Privacy Risks in Time Series Models

Membership Inference in Deep Learning-Based Time Series Forecasting Models

Master's thesis in Computer science and engineering

Nicolas Johansson

Tobias Olsson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Privacy Risks in Time Series Models

Membership Inference in Deep Learning-Based Time Series Forecasting Models

Nicolas Johansson
Tobias Olsson

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Privacy Risks in Time Series Models
Membership Inference in Deep Learning-Based Time Series Forecasting Models
Nicolas Johansson
Tobias Olsson

Supervisor: Stefano Sarao Mannelli, Computer Science and Engineering
Advisors: Fazeleh Hoseini, Daniel Nilsson, and Johan Östman, AI Sweden
Examiner: Marina Axelson-Fisk, Mathematical Sciences

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Image illustrating the intersection of time series data and privacy, generated using OpenAI's DALL-E through ChatGPT.

Typeset in LaTeX
Gothenburg, Sweden 2025

Privacy Risks in Time Series Models
Membership Inference in Deep Learning-Based Time Series Forecasting Models

Nicolas Johansson
Tobias Olsson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

As machine learning models become increasingly prevalent in time series forecasting applications—such as healthcare or energy systems—their potential to leak sensitive data, for example through membership inference attacks (MIAs), raises serious privacy concerns. While MIAs are well-studied in domains like image classification, their implications for time series models remain underexplored. This thesis investigates the vulnerability of deep learning-based time series forecasting models to MIAs, focusing on univariate and multivariate prediction tasks using LSTM and N-HiTS architectures. We conduct systematic experiments on two real-world datasets—EEG signals and electricity load usage—from distinct domains, characterized by varying temporal granularity and privacy sensitivity.

Several MIA techniques are evaluated, including LiRA and RMIA—originally state-of-the-art in other data modalities and adapted here for time series—as well as an ensemble-based attack previously shown to be effective for time series models. In addition, we introduce two new attacks: (i) Multi-Signal LiRA, a multi-signal variant of LiRA that can leverage a diverse set of time series-specific signals such as trend, seasonality, and TS2Vec embeddings; and (ii) Deep Time Series (DTS) attack, a novel deep learning-based MIA tailored specifically for time series forecasting. Performance is assessed across both sample-level and user-level inference scenarios.

Our findings reveal key patterns in MIA performance relative to model type, dataset characteristics, and attack signals, providing insight into how various attack settings affect privacy leakage in time series domains. These discoveries underscore the need for robust privacy auditing tools and the integration of privacy-preserving techniques when working with sensitive temporal data, while also providing guidance for future research on privacy risks in time series machine learning models.

Keywords: machine learning, deep learning, membership inference attacks, time series forecasting, data privacy, EEG, electricity load, privacy auditing

Acknowledgements

We would like to express our sincere gratitude to our advisors in the LeakPro project at AI Sweden: Fazeleh Hoseini, Johan Östman, and Daniel Nilsson. Without your continued support, insightful discussions and valuable feedback, this thesis would not be possible.

We also wish to thank our academic supervisor at Chalmers, Stefano Sarao Mannelli, for your guidance and support throughout the project.

Nicolas Johansson and Tobias Olsson, Gothenburg, 2025-06-06

Contents

List of Figures

List of Tables

1

Introduction

The field of artificial intelligence (AI) is evolving with an ever-growing pace, attributable to increased data availability, computing power, and technological breakthroughs [1], and the industrial use of machine learning (ML) models for time series data is becoming increasingly common in industries such as healthcare, finance, and Internet of Things (IoT) applications [2]. However, the rapid development in the field raises concerns for data security. Preserving privacy in models trained on private information is crucial, since potential data leakage can lead to exposure of sensitive information through attacks such as *model inversion* [3] or *training data extraction* [4, 5]. This can in turn lead to consumer trust loss, malicious use of sensitive data, and legal and financial penalties. Furthermore, data leakage may violate regulations, for example, the General Data Protection Regulation (GDPR) [6]. These ethical and legal concerns about data leakage may lead to reluctance of sharing knowledge and trained models in sensitive domains. Therefore, careful risk assessments strategies are needed to promote collaboration [7, 8].

One of the most widely studied methods for evaluating data privacy in machine learning models is the membership inference attack (MIA) [9, 10]. MIAs exploit model predictions to determine whether a specific data point was included in the model's training dataset, risking exposure of sensitive information [11]. These types of attacks often make unrealistic assumptions about the adversary's model and data access, for example detailed knowledge about the training procedure and access to the training data itself. Hence MIAs might not be practical in a real attack scenario, but are commonly used as a measure of a model's training data privacy. Furthermore, studies show that defenses against MIAs also provide protection against other attacks, demonstrating the connection between them and the viability of using MIAs as data privacy assessment [12]. Besides auditing a model's training data leakage, MIAs have been shown to be useful in machine unlearning [13], evaluation of differential privacy [14], model extraction attacks [15], and data extraction attacks [4, 5].

While MIAs are well-studied in areas such as image classification and natural language processing [16, 17], their implications for time series models, characterized by unique temporal dependencies, remain underexplored. This thesis aims to address this gap by further investigating the vulnerability of time series forecasting models to MIAs, contributing to the growing need for robust privacy auditing tools to secure ML applications in sensitive domains. Understanding and mitigating this

vulnerability requires systematic analysis of attacks specific to time series models and their unique data properties.

Recent studies have begun exploring MIAs in the context of time series models, revealing new attack methodologies and their implications for privacy [18, 19]. These studies suggest that the characteristics of time series models, including seasonality and trend components, can be exploited within MIAs. However, further research is necessary to determine the full extent of this vulnerability. While the mentioned initial studies have made important groundwork, they remain limited in scope and do not capture the full range of risks posed by different attacks, attack settings, and application domains. A more robust and systematic framework is needed—one that incorporates a diverse set of MIAs to comprehensively evaluate privacy risks in time series forecasting models. Without such thorough assessment, time series forecasting ML systems may continue to expose sensitive information in ways that remain undetected.

Time series data used in forecasting tasks can vary widely in both temporal resolution and sensitivity. For instance, electroencephalogram (EEG) signals—which measure electrical activity of the brain—are often recorded at high frequencies (e.g., millisecond resolution) and are inherently biometric, making them closely tied to an individuals identity and even suitable for strong authentication methods [20]. Models trained on such data may inadvertently memorize personal information, increasing the risk of re-identification through MIAs. In contrast, electricity load forecasting datasets are often aggregated over households or regions and recorded at lower temporal resolutions (e.g., hourly), which might suggest lower privacy risk. However, prior work has shown that even seemingly coarse-grained usage patterns can be used to infer personal habits, occupancy patterns, or appliance usage in smart grid environments [21, 22]. These examples illustrate that both fine-grained biomedical data, such as EEG, and coarser behavioral signals derived from electricity consumption can pose meaningful privacy risks, highlighting the importance of evaluating model vulnerabilities across diverse time series domains.

1.1 Purpose

The purpose of this study is to systematically investigate privacy leakage risks in deep learning-based time series forecasting models, a domain that remains largely underexplored. Through a literature review, we identify and adapt current state-of-the-art attacks, originally designed for classification models, to the time series domain. Furthermore, we explore potential attack signals and develop new attacks specifically tailored to time series models.

AI Sweden's LeakPro project, an open-source privacy auditing tool for ML models, currently supports image, tabular, and graph-based data but does not address time series [23]. By extending LeakPro and developing a robust evaluation framework specific to time series data, this thesis contributes both to the scientific understanding of privacy vulnerabilities in time series models and to the practical development of tools for privacy risk assessment—ultimately providing a foundation for future

research.

1.2 Scope

This thesis focuses on investigating privacy risks in deep learning-based time series forecasting models using membership inference attacks. Specifically, we focus on univariate and multivariate forecasting tasks where the goal is to predict future values based on past observations. Exogenous variables are not considered.

Other time series tasks such as classification or probabilistic forecasting are outside the scope of this study. The reason for this is the inherent differences in how the models operate across the different tasks—a forecasting model outputs raw predictions of future time series values, whereas classification models output confidence estimates over a fixed set of classes. Due to the availability of model output confidences, the latter case is more similar to scenarios that have already been studied for other modalities, such as image classification or natural language processing, and was therefore not prioritized. Probabilistic forecasting is more similar to regular forecasting but differs in its outputs: it provides distributions or quantiles rather than point estimates. Because membership inference attacks rely on model outputs to infer membership, this distinction makes it a technically separate problem. For this reason, and due to time constraints, it was excluded from the scope of this study.

Our experiments consider black-box and gray-box adversary MIA scenarios, excluding white-box settings, and does not evaluate other kinds of privacy risks such as data poisoning or model inversion. We consider only deep learning-based forecasting models, specifically the LSTM and N-HiTS architectures. Furthermore, we restrict our analysis to two real-world datasets—EEG signals and electricity load usage—and do not explore any privacy-preserving training methods or defenses.

2

Theory

This chapter presents the theoretical foundation for the thesis, including key concepts, mathematical formulations, and relevant MIA methodologies. The theories discussed here form the basis for the methods proposed in Chapter 3 and their application in the context of risk assessment for time series models.

2.1 Time Series Forecasting

In time series forecasting, the goal is to make predictions based on historical temporal data. This could be anything from financial stock prices to electricity demand. In this thesis, we consider forecasting M numerical variables H steps into the future based on a history of the previous T time steps. We define the *lookback* history as $X \in \mathbb{R}^{T \times M}$ and the *horizon* of values to be predicted as $Y \in \mathbb{R}^{H \times M}$, where T and H represent the respective time dimensions.

Traditional time series forecasting methods focus on statistical techniques. This includes autoregressive (AR) models which model the next value of a series as a linear combination of the past values and white noise, and moving average (MA) models which instead predict the next value based on a linear combination of the past *estimation errors* [24]. More advanced statistical models often combine these two components (ARMA) or extend the framework with more components. For example, ARIMA extends ARMA by introducing an *integrated* component which subtracts the time series with a lagged version of itself, and SARIMA extends this further by introducing an additional set of AR and MA components for the *seasonality component* of the time series [24]. There is also the possibility of incorporating *exogenous variables* (SARIMAX), which are external factors that can impact the time series but are not influenced by it; consider for example, in the context of load forecasting, weather or electricity price.

Although Makridakis et al. [25] provided evidence for the continued superiority of statistical time series forecasting methods, deep learning models have in more recent years demonstrated a capacity to achieve higher accuracy, at the cost of vast computational costs [26]. These models utilize advanced neural networks such as the Transformer, WaveNet, and EnsembleDL together with large amounts of data to learn complex features to aid in forecasting. In this context, data preprocessing, including normalization, is often performed in order to ensure robust models. In supervised learning, samples are created by applying a *sliding window* of fixed size

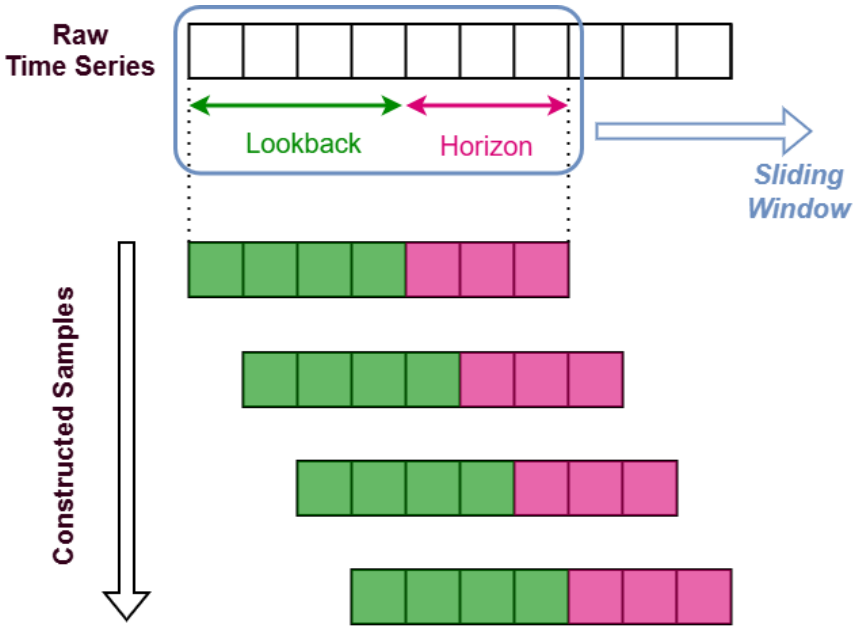


Figure 2.1: Univariate ($M = 1$) sample extraction with sliding window.

over the raw time series, where each position yields a training sample with specified lookback and horizon. This is illustrated in Figure 2.1. For normalization, some common approaches include *min-max scaling* which transforms all values into the range $[0, 1]$ based on the minimum and maximum values in the series, and *standard scaling* which removes mean and scales values to unit variance. To handle data with outliers, *robust scaling* may be utilized to scale the data similarly to the standard scaler, but based solely on the interquartile range (IQR) of the data; that is, statistics are extracted from the difference between the 75th ($Q3$) and 25th ($Q1$) percentiles of the data [27]. The formula for robust scaling is given by

$$X_{\text{robust}} = \frac{X - \mu_{IQR}}{\sigma_{IQR}} \tag{2.1}$$

where μ_{IQR} and σ_{IQR} denotes the mean and standard deviation, respectively, when computed over $X_{IQR} = \{x \in X \mid Q1 \leq x \leq Q3\}$.

In summary, time series forecasting includes a wide range of methods, from classical statistical models to modern deep learning approaches. Proper data preprocessing and sampling are crucial steps to ensure that models can effectively learn temporal patterns and make accurate forecasts. The following sections will dive deeper into different deep learning architectures, time series signals, and how these can be utilized for membership inference attacks.

2.2 Models

This section provides an overview over the model architectures used in this study, from the theory of their underlying components to their application in time series forecasting.

2.2.1 Long Short-Term Memory

A Recurrent Neural Network (RNN) is a type of neural network specifically designed for processing sequential data [28]. Unlike traditional feedforward networks, RNNs architectures utilizes *recurrent connections* which allows them to retain information from previous inputs in the sequence [29].

The *vanishing and exploding gradients problem* is major shortcoming of RNNs. It occurs when the gradients either go to zero (vanishes) or become very large (explodes) due to repeated multiplications when backpropagating through time. This problem becomes more pronounced the more time-steps we want to process, and as a result the RNN may fail to capture long-term dependencies. One solution to the vanishing gradient problem is the Long Short-Term Memory (LSTM) unit, introduced by Hochreiter and Schmidhuber in 1997 [30].

The LSTM utilizes memory cells to efficiently store and process information over time, and is illustrated in Figure 2.2. Each cell consists of three gates that together determine how the memory is updated over time:

1. The *forget gate*, f_t, controls how much information that is kept from the previous cell state c_{t-1}. Based on the previous hidden state, h_{t-1}, and the current input, x_t, the forget gate computes a value between 0 and 1 for each element in the previous cell state:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.2}$$

   where W_f is a weight matrix, b_f is a bias vector, and σ is the sigmoid activation function. f_t is then element-wise multiplied with c_{t-1} to regulate the amount of information kept.

2. The *input gate*, i_t, decides how much new information from the current input, x_t, should be stored in the updated cell state c_t. Similar to the forget gate, the input gate computes a value between 0 and 1 for each element:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.3}$$

   where W_i and b_i are the learnable weights of the input gate. Simultaneously, the *candidate cell state*, $\tilde{c}_t$, is computed as

$$\tilde{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.4}$$

   where *tanh* denotes the hyperbolic tangent function and W_c, b_c are additional learnable weights.

3. The *output gate*, o_t, determines what information to output as the current hidden state h_t. The output gate activation is computed as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o). \tag{2.5}$$

   To produce the updated *hidden state*, h_t, the output o_t is multiplied with the hyperbolic tangent of the updated memory cell state c_t:

$$h_t = o_t \odot \tanh(c_t) \tag{2.6}$$

where $\odot$ denotes the Hadamard (element-wise) product and c_t is given by

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t. \tag{2.7}$$

The updated hidden state h_t can then be passed into subsequent LSTM layers, or used as a representation of the input sequence for downstream tasks, for example classification or forecasting.

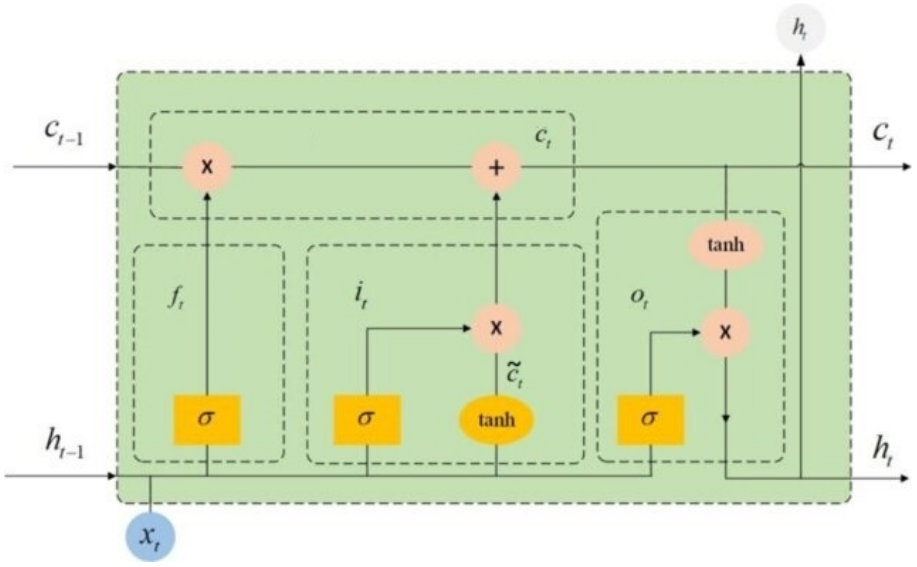


Figure 2.2: Diagram of an LSTM cell, illustrating the cell (c_t) and hidden state (h_t) update mappings with the forget (f_t), input (i_t), and output (o_t) gates. Reproduced from [31], CC BY 4.0.

2.2.2 N-BEATS

Oreshkin et al. [32] introduces the *Neural Basis Expansion Analysis for interpretable Time Series forecasting* (N-BEATS) model, a univariate deep learning forecasting model achieving state-of-the-art performance across many benchmark datasets. Besides performing well on benchmarks such as M4 and TOURISM [33, 34], it also enables interpretable forecasting inspired by traditional time series decomposition methods.

The N-BEATS architecture is based on backward and forward residual links and deep stacks of blocks of fully connected (FC) layers with ReLU activations. Each block takes as input a lookback, and produces an output of length *lookback* + *horizon*: predictions of the *basis expansion* coefficients of the backcast and forecast. The difference between the backcast and input is used as the *residual*, enhancing the forecast of downstream blocks when stacked together by removing the portion of the input signal that is already well approximated. The final block in the stack outputs a *stack residual* which is passed to the next stack, while also aggregating the block forecasts to a single *stack forecast*.

Each block may transform its output with different basis functions, for example modeling periodical fluctuations or polynomial trends. Further, each stack is tailored

to capture different patterns in the input time series, allowing the model to learn from complex data. These stacks are hierarchically structured and the input flows from the first stack until the last, where the final forecast is the aggregate of all the stack forecasts. Figure 2.3 outlines an overview of the N-BEATS model architecture.

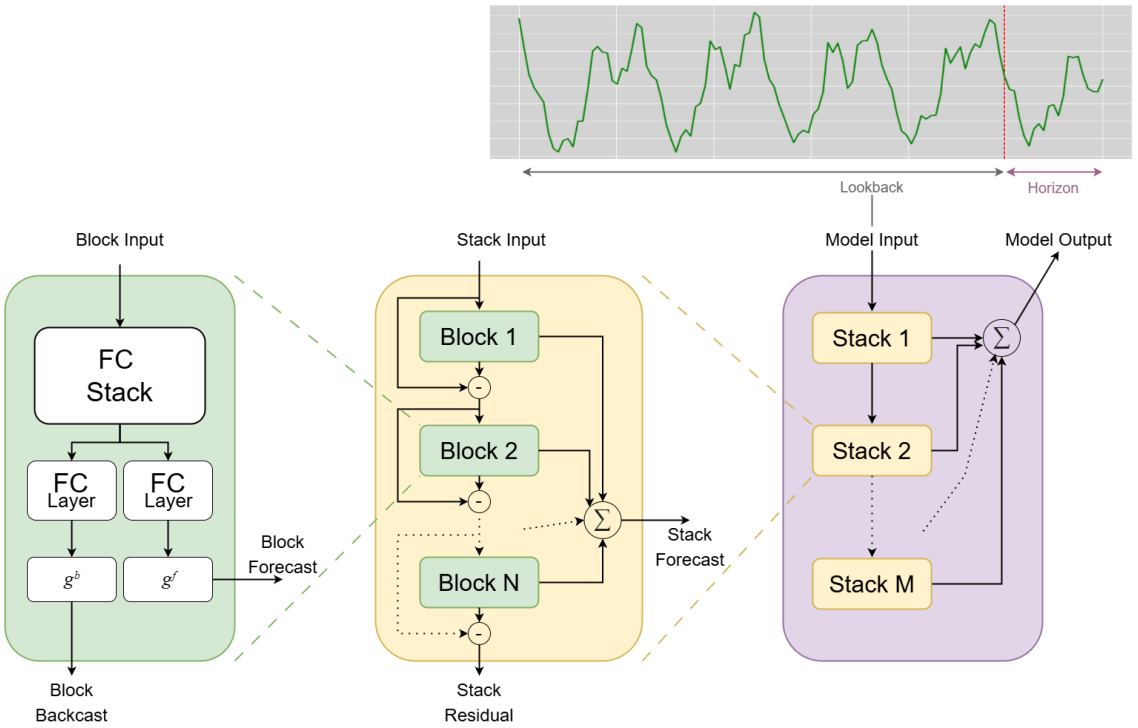


Figure 2.3: The univariate N-BEATS model architecture; featuring several hierarchically structured stacks, each containing several blocks. Here, g^b and g^f denote the backward and forecast basis functions, respectively.

2.2.3 N-HiTS

Challu et al. [35] introduces the *Neural Hierarchical interpolation for Time Series* (N-HiTS) architecture; a hierarchical, multiresolution model that improves both accuracy and computational efficiency from previous state-of-the-art neural forecasting methods. At the time of its publication, N-HiTS outperformed the latest Transformer architectures by 20% while also being 50 times faster to train. N-HiTS extends the fully-connected N-BEATS architecture described in Section 2.2.2, by enhancing its input decomposition via multi-rate data sampling and its output synthesizer via multi-scale interpolation, resulting in a hierarchical forecast construction that greatly reduces computational costs. In contrast to its predecessor, N-BEATS, the N-HiTS model supports *multivariate* time series and allows for *exogenous* inputs.

2.3 Membership Inference Attacks

A Membership Inference Attack aims to determine whether a specific datapoint was part of a target model's training dataset (member) or not (non-member) [11]. In

Game 1 this is formalized as a privacy-game using the framework proposed by Salem et al. [36].

Game 1 Basic Membership Inference game

1: $S \sim \mathcal{D}^n$ ▷ Sample n datapoints from population without replacement
2: $b \sim \{0, 1\}$ ▷ Flip a fair coin
3: **if** $b = 0$ **then**
4: $\quad x \sim S$ ▷ Sample a challenge datapoint x from training data S
5: **else**
6: $\quad x \sim \mathcal{D} \setminus S$ ▷ Sample a challenge datapoint x not in training data S
7: **end if**
8: $\theta_{\text{target}} \leftarrow \mathcal{T}(S)$ ▷ Train target model θ_{target} on training set S
9: $\tilde{b} \leftarrow \mathcal{A}(\mathcal{T}, \mathcal{D}, n, \theta_{\text{target}}, x)$ ▷ Adversary predicts membership of x

In this game the adversary $\mathcal{A}$ aims to determine whether a challenge datapoint x was part of the models training set ($b = 0$) or not ($b = 1$) and succeeds if $\tilde{b} = b$. The adversary is assumed to have access to the stochastic training algorithm $\mathcal{T}$, data distribution $\mathcal{D}$ (also called population), the number of training examples n, the target ML model θ_{target} and the challenge datapoint x.

There are many variations of the membership inference game which imply different attack settings. Generally, these different attack settings concern the assumptions of what information and computational budget is available to the adversary. One important assumption is the level of access to the ML model—white-box implying that the adversary has access to the model weights and architecture whilst black-box implying access to model through an API. Usually, the attack setting falls somewhere in between, sometimes referred to as a gray-box scenario. Even in the black- and gray-box scenario, there are different levels of access this API can grant, in the context of classification models this could be access to logits, loss or class prediction.

