Methods and Techniques for Efficient Processing of Aggregated Data

by

Fan Yang

B.S., Dalian University of Technology, 2014

M.S., Worcester Polytechnic Institute, 2016

Submitted to the Graduate Faculty of

the School of Computing and Information in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2022

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION

This dissertation was presented

by

Fan Yang

It was defended on

March 18, 2022

and approved by

Vladimir I. Zadorozhny, University of Pittsburgh

Christos Faloutsos, Carnegie Mellon University

Paul Munro, University of Pittsburgh

Konstantinos Pelechrinis, University of Pittsburgh

Methods and Techniques for Efficient Processing of Aggregated Data

Fan Yang, PhD

University of Pittsburgh, 2022

With the explosion of information, massive amounts of data are being generated daily from different sources. Due to the limited infrastructure and human capacity for data integration and the requirement of efficient processing, some data, especially historical data, are stored in an aggregated form at different levels of aggregation (e.g., aggregated by different time intervals). For example, epidemiological data preserves *monthly* counts of infected people. Meanwhile, data analysis and machine learning models often require elaborate knowledge of data for accurate analysis and prediction. This information should be obtained either from original or from aggregated data.

Motivated by the above challenge, this thesis aims to facilitate the generation and utilization of aggregated data from three aspects: 1) reconstructing higher-resolution time series from aggregated data with *acceptable performance*; 2) selecting aggregated data for analysis with *minimal hurt for performance*, e.g., detecting outbreaks of measles using monthly counts may have comparable performance with the raw data; 3) generating aggregated data for future studies with *less information loss*, e.g., aggregating data with different resolutions on different parts based on the importance.

Most data reconstruction methods utilize domain knowledge, e.g., smoothness, periodicity, or sparsity, to improve reconstruction accuracy. Meanwhile, domain knowledge is limited and may be inaccurate in many applications, which leads to a worse reconstruction. In order to tackle this, I present two advanced methods: 1) ARES (*Automatic REStoration*) that performs data reconstruction by automatically discovering patterns in the time series using *annihilating filter technique*, 2) TURBOLIFT that aims to improve the quality of any existing disaggregation methods by *refining* the initial reconstruction.

Despite that reconstruction provides an elaborate view of data, its performance may vary depending on the data aggregation level, and it requires extra computational cost. Moreover, in some cases, analyzing coarse data may be sufficient to achieve acceptable accuracy. Therefore, I propose a framework, SMARTPROGNOSIS, to automatically suggest aggregation levels, which maximizes

the performance under specific machine learning models.

It is noteworthy that most aggregation methods face information loss when aggregation levels increase. That results in lossy aggregated data, e.g., with annual counts, it is hard to capture the detailed trade during the year. In order to tackle this drawback, I propose a novel summarization algorithm, I-AGG, to aggregate data by emphasizing the critical information of original data.

Table of Contents

1.0 INTRODUCTION 1

1.1 RESEARCH QUESTIONS 2

1.1.1 Data Disaggregation 3

1.1.2 Data Navigation 5

1.1.3 Data Summarization 5

1.2 THESIS ORGANIZATION 6

2.0 BACKGROUND 7

2.1 DATA DISAGGREGATION 7

2.1.1 Data Disaggregation Problem 7

2.1.2 Related Work 9

2.2 DATA NAVIGATION 11

2.2.1 Data Navigation Problem 11

2.2.2 Related Work 12

2.3 DATA SUMMARIZATION 13

2.3.1 Data Summarization Problem 13

2.3.2 Related Work 14

2.3.2.1 Data Sampling 14

2.3.2.2 Data Aggregate Approximation 15

2.3.2.3 Frequency Domain Transformation Approximation 16

2.3.2.4 Machine Learning based Method 18

2.3.2.5 Methods for Dynamic data 18

2.3.2.6 Representation learning 19

3.0 DATA DISAGGREGATION: RECONSTRUCTING AGGREGATED HISTORICAL DATA 20

3.1 AUTOMATIC RECONSTRUCTION WITH PATTERN DISCOVERY 21

3.1.1 ARES 21

3.1.1.1 Informal Explanation of Annihilating Filter (AF) 21
3.1.1.2 Generating Annihilating Filter 22
3.1.1.3 ARES Model 24
3.1.2 Automatic and Iterative update 25
3.1.2.1 Automatic Selection 25
3.1.2.2 ITERATIVE ARES 26
3.1.3 Complexity of ARES 27
3.2 FAST ACCURACY LIFTING FOR RECONSTRUCTION 28
3.2.1 TURBOLIFT: Iterative Solution 28
3.2.2 TURBOLIFT: Analytical Solution 29
3.2.2.1 Complexity of Analytical TURBOLIFT 32
3.2.3 TURBOLIFT: Impact of Initial Solution 33
3.3 EXPERIMENTAL SETUP 35
3.3.1 Data 35
3.3.2 Aggregation configuration 36
3.3.3 Evaluation metrics and baselines 38
3.4 EXPERIMENTAL RESULT 38
3.4.1 Effectiveness Evaluation 38
3.4.2 Impact of the First Phase Approximation in ARES 45
3.4.3 Practitioner's Guide for Disaggregation methods selection 46
3.4.4 Scalability 47
3.5 Conclusion 47
4.0 DATA NAVIGATION: INTEGRATED DATA WAREHOUSE WITH INTELLIGENT DECISION NAVIGATOR 49
4.1 BACKGROUND: PROGNOSIS AFTER CARDIAC ARREST 50
4.1.1 Quantitative Electroencephalography (qEEG) 51
4.1.2 Major Challenges 51
4.2 BRAINFLUX: INTEGRATED DATA WAREHOUSING INFRASTRUCTURE FOR DYNAMIC HEALTH DATA 52
4.2.1 BrainFlux Architecture 53

4.3 SMARTPROGNOSIS: INTELLIGENT DECISION NAVIGATOR 54
4.3.1 Proposed Framework 56
4.3.1.1 Automated Machine Learning 56
4.3.1.2 Genetic Programming for Pipeline Selection 56
4.3.1.3 Ensemble Learning for Prediction 60
4.4 EXPERIMENTS 61
4.4.1 Experimental Setup 62
4.4.1.1 Dataset 62
4.4.1.2 Training and Test setting 62
4.4.2 Experimental Results 63
4.4.3 Discussion 63
4.4.3.1 Pipeline hyperparameter 64
4.4.3.2 Ensemble Learning 65
4.4.3.3 Misclassification 66
4.4.3.4 Limitations 67
4.4.4 Future works 67
4.5 CONCLUSION 68
5.0 DATA SUMMARIZATION: TARGET-SENSITIVE DATA SUMMARIZATION FOR TIME SERIES DATA 69
5.1 BACKGROUND 70
5.1.1 Generative Adversarial Network (GAN) 70
5.1.2 Variational AutoEncoder (VAE) 70
5.1.3 Hybrid of GANs and VAEs 71
5.2 Time Series Data Summarization Framework 73
5.2.1 Score generation 73
5.2.2 Aggregation generation 76
5.3 Experiments 78
5.3.1 Experimental Setting 78
5.3.2 Experimental result 79
5.3.3 Analysis 81

5.3.4 Practitioner's Guide 83
5.4 Conclusion 84
6.0 CONCLUSION AND FUTURE DIRECTION 88
6.1 Concluding 88
6.2 Future Direction 88
Appendix. DATA DISAGGREGATION ADDITIONAL RESOURCES 90
A.1 Formal Justification of the TURBOLIFT Analytical Solution 90
A.1.1 Preliminaries 90
A.1.2 Proof of Lemma 3.2.1 92
A.2 Additional Experiments of TURBOLIFT 96
A.2.1 Data 96
A.2.2 Spiky historical data 97
A.2.3 Sparse historical data 98
Bibliography 101

List of Tables

Table 1 Notations used in Chapter 3 22
Table 2 Run-time of Fast Randomized SVD on large scale matrices [113] 33
Table 3 The average RMSE of ARES compared to different baseline methods. 42
Table 4 The average RMSE of TURBOLIFT compared to different baseline methods. . . 42
Table 5 Notations used in Chapter 4 50
Table 6 Clinical characteristics of patients 63
Table 7 Parameterization 64
Table 8 Experimental results 65
Table 9 Analysis capability comparison on different datasets (Length and Time spend) . 80
Table 10 Analysis capability comparison on different datasets (Classification accuracy) . 81
Table 11 Summarization quality comparison on different datasets 82
Table 12 Imputation capability on Cricket dataset with different missing ratios. 83

List of Figures

Figure 1 Illustration of research objectives. 3
Figure 2 Examples of aggregated historical reports and corresponding linear system. . . 8
Figure 3 Illustration of aggregation navigator [1]. 12
Figure 4 Examples of PAA and SAX 16
Figure 5 The different between Fourier Transform and Wavelet Transform [10]. 17
Figure 6 Examples for reconstructing weekly counts of people infected measles in NY. . 21
Figure 7 Illustration of ARES AF generation algorithm 23
Figure 8 Illustration of the trend of cost function for New York measles data 27
Figure 9 Illustration of TURBOLIFT iterative steps. 30
Figure 10 TURBOLIFT reconstructed series at multiple iterations. 30
Figure 11 Illustration of TURBOLIFT analytical solution. 32
Figure 12 Convergence. Iterative solution converges to the analytical solution. 32
Figure 13 The impact of the initial solution on TURBOLIFT reconstruction. 34
Figure 14 Time series used in the experiments. 36
Figure 15 Different aggregation configurations. 37
Figure 16 Error change rate of ARES on NY Measles. 39
Figure 17 The error change rate of TURBOLIFT on Tycho NY Measles data. 40
Figure 18 H-FUSE reconstruction results of NY Measles when $RD = 52$. 42
Figure 19 ARES error change rate on different datasets comparing to different baseline. . 43
Figure 20 TURBOLIFT error change rate on different datasets with different initials. . . 44
Figure 21 Performance of ARES and ARES+ is comparable. 46
Figure 22 Execution time. Both methods scale well with a linear relation. 48
Figure 23 General BrainFlux Architecture 53
Figure 24 The framework of SMARTPROGNOSIS. 55
Figure 25 The impact of number of pipelines in ensemble learning. 66
Figure 26 Architecture of GANs [136] 71

Figure 27 Structure of VAE/GAN [82] 72
Figure 28 Architecture of proposed method for data aggregation. 74
Figure 29 Example of the aggregation frequency over different segments. 77
Figure 30 Example of the score output for data with 30% of missing. 85
Figure 31 Illustration of the effectiveness of I-AGG over PAA with three examples. . . . 86
Figure 32 Example of gunpoint dataset. 87
Figure 33 Reconstruction performance under different levels of noise. 87
Figure 34 TURBOLIFT error change rate on spiky NY Measles data. 97
Figure 35 Comparison of the TURBOLIFT and the baseline methods. 98
Figure 36 TURBOLIFT error change rate on sparse NY Measles data. 99
Figure 37 Histogram of number of hits (detected non-zero time-ticks). 100

1.0 INTRODUCTION

There are numerous amounts of historical datasets collected and made available to the public by different groups worldwide, such as the Institute for Quantitative Social Science (IQSS) at Harvard University, Great Britain Historical GIS at the University of Portsmouth, the International Institute of Social History in Amsterdam, and the World-Historical Database at the University of Pittsburgh. Interpreting and mining these historical data involve consolidating and fusing large amounts of data from different sources. In health science, for instance, the Vaccine Modeling Initiative at the University of Pittsburgh aims to gather and analyze information from *thousands* of reports on epidemiological data in the United States, spanning more than 100 years [131, 132].

Historical data commonly include reports on time series that have a low temporal resolution, e.g., the monthly counts of people infected with measles. Such reports are seen as aggregates of multiple data atoms in a higher resolution series. Usually, aggregation can be performed in different ways, such as 1) *temporally*, e.g., the monthly counts of people infected with measles, 2) *spatially*, e.g., the population of New York, and 3) *others*, e.g., counts of students in computer science department (i.e., same affiliation). In this thesis, we mainly focus on the temporal one, which is the most common scenario in practice, especially historical data. Clearly, data aggregation could bring benefits from different aspects, such as mitigating the requirement of storage resources, communication cost [100], and enhancing privacy [124]. Additionally, we also found frequent situations where finer resolution data is not available. For example, the energy aggregation data contains the energy consumption for a whole building where the appliance-specific data is inaccessible [76].

