize, AICHRONOLENS links relevance scores with temporal characteristics of the input sequences in a unique manner. The output of the tool is correlation coefficients that, if observed over time, generate patterns (series of triangles of positive or negative values) that can be geometrically interpreted to spot different causes of errors. In §5 we will show two techniques for pattern recognition that uniquely identify different causes of errors.

3.3 Computational Complexity

The computational complexity of AICHRONOLENS has three components, one for each of the main modules: the computation of the relevance scores with legacy XAI techniques like SHAP and LRP (module ❶), the transformation of X_t into the GAF (module ❷), and the correlation analysis (module ❸).

Relevance Scores. The complexity of generating relevance scores depends on the explainability method applied:

- The computational complexity of SHAP is typically $O(n \cdot 2^n)$ due to the combinatorial nature of Shapley values. Natively, SHAP is computationally very expensive.

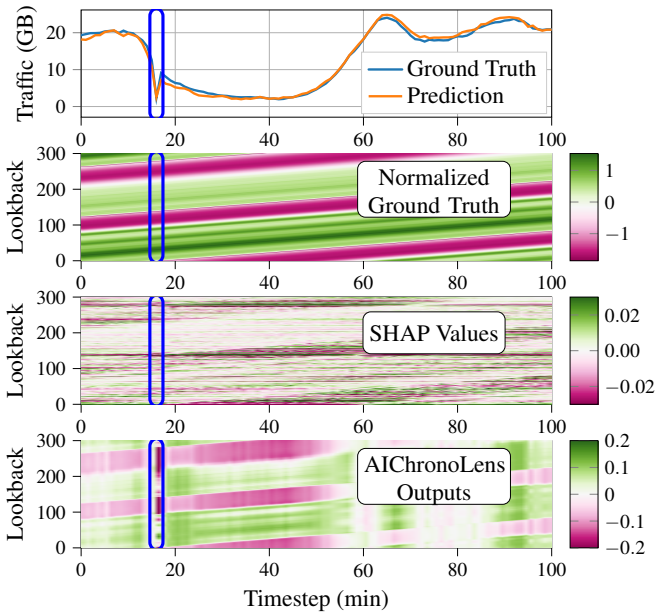


Fig. 4. Visual comparison of the analysis of AICHRONOLENS and SHAP

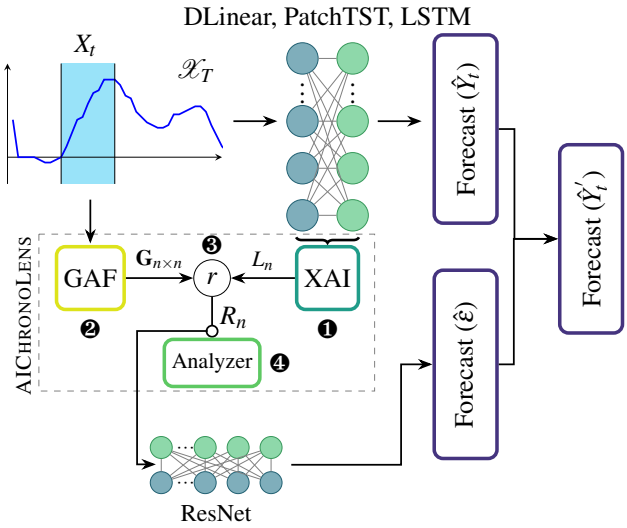


Fig. 5. AICHRONOLENS architecture for meta-learning: the objective is to predict the error of the original model and fix it

For deep neural networks like those used in this work, DeepSHAP was developed [54] whose computational complexity is $O(n \cdot k \cdot a)$, where k is the number of neurons in the neural network and a is the number of samples used for the approximation.

- The computational complexity of LRP is $O(n \cdot k)$, where k is the number of neurons in the neural network.

GAF Transformation. Transforming the input sequence X_t of length n into a GAF matrix involves pairwise cosine correlations between all elements. Thus the computational complexity is $O(n^2)$.

Correlation Analysis. Calculating the Pearson correlation between the $\mathbb{G}$ and the relevance scores L involves element-wise operations over the matrix and the relevance vector, leading to a complexity of $O(n^2)$.

Considering all the above components together, the computational complexity of AICHRONOLENS is $O(n^2+n\cdot d)$ using LRP and $O(n^2 + n \cdot k \cdot a)$ using SHAP.

3.4 AICHRONOLENS for Meta-Learning

Since AICHRONOLENS's outputs $\mathbf{C}$ or $\mathbf{S}^T$ are images (two-dimensional with spatial relationships) and are sensitive to areas where the predictor could struggle to make accurate forecasts, we can leverage existing ResNet alongside the forecasting models for meta-learning. Figure 5 portrays the combined architecture. We choose ResNet for meta-learning, as compared to other well-known models like MobileNet [55], EfficientNet [56] or DenseNet [57], ResNet is specifically tailored to find patterns in images that consist of structured patterns and variations in intensity and color like the GAF. In contrast, other well-known models are more optimized for image classification tasks where features like edges, textures, and objects are present, which is not the case for the GAF. By using AICHRONOLENS's outputs as input of a ResNet we can predict potential errors that the original models might incur and then correct them to improve the overall accuracy. For this purpose, we need a second stage of training where the meta-learning model parameters are updated based on a new loss function $Z(\epsilon, \hat{\epsilon})$ where $\epsilon = Y - \hat{Y}$ is the error between the ground truth and the prediction and $\hat{\epsilon}$ is the prediction of ϵ made by the meta-learning model. Formally, we are using a model $f'(R_n)$, where $R_n \in \mathbf{C}$ is AICHRONOLENS's output at time t, to generate a prediction $\hat{\epsilon}_t$ that closely mimics the error $\epsilon_t = Y_t - \hat{Y}_t$ of the original forecasting model $f(X_t)$. The corrected prediction $\hat{Y}'_t = \hat{Y}_t + \hat{\epsilon}$ would be then just the original prediction $\hat{Y}_t$ plus the error $\hat{\epsilon}$ predicted by the meta-learning model. Note that if the MSE$(\epsilon, \hat{\epsilon})$ is higher than the MSE$(Y, \hat{Y})$ then the MSE$(Y, \hat{Y}')$ also will be, since $\epsilon - \hat{\epsilon} = Y - \hat{Y}'$.

4 EVALUATION SETUP

In this section, we first describe the datasets (§4.1) and AI models (§4.2), and then the forecasting tasks (§4.3) that form the evaluation setup we use to validate AICHRONOLENS.

4.1 Datasets

Datasets. We rely on two different datasets:

• D_1: The first dataset contains measurements of traffic volumes recorded in a production 4G network that serves a large metropolitan region in Europe. The dataset provides fine-grained information at 3-minute granularity about the traffic volumes at each BS. The dataset covers 3 months.

• D_2: The second dataset contains the estimated number of active users currently connected to production BSs [34] in Madrid. The dataset has been recorded using a popular LTE passive monitoring tool that decodes unencrypted information that the BSs exchange with the associated users. The dataset contains information at the millisecond level about the temporary user ID currently associated with the user, *i.e.*, the Radio Network Temporary Identifier (RNTI), and scheduling information. We use the methodology proposed in [58] to estimate the number of active users every 6 minutes. The dataset contains information from six different BSs and for this work, we have focused on the BS with carrier frequency equal to 816 MHz located in suburban areas of Madrid (zone II).