In practical applications, machine learning models are often trained on datasets where each individual contributes multiple data points. These data points might include, for example, an entire time series of EEG signals for a patient, or multiple measurements of electricity load usage for a household. This scenario motivates a variant of membership inference (MI) known as user-level MI, where the adversary's goal is to determine whether any data associated with a particular individual was included in the model's training set. In contrast to sample-level MI attacks that target individual data points, user-level MI targets the presence of an individual's entire dataset. This concept can be formalized as a privacy game, as illustrated in Game 2, where the adversary attempts to infer the membership of a collection of datapoints S_{chal} from one individual. Note that in this game, each user's dataset D_i is disjoint from other users.

Game 2 User-level Membership Inference game

$b \sim \{0, 1\}$ ▷ Flip a fair coin
$\mathcal{D}_1, ..., \mathcal{D}_m \sim \mathcal{D}$ ▷ Sample m users from population without replacement
for $i = 1, ..., m$ **do**
    $S_i \sim \mathcal{D}_i^n$ ▷ Sample n datapoints from each user without replacement
end for
if $b = 0$ **then** ▷ Challenge user *is* in the training set
    $i \sim \{1, ..., m\}$
    $S_{\text{chal}} \leftarrow S_i$
else ▷ Challenge user *is not* in the training set
    $\mathcal{D}_{\text{chal}} \sim \mathcal{D} \setminus \{\mathcal{D}_1, ..., \mathcal{D}_m\}$
    $S_{\text{chal}} \sim \mathcal{D}_{\text{chal}}^n$
end if
$\theta_{\text{target}} \leftarrow \mathcal{T}(\bigcup_{i=1}^m S_i)$ ▷ Train target model θ_{target} on training set
$\tilde{b} \leftarrow \mathcal{A}(\mathcal{T}, \mathcal{D}, n, m, \theta_{\text{target}}, S_{\text{chal}})$ ▷ Adversary predicts membership of S_{chal}

2.3.1 Existing Attacks

Membership inference attacks is a rapidly expanding field of research, where new attacks have often been able to improve upon previous work by orders of magnitude in performance. Most research target classification models and thus use either loss, logits, confidences or predicted label as a signal to discriminate between in- and out-members. For black-box attacks, there are two main approaches to inferring membership in literature; statistical hypothesis testing [16, 17, 37] or using binary classifiers [11, 38].

The Likelihood Ratio Attack (LiRA) [16] and the Robust Membership Inference Attack (RMIA) [17] are two state-of-the-art membership inference attacks targeting classification models. LiRA boasts high performance at very low false positive rates and the more recent RMIA demonstrates significantly better performance when using a small amount of shadow models. Both attacks utilize statistical hypothesis testing to infer membership. The Ensemble Attack [38] instead uses an ensemble of classifiers to infer membership. Despite being originally designed to attack classification models, it can be adapted to time series models since it can utilize any input signal [19].

The attack definitions below assume a target model θ_{target} and a challenge datapoint $x \in \mathcal{D}$ that you want to test for membership. The attacks will return a $\text{Score}_{\text{MIA}}(x; \theta_{\text{target}})$ which can be thresholded by β to determine membership. In other words, classify x as a member if $\text{Score}_{\text{MIA}}(x; \theta_{\text{target}}) \geq \beta$ and non-member otherwise. Since all the following attacks target classification models, let $f_\theta(x)$ represent the softmax probability outputs of the model θ for a given datapoint x. Additionally, let $f_\theta(x)_y$ denote the softmax probability assigned to the true class y associated with x.

Likelihood Ratio Attack (LiRA)

LiRA works by training a set of *shadow models*—auxiliary models resembling the target model, used to estimate how the its outputs vary depending on whether a datapoint was part of the training set. These are trained on different overlapping subsets of the population, such that there is a non-empty set of *in* models $\Theta_{\text{in}} = \{\theta_{\text{in}}^1, ..., \theta_{\text{in}}^{n_{\text{in}}}\}$ that had the challenge datapoint x as part of their training dataset and a non-empty set of *out* models $\Theta_{\text{out}} = \{\theta_{\text{out}}^1, ..., \theta_{\text{out}}^{n_{\text{out}}}\}$ which did not. In an optimal attack setting, these shadow models should match the target model in terms of architecture and training procedure, but a mismatch is shown to only have a slight impact on the attack performance [16]. Next, softmax probabilities $f_\theta(x)_y$ are collected for target and shadow models which are then rescaled according to:

$$\phi_\theta(x) = \ln \frac{f_\theta(x)_y}{1 - f_\theta(x)_y}. \tag{2.8}$$

This results in a rescaled confidence $\phi_{\theta_{\text{target}}}(x)$ for the target model, and a set of rescaled confidences for the in and out models.

$$\Phi_{\text{in}} = \{\phi_\theta(x) \mid \theta \in \Theta_{\text{in}}\} \tag{2.9}$$
$$\Phi_{\text{out}} = \{\phi_\theta(x) \mid \theta \in \Theta_{\text{out}}\} \tag{2.10}$$

Assuming Φ_{in} and Φ_{out} are Gaussian distributed, their distributions can be estimated using the empirical mean and variance of each set.

$$\mu_{\text{in}} = \text{mean}(\Phi_{in}) \qquad \sigma_{\text{in}} = \text{std}(\Phi_{in})$$
$$\mu_{\text{out}} = \text{mean}(\Phi_{out}) \qquad \sigma_{\text{out}} = \text{std}(\Phi_{out})$$

Finally, a likelihood ratio test is performed to determine the *MIA score*:

$$\text{Score}_{\text{MIA}}(x; \theta) = \frac{p(\phi_{\theta_{\text{target}}}(x) \mid \mathcal{N}(\mu_{\text{in}}, \sigma_{\text{in}}^2))}{p(\phi_{\theta_{\text{target}}}(x) \mid \mathcal{N}(\mu_{\text{out}}, \sigma_{\text{out}}^2))} \tag{2.11}$$

The original paper also describes an offline version of their attack (as opposed to the online attack described above) which only requires *out* shadow models. Since it relies solely on *out* models, this approach allows one to pretrain a set of shadow models and later query any datapoint without retraining. This leads to the following simplified test:

$$\text{Score}_{\text{MIA}}(x; \theta_{\text{target}}) = 1 - \Pr[\phi_{\theta_{\text{target}}}(x) > Z] \tag{2.12}$$

where $Z \sim \mathcal{N}(\mu_{out}, \sigma_{out}^2)$.

Robust Membership Inference Attack (RMIA)

RMIA improves upon previous membership inference attacks by grounding its methodology in a more mathematically rigorous Bayesian hypothesis testing framework. Empirically, RMIA has been shown to outperform previous attacks, especially in situations where the computational budget is limited and few shadow models are

available. Moreover, RMIA demonstrates greater robustness to distribution shifts, such as mismatches in training procedure, training set distribution, or model architecture between the shadow models and the target model [17].

RMIA works similarly to LiRA in terms of training shadow models, but specifically requires an equal ratio of *in* versus *out* models for a given datapoint. RMIA defines a pair-wise likelihood ratio test for challenge point x relative to a random datapoint $z \in \mathcal{D}$ given target model θ_{target}:

$$\text{LR}_{\theta_{\text{target}}}(x, z) = \left(\frac{\Pr(x \mid \theta_{\text{target}})}{\Pr(x)}\right) \cdot \left(\frac{\Pr(z \mid \theta_{\text{target}})}{\Pr(z)}\right)^{-1}. \tag{2.13}$$

$\Pr(x \mid \theta)$ is here defined as the likelihood function of model θ evaluated on data point x. In the case of classification models, this is equivalent to the softmax score for the true class of x, i.e. $\Pr(x \mid \theta) = f_\theta(x)_y$ using the model θ. $\Pr(x)$ is the marginal likelihood of x, which for the online attack is defined as:

$$\Pr(x) = \frac{1}{2}\Big(\underbrace{\frac{1}{k}\sum_{\theta \in \Theta_{\text{in}}} \Pr(x \mid \theta)}_{\Pr(x)_{\text{in}}} + \underbrace{\frac{1}{k}\sum_{\theta \in \Theta_{\text{out}}} \Pr(x \mid \theta)}_{\Pr(x)_{\text{out}}}\Big) \tag{2.14}$$

where $2k$ is the number of shadow models. The marginal $\Pr(z)$ is calculated similarly utilizing only *out* models:

$$\Pr(z) = \frac{1}{k}\sum_{\theta'_{\bar{x}}} \Pr(z \mid \theta'_{\bar{x}}) \tag{2.15}$$

Now that the pair-wise likelihood ratio test is defined, the attack proceeds as follows:

1. Sample a set of datapoints $z \sim \mathcal{D}$ from the population.
2. Calculate the pair-wise likehood ratio test $\text{LR}_\theta(x, z)$ for each z.
3. Determine $\text{Score}_{\text{MIA}}(x; \theta_{\text{target}})$ as the fraction of z samples where $\text{LR}_\theta(x, z) \geq \gamma$, with γ as a parameter of the attack.

An offline version of the attack is also defined, where the marginal likelihood is instead calculated by

$$\Pr(x) \approx \frac{1}{2}\Big((1 + a)\Pr(x)_{\text{out}} + (1 - a)\Big) \tag{2.16}$$

where a is an additional attack parameter. Similarly to LiRA, only *out* models are utilized in the offline attack.

The Ensemble Attack

Shachor et. al. [38] propose the Ensemble attack, a method which involves training many ML models on small distinct subsets of the data to classify membership. The purpose of this is to have an ensemble of many small specialized attack models, able to capture different patterns in the data.

The method to train the ensemble of classifiers can be defined as follows:

1. Select a fixed amount of disjoint, fixed-size subsets of the population containing exactly 50% members/non-members.
2. Within each subset, randomly assign 50% of the data as train and the rest as test.
3. Using the training data within a subset, train many different types of classifiers to predict the membership of these datapoints. The input feature to the classifier will be the target model output $f_{\theta_{\text{target}}}(x)$ for a datapoint or some other feature derived from this.
4. Evaluate these classifiers on the test set and choose the best classifier using some metric (for example area under ROC-curve).
5. Repeat step 2-4 for different random train/test splits and choose the best model using the same metric as step 4.
6. Collect the best model for each subset to form an ensemble. Repeat step 1-5 for n instances to get different subsets and form a final ensemble of models.

Using this ensemble of classifiers, $\text{Score}_{\text{MIA}}(x; \theta_{\text{target}})$ can be determined as the fraction of models that classify the datapoint as a member.

The process of training the classifiers requires the adversary to know the membership of the datapoints, creating an unrealistic attack scenario. The authors propose an alternative method where shadow models are trained, and the shadow models, with membership labels realistically known to an adversary, are used when training classifiers instead of the target model. The authors call these two methods audit mode, which involves training on the target model and true membership labels, and attack mode, which involves training on shadow models and their membership labels. In this thesis, we focus solely on audit mode.

2.3.2 Quantifying the Performance of MIAs

MIAs can be framed as a binary hypothesis testing problem, where the goal is to label each data point in a given set as either a **member** or a **non-member**. The attacks described in Section 2.3.1 compute a membership inference score for each data point, reflecting the attacks' confidence that the point was part of the training set. A decision threshold β is then applied to classify each point; classifying x as a **member** if $\text{Score}_{\text{MIA}}(x; \theta_{\text{target}}) \geq \beta$, and **non-member** otherwise. The threshold β directly controls the trade-off between sensitivity and specificity: lowering β increases the true positive rate but may also lead to more false positives, whereas raising it reduces

false positives at the cost of more false negatives. By varying β, one can create a Receiver Operating Characteristic (ROC) curve, which quantifies the effectiveness of an attack across different decision thresholds. One can also select the decision threshold that maximizes the overall accuracy, precision, recall or some other metric.

Many papers have considered average-case metrics such as accuracy (Equation 2.17), precision (Equation 2.18), recall (Equation 2.19) or area under ROC-curve when evaluating attacks [11, 39–43].

$$\text{Accuracy}_{\mathcal{A}} = \max_{\beta} \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.17}$$

$$\text{Recall}_{\mathcal{A}} = \max_{\beta} \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.18}$$

$$\text{Precision}_{\mathcal{A}} = \max_{\beta} \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.19}$$

Carlini et al. [16] argue however, that average-case metrics are not useful for evaluating membership inference attacks. In a privacy setting false positives directly impact the credibility of an attack whilst false negatives merely reduces the amount of points able to be extracted. Therefore Carlini et al. argue that results of MIAs should be reported using an ROC curve or the true positive rate (TPR) at very low false positive rates (FPR), typically $\leq 1\%$.

2.4 Signals

This section describes the various time series signals used as *attack features* in our study—that is, properties extracted from model predictions, such as error metrics or representations, which are used to statistically distinguish between in and out models for a given sample. For this purpose, we will denote the target time series as $Y = [y_1, ..., y_H] \in \mathbb{R}^{H \times M}$, and corresponding prediction as $\hat{Y} = [\hat{y}_1, ..., \hat{y}_H] \in \mathbb{R}^{H \times M}$, where H refers to the length of the series (horizon), and M is the number of variables.

2.4.1 Mean Squared Error

The Mean Squared Error (MSE) is one of the most common metrics in time series forecasting, and measures the average of the squared errors. The formula used for computing MSE in the multivariate case of M predicted variables is given by

$$\text{MSE}(Y, \hat{Y}) = \frac{1}{H \cdot M} \sum_{h=1}^{H} \sum_{m=1}^{M} (\hat{Y}_{h,m} - Y_{h,m})^2. \tag{2.20}$$

Note that MSE is bounded by $[0, +\inf)$ and is not invariant to scaling.

2.4.2 Mean Absolute Error

The Mean Absolute Error (MAE) is also one of the most common metrics in time series forecasting, measuring the average of the absolute errors. The formula used for computing MAE in the multivariate case of M predicted variables is given by

$$\text{MAE}(Y, \hat{Y}) = \frac{1}{H \cdot M} \sum_{h=1}^{H} \sum_{m=1}^{M} |\hat{Y}_{h,m} - Y_{h,m}|. \tag{2.21}$$

Similar to MSE, MAE is bounded by $[0, +\inf)$ and is not invariant to scaling.

2.4.3 Symmetric Mean Absolute Percentage Error

The Symmetric Mean Absolute Percentage Error (SMAPE) is also a common metric for determining the accuracy of a forecast. The formula for computing SMAPE is given by

$$\text{SMAPE}(Y, \hat{Y}) = \frac{1}{H \cdot M} \sum_{h=1}^{H} \sum_{m=1}^{M} \frac{|\hat{Y}_{h,m} - Y_{h,m}|}{|\hat{Y}_{h,m}| + |Y_{h,m}|}. \tag{2.22}$$

To make the equation numerically stable, a small value ϵ is added to the denominator of the fraction within the sum. Unlike the previous two signals, SMAPE is bounded by $[0, 1]$ and is *invariant to scaling*.

To map SMAPE from the bounded interval $[0, 1]$ to an unbounded scale, a logit-like transformation (2.23) can be applied and the result refered to as *rescaled SMAPE*. The motivation for this is the fact that some attacks perform better on an unbounded scale and a logit-like transformation is experimentally shown to produce more gaussian-like distributions in some scenarios [16].

$$\text{RSMAPE}(Y, \hat{Y}) = \log\left(\frac{\text{SMAPE}(Y, \hat{Y})}{1 - \text{SMAPE}(Y, \hat{Y})}\right) \tag{2.23}$$

2.4.4 Trend

Trend is the overall, long-term change in a time series. One technique for estimating the trend is fitting a polynomial of degree d to each variable in the time series [19]. For multivariate time series this can be represented as

$$\mathbf{Y} = \mathbf{PA} \tag{2.24}$$

where $\mathbf{A}$ is the coefficient matrix modeling the trend, and $\mathbf{P}$ is the Vandermonde matrix [44]. The Vandermonde matrix is constructed from the time vector $\boldsymbol{t}$ where $t_i = \frac{i-1}{H}$ for $i = 1, 2, ..., H$, containing powers of t up to $d - 1$, and is given by

$$\mathbf{P} = \begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{d-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{d-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_H & t_H^2 & \cdots & t_H^{d-1} \end{bmatrix}. \tag{2.25}$$

The coefficient matrices for the prediction and target series are then obtained by the least squares solution:

$$\mathbf{A} = (\mathbf{P^T P})^{-1}\mathbf{P^T Y} \tag{2.26}$$

$$\hat{\mathbf{A}} = (\mathbf{P^T P})^{-1}\mathbf{P^T \hat{Y}} \tag{2.27}$$

Finally, the L_2 norm between the coefficients of the true and predicted values is used as the attack feature:

$$\|\mathbf{A} - \hat{\mathbf{A}}\|_2 = \sqrt{\sum_{h=1}^{H}\sum_{m=1}^{M}(\mathbf{A}_{h,m} - \hat{\mathbf{A}}_{h,m})^2} \tag{2.28}$$

2.4.5 Seasonality

Seasonality is the regular, periodic fluctuations in a time series. In multivariate data, this can be extracted with the 2-dimensional Discrete Fourier Transform (2D-DFT) which breaks down the series into its frequency components considering both the timeline and the interaction of the different variables [19]. The 2D-DFT is applied to the target time series matrix $\mathbf{Y} \in \mathbb{R}^{M \times H}$ with two sequential 1D-DFTs; first a column-wise transformation with the Fourier matrix F_M over the variables, then a row-wise transformation with F_H over the time dimension. This can be expressed simply as

$$\mathbf{C} = \mathbf{F_M Y F_H}^{\mathbf{T}} \tag{2.29}$$

where C is the Fourier coefficient matrix and the 1D-DFT matrices F_M, F_H are given by

$$F_M = \begin{bmatrix} \cos(2\pi \cdot 0 \cdot \frac{0}{M}) & \dots & \cos(2\pi \cdot 0 \cdot \frac{M-1}{M}) \\ \vdots & \vdots & \vdots \\ \cos(2\pi \cdot \frac{M}{2} \cdot \frac{0}{M}) & \dots & \cos(2\pi \cdot \frac{M}{2} \cdot \frac{M-1}{M}) \\ \sin(2\pi \cdot 0 \cdot \frac{0}{M}) & \dots & \sin(2\pi \cdot 0 \cdot \frac{M-1}{M}) \\ \vdots & \vdots & \vdots \\ \sin(2\pi \cdot \frac{M}{2} \cdot \frac{0}{M}) & \dots & \sin(2\pi \cdot \frac{M}{2} \cdot \frac{M-1}{M}) \end{bmatrix}, \tag{2.30}$$

$$F_H = \begin{bmatrix} \cos(2\pi \cdot 0 \cdot \frac{0}{H}) & \dots & \cos(2\pi \cdot 0 \cdot \frac{H-1}{H}) \\ \vdots & \vdots & \vdots \\ \cos(2\pi \cdot \frac{H}{2} \cdot \frac{0}{H}) & \dots & \cos(2\pi \cdot \frac{H}{2} \cdot \frac{H-1}{H}) \\ \sin(2\pi \cdot 0 \cdot \frac{0}{H}) & \dots & \sin(2\pi \cdot 0 \cdot \frac{H-1}{H}) \\ \vdots & \vdots & \vdots \\ \sin(2\pi \cdot \frac{H}{2} \cdot \frac{0}{H}) & \dots & \sin(2\pi \cdot \frac{H}{2} \cdot \frac{H-1}{H}) \end{bmatrix}. \tag{2.31}$$

The same approach is used to extract seasonality for the predicted values:

$$\hat{\mathbf{C}} = \mathbf{F_M}\hat{\mathbf{Y}}\mathbf{F_H}^{\mathbf{T}} \tag{2.32}$$

Finally, the L_2 norm between the coefficients of the true and predicted values is used as the attack feature:

$$\|\mathbf{C} - \hat{\mathbf{C}}\|_2 = \sqrt{\sum_{h=1}^{H}\sum_{m=1}^{M}(\mathbf{C}_{h,m} - \hat{\mathbf{C}}_{h,m})^2} \tag{2.33}$$

2.4.6 TS2Vec

TS2Vec is a framework introduced by Yue et al. [45] for learning representations of univariate and multivariate time series at arbitrary semantic levels, leveraging contrastive learning in a hierarchical way to produce robust contextual representations for each time step. These representations may then be aggregated to obtain a high dimensional vector representation over the entire series.

The study investigates a wide range of datasets and results demonstrate a significant performance increase over previous state-of-the-art unsupervised time series representations. Further, the representations was proven to work very well as features in several time series tasks (classification, forecasting, and anomaly detection). Utilizing only a linear regression model with L2 regularization on top of the learned representations, it was found to beat advanced models such as Temporal Convolutional Network (TCN) and the Informer in multivariate forecasting [45].

An overview of the TS2Vec architecture is shown in Figure 2.4. First, two overlapping subseries are randomly sampled from the input time series, where the aim is to ensure consistent contextual representations within their shared segment. These raw inputs are fed into the shared *encoder* which consists of three components:

1. The *input projection layer* which maps the observation at each timestamp into a high-dimensional latent space.
2. The *timestamp masking module* that generates augmented context views by randomly hiding portions of the sequence.
3. A dilated Convolutional Neural Network (CNN) module with ten residual blocks which extracts the contextual representation at each timestamp.

The resulting representations are fed into a *hierarchical contrasting* model (right part of figure) which applies max pooling to produce multiple levels of different resolutions. Loss functions are then applied to all granularity levels, forcing the encoder to learn representations at various scales [45].

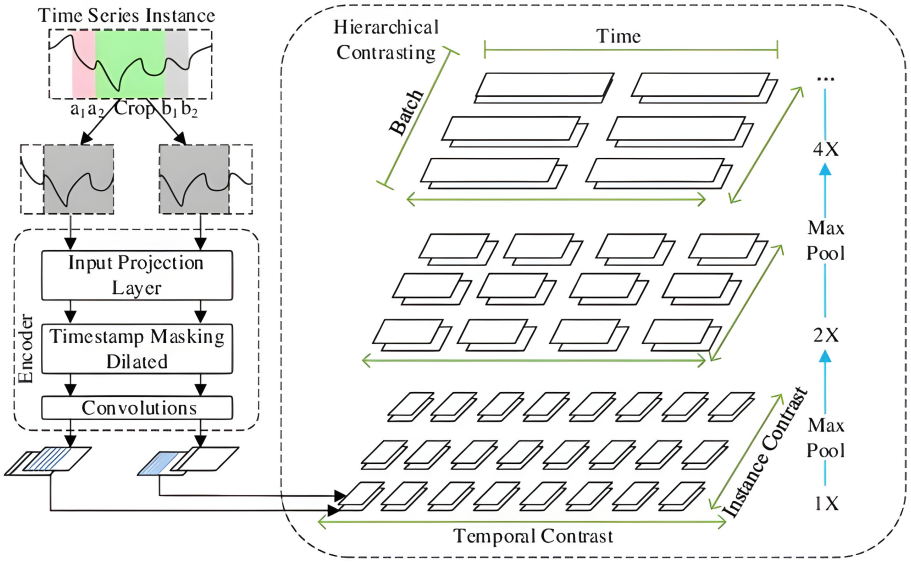


Figure 2.4: The architecture of TS2Vec proposed in Yue et al. [45]. Supports both univariate (as the example input) and multivariate time series. Figure reproduced from [46], CC BY 3.0.

Utilizing this model to encode one-dimensional representations $\mathbf{v}, \hat{\mathbf{v}}$ for the target $\mathbf{Y}$ and predicted series $\hat{\mathbf{Y}}$, the distance (L_2 norm) between the representation vectors can be used as an attack feature:

$$||\mathbf{v} - \hat{\mathbf{v}}||_2 = \sqrt{\sum_{d=1}^{D} (\mathbf{v}_d - \hat{\mathbf{v}}_d)^2} \tag{2.34}$$

where D is the TS2Vec encoding dimension.

2.5 Related Work

As previously mentioned, there are relatively few studies on time series MIAs. There are two papers directly relevant for our study, which will now be briefly summarized.

Firstly, Hisamoto et al. [18] investigates MIA on *sequence-to-sequence* models, using *machine translation* as an example. They highlight the complex nature of recurrent sequence-to-sequence models and their large, complex output space which makes it hard to determine the quality or *confidence* in the outputs; demonstrating how simple attacks based on shadow models show limited transferability—despite their proven success in the context of classification models. Nevertheless, the authors also discuss that there are some cases where sequence-to-sequence models may leak

privacy; for example in the case of overfitting, the model might memorize *rare sequences*. Also, it was found that *out-of-domain data*, which is not well represented in the training set, are easier to attack.