Although data aggregation has many favorable properties and is inevitable in some cases, it comes with a severe drawback – blurring the details of data. As we know, many state-of-the-art machine learning models require a considerable amount of data with detailed information for high accuracy, which makes aggregated data undesirable. For example, detecting unusual traffic using the New York taxi flows prefers hourly log data. The aggregated data, in this case, is unable to reflect the timely information. Moreover, several researchers demonstrate the ineffectiveness of data aggregation in data mining and data analysis. For example, in the economic field, they

present that aggregated data would result in information loss and thus misleads the conclusions of individual economic behavior [29, 47].

However, aggregated data is not unworthy in all situations. It is noteworthy that for some specific problems, elaborate information is not necessary for decision making, making the aggregated data preferable by considering the computational cost. For example, the clinic providers are able to accurately prognosis patients based on five minutes summarization of brain activity data recorded by every 0.2 seconds [95]. Moreover, Kourentzes et al. [78] mentioned that analyzing through multiple aggregated data with different levels (i.e., time interval) could facilitate the performance of data study.

1.1 RESEARCH QUESTIONS

In this thesis, we aim to tackle this trade-off problem – how to efficiently process and utilize the aggregated data while preserving the performance of analysis, by answering the following research questions:

Question 1 [Data disaggregation]: How to *reconstruct* higher-resolution time series from aggregated data with acceptable accuracy, e.g., estimating the number of weekly measles infected people from monthly counts?

Question 2 [Data navigation]: How to *select* appropriate aggregated data for analysis with minimal decrease in performance, e.g., detecting outbreaks of measles using monthly counts have comparable accuracy with daily counts?

Question 3 [Data summarization]: How to *generate* aggregated data utilizing a more intelligent method with less information loss, e.g., storing the weekly count on summer season (i.e., peak time) while monthly for the rest?

Figure 1 visually illustrates how these three questions help with the processing of aggregated data. Arrows in different colors indicate the direction of influence for each research question. Firstly, reconstructing the aggregated data before machine learning analysis would provide an elaborate understanding of the data and thus improve model accuracy. Meanwhile, considering

that data reconstruction requires extra computational resources and, in some cases, aggregated data has the ability to reveal essential findings, carefully navigating to an appropriate aggregation level (i.e., time interval to aggregate) for a machine learning task could achieve comparable accuracy as the original data with a less computational cost. Last but not least, generating aggregation reports in an intelligent way could mitigate information loss and result in a higher performed analysis.

Next, we describe the motivation and proposed algorithms for each research question in detail.

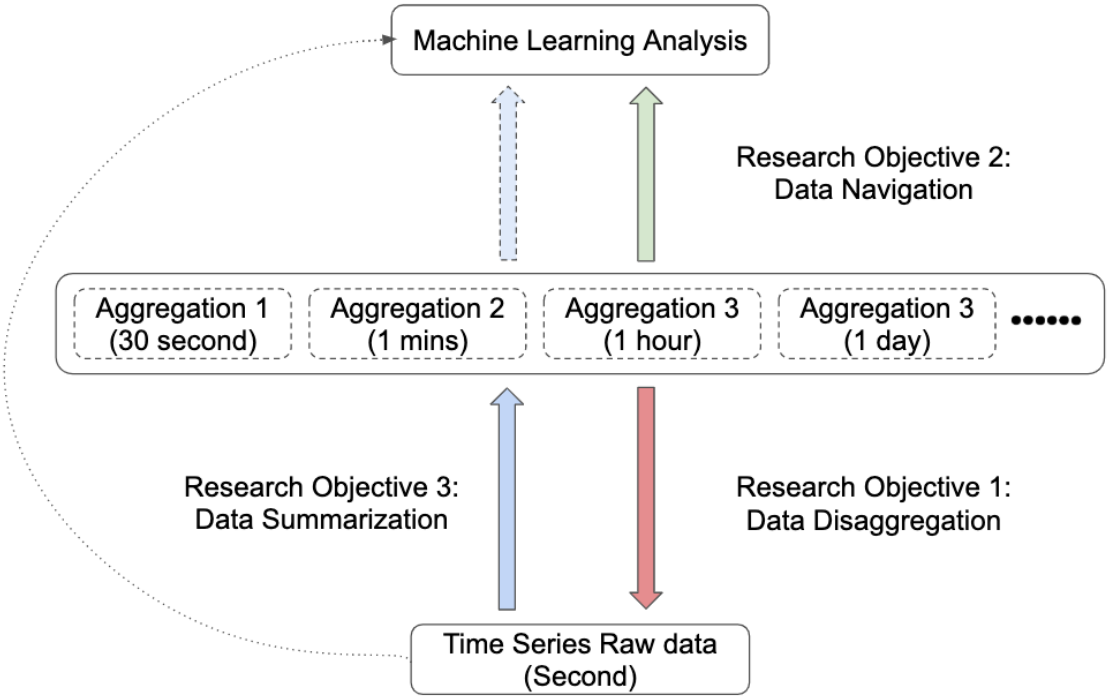


Figure 1: Illustration of research objectives.

1.1.1 Data Disaggregation

The goal of data disaggregation techniques is to reconstruct the desired high-resolution time series from available aggregated reports before analyzing. In this thesis, we use the term aggregate to refer to the sum. Practically, the disaggregation models could face several well-known challenges that emerge in real-life databases, e.g., 1) **overlap**: we may have multiple monthly and one annual reports from different authorities about cases of measles in Los Angeles covering one

year; 2) **conflict**: the sum of the monthly reports for a particular year is not the same as the value of the annual report for that year; 3) **missing**: the reports of patient counts do not cover the period of 1940-1944 because of World War II. Moreover, data disaggregation is an under-determined problem where the number of aggregated reports is far less than the number of target time-ticks. Therefore, classic methods exploit additional information or utilize domain knowledge about the data, e.g., smoothness [87], periodicity [87], or sparsity [5], to improve reconstruction accuracy.

However, we frequently find situations where there is no available domain knowledge or extra information about the data to be exploited. Imposing a random or inaccurate constraint would certainly degrade the quality of reconstructed series resulting in misleading the following analysis. In this thesis, we present a novel and efficient approach, called ARES (*Automatic REStoration*), which reconstructs historical data by automatically discover a dominant pattern of the target series. ARES is composed of two phases: (1) first, it estimates the sequential data utilizing domain knowledge, such as smoothness and periodicity of historical events; (2) then, it uses the estimated sequence to derive notable patterns in the target sequence to refine the reconstructed time series. ARES applies an *annihilating filter technique* that is reminiscent of ideas encountered in spectral estimation and compressed sensing [63, 129]. The idea of annihilating filtering is to learn a linear shift-invariant operator with length L whose response to the desired sequence is (approximately) zero (i.e., any L successive time-ticks multiplied by the operator should sum up to zero). This operator is traded as the dominant pattern shared by any continued L time-ticks. ARES further improves the reconstruction accuracy by applying the annihilating filtering at the second phase iteratively.

Although ARES bypasses the limitation of lacking domain knowledge, the dominant pattern discovered by it is not guaranteed. For example, spiky time-ticks could disturb the pattern recognition, since they are identified as abnormal, which differ significantly from the majority of data. Motivated by this, we present a novel approach, called TURBOLIFT, that refines the solutions provided by existing disaggregation methods and significantly lifts the accuracy of the reconstructed high-resolution time series. Starting from an initial solution produced by a specific disaggregation method, TURBOLIFT *iteratively* finds a new solution that minimizes the disaggregation error *and* is close to the current solution. A notable advantage of TURBOLIFT is that it can reap the benefits inherited from the initial solution and release the initial solution from any inexact constraint im-

posed by the starting method. Moreover, we derive a closed-form solution that enables us to obtain the solution to the formulation of TURBOLIFT *analytically*, without the need to perform resource and time-consuming iterations.

1.1.2 Data Navigation

Despite the above methods providing an elaborate view of data resulting in high-performance analysis, they apparently demand extra computational time and resources. Moreover, the performance of data disaggregation varies depending on the quality of available aggregated reports (see detail in Section 3.4). Therefore, several researchers are attracted to explore the analytical ability of aggregated data [13, 70, 135]. Borgatta and Jackson [15] claim that the aggregated data, in some cases, has the capability to suggest finding at an individual level.

In order to advise (i.e., navigate) the selection of aggregated data as well as the machine learning model, which reveals more insights for target analysis, we propose an automatic machine learning framework. We illustrate one application of analyzing aggregated data in the field of medication – prognosis after cardiac arrest, where we show how our technique can be effectively used to navigate the selection of appropriate aggregation reports with high performance. We name the framework as SMARTPROGNOSIS based on its application. Intuitively, SMARTPROGNOSIS adopts automatic machine learning techniques and ensemble learning to automatically generate and assemble candidate machine learning pipelines (i.e., a sequential combination of data preprocessing and machine learning algorithms.). The candidate pipelines are selected by maximizing the sensitivity of predicting the poor neurological outcome with a fixed, extremely low error rate in misclassifying patients with good outcomes.

1.1.3 Data Summarization

Data summarization is a process of creating a concise yet informative version of data [2]. In general, summarization can be produced by 1) data aggregate approximation (e.g., sum, mean, etc.), which summarizes statistic information of a group data, 2) data sampling, which selects a subset of representative data, or 3) machine learning methods (e.g., clustering), which find the natural or intrinsic of similar data.

A common approach for historical time series data, as we mentioned above, is aggregation. It tends to compute a statistical value over a range of data, where the range is evenly distributed. In this case, the summarized result may eliminate the crucial details and result in information loss. For example, the annual aggregation of infected people counts is unable to capture the seasonal tread of disease. Therefore, we propose a novel data summarization method for time series data by intelligently deciding an appropriate aggregation level for each time interval (i.e., aggregating data with different time intervals on different segments.). The aggregation levels are decided based on the information contained in the segments – the more crucial information, the finer resolution we use. We learn the importance of each data point by a generative adversarial model [92], which identifies the relatively important time ticks that can be used to recover the original data. In detail, we first generate important scores for each time-ticks through a Bi-LSTM layer and then reconstruct the weighted sequence (i.e., weighted by important scores) back to the original sequence and distinguish it from the real data using a generative adversarial model. We maximize the ability of weighted sequence to capture the sequence information.

1.2 THESIS ORGANIZATION

The rest of this thesis is organized as follows. I go through the background information and literature review regarding each research objectives in Chapter 2. In Chapter 3, I explain the detail of a historical data reconstruction algorithm (ARES) and a lifting method to improve the recovered result (TURBOLIFT) further. Chapter 4 presents a new time series data warehouse where multi-resolution data are stored and an intelligent data navigation method for aggregations selection (SMARTPROGNOSIS) built on top of the warehouse. Chapter 5 illustrates an algorithm (I-AGG) to summarize time series data, which aims to preserve the most crucial information of the original data. Chapter 6 concludes the thesis and provides a discussion on future work.

2.0 BACKGROUND

In this chapter, we provide a background and related work review for each research question, respectively.

Notation: bold capital letters (e.g., $\mathbf{X}$) denote matrices; small letters with upper arrow (e.g., $\vec{x}$) denote vectors; $\mathbf{X}^{\dagger}$ is the Moore-Penrose pseudo-inverse of a matrix $\mathbf{X}$; $\mathbf{X}^{T}$ denotes the transpose of $\mathbf{X}$. $\vec{x}_n$ is the n^{th} element in vector $\vec{x}$.

2.1 DATA DISAGGREGATION

2.1.1 Data Disaggregation Problem

The problem of data disaggregation, also known as information disaggregation, considered in this thesis is a special case of information fusion. The concept of information fusion has been used in a wide range of areas, including multi-sensor data fusion [57], human-centered information fusion [24, 56], and information fusion for data integration [14]. Multi-sensor data fusion utilizes multiple sensors to improve the signal-to-noise ratio and the robustness and reliability of the sensor network in the presence of sensor failures. Human-centered information fusion enhances fusion techniques by including human observations as well as web-based information about interactions between humans (e.g., social networks). In this direction, researchers developed automatic information fusion methods that exploit the collective intelligence of mobile robots and human operators/observers, and efficiently crowd-source the victim detection task [147]. The problem of information fusion also appears in image and signal processing applications, such as motion estimation [149], edge detection [120], and super-resolution image reconstruction [98].