TABLE 1
Configuration and accuracy of the LSTM models trained for D_1 for single-step forecasting tasks

MODEL ID	NEURONS	LEARNING RATE	MAE
A	200	0.0001	0.96
B	100	0.0001	0.99
C	50	0.0001	1.09
A⋆	200	0.001	0.67
B⋆	100	0.001	0.68
C⋆	50	0.001	0.95

4.2 AI Models

We use two different sets of AI models depending on the objective, *i.e.*, time series forecasting or meta-learning. The models adopted for each objective are presented next.

Time series forecasting models. For the forecasting component of our framework, we considered the three following architectures.

- LSTM [59] is a Recurrent Neural Networks (RNN) model composed of cells. These cells feature various gates that enable the model to store both long-term and short-term memory. The design of LSTM aims to address a major limitation of standard RNNs: the vanishing gradient problem.
- PatchTST [15] is a transformer-based model that incorporates channel independence and sequence patching, significantly enhancing its performance compared to previous state-of-the-art models.
- DLinear [14] is a linear model that leverages a single projection layer that incorporates a pre-processing step separating the input data into its seasonal and trend components.

Meta learning model. For meta-learning, we use ResNet [33], which is a neural network based on Multilayer Perceptron (MLP) and convolutional layers that leverage residual connections. These connections allow for a significant increase in the maximum number of layers that can be added to the model, thereby notably enhancing its performance.

4.3 Time Series Forecasting Tasks

We considered three different forecasting tasks: single-step forecasting (*i.e.*, $H = 1$), multi-step horizon forecasting (*i.e.*, $H > 1$), and meta-learning.

Single-step Forecasting. In this scenario we build three different single-layer LSTM models (A, B, C), each with a varying number of LSTM cells, and train each model using two different learning rates (where model A* has a higher learning rate than A). Table 1 summarizes the details of these models and their accuracy.

Multi-step Forecasting. In addition to a vanilla LSTM model with 25 cells, for this task we also use two State-of-the-art (SoTA) performers namely PatchTST [15] and DLinear [14]. We set lookback windows of 300 and 150 values for D_1 and D_2 respectively. As for the horizon, we set $L = 60$ and $L = 30$ accordingly. Results are summarized in Table 2.

Meta-learning Error Prediction. For this experiment, we trained PatchTST and DLinear on datasets D_1 and D_2, using a lookback window of 60 values and a horizon of 20. We then generated AICHRONOLENS's outputs for both models. Since the forecasting task is multi-horizon, we have to compute the SHAP values for each element in the horizon, adding an extra dimension to the correlation vectors to accommodate all the different elements. Subsequently, we train a ResNet18 using AICHRONOLENS's outputs as input and the error $(Y - \hat{Y})$ as the target. Lastly, we correct the original prediction $\hat{Y}$ with the error prediction to get a new prediction $\hat{Y}'$ that hopefully is closer to Y than $\hat{Y}$. Table 3 shows the mean and standard deviation of MSE $(Y, \hat{Y}')$ over five trials with different random seeds and compares it with the original loss of the forecasting model MSE $(Y, \hat{Y})$. The results indicate that DLinear SHAP values significantly enhance the meta-learning process, leading to improvements in the original models across both datasets and models, and reaching a mean error rate reduction of $26.5\,\%$ in D_1. This may be attributed to the properties of the DLinear SHAP values, where a few samples within the window contribute most significantly to the final output. We also found that datasets with lower variance and a high number of observations (such as D_1) are more suitable for improving accuracy through meta-learning.

TABLE 2
Long-horizon accuracy of the forecasting models

DATASET	METRIC	MODELS DLinear	PatchTST	LSTM
D_1	MAE	0.217	0.179	0.182
	MSE	0.106	0.062	0.076
D_2	MAE	0.313	0.304	0.257
	MSE	0.161	0.158	0.116

TABLE 3
Meta-learning results

DATASET	METRIC	MODELS DLinear	PatchTST
D_1	MSE $(Y, \hat{Y})$	0.226	0.109
	MSE $(Y, \hat{Y}')$	0.098 ± 0.014	0.074 ± 0.007
D_2	MSE $(Y, \hat{Y})$	0.210	0.284
	MSE $(Y, \hat{Y}')$	0.197 ± 0.009	0.321 ± 0.022

5 EVALUATION RESULTS

In this section, we empirically evaluate AICHRONOLENS's explanations, showing the cause of model errors and optimizing model performance either via data augmentation and re-training or with meta-learning. We showcase the breadth of the explanations generated by AICHRONOLENS across all the time series forecasting models for single-step and multi-step forecasting trained for D_1 and D_2. Our main results are explanations that go beyond the simple attribution of relevance scores to the input sequence. Next, we specifically demonstrate that:

- AICHRONOLENS's output provides information about the model response in the form of points of interest that the legacy XAI techniques like SHAP can not reveal alone;
- The analysis of the points of interest makes it possible to distinguish cases when the model struggles because of a lack of knowledge or because data is inherently hard to predict.
- If the model struggles because of poor design, re-training with data augmentation or meta-learning are effective techniques to improve the model's performance.

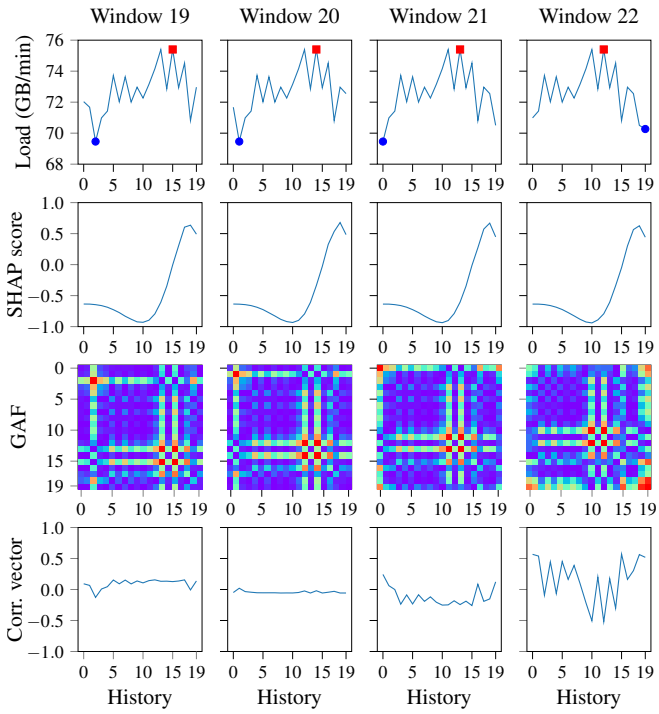


Fig. 6. A detailed look at AICHRONOLENS. Red squares and blue dots represent the local maxima and local minima respectively.

In summary, our results, derived from the quantitative analysis performed on the test set of both datasets are the following:

• R_1: In cases where LRP and SHAP produce relevance scores that are very similar over time and thus not informative, the correlation vectors that are the output of AICHRONOLENS pinpoint the temporal characteristics that stimulate the model. These are samples entering or leaving the input sequence whose values are either local maxima and local minima or very close to the local maxima or minima. The absence of such samples entering or leaving the input sequence turns strong positive or negative correlation values into weak correlation values. The observed behavior holds in general (*i.e.*, in the analyzed test sets of both datasets).