Secondly, Koren et al. [19] conducted the first research on MIAs against time series forecasting models. Their main contribution is introducing two novel attack features tailored to numerical time series, namely *trend* and *seasonality*, as described in Sections 2.4.4 and 2.4.5. They performed empirical evaluation of MIA against a variety of target models using two medical datasets; ECG and EEG. They utilize the *ensemble attack* described in Section 2.3.1 and the results demonstrate a significant performance increase compared to attacks relying solely on loss-based metrics such as MSE or MASE, highlighting the importance of incorporating more sophisticated signals for effective MIAs on time series models.

This study aims to build upon previous work by investigating how existing MIAs can be adapted for time series models, as well as by exploring novel attacks and attack features tailored specifically for time series data. Although Koren et al. obtained significant results, we argue that they do not adequately compare their ensemble attack to other state-of-the-art MIAs. Moreover, their proposed attack features, trend and seasonality, may be more effectively leveraged within more robust attack frameworks such as LiRA and RMIA, as discussed in Section 3.4.1 and 3.4.3.

3

Method

This chapter presents the datasets and preprocessing, model and attack implementations, as well as the training and evaluation procedures; forming the basis for the experimental results shown in Chapter 4.

3.1 Datasets

This section provides an overview of the datasets studied in our experiments, along with the applied preprocessing procedures.

3.1.1 The Temple University Hospital EEG Data Corpus

The *Temple University Hospital EEG Data Corpus* (TUH-EEG) is an archive of 26,846 clinical electroencephalogram (EEG) recordings collected from 1500 individuals between 2002-2017 at the Temple University Hospital [47]. EEG recordings measure electrical activity in the brain, and each channel corresponds to activity measured by a specific electrode on the scalp. The amount of channels vary between the different recordings in the dataset; most commonly there are 31 channels, however, in some cases there are as few as 20. Further, many recordings contain supplementary channels such as ECG and photic stimulation.

Manual inspection and data cleaning was performed in order to ensure a more fair evaluation, given the prescence of many artefacts in the data. Because of the time constraints this was limited to the first 300 individuals in the corpus. The first 60 seconds of each time series were cut, since the start of the recordings often consisted of perfect square waves—likely a calibration sequence or some other sign of the equipment starting up. Also, recordings with clear and frequent artifacts were removed, for example, recordings containing flat signals (dead electrodes). Since most of the patients had multiple recordings from multiple sessions, these were compared in order to determine if the data was likely corrupted, rather than specific to the individual.

For the purpose of forecasting EEG time series, we choose to use the 3-lead channels for each patient, as done in Koren et al. [19]. These are the *FP1*, *FP2* and *F3* channels of the standard 10-20 EEG system, and all correspond to electrodes positioned over the frontal lobe. We randomly selected a subset of 100 individual recordings, each sampled at a frequency of 250 Hz. The time series were truncated

to an identical length, namely the first 15000 time steps, corresponding to 1 minute of each EEG recording. Samples were constructed by applying a standard sliding window, as described in Section 2.1, with a lookback of 100 time steps, and the horizon set to 20 steps. Each feature was normalized using IQR scaling, as described in Section 2.1, and implemented via the *RobustScaler* from the scikit-learn library [27]. The reason IQR was chosen is due to the high inter-subject variability in EEG recordings, where signal can differ substantially between individuals. This method is less sensitive to outliers than alternatives such as min-max scaling, which would compress the majority of values into a very narrow range. Robust scaling was also empirically proven to work best across all evaluation metrics, when evaluating our target models' performance in the original, unscaled units.

3.1.2 Electricity Load Diagrams Dataset

The ElectricityLoadDiagrams20112014 dataset [48], or henceforth referred to as the ELD dataset is a common benchmark dataset used in time series forecasting research. It contains recordings of total electricity load usage in 370 households throughout 2011 to 2014 in Portugal. Electricity usage is given in kilowatts with a time resolution of 15 minutes. Some households are not monitored throughout this whole period and the data for these individuals is zero-padded. Figure 3.1 shows a subset of three households, demonstrating the magnitude of difference in scale between different households.

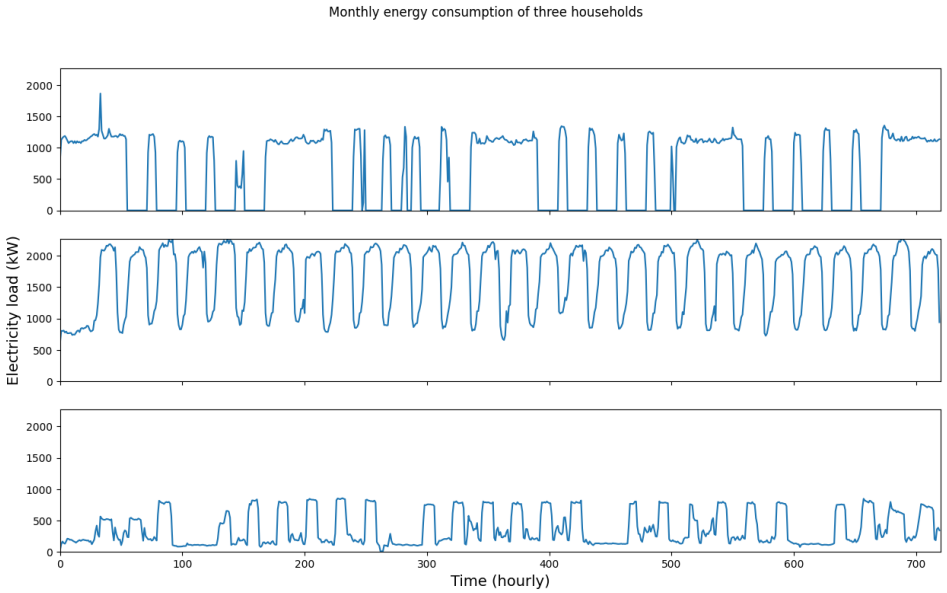


Figure 3.1: A month of electricity usage of three households from the ELD dataset, resampled to 1 hour time resolution.

We removed zero-padding at the start and end of each household time series and resampled the data to hourly consumption by summing four consecutive 15-minute recordings. Households with less than 15000 recorded time steps after removing padding were excluded from our experiments. Additionally any households with a

mean electricity usage below 200 KW/h or above 2000 KW/h were removed, since these outliers proved to significantly affect attack performance. For our experiments a random subset of 100 households were used for our experiments. These time series were truncated to 15000 time steps and rescaled using IQR scaling (for the same reasons as described in Section 3.1.1 for the EEG data). Samples were then created using the sliding window technique described in Section 2.1, using a lookback of 100 and a horizon of 20.

3.1.3 Creating Data Subsets

After preprocessing, the datasets are structured into size $N \times S$ where N is the number of individuals (households for ELD, patients for TUH-EEG), and S is the number of samples created from each individual's time series using the sliding window technique. Individual-based partitioning is then applied to divide the dataset into three subsets: the audit dataset D_{audit} (40%), the auxillary dataset D_{aux} (40%) and the validation dataset D_{val} (20%). This ensures that samples from the same individual never appear across different subsets. Finally, the audit dataset is further divided evenly (50%-50%), again by individual, into training D_{train} and testing D_{test} sets. Figure 3.2 illustrates this partitioning process.

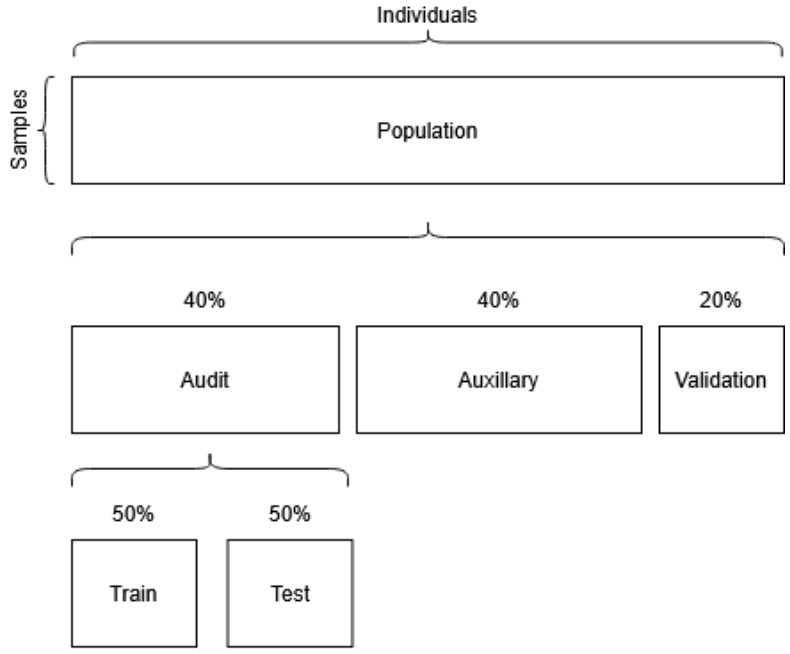


Figure 3.2: Diagram of the individual partitioning of the datasets into audit, auxiliary and validation sets.

The validation set is used to monitor validation loss and perform early stopping during training of both the target and shadow models. The target model is trained on the training set, while membership inference attacks are evaluated using the full audit dataset, comprising both the training set and the test set. The attacks thus aim to distinguish which data points in D_{audit} were part of D_{train} and which were part of D_{test}.

For the online attacks, each shadow model is trained on a randomly selected 50% of the individuals in D_{audit}. This split is independent of the earlier train-test partitioning of D_{audit} into D_{train} and D_{test}. In contrast, for offline attacks, each shadow model is trained on 50% of the individuals from D_{aux}, which is disjoint from the audit dataset.

3.2 Model Implementations

This section briefly describes the implementations of the model architectures described in Section 2.2, including modifications and hyperparameters.

3.2.1 LSTM

The LSTM model (see Section 2.2.1) was easily adapted to different input dimensions; in PyTorch, the amount of variables was simply specified as the *input_size*. We used two unidirectional LSTM layers with 64 hidden units each. For the predictions, we fed the final LSTM hidden states into a single fully connected layer with $H \times M$ neurons. The final output was constructed by just reshaping the FC output to the expected dimensions (*batch_size, H, M*).

3.2.2 N-HiTS

For the N-HiTS model, we used the official implementation by Cristian Challu [49]. Only some minor changes were made to ensure it worked with our data (for example switching the input dimensions and omitting exogenous variables), however the functionality is the same as the original code. We used three stacks of one block, each block consisting of two layers with 512 neurons. Although more advanced basis functions can be used to transform the output of each block (modeling for example trend or seasonality), for simplicity we choose to use only the `IdentityBasis` function which simply splits each block output into the backcast and forecast components. Furthermore, we used the standard hyperparameters; linear interpolation and max pooling mode, Glorot normal initialization, no batch normalization, no dropout, no shared weights, and ReLU activations.

3.3 Training and Evaluation Procedure

To train the models, we used the Adam optimizer [50] with the default learning rate of 0.001, minimizing the MAE loss (see Section 2.4). The max amount of epochs was set to 50, utilizing early stopping by monitoring the validation loss; mitigating overfitting by stopping and restoring best weights when no improvement has been observed over the last three epochs. An overview of the parameters used for dataset preprocessing and training can be found in Table 3.1. Note that this procedure was used for both the target models and shadow models.

In order to evaluate our target models, we look at four metrics, namely MSE, MAE and SMAPE as described in Section 2.4, as well as normalized deviation (ND) as

Parameter		Dataset	
		TUH-EEG	**ELD**
Number of individuals in different subsets	D	100	100
	⌞ D_{aux}	40	40
	⌞ D_{val}	20	20
	⌞ D_{audit}	40	40
	⌞ D_{train}	20	20
	⌞ D_{test}	20	20
Preprocessing	Scaling	IQR	IQR
	Time series dimensionality	3	1
	Lookback	100	100
	Horizon	20	20
	Truncated length	15000	15000
	Resulting samples per individual	14881	14881
Training	Loss function	MAE	MAE
	Optimizer	`Adam`	`Adam`
	Learning rate	0.001	0.001
	Batch size	1024	1024
	Early stopping	True	True
	Patience	3	3

Table 3.1: Overview of preprocessing and training parameters for the two datasets. Notably, the only difference across datasets is time series dimensionality.

given in Equation 3.1. Recall that $Y = [y_1, ..., y_H] \in \mathbb{R}^{H \times M}$ denotes the target time series, and $\hat{Y} = [\hat{y}_1, ..., \hat{y}_H] \in \mathbb{R}^{H \times M}$ the corresponding prediction. MSE and MAE are the most common time series forecasting metrics, while SMAPE and ND measures relative errors in proportion to the actual values. Since we are conducting individual-based partitioning, we argue that the latter two gives a better indication of true overfitting; considering that the scales of the time series can vary a lot between the different individuals.

$$\text{ND}(Y, \hat{Y}) = \frac{\sum_{h=1}^{H} \sum_{m=1}^{M} \left| Y_{h,m} - \hat{Y}_{h,m} \right|}{\sum_{h=1}^{H} \sum_{m=1}^{M} |Y_{h,m}|} \tag{3.1}$$

3.4 Attack Implementations

The attacks were implemented into the LeakPro framework [23]. LiRA and RMIA were already implemented in the framework and were only modified to support time series modality. The Ensemble attack was implemented into the LeakPro framework

according to the description given in Section 2.3.1. Two new attacks, Multi-Signal LiRA and a Deep Time Series Attack, were also implemented into LeakPro, according to the description in Sections 3.4.2 and 3.4.5, respectively.

3.4.1 LiRA

The base implementation of LiRA in LeakPro is based on the description in the original paper [16], alongside the code provided by the authors. The implementation features three different ways to calculate the standard deviation of the parameterized distribution:

- *fixed*: Uses one fixed in and out variance calculated using all samples.
- *carlini*: Uses per-sample variance as described in Section 2.3.1 unless number of shadow models is less than 64 in which it uses a fixed variance.
- *individual carlini*: Uses per-sample variance unless the number of models that would be used to estimate variance (number of in-models when calculating σ_{in} and vice versa) is less than 32 in which case it uses a fixed variance.

We used the *carlini* variance calculation method for our experiments, which ensured per-sample variance in all our experiments.

Since the current implementation only supported classification tasks, we adapted LiRA to be able to use any arbitrary signal from Section 2.4 instead of logit rescaled confidence values. The algorithm for calculating $\text{Score}_{\text{MIA}}(x;\theta)$ is unchanged and involves estimating the in- and out-distributions of any arbitrary signal.

3.4.2 Multi-Signal LiRA

Since we experiment with multiple attack signals, an interesting question is how the performance of LiRA (Section 3.4.1) varies depending on the signal used. This also raises the question of whether certain signals can identify membership information that others cannot. To investigate this, we conducted a correlation analysis of the different signals (see Appendix A). The results suggest that combining signals may be beneficial, which is what we do in the *Multi-Signal* LiRA attack.

The main difference from the original LiRA is that we estimate *multivariate* Gaussians for the in/out members, where each variable corresponds to a signal. This requires computing not only the mean of each signal (as before), but also the *covariance matrix* (rather than just the variance as in the original LiRA). While a mathematically rigorous extension, this approach quickly becomes unfeasible as the number of signals increases, due to the large number of shadow models needed for accurate estimation. To overcome this limitation, we instead assume independence between signals and model them as separate *univariate* Gaussians. Each signal then generates a membership probability, which we aggregate by multiplication to obtain a joint probability. The MIA score computation thus become

$$\text{Score}_{\text{MIA}}(x;\theta) = \prod_{s \in S} \frac{p\left(\phi^s_{\theta_{\text{target}}}(x) \mid \mathcal{N}\left(\mu^s_{\text{in}}, (\sigma^s_{\text{in}})^2\right)\right)}{p\left(\phi^s_{\theta_{\text{target}}}(x) \mid \mathcal{N}\left(\mu^s_{\text{out}}, (\sigma^s_{\text{out}})^2\right)\right)} \tag{3.2}$$

where $\phi^s_{\theta_{\text{target}}}(x)$ denotes the value of signal $s \in S$ computed from the target model's output for input x. The variables μ^s_{in}, μ^s_{out} are the corresponding means for the in/out distributions and σ^s_{in}, σ^s_{out} are the standard deviations. The independence assumption here may appear naive, since we know that the signals are in fact correlated, as illustrated in Appendix A. However for the purpose of our attack, exact likelihoods are not required—what matters is that resulting membership probabilities are *monotonic* to the true underlying distributions. In other words, we aim for scores that preserve the ordering between in- and out-members, even if their absolute values are not mathematically accurate.

3.4.3 RMIA

The base implementation of RMIA for classification models also already exists in LeakPro. To modify RMIA to support time series forecasting we needed to define the likelihood function $\Pr(x|\theta)$ for time series, using any arbitrary signal instead of softmax confidence. Notably we wanted to keep the following properties of $\Pr(x|\theta)$:

1. $\Pr(x|\theta) \in [0, 1]$
2. In-models should generally have a higher likelihood $\Pr(x|\theta)$ on a given datapoint than out-models. In other words a higher $\Pr(x|\theta)$ should indicate a better fit on this datapoint.

Since all signals in Section 2.4 represent some sort of error between target and prediction, we used the following monotonically decreasing mapping:

$$\Pr(x|\theta) = \frac{1}{1 + \text{signal}(x|\theta)} \tag{3.3}$$

where $\text{signal}(x|\theta)$ is the signal value (e.g. MSE) of datapoint x applied to model θ. This mapping makes the likelihood function uphold the desired properties for signals $\in [0, \infty)$. Note that rescaled SMAPE is an unbounded signal and is thus not used in RMIA attacks.

3.4.4 Ensemble Attack

The ensemble attack was implemented into the LeakPro framework according to the description given in Section 2.3.1. We used the same parameters for the attacks as in Koren et al. [19], detailed in Table 3.2. We use mean aggregation within the ensemble, following Shachor et al. [38], and select the best model for each run based on ROC-AUC.

Parameter	Value
num_instances	5
num_runs	3
subset_size	50
num_pairs	9

Table 3.2: Attack parameters used in Ensemble attack.

3.4.5 Deep Time Series Attack

All previously discussed attacks are built on top of various signals; statistics are extracted by comparing signals from outputs of in- respectively out-models, which are then used to classify members. However, finding good attack signals is not a trivial task, and there is always the concern that we may lose information when transforming the raw time series (output and target) into signals. TS2Vec (see Section 2.4.6) explores utilizing deep learning to extract useful signals, however, the method is unsupervised, quite correlated with seasonality and MAE (see Appendix A), and is not optimized for membership inference. One idea is to *learn*, in an *end-to-end* manner, which features are actually useful—motivating our proposed attack: the Deep Time Series (DTS) attack. Unlike previous attacks, DTS does not rely on signal extraction. Instead, it utilizes deep learning to automatically map raw time series to membership labels.

First, a *Membership Inference Classification* (MIC) dataset S_{MIC} is constructed. Each MIC sample consists of the output forecast $\hat{Y}$ from a shadow model θ_i, the corresponding target time series Y, and a binary label indicating whether the sample (X, Y) was seen during training of that shadow model. This is illustrated in Figure 3.3. To construct S_{MIC}, MIC data is collected from all available shadow models by sampling, for each one, a random subset of data points from the audit dataset D_{audit}, using a user-specified fraction $0 < f < 1$. The number of shadow models and f must be balanced to obtain an S_{MIC} dataset of appropriate size.

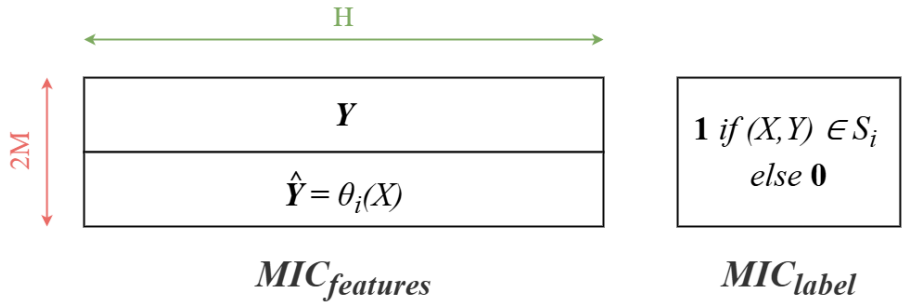


Figure 3.3: Illustration of how a shadow model θ_i and a sample (X, Y) are used to construct a MIC sample $(MIC_{features}, MIC_{label})$. Here, M denotes the dimensionality and H the horizon of the original time series. S_i is the training set of shadow model θ_i, and the binary label MIC_{label} indicates whether (X, Y) was included in this training set.

Then, an MIC model θ_{MIC} is trained on the MIC data, and finally evaluated on the target model outputs on the audit dataset. The MIC model can be any multivariate time series classification model, but it will have a significant effect on the attack's performance, as we will later show in the results (Section 4.2.5). A detailed description of the *online* version of DTS is provided in Algorithm 1, where $\mathcal{T}$ denotes the training procedure for the shadow models, and $\mathcal{T}_{\text{MIC}}$ denotes the training procedure for the MIC model. An offline mode is also supported, with the only difference being that the shadow models are trained solely on the auxiliary dataset, and that the MIC dataset S_{MIC} is constructed using the corresponding forecasts and targets from this dataset. In other words, in the offline setting the MIC model is trained without any knowledge of the audit set.

For the MIC model we choose to experiment with both a simple *LSTM classifier*, and the more advanced *InceptionTime* architecture, as described in Fawaz et al. [51]. A brief description of each MIC model implementation is found below. The classifiers were trained with the Adam optimizer, using early stopping with patience three, and max number of epochs set to 64. Further, we used 64 shadow models and fraction $f = 0.1$ when extracting MIC samples for each shadow model.

LSTM Classifier

As described in Section 2.2.1, the final hidden state of the last LSTM layer can serve as a learned representation for downstream tasks. To construct the LSTM-based membership classifier, we simply connected the final LSTM hidden representation to a single output neuron with sigmoid activation. We utilized two unidirectional LSTM layers, each with 128 hidden units.

InceptionTime

There exists an official implementation of InceptionTime by one of the original authors [52], but it does not support PyTorch and was therefore incompatible with our testbed platform, LeakPro. Consequently, we implemented our own version of InceptionTime, aiming to remain as faithful as possible to the original design while introducing a few modifications to better suit our task.

While the original implementation uses an ensemble of five models, we opted to use a single predictor due to time and resource constraints. Additionally, we adjusted the inception block kernel sizes to better fit our MIC data. Specifically, since our MIC time series are relatively short (as the horizon in the target data set), we employed kernel sizes 2, 4, 8, instead of the original 10, 20, 40. This modification is consistent with findings from the InceptionTime paper, where the smaller kernel size set (2, 4, 8) was shown to outperform larger kernels for time series shorter than 81 time steps [51]. Finally, since membership inference is a binary classification task, a sigmoid output activation was used instead of softmax.

Algorithm 1 Deep Time Series Attack *online*

Input: Target model θ_{target}, audit dataset D_{audit}

$\Theta \leftarrow \mathcal{T}(D_{\text{audit}})$ ▷ Train shadow models on audit data
$S_{\text{MIC}} \leftarrow \emptyset$ ▷ Initialize empty MIC dataset

for θ_i in Θ **do**
    $s_i \sim D_{\text{audit}}$ ▷ Sample $s_i \subset D_{\text{audit}}$, where $|s_i| = f \cdot |D_{\text{audit}}|$ for $0 < f < 1$
    for (X, Y) in s_i **do**
        $\hat{y} \leftarrow \theta_i(X)$ ▷ Get shadow model forecast
        $X_{\text{MIC}} \leftarrow \text{stack}(y, \hat{y})$ ▷ Construct MIC input features
        $y_{\text{MIC}} \leftarrow \mathcal{T}$.membership_labels ▷ Assign 1 if (X, Y) was used to train θ_i, else 0
        $S_{\text{MIC}} \leftarrow S_{\text{MIC}} \cup \{(X_{\text{MIC}}, y_{\text{MIC}})\}$
    end for
end for

function `MIC_TARGET_FEATURES`(X, Y)
    $\hat{y} \leftarrow \theta_{\text{target}}(X)$ ▷ Get target model forecast
    return $\text{stack}(y, \hat{y})$ ▷ Return target MIC features
end function

$\theta_{\text{MIC}} \leftarrow \mathcal{T}_{\text{MIC}}(S_{\text{MIC}})$ ▷ Train MIC model on S_{MIC}
for (X, Y) in D_{audit} **do** ▷ Compute membership scores
    $\text{Score}_{\text{MIA}}((X, Y), \theta_{\text{target}}) \leftarrow \theta_{\text{MIC}}(\texttt{MIC\_target\_features}(X, Y))$
end for

3.5 Attack Evaluation

All previously described attacks returns a list of *sample-level* MIA scores, that is, an approximated probability of target model training set membership for each sample in the audit dataset. To evaluate an attack, as described in Section 2.3.2, we varied the decision threshold to create a ROC curve and observed the true positive rate at different fixed false positive rates. As proposed by Carlini et al. [16], primarily the low FPR regime is of interest from the privacy perspective, confidently identifying even a few number of members is more impactful than making average-case inferences. Therefore we report only the TPRs at fixed FPRs in Chapter 4.