The emergence of the Internet and communication networks facilitates access to different data sources with varying reliability. These sources might be mutually inconsistent due to conflicting data [36]. The challenge of resolving data conflicts and data inconsistencies has been addressed in the task of data integration [14, 35, 111, 146].

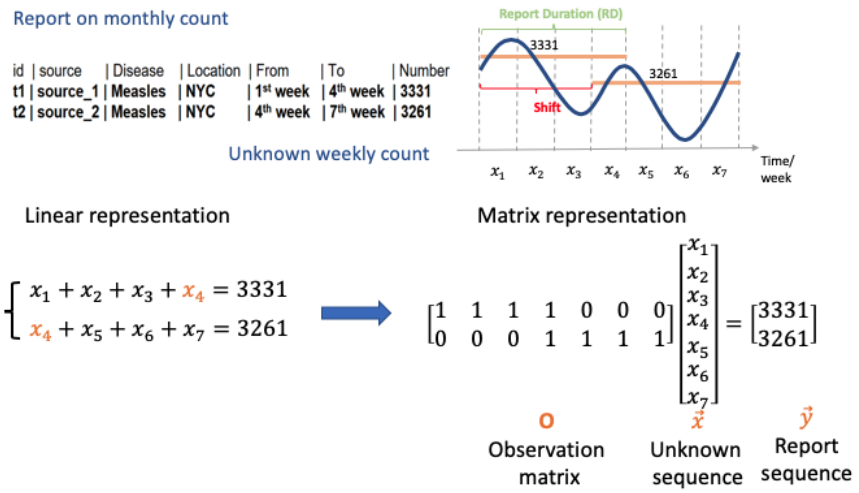


Figure 2: Examples of aggregated historical reports and corresponding linear system.

Data disaggregation is a key component of information fusion and has been studied in various domains, e.g., economics [102], supply chains [69], and epidemiology [87, 5]. The most common type of disaggregation is the temporal disaggregation [121, 44, 101]. Given low-resolution observations about a time series (e.g., a mix of monthly and quarterly sales, or weekly stock market index), the temporal disaggregation models aim to fuse the available observations to reconstruct a high-resolution series (e.g., weekly sales, or daily stock market index) that satisfies the aggregation constraints.

Specifically, in the disaggregation task, the goal is to fuse the given aggregated observations to reconstruct the target high-resolution time series. Figure 2 shows an example of two historical reports on the number of measles infections in NY. Each report covers a time interval of 4 weeks, with one week overlap between them. The number of time-ticks in the interval covered by a given report is referred to as Report Duration (RD). In this particular example, the task of the information disaggregation is to reconstruct the *weekly* counts of measles cases from the *monthly* counts.

The lower part of Figure 2 shows an example of the *linear system* formed from the aggregation constraints using the two reports about measles infections in NY. Each report is represented as a binary row vector in the observation/aggregation matrix $\mathbf{O} \in \mathcal{R}^{M \times N}$, with ones at the indices that correspond to the time-ticks covered by this report. This information disaggregation prob-

lem can be stated as finding the solution to the linear system in Figure 2. Formally, information disaggregation methods aim to minimize the following deviation:

$$\mathcal{F}(\vec{x}) = \sum_{m=1}^{M} (\vec{v}_m - \sum_{n=1}^{N} \mathbf{O}_{m,n} \vec{x}_n)^2 \tag{1}$$

or, in the matrix form:

$$\mathcal{F}(\vec{x}) = ||\vec{v} - \mathbf{O}\vec{x}||_2^2 \tag{2}$$

where N is the total number of time-ticks in $\vec{x}$, M is the total number of reports, $\vec{v}$ is the vector containing the values of aggregated reports, and $\mathbf{O}$ is the observation/aggregation matrix that maps the available aggregated reports with the high-resolution target series.

Ideally, the deviation between $\mathbf{O}\vec{x}$ and the reports $\vec{v}$ should be zero. However, in practice, the above linear system in Figure 2 is super under-determined, as the number of reports is much smaller than the number of time-ticks, i.e., $M \ll N$. As a result, it has an infinite number of solutions if the system is consistent, i.e., the reports are not conflicting. The Least Squares (LSQ) chooses the solution that minimizes $\mathcal{F}(.)$ and has the minimum-norm ($\min\ ||\vec{x}||_2^2$). However, in most real data, the true solution is not necessarily the one that has (or is close to) the minimum norm.

The previous discussion reveals that the time series disaggregation problem is ill-posed in general, making it a challenging task. An ill-posed problem is one that does not meet the three criteria to be well-posed [68]. These criteria are: 1) A solution exists. 2) The solution is unique. 3) The solution's behavior changes continuously with the initial conditions. There are different approaches proposed to handle the time series disaggregation problem in the literature; we summarize them in the next section.

2.1.2 Related Work

Time series disaggregation is a popular problem, especially in the statistical and economics literature [25, 117]. A class of the existing disaggregation approaches leverage extra information and priors in the disaggregation task, such as integrating multiple views of the same data, each is aggregated in a different dimension [4, 25, 33, 101, 102, 117]. For instance, we are interested in estimating the quarterly Gross Regional Product (GRP) values for regions of a country, given: 1) the

annual GRP per region (temporal aggregates), and 2) the quarterly Gross Domestic Product (GDP) national accounts (aggregated over regions). In addition, some algorithms assume a linear regression model between the target high-resolution series and some related series that are available in high-resolution [25, 101, 102]. For instance, the daily stock market of an oil company is estimated as a linear combination of the stock market of other oil and relevant companies. The advantage of exploiting multiple views or assuming a linear regression relation is to reduce the number of variables versus equations; however, the extra information is not available in our context.

In cases where no extra information is available, classic techniques exploit domain knowledge to tackle the time series disaggregation problem [44, 87]. For instance, Liu et al. [87] introduced a method, called H-FUSE, that imposes smoothness and/or periodicity constraints on the solution to the disaggregated series, in their attempt to make the problem over-determined. The smoothness (H-FUSE-S) is imposed on the solution by penalizing the large differences between adjacent time-ticks in $\vec{x}$. With a trivial example of a series of length $N = 4$, H-FUSE-S imposes the following soft constraint to the least-squares criterion in (2).

$$\left\| \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \right\|_2^2 \tag{3}$$

In some applications, periodicity is expected, e.g., yearly periodicity in weather data. H-FUSE also imposes periodicity constraint (called H-FUSE-P). With the same concept, large differences between each time-tick and its equivalents in a predetermined period are penalized [87]. With some time series, constraining the solution with both smoothness *and* periodicity (named H-FUSE) leads to higher accuracy.

Although the domain knowledge assumptions described above are reasonable and effective in many applications, they can be restrictive when the time series does not *exactly* follow the imposed constraints. Recently, the work in [5] proposed to circumvent the under-determinacy of the problem by solving for the disaggregated series in the frequency domain. Specifically, this method searches for the coefficients of the Discrete Cosine Transform (DCT) that represent the time series of interest. A small number of coefficients is sufficient to approximate the series, i.e.,

the series is sparse in the DCT domain. Basis Pursuit (BP) [23], and its variations, are employed to find such representation. BP is an optimization technique that finds a sparse solution to an under-determined system by choosing the l_1 norm of the solution as a criterion. In this context, DCT is used as a sparsifying dictionary. One advantage of the approach in [5] is that it automatically detects the prominent periodicities in the data, as opposed to assuming a *known* periodicity in H-FUSE. However, the estimation accuracy of this method hinges upon the periodicity degree of the time series. If the data is known to have a few dominant periodicities, i.e., quasi-periodic, then this method is expected to achieve a good reconstruction.

2.2 DATA NAVIGATION

2.2.1 Data Navigation Problem

In order to efficiently process and analyze data, storing data in the aggregation format with different aggregation levels is favorable in many domains, such as economics and health care. However, it is undeniable that data in different aggregation levels could reveal or conceal different features of the original data. For example, data aggregation could smooth the high-frequency features and provide a better estimation of long-term dynamics, especially for fast-move data. While, for slow-move data, aggregation may bring the opposite effect by reducing (or even removing) the intermittence of the data [106]. Therefore, it is a challenging task for users to decide the suitable aggregation records for a specific problem when considering the trade-off between performance and efficiency. Data navigation is one of the techniques that assist this process. It takes the users' requests and automatically redirects them to suitable aggregation records. Our informal problem definition is given as follows:

Informal Problem 1 (Navigation)**.**

1. *Given*: 1) the multiple aggregations of dataset D with different aggregation levels ($A = \{A_1, A_2, ..., A_n\}$) and 2) a user query Q.
2. *Redirect*: the query Q to one or multiple appropriate aggregation records that could answer the query efficiently and properly.

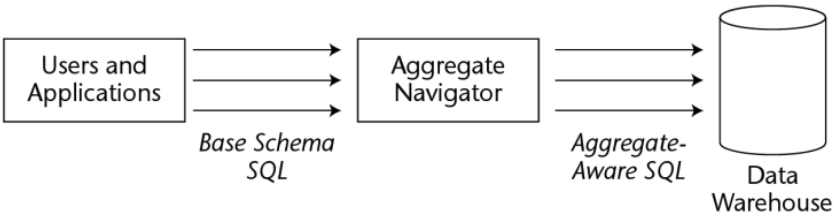


Figure 3: Illustration of aggregation navigator [1].

2.2.2 Related Work

The idea of data navigation was first presented in the data warehouse application known as the aggregation navigator. Data warehouse [65] is a central repository of information, which contains data gathered and integrated from different sources, and it is usually used for reporting and analyzing data. Unlike the operational database, a data warehouse aims to store large quantities of historical data and enable fast, complex queries across all the data. Therefore, one fundamental part of the data warehouse is aggregation, which summarizes multi-dimensional data into a new level of granularity and could provide better comprehensibility in data analysis and data visualization [52]. To avoid low responses due to the system summing data up to higher levels, aggregation in different granularity can be pre-computed and stored in a separate table. However, in this case, users are required to generate queries that consume the appropriate aggregation table by themselves. For example, computing the annual average salary of all employees using the seasonally average salary data is more favorable than the monthly average wage. However, it is undeniable that this will increase the workload for end-users and may cause inaccurate analyzing results, especially when the number of aggregation tables increases or the aggregation tables are not in sync with the base tables. Therefore, an essential component, named aggregation navigator, is designed to guide the utilization of aggregation data. Figure 3 shows a flowchart of the aggregation navigator. Aggregation navigator rewrites users' queries by redirecting them to the most efficient aggregation schema [1, 74]. The principle of aggregation navigator is to avoid the unsynchronized aggregation tables and select an appropriate granularity to answer the query.

In addition to boosting the data query speed, another group of researchers has explored the

capability of aggregated data for analysis. They propose to utilize multiple aggregated data with different resolutions to facilitate the analysis, called multi-resolution learning. The main idea is to integrate results from different models focusing on different resolutions. This method has been successfully applied for different applications, such as forcasting [106], image analysis [105], and knowledge transfer [39]. However, for most of these algorithms, the multi-resolution data is selected based on available or generated by increasing aggregation levels.

2.3 DATA SUMMARIZATION

2.3.1 Data Summarization Problem

In the era of data explosion, a vast number of data has been created daily from different sources, such as Sensor Networks, Cloud Storages, Social Networks, etc. In order to manage and analyze these data, many researchers have investigated the techniques of data summarization. The objective of data summarization is to find a synoptic representation of the original data.

In general, a good summarization should satisfy several properties, including compressed, informative, and be able to infer the approximate over the original data [60]. Therefore, data summarization plays a vital role in big data processing (e.g., storing, retrieving, and computing) by significantly reducing resource consumption. It has been widely used in many domains, such as public health [108], social medial [9], economic [25], etc.. For example, the clinicians could access massive data of a patient collected from different sources, such as patient demographics, medications, lab, and test results, etc.. However, it takes effort to extract useful information from background noise for diagnosis [133]. Moreover, communicating enormous clinical data between hospitals or clinicians faces a risk of delay and inaccurate and thus results in inaccurate diagnosis [80]. Therefore, a clinical summarization, which can describe crucial medical information, is favorable.