• R_2: AICHRONOLENS can spot errors that are due to poor model design and errors that are specific to data that are inherently hard to predict. The different types of errors can be identified by analyzing the patterns within $\mathbf{C}$ and, specifically, the points of interest as defined in §3.2.

• R_3: For LSTM models, higher learning rates (*i.e.*, models A*, B*, and C*) produce weaker correlations concerning models featuring smaller learning rates (*i.e.*, A, B, C). This behavior occurs regardless of the number of neurons.

• R_4: The outputs of AICHRONOLENS can be used to enhance model accuracy online through meta-learning. The key intuition behind this is to leverage the well-known ResNet model to predict when the original time series forecasting model is going to make errors, and then compensate for them before outputting the final prediction.

Finding R_1. Fig. 6 demonstrates qualitatively R_1. In the figure, we show in a combined fashion from top to bottom the input sequence (*e.g.*, time steps), the output of the XAI techniques (SHAP in this case), the GAF of the corresponding input sequence, and, finally, the correlation vectors. Note that the patterns of the correlation vectors differ from that of Fig. 4. We observe that AICHRONOLENS's output patterns depend on the number of elements of the lookback and not on model architecture (transformer-based like PatchTST, recurrent like LSTM, etc...) nor the length of the prediction horizon H. The patterns differ depending on the model's capabilities. Fig. 6 shows that while the SHAP scores are highly similar, the correlation vectors vary considerably. Specifically, we can appreciate in window 20 (used to predict time step 21) almost no correlation at all. This is because that SHAP assigns high relevance to samples in the input sequence closest in time to the next prediction (see history $15-19$ in the bottom), while these samples are not particularly relevant from the input sequence perspective (the GAF highlights the corresponding values with dark colors). In contrast, in time step 22 a new local minimum enters: the alignment between SHAP and GAF triggers a significant modification to the correlation vector if compared with the correlation vector of the previous time steps. In contrast, if left alone, SHAP would not capture such a change. We will see next why being blind to such changes is detrimental to model performance.

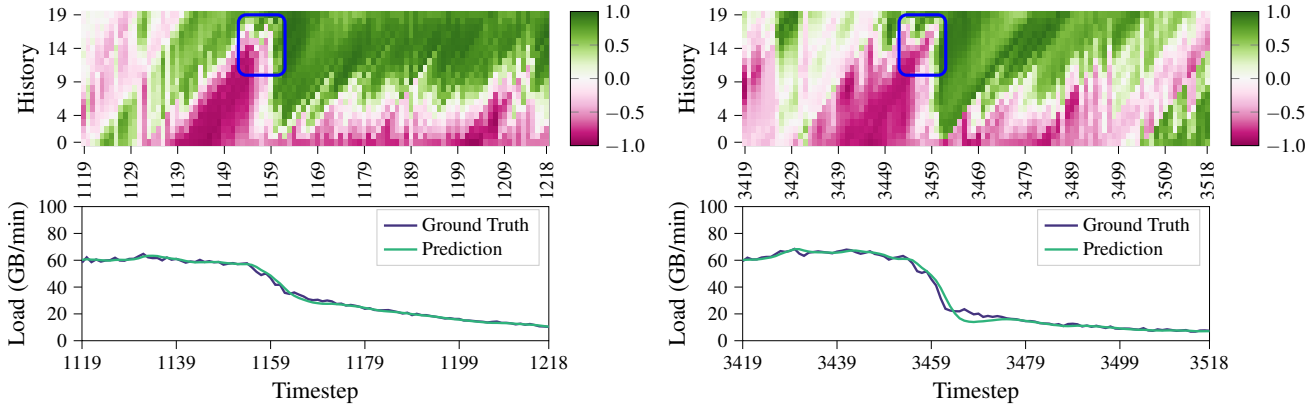
Finding R_2. We show quantitatively that AICHRONOLENS can detect different categories "E" of errors:

• E_1: is attributed to poor model design (shown for D_1),
• E_2: is specific to the dataset when using an optimized model (shown for D_2).

Analysis of E_1. Next, we will show that by tracing the root cause of the errors, it is possible to identify weaknesses due to poor model design that are not captured by coarse evaluation metrics like MAE or MSE. Finally, we will show that an informed model re-design can address such a shortcoming.

We perform a complete analysis over the AICHRONOLENS output $\mathbf{C}$ computed on the test set for all the trained models with both SHAP and LRP techniques. We observe that in the presence of trend changes in the time series, correlation vectors exhibit triangles with negative correlation followed by triangles with positive correlation. We find that the shape of the triangles varies. Well-formed, sharply outlined triangles like those in Fig. 7(a) (top) indicate that in the corresponding part of the time series, the model does not make significant mistakes (see Fig. 7(a) (bottom)). We define these triangles as *sharp*. In contrast, noisy *non-sharp* triangles like that of Fig. 7(b) (top) lead to high errors (see Fig. 7(b) (bottom)) in the presence of abrupt falls where the model is not able to accurately predict when the decrease stops in the actual data. This behavior is systematically observed throughout all the decreasing slopes present in the test set.

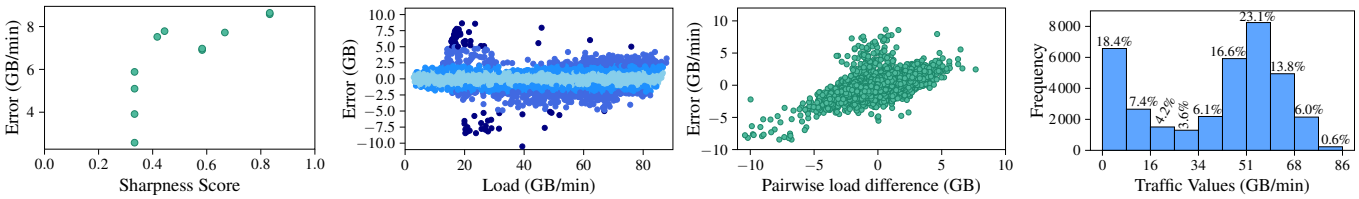
We now show a pattern recognition technique that identifies sharp and non-sharp triangles. Numerically, we identify the transition between triangles by computing the difference of the median correlation scores between two consecutive correlation vectors G_t and G_{t+1} in time steps t and $t+1$. Upon finding the column that interrupts the triangle, we set a window w of observation centered in such column. For example, $W=6$ indicates that we observe 3 preceding and succeeding columns forming a matrix $C_{n\times w}$. To spot sharpness, we perform the following operation on



(a) Sharp triangle. Top matrix $\mathbf{C}$ and bottom model errors.

(b) Non sharp triangle. Top matrix $\mathbf{C}$ and bottom model errors.