In the context of training on data from multiple individuals (recall the individual-based data partitioning described in Section 3.1.3), determining whether a particular individual's data was included in the training set of a machine learning model is of particular interest. This is known as *user-level* membership inference, as described in Section 2.3, distinct from the sample-level membership inference previously discussed.

To accomplish this task, the sample-level MIA scores were aggregated across all sam-

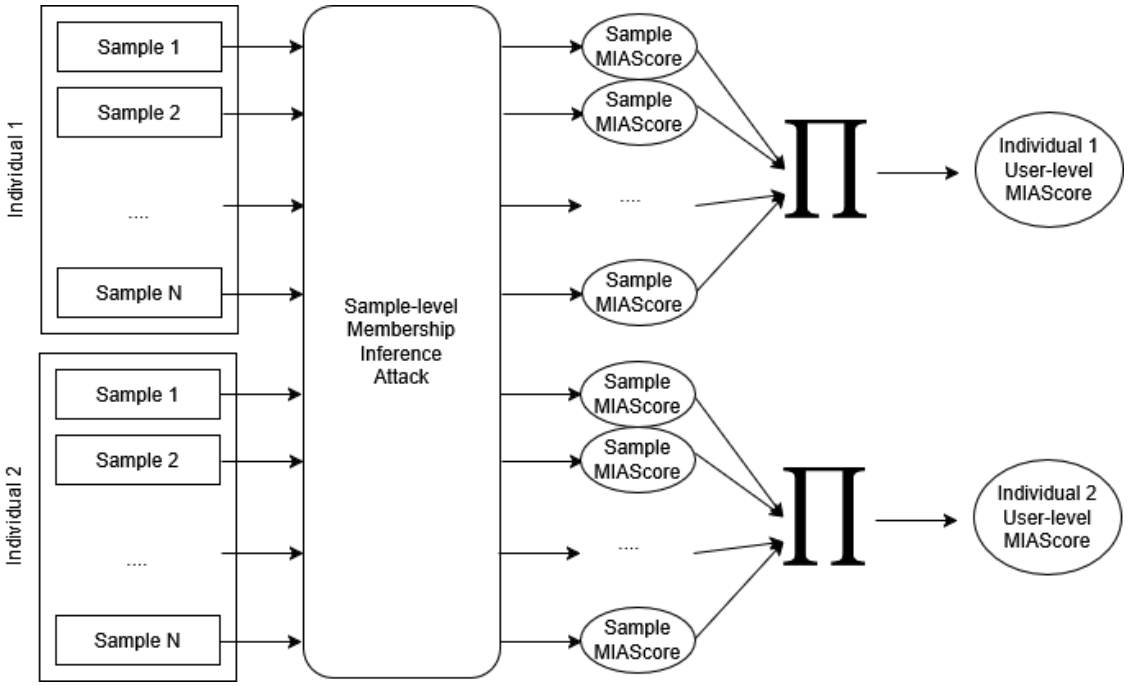


Figure 3.4: Illustration of user-level membership inference, where the user-level $\text{Score}_{\text{MIA}}$ is aggregated across multiple samples belonging to an individual.

ples belonging to each individual to estimate whether that individual was a member of the training set. Several aggregation strategies can be considered—for example, taking the mean or median of the scores, or more advanced techniques such as focusing on outlier samples with unusually high or low scores. For simplicity, and after some empirical experiments, we choose to proceed with simply taking the product of the sample-level probabilities. This corresponds to the individual membership likelihood under the (naive) assumption that sample membership predictions are independent, and was computed in practice by summing the *logged* sample-level probabilities. This aggregation is formalized below:

$$\text{Score}_{\text{MIA}}(X, \theta_{target}) = \prod_{x \in X} \text{Score}_{\text{MIA}}(x, \theta_{target}) \tag{3.4}$$

where the goal is to determine a $\text{Score}_{\text{MIA}}$ for an individual $X = [x_0, x_1, \dots]$ consisting of multiple samples, under the assumption of independence. As with the sample-level attacks, we focus on the low false positive rate regime; due to the limited number of individuals, the true positive rate at 0.0% FPR is used as the primary evaluation metric. An overview of this user-level membership inference process is illustrated in Figure 3.4.

4

Results

This chapter covers the experimental results of training target models and performing different attacks with varying attack parameters. Further, the chapter includes results from various ablation studies, where additional attack or dataset parameters are varied while the attack type is fixed.

The entire data preprocessing, partitioning, training stage, and attacks were run five times using different random seeds to reduce the effect of randomness. Thus, all results show both mean and standard deviation of each metric.

4.1 Target Models

The N-HiTS and LSTM models were trained on both datasets using the training parameters outlined in Table 3.1. Their performance, evaluated using four metrics—MSE, MAE, SMAPE, and ND—is summarized in Tables 4.1 and 4.2.

Across both datasets and all data splits (Train, Validation, and Test), N-HiTS consistently outperforms LSTM on every metric. The performance gaps between the training, validation, and test sets appear to be similar for both models across all metrics.

All reported results are averaged over five runs, accounting for variability that arises not only from random initialization but also from differences in which individuals were included in each dataset partition. Notably, MSE and MAE exhibit greater variance across these runs, making them less reliable for direct comparison. In contrast, the scale-invariant metrics—SMAPE and ND—provide a more stable and meaningful basis for evaluation, especially given the differing scales between the two datasets.

Model	Subset	Metrics			
		MSE	MAE	SMAPE	ND
LSTM	Train	6.68×10^{-10} $\pm4.46\times10^{-10}$	1.28×10^{-5} $\pm2.77\times10^{-6}$	0.247 ±0.0285	0.198 ±0.0323
	Val	4.42×10^{-9} $\pm4.87\times10^{-9}$	2.05×10^{-5} $\pm5.29\times10^{-6}$	0.271 ±0.0186	0.296 ±0.059
	Test	4.15×10^{-9} $\pm2.2\times10^{-9}$	2.3×10^{-5} $\pm4.45\times10^{-6}$	0.285 ±0.0111	0.319 ±0.00633
N-HiTS	Train	2.6×10^{-10} $\pm1.47\times10^{-10}$	8.92×10^{-6} $\pm1.43\times10^{-6}$	0.208 ±0.0301	0.139 ±0.0196
	Val	6.97×10^{-10} $\pm3.04\times10^{-10}$	1.49×10^{-5} $\pm2.45\times10^{-6}$	0.258 ±0.0195	0.216 ±0.0294
	Test	8.43×10^{-10} $\pm5.1\times10^{-10}$	1.6×10^{-5} $\pm2.46\times10^{-6}$	0.267 ±0.00476	0.223 ±0.0201

Table 4.1: Performance of the target models on the TUH-EEG dataset. We report the mean ± standard deviation of each metric across the five different random seeds. Results are presented using unscaled metrics for the Train, Validation, and Test subsets, rounded to three significant figures.

Model	Subset	Metrics			
		MSE	MAE	SMAPE	ND
LSTM	Train	1.47×10^{4} $\pm6\times10^{3}$	65.8 ±12.9	0.0607 ±0.0178	0.0984 ±0.0177
	Val	2.77×10^{4} $\pm1.9\times10^{4}$	88 ±25	0.0668 ±0.0226	0.116 ±0.0315
	Test	1.85×10^{4} $\pm6.65\times10^{3}$	73.8 ±6.9	0.0754 ±0.00558	0.123 ±0.00891
N-HiTS	Train	7.92×10^{3} $\pm2.8\times10^{3}$	49.1 ±9.6	0.0471 ±0.0164	0.0731 ±0.0126
	Val	1.95×10^{4} $\pm1.29\times10^{4}$	69.5 ±19.6	0.0556 ±0.025	0.0926 ±0.0306
	Test	1.37×10^{4} $\pm3.45\times10^{3}$	61.1 ±10.9	0.0645 ±0.00883	0.102 ±0.0143

Table 4.2: Performance of the target models on the ELD dataset. We report the mean ± standard deviation of each metric across the five different random seeds. Results are presented using unscaled metrics for the Train, Validation, and Test subsets, rounded to three significant figures.

4.2 Attack Results

This section presents and analyzes the performance of five attacks: LiRA, Multi-signal LiRA, RMIA, the Ensemble attack, and DTS. We report both sample-level and user-level TPRs at various low fixed FPRs. Sample-level ROC curves are presented in Appendix B, with further evaluation metrics available in Appendix C.

4.2.1 LiRA

We conducted multiple instances of the LiRA attack, utilizing each signal presented in Section 2.4. These attacks were executed across all datasets and target models, in both the online and offline settings. For all attacks, we employed 64 shadow models, each trained to match the target model in both architecture and training procedure.

The results of the online and offline attacks using each signal are presented in Tables 4.3 and 4.4. Each table reports the sample-level true positive rate at fixed low false positive rates—specifically 0.1% and 0.01%—as well as the user-level TPR at 0% FPR. Results are averaged across five runs, displaying both mean and standard deviation of the TPR.

Across all these evaluations, several consistent patterns emerge. First, the N-HiTS model is substantially more vulnerable to membership inference attacks than the LSTM model, regardless of dataset, attack setting, or evaluation metric. Second, the TUH-EEG dataset proves significantly easier to attack than the ELD dataset under both sample-level and user-level, and in both online and offline attack settings.

When focusing on signal performance in the online setting, TS2Vec consistently achieves the highest TPR at fixed FPRs, outperforming other signals across most configurations, though it falls slightly behind in a few cases. In contrast, signal performance in the offline setting is less consistent, but Rescaled SMAPE tends to show relatively strong results.

An especially notable result is observed in the online LiRA attacks against the N-HiTS model on the TUH-EEG dataset: all signals except Trend achieve 100% user-level TPR at 0% FPR across all five random seeds.

	Model	N-HiTS			LSTM		
		Sample-level		User-level	Sample-level		User-level
	Signal	0.1% FPR	0.01% FPR	0% FPR	0.1% FPR	0.01% FPR	0% FPR
Online	MSE	4.02 ±2.81	0.19 ±0.24	**100.0** ±0.00	3.05 ±2.35	2.27 ±2.03	22.00 ±10.77
	MAE	6.73 ±2.60	1.35 ±1.79	**100.0** ±0.00	3.43 ±2.47	2.58 ±2.19	28.00 ±15.03
	SMAPE	5.04 ±2.20	1.11 ±1.10	**100.0** ±0.00	3.19 ±2.37	1.93 ±2.08	34.00 ±12.41
	Rescaled SMAPE	6.65 ±2.39	2.26 ±1.83	**100.0** ±0.00	3.50 ±2.40	2.23 ±2.13	**35.00** ±13.42
	Trend	5.19 ±4.12	2.42 ±2.05	94.00 ±4.90	2.26 ±2.51	1.99 ±2.39	15.00 ±5.48
	Seasonality	7.28 ±2.50	1.77 ±2.12	**100.0** ±0.00	3.42 ±2.41	**2.61** ±2.13	23.00 ±12.08
	TS2Vec	**7.41** ±0.97	**2.64** ±1.49	**100.0** ±0.00	**3.57** ±2.19	2.57 ±2.19	31.00 ±9.70
Offline	MSE	0.01 ±0.02	0.00 ±0.00	2.00 ±2.45	1.66 ±2.16	1.35 ±2.13	4.00 ±4.90
	MAE	0.07 ±0.10	0.00 ±0.00	6.00 ±8.00	1.97 ±2.35	**1.80** ±2.29	6.00 ±3.74
	SMAPE	0.20 ±0.11	0.05 ±0.03	16.00 ±5.83	1.88 ±2.25	1.54 ±1.90	**8.00** ±6.00
	Rescaled SMAPE	**1.04** ±0.93	0.12 ±0.08	**22.00** ±9.80	**2.12** ±2.38	1.69 ±1.98	**8.00** ±4.00
	Trend	0.55 ±0.57	**0.13** ±0.14	12.00 ±8.72	1.84 ±2.14	1.46 ±1.98	3.00 ±4.00
	Seasonality	0.08 ±0.11	0.01 ±0.01	10.00 ±7.07	1.97 ±2.34	1.71 ±2.22	6.00 ±3.74
	TS2Vec	0.33 ±0.34	0.05 ±0.03	8.00 ±6.78	1.90 ±2.24	1.71 ±2.22	5.00 ±3.16

Table 4.3: LiRA attack performance on the TUH-EEG dataset. For each signal, we report the mean ± standard deviation of the sample-level true positive rate (TPR %, rounded to two decimals) at false positive rates of 0.1 % and 0.01 %, as well as the user-level TPR at 0 % FPR. Results are shown for both N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing signal is highlighted in **bold**.

	Model	N-HiTS			LSTM		
		Sample-level		User-level	Sample-level		User-level
	Signal	0.1% FPR	0.01% FPR	0% FPR	0.1% FPR	0.01% FPR	0% FPR
Online	MSE	1.06 ±0.57	0.28 ±0.21	43.00 ±28.21	2.26 ±3.18	0.84 ±1.19	29.00 ±20.10
	MAE	1.99 ±1.00	0.65 ±0.36	49.00 ±26.72	3.29 ±2.64	1.47 ±0.88	35.00 ±15.49
	SMAPE	1.50 ±0.61	0.46 ±0.23	43.00 ±26.76	2.18 ±1.46	0.78 ±0.29	29.00 ±11.58
	Rescaled SMAPE	1.94 ±0.79	0.59 ±0.26	**53.00** ±29.43	3.10 ±1.75	0.93 ±0.51	**38.00** ±12.08
	Trend	1.34 ±0.29	0.45 ±0.20	50.00 ±19.49	1.17 ±0.72	0.31 ±0.32	27.00 ±19.65
	Seasonality	2.02 ±0.85	0.76 ±0.37	49.00 ±26.72	3.46 ±3.07	1.43 ±1.14	34.00 ±13.56
	TS2Vec	**2.16** ±1.08	**0.88** ±0.45	48.00 ±25.42	**3.50** ±2.86	**1.68** ±1.25	33.00 ±12.08
Offline	MSE	0.07 ±0.07	0.01 ±0.01	4.00 ±5.83	0.03 ±0.03	0.00 ±0.00	3.00 ±4.00
	MAE	0.26 ±0.17	0.07 ±0.07	9.00 ±7.35	0.21 ±0.24	0.02 ±0.02	7.00 ±4.00
	SMAPE	0.35 ±0.23	0.09 ±0.06	10.00 ±7.07	0.18 ±0.14	0.05 ±0.06	8.00 ±7.48
	Rescaled SMAPE	**0.59** ±0.27	0.13 ±0.10	**12.00** ±9.27	0.34 ±0.26	0.07 ±0.04	**9.00** ±4.90
	Trend	0.22 ±0.10	0.05 ±0.03	6.00 ±3.74	0.06 ±0.08	0.01 ±0.02	5.00 ±4.47
	Seasonality	0.30 ±0.18	0.08 ±0.09	10.00 ±8.37	0.24 ±0.18	0.07 ±0.09	8.00 ±5.10
	TS2Vec	0.52 ±0.36	**0.26** ±0.23	9.00 ±5.83	**0.41** ±0.32	**0.10** ±0.06	7.00 ±5.10

Table 4.4: LiRA attack performance on the ELD dataset. For each signal, we report the mean ± standard deviation of the sample-level true positive rate (TPR %, rounded to two decimals) at false positive rates of 0.1 % and 0.01 %, as well as the user-level TPR at 0 % FPR. Results are shown for both N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing signal is highlighted in **bold**.

4.2.2 Multi-Signal LiRA

The Multi-Signal LiRA attack—utilizing the signals MSE, MAE, Rescaled SMAPE, Trend, Seasonality and TS2Vec—was executed across all datasets and target models in both online and offline setting utilizing 64 shadow models. The fixed FPR results are given in Tables 4.5 and 4.6.

In the online attack setting, Multi-Signal LiRA generally outperforms LiRA variants using individual signals, particularly in terms of sample-level TPR. However, this trend does not extend to the offline setting, where Multi-Signal LiRA often underperforms compared to individual signals such as Rescaled SMAPE or TS2Vec.

	N-HiTS			LSTM		
	Sample-level		User-level	Sample-level		User-level
	0.1% FPR	0.01% FPR	0% FPR	0.1% FPR	0.01% FPR	0% FPR
Online	9.11 ±2.86	2.71 ±2.88	100.00 ±0.00	3.79 ±2.51	2.99 ±2.35	32.00 ±8.72
Offline	0.12 ±0.16	0.01 ±0.00	10.00 ±8.94	2.12 ±2.46	1.88 ±2.35	5.00 ±3.16

Table 4.5: Multi-Signal LiRA attack performance on the EEG dataset. We report the mean ± standard deviation of the sample-level true positive rate (in %, rounded to two decimals) at false positive rates of 0.1 % and 0.01 %, as well as the user-level TPR at 0 % FPR. Results are shown for both N-HiTS and LSTM target models, in both online and offline modes.

	N-HiTS			LSTM		
	Sample-level		User-level	Sample-level		User-level
	0.1% FPR	0.01% FPR	0% FPR	0.1% FPR	0.01% FPR	0% FPR
Online	2.32 ±1.01	0.85 ±0.41	48.00 ±26.76	4.15 ±4.13	2.29 ±2.27	34.00 ±12.81
Offline	0.32 ±0.20	0.08 ±0.08	10.00 ±6.32	0.37 ±0.42	0.06 ±0.06	8.00 ±4.00

Table 4.6: Multi-Signal LiRA attack performance on the ELD dataset. We report the mean ± standard deviation of the sample-level true positive rate (in %, rounded to two decimals) at false positive rates of 0.1 % and 0.01 %, as well as the user-level TPR at 0 % FPR. Results are shown for both N-HiTS and LSTM target models, in both online and offline modes.

4.2.3 RMIA

Multiple instances of the RMIA attack was executed utilizing the signals: MSE, MAE, SMAPE, Trend, Seasonality, and TS2Vec. The attacks were executed in both the online and offline setting, for both datasets and target models. For each of these attacks we used 64 shadow models, a γ value of 1.0, and $a = \frac{1}{3}$ for offline attacks. The results of the attacks are presented in Tables 4.7 and 4.8.

On the TUH-EEG dataset, RMIA appears to generally outperform LiRA in user-level membership inference but performs significantly worse at the sample level. In contrast, results on the ELD dataset are more mixed: RMIA occasionally surpasses LiRA in user-level performance but not consistently. For sample-level attacks, RMIA performs worse than LiRA in the online setting, yet shows improved performance in the offline setting. Across both models and datasets, no single signal consistently outperforms the others, with the best-performing signal varying depending on the specific attack setting and evaluation level.

	Model	N-HiTS			LSTM		
		Sample-level		User-level	Sample-level		User-level
	Signal	0.1% FPR	0.01% FPR	0% FPR	0.1% FPR	0.01% FPR	0% FPR
Online	MSE	3.28 ±2.02	1.38 ±1.98	**100.0** ±0.00	1.90 ±1.92	0.16 ±0.24	32.00 ±26.94
	MAE	**3.40** ±1.91	**1.48** ±1.93	**100.0** ±0.00	**2.02** ±2.12	0.36 ±0.37	**41.00** ±25.38
	SMAPE	1.70 ±1.57	0.46 ±0.53	**100.0** ±0.00	1.40 ±2.27	1.06 ±2.11	36.00 ±23.54
	Trend	1.01 ±1.43	0.38 ±0.68	58.00 ±21.12	1.19 ±1.90	0.58 ±1.11	10.00 ±7.07
	Seasonality	2.08 ±1.56	0.74 ±1.09	**100.0** ±0.00	1.49 ±1.84	0.56 ±0.90	30.00 ±21.68
	TS2Vec	2.61 ±1.80	1.31 ±1.98	**100.0** ±0.00	1.84 ±2.30	**1.28** ±2.23	37.00 ±23.37
Offline	MSE	1.24 ±2.25	0.00 ±0.00	**42.00** ±30.59	1.51 ±1.24	0.66 ±1.32	17.00 ±14.35
	MAE	1.10 ±2.06	**1.04** ±2.09	40.00 ±29.50	1.67 ±1.17	0.60 ±1.21	15.00 ±10.49
	SMAPE	2.20 ±1.58	0.00 ±0.00	27.00 ±18.60	1.24 ±1.39	0.01 ±0.01	12.00 ±6.78
	Trend	0.46 ±0.28	0.03 ±0.05	9.00 ±4.90	0.38 ±0.41	0.02 ±0.03	10.00 ±5.48
	Seasonality	**2.37** ±2.19	0.97 ±1.45	14.00 ±10.68	1.87 ±1.90	**0.73** ±1.39	8.00 ±8.12
	TS2Vec	0.29 ±0.59	0.00 ±0.00	41.00 ±25.38	**1.94** ±1.27	0.00 ±0.00	**19.00** ±12.41

Table 4.7: RMIA attack performance on the TUH-EEG dataset. For each signal, we report the mean ± standard deviation of the sample-level true positive rate (in %, rounded to two decimals) at false positive rates of 0.1 % and 0.01 %, as well as the user-level TPR at 0 % FPR. Results are shown for both N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing signal is highlighted in **bold**.

	Model	N-HiTS			LSTM		
		Sample-level		User-level	Sample-level		User-level
	Signal	0.1% FPR	0.01% FPR	0% FPR	0.1% FPR	0.01% FPR	0% FPR
Online	MSE	1.12 ±1.26	**0.31** ±0.37	44.00 ±20.35	**1.16** ±0.58	**0.15** ±0.14	**19.00** ±17.44
	MAE	**1.19** ±1.20	0.26 ±0.40	59.00 ±19.60	1.00 ±0.60	0.13 ±0.15	16.00 ±12.41
	SMAPE	0.67 ±0.44	0.09 ±0.12	47.00 ±21.59	0.24 ±0.21	0.04 ±0.05	16.00 ±11.58
	Trend	0.17 ±0.12	0.02 ±0.04	21.00 ±7.35	0.09 ±0.06	0.01 ±0.01	13.00 ±11.66
	Seasonality	0.97 ±0.69	0.04 ±0.05	**73.00** ±19.65	0.46 ±0.37	0.08 ±0.10	**19.00** ±14.97
	TS2Vec	0.79 ±0.50	0.03 ±0.05	64.00 ±30.23	0.40 ±0.47	0.10 ±0.19	18.00 ±11.66
Offline	MSE	0.06 ±0.13	0.00 ±0.00	7.00 ±6.00	1.28 ±1.33	0.26 ±0.23	6.00 ±5.83
	MAE	0.50 ±0.66	0.00 ±0.01	**8.00** ±6.78	**1.29** ±1.44	0.03 ±0.06	5.00 ±6.32
	SMAPE	0.02 ±0.04	0.00 ±0.00	**8.00** ±2.45	0.61 ±0.45	0.00 ±0.00	4.00 ±4.90
	Trend	0.34 ±0.15	0.03 ±0.06	6.00 ±5.83	0.16 ±0.09	0.02 ±0.03	6.00 ±5.83
	Seasonality	**0.89** ±0.86	**0.31** ±0.50	7.00 ±6.00	0.68 ±1.02	**0.42** ±0.84	5.00 ±6.32
	TS2Vec	0.61 ±0.90	0.03 ±0.04	7.00 ±5.10	0.77 ±1.03	0.36 ±0.73	**7.00** ±7.48

Table 4.8: RMIA attack performance on the ELD dataset. For each signal, we report the mean ± standard deviation of the sample-level true positive rate (in %, rounded to two decimals) at false positive rates of 0.1 % and 0.01 %, as well as the user-level TPR at 0 % FPR. Results are shown for both N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing signal is highlighted in **bold**.

4.2.4 Ensemble Attack

Multiple instances of the Ensemble attack was run, utilizing each of the signals in Section 2.4. The attacks were run in only audit mode, using the attack parameters defined in Section 2.3.1. The results are given in Tables 4.9 and 4.10.

Since the ensemble attack is executed in audit mode—comparable to the online setting but assuming a stronger adversary—a fair comparison should be made against the performance of the online attacks. Under this comparison, the ensemble attack consistently and significantly underperforms relative to the aforementioned online attacks, for both user-level and sample-level membership inference.