2.3.2 Related Work

Since the mass of data comes from different sources, they could be collected in different formats, such as numeric, category, and text. Correspondingly, various types of data summarization techniques have been developed to tackle these different types of data. Technically, these summarization methods could be divided into two categories according to the data types: structured data summarization and unstructured data summarization. Unstructured data refers to the information that either does not have a predefined data model or is not organized in a predefined manner [16]. In most scenarios, unstructured data is text-heavy and thus can be covered by text summarization techniques, such as extractive summarization, topic representation, and knowledge-base summarization [3].

Conversely, structured data represents the data with a fixed format (row and columns) and usually can be stored in a database. The time-series data we focus on in this thesis is an example of structured data as it only contains numerical values at each time-ticks (i.e., each feature is a column, and each time-tick is a row.). Therefore, we focus on the structured data summarization in this thesis. Due to the standard format of structured data, this type of summarization method can be classified into several categories, e.g., data sampling, data aggregate approximation, machine learning, etc.. We describe the details of each category as follows.

2.3.2.1 Data Sampling

Data sampling is a process to systematically select a relatively smaller number of representative data points from the original data. Importantly, any query or analysis applied to the sampled data should be able to generalize to the whole dataset. In practice, there are different types of techniques for generating sample data and usually can be classified into two main categories: probability sampling and non-probability sampling.

Probability sampling is, obviously, any method that the sample data are selected using a method based on the theory of probability. The simplest example is *simple random sampling* [96], which randomly selects data points based on a predefined distribution, e.g., uniform distribution or Gaussian distribution. Considering that the data may contain a bias towards one particular group of data, e.g., given a skew distributed data, data points around the tail are hard to be selected. *Stratified*

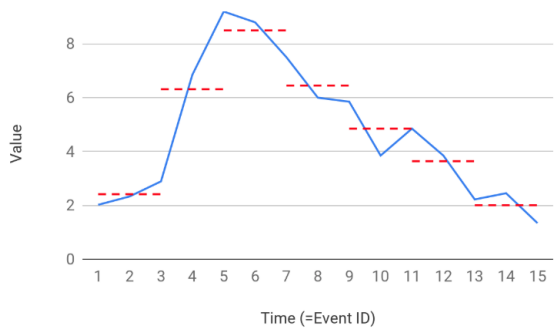
Random Sampling [127] has been proposed to bypass this drawback, which divides the dataset into non-overlapping subsets (i.e., strata.) and applies the simple random sampling on each stratum. Moreover, some methods suggest applying simple random sampling in clustered data to guarantee the represented data is evenly distributed in each group. This method refers to *cluster random sampling* [130]. Besides the random sampling, *systematic sampling* [54] provides another strategy to generate the data subset, which selects the data points from a specified starting point to the end with equal intervals. In particular, given the position of the start point (2^{nd}) and the interval 3, the sampled data is $2^{nd}, 5^{nd}, 8^{nd}$, etc..

Contrariwise, non-probability sampling techniques do not follow any methodological decision based on probability, and thus the sampled data is hard to be used to infer the general population in statistical terms. For example, the sample data could be extracted by predefined criteria. Any data that satisfies the criteria would be included until the sample size is achieved. Therefore, this type of sampling technique is not appropriate for data summarization in our cases as it could not provide sufficient information.

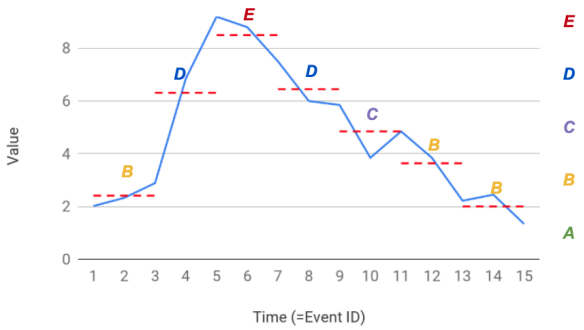
2.3.2.2 Data Aggregate Approximation

Data aggregation approximation is a reverse process of the above data disaggregation, which gathers statistical measurements of data points located in the same segment, such as mean, sum, variance, etc. These segments are usually divided depending on the value of attributes with equal probability. For example, the weekly counts of measles infection could be clustered by different months and then aggregated to monthly counts data. In general, this technique refers to Piecewise Aggregate Approximation (PAA) when it applies to time series data, which we are interested most in this thesis. Figure 4 (a) shows an example of the PAA for a time series data with half-length reduced. In each segment, the average values (red dash line) of every three consecutive time-ticks are recorded to represent the entire data (blue line).

Considering that a simple statistic value is not sufficient to represent the data in each segment, some researchers proposed to include multiple measurements from different aspects to provide a comprehensive understanding. For example, Ren et al. [112] suggest dividing each segment vertically into several regions further based on the value of data in that segment to capture the structure



(a) Piecewise Aggregate Approximation with average measurement for each segment.



(b) Symbolic Aggregate approXimation with letter representation.

Figure 4: Examples of PAA and SAX

in the space of amplitudes. Another work in [148] propose to divide each segment into two part and use the numerical average, respectively, to represent the trend of data. However, it is noteworthy that a trade-off exists between the informativeness and the compactness of the aggregation. In other words, with more types of information recorded in segments, the benefits of aggregation, such as reducing the computational resources, are decreasing. In addition to representing the segments by numerical data, another community proposes to convert the statistic value into a string (e.g., letters) [86, 71], known as Symbolic Aggregate approXimation (SAX). We show an example of SAX using the same time series in Figure 4 (b). The time series is converted to a string vector $[B, D, E, D, C, B, B]$ by representing each average value with a letter. The distribution of these letters is decided by the range of the time series data. In practice, this symbolic representation allows researchers to understand the data better and brings benefits for future analysis. For instance, researchers are able to apply text mining techniques to numerical data.

2.3.2.3 Frequency Domain Transformation Approximation

Frequency domain transformation approximation is a kind of summarization method that works on the frequency domain. The benefit of transforming to the frequency domain is discovering the information that cannot be readily seen in the time domain. This kind of method aims to ana-

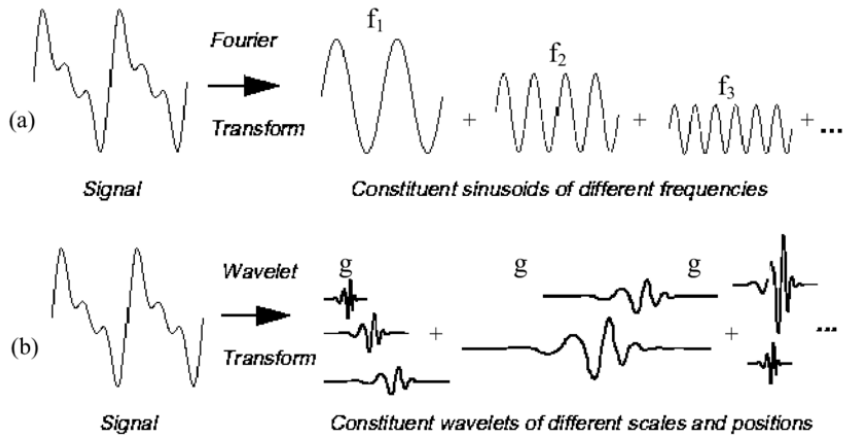


Figure 5: The different between Fourier Transform and Wavelet Transform [10].

lyze the frequency components of the signal where the low frequency represents low variant parts, while the high frequency shows more fine details with a high variant. One basic method in this category is the Fourier Transform, representing the time series data with the frequency spectrum. However, the spectrum contains no time information that cannot tell what instant a particular frequency rises [27]. One solution to tackle this drawback is to analyze the spectrum within every short time window, called Short-time Fourier transform (STFT). But it raises another problem – the selection of window size. A small window size will yield a poor frequency resolution, while a wide window cannot provide useful localization. Therefore, a further solution, Wavelet Transforms, is proposed to deal with this problem. Wavelet Transforms is a time-frequency representation that can provide the time intervals information where particular spectral components occur. Different than the sine-wave in Fourier transforms, Wavelet transforms learn the representation with a series of functions called wavelets, which can be used to divide a given signal into different scales components. Figure 5 shows an example of the Fourier transform, and Wavelet transform on the same time-series signal. The left part is the time series in the time domain, and the right part shows transform representations. Clearly, the Fourier transform breaks the signal into several sine waves with various frequencies, while the Wavelet transform consists of shifted and scaled versions of the mother wavelet.

Generally, the frequency representations mentioned above have the same length as the original

signal. To approximate it with a shorter length (i.e., summarization), a predefined hyper-parameter, amplitude threshold, frequency band, or level of decomposition (in wavelet transform) is needed. For example, we represent a signal with frequencies lower than a cut-off frequency (i.e., low pass filter) and neglect the other frequencies.

2.3.2.4 Machine Learning based Method

Machine learning-based summarization is a kind of technique that summarizes data by mining the natural or intrinsic information of the data. In general, this type of method provides more flexibility in controlling the quality of the summarization. Chandola et al. [21], for example, use the evaluation criteria, such as conciseness and information loss, as the objective function to train the frequent itemsets selection threshold.

Clustering is another commonly used machine learning method for data summarization, whose aim is to find groups of data points that the points within one group are highly similar while across groups are dissimilar. With the group information, two types of summarization methods have been proposed: 1) *centroid based summarization* and 2) *feature-wise intersection based summarization*. In particular, centroid-based summarization represents the groups by the centroids, which is usually the arithmetic mean of all the data points that belong to that group. The method proposed by Ha-Thuc et al [55] is an example of centroid-based summarization. They propose a quality-threshold data summarization based on the K-Means clustering method that splits the clusters until the distortion hits a predefined threshold and sets the cluster centroid as the summarized data. On the other hand, the feature-wise intersection-based summarization [21] suggests representing a single group by intersecting the attributes of all the data points inside the groups. This type of approach would benefit the situation when the whole group shares one identical attribute.

2.3.2.5 Methods for Dynamic data

Apparently, the above types of methods are only applicable for finite (i.e., static) structure data. However, many data are generated rapidly (i.e., dynamic data), such as computer network traffic, web searches, sensor data, etc. The summarization of these data requires special care in terms of memory. Specifically, the memory requirement should be independent of the size of the

data. The common method to summarize dynamic data is data sketch, which represents the data characteristics by a small data structure. For example, count-min sketch[31] counts the frequency and Flajolet–Martin Sketch[46] tries to approximate the number of distinct elements. Considering the slow generation speed of historical data, in this thesis, we only discuss data summarization for static time series data.

2.3.2.6 Representation learning

Representation learning is a different kind of method that learns a concise and informative representation of big data.

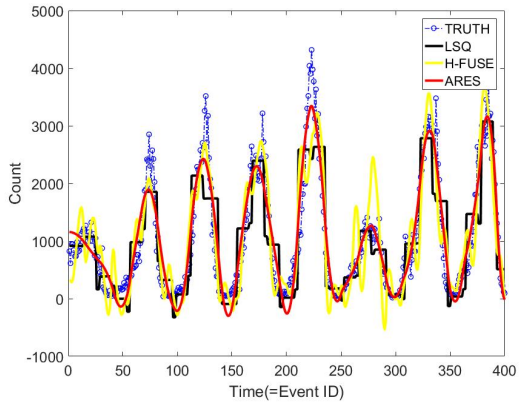
3.0 DATA DISAGGREGATION: RECONSTRUCTING AGGREGATED HISTORICAL DATA

As mentioned in Section 2.1.1, the data disaggregation problem can be nicely formatted into linear equations (Eq.(1)) and then solved by imposing different domain knowledge, e.g., smoothness, periodicity, and sparsity, etc. However, we frequently find situations where there is no available domain knowledge about the data to exploit. Thus, automatically recognizing and utilizing sequential patterns can benefit reconstruction methods, especially when domain knowledge is not available or trustworthy.