Fig. 7. Relating triangle sharpness of AIChronoLens output $\mathbf{C}$ with model errors



(a) Errors versus sharpness score

(b) Errors versus load

(c) Errors versus load changes

(d) Train set

Fig. 8. Root cause analysis of model errors

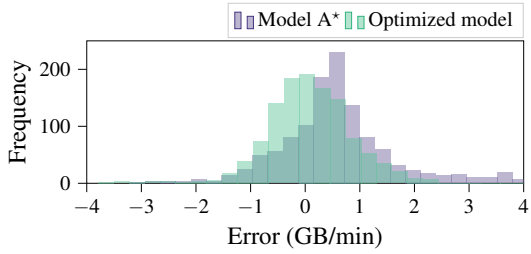


Fig. 9. Error before and after AIChronoLens diagnosis

each element $c_{i,j}$ of the matrix:

$$\overline{c}_{i,j} = \begin{cases} -1 & \text{if } -1 \leq c_{i,j} < 0; \\ 1 & \text{if } 0 < c_{i,j} \leq 1; \\ 0 & \text{if } c_{i,j} = 0. \end{cases} \tag{12}$$

On the resulting $\overline{C}_{n \times w}$, we compute the number h as the signed difference between the number of positive and negative values per each secondary diagonal of length w and store it into an array. By construction values of h are in the range $[-w : w]$. Next, we compute a sharpness score σ on the resulting array as follows:

$$\sigma = 1 - \frac{\sum_{i=1}^{n-(w-1)} h_i}{h_i \cdot (w+1)}. \tag{13}$$

For $0 < \sigma < 1$, the higher the value of σ, the higher the degree of non-sharpness. For $-1 < \sigma < 0$, the lower the value of σ, the higher the degree of sharpness. Finally, for $\sigma > 1$, the higher the value of σ, the higher the degree of sharpness. Our analysis shows a relation between the sharpness score and the error. Specifically, we observe that as the sharpness score increases, the error does too (see Fig. 8(a)). By taking a close look at the errors in the entire test set (see Fig. 8(b)), we observe that the highest absolute errors (5-8 GB/min) occur at moderate to low loads that are all connected with abrupt falls. Fig. 8(c) reveals that the model significantly underestimates the ground truth in the presence of severe load decrease (bottom-left in the plot). A careful analysis of the train set reveals a lack of training samples exactly in the proximity of traffic volumes for which the model is often mistaken (see Fig. 8(d)).

Ultimately, AIChronoLens allows appreciating that the model does not generalize in the presence of abrupt load decrease, as it has not observed such trends in the training set. In this way, our tool points to the solution to the model shortcoming, *i.e.*, data augmentation. Specifically, we copy from the train set several samples that represent 3 days (overall the train set was about 8 weeks) and append it to the end of the train set. Next, in the presence of load drops, we carefully remove samples with the objective of including those abrupt load decrease occurrences that were missing. We then train a new model, starting from model A* settings, with the augmented training dataset. The new model differs from A* only by the presence of a sigmoid activation function before the output layer. Fig. 9 outlines that the new optimized model outperforms the baseline model A* on two fronts. First, the optimized model reduces the errors of high magnitude (errors are computed as $x_{t+1} - \hat{x}_{t+1}$), which is especially clear on the right inside of the plot where the errors indicate that the model underestimates the traffic load. Second, the optimized model reduces the frequency of errors with small magnitude and yields an error bell centered around zero. These are all indicators of the poor training process of model A* vis-a-vis the optimized counterpart. Overall, by only considering windows around the abrupt load decrease, model A* would lead to a MAE $= 0.921$ (which is higher than the overall MAE computed over the entire test set, see Table 1) while the optimized model would lead to MAE $= 0.619$ which is an improvement of

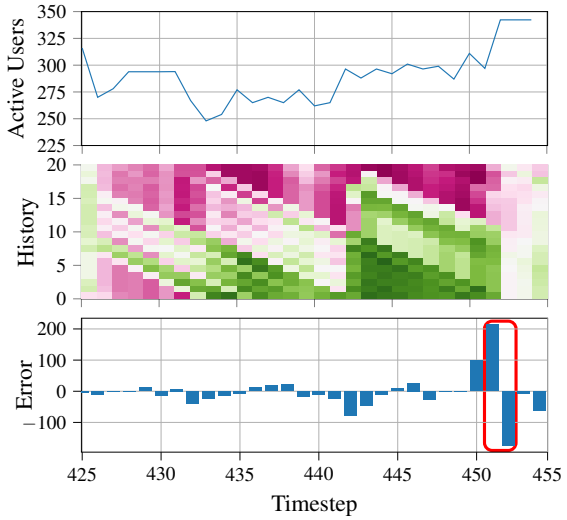
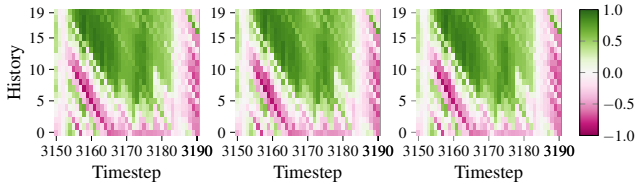
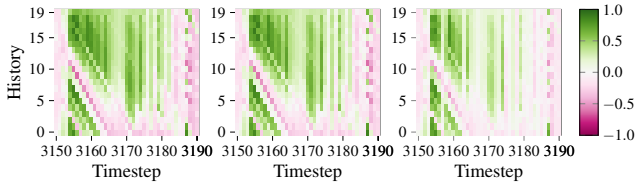


Fig. 10. Consecutive errors with high magnitude and sign change



(a) From left to right: model A, B, C



(b) From left to right: model A*, B*, C*

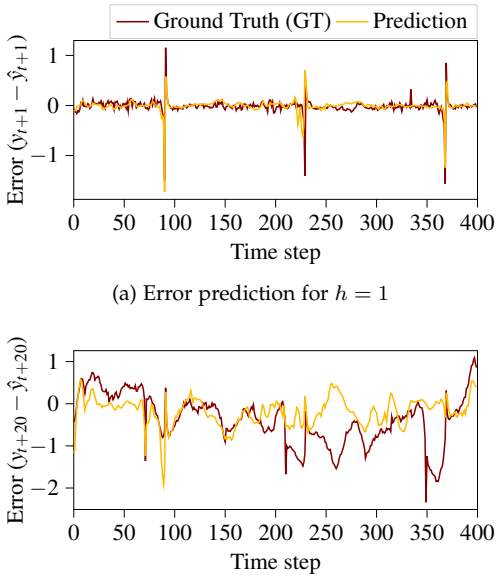
Fig. 11. Correlation vector for models with different learning rates: top 0.0001 and bottom 0.001

32 %. When applied to the entire test set, the optimized model achieves MAE $= 0.69$, only a 2 % decrease in accuracy concerning model A*.

Analysis of E_2. Unfortunately, even after addressing the category of errors E_1, there may be model errors due to the characteristics of the data itself. We now show that AICHRONOLENS can identify those too by analyzing the output $\mathbf{S}$ for D_2.

Fig. 10 shows qualitatively that there exist consecutive errors with high magnitude with a change of sign (*e.g.*, first positive than negative or vice versa). AICHRONOLENS identifies this behavior with triangles of positive or negative correlations over time that are interrupted by a full column with a weak correlation. Call G_t and G_{t+1} the correlation vectors in the time steps t and $t+1$: their similarity can be assessed via the Euclidean distance $d(G_t, G_{t+1}) = \sqrt{\sum_{i=1}^{n}(G_t^i - G_{t+1}^i)^2}$. We compute the Euclidean distance d between each two subsequent correlation vectors in the test set and normalize it in the range $[0 : 1]$. When $d > 0.6$, in 65 % of the cases, we find a change of error sign with a corresponding MAE $= 0.46$, much higher than the MAE computed over the entire dataset (*i.e.*, MAE $= 0.13$).

