Privacy Preserving Anomaly Detection in Time Series Data with Differential Privacy

Juan Diego Prada

December 2024

Thesis Advisor: Juan F. Pérez

1 Introduction

Energy use is critical across modern society, powering everything from our homes to industrial complexes. However, ensuring we use energy efficiently and reliably presents challenges, particularly in identifying and addressing abnormal consumption patterns. These anomalies can be caused by various sources, such as faulty equipment, human error, or even deliberate actions, resulting in wasted energy and potential threats to infrastructure stability and sustainability [15]. Anomalies not only impact energy providers but also affect consumers. For instance, detecting unusually high energy consumption when it should not occur can lead to additional costs for users. Conversely, registering abnormally low consumption levels can result in revenue loss for energy companies. Therefore, the development of effective anomaly detection methods in energy consumption has become a critical area of research, aimed at mitigating inefficiencies and enhancing the resilience of energy systems [12]. Energy consumption is recorded at regular intervals, resulting in a timestamped history of energy usage over time. These structured data forms a time series, where each timestamp corresponds to a recorded value, allowing for the analysis of energy consumption patterns and trends over time.

A time series is a sequence of observations recorded over time, arranged in chronological order. It typically consists of three main components: trends, seasonal patterns, and residuals. Time series analysis is valuable for understanding how variables evolve over time and can aid in forecasting future values. Additionally, it can help detect anomalies or unusual occurrences in the data, such as sudden spikes or drops that deviate significantly from the expected patterns. Various methods can be employed to analyze time series data, and in this paper, we will focus on using neural networks for forecasting. Specifically, we will utilize the recurrent neural network, Long Short-Term Memory (LSTM) model, to forecast the behavior of the energy consumption time series.

Long Short-Term Memory (LSTM) [13] is a type of recurrent neural network that specializes in learning from sequential data by selectively retaining useful information from previous time steps while discarding irrelevant details. This selective memory is facilitated through the use of specialized memory cells with gating mechanisms. These gates, including the input gate and the forget gate, control the flow of information, allowing the LSTM network to distinguish between relevant and irrelevant past information [3]. By leveraging this capability, LSTMs can effectively capture both short-term and long-term dependencies in sequential data, enabling them to identify complex patterns. This makes LSTMs well-suited for various tasks involving sequential data, such as speech recognition, machine translation, and time series forecasting.

Due to their ability to capture both short-term and long-term dependencies, LSTM networks are well-suited for evaluating energy consumption anomalies over time. This is because we need to observe the immediate past to understand consumption behavior while also considering the historical behavior of clients to make accurate predictions. However, when using such models, it is crucial to prioritize user privacy. While these models require the users data for prediction purposes, it is essential to implement measures to protect users privacy, as their information should not be compromised or filtered.

In recent years, data has become one of the most utilized and sought-after assets across all sectors, playing a fundamental role in both business and daily life. Data privacy and information security have thus become significant concerns in terms of data analysis. In line with the increment in data analysis models, security issues have arisen. We willingly provide our

data while also relying on it in our everyday activities. Effectively managing these data is useful as it help us to comprehend patterns and behaviors. Often, databases contain critical information that could be used to create relevant models capable of addressing problems that helps understand relations among observed processes. However, the use of such databases is restricted due to the presence of sensitive information that must be safeguarded such as medical histories, financial records, or other private details individuals wish to keep confidential. Consequently, ensuring data privacy is crucial. Any unauthorized disclosure of sensitive information could expose individuals to various risks, leaving them vulnerable to potential attacks by third parties. A number of solutions to anonymize data to protect users have been proposed. Techniques such as K-anonymity [6] involve removing all columns that indicate a user's identity, such as name, phone number or address. This process is aimed to obtain anonymous databases. While the initial approach of anonymizing data by removing identifiable details may seem straightforward, the challenge arises from the potential of re-identifying individuals through the combination of data from multiple sources, even in the absence of explicit personal identifiers. It has been shown [8] that this method is not as effective as it seems; an attack can be performed to identify people in the database, making it useless for any private predictive model because the protection of the information cannot be guaranteed.