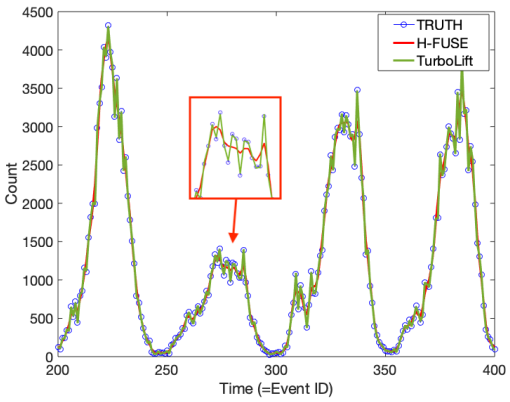
In this chapter, we present two disaggregation methods – ARES and TURBOLIFT, which aim to recover the time series data by automatically discovering a pattern for reconstruction and refine the existing reconstruction result by correcting the imposed patterns, respectively. ARES utilizes Annihilating Filters (AF) [63, 129], to automatically discover patterns in the neighboring time-ticks. Precisely, ARES constraints each L successive time-ticks to follow the pattern learned by the AF (i.e., each L successive time-ticks multiplied by the AF coefficients should sum up to zero). TURBOLIFT refines the existing reconstruction result by iteratively finding a new solution that reduces the disaggregation error and is close to the initial one.

We evaluate ARES and TURBOLIFT on real datasets from different domains, including epidemiological data, retail sales data, and criminological data. Figure 6 shows a simple example of reconstructing the weekly counts of measles infection using both methods and their competitors. We observe that both proposed reconstructions are visibly better than the baseline methods. In Section 3.4, we provide comprehensive experiments that showcase the effectiveness of ARES and TURBOLIFT in improving the performance of different reconstruction methods, including LSQ and the more advanced H-FUSE method. We also provide an elaborate analysis regarding the complexity and demonstrate that both methods scale well.

In Table 1, we summarize the symbols used frequently throughout this chapter.



(a) Real weekly counts data (blue), LSQ (black), H-FUSE (yellow), and ARES (red) reconstructions.



(b) Real weekly counts data (blue), H-FUSE (red), refined H-FUSE by TURBOLIFT (green) reconstructions.

Figure 6: Examples for reconstructing weekly counts of people infected measles in NY.

3.1 AUTOMATIC RECONSTRUCTION WITH PATTERN DISCOVERY

3.1.1 ARES

In this section, we explain our ARES method in detail. As mentioned before, smoothness and periodicity constraints are examples of domain knowledge about simple sequential patterns in a time series. Whereas, in most cases, these constraints are not available. Therefore, ARES applies *Annihilating Filter (AF)* to derive more dominant sequential patterns in a historical time series to reconstruct/disaggregate it.

Next, we introduce the calculation of AF and the disaggregation problem with AF constraints imposed.

3.1.1.1 Informal Explanation of Annihilating Filter (AF)

An *AF* is a linear shift-invariant operator of finite support, which completely suppresses a certain signal. That is, a filter with impulse response $\vec{h} \in \mathbb{R}^{L\times 1}$, such that $\vec{x}\vec{h} := \sum_{i=0}^{L} x_{t+i}h_i = 0, \forall t$. This concept has been introduced in spectral analysis, particularly line spectral estima-

Table 1: Notations used in Chapter 3

Notation	Description
$\vec{x} \in \mathbb{R}^N$	The historical time series in high-resolution.
N	The total number of time-ticks.
$\vec{v} \in \mathbb{R}^M$	Vector containing the aggregated reports.
M	The total number of reports.
$\mathbf{O} \in \mathbb{R}^{M \times N}$	Observation/aggregation matrix.
$\vec{z}_0$	The initial solution provided by a baseline.
RD	Report duration: #time-ticks covered by a report.
RD_{max}	The maximum #time-ticks in one report.
Shift	The difference between the starting time-ticks of two successive reports

tion [63, 129], and has been subsequently used in other applications, notably in compressed sensing by Vetterli *et al.* [134]. The basic idea is as follows. To recover the fine-scale data, we need additional information – ideally, more measurements. Instead, if we know a linear shift-invariant operator that annihilates the signal of interest, then we can use it to build additional equations that the signal satisfies, without the need for additional data – simply because that data would be zero anyway. In practice, the filter will only approximately annihilate the signal of interest when the latter is not a sum of a few sinusoids, so these virtual equations are only approximate. However, we can use a least-squares approach to account for errors in both actual and 'virtual' measurements. This is the starting point of our approach.

3.1.1.2 Generating Annihilating Filter

Next, we describe the steps of generating *AF* for a time series data in detail.

Given a predefined AF length L, the AF tends to find a stable pattern within any L contiguous

time-ticks. In other words, *AF* can be written as a column vector ($\overrightarrow{h}$) such that $\overrightarrow{y} = \mathbf{X}_{mat}\overrightarrow{h}$, where $\overrightarrow{h} \in \mathbb{R}^{L\times 1}$ holds the impulse response of the filter, and $\mathbf{X}_{mat} \in \mathbb{R}^{(N-L+1)\times L}$ is constructed by all L contiguous time-ticks from the input sequence ($\vec{x} \in \mathbb{R}^{N\times 1}$, see details in Figure 7). Hence, $\vec{h}$ can be approximated by minimize the equation $||\mathbf{X}_{mat}\overrightarrow{h}||^2$ ($= \overrightarrow{h}^T\mathbf{X}_{mat}^T\mathbf{X}_{mat}\overrightarrow{h}$), which is equivalent to find the eigenvector corresponding to the smallest eigenvalue of $\mathbf{X}_{mat}^T\mathbf{X}_{mat}$. The reason is explained as follows, any $\overrightarrow{h}$ can be written as $\mathbf{U}\overrightarrow{\lambda}$, where $\overrightarrow{\lambda}$ is a vector of coefficients, whose norm must be 1, and $\mathbf{U}$ contains the eigenvectors of $\mathbf{X}_{mat}^T\mathbf{X}_{mat}$, which are orthogonal [129]. We summarize the steps of generating AF in Algorithm 1. With determined $\vec{h}$, AF can be rewritten as $\overrightarrow{y} = \mathbf{H}_a\overrightarrow{x}$, where $\mathbf{H}_a$ is a Toeplitz matrix [1] constructed from $\overrightarrow{h}$. Thus $\mathbf{H}_a$ is an *annihilating matrix* that we use to define the annihilating constraint below.

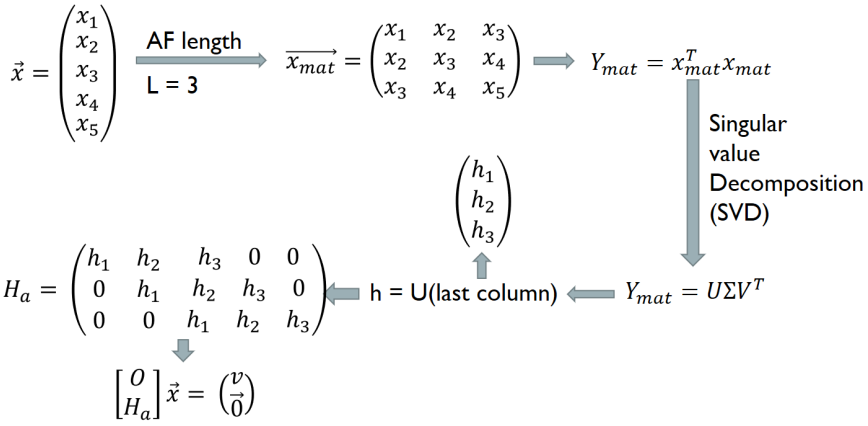


Figure 7: Illustration of ARES AF generation algorithm

Figure 7 shows a general flow of the process starting from an input sequence $\vec{x}$ to the constructed annihilating matrix $\mathbf{H}_a$ with $L = 3$. The $\mathbf{H}_a$ matrix is then used as an additional constraint to facilitate the reconstruction (bottom part of the figure).

[1] The Toeplitz matrix is a matrix in which each descending diagonal from left to right is constant [59].

Algorithm 1 Generate Annihilating filter

Input: $\vec{x}$

Output: $\mathbf{H}_a$ (Annihilating filter constraint)

1: **function** AFILTER($\vec{x}$)
2: Select an appropriate AF length. L
3: Build matrix $\mathbf{X}_{mat}$ by any successive L time-ticks of input sequence $\vec{x}$. $\mathbf{X}_{mat} \in \mathbb{R}^{(N-L+1)\times L}$
4: Calculate $\mathbf{Y}_{mat}$ $Y_{mat} = \mathbf{X}_{mat}^T \mathbf{X}_{mat}$
5: Perform Singular Value Decomposition and extract the last column. $\mathbf{U} \leftarrow SVD(\mathbf{Y}_{mat}), \vec{h} \leftarrow \mathbf{U}$
6: Construct matrix $\mathbf{H}_a$. $\mathbf{H}_a \in \mathbf{R}^{(N-L+1)\times L}$

3.1.1.3 ARES Model

It is noteworthy that, in the above example, AF coefficients are generated using the real sequence to capture the dominant patterns with more accuracy. However, the target sequence is not available in the data disaggregation problem, which makes it hard to discover AFs patterns accurately. In order to obtain some knowledge of the original sequence, we propose to apply a two-phase reconstruction strategy in ARES. The intuitive idea is to reconstruct an approximated sequence with a state-of-the-art method in the first phase and then build the AF based on it (i.e., second phase). Here we apply H-FUSE [87] with smoothness and periodicity constraints in the first phase since it has been approved to be more efficient than imposing only one or other constraints. The expectation of this two-phase approach is that the first phase reconstruction sequence can refine the information about the target series to the extent which makes it reusable to learn *AFs* representing reasonably accurate sequential patterns for the target series. The *AFs* built on refined series should have the ability to improve reconstruction accuracy.

The detail steps are list as below. In the first phase of ARES, we apply the combined smoothness and periodicity constraint:

$$\mathcal{C}_{sp}(\vec{x}) = \frac{1}{2}\mathcal{C}_s(\vec{x}) + \frac{1}{2}\mathcal{C}_p(\vec{x}),$$

where $\mathcal{C}_s(\vec{x})$ is the smoothness constraint to penalize big jumps between successive time-ticks (see details in Eq.(3)), $\mathcal{C}_p(\vec{x})$ is the periodicity constraint which enforces the event at time-tick t to be close to the one at $t + P$ (P is a predetermined period). More formally, ARES solves the following

optimization problem at first phase:

$$\min_{\vec{x}} \mathcal{L}(\vec{x}) = \min_{\vec{x}} \ (\mathcal{F}(\vec{x}) + \mathcal{C}_{sp}(\vec{x})), \tag{4}$$

where $\mathcal{F}(\vec{x})$ is defined as in Eq.(2).

We then extract the *AF* based on the reconstructed result ($\vec{x}_{sp}$) from the first phase and applies the new pattern constraint to the characteristic linear system (Eq.(2)) to improve the reconstruction accuracy. Mathematically, in the second phase, ARES solves the following optimization problem:

$$\min_{\vec{x}} \mathcal{L}(\vec{x}) = \min_{\vec{x}} \ (\mathcal{F}(\vec{x}) + \mathcal{C}_{a}(\vec{x})), \tag{5}$$

where $\mathcal{C}_a(\vec{x})$ corresponds to the annihilating constraint that we define next:

- Annihilating constraint $\mathcal{C}_a$: Based on the definition of *Annihilating Filter* that any successive L time-ticks should follow the pattern discovered by the AF (i.e., consecutive L time-ticks multiplied by corresponding AF coefficients should sum up to zero). Thus, the Annihilating constraint $\mathcal{C}_a$ can be represented with formulation:

$$\mathcal{C}_a(\vec{x}) = \sum_{t=1}^{N-L+1} (\sum_{l=1}^{L} (h_l x_{l+t}))^2 = ||\mathbf{H}_a \vec{x}||_2^2 \tag{6}$$

  where $\mathbf{H}_a \in \mathbb{R}^{(N-L+1)\times N}$ is an *annihilating matrix*, whose t^{th} row contains $\vec{h}$ starting from t^{th} column.

To summarize, ARES first applies reconstruction method with smoothness and periodicity constraints to obtain an approximate sequence $\vec{x}_{sp}$, then builds a corresponding annihilating constraint $\mathcal{C}_a(\vec{x})$ based on the information provided by $\vec{x}_{sp}$. We claim that ARES refines the reconstructed sequence $\vec{x}_{sp}$ to a finer sequence $\vec{x}$ using Eq.(5).