Finding R_3. To demonstrate qualitatively R_3, we portray in Fig. 11 examples of the correlation vectors in a window



(a) Error prediction for $h = 1$

(b) Error prediction for $h = 20$

Fig. 12. Meta-learning error prediction for two elements of the horizon H

of $W = 40$ time steps in the test set for all the models in Table 1. Here, the models with the lowest learning rate (on top) tend to exhibit a strong positive or negative correlation, with values approaching either 1 or -1. In contrast, the correlation scores tend to cluster around zero for models with higher learning rates, which indicates a weaker or negligible correlation. These behaviors are consistently observed across the test set and can be linked with the fact that there exists a trade-off between training cost and model adaptability. A higher learning rate makes it possible for a model to easily adapt to new contexts and unseen conditions but requires more computational resources to be trained. By contrast, training a model with a lower learning rate is cheaper at the expense of lower adaptability. We thus conclude that AICHRONOLENS offers precise insights into the heterogeneous accuracy of models trained with different learning rates. Orthogonal to the discussion on the learning rate, by analyzing Fig. 11 we also observe that the depth of the LSTM architecture, *i.e.*, the number of neurons, plays a marginal role for this specific dataset.

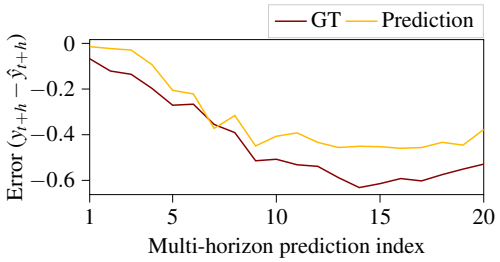
Finding R_4. To demonstrate R_4, we perform different analyses. In all, we use a ResNet model that is applied to the outputs of AICHRONOLENS to predict when the original time series forecasting model makes mistakes.

Fig. 12 illustrates DLinear error predictions through a meta-learning model over 400 time steps for both single step (Fig. 12(a)) and multi-step horizon predictions (Fig. 12(b)) of the original model. We remark that the ground truth for these experiments is the error that the original DLinear model makes. We observe that especially for $h = 1$, the meta-learner does a remarkably good job of spotting the error and this makes it possible to fix it. The error forecaster is less precise for $h = 20$, which is understandable given how hard this forecasting problem is. Since the original time series forecaster is predicting 20 time steps ahead (each of the time steps is of 3 min granularity), for $h = 20$ the model attempts to forecast the traffic load of the next hour.

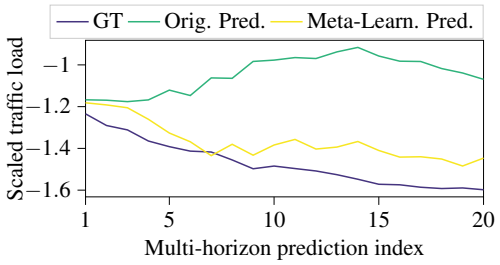
Next, for the same model and dataset, Fig. 13(a) portrays a

qualitative example of a multi-horizon prediction by varying the focus of the observation (i.e., $h = \{1, 2, \ldots, H\}$ in the x-axis). Note that also in this case the ground truth is the error that the original time series forecasting model makes. Figure 13(b) shows the corresponding accuracy improvement of meta-learning compared to the original time series forecasting model. Note that in this case, the ground truth is the traffic load, i.e., the original time series.

Finally, Fig. 14 portrays quantitatively the advantage of using as inputs to the meta-learner the outputs of AIChronoLens over the SHAP values directly across a variety of settings (datasets and models). The y-axis of the plots shows the model accuracy of the test set, having highlighted with a dashed line the original predictor's average accuracy without meta-learning. We observe that meta-learning improves in all the cases the average accuracy, and using as inputs of the meta-learner AIChronoLens's outputs is beneficial in most of the cases, but PatchTST. For example, in the case of DLinear for D_1 the accuracy improvement is 39.25% (Fig. 14(b)). PatchTST does not improve the average accuracy, but the distribution of the error is much wider with SHAP than with AIChronoLens.



(a) DLinear's error prediction



(b) Traffic load prediction

Fig. 13. Meta-learning error correction for a single observation

6 Discussion

In this section, we discuss limitations of the tool and areas of future work.

Focus on Uni-variate Time Series. In this work, we take into consideration AI models that work with uni-variate time series, *e.g.*, using traffic load to predict traffic load. Therefore, AIChronoLens is not a fit for models that incorporate additional features beyond historical traffic volumes as inputs, such as using both historical traffic volumes and the number of active users to predict traffic load.

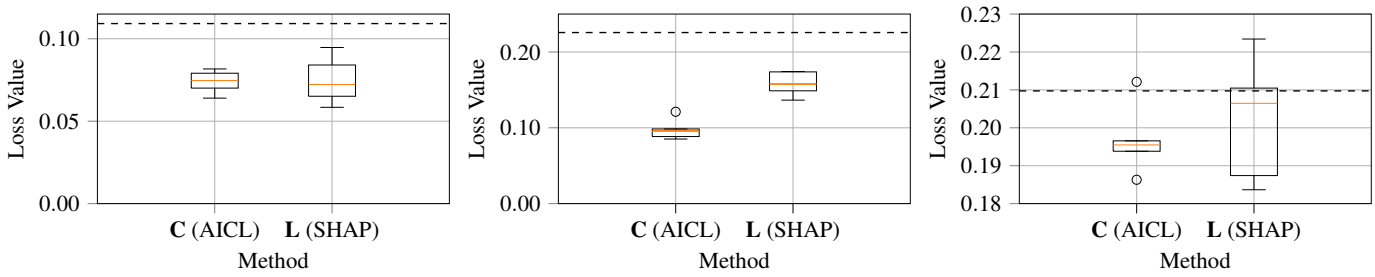
In time series forecasting, the use of multi-variate is beneficial to the accuracy of the predictor only in the presence of rich cross-correlation between the variates [60]. Hence, our focus to the uni-variate time series is an inherent limitation of the tool, but justified by the fact that models dealing with these type of time series are actually relevant. PatchTST [15], a recent state-of-the-art model is designed with a channel independence principle exactly to focus on one variate (channel) at a time. Furthermore, AI models for uni-variate time series inherently limit the forecasting input to one variate which makes it possible to isolate the contributions of each element of the history to the forecast.

An inherent limitation of focusing on uni-variate time series makes AIChronoLens not always applicable to other families of AI models such as Deep Reinforcement Learning (DRL). The reason lies in the way the Markov Decision Process (MDP) is defined which are the features that compose the action and state spaces. MDP is the mathematical framework used for modeling the decision-making problems that are a well-match for DRL agents. Specifically, to be applied to any DRL agents, AIChronoLens would require that the action is uni-modal and that it builds upon the observation of a single feature provided with historical values. This would map the exact setting we used for time series forecasting. However, to the best of our knowledge, this is not common in DRL. Our prior work [61], [39] analyzes the ColO-RAN agent (for Open RAN resource allocation/scheduling [62]), which uses multiple Key Performance Indicator (KPI) time series and a multi-modal action deciding allocation of physical resources and scheduling policy at each time step. This complexity makes it difficult for AIChronoLens to correlate neural network operation with the input sequences. Pensieve [63], while using a single-modal action (bit-rate allocation), relies on a state space with both historical (throughput, download times of past k chunks) and non-time-series features (next chunk sizes, buffer level, remaining chunks, last chunk bitrate), which are incompatible with AIChronoLens. Other DRL agents, such as [64], do not use historical measurements at all, thereby making the use of AIChronoLens not feasible.