On the other hand, eliminating all potentially sensitive data to prevent tracking individuals could present another challenge. This approach might result in the removal of information that, despite being sensitive, is essential to comprehend the data behavior and derive accurate conclusions during the analysis phase. Such omission could lead to flawed or uncertain models that depend on this information to be effective. This is where the value of differential privacy tools becomes apparent, as these mitigate such types of attacks.

Differential privacy, proposed by Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith in 2006, is a method used to address privacy concerns by offering mathematically guaranteed protections. This approach ensures that the level of privacy can be quantified through mathematical proofs, allowing for adjustments to enhance or diminish privacy levels. It essentially operates as a noise perturbation mechanism, manipulating data by adding noise to make it sufficiently different for an attacker to identify any user, yet statistically, the output does not significantly vary, providing results as trustworthy as if no noise were added. The three main parameters that are used in differential privacy are ϵ, δ and Δf , ϵ is a parameter that indicates how private a dataset will be, it regulates how much noise will be added.A smaller ϵ will yield better privacy (and less accurate responses). δ is really an additional parameter, it will provide more noise if necessary, but it is optional, is not necessary to use it when doing differential privacy.lastly, the sensitivity or Δf measures how much the output of a function f can change when a single individual's data point is added or removed from the dataset, helping to determine the amount of noise required to achieve a desired level of privacy when analyzing data. Differential privacy works mostly by adding noise to the data, creating a perturbation. There are different methods to do this; the noise can be obtained from the Gaussian, Exponential, Binomial, Laplace distribution, among others [9]. In this work, we will use the Laplace mechanism, which involves adding noise to the data drawn from the Laplace distribution with parameter $Lap(0, \epsilon/\Delta f)$ Differential privacy addresses the paradox of learning nothing about an individual while learning useful information about a population [8]. Moreover, it ensures that any result obtained from an experiment conducted with the data remains consistent regardless of the addition or removal of individuals from the database.

Differential privacy is formally defined as follows [8]: A randomized algorithm M with domain $N^{|X|}$ is (ϵ, δ) -differentially private if for all S $\subset$ $Range(M)$ and for all $x, y \in N^{|x|}$ such that $||x - y||_1 \leq 1$:

$$P[M(x) \in S] \leq e^{\epsilon} P[M(y) \in S] + \delta$$

Post-processing is a crucial concept in differentially private data. It states that if a dataset is differentially private and there is no new additional information gained about the database and the original data is already private, then applying any algorithm or function to the data will not reduce its privacy. Therefore, regardless of the method an attacker employs to analyze the data, it will remain as private as it was initially. Formally, let $M : N^{|X|} \rightarrow R$ be a randomized algorithm that is $(\epsilon, \delta)-$ differentially private. let $f : R \rightarrow R'$ be an arbitrary randomized mapping. Then $foM : N^{|X|} \rightarrow R'$ is $(\epsilon, \delta)-$ differentially private

2 Objectives

2.1 General Objective

Develop a recurrent neural network (LSTM) for a time series that can predict energy consumption and detect any anomalous behavior, while preserving data privacy by implementing differential privacy techniques. These techniques will ensure data protection while striving to maintain the model's maximum utility

2.2 Specific objectives

- Conduct a state-of-the-art review of the machine learning problem with differential privacy, analyzing various methodologies and variants used in the literature, as well as the most frequently used algorithms. It will help implement a useful algorithm for time series analysis that achieves the best possible results
- Select, implement, and validate multiple predictive models with the capability to accurately detect anomalies in time series data.
- Develop an algorithm that correctly incorporates differential privacy in a dataset, ensuring the predictive model maximizes its utility.
- Compare the utility of the predictive models under different parameter settings of differentially private learning algorithms.