Model	N-HiTS			LSTM		
	Sample-level		User-level	Sample-level		User-level
Signal	0.1% FPR	0.01% FPR	0% FPR	0.1% FPR	0.01% FPR	0% FPR
MSE	0.00 ±0.00	0.00 ±0.00	6.00 ±7.35	0.00 ±0.00	0.00 ±0.00	2.00 ±2.45
MAE	0.00 ±0.00	0.00 ±0.00	8.00 ±8.12	0.00 ±0.00	0.00 ±0.00	7.00 ±5.10
SMAPE	0.00 ±0.00	0.00 ±0.00	6.00 ±2.00	0.00 ±0.00	0.00 ±0.00	5.00 ±5.48
Rescaled SMAPE	0.00 ±0.00	0.00 ±0.00	8.00 ±4.00	0.00 ±0.00	0.00 ±0.00	**10.00** ±7.07
Trend	**0.03** ±0.05	0.00 ±0.00	4.00 ±5.83	**0.01** ±0.01	0.00 ±0.00	5.00 ±4.47
Seasonality	**0.03** ±0.03	0.00 ±0.00	9.00 ±8.00	0.00 ±0.00	0.00 ±0.00	4.00 ±4.90
TS2Vec	0.00 ±0.00	0.00 ±0.00	**15.00** ±14.83	0.00 ±0.00	0.00 ±0.00	1.00 ±2.00

Table 4.9: Ensemble attack performance on the TUH-EEG dataset. For each signal, we report the mean ± standard deviation of the sample-level true positive rate (in %, rounded to two decimals) at false positive rates of 0.1 % and 0.01 %, as well as the user-level TPR at 0 % FPR. Results are shown for both N-HiTS and LSTM target models. In each column, the best-performing signal is highlighted in **bold**.

Model	N-HiTS			LSTM		
	Sample-level		User-level	Sample-level		User-level
Signal	0.1% FPR	0.01% FPR	0% FPR	0.1% FPR	0.01% FPR	0% FPR
MSE	0.01 ±0.02	0.00 ±0.00	8.00 ±6.78	0.00 ±0.00	0.00 ±0.00	4.00 ±4.90
MAE	0.00 ±0.00	0.00 ±0.00	6.00 ±7.35	0.00 ±0.00	0.00 ±0.00	5.00 ±5.48
SMAPE	**0.02** ±0.02	0.00 ±0.00	2.00 ±4.00	**0.02** ±0.05	0.00 ±0.00	3.00 ±6.00
Rescaled SMAPE	0.00 ±0.00	0.00 ±0.00	3.00 ±4.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
Trend	0.01 ±0.02	0.00 ±0.00	5.00 ±4.47	0.00 ±0.00	0.00 ±0.00	2.00 ±2.45
Seasonality	0.00 ±0.00	0.00 ±0.00	**9.00** ±9.70	0.00 ±0.00	0.00 ±0.00	**6.00** ±3.74
TS2Vec	0.00 ±0.00	0.00 ±0.00	2.00 ±4.00	0.00 ±0.00	0.00 ±0.00	3.00 ±4.00

Table 4.10: Ensemble attack performance on the ELD dataset. For each signal, we report the mean ± standard deviation of the sample-level true positive rate (in %, rounded to two decimals) at false positive rates of 0.1 % and 0.01 %, as well as the user-level TPR at 0 % FPR. Results are shown for both N-HiTS and LSTM target models. In each column, the best-performing signal is highlighted in **bold**.

4.2.5 Deep Time Series Attack

The DTS attack was run across all datasets and target models in both the online and offline setting, utilizing 64 shadow models and attack parameters stated in Section 3.4.5. Furthermore, each setting was evaluated under two MIC architectures: the LSTM Classifier and the InceptionTime model, as described in Section 3.4.5. The fixed FPR results are given in Tables 4.11 and 4.12.

The results indicate that the optimal MIC model depends strongly on both the dataset and the target model, making it difficult to draw any general conclusions. However, it is worth noting that, despite comparable or slightly lower performance in certain scenarios, the LSTM Classifier is significantly faster to train due to its lower architectural complexity. Compared to the other attacks, DTS consistently performs well, achieving the best sample-level MIA results in all online settings except for TUH-EEG with the LSTM target model. In contrast, when evaluated in the offline setting, DTS performs worse and is outperformed by RMIA, although it still achieves reasonable results.

	Model	N-HiTS			LSTM		
		Sample-level		User-level	Sample-level		User-level
	MIC model	0.1% FPR	0.01% FPR	0% FPR	0.1% FPR	0.01% FPR	0% FPR
Online	LSTM Classifier	**23.32** ±6.65	15.62 ±7.22	**96.00** ±4.90	**2.97** ±3.75	**2.15** ±2.69	**30.00** ±23.45
	InceptionTime	22.39 ±7.80	**15.81** ±7.35	90.00 ±6.32	2.49 ±4.99	1.70 ±3.39	20.00 ±19.24
Offline	LSTM Classifier	**2.09** ±3.13	**0.67** ±1.01	**27.00** ±16.61	0.17 ±0.16	0.00 ±0.01	**9.00** ±15.62
	InceptionTime	1.12 ±1.45	0.64 ±1.28	22.00 ±23.79	**0.23** ±0.29	**0.05** ±0.10	5.00 ±7.75

Table 4.11: DTS attack performance on the TUH-EEG dataset. For each MIC classifier model report the mean ± standard deviation of the sample-level true positive rate (in %, rounded to two decimals) at false positive rates of 0.1 % and 0.01 %, as well as the user-level TPR at 0 % FPR. Results are shown for both N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing MIC classifier is highlighted in **bold**.

	Model	N-HiTS			LSTM		
		Sample-level		User-level	Sample-level		User-level
	MIC model	0.1% FPR	0.01% FPR	0% FPR	0.1% FPR	0.01% FPR	0% FPR
Online	LSTM Classifier	6.34 ±3.91	2.77 ±1.87	56.00 ±22.45	**4.78** ±4.56	**2.57** ±2.42	**34.00** ±21.54
	InceptionTime	**6.66** ±3.48	**2.94** ±1.69	**65.00** ±21.21	4.49 ±4.83	2.40 ±3.20	24.00 ±25.77
Offline	LSTM Classifier	0.57 ±0.39	0.15 ±0.18	**13.00** ±12.88	0.76 ±0.85	0.26 ±0.39	**9.00** ±5.83
	InceptionTime	**0.68** ±0.71	**0.21** ±0.26	11.00 ±13.19	**0.78** ±0.86	**0.30** ±0.47	6.00 ±4.90

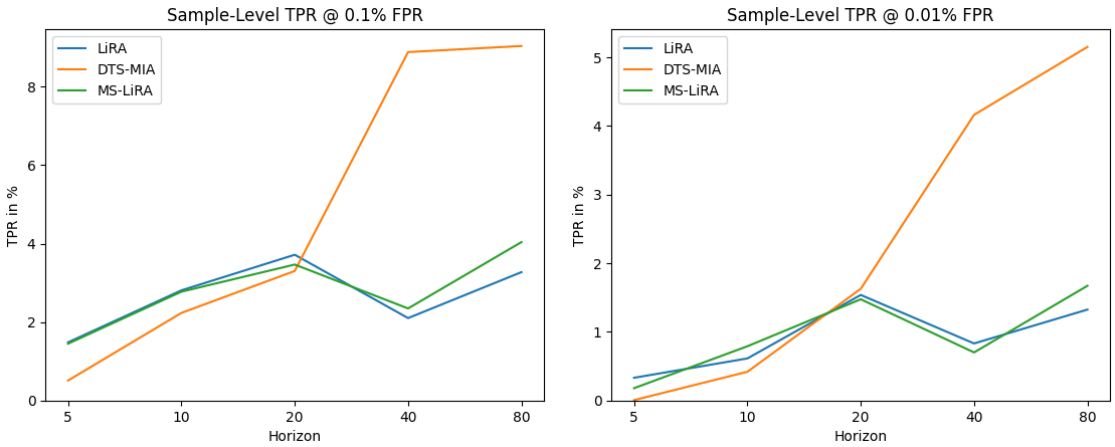
Table 4.12: DTS attack performance on the ELD dataset. For each MIC classifier model report the mean ± standard deviation of the sample-level true positive rate (in %, rounded to two decimals) at false positive rates of 0.1 % and 0.01 %, as well as the user-level TPR at 0 % FPR. Results are shown for both N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing MIC classifier is highlighted in **bold**.

4.3 Ablation Studies

In addition to the previous presented results, we also conducted two ablation studies, investigating the effect of varying the *horizon* and *number of individuals*. Here, due to the time constraints, we only consider the *N-HiTS* model trained on the *ELD* data. Further, we limit the experiments to a subset of the attacks, namely *LiRA* with the TS2Vec signal (the best signal for this setting), *Multi-Signal LiRA* utilizing the signals MSE, MAE, Rescaled SMAPE, Trend, Seasonality and TS2Vec (as in Section 4.2.2), and *DTS* with InceptionTime (the best MIC model for this setting). All attacks were executed in the online setting.

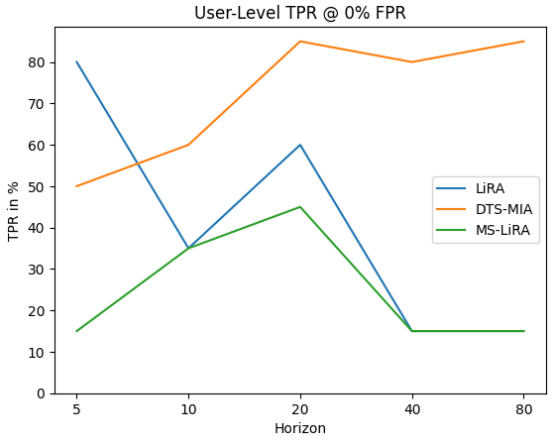
4.3.1 Varying Horizon

To investigate the impact of the forecasting horizon, we kept all other variables fixed while only varying the horizon used to construct the dataset. Although only one random seed was used, all attack runs were evaluated with the same partitioning of individuals, reducing the interference of randomness. We evaluated horizons of 5, 10, 20, 40, and 80 time steps, and report the sample-level and user-level results in Figure 4.1. From these results, a clear positive trend can be seen with the increase of horizon length, indicating a higher risk of data leakage for larger horizons in the sample-level setting. In contrast, the user-level results are more divergent: while DTS shows a clearly increasing TPR with longer horizons, the performance of the LiRA attacks declines beyond a horizon of 20.



(a) Sample-level TPR at 0.1% FPR

(b) Sample-level TPR at 0.01% FPR

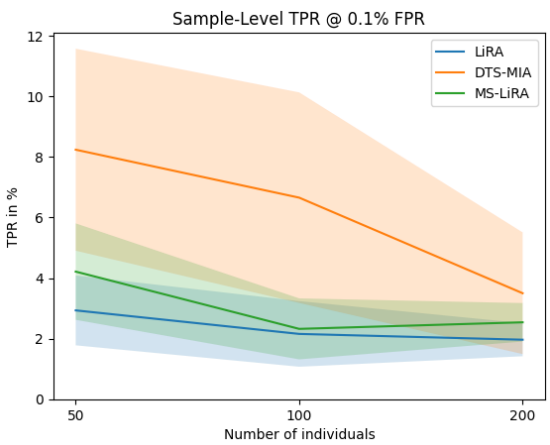


(c) User-level TPR at 0% FPR

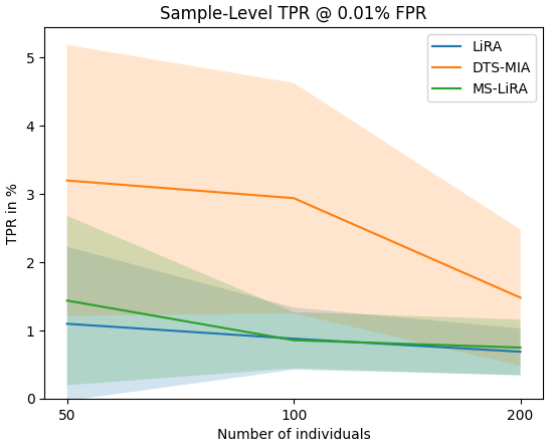
Figure 4.1: TPR for various forecasting horizons.

4.3.2 Varying Number of Individuals

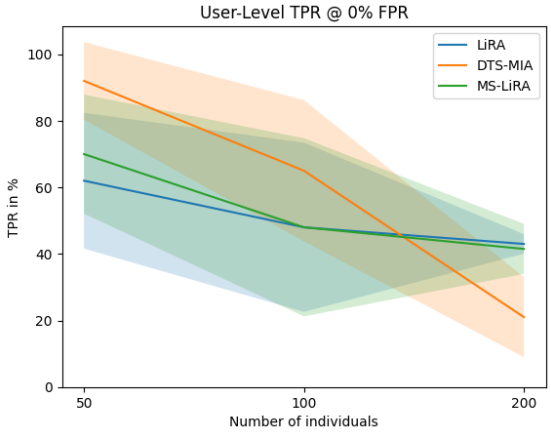
For investigating the attack performance on different number of individuals, only the number of individuals was varied while all other variables were fixed. Given the expected variability introduced by random data splits, each configuration was evaluated using five different random seeds to ensure fair comparison. We evaluated three configurations: 50, 100 (as in the main results in Section 4.2), and 200 individuals. Importantly, these refer to the total number of individuals before dataset partitioning (see Section 3.1.3), corresponding to audit sets D_{audit} with 20, 40, and 80 individuals, respectively. Sample-level and user-level results, averaged over the five seeds and reported with standard deviations, are shown in Figure 4.2. The results demonstrate a clear negative trend in both the sample-level and user-level settings, indicating that a higher number of individuals decreases susceptibility to MIAs under individual-based partitioning.



(a) Sample-level TPR at 0.1% FPR



(b) Sample-level TPR at 0.01% FPR



(c) User-level TPR at 0% FPR

Figure 4.2: TPR for various number of individuals, with confidence intervals of ± one standard deviation.

5

Discussion

5.1 Analysis of Results

This section presents a comprehensive analysis of our membership inference attacks on time series models. We start by comparing the attacks themselves, including their strengths, weaknesses, and interpretability, followed by an evaluation of the different attack signals used. We then explore how dataset characteristics influence attack performance before examining the effects of different evaluation settings, such as online versus offline and user-level versus sample-level inference. Finally, we compare our results to prior work.

5.1.1 Comparison of Attacks

We evaluted five different attacks: LiRA and RMIA, which were adapted from classification to the time series forecasting setting; the Ensemble attack, from prior work on time series MIA; and two novel attacks introduced in this thesis: Multi-Signal LiRA and DTS.

The results in Chapter 4 demonstrate that most of our attacks perform considerably well, achieving a relatively high TPR at low FPRs—especially on the user-level. However, their effectiveness varies significantly across different datasets, attack settings, and target models. No single attack consistently outperforms the others—except for the ensemble attack, which clearly underperforms, barely exceeding random guessing. Therefore, we compare the attacks based on their respective strengths and weaknesses.

LiRA and Multi-Signal LiRA not only perform well, but are also arguably the most intuitive of our attacks: modeling distributions over output signals for in- versus out-models. From a scientific perspective, this is very convenient since the distributions can be visualized to understand why, for example, some signals perform better than others. Additionally, LiRA is relatively computationally inexpensive (scaling $\mathcal{O}(n)$ to the number of datapoints), apart from the cost of training shadow models. With proper caching of model outputs, the multi-signal extension introduces only a minor computational overhead from computing multiple likelihoods per sample, while offering more stable attack performance. This reduces the need to run multiple separate LiRA attacks to identify the most effective signal.

RMIA is arguably even more mathematically rigorous, although it introduces some

difficulties in our case. Notably, in the context of time series forecasting, it is very dependent on how the attack signals are transformed into likelihoods. Although we found a reasonable transformation technique (see Section 3.4.3), there could be better approaches to utilizing the signals, and the gamma parameter could have been more carefully tuned. Despite this, RMIA achieves promising results, and seems to be a good choice especially for the *offline setting*. A notable downside of RMIA is its computational cost, especially on large datasets. Assuming the number of z samples scales proportionally with the dataset size, the number of operations required to compute marginal likelihoods grows with $\mathcal{O}(n^2)$. This quadratic scaling is not discussed in the original paper by Zarifzadeh et al. [17], but it can pose a significant bottleneck in real-world applications where datasets may be large.

As previously mentioned, the results of the ensemble attack are clearly inadequate compared to those of the other attacks across all settings. A closer inspection of the algorithm provides insight into this outcome: it trains a large number of classifiers to form the ensemble, but each model is trained on only *50 data points*, inevitably resulting in very weak predictors. Increasing the amount of training data per model would likely improve the overall performance of the ensemble. Furthermore, the ensemble attack is difficult to interpret, making it challenging to understand why certain signals—or combinations thereof, which we did not examine—are effective.

Similar to the ensemble attack, the DTS attack also adopts a black-box approach; however, unlike the ensemble, it achieves competitive results across all evaluated settings. The key advantage is that it removes the need to manually design attack signals, instead relying on a deep learning model to automatically learn useful features. However, as demonstrated by our results, this approach requires careful optimization, as the best configuration is highly dependent on the specific dataset and target model. Not only must the MIC model architecture and training procedure be tuned, but the number of shadow models must also be balanced with the fraction of MIC data extracted from each of these. Notably, the attack is GPU heavy and can quickly become computationally expensive as the amount of MIC data increases.

5.1.2 Comparison of Attack Signals

Based on the results from attacks using different signals—MSE, MAE, SMAPE, Rescaled SMAPE, TS2Vec, Trend, and Seasonality—we can draw the following conclusions. For RMIA and Ensemble, no single signal consistently outperforms the others. In contrast, for LiRA, TS2Vec and Rescaled SMAPE tend to perform better. This makes sense: TS2Vec learns representations specific to the underlying time series, which may capture more relevant patterns than error-based signals. As for Rescaled SMAPE, it uses the same scaling technique applied to logits in Carlini et al. [16], where the scaling was shown to result in more Gaussian-like output distributions. Since the LiRA attack relies on estimating a Gaussian distribution, this kind of scaling may improve performance in a similar way here, even if we do not have direct evidence that Rescaled SMAPE produces Gaussian-shaped distributions.

In a similar vein to TS2Vec, the DTS attack aims to learn useful time series representations, but optimized for the membership classification task. This automatic

feature extraction is both a strength and weakness: replacing signal handcrafting with deep learning optimization. One explanation of why DTS performs well is the inevitable loss of information when transforming raw time series into signals as those described in Section 2.4. All other attacks rely on this transformation preserving the distinguishing characteristics of membership. However, identifying effective signals is challenging, time-consuming to evaluate empirically, and costly due to the dependency on the specific attack. Therefore, DTS may serve as a practical alternative, provided that a more universal configuration can be established, reducing the need to tune attack parameters.

5.1.3 User-Level Versus Sample-Level

In general, the results on user-level versus sample-level membership inference show that it is much easier to classify the membership of an individual than that of its constituent samples. This is not entirely surprising, as user-level attacks can utilize model predictions across many samples, offering more information for inference.

Importantly, this gap in performance can lead to misleading conclusions if only sample-level metrics are considered. Some attacks may appear relatively weak when evaluated at the sample level but become highly effective once aggregated at the user level. For example, RMIA in the online setting against the N-HiTS model on the ELD dataset achieves just 1.19% TPR at 0.1% FPR on sample-level, yet obtains a 73% average user-level TPR at 0% FPR—revealing a much more serious privacy vulnerability.

5.1.4 Comparison Across Datasets

In general, the results indicate that models trained on the EEG dataset are significantly more vulnerable to leakage than those trained on the ELD dataset. Several reasons may explain this difference. First, for the EEG data we used three channels, compared to only one channel for ELD, whilst using the same horizon of 20, effectively tripling the information available to an attacker. Second, as discussed in Chapter 1, EEG signals exhibit a high degree of uniqueness [20]. While prior studies have shown that household electricity consumption can also uniquely identify individuals [53], electricity usage patterns may still be inherently less distinctive than EEG signals due to greater similarity in daily routines and overlapping behaviors across individuals. Additionally, EEG data offers a significantly finer temporal resolution (4 milliseconds) compared to ELD (1 hour), providing attackers with more detailed information. Together, these factors may explain the stronger membership inference performance observed with EEG data.

5.1.5 Online Versus Offline

Overall, online attacks consistently outperformed offline attacks, often by a substantial margin. This performance gap appears larger than what has been observed in previous work on other data modalities [16, 17], suggesting that MIAs in the time series domain may be particularly sensitive to distribution shifts. Alternatively, a

higher distribution shift may arise from our *individual-based partitioning* strategy. With only 100 highly heterogeneous individuals, there is limited representational overlap and redundancy between the audit and auxiliary sets. In contrast, prior work often partitions at the sample level, leveraging potentially millions of data points to maintain overlap and reduce variance across subsets. The combination of small sample size and pronounced inter-individual variability in our setting likely amplifies distribution shifts, thereby widening the performance gap between online and offline attacks.

5.1.6 Comparison with Prior Work

The previous work by Koren et al. [19] evaluated the Ensemble attack on both the TUH-EEG dataset and an ECG dataset. In our study, we used several of the same signals—MSE, Trend, and Seasonality—but excluded MASE due to consistently poor attack performance.

Our attacks—LiRA, Multi-Signal LiRA, RMIA, and DTS—substantially outperformed the Ensemble attack on the TUH-EEG dataset, which exhibited very weak performance in our setting. However, the performance of the Ensemble attack is not fully consistent with the results reported in [19], where the authors achieved a 44% TPR at 1% FPR against an LSTM-based model on the TUH-EEG dataset using the Trend signal. We believe this discrepancy arises from differences in evaluation methodology. Specifically, our setup uses a more realistic and fair evaluation scheme: our target models are trained with early stopping, and achieves test MSE orders of magnitude lower than theirs (10^{-9} versus 10^{-2}). This suggests that their models may have been severely overfitted (or underfitted, since they do not provide train metrics), potentially inflating attack performance. Alternatively, the metrics they provide might just be scaled MSE values, which makes it harder to interpret the results or reproduce them accurately.

5.2 Privacy Implications for Time Series Models

Our empirical evaluation demonstrates that deep learning forecasting models remain vulnerable to MIAs in realistic scenarios. Although steps have been taken to mitigate overfitting and bias—such as data preprocessing, careful model selection, and early stopping—our attacks are still able to successfully identify training set members with high confidence. In particular, the superior performance of N-HiTS comes at the cost of greater privacy leakage compared to LSTM. Its sharper fit to the training data seems to amplify exploitable output patterns, highlighting an inherent *utility-privacy trade-off* in time series forecasting. More complex models may tend to memorize more specific temporal characteristics, increasing the risk for data leakage.

Although the attacks demonstrated in this work may not pose an immediate threat on their own, they highlight a significant risk of data leakage in time series forecasting models. In real forecasting applications—whether for healthcare monitoring, smart grid management, or other sensitive domains—this vulnerability may lead to concrete privacy exposure. For example, in a healthcare setting, this vulnerability

might reveal sensitive health conditions, while in energy systems, it could expose private behavioral patterns such as occupancy or appliance usage.

To mitigate these risks, developers should consider various privacy-preserving techniques. The most common approaches include *differential privacy* mechanisms which provide a mathematical guarantee by adding *controlled noise* to the data or model updates, and homomorphic encryption, which allows training ML models on encrypted data [54]. Another interesting method is *mimic learning*, which utilizes a teacher/student concept to perform a form of *knowledge distillation*. Keeping accuracy at an acceptable level seems to be one of the main challenges, together with handling computation costs [54]. Techniques such as activation function approximation or partial sharing might reduce the model's accuracy, while methods such as homomorphic encryption introduce substantial computational overhead. Therefore careful calibration is required to balance utility and privacy. Notably, MIAs—such as those studied in this thesis—can be used for evaluating differential privacy in machine learning by providing empirical lower bounds that complements theoretical analysis, as demonstrated by Nasr et al. [14].

Moreover, defenses and evaluation should account for different levels of membership inference: MIAs can target either individual samples (sliding-window segments) or entire user time series. Our aggregation experiments show that user-level MIA is significantly easier than sample-level MIA, as a large portion of users can be identified at 0.0% FPR even when sample-level TPR remains relatively low. Importantly, *user-level membership inference is more critical* from a privacy perspective, as it reveals information about a real individual rather than an isolated, potentially ambiguous sample. This can expose personal involvement in sensitive activities such as medical treatments or energy consumption habits, making the privacy breach more substantial and identifiable. Additionally, defenses that are effective at the sample level may fail to protect against user-level attacks, highlighting the need for evaluation and mitigation strategies that explicitly address both.