Our future work will be devoted to extend AIChronoLens to tackle the case of multi-variate time series, which, in turn, will allow to apply AIChronoLens also for DRL.

Focus on Linear Correlations between $\mathbf{G}$ and L. The core motivation for using the Pearson correlation coefficient in AIChronoLens to provide an interpretable representation of what a model learns from time series data encoded as GAFs and is exemplified in Fig. 4. While a perfect model would fully capture the patterns in the normalized ground truth (represented as a heatmap), legacy XAI methods like SHAP often provide only a coarse approximation. AIChronoLens uses the Pearson correlation between the GAF $\mathbf{G}$ and the legacy XAI relevance scores L to better reflect the model's actual learning, highlighting both learned and, more specifically, unlearned patterns (*i.e.*, points of interest related to model errors). The Pearson coefficient is effective because it measures the linear relationship between the relevance scores and each row of the GAF, indicating how each input value X_t relates to all others, thus revealing linear dependencies like rises and falls in the time series.

Although other metrics like mutual information, Kendall's τ and Spearman's ρ can capture non-linear dependencies, they were ultimately less suitable for this specific application. Kendall's τ and Spearman's ρ both measure



(a) PatchTST - Dataset: D_1 (b) DLinear - Dataset: D_1 (c) DLinear - Dataset: D_2

Fig. 14. Predictor accuracy when using AICHRONOLENS C and SHAP L for meta-learning with different models and datasets. The dotted line is the original accuracy of the forecasting model.

monotonic relationships and they would be appropriate if the goal were specifically to assess rank-based associations which is not the problem when observing $\mathbf{G}$ and L. In contrast, mutual information is capable of detecting both linear and non-linear relationships. Our preliminary analysis produced results that were difficult to interpret in the context of model learning and error identification vis-a-vis with the Pearson correlation. For this reason, we chose to use for AICHRONOLENS a linear metric like the Pearson correlation coefficient. We reserve to our future work the in-depth analysis of non-linear metrics.

7 Related Work

Relevant to our work are studies on XAI techniques like visualization tools, and XAI, and LSTM-based forecasting applied to mobile networks. Despite promising, the application of imaging techniques to time series like GAF has found limited applicability in forecasting signal quality indicators [65].

XAI Visualization Tools. Visualization tools usually build on top of the legacy XAI techniques and make it possible to identify which part of the input was responsible for the output of the prediction and track the associated hidden state changes. TSViz [66] provides a 3D visualization tool for convolutional deep learning models. Long-Short Term Memories (LSTM)-Vis [67] and Sequence to Sequence (Seq2Seq)-Vis [68] are visualization tools that apply respectively to LSTM and Seq2Seq models. Both tools are tailored to NLP applications. Our work departs from the class of visualization tools because AICHRONOLENS provides a way to quantify the hidden relationships between explanations and the input.

XAI For Mobile Networks. Future 6G networks embrace the vision for native, explainable network intelligence. A seminal work [69] motivates the need for XAI and stresses that the lack of explainability may lead to poor AI/ML model design. This has been proved detrimental in the presence of adversarial attacks [19]. All the areas where AI is applied to mobile networking tasks can benefit from explainability. These include the physical and MAC layer design, network security mobility management, and localization [70]. One of the shortcomings of the existing XAI tools is the lack of deep relation between input data and the explanations [71]. While the foundations of AICHRONOLENS lie in harnessing such relationship, our work goes beyond [71] as *(i)* we formally show that it exists an ambiguity as legacy XAI techniques may assign the same relevance scores to diverse input sequences, and *(ii)* we resolve such ambiguity and exploit the richer expressiveness of the outputs of AICHRONOLENS to better comprehend the LSTM operations and optimize models performance.

Applications of Forecasting. The recent years have witnessed a surge of interest in applying deep neural networks for forecasting as they entail higher quality predictions than other approaches like statistical models [72]. The prediction of future traffic volumes forms the cornerstone of several applications that include anomaly event detection [8], scheduling of pilot signals for channel estimation [9], user throughput [10], buffer status reports [12], and to infer PRB utilization [13]. While all the above works rely on simple LSTM models, the works [73], [74], [75] are more complex ML architectures proposed with the unifying theme of better exploiting temporal characteristics of the inputs.

8 Conclusions

In this paper, we have investigated the timely and challenging problem of improving the understanding of AI models for time series forecasting like PatchTST, DLinear, and LSTM. We perform a quantitative and qualitative study that reveals the shortcomings of existing XAI techniques and propose AICHRONOLENS, a first-of-its-kind tool in the area of XAI. By linking the temporal characteristics of the input with relevance scores produced by existing XAI techniques, AICHRONOLENS can dive deep into the analysis of models' behavior. Via extensive evaluations with real-world mobile traffic traces, we show that AICHRONOLENS makes it possible to spot different categories of model errors, trace back the root causes, and possibly improve the poor model design either with re-training or online with meta-learning. Concerning re-training, a combined targeted data augmentation, and minor changes to the hyperparameters can improve performance by 32 %. Concerning meta-learning, feeding AICHRONOLENS's outputs to a ResNet model that predicts the errors of the original time series forecasting and corrects them can improve the average predictor's performance by up to 39 %.

Acknowledgments

This work is partially supported by bRAIN project PID2021-128250NB-I00 funded by MCIN/ AEI /10.13039/501100011033/ and the European Union ERDF "A way of making Europe"; by Spanish Ministry of Economic Affairs and Digital Transformation, European Union NextGeneration-EU projects MAP-6G TSI-063000-2021-63,

RISC-6G TSI-063000-2021-59 and AEON-ZERO TSI-063000-2021-52; P. Fernandéz received funding from "Programa Investigo" (grant 2022-C23.I01.P03.S0020-0000038) funded by EU-NextGenerationEU and SEPE/PRTR. C. Fiandrino is a Ramón y Cajal awardee (RYC2022-036375-I), funded by MCIU/AEI/10.13039/501100011033 and the ESF+.

REFERENCES

[1] Ericsson, "Mobility Report, June 2024. Technical Report." 2023, Accessed on 10/07/2024: https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/june-2024.

[2] P. D. Francesco, F. Malandrino *et al.*, "Assembling and using a cellular dataset for mobile network analysis and planning," *IEEE Transactions on Big Data*, vol. 4, no. 4, pp. 614–620, 2018.

[3] C. Fiandrino, C. Zhang *et al.*, "A machine learning-based framework for optimizing the operation of future networks," *IEEE Communications Magazine*, vol. 58, no. 6, 2020.

[4] S. Zhao, X. Jiang *et al.*, "Cellular network traffic prediction incorporating handover: A graph convolutional approach," in *Proc. of IEEE SECON*, 2020, pp. 1–9.