3 Literature Review

Existing differential privacy mechanisms for time series have been published in recent years by various authors. Many approaches have been developed to ensure data privacy. These approaches can be divided into three categories: Event Level, User Level, and Window – Event Level [7]. Event level methods aim to protect either a single timestamp or a single event within the time series. These methods are useful when the objective is to safeguard a single query or timestamp. User level methods, on the other hand, aim to protect all timestamps and data associated with each user. This means that these methods can protect data against any query and can be used to reveal all user information. The Window - event Level method represents a fusion of the preceding approaches. It exclusively protects a defined time window within the data. This necessitates the establishment of a time window definition, after which the method safeguards data within that specific time frame. Although there are many papers and previous works related to time series, using them for anomaly detection in energy consumption presents some complications.

The use of Geo- Indistinguishability mechanisms [2] proposes an adaptive location preserving mechanism that adjusts the amount of noise required to obfuscate the users location based on the correlation level with its previous location. It relies on adjusting only the noise to improve privacy for the next time step, based on the correlation with the previous moment. Our work, on the other hand, deals with datasets that correlate with more than one preceding instance, requiring a more comprehensive strategy.

Just as in the previous example, Rui Chen, in his paper 'Publication of Differentially Private Sequential Data via n-grams,' extracts the essential information of a sequential database in terms of a set of variable-length n-grams [4]. He works with sequential data, but focuses on things such as phrases. Therefore, there is high dimensionality and significant sequentiality in these problems. He creates an algorithm to implement differential privacy by using tree structures and a set of novel techniques based on the Markov assumption to add noise. While these approaches succeed in reducing the magnitude of added noise, they are not suitable for identifying anomalies in time series analysis. Although effective for sequential data such as words or phrases, implementing them becomes challenging with the type of data we are working with the anomaly detection context. The method estimates the probability of the next item given a sequence of n-1, relying on Markov independence, and is particularly effective when n is small. The limitation is very similar to the previous example; it works with few previous data, and the use of Markov and tree models cannot be applied in our context.

Farhad Farokhi [11] introduced a method for utilizing evolving datasets over an infinite horizon, termed discounted differential privacy. His work is based on the idea that when a time series is made diferentially private, the amount of noise must be uniformly distributed across all series and as long as the data grows, there is an increase in noise diminishing the data utility for analysis. The problem this paper solves is that the magnitude of privacy preserving additive noise must grow without bound to ensure differential privacy over an infinite horizon. The author's solution suggests that privacy losses incurred in the distant past are considered less significant than

those in the present for individuals. They propose employing exponential and hyperbolic discounting mechanisms to mitigate privacy loss over time under continuous observations. Consequently, in this solution, observations from the distant past are granted less privacy compared to current ones, this means that the noise added to the observations of the distant past is null or undetectable. However, a significant concern with this approach arises from the underlying assumption that behavioral patterns from the past have shifted, resulting in less relevant data and therefore reduced privacy concerns. This assumption, however, may not universally hold true, making the adoption of this technique potentially risky.

Liyue Fan and Li Xiong propose the FAST algorithm, a novel framework for releasing real-time aggregate statistics under differential privacy, which relies on filtering and adaptive sampling [10]. The problem they are trying to solve is that a challenge arises when working with extensive databases, particularly due to the "composition theorem," which complicates the implementation of differential privacy when consecutive data points are correlated. To address this issue, they adopt a strategy of adding Laplace distribution noise to different sections of the database through random samplings and perturbations, this process is iteratively applied with filtering until the initial privacy budget is exhausted, which means that not all the dataset will become private, just the fragments that are randomly selected. However, a drawback of this method is its assumption that certain parts of the time series can remain unperturbed and not subject to differential privacy constraints, which raises a concern regarding the privacy across the entire time series.

PeGaSus [5] is an algorithm that can simultaneously support a variety of stream processing tasks– counts, sliding windows, event monitoring– over multiple resolutions of the streams. It uses a Perturber to release noisy counts, a data-adaptive Perturber to identify stable uniform regions in the stream, and a query specific Smoother, which combines the outputs of the Perturber and Grouper to answer queries with low error.