Ultimately, the deployment of time series models in sensitive domains calls for a comprehensive assessment of potential data leakage, carefully balancing utility and privacy. This includes the deliberate selection of model architectures—potentially favoring slightly less accurate but more privacy-preserving alternatives—the application and fine-tuning of privacy-preserving mechanisms, and the continuous monitoring for stronger MIA techniques.

5.3 Limitations

This study investigates MIAs in the context of time series forecasting, by testing a wide range of attacks to both evaluate the effectiveness of them, as well as identify inherent vulnerabilities in specific models and datasets. The benchmarking suite, as well as the attacks, make assumptions about adversarial capability in terms of access to data and computational resources. However, this setup may not fully reflect realistic conditions when considering how these results translate to real-life deployments of time series models.

5.3.1 Data Distribution

A key limitation is the assumption that the adversary has access to a population drawn from the same distribution (i.e., from the same dataset) as the target model training data to train the shadow models. In a realistic attack scenario, the adversary would likely be able to produce a population dataset of a similar distribution, but not one that is exactly identical. Likewise, the adversary is unlikely to know the exact target model architecture and training procedure. Data distribution shift and model mismatch have been studied and found to affect attack effectiveness [16, 55, 56].

5.3.2 Attack Settings

Our benchmark features the three distinct attack settings: *online*, *offline*, and *audit mode*. Both online and offline modes are applicable to all the evaluated attacks except the ensemble attack and are common attack settings in research [16, 17, 57]. The distinction between *online* and *offline* lies in computational constraints: an *offline* adversary can prepare shadow models in advance and query any point not used to train these models, whereas an *online* adversary must train new shadow models whenever querying new points. The ensemble attack [38] operates under a distinct setting known as audit mode. Like the online setting, it allows the attack to be tailored to the audit set, but it also assumes access to the true membership labels of that set. This effectively gives the adversary knowledge of exactly which individuals were used to train the target model—a clearly unrealistic assumption. While audit mode can help evaluate an upper bound on attack performance, it does not reflect any practical threat scenario. Notably, even with this strong advantage, the ensemble attack underperformed in our experiments.

Notably, there is a significant difference between auditing and executing a real-world attack, which limits the practical applicability of our evaluation methods. For instance, during auditing, we sweep the decision threshold to evaluate our attacks across different FPRs, whereas a real-world adversary would have to carefully determine this threshold. In attacks such as LiRA, threshold selection can be relatively straightforward—assuming the shadow model's in- and out-sample distributions are well separated. In contrast, black-box attacks like DTS offer no obvious strategy for threshold selection, often requiring empirical tuning through simulated attack scenarios.

5.3.3 Individual-Level Partitioning

We partition the data by individual rather than by time, diverging from standard practice in time series forecasting [58, 59]. We argue that this is the most practical and intuitive method for performing membership inference in time series contexts.

Avoiding data leakage between subsets is essential for membership inference to be meaningful in time series settings. With a sliding window approach, adjacent samples share all but one of their time steps. For example, if the model is trained on one window and tested on the next, it has effectively seen almost the entire content

of the test sample—even though it's technically labeled a non-member. In this case, two nearly identical samples could receive opposite membership labels, which makes the membership inference task practically impossible.

Because we not only split the dataset into audit, auxiliary, and validation subsets, but also resample new training sets each time we train a shadow model, we need a partitioning strategy that consistently avoids data leakage. Splitting by time would require complex logic to remove all overlapping segments between training and test sets, which not only introduces implementation challenges but also unnecessarily reduces the number of usable datapoints. In contrast, individual-level partitioning is simple to implement and guarantees that no overlapping samples appear across subsets. It also naturally enables user-level membership inference, where the attackers goal is to determine whether an entire individual's time series was included in training.

5.3.4 Data Preprocessing

Another factor that affects the realism of our attacks is the preprocessing applied to the datasets. The EEG data underwent some manual data cleaning to remove corrupted datapoints and artifacts. However, this process does not ensure that the data was entirely artifact-free or of consistently high quality. In contrast, a real-world deployment with cleaner, artifact-free data could lead to lower memorization of such outliers—thereby reducing attack effectiveness. Furthermore, we applied IQR scaling for both datasets. For simplicity, this scaling was performed globally—using statistics computed over all splits—which introduces a degree of data leakage between the training, testing, and auxiliary sets. Although this choice simplified preprocessing, it further reduces the realism of our evaluation and may affect attack performance.

5.4 Future Work

This study only considered two datasets, TUH-EEG and ELD, as described in Section 3.1.1 and 3.1.2. Although a useful start, these datasets offer limited insight into privacy issues in the healthcare and electricity domains. For instance, the healthcare domain includes many other relevant subfields, such as ECG (electrocardiogram) data and sleep patterns, which are collected in real-time by popular wearable devices like smartwatches. Beyond healthcare and electricity, additional domains—such as financial transactions, IoT sensor data, and location traces (e.g., GPS or public transit usage)—also involve time series data with significant privacy concerns. As shown in Chapter 4, models trained on EEG data exhibit a significantly greater susceptibility to MIAs, compared to ELD models. Evaluating risks in a broader range of domains is therefore essential for developing a more comprehensive understanding of privacy vulnerabilities in time series forecasting models.

Moreover, due to time constraints, we limited our evaluation to two model architectures: the RNN-based LSTM and the N-HiTS model with its hierarchical interpolation mechanism. However there are many more relevant models in the context of

time series forecasting. For example, promising complementary architectures include transformer-based models [60, 61] and architectures built on dilated convolutions, such as the TCN [62]. Investigating additional target models would undoubtedly contribute to a more nuanced risk assessment and could help identify architectures that offer a more favorable balance between performance and privacy.

Another area worth exploring is the impact of exovariates. Many practical forecasting systems benefit from incorporating exogenous variables such as weather, public holidays or time of day [63]. Investigating whether these additional variables amplify data memorization, and thus potentially making the model more vulnerable to attacks, would be a valuable contribution. Similarly, extending this work to probabilistic forecasting models—which produce quantiles or full predictive distributions—could offer a deeper understanding of how richer model outputs influence attack efficacy, potentially in ways comparable to the trends observed when increasing the forecasting horizon (see Section 4.3.1).

While this thesis focused on time series forecasting, privacy concerns are equally relevant in time series classification tasks, such as activity recognition, fraud detection, or medical diagnosis. Applying wide-studied state-of-the-art classification MIAs to time series could reveal new risks. Classification may even be more vulnerable than forecasting, as statistical attacks like LiRA and RMIA can be directly applied. However, the reduced output information per sample—logits instead of full forecasts—might also make such attacks more challenging.

Lastly, since user-level attacks are arguably more critical from a privacy perspective, we believe they deserve further investigation in future time series research. Importantly, our current user-level inference methodology simply involves multiplying the MIA scores of every sample to determine the membership of an individual—treating all samples as independent and equally weighted. To improve upon this, future work could investigate more nuanced methods of aggregating sample-level scores, possibly taking into account the overlap in samples, and their position in time. Alternatively, develop end-to-end user-level attacks that operate directly on the full set of sample predictions.

5.5 Ethical Considerations

The purpose of this study was to identify vulnerabilities in ML models and foster careful risk assessment when dealing with sensitive time series data. However on the contrary, the findings of our work may expose vulnerabilities which can be exploited for malicious purposes. Therefore we have made sure to only study publicly available benchmarking datasets, as well as sharing all our findings for further research purposes. Additionally, by incorporating our work into the privacy auditing tool LeakPro, we allow defenders to evaluate their protection against the enhanced attacks discovered in our study.

It is impossible to predict the strength or timing of future attacks. However, by continuously incorporating more powerful attacks and benchmarking our models against them, we can begin to approximate an evolving upper bound on data leakage, con-

strained by our current knowledge. This, in turn, enables a more careful evaluation of risks and safeguards, ultimately fostering a safer environment for developing time series models in sensitive domains.

5.6 Conclusion

This thesis presented a comprehensive study of membership inference attacks targeting deep learning-based time series forecasting models. While MIAs have been widely studied in classification settings, their applicability to forecasting tasks has remained largely unexplored. To address this, we adapted two state-of-the-art classification-focused attacks—LiRA and RMIA—to the time series forecasting domain. Furthermore, we proposed an extension of LiRA into a multi-signal variant, capable of leveraging multiple attack features. In addition, we implemented the Ensemble attack from prior work and introduced a new method, the DTS attack, which learns membership inference end-to-end without relying on handcrafted signals.

Our experiments on two real-world datasets (EEG, electricity load) and two model architectures (LSTM, N-HiTS) revealed that time series forecasting models are indeed vulnerable to MIAs, particularly under user-level attack scenarios. We found that both traditional statistical MIA approaches—such as LiRA and RMIA—and deep learning-based methods like DTS perform well in the time series forecasting setting. However, statistical attacks rely on careful selection of attack signals, while DTS requires careful tuning of parameters. Even so, both approaches consistently outperform the prior state-of-the-art for time series MIAs (the Ensemble attack) by a wide margin.

In addition to developing and benchmarking attacks, we laid groundwork for user-level membership inference in time series forecasting, a threat model we believe is highly relevant for real-world deployments and essential to address in future research. Our findings establish a strong baseline and offer practical tools for auditing and understanding privacy risks in time series forecasting systems.

Bibliography

[1] V. C. Storey, W. T. Yue, J. L. Zhao, and R. Lukyanenko, "Generative artificial intelligence: Evolving technology, growing societal impact, and opportunities for information systems research," *Information Systems Frontiers*, Feb. 25, 2025. DOI: 10.1007/s10796-025-10581-7.

[2] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194, p. 20 200 209, Feb. 2021, ISSN: 1471-2962. DOI: 10.1098/rsta.2020.0209. [Online]. Available: http://dx.doi.org/10.1098/rsta.2020.0209.

[3] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15, Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 1322–1333, ISBN: 9781450338325. DOI: 10.1145/2810103.2813677. [Online]. Available: https://doi.org/10.1145/2810103.2813677.

[4] N. Carlini *et al.*, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*, USENIX Association, Aug. 2021, pp. 2633–2650, ISBN: 978-1-939133-24-3. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

[5] N. Carlini *et al.*, *Extracting training data from diffusion models*, 2023. arXiv: 2301.13188 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2301.13188.

[6] European Parliament and Council of the European Union. "Regulation (EU) 2016/679 of the European Parliament and of the Council," of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). (May 4, 2016), [Online]. Available: https://data.europa.eu/eli/reg/2016/679/oj (visited on 04/13/2023).

[7] Integritetskyddsmyndigheten (IMY), "Federerad maskininlärning mellan två vårdgivare," no. IMY-2023-260, Mar. 15, 2023. [Online]. Available: %7Bhttps://www.imy.se/globalassets/dokument/rapporter/slutrapport-om-imys-pilotprojekt-med-regulatorisk-testverksamhet-om-dataskydd-230315.pdf%7D.

[8] AI-kommissionen, *Ai-kommissionens färdplan för sverige*, Nov. 26, 2024.

[9] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, *Membership inference attacks on machine learning: A survey*, 2022. arXiv: 2103.07853 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2103.07853.

[10] L. Bai, H. Hu, Q. Ye, H. Li, L. Wang, and J. Xu, *Membership inference attacks and defenses in federated learning: A survey*, 2024. arXiv: 2412.06157 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2412.06157.

[11] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, *Membership inference attacks against machine learning models*, 2017. arXiv: 1610.05820 [cs.CR]. [Online]. Available: https://arxiv.org/abs/1610.05820.

[12] Z. Yang, B. Shao, B. Xuan, E.-C. Chang, and F. Zhang, *Defending model inversion and membership inference attacks via prediction purification*, 2020. arXiv: 2005.03915 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2005.03915.

[13] N. Sula, A. Kumar, J. Hou, H. Wang, and R. Tourani, *Silver linings in the shadows: Harnessing membership inference for machine unlearning*, 2024. arXiv: 2407.00866 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2407.00866.

[14] M. Nasr, S. Song, A. Thakurta, N. Papernot, and N. Carlini, *Adversary instantiation: Lower bounds for differentially private machine learning*, 2021. arXiv: 2101.04535 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2101.04535.

[15] Y. Xiao, Q. Ye, H. Hu, H. Zheng, C. Fang, and J. Shi, "Mexmi: Pool-based active model extraction crossover membership inference," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 10 203–10 216. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/4241c27d3161c7a7064bfc1a6e539563-Paper-Conference.pdf.

[16] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramèr, "Membership inference attacks from first principles," *CoRR*, vol. abs/2112.03570, 2021. arXiv: 2112.03570. [Online]. Available: https://arxiv.org/abs/2112.03570.

[17] S. Zarifzadeh, P. Liu, and R. Shokri, *Low-cost high-power membership inference attacks*, 2024. arXiv: 2312.03262 [stat.ML]. [Online]. Available: https://arxiv.org/abs/2312.03262.

[18] S. Hisamoto, M. Post, and K. Duh, "Membership inference attacks on sequence-to-sequence models," *CoRR*, vol. abs/1904.05506, 2019. arXiv: 1904.05506. [Online]. Available: http://arxiv.org/abs/1904.05506.

[19] N. Koren, A. Goldsteen, G. Amit, and A. Farkash, *Membership inference attacks against time-series models*, 2024. arXiv: 2407.02870 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2407.02870.

[20] A. Jalaly Bidgoly, H. Jalaly Bidgoly, and Z. Arezoumand, "A survey on methods and challenges in eeg based authentication," *Computers & Security*, vol. 93, p. 101 788, 2020, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2020.101788. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820300730.

[21] M. R. Asghar, G. Dán, D. Miorandi, and I. Chlamtac, "Smart meter data privacy: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2820–2835, 2017. DOI: 10.1109/COMST.2017.2720195.

[22] M. A. Lisovich, D. K. Mulligan, and S. B. Wicker, "Inferring personal information from demand-response systems," *IEEE Security & Privacy*, vol. 8, no. 1, pp. 11–20, 2010. DOI: 10.1109/MSP.2010.40.

[23] AI Sweden, *Leakpro: Leakage profiling and risk oversight of machine learning models.* [Online]. Available: https://github.com/aidotse/LeakPro.

[24] G. Ziffer, "Time series analytics: Temporal dependence, ARIMA and order estimation," Lecture, Politecnico di Milano, Sep. 2023.

[25] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PLOS ONE*, vol. 13, no. 3, pp. 1–26, Mar. 2018. DOI: 10.1371/journal.pone.0194889. [Online]. Available: https://doi.org/10.1371/journal.pone.0194889.

[26] S. Makridakis, E. Spiliotis, V. Assimakopoulos, A.-A. Semenoglou, G. Mulder, and K. N. and, "Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward," *Journal of the Operational Research Society*, vol. 74, no. 3, pp. 840–859, 2023. DOI: 10.1080/01605682.2022.2118629. eprint: https://doi.org/10.1080/01605682.2022.2118629. [Online]. Available: https://doi.org/10.1080/01605682.2022.2118629.

[27] scikit-learn, *sklearn.preprocessing.robustscaler*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html.

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/.

[29] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990. DOI: https://doi.org/10.1207/s15516709cog1402_1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog1402_1. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1.

[30] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[31] Z. Xu, X. Chen, L. Yang, J. Xu, and S. Zhou, Licensed under CC BY-SA 4.0: https://creativecommons.org/licenses/by-sa/4.0/, Jan. 2024. DOI: 10.3934/era.2024183. [Online]. Available: https://www.researchgate.net/publication/381727159_Multi-modal_adaptive_feature_extraction_for_early-stage_weak_fault_diagnosis_in_bearings.

[32] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, *N-beats: Neural basis expansion analysis for interpretable time series forecasting*, 2020. arXiv: 1905.10437 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1905.10437.

[33] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018, ISSN: 0169-2070. DOI: https://

doi.org/10.1016/j.ijforecast.2018.06.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207018300785.

[34] G. Athanasopoulos, R. J. Hyndman, H. Song, and D. C. Wu, "The tourism forecasting competition," *International Journal of Forecasting*, vol. 27, no. 3, pp. 822–844, 2011, Special Section 1: Forecasting with Artificial Neural Networks and Computational Intelligence Special Section 2: Tourism Forecasting, ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2010.04.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016920701000107X.

[35] C. Challu, K. G. Olivares, B. N. Oreshkin, F. Garza, M. Mergenthaler-Canseco, and A. Dubrawski, *N-hits: Neural hierarchical interpolation for time series forecasting*, 2022. arXiv: 2201.12886 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2201.12886.

[36] A. Salem *et al.*, *Sok: Let the privacy games begin! a unified treatment of data inference privacy in machine learning*, 2023. arXiv: 2212.10986 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2212.10986.

[37] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri, *Enhanced membership inference attacks against machine learning models*, 2022. arXiv: 2111.09679 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2111.09679.

[38] S. Shachor, N. Razinkov, and A. Goldsteen, *Improved membership inference attacks against language classification models*, 2024. arXiv: 2310.07219 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2310.07219.

[39] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, *Privacy risk in machine learning: Analyzing the connection to overfitting*, 2018. arXiv: 1709.01604 [cs.CR]. [Online]. Available: https://arxiv.org/abs/1709.01604.

[40] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, *Memguard: Defending against black-box membership inference attacks via adversarial examples*, 2019. arXiv: 1909.10594 [cs.CR]. [Online]. Available: https://arxiv.org/abs/1909.10594.

[41] M. Nasr, R. Shokri, and A. Houmansadr, *Machine learning with membership privacy using adversarial regularization*, 2018. arXiv: 1807.05852 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1807.05852.

[42] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, IEEE, May 2019, pp. 739–753. DOI: 10.1109/sp.2019.00065. [Online]. Available: http://dx.doi.org/10.1109/SP.2019.00065.

[43] L. Song, R. Shokri, and P. Mittal, "Privacy risks of securing machine learning models against adversarial examples," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS 19, ACM, Nov. 2019, pp. 241–257. DOI: 10.1145/3319535.3354211. [Online]. Available: http://dx.doi.org/10.1145/3319535.3354211.

[44] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1985. DOI: https://doi.org/10.1017/CBO9780511810817.

[45] Z. Yue *et al.*, *Ts2vec: Towards universal representation of time series*, 2022. arXiv: 2106.10466 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2106.10466.

[46] H.-J. Zhou, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0/>, May 2023. DOI: 10.1088/1742-6596/2493/1/012021. [Online]. Available: https://www.researchgate.net/publication/370782619_TS2VLGNN-based_prediction_of_water_temperature_at_the_outlet_valve_of_a_valve_cooling_system.

[47] I. Obeid and J. Picone, "The temple university hospital eeg data corpus," *Frontiers in Neuroscience*, vol. Volume 10 - 2016, 2016, ISSN: 1662-453X. DOI: 10.3389/fnins.2016.00196. [Online]. Available: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2016.00196.

[48] A. Trindade, *ElectricityLoadDiagrams20112014*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C58C86, 2015.

[49] *cchallu N-HiTS*. [Online]. Available: https://github.com/cchallu/n-hits.

[50] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1412.6980.

[51] H. Ismail Fawaz *et al.*, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, Sep. 2020, ISSN: 1573-756X. DOI: 10.1007/s10618-020-00710-y. [Online]. Available: http://dx.doi.org/10.1007/s10618-020-00710-y.

[52] *hfawaz InceptionTime*. [Online]. Available: https://github.com/hfawaz/InceptionTime.

[53] A. Voyez, T. Allard, G. Avoine, P. Cauchois, E. Fromont, and M. Simonin, *Unique in the smart grid -the privacy cost of fine-grained electrical consumption data*, 2022. arXiv: 2211.07205 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2211.07205.

[54] A. Boulemtafes, A. Derhab, and Y. Challal, "A review of privacy-preserving techniques for deep learning," *Neurocomputing*, vol. 384, pp. 21–45, 2020, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.11.041. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219316431.

[55] A.-M. Cretu, D. Jones, Y.-A. de Montjoye, and S. Tople, *Investigating the effect of misalignment on membership privacy in the white-box setting*, 2024. arXiv: 2306.05093 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2306.05093.

[56] S. Li, Y. Wang, Y. Li, and Y.-a. Tan, *L-leaks: Membership inference attacks with logits*, 2022. arXiv: 2205.06469 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2205.06469.

[57] D. Willmott, A. K. Sahu, F. Sheikholeslami, F. Condessa, and Z. Kolter, *You only query once: Effective black box adversarial attacks with minimal repeated queries*, 2021. arXiv: 2102.00029 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2102.00029.

[58] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *Interna-*

tional Journal of Forecasting, vol. 37, no. 1, pp. 388–427, 2021, ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2020.06.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207020300996.

[59] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach," *Expert Systems with Applications*, vol. 140, p. 112 896, 2020, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.112896. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419306128.

[60] H. Zhou *et al.*, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 11 106–11 115, May 2021. DOI: 10.1609/aaai.v35i12.17325. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17325.

[61] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, *Temporal fusion transformers for interpretable multi-horizon time series forecasting*, 2020. arXiv: 1912.09363 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1912.09363.

[62] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang, "Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting," *Electronics*, vol. 8, no. 8, 2019, ISSN: 2079-9292. DOI: 10.3390/electronics8080876. [Online]. Available: https://www.mdpi.com/2079-9292/8/8/876.

[63] A. Mohammadi, G. Nápoles, and Y. Salgueiro, "Deep learning for time-series forecasting with exogenous variables in energy consumption: A performance and interpretability analysis," *IEEE Access*, vol. PP, pp. 1–1, Jan. 2025. DOI: 10.1109/ACCESS.2025.3570618.

A

Signal Correlation Investigation

To determine whether the different signals described in Section 2.4 capture *distinct* membership traits from the raw outputs — and ultimately if there is a rationale for combining them — we conducted a correlation analysis on the *LiRA online* attack. Specifically, we computed the *Pearson correlation coefficient* (PCC) between the *MIA scores* (log likealihood ratios) obtained from the various signals.

For each of the 7 signals, we obtained an array with the MIA scores used to classify sample membership. These arrays are aligned with respect to the samples, so the PCC between two signals s_1, s_2, is given by

$$r_{1,2} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where x, y denote the MIA scores obtained from LiRA using signals s_1 and s_2, respectively. Since the analysis was repeated across five random seeds, we aggregated the results to show both the mean and standard deviation of each PCC. The results for both datasets and target models are presented in the *correlation matrices* in Figures A.1, A.2, A.3, and A.4.

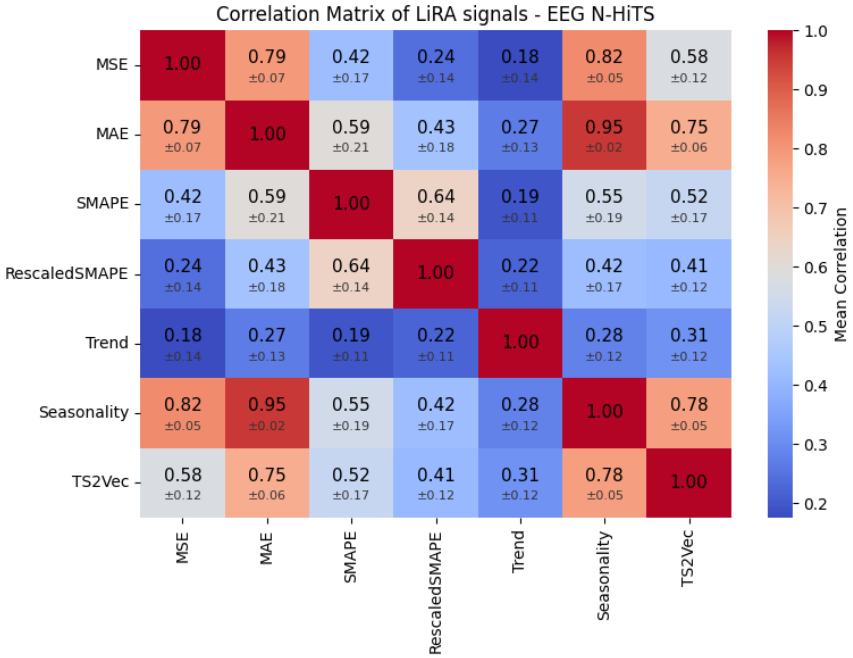


Figure A.1: The correlation matrix for the different attack signals used in LiRA online, evaluated on the TUH-EEG data set and target N-HiTS model. Each PCC is given as the mean, ± the standard deviation, over five different random seeds.