[5] L. Chen, T.-M.-T. Nguyen *et al.*, "Data-driven C-RAN optimization exploiting traffic and mobility dynamics of mobile users," *IEEE Transactions on Mobile Computing*, vol. 20, no. 5, pp. 1773–1788, 2021.

[6] D. Bega, M. Gramaglia *et al.*, "DeepCog: Optimizing resource provisioning in network slicing with AI-based capacity forecasting," *IEEE JSAC*, vol. 38, no. 2, pp. 361–376, 2020.

[7] J. Lin, Y. Chen *et al.*, "A data-driven base station sleeping strategy based on traffic prediction," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2021.

[8] H. D. Trinh, L. Giupponi *et al.*, "Urban anomaly detection by processing mobile traffic traces with LSTM neural networks," in *Proc. IEEE SECON*, 06 2019, pp. 1–8.

[9] C. Fiandrino, G. Attanasio *et al.*, "Traffic-driven sounding reference signal resource allocation in (beyond) 5G networks," in *Proc. of IEEE SECON*, 2021, pp. 1–9.

[10] J. Lee, S. Lee *et al.*, "PERCEIVE: Deep learning-based cellular uplink prediction using real-time scheduling patterns," in *Proc. ACM MobiSys*, 2020, p. 377–390.

[11] D. Overbeck, N. A. Wagner *et al.*, "Proactive resource management for predictive 5G uplink slicing," in *Proc. of IEEE GLOBECOM*, 2022, pp. 1000–1005.

[12] Q. Zhang, A. Nikou *et al.*, "Predicting buffer status report (BSR) for 6G scheduling using machine learning models," in *Proc. of IEEE WCNC*, 2022, pp. 632–637.

[13] Y. Xu, F. Yin *et al.*, "Wireless traffic prediction with scalable gaussian process: Framework, algorithms, and verification," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1291–1306, 2019.

[14] A. Zeng, M. Chen *et al.*, "Are transformers effective for time series forecasting?" in *Proc. of the AAAI conference on Artificial Intelligence*, vol. 37, no. 9, 2023, pp. 11 121–11 128.

[15] Y. Nie, N. H. Nguyen *et al.*, "A time series is worth 64 words: Long-term forecasting with transformers," in *Proc. of ICLR*, 2023.

[16] H. D. Trinh, L. Giupponi *et al.*, "Mobile traffic prediction from raw data using LSTM networks," in *Proc. IEEE PIMRC*, Sep. 2018, pp. 1827–1832.

[17] W. Huang, X. Peng *et al.*, "Adversarial attack against LSTM-based DDoS intrusion detection system," in *Proc. of IEEE ICTAI*, 2020, pp. 686–693.

[18] D. Adesina, C.-C. Hsieh *et al.*, "Adversarial machine learning in wireless communications using RF data: A review," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 77–100, 2023.

[19] S. Moghadas Gholian, C. Fiandrino *et al.*, "Spotting deep neural network vulnerabilities in mobile traffic forecasting with an explainable AI lens," in *IEEE INFOCOM*, 2023.

[20] S. M. Lundberg, G. Erion *et al.*, "From local explanations to global understanding with explainable AI for trees," *Nature machine intelligence*, vol. 2, no. 1, pp. 56–67, 2020.

[21] A. Mahimkar, A. Sivakumar *et al.*, "Auric: Using data-driven recommendation to automatically generate cellular configuration," in *Proc. of the ACM SIGCOMM*, 2021, p. 807–820.

[22] T. Rojat, R. Puget *et al.*, "Explainable artificial intelligence (XAI) on timeseries data: A survey," 2021.

[23] G. Montavon, A. Binder *et al.*, *Layer-Wise Relevance Propagation: An Overview*. Springer International Publishing, 2019, pp. 193–209.

[24] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. of NIPS*, 2017, pp. 4768–4777.

[25] M. T. Ribeiro, S. Singh *et al.*, ""Why Should I Trust You?": Explaining the predictions of any classifier," in *Proc. of ACM SIGKDD*, 2016, p. 1135–1144.

[26] A. Shrikumar, P. Greenside *et al.*, "Learning important features through propagating activation differences," in *Proc of ICMLR*, vol. 70, Aug 2017, pp. 3145–3153.

[27] D. Han, Z. Wang *et al.*, "DeepAID: Interpreting and improving deep learning-based anomaly detection in security applications," in *Proc. of ACM CCS*, 2021, pp. 3197–3217.

[28] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at AAAI Conf. on Artificial Intelligence*, January 2015.

[29] X. Wang, K. Smith-Miles *et al.*, "Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series," *Neurocomputing*, vol. 72, no. 10-12, pp. 2581–2594, 2009.

[30] C. Lemke and B. Gabrys, "Meta-learning for time series forecasting and forecast combination," *Neurocomputing*, vol. 73, no. 10, pp. 2006–2016, 2010.

[31] B. N. Oreshkin, D. Carpov *et al.*, "Meta-learning framework with applications to zero-shot time-series forecasting," in *Proc. of AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9242–9250.

[32] M. Guimarães and D. Carneiro, "A meta-learning approach to error prediction," in *Proc. of CISTI)*, 2021, pp. 1–6.

[33] K. He, X. Zhang *et al.*, "Deep residual learning for image recognition," in *Proc. of IEEE CVPR*, 2016, pp. 770–778.

[34] P. F. Fernández Pérez, C. Fiandrino *et al.*, "Characterizing and modeling mobile networks user traffic at millisecond level," in *Proc. of ACM WiNTECH*, 2023, p. 64–71.

[35] C. Fiandrino, E. Pérez Gómez *et al.*, "AIChronoLens: advancing explainability for time series AI forecasting in mobile networks," in *Proc. of IEEE INFOCOM*, 2024, Available online: https://git2.networks.imdea.org/wng/aichronolens.

[36] D. Gunning and D. Aha, "DARPA's explainable artificial intelligence (XAI) program," *AI magazine*, vol. 40, no. 2, pp. 44–58, 2019.

[37] S. E. Middleton, E. Letouzé *et al.*, "Trust, regulation, and human-in-the-loop AI: within the european region," *Communications of the ACM*, vol. 65, no. 4, p. 64–68, Mar 2022.

[38] S. Wang, M. A. Qureshi *et al.*, "Explainable AI for 6G use cases: Technical aspects and research challenges," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 2490–2540, 2024.

[39] C. Fiandrino, L. Bonati *et al.*, "EXPLORA: AI/ML explainability for the Open RAN," *Proc. ACM Netw.*, vol. 1, no. CoNEXT3, Nov 2023.

[40] D. A. Broniatowski *et al.*, "Psychological foundations of explainability and interpretability in artificial intelligence," *NIST, Tech. Rep*, 2021.

[41] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." *Queue*, vol. 16, no. 3, p. 31–57, jun 2018.

[42] A. Abanda, U. Mori *et al.*, "Ad-hoc explanation for time series classification," *Knowledge-Based Systems*, vol. 252, p. 109366, 2022.

[43] R. Mochaourab, A. Venkitaraman *et al.*, "Post hoc explainability for time series classification: Toward a signal processing perspective," *IEEE Signal Processing Magazine*, vol. 39, no. 4, pp. 119–129, 2022.