4 Methodology

This section introduces the methodology to be employed for identifying an approach to predict anomalies in time series forecasting, involving two distinct methods. Both methods are applicable to any time series analysis incorporating differential privacy. The first method involves ensuring the data's differential privacy during the model training phase. This entails providing raw data to the forecasting model while simultaneously applying the differential privacy algorithm. In contrast, the second approach focuses on making the time series differentially private before passing the data through the forecasting model, utilizing the concept of post-processing. The objective is to evaluate the performance of both models and determine which method is better based on the obtained results.

4.1 Stochastic gradient descent perturbation

In stochastic gradient descent perturbation, the privacy guarantee is established during the model training phase. This is because applying noise directly to the model's parameters after training can significantly impair the model's effectiveness [1]. Therefore, it is crucial to adopt a more cautious approach where modifications are made during the training process, particularly in the stochastic gradient descent algorithm. This perturbation process is integrated into the loss function, a crucial component of any neural network. The loss function quantifies the penalty for discrepancies between the model's predictions and the training data, with the aim of minimizing this error. Gradient descent is a commonly used method in which, at each step, the gradient is computed to update the model's parameters in the direction of a local minimum of the error.

The work will be done on TensorFlow, where the SGD DP will be implemented as it is described in Heber Hwang and Jean-François Couchot paper[3]

4.2 Adding correlated noise to time series

The correlated time series differential privacy (CTS-DP) was introduced by Hao Wang and Zhengquan Xu [16] with the intention of publishing time series data. Despite this, they did not evaluate the forecasting results using this correlated data. In their work, they establish that when there is noise added to each timestamp in a time series, it can be removed by utilizing a refinement method. For instance, consider a time series where $Y_t = T_t + S_t + R_t$ where Y_t represents each timestamp, T_t is the trend component, S_t is the seasonality component and R_t is the residual. When noise is added, it affects only the residual component; if the time series is decomposed into these components, then the trend and seasonality components can be exposed by removing

the noise from the residual. Following their method, Series-Indistinguishability guarantees that the perturbed outputs of two adjacent datasets, differing by only one record, are indistinguishable to potential adversaries (Wang et al., 2017). This noise cannot be IID (independent and identically distributed), so the solution to this problem is to create a correlated Laplace mechanism. This mechanism considers the autocorrelation of the time series and generates noise that follows the same order as the original data. Therefore, if both series are the same, the noise-added series and the original series are indistinguishable to any adversary.

4.3 Results Comparison

Once both methods are implemented, two crucial factors will determine their effectiveness. Firstly, the anomaly detection capability will assess how well the model detects anomalies within the forecasted data. Secondly, the data privacy aspect will evaluate how the quality of results fluctuates with changes in the privacy budget. It is essential to observe how both models behave as the privacy level increases or decreases. The objective is to evaluate the strengths and weaknesses of both approaches.

5 Results

When attempting to replicate the correlated time series differential privacy mechanism proposed by Wang and Xu [16], we encountered several challenges that ultimately made it impossible to fully replicate their method. The primary issue stems from a key assumption in their approach: it requires the time series subjected to differential privacy to exhibit strong autocorrelation throughout its entirety. This assumption is critical because, in the proposed algorithm, the data owner generates four white noise series that are then passed through a filter designed to match the autocorrelation of the original time series. The complete algorithm is outlined in Algorithm 1. However, due to various factors, the original time series may contain unexpected noise. As a result, when applying Algorithm 1, the autocorrelation function of the original time series will not match that of the noise series; while it may be similar, it will not be identical. The solution they propose is to introduce a noise-tolerant interval that preserves the robustness of the solution.