3.1.2 Automatic and Iterative update

3.1.2.1 Automatic Selection

As observed in experiments, the ability of AFs to reveal the insight of the time series data (i.e., reconstruction accuracy) is associated with the predefined length L. For example, a pattern with a shorter length is universal among most time series and hard to exhibit the essential informa-

tion. While the longer pattern faces the challenge of finding an agreement within all L successive sub-sequences. In addition, L has various impacts on different report configurations (i.e., report durations and overlaps). In order to select the best AF length L for each data as well as configuration, we propose an automatic selection criteria inspired by the *Minimum Description Length (MDL)* principle [114]. The philosophy of MDL is to pick the most parsimonious model that describes the data well enough. Hence, L could be determined by the following Selection Criteria (SC):

$$Selection\ criteria: SC(L) = \|\vec{v} - \mathbf{O}\vec{x}\|_2^2 + \|\mathbf{H}_a\vec{x}\|_2^2 + L, \tag{7}$$

where the first two items are the same as the optimization function (Eq.(5)), and the third term penalizes model complexity (i.e., the length of *AF*). In other words, we penalize big L for avoiding over-fitting because with longer L, we may zero out any signal, and this does not yield useful information.

3.1.2.2 ITERATIVE ARES

At last, we propose ITERATIVE ARES to improve the reconstruction accuracy further. The intuitive idea is to refine the reconstructed sequence by repeating the second-phase (AF-based reconstruction) to optimize the AF and reconstructed sequence alternatively. This process follows the idea of Alternating Least Squares Optimization (ALS), which updates two variables, reconstructed series ($\vec{x}$) and AF ($\vec{h}$), alternatively by minimizing the cost function. Specifically, we revise $\vec{h}$ using the reconstructed result $\vec{x}$ from Eq.(5) and refine $\vec{x}$ by imposing the updated $\vec{h}$.

In order to decide the number of iterations, we use Eq.(5) as the cost function $\mathcal{P}(\vec{x})$ of ITERATIVE ARES, whose value decreases at each iteration until convergence.

$$\mathcal{P}(\vec{x}) = \|\mathcal{F}(\vec{x})\|^2 + \|\mathcal{C}_a(\vec{x})\|^2 \tag{8}$$

We show the cost function $\mathcal{P}(\vec{x})$ of a problem – reconstructing the weekly counts of Measles infected patients in New York, at each iteration in Figure 8. The cost function decreases negligibly after the 3^{rd} iteration. Therefore, we set the iteration time as three empirically.

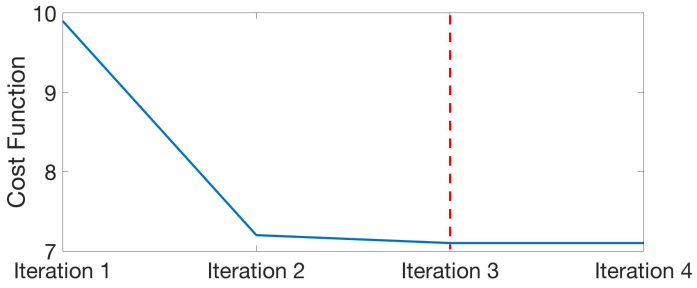


Figure 8: Illustration of the trend of cost function for New York measles data

3.1.3 Complexity of ARES

In this section, we analyze the complexity of ARES. Although ARES contains two-phase steps, both phases aim to solve a similar objective function – a sparse linear system with an additional constraint (Eq.(4) and Eq.(5)). Thus, we discuss them following a same lemma (Lemma 3.1.1).

Lemma 3.1.1. *For any report setting, let RD_{max} be the maximum number of time-ticks covered by one aggregated report in $\vec{v}$, $\mathbf{H}$ be the constraint matrix (e.g., $\mathbf{H}_s$ and $\mathbf{H}_a$) with bandwidth b and N be the total number of events to recover. Then the total computational time for solving the linear system with an additional constraint*

$$\min_x \|\vec{v} - \mathbf{O}\vec{x}\|_2^2 + \|\mathbf{H}\vec{x}\|_2^2$$

is

$$O(Nlog(N) + 4 \times max(b, RD_{max})^2)$$

Proof. The most time-consuming part is matrix inversion. Since $\mathbf{O}$ is a banded Toeplitz matrix with bandwidth RD_{max}, $\mathbf{O}^T\mathbf{O}$ is also a banded Toeplitz matrix of bandwidth $2RD_{max} - 1$; and the same holds for $\mathbf{H}$ and $\mathbf{H}^T\mathbf{H}$ Then, the result follows from [67]. □

By summing up the computational time of two phases, the time complexity of ARES is

$$O(2Nlog(N) + 4max(b_{sp}, b_a, RD_{max})^2),$$

where b_{sp} and b_a are the bandwidth of $\mathbf{H}_{sp}$ and $\mathbf{H}_a$, respectively. As the fact that b_{sp} and b_a are

$\ll RD_{max}$ in practice, the computational time can be approximated to $O(Nlog(N))$. Thus, ARES scales well. We also show the complexity analysis on real data in Section 3.4.4.

3.2 FAST ACCURACY LIFTING FOR RECONSTRUCTION

The above section has provided an effective data disaggregation solution, especially when limited domain knowledge is accessible. However, the reader may doubt the accuracy of *AF* pattern as well as the domain knowledge provided by experts. As we noticed in the experiments, an unfaithful pattern constraint may mislead the reconstruction method to a worse series resulting in an inaccurate analysis. Therefore, in this section, we propose a novel approach, called TURBOLIFT, to refine and improve upon the quality of the solution provided by existing time series disaggregation methods. TURBOLIFT aims to inherit the accurate constraints and filter out the others. Remarkably, TURBOLIFT follows the "first, do no harm" principle [126] that it either improves the disaggregation accuracy of the initial solution provided by the baselines or preserves its accuracy in the worst case.

Intuitively, starting from an initial estimate of the target disaggregated series, at every iteration $k \in \mathbb{N}$, TURBOLIFT *iteratively* finds a solution $\vec{x}$ that: (i) minimizes the deviation from the aggregation constraints ($\mathbf{O}\vec{x} = \vec{v}$, see detail in Eq.(2)), and (ii) is close to the current solutions. This leads to the following problem formulation.

$$\min_{\vec{x},\vec{z}} \quad \|\mathbf{O}\vec{x} - \vec{v}\|_2^2 + \|\vec{x} - \vec{z}\|_2^2 \tag{9}$$

where $\vec{z}$ is initialized with the solution provided by the baseline disaggregation method we wish to improve.

3.2.1 TURBOLIFT: Iterative Solution

Evidently, the optimization problem in Eq.(9) is bilinear. As such, we utilize the *Alternating Least Squares* (ALS) algorithm to iteratively update the two variables $\vec{x}$ and $\vec{z}$ in a cyclic fashion. Starting from an initial solution $\vec{z}_0$ produced by a particular baseline method we seek to improve,

at each iteration $k \in \mathbb{N}$, we fix one variable and update the other.

First, the problem with respect to $\vec{x}$ can be equivalently expressed as

$$\underset{\vec{x}}{\text{argmin}} \quad \| \underbrace{\begin{bmatrix} \mathbf{O} \\ \mathbf{I} \end{bmatrix}}_{\mathbf{A}} \vec{x} - \underbrace{\begin{bmatrix} \vec{v} \\ \vec{z} \end{bmatrix}}_{\vec{y}} \|_2^2 \tag{10}$$

where $\mathbf{I}$ is an $N \times N$ identity matrix. Hence, $\vec{x}$ admits the following closed-form solution:

$$\vec{x} = \mathbf{O}^\dagger \vec{y} = (\mathbf{O}^T\mathbf{O} + \mathbf{I})^{-1} \underbrace{(\mathbf{O}^T\vec{v} + \vec{z})}_{\vec{r}} \tag{11}$$

where $\mathbf{O}^\dagger = (\mathbf{O}^T\mathbf{O})^{-1}\mathbf{O}^T$ is the Moore-Penrose pseudo-inverse of the matrix $\mathbf{O}$. The update of $\vec{z}$ is straightforward by setting it to the current value of $\vec{x}$ at every iteration.

The iterative procedure described above exhibits a monotonically non-increasing loss value; therefore, we perform the updates until convergence. A flow diagram is shown in Figure 9 to illustrate the iterative steps to solve Eq.(9). In Figure 10, we show an example of the iterative solution provided by TURBOLIFT at iteration 1 and 10 using real data of the weekly counts of measles infections in New York. We can see that the solution gets closer to the ground-truth series with iterations.

3.2.2 TURBOLIFT: Analytical Solution

Regarding the complexity of the TURBOLIFT iterative solution, it may appear that updating $\vec{x}$ using Eq.(11) can be computationally expensive owing to the fact that it requires computing the inverse of a matrix, which can cost $\mathcal{O}(N^3)$. However, by means of recognizing and exploiting the structure of the matrices, we provide a closed-form analytical solution to TURBOLIFT that enables us to bypass performing the inverse process and (hundreds of) iterations. We define the analytical expression of the TURBOLIFT method through the *limit* of the iterative solution in the following Lemma (the preliminaries and detailed proof are provided in Appendix A.1.2).

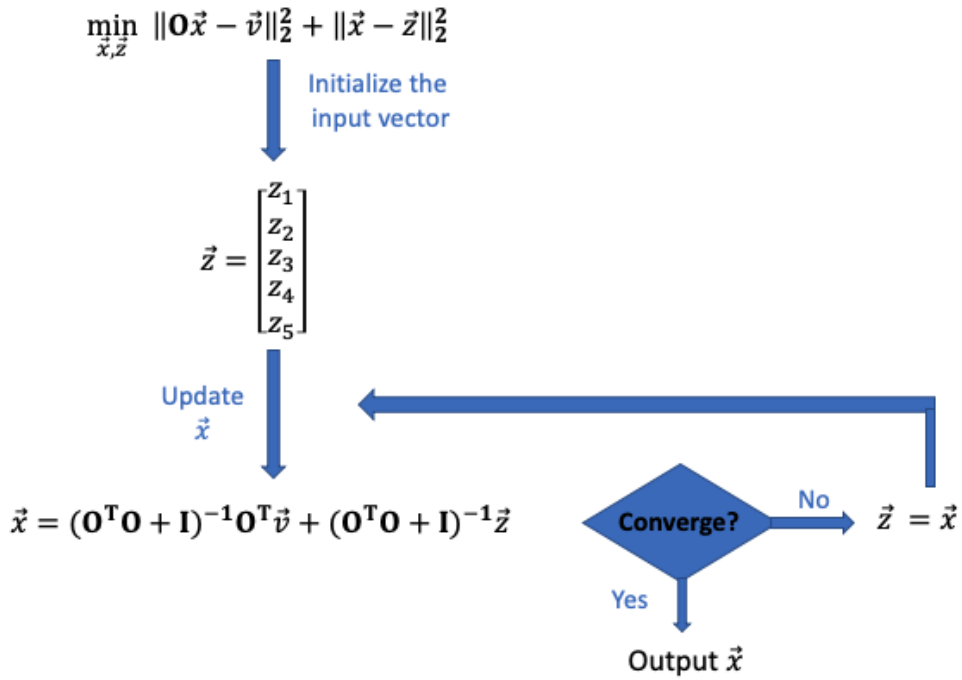


Figure 9: Illustration of TURBOLIFT iterative steps.