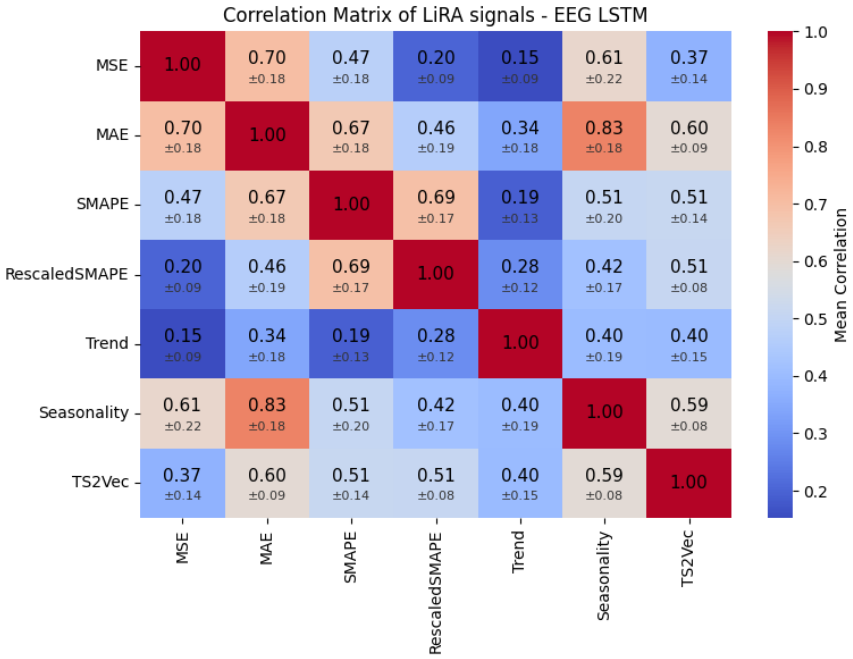


Figure A.2: The correlation matrix for the different attack signals used in LiRA online, evaluated on the TUH-EEG data set and LSTM target model. Each PCC is given as the mean, ± the standard deviation, over five different random seeds.

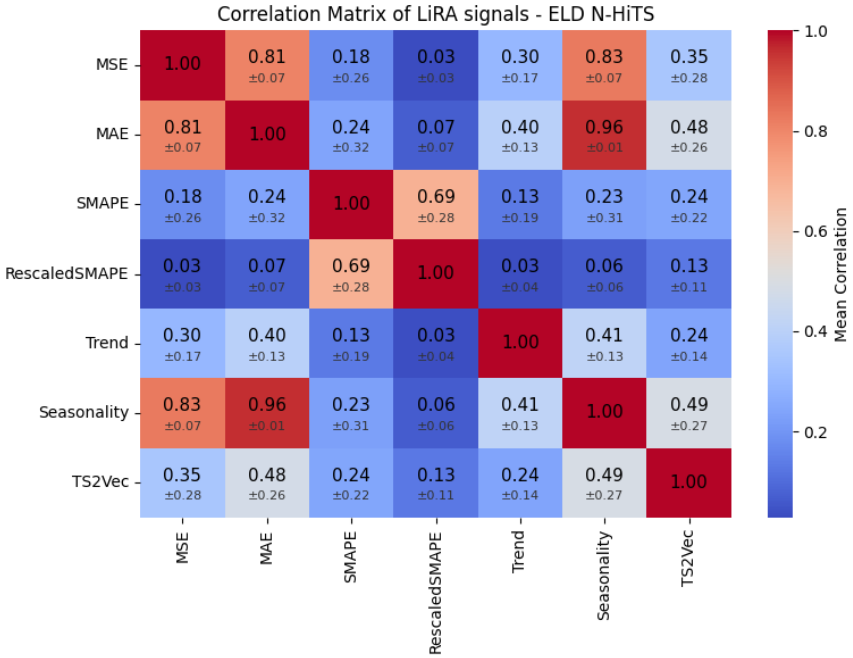


Figure A.3: The correlation matrix for the different attack signals used in LiRA online, evaluated on the ELD data set and N-HiTS target model. Each PCC is given as the mean, ± the standard deviation, over five different random seeds.

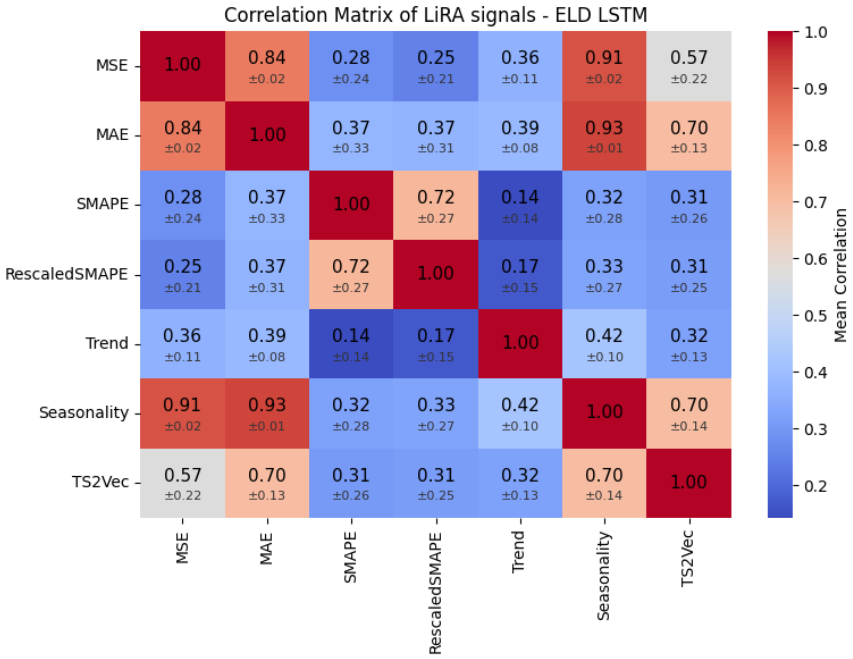


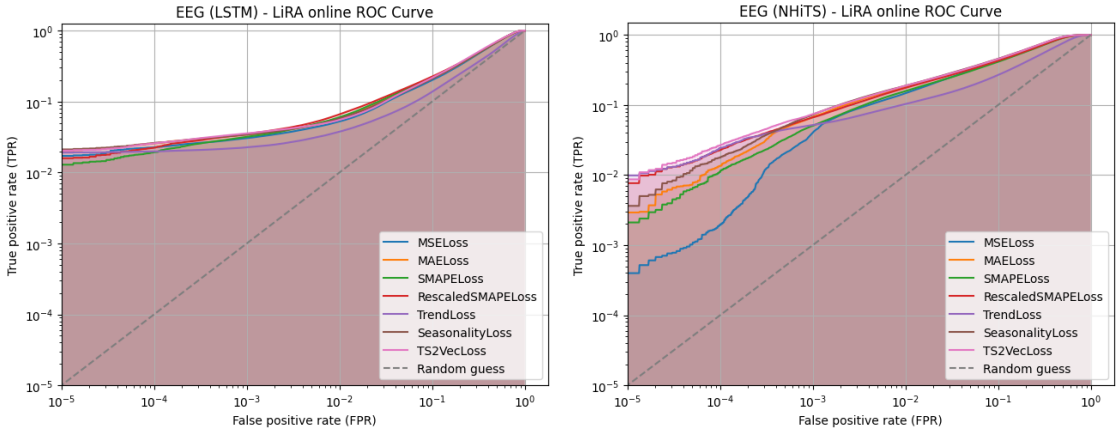
Figure A.4: The correlation matrix for the different attack signals used in LiRA online, evaluated on the ELD data set and LSTM target model. Each PCC is given as the mean, ± the standard deviation, over five different random seeds.

B

Attack ROC Curves

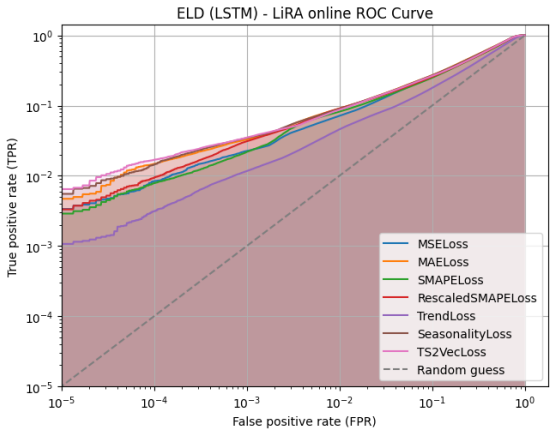
The following sections present sample-level ROC curves, generated by thresholding the MIA scores at varying levels. Both the false positive rate (FPR) and true positive rate (TPR) are shown on a logarithmic scale, ranging from 10^{-5} to 10^{0}. The curves display the mean TPR across five runs.

B.1 LiRA

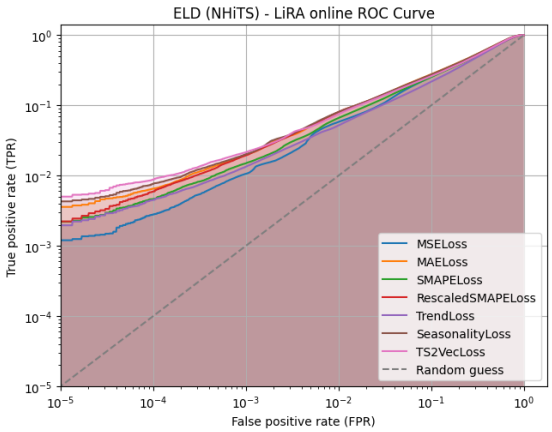


(a) EEG LSTM

(b) EEG NHiTS

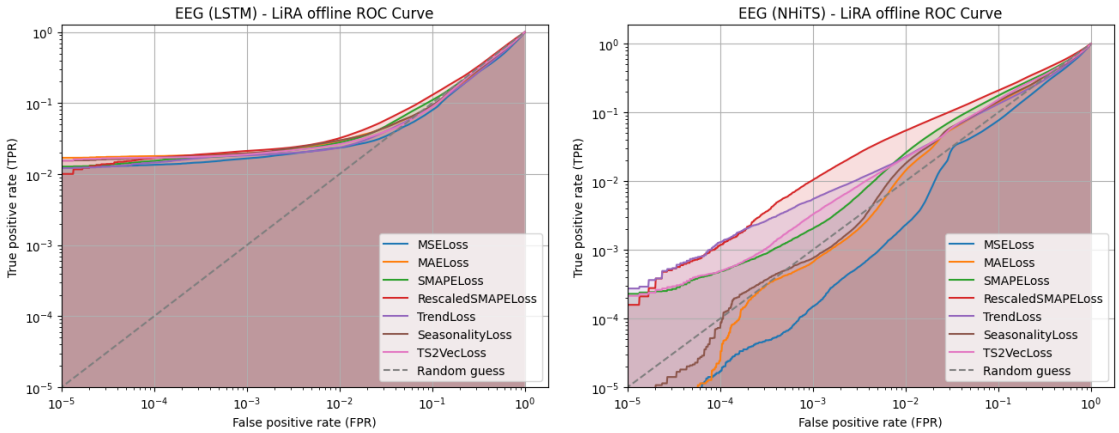


(c) ELD LSTM



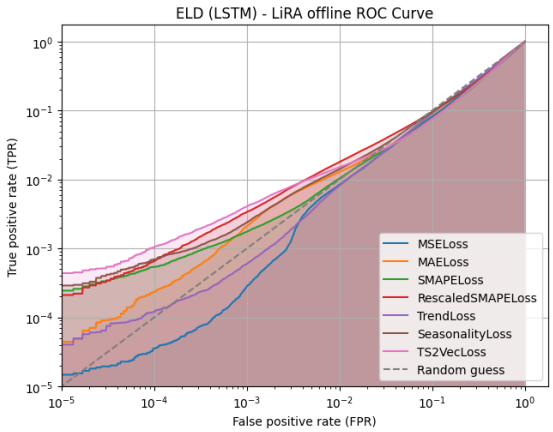
(d) ELD NHiTS

Figure B.1: LiRA online sample-level ROC Curve, for both datasets and target models.

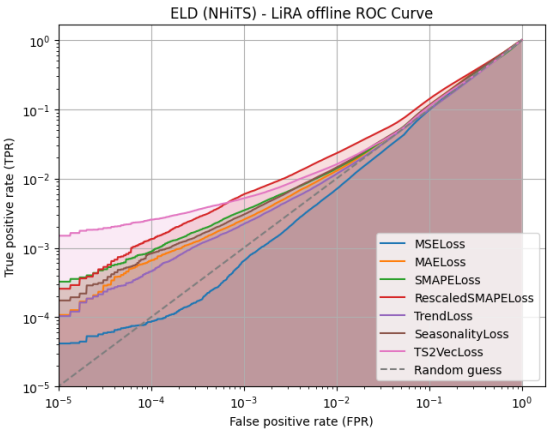


(a) EEG LSTM

(b) EEG NHiTS



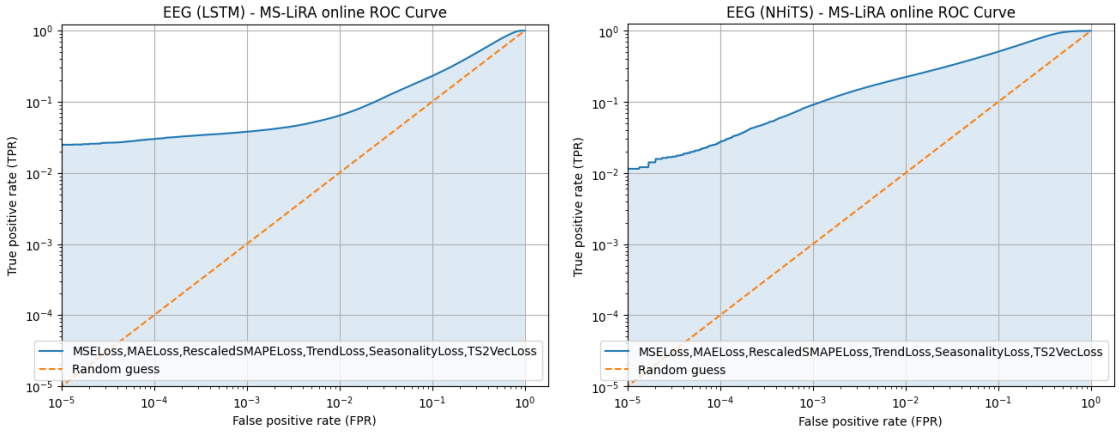
(c) ELD LSTM



(d) ELD NHiTS

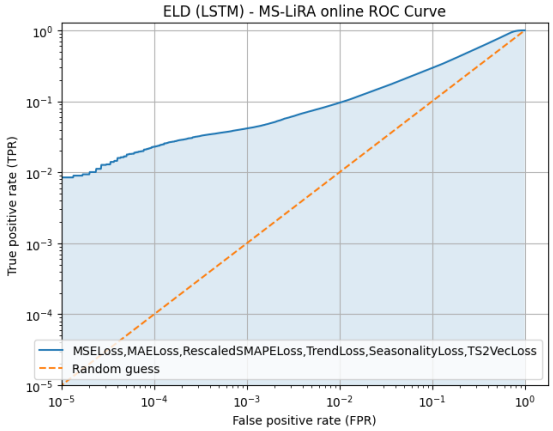
Figure B.2: LiRA offline sample-level ROC Curve, for both datasets and target models.

B.2 Multi-Signal LiRA

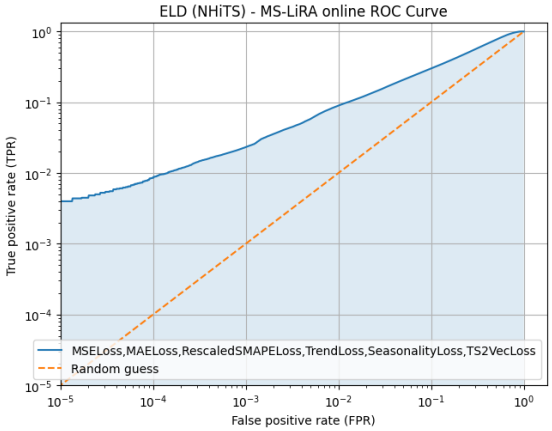


(a) EEG LSTM

(b) EEG NHiTS

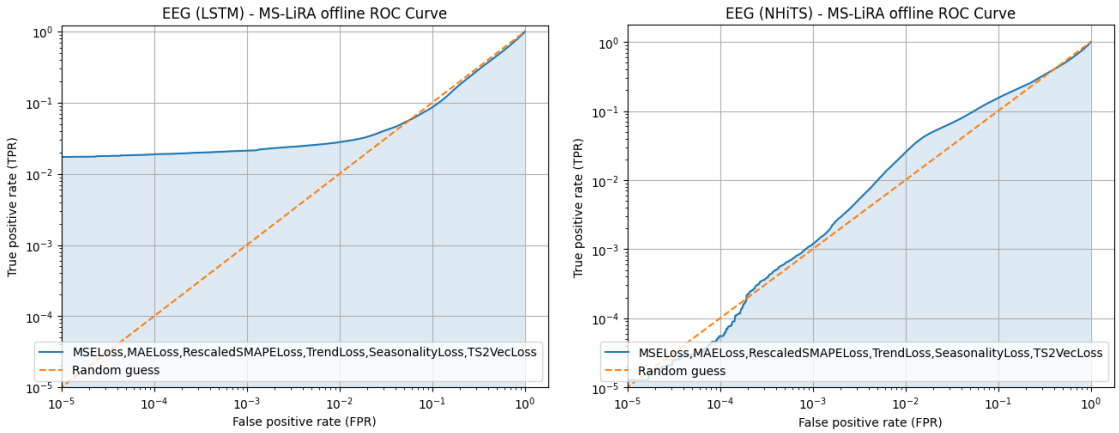


(c) ELD LSTM



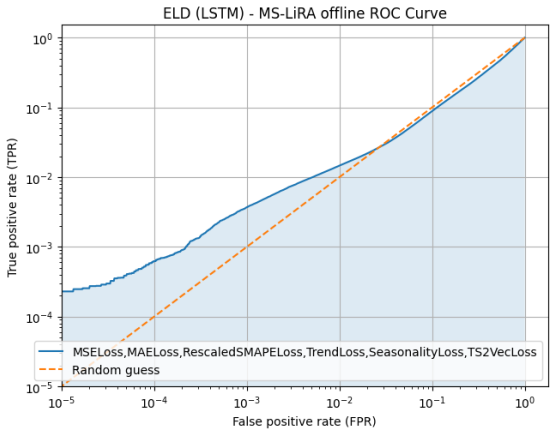
(d) ELD NHiTS

Figure B.3: Multi-Signal LiRA online sample-level ROC Curve, for both datasets and target models.

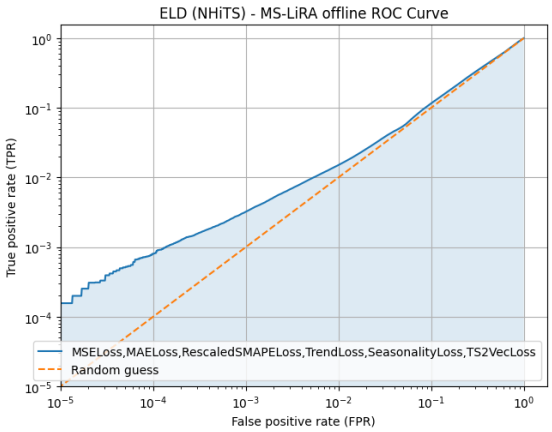


(a) EEG LSTM

(b) EEG NHiTS



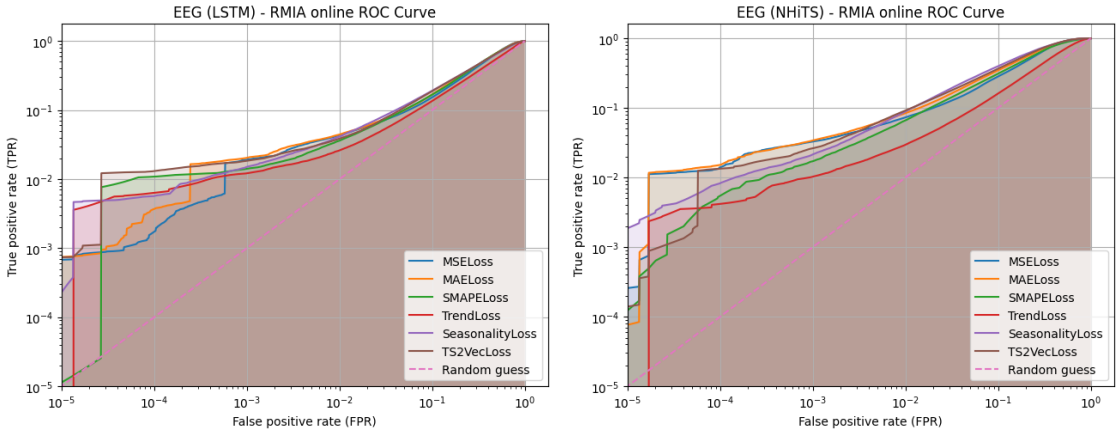
(c) ELD LSTM



(d) ELD NHiTS

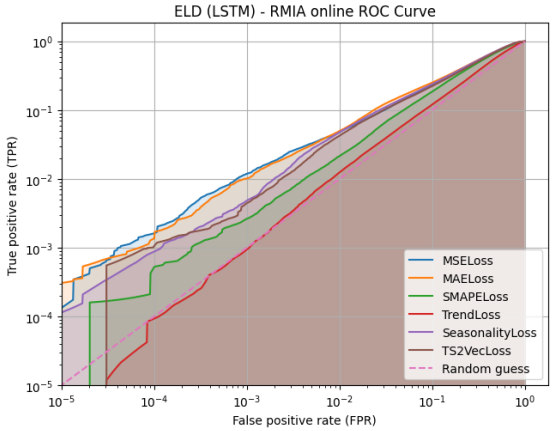
Figure B.4: Multi-Signal LiRA offline sample-level ROC Curve, for both datasets and target models.

B.3 RMIA

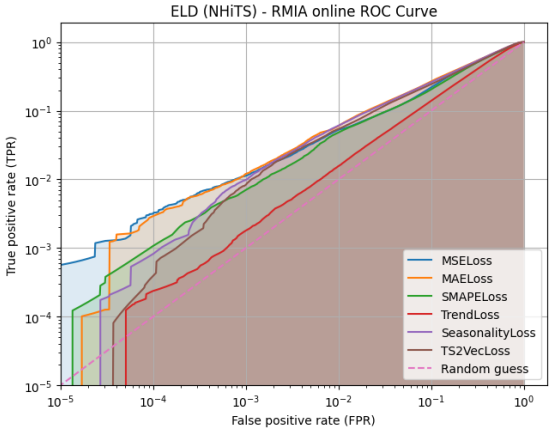


(a) EEG LSTM

(b) EEG NHiTS

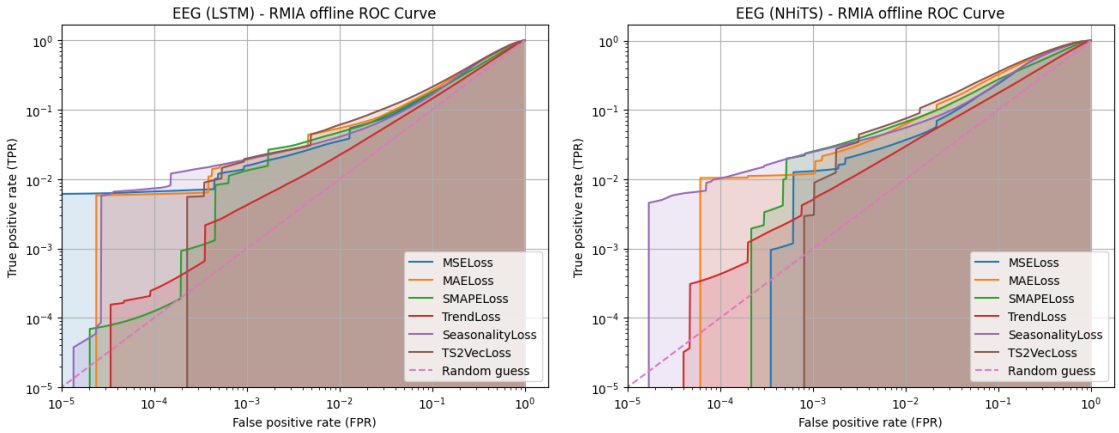


(c) ELD LSTM



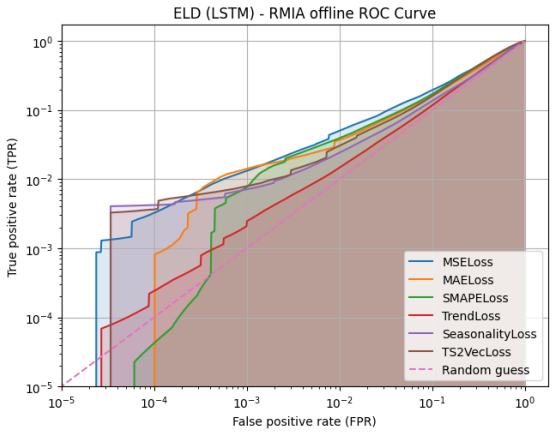
(d) ELD NHiTS

Figure B.5: RMIA online sample-level ROC Curve, for both datasets and target models.