$$-\mu \leq \frac{R_Z(\tau)}{R_{XX}(\tau)} \leq \mu$$

It is implied that the autocorrelation functions of the noise series and the original series are indistinguishable to an adversary up to the limit of e^{μ}. The problem with the original series being correlated arises with this noise-tolerant interval. When the original autocorrelation R_{xx} is close to zero, then μ will be extremely high,if μ is a large number, it means that the privacy level will decrease, making it easier to deduce the data because the privacy level is being reduced. To eliminate this problem, the solution is for the original data to be highly autocorrelated, but this is not always possible. When a time series is long enough, the autocorrelation will inevitably decrease when many lags are included. For example, consider an AR(1) (autoregressive order 1) time series, which is described as:

$$X_t = \phi * X_{t-1} + \epsilon$$

where ϕ is the coefficient that measures the relationship between the current value of the time series and its value in the previous period, when ϕ is closer to 0, there will be less autocorrelation, and when it is closer to 1, the time series will be highly autocorrelated. Here ϵ is the noise added. The autocorrelation at lag k is calculated as:

$$\rho(k) = \phi^k$$

This means that the autocorrelation decays exponentially as k increases. Even if ϕ is 0.99, the term ϕ^k will eventually approauch zero as k increases. The problem becomes clear: if a time series has enough timesteps, the autocorrelation function will eventually have enough lags where the autocorrelation is effectively zero. When applying CTS-DP, there will be timesteps with little to no privacy, making them vulnerable to potential attacks. As a result, the entire dataset could be compromised, as having enough timesteps would allow an adversary to identify the complete time series. In this paper, we aim to make time series predictions using LSTM networks. For these neural networks to function effectively, a significant amount of data is required to train the model, meaning that the time series must contain many data points. Consequently, the challenge becomes more pronounced when attempting to apply the CTS-DP mechanism. This creates a counterproductive effect, while a robust model requires a large dataset, ensuring proper privacy through CTS-DP limits the size of the time series. Therefore, the conflict between the need for extensive data in neural networks and the constraints of privacy protection becomes evident.

On the other hand, there is a second problem when applying the CTS-DP algorithm, which occurs when the auto correlation function of the added noise R_Z,is as close to zero as possible. When it approaches zero, μ will also approach zero; consequently, this means that there is no auto correlation in the noise. This implies that we are adding normal white noise, rendering the algorithm ineffective. Normal white noise can be easily removed by a refinement method, as mentioned in Section 4.2. Therefore, when μ is zero, the algorithm fails to provide privacy, as it effectively adds normal white noise that cannot guarantee privacy. Thus, the CTS-DP is not valid in these cases.

The third problem is related to the properties of differential privacy. The parameter ϵ controls the trade-off between privacy and accuracy. If ϵ is small, it implies stronger privacy, as more noise is added to the data to obscure individual information. Conversely, if ϵ is large, less noise is added, and the privacy guarantee weakens. In other words,ϵ is inversely proportional to the level of noise added for privacy.

In the methodology established in the paper, the Gaussian noise used must be distributed as $N(0, \sqrt{2\lambda})$ where $\lambda = \frac{\Delta f}{\epsilon}$. This means that when ϵ is small, λ becomes large, resulting in a higher variance of the noise. Consequently, the added noise will also be greater due to this increased variance. At this point,

the properties of differential privacy are being satisfied. The issue emerges when the Gaussian noise is passed through the filter intended to match the auto correlation of the original time series. The filter that must be applied is:

$$h(\tau) = \sqrt{\frac{R_{xx}(\tau)}{16\pi N_0}}$$

where N_0 is the power spectral density (PSD) of the Gaussian noise, By definition, the PSD of Gaussian noise is its variance, allowing the filter to be redefined as:

$$h(\tau) = \sqrt{\frac{R_{xx}(\tau)}{16\pi\sqrt{2\lambda}}}$$

This implies that if the variance is large, the value of the filter $h(\tau)$ will be small. When the filter is small, passing the Gaussian noise through it reduces the magnitude of the noise. This result is contradictory because the intended property of differential privacy dictates that, when ϵ is small, the noise should be large. However, the filter reduces the noise, effectively violating this principle of differential privacy.