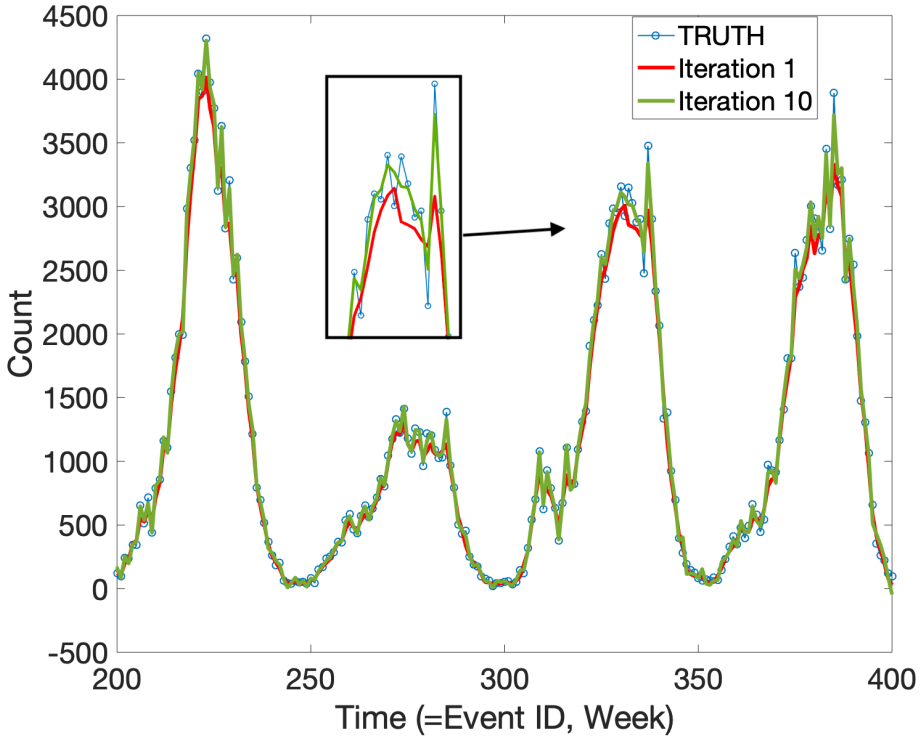


Figure 10: TURBOLIFT reconstructed series at multiple iterations.

Lemma 3.2.1. *Consider the optimization function*

$$\min_{\vec{x},\vec{z}} \quad \|\mathbf{O}\vec{x} - \vec{v}\|_2^2 + \|\vec{x} - \vec{z}\|_2^2$$

where $\mathbf{O} \in \mathbb{R}^{M\times N}$ *is a Toeplitz matrix,* $\vec{x} \in \mathbb{R}^{N\times 1}$*,* $\vec{z} \in \mathbb{R}^N$ *and* $\vec{v} \in \mathbb{R}^M$*. The optimization function can be solved by the Alternating Optimization with the following iterative updates.*

$$\vec{x}_{k+1} = (\mathbf{O}^T\mathbf{O} + \mathbf{I})^{-1}\mathbf{O}^T\vec{v} + (\mathbf{O}^T\mathbf{O} + \mathbf{I})^{-1}\vec{z}_k$$
$$\vec{z}_{k+1} = \vec{x}_{k+1}$$

Furthermore, with $k \to \infty$ *and the initial value* $\vec{z}_0$*, the solution converges to the following stationary point.*

$$\vec{x}_\infty = \mathbf{U_1}{\mathbf{\Sigma_1}}^{-1}{\mathbf{U_1}}^T\mathbf{O}^T\vec{v} + \mathbf{U_2}{\mathbf{U_2}}^T\vec{z}_0 \tag{12}$$

where $\begin{bmatrix}\mathbf{U_1} & \mathbf{U_2}\end{bmatrix} = \mathbf{U} \in \mathbb{R}^{N\times N}$ *is the set of singular vectors of* $\mathbf{O}^T\mathbf{O}$*,* $\mathbf{U_1}$ *is corresponding to the non-zero singular values, while* $\mathbf{U_2}$ *is the remaining part.* $\mathbf{\Sigma}_1$ *contains all non-zero singular values of* $\mathbf{O}$*.*

Figure 11 shows a simple example to illustrate the components of the analytical solution to TURBOLIFT. Based on the above derivations, we implement the analytical solution following the steps listed in Algorithm 2. A clear advantage of the analytical solution is that we get the target reconstructed sequence $\vec{x}$ through a closed-form expression (Eq.(12)), avoiding the time and resource-consuming iterations.

Algorithm 2 Analytical TURBOLIFT

Input: $\mathbf{O}, \vec{v}, \vec{z}_0$
Output: $\vec{x}$ (reconstructed result)

1: **function** TURBOLIFT($\mathbf{O}, \vec{v}, \vec{z}_0$)
2: Get the size of observation matrix. $M, N \leftarrow Size(\mathbf{O})$
3: Singular Value Decomposition. $\mathbf{U}, \mathbf{\Sigma} \leftarrow SVD(\mathbf{O}^T\mathbf{O})$
4: Extract the first m columns of $\mathbf{U}$ as $\mathbf{U}_1$, last $(N - M)$ columns as $\mathbf{U}_2$.
5: $\mathbf{U}_1 \leftarrow \mathbf{U}[:, 1:M], \mathbf{U}_2 \leftarrow \mathbf{U}[:, M+1:N]$
6: Extract the non-zero singular values. $\mathbf{\Sigma}_1 \leftarrow \mathbf{\Sigma}[1:M, 1:M]$
7: Compute the reconstructed sequence. $\vec{x}_\infty \leftarrow \mathbf{U}_1\mathbf{\Sigma}_1^{-1}\mathbf{U}_1^T\mathbf{O}^T\vec{v} + \mathbf{U}_2\mathbf{U}_2^T\vec{z}_0$

Figure 12 illustrates the relationship between the iterative and the analytical solutions to TUR-

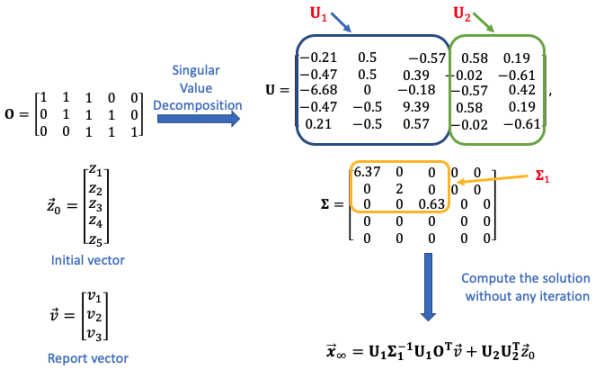


Figure 11: Illustration of TURBOLIFT analytical solution.

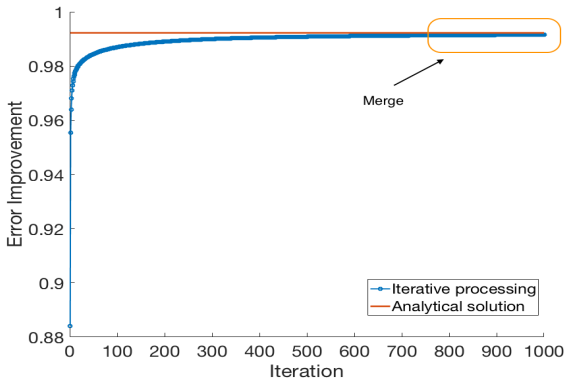


Figure 12: Convergence. Iterative solution converges to the analytical solution.

BOLIFT. In this example, both approaches are given the same initial solution $\vec{z}_0$ produced by the baseline H-FUSE [87] to reconstruct the series of the weekly counts of measles infections in NY from multiple aggregated reports. The x-axis is the iteration count, and the y-axis indicates the improvement of each solution (iterative and analytical) over the initial solution $\vec{z}_0$ (Eq.(13)), the higher value, the better. The blue dotted line shows the error improvement of the iterative solution with iterations (each dot represents the quality of the solution at a given iteration), and the red horizontal line shows the error improvement value of the analytical solution. Evidently, the iterative TURBOLIFT steadily improves the reconstruction accuracy and converges to the analytical solution after a certain number of iterations. In the experiment of Figure 12, the iterative solution consumes more than **1000x** the computational time of the analytical solution in order to reach the same accuracy (**55 seconds compares to 0.05 seconds**).

3.2.2.1 Complexity of Analytical TURBOLIFT

By looking into Algorithm 2, we can see that the most time-consuming step of the analytical solution is computing the Singular Vector Decomposition (SVD). For a matrix $\mathbf{O} \in \mathbb{R}^{M \times N}$, the

complexity of traditional SVD is usually proportional to

$$\mathcal{O}(aM^2N + bN^3),$$

where a and b are two parameters [49]. Therefore, the cost of SVD of $\mathbf{O}^T\mathbf{O} \in \mathbb{R}^{N\times N}$, in our case, is nearly $\mathcal{O}(N^3)$. However, several efficient SVD algorithms have been proposed recently to handle the computational cost of SVD, especially for large-scale sparse matrices, such as Fast Randomized SVD [113], QUIC-SVD [62], and Fast stochastic algorithm for SVD [123].

Table 2 shows the practical run-time of Fast Randomized SVD in terms of matrix size and sparsity. Since the matrix $\mathbf{O}^T\mathbf{O}$ is a banded matrix with a bandwidth $b = (2RD_{max} - 1)$, the number of non-zero elements in $\mathbf{O}^T\mathbf{O}$ is less than bN, where $b \ll N$ in practical settings. Consequently, the practical run-time of the analytical TURBOLIFT scales well as we will show in Section 3.4.4.

Table 2: Run-time of Fast Randomized SVD on large scale matrices [113]

Row	**Column**	**Number of non-zeros**	**Time**
10^6	10^5	10^7	1 second
10^6	10^5	10^8	5 seconds
10^5	10^5	dense	120 seconds

3.2.3 TURBOLIFT: Impact of Initial Solution

Owing to the fact that Eq.(9) is non-convex, TURBOLIFT converges to different stationary solutions when it starts from different initial vectors. To demonstrate the impact of the initial solution, we generate a contour plot in Figure 13 to show the Root Mean Square Error (RMSE) of TURBOLIFT reconstruction results starting from different initial points. In this plot, the initial vectors fed to TURBOLIFT are built by adding noise to two randomly chosen time-ticks in the original series (ground-truth). In other words, the distance between the initial vector to the ground truth vector is

$$\sqrt{(Noise\ 1)^2 + (Noise\ 2)^2}.$$

The x-axis and y-axis represent the deviation between the value of two noisy time-ticks and its real value, respectively. The color corresponds to the value of the RMSE of TURBOLIFT solution starting from different noisy initial vectors.

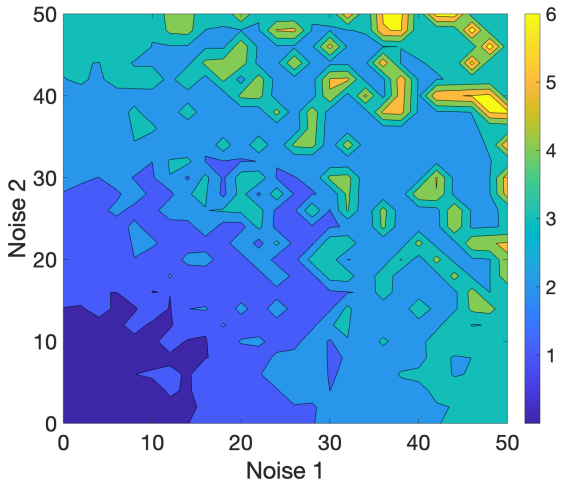


Figure 13: The impact of the initial solution on TURBOLIFT reconstruction.

Figure 13 shows that the solution provided by TURBOLIFT converges to different points with different levels of RMSE, which demonstrates that the solution quality depends on the starting point. As the initial vector gets closer to the real solution, the estimation of TURBOLIFT becomes more accurate. As such, we conclude that TURBOLIFT reconstruction with an initial vector that has higher accuracy leads to a better result. We recommend choosing the algorithm from the state-of-the-art disaggregation methods to obtain $\vec{z}_0$ to initialize TURBOLIFT carefully based on the structure of the series of interest. In order to help with the selection of the initial vector $\vec{z}_0$ under different scenarios, we provide a practitioner's guide in Section 3.4.3. However, as we will see in the experimental results in Section 3.4, TURBOLIFT never reduces the accuracy of the initial reconstruction, following the "first, do not harm" principle.

3.3 EXPERIMENTAL SETUP

In this section, we provide a detailed description of the setup we use in our experiments. First, we describe the datasets used in the experiments. Then, we explain the strategy applied to the data to generate aggregated observations. Last, we present the baselines and evaluation metrics.