[44] W. Ge, J.-W. Huh *et al.*, "An interpretable ICU mortality prediction model based on logistic regression and recurrent neural networks with LSTM units." in *Proc. of the AMIA Annual Symposium*, vol. 2018. American Medical Informatics Association, 2018, p. 460.

[45] M. Korobov and K. Lopuhin, "ELI5 is a python library - v. 0.11," 2021, available at (accessed 26/10/2021): https://eli5.readthedocs.io/en/latest/.

[46] U. Schlegel, H. Arnout *et al.*, "Towards a rigorous evaluation of XAI methods on time series," in *Proc. of IEEE/CVF ICCVW*, 2019, pp. 4197–4201.

[47] H. Turbé, M. Bjelogrlic *et al.*, "Evaluation of post-hoc interpretability methods in time-series classification," *Nature Machine Intelligence*, vol. 5, no. 3, pp. 250–260, 2023.

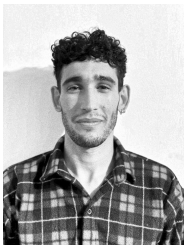
[48] F. Petitjean, A. Ketterlin *et al.*, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[49] M. Cuturi and M. Blondel, "Soft-DTW: a differentiable loss function for time-series," in *Proc. of PMLR ICML*, D. Precup and Y. W. Teh, Eds., vol. 70, 08 2017, pp. 894–903.

[50] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.
[51] J. C. B. Gamboa, "Deep learning for time-series analysis," in *arXiv*, 2017.
[52] Y. Zhang, Y. Hou *et al.*, "Multi-scale signed recurrence plot based time series classification using inception architectural networks," *Pattern Recognition*, vol. 123, p. 108385, 2022.
[53] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.
[54] H. Chen, S. M. Lundberg *et al.*, "Explaining a series of models by propagating shapley values," *Nature communications*, vol. 13, no. 1, p. 4512, 2022.
[55] A. G. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[56] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. of ICML*, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97, Jun 2019, pp. 6105–6114.
[57] G. Huang, Z. Liu *et al.*, "Densely connected convolutional networks," in *Proc. of IEEE CVPR*, 2017, pp. 2261–2269.
[58] G. Attanasio, C. Fiandrino *et al.*, "In-depth study of RNTI management in mobile networks: Allocation strategies and implications on data trace analysis," *Computer Networks*, vol. 219, p. 109428, 2022.
[59] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.
[60] J. du Preez and S. F. Witt, "Univariate versus multivariate time series forecasting: an application to international tourism demand," *International Journal of Forecasting*, vol. 19, no. 3, pp. 435–451, 2003.
[61] A. Duttagupta, M. Jabbari *et al.*, "SymbXRL: symbolic explainable deep reinforcement learning for mobile networks," in *Proc. of IEEE INFOCOM*, 2025, Available online: https://github.com/RAINet-Lab/symbxrl.
[62] M. Polese, L. Bonati *et al.*, "ColO-RAN: Developing machine learning-based xApps for Open RAN closed-loop control on programmable experimental platforms," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2022.
[63] H. Mao, R. Netravali *et al.*, "Neural adaptive video streaming with pensieve," in *Proc. of ACM SIGCOMM*, 2017, p. 197–210.
[64] Q. An, S. Segarra *et al.*, "A deep reinforcement learning-based resource scheduler for massive MIMO networks," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 1, 2023.
[65] B. Y. L. Kimura, J. Almeida *et al.*, "Deep learning in beyond 5G networks with image-based time-series representation," *arXiv preprint arXiv:2104.08584*, 2021.
[66] S. A. Siddiqui, D. Mercier *et al.*, "TSViz: Demystification of deep learning models for time-series analysis," *IEEE Access*, vol. 7, 2019.
[67] H. Strobelt, S. Gehrmann *et al.*, "LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks," *IEEE TVCG*, vol. 24, no. 1, pp. 667–676, 2018.
[68] H. Strobelt, S. Gehrmann *et al.*, "Seq2seq-Vis: A visual debugging tool for sequence-to-sequence models," *IEEE TVCG*, vol. 25, no. 1, pp. 353–363, 2019.
[69] W. Guo, "Explainable artificial intelligence for 6G: Improving trust between human and machine," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 39–45, 2020.
[70] U. Challita, H. Ryden *et al.*, "When machine learning meets wireless cellular networks: Deployment, challenges, and applications," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 12–18, 2020.
[71] C. Fiandrino, G. Attanasio *et al.*, "Toward native explainable and robust AI in 6G networks: Current state, challenges and road ahead," *Computer Communications*, vol. 193, pp. 47–52, 2022.
[72] S. P. Sone, J. J. Lehtomäki *et al.*, "Wireless traffic usage forecasting using real enterprise network data: Analysis and methods," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 777–797, 2020.
[73] L. Mei, J. Gou *et al.*, "Realtime mobile bandwidth and handoff predictions in 4G/5G networks," *Computer Networks*, vol. 204, p. 108736, 02 2022.
[74] F. Li, Z. Zhang *et al.*, "A meta-learning based framework for cell-level mobile network traffic prediction," *IEEE Transactions on Wireless Communications*, vol. 22, no. 6, pp. 4264–4280, 2023.
[75] H. Nan, X. Zhu *et al.*, "MSTL-GLTP: A global–local decomposition and prediction framework for wireless traffic," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 5024–5034, 2023.

Pablo Fernández Pérez is a research engineer at IMDEA Networks Institute. His research focus on resilient and explainable AI, and AI for time series analysis.

Claudio Fiandrino is a Research Assistant Professor at IMDEA Networks Institute. His research focuses on explainable and robust AI for next-generation mobile networks. Claudio has received numerous awards for his research, including a Fulbright scholarship, several Spanish national grants and several Best Paper Awards. He is member of IEEE and ACM, and of the Editorial Board of IEEE NETWORKING LETTERS and ELSEVIER COMPUTER NETWORKS.



Eloy Pérez Gómez was an intern student at IMDEA Networks Institute during April-June of 2022, working towards his B.Sc. degree.

Hossein Mohammadalizadeh is currently a PhD student at Hasso Plattner Institute (HPI), Germany. He was a visiting M.Sc. student at IMDEA Networks Institute in March-July 2023.

Marco Fiore is a Research Professor at IMDEA Networks Institute, where he leads the Networks Data Science group, and CTO at Net AI Tech Ltd. He received a PhD degree from Politecnico di Torino, and a Habilitation a Diriger des Recherches from Universite de Lyon. He held tenured positions at INSA Lyon in France, and Consiglio Nazionale delle Ricerche in Italy. He was a recipient of a European Union Marie Curie fellowship and a Royal Society International Exchange fellowship. His research interests are the interface of computer networks, data analysis and machine learning.

Joerg Widmer is Research Director and Research Professor of IMDEA Networks in Madrid, Spain. His research focuses on wireless networks, ranging from extremely high frequency millimeter-wave communication and MAC layer design to mobile network architectures. He authored more than 150 conference and journal papers, three IETF RFCs and holds 13 patents. He was awarded an ERC consolidator grant, the Friedrich Wilhelm Bessel Research Award, a Spanish Ramón y Cajal grant, as well as eight best paper awards.