One of the methods used in this paper is Tensorflow Privacy. As mention in section 4.1, this method is based on the stochastic gradient descent, where there is a modification in this process that will make the predictions private.

In the context of stochastic gradient descent (SGD) the process typically follows these steps: First, a mini-batch of training points (X,Y) is sampled, where X represents the input data and Y the corresponding labels. Next, the loss function $L(\theta, X, Y)$ is calculated, measuring the difference between the model's predictions F_θ and the actual labels Y, with θ representing the models parameters. This loss quantifies the extend to which the models predictions deviate from the true values. The following step involves computing the gradient of the loss with respect to the model parameters θ. This gradient indicates both the magnitude and direction in which the model parameters should be adjusted to minimize the loss. Finally, the gradients are scaled by the learning rate and used to update the model parameters guiding the model to improve its predictive accuracy.

To achieve a differentially private stochastic gradient descent (DP-SGD), some modifications must be made to the original method. First, it is necessary to bound the sensitivity of each gradient by applying gradient clipping. This step limits the influence of individual training points on the computed gradient, preventing any single data point from exerting too much impact on the parameter updates. Regardless of the gradient's magnitude, this clipping constrains its effect, meaning that the model may require more steps to reach an optimal solution. Gradient clipping also ensures that, if a specific example is disproportionately affecting the model's gradient, it will not dominate the training process, enhancing both privacy and robustness.The second modification involves randomizing the algorithm's behavior to make it statistically impossible to determine whether a specific data point was included in the training set. This is achieved by adding randomly sampled noise to the clipped gradient. By introducing this noise, DP-SGD maintains the privacy of individual data points while preserving the model's ability to generalize effectively.

The steps required to perform Differentially private stochastic gradient descent (SGD) are as follows: First, sample a mini-batch of training points (X,Y). where X represents the input and Y the corresponding labels. Dividing the batch into mini-batches helps limit the individual impact of each sample. Next, compute the loss function $L(\theta, X, Y)$ between the models prediction and the label Y. Then, calculate the gradient of the loss function with respect to the model parameters. Each gradient is then clipped per training example in the mini-batch to ensure it has a bounded Euclidean norm. Add random noise to the clipped gradients, and average all gradients within each mini-batch to obtain a single gradient. Finally, scale this averaged gradient by the learning rate and apply it to update the model parameters

This algorithm is implemented using the TensorFlow Privacy library. To apply it, the neural network's optimizer must be replaced with the TensorFlow Privacy optimizer, which requires three key parameters. First, the norm clip parameter defines the maximum Euclidean norm each gradient can have, limiting its impact on the model. Next, the noise multiplier controls the amount of noise added to the gradients before being applied by the optimizer, enhancing privacy. Finally, the number of microbatches specifies how the batch is divided, and it should be a divisor of the batch size to ensure an integer division.

TensorFlow Privacy has some limitations and disadvantages that need to be taken into account. First, due to norm clipping, the algorithm is computationally much heavier than TensorFlow without privacy and may require more steps to minimize the loss function. The second issue is that this algorithm only en-

sures the privacy of the predictions, meaning that if you want to view any original data, this will not be possible as it is not private. Therefore, the amount of accessible data is limited to the predictions.Also, due to the micro batch, this stablish a new problem, because if there is a model that has been done without tensorflow privacy and know it wants to transform it an make it private, the number of batches that is going to be used, needs to be a divisor of the number of data that will be used in the training model, so this limits the number of batches that can be used because it can not let any batch incomplete, all the batches need to have the same amount of data, this doesnt happen in normal tensorflow, so with this restriction, the number of batches will be limited. Additionally, the use of microbatches introduces a new challenge. When converting an existing model to use TensorFlow Privacy, the batch size must be a divisor of the total number of training samples to ensure that all batches contain the same amount of data. This requirement limits the flexibility in choosing batch sizes, as each batch must be complete, unlike in standard TensorFlow where incomplete batches are allowed. As a result, the number of viable batch sizes is more restricted when applying TensorFlow Privacy