3.3.1 Data

In order to test the effectiveness and generality of ARES and TURBOLIFT, we use data from different domains, which are readily available online:

- *Epidemiological data*: The data is from project Tycho [131], which contains real epidemiological time series of weekly reported cases in the 50 US states spanning more than 100 years. We choose the time series of Measles in New York and Smallpox in California), which visibly exhibit different patterns—see Figure 14. In this repository, most time series does not cover all the weeks within the covered period, i.e., has missing values. To assess the performance with data exhibiting periodic and/or smooth structure, we choose 400 *consecutive* weeks without missing values for each series. We refer to these two series as NY Measles and CA Smallpox.
- *Retail sales data*[2]: Historical weekly sales data made available online for 45 Walmart stores in different locations. We present results using Department number 1 in Store number 1 in this data, as it has some spikes due to the high demand in some weeks (i.e., spikes around Christmas and New Year). This serves our purpose in analyzing the performance with data with different patterns.
- *Criminological data*[3]: Reported incidents of crime that occurred in the city of Chicago from 2001 to the present. Each incident is marked with a code indicating the crime type. There are 388 crime types in total in this dataset. We chose the *theft* time series in our experiments as it is the densest among all the other crime types. The series in our testing is in a weekly resolution and covers 400 weeks.

[2]https://www.kaggle.com/bletchley/course-material-walmart-challenge
[3]https://www.kaggle.com/chicago/chicago-crime/activity

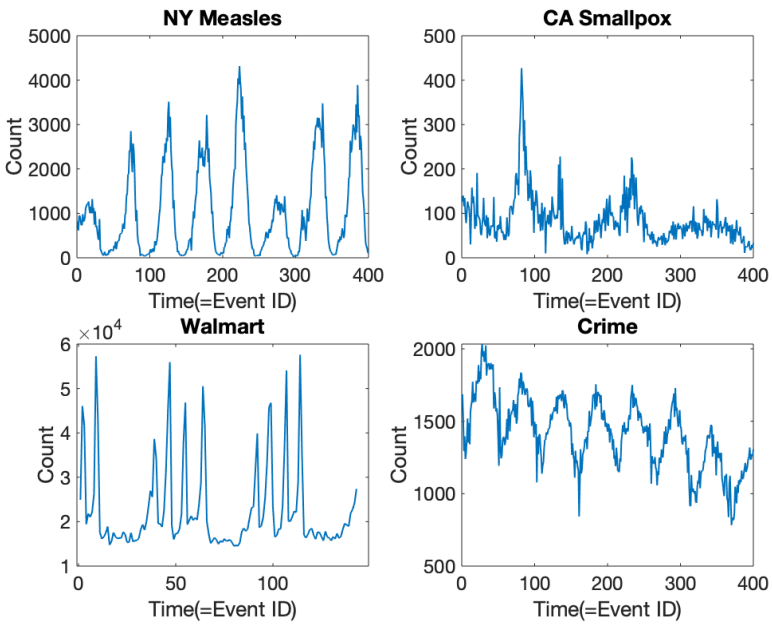


Figure 14: Time series used in the experiments.

Figure 14 shows all the time series described above, x-axis represents the time-ticks (i.e., weeks), and the y-axis represents the value (i.e., the count of infected patients in the epidemiological data, number of sold items in Walmart data, and number of crime incidents in the crime data). As observed, each dataset has a notably different pattern, e.g., varying degrees of smoothness and periodicity. This provides an abundant test set to evaluate the performance of both methods (ARES and TURBOLIFT).

In order to demonstrate the benefit of TURBOLIFT that relieves the imposed inaccurate constraints, we test TURBOLIFT on two extreme cases, sparsity and spiky, besides the datasets described above. The results are include in the Appendix A.2

3.3.2 Aggregation configuration

For all datasets, we use the weekly event counts as the finest granularity (ground-truth) and refer to each week as a time-tick. We generate coarse aggregate observations from the ground truth to be used as inputs to the disaggregation methods. In the experiments, we generate different

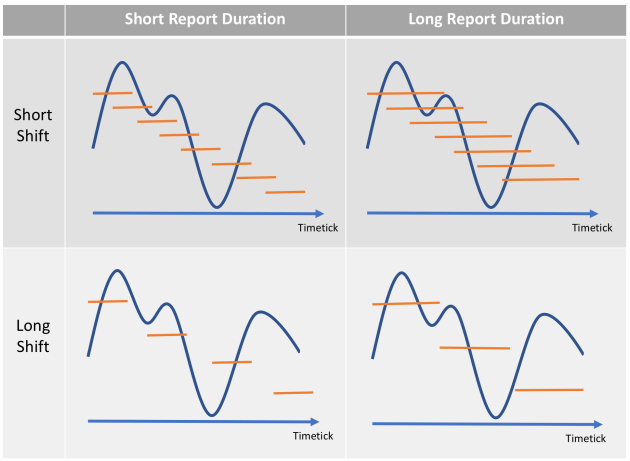


Figure 15: Different aggregation configurations.

aggregated observations (*reports*) covering successive time-ticks. More specifically, we explore different values of report duration, but we use the same report duration for all the reports in the linear system. Recall that *Report Duration(RD)* is the number of time-ticks covered by one report. The difference between the starting time-tick of consecutive reports is referred to as *Shift* (*report frequency*), as explained in Figure 2. We generate different aggregated observations by varying the *RD* and the *Shift* to test the sensitivity of ARES and TURBOLIFT. The reports start from the beginning of $\vec{x}$ (x_1) until the last report hits the last element of $\vec{x}$ (x_N). Varying the *RD* and *Shift* allows our experiments to include different scenarios, including overlapping and non-overlapping cases. Figure 15 shows different sampling configurations with short/long *RDs* and short/long *Shifts*. Each orange line represents a report. As one can see, with short *Shifts* and short *RDs*, we have good report coverage. On the other hand, long *Shifts* and short *RDs* result in gaps between the reports, i.e., some time-ticks are not covered by any report. With short *Shifts* and long *RDs*, the reports overlap and introduce redundant information. In contrast, when the reports have long *Shifts* and long *RDs*, the amount of information from the aggregated reports is not sufficient for accurate reconstruction.

We change the *RD* ranging from 2 to 52 and *Shift* ranging from 1 to 26 in our experiments. Those ranges have been chosen with the awareness that the historical data have a prominent yearly periodicity of 52 weeks. Specifically, $RD = 52$ means that each report covers a year, and $Shift = 26$ means that the difference between the starting time-tick of reports is half a year.

3.3.3 Evaluation metrics and baselines

We compare our proposed methods with the state-of-the-art disaggregation methods explained in Section 2.1.2, including 1) **LSQ**: which finds $\vec{x}$ with minimum norm (min $\|\vec{x}\|_2^2$) and 2) **H-FUSE**: where smoothness and periodicity constraints are imposed on the solution to the linear system with equal weight. We select H-FUSE because Liu et al. [87] claim that it performs better than the other constraints (i.e., only smoothness or periodicity).

We measure the improvement by comparing the RMSE of the baseline solution with the RMSE of the solution provided by the proposed method. In addition, to compare the RMSE improvement among various configurations, we design an *error change rate* equation, formulated as

$$Rate = \frac{RMSE_{Baseline} - RMSE_{Proposed\ method}}{\max(RMSE_{Baseline}, RMSE_{Proposed\ method})} \tag{13}$$

where the $RMSE_{Proposed\ method}$ is the RMSE of the reconstructed series using ARES or TURBOLIFT (with the initial vector $\vec{z}_0 = \vec{x}_{Baseline}$), and the $RMSE_{Baseline}$ is the RMSE of the baseline reconstructed result $\vec{x}_{Baseline}$. The value of the rate represents the degree of improvement (positive) or degradation (negative) of proposed method compared with the baseline method.

3.4 EXPERIMENTAL RESULT

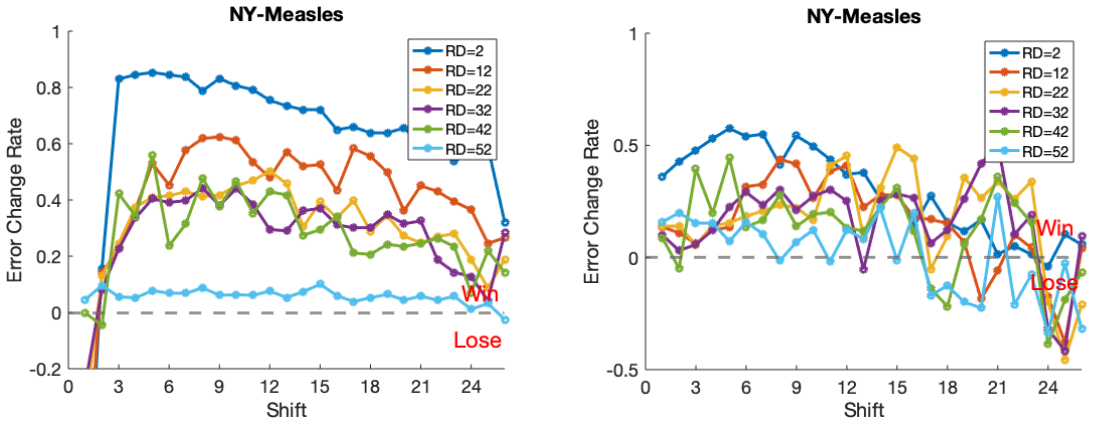
In this section, we demonstrate the effectiveness of ARES and TURBOLIFT. We present and analyze the experimental results in terms of the following aspects: 1) the performance of proposed methods in terms of disaggregation accuracy, 2) discussion of the observations about the performance, and 3) scalability in terms of run-time with data size and sparsity.

3.4.1 Effectiveness Evaluation

We investigate the disaggregation accuracy of ARES and TURBOLIFT with different aggregation configurations using real data from different domains.

In Figure 16 and 17, we plot the error change rate of ARES/TURBOLIFT in comparison with LSQ and H-FUSE/H-FUSE and ARES for several report configurations using the normal NY

Measles data, respectively. Since LSQ reconstruct the sequence by imposing no additional information as constraint, we do not include it as an initial solution for TURBOLIFT. In both figures, the x-axis represents the *Shift* ranging from 1 to 26, and the y-axis shows the error change rate following Eq. (13). Each line represents a specific *RD*, ranging from 2 to 52 with increments of 10. In this case, the results cover all the types of configurations described in Figure 15.



(a) ARES compared to LSQ on NY Measles

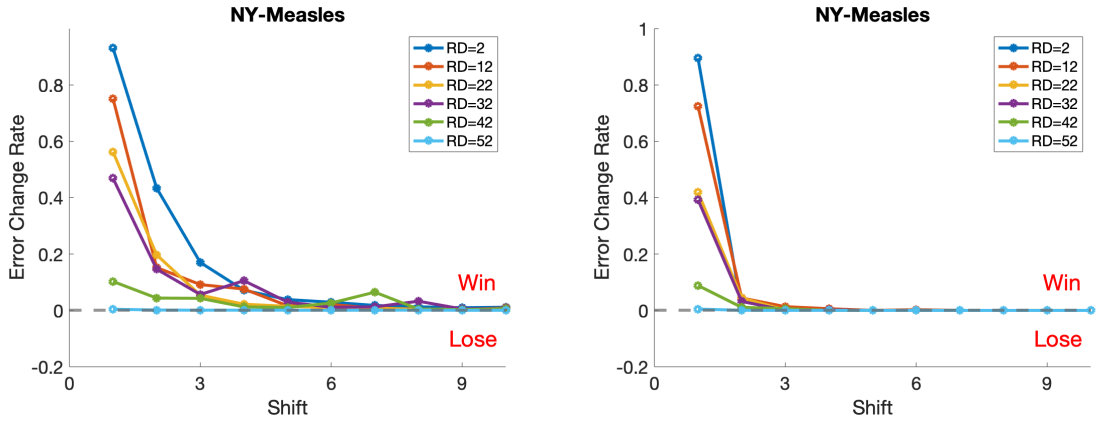
(b) ARES compared to H-FUSE on NY Measles

Figure 16: Error change rate of ARES on NY Measles.

We observe that in most cases, ARES considerably outperforms the baseline methods (see Figure 16), and the improvement can achieve up to 80%. However, it is also undeniable that in a few cases, ARES fails in the competition with baseline methods. We explain each scenario as follows. 1) LSQ works better around $Shift = 1$ is because each time-tick, x_n, is covered by two reports (i.e., $shift = 1$) except for the first and last weeks, resulting in an easy problem for LSQ solution since $\mathbf{O}$ is "almost" square and full rank. 2) ARES loses H-FUSE with larger $Shift$ values (e.g., $Shift \geq 16$). Obviously, a larger $Shift$ causes fewer available reports and makes the problem more difficult. Moreover, note that H-FUSE imposes smoothness and periodicity on