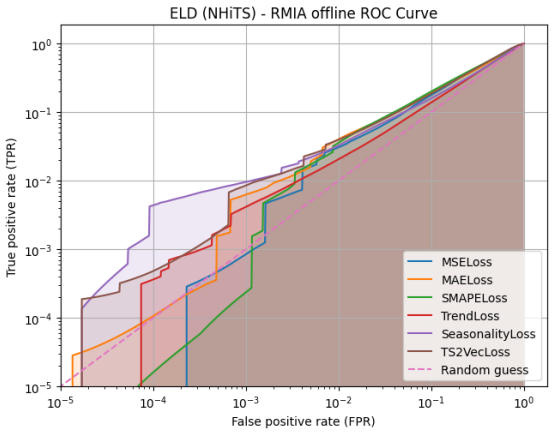


(a) EEG LSTM

(b) EEG NHiTS



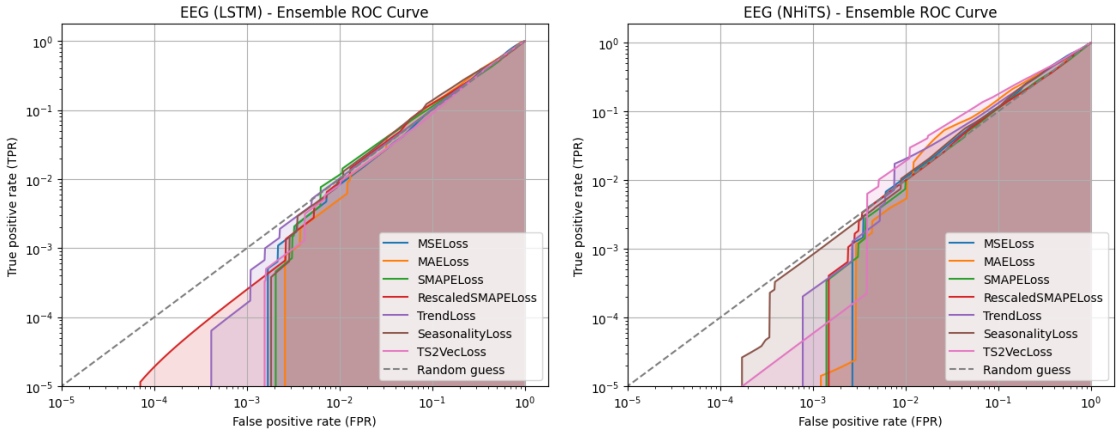
(c) ELD LSTM



(d) ELD NHiTS

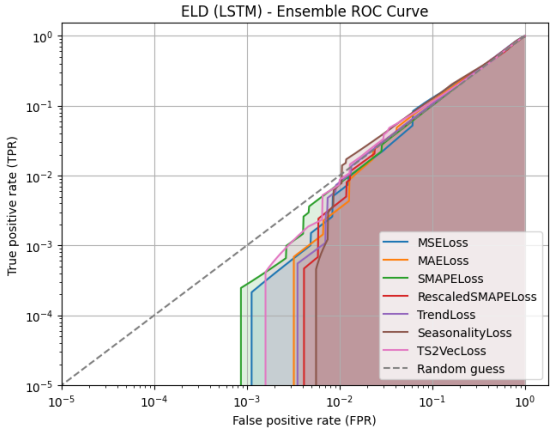
Figure B.6: RMIA offline sample-level ROC Curve, for both datasets and target models.

B.4 Ensemble Attack

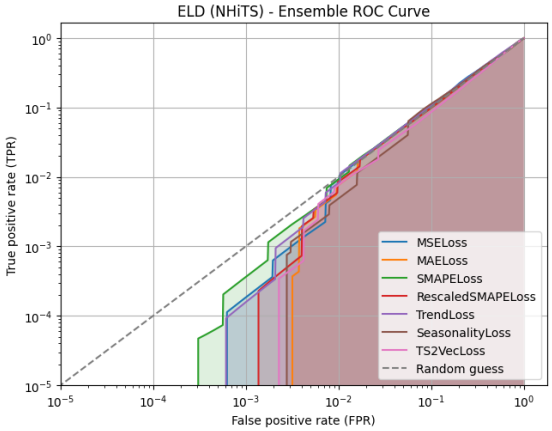


(a) EEG LSTM

(b) EEG NHiTS



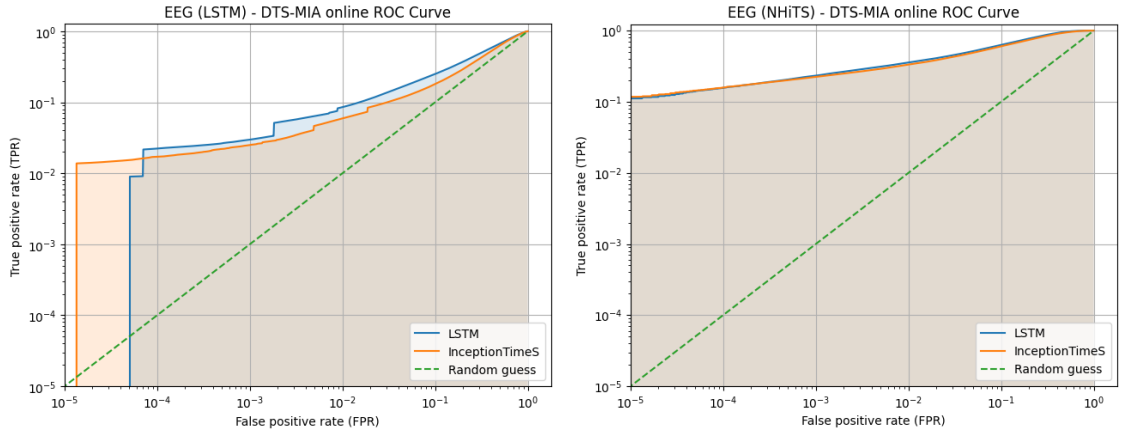
(c) ELD LSTM



(d) ELD NHiTS

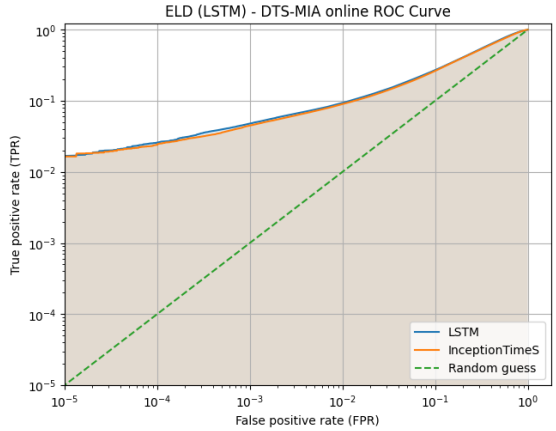
Figure B.7: Ensemble sample-level ROC Curve, for both datasets and target models.

B.5 Deep Time Series Attack

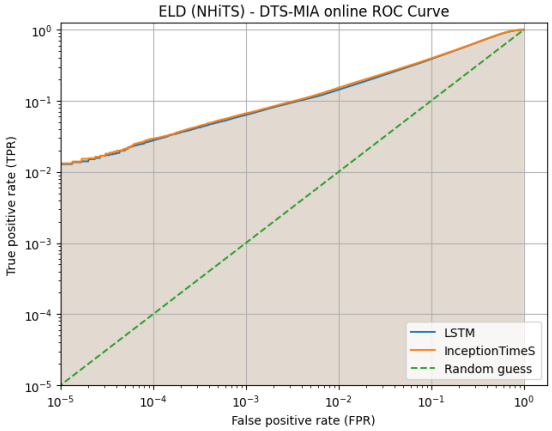


(a) EEG LSTM

(b) EEG NHiTS

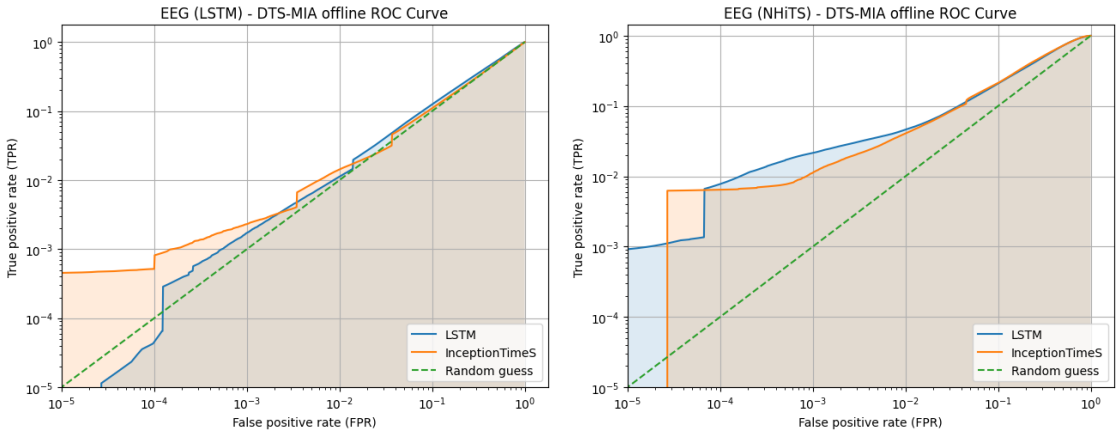


(c) ELD LSTM



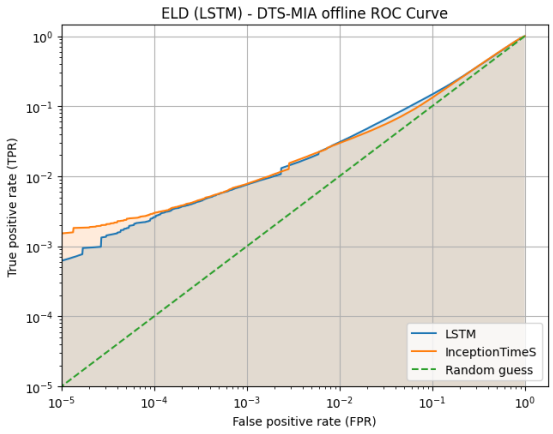
(d) ELD NHiTS

Figure B.8: DTS online sample-level ROC Curve, for both datasets and target models.

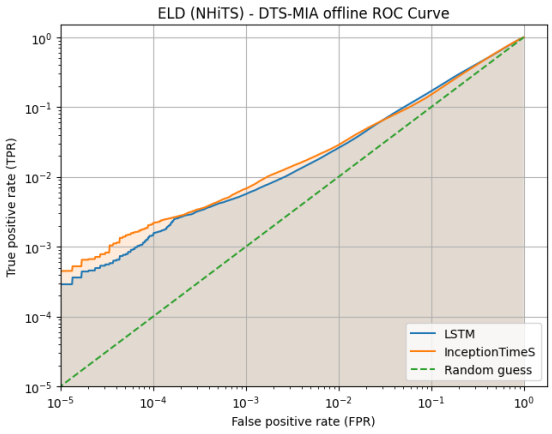


(a) EEG LSTM

(b) EEG NHiTS



(c) ELD LSTM



(d) ELD NHiTS

Figure B.9: DTS offline sample-level ROC Curve, for both datasets and target models.

C

Additional Metrics

In the following sections, the Area Under the ROC Curve (ROC-AUC) and Balanced Accuracy are reported for the various attacks. Each table presents the mean ± standard deviation of both metrics, computed across five runs.

C.1 LiRA

	Model	N-HiTS		LSTM	
	Signal	AUC	Accuracy	AUC	Accuracy
Online	MSE	0.811 ±0.00994	0.726 ±0.0137	0.64 ±0.0333	0.602 ±0.0308
	MAE	0.816 ±0.0102	0.728 ±0.0128	0.646 ±0.0306	0.603 ±0.0299
	SMAPE	0.796 ±0.00889	0.71 ±0.0106	0.647 ±0.0186	0.601 ±0.0232
	Rescaled SMAPE	0.801 ±0.00999	0.713 ±0.0113	0.65 ±0.0202	0.603 ±0.0234
	Trend	0.697 ±0.0204	0.637 ±0.0187	0.552 ±0.0159	0.547 ±0.0172
	Seasonality	0.821 ±0.0112	0.732 ±0.0136	0.646 ±0.0315	0.603 ±0.0312
	TS2Vec	**0.822** ±0.0147	**0.733** ±0.0167	**0.655** ±0.0374	**0.609** ±0.0346
Offline	MSE	0.44 ±0.0284	0.51 ±0.0088	0.445 ±0.0321	0.513 ±0.0111
	MAE	0.495 ±0.0288	0.527 ±0.0176	0.474 ±0.0355	0.515 ±0.00916
	SMAPE	0.521 ±0.0196	0.542 ±0.0101	0.486 ±0.0257	0.514 ±0.0105
	Rescaled SMAPE	**0.563** ±0.0213	**0.562** ±0.0143	**0.518** ±0.0304	**0.524** ±0.0118
	Trend	0.47 ±0.0364	0.523 ±0.0175	0.459 ±0.0548	0.511 ±0.0101
	Seasonality	0.495 ±0.0305	0.53 ±0.0195	0.476 ±0.0381	0.518 ±0.00922
	TS2Vec	0.506 ±0.0182	0.533 ±0.0145	0.48 ±0.0231	0.516 ±0.0115

Table C.1: LiRA attack performance on the TUH-EEG dataset. For each signal, we report the ROC-AUC and balanced accuracy of the attack. Results are shown for both the N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing signal is highlighted in **bold**.

	Model	N-HiTS		LSTM	
	Signal	AUC	Accuracy	AUC	Accuracy
Online	MSE	0.672 ±0.0445	0.622 ±0.0384	0.672 ±0.0519	0.622 ±0.0292
	MAE	0.682 ±0.0433	0.626 ±0.0385	**0.681** ±0.0444	**0.628** ±0.0274
	SMAPE	0.66 ±0.0366	0.613 ±0.0344	0.661 ±0.0366	0.614 ±0.0197
	Rescaled SMAPE	0.668 ±0.0361	0.616 ±0.0338	0.666 ±0.0297	0.617 ±0.0152
	Trend	0.646 ±0.0169	0.606 ±0.0141	0.602 ±0.0397	0.583 ±0.0187
	Seasonality	**0.682** ±0.0437	**0.626** ±0.0382	0.68 ±0.0437	0.628 ±0.0262
	TS2Vec	0.669 ±0.0434	0.619 ±0.0386	0.672 ±0.0495	0.624 ±0.0307
Offline	MSE	0.51 ±0.0689	0.535 ±0.031	0.461 ±0.084	0.528 ±0.0322
	MAE	0.523 ±0.0558	0.537 ±0.0259	0.469 ±0.078	0.529 ±0.0333
	SMAPE	0.495 ±0.0146	0.516 ±0.00844	0.461 ±0.0398	0.516 ±0.0171
	Rescaled SMAPE	**0.532** ±0.0149	0.531 ±0.00868	**0.488** ±0.0414	0.517 ±0.0151
	Trend	0.492 ±0.0654	0.525 ±0.026	0.477 ±0.0632	0.521 ±0.0256
	Seasonality	0.521 ±0.0568	**0.538** ±0.0255	0.469 ±0.0833	**0.531** ±0.039
	TS2Vec	0.5 ±0.0331	0.519 ±0.0165	0.464 ±0.0405	0.515 ±0.0126

Table C.2: LiRA attack performance on the ELD dataset. For each signal, we report the ROC-AUC and balanced accuracy of the attack. Results are shown for both the N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing signal is highlighted in **bold**.

C.2 Multi-Signal LiRA

	Model	N-HiTS		LSTM	
		AUC	Accuracy	AUC	Accuracy
Online		0.847 ±0.0119	0.756 ±0.0132	0.661 ±0.0312	0.613 ±0.031
Offline		0.492 ±0.0278	0.532 ±0.02	0.469 ±0.0378	0.514 ±0.0107

Table C.3: Multi-Signal LiRA attack performance on the TUH-EEG dataset. We report the ROC-AUC and balanced accuracy of the attack. Results are shown for both the N-HiTS and LSTM target models, in both online and offline modes.

	Model	N-HiTS		LSTM	
		AUC	Accuracy	AUC	Accuracy
Online		0.703 ±0.0439	0.643 ±0.0391	0.7 ±0.0464	0.641 ±0.0309
Offline		0.517 ±0.0567	0.537 ±0.0236	0.459 ±0.0812	0.526 ±0.0331

Table C.4: Multi-Signal LiRA attack performance on the ELD dataset. We report the ROC-AUC and balanced accuracy of the attack. Results are shown for both the N-HiTS and LSTM target models, in both online and offline modes.

C.3 RMIA

	Model	N-HiTS		LSTM	
	Signal	AUC	Accuracy	AUC	Accuracy
Online	MSE	0.744 ±0.0201	0.707 ±0.0124	0.602 ±0.0302	**0.598** ±0.0247
	MAE	0.775 ±0.0142	0.711 ±0.0104	0.614 ±0.0384	0.593 ±0.0319
	SMAPE	0.747 ±0.00885	0.689 ±0.00782	**0.614** ±0.0233	0.59 ±0.0208
	Trend	0.61 ±0.0121	0.59 ±0.014	0.527 ±0.0122	0.537 ±0.023
	Seasonality	0.778 ±0.0282	**0.718** ±0.0126	0.594 ±0.0593	0.591 ±0.0405
	TS2Vec	**0.782** ±0.0173	0.715 ±0.0168	0.609 ±0.0515	0.588 ±0.0402
Offline	MSE	0.724 ±0.0392	0.674 ±0.0317	0.631 ±0.0395	**0.609** ±0.0242
	MAE	0.744 ±0.0443	0.681 ±0.0341	**0.637** ±0.0406	0.609 ±0.0261
	SMAPE	0.665 ±0.0334	0.616 ±0.021	0.588 ±0.0265	0.564 ±0.0187
	Trend	0.588 ±0.0203	0.562 ±0.0123	0.551 ±0.0267	0.539 ±0.0142
	Seasonality	0.706 ±0.0423	0.658 ±0.038	0.622 ±0.0262	0.599 ±0.0142
	TS2Vec	**0.748** ±0.0488	**0.683** ±0.037	0.634 ±0.0416	0.599 ±0.0322

Table C.5: RMIA attack performance on the TUH-EEG dataset. For each signal, we report the ROC-AUC and balanced accuracy of the attack. Results are shown for both the N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing signal is highlighted in **bold**.

	Model	N-HiTS		LSTM	
	Signal	AUC	Accuracy	AUC	Accuracy
Online	MSE	0.646 ±0.0454	0.618 ±0.0416	0.658 ±0.0362	**0.629** ±0.0248
	MAE	**0.665** ±0.047	**0.62** ±0.0401	**0.669** ±0.0322	0.624 ±0.0189
	SMAPE	0.628 ±0.0554	0.603 ±0.0298	0.636 ±0.0207	0.602 ±0.0131
	Trend	0.581 ±0.019	0.561 ±0.02	0.559 ±0.00678	0.548 ±0.00504
	Seasonality	0.66 ±0.0483	0.614 ±0.0362	0.666 ±0.0312	0.62 ±0.0162
	TS2Vec	0.648 ±0.0414	0.603 ±0.031	0.654 ±0.0316	0.61 ±0.0172
Offline	MSE	0.592 ±0.0326	**0.58** ±0.0169	0.593 ±0.0466	**0.588** ±0.0212
	MAE	0.593 ±0.0515	0.572 ±0.0285	0.58 ±0.0498	0.574 ±0.0288
	SMAPE	0.6 ±0.0135	0.575 ±0.00922	0.593 ±0.0358	0.575 ±0.0246
	Trend	0.55 ±0.0534	0.544 ±0.0311	0.535 ±0.0394	0.537 ±0.0229
	Seasonality	0.558 ±0.0767	0.558 ±0.0359	0.537 ±0.0699	0.555 ±0.0299
	TS2Vec	**0.607** ±0.0223	0.575 ±0.0156	**0.599** ±0.036	0.579 ±0.0225

Table C.6: RMIA attack performance on the ELD dataset. For each signal, we report the ROC-AUC and balanced accuracy of the attack. Results are shown for both the N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing signal is highlighted in **bold**.

C.4 Ensemble Attack

Model	N-HiTS		LSTM	
Signal	AUC	Accuracy	AUC	Accuracy
MSE	0.546 ±0.0823	0.565 ±0.0611	0.53 ±0.042	**0.549** ±0.0297
MAE	0.532 ±0.0909	0.559 ±0.0645	**0.532** ±0.0525	0.545 ±0.0304
SMAPE	0.491 ±0.0476	0.524 ±0.0199	0.495 ±0.0312	0.52 ±0.0184
Rescaled SMAPE	0.494 ±0.0454	0.523 ±0.0217	0.499 ±0.0392	0.518 ±0.0104
Trend	0.537 ±0.0291	0.542 ±0.02	0.506 ±0.03	0.518 ±0.02
Seasonality	0.538 ±0.0911	0.559 ±0.0625	0.529 ±0.0483	0.542 ±0.0283
TS2Vec	**0.555** ±0.111	**0.58** ±0.0714	0.507 ±0.0795	0.537 ±0.0412

Table C.7: Ensemble attack performance on the TUH-EEG dataset. For each signal, we report the ROC-AUC and balanced accuracy of the attack. Results are shown for both the N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing signal is highlighted in **bold**.

Model	N-HiTS		LSTM	
Signal	AUC	Accuracy	AUC	Accuracy
MSE	0.504 ±0.0908	**0.541** ±0.036	**0.509** ±0.0764	0.544 ±0.0288
MAE	0.503 ±0.0709	0.537 ±0.0383	0.5 ±0.0767	**0.545** ±0.0307
SMAPE	0.496 ±0.0377	0.52 ±0.0159	0.482 ±0.0711	0.527 ±0.0251
Rescaled SMAPE	0.493 ±0.0385	0.516 ±0.017	0.484 ±0.0648	0.529 ±0.0265
Trend	**0.509** ±0.0622	0.53 ±0.0327	0.496 ±0.0599	0.527 ±0.0268
Seasonality	0.484 ±0.0905	0.537 ±0.0387	0.502 ±0.0719	0.539 ±0.0308
TS2Vec	0.477 ±0.0599	0.525 ±0.0294	0.499 ±0.075	0.537 ±0.0339

Table C.8: Ensemble attack performance on the ELD dataset. For each signal, we report the ROC-AUC and balanced accuracy of the attack. Results are shown for both the N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing signal is highlighted in **bold**.

C.5 Deep Time Series Attack

	Model	N-HiTS		LSTM	
	MIC Model	AUC	Accuracy	AUC	Accuracy
Online	LSTM	**0.884** ±0.0303	**0.789** ±0.0361	**0.637** ±0.0378	**0.61** ±0.0242
	InceptionTime	0.866 ±0.0389	0.774 ±0.0406	0.597 ±0.054	0.585 ±0.0337
Offline	LSTM	0.653 ±0.0288	0.613 ±0.0194	**0.529** ±0.0355	**0.533** ±0.0227
	InceptionTime	**0.66** ±0.0503	**0.619** ±0.025	0.504 ±0.0507	0.524 ±0.0207

Table C.9: DTS attack performance on the TUH-EEG dataset. For each MIC classifier model, we report the ROC-AUC and balanced accuracy of the attack. Results are shown for both the N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing MIC classifier is highlighted in **bold**.

	Model	N-HiTS		LSTM	
	MIC Model	AUC	Accuracy	AUC	Accuracy
Online	LSTM	0.754 ±0.0131	0.684 ±0.00776	**0.667** ±0.069	**0.618** ±0.0444
	InceptionTime	**0.758** ±0.0137	**0.685** ±0.011	0.658 ±0.081	0.612 ±0.0574
Offline	LSTM	**0.574** ±0.0395	**0.564** ±0.0274	0.549 ±0.0312	0.538 ±0.0202
	InceptionTime	0.571 ±0.0296	0.56 ±0.0218	**0.55** ±0.0265	**0.539** ±0.0171

Table C.10: DTS attack performance on the ELD dataset. For each MIC classifier model, we report the ROC-AUC and balanced accuracy of the attack. Results are shown for both the N-HiTS and LSTM target models, in both online and offline modes. In each column, the best-performing MIC classifier is highlighted in **bold**.