The last method that was used, is based on kim and Woo paper [14] were differentially private time series are created by exploring decomposition methods. in particular they use the seasonal trend decomposition using Loess (STL). This method extract the trend and the seasonal component, the idea is that if they transform those components with the fourier transformation and take them in the frequency domain, they could perform the fourier perturbation algorithm that is just the laplace mechanism but applied to the fourier coefficients. They say that noises injected in the frequency domain are no longer independent but are correlated and they assure this method will provide differentually private time series. The biggest problem with this method, is that there is no mathematical proof that this method is indeed private, so there is no evidence that this method accomplish differential privacy.

The last method used is based on the paper by Kim and Woo [14], which explores the creation of differentially private time series through decomposition methods. In particular, they utilize Seasonal-Trend Decomposition using LOESS (STL). This method extracts the trend and seasonal components; the idea is that by transforming these components with a Fourier transformation and analyzing them in the frequency domain, they can apply the Fourier Perturbation Algorithm, which is essentially the Laplace mechanism applied to the Fourier coefficients. They note that the noise injected in the frequency domain is no longer independent but correlated, and they assert that this method will produce differentially private time series. However, the main issue with this approach is the lack of mathematical proof confirming that it indeed guarantees privacy, which means there is no evidence that this method achieves differential privacy.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] Raed Al-Dhubhani and Jonathan M Cazalas. An adaptive geo-indistinguishability mechanism for continuous lbs queries. *Wireless Networks*, 24(8):3221–3239, 2018.

[3] Héber Hwang Arcolezi, Jean-François Couchot, Denis Renaud, Bechara Al Bouna, and Xiaokui Xiao. Differentially private multivariate time series forecasting of aggregated human mobility with deep learning: Input or gradient perturbation? *Neural Computing and Applications*, 34(16):13355–13369, 2022.

[4] Rui Chen, Gergely Acs, and Claude Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 638–649, 2012.

[5] Yan Chen, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Pegasus: Data-adaptive differentially private stream processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1375–1388, 2017.

[6] Damien Desfontaines. A friendly, non-technical introduction to differential privacy. https://desfontain.es/blog/friendly-intro-to-differential-privacy.html, 09 2021. Ted is writing things (personal blog).

[7] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724, 2010.

[8] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[9] Ahmed El Ouadrhiri and Ahmed Abdelhadi. Differential privacy for deep and federated learning: A survey. *IEEE access*, 10:22359–22380, 2022.

[10] Liyue Fan and Li Xiong. An adaptive approach to real-time aggregate monitoring with differential privacy. *IEEE Transactions on knowledge and data engineering*, 26(9):2094–2106, 2013.

[11] Farhad Farokhi. Temporally discounted differential privacy for evolving datasets on an infinite horizon. in 2020 acm/ieee 11th international conference on cyber-physical systems (iccps). *IEEE, 1śś8*, 2020.

[12] Marta Moure Garrido, Celeste Campo, and Carlos García Rubio. Anomalies detection using entropy in household energy consumption data. In *Intelligent Environments 2020 Workshop Proceedings of the 16th International Conference on Intelligent Environments*, pages 311–320. IOS Press, 2020.

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

[14] Kyunghee Kim, Minha Kim, and Simon S Woo. Stl-dp: Differentially private time series exploring decomposition and compression methods. In *CIKM Workshops*, 2022.

[15] Haipeng Pan, Zhongqian Yin, and Xianzhi Jiang. High-dimensional energy consumption anomaly detection: A deep learning-based method for detecting anomalies. *Energies*, 15(17):6139, 2022.

[16] Hao Wang and Zhengquan Xu. Cts-dp: Publishing correlated time-series data via differential privacy. *Knowledge-Based Systems*, 122:167–179, 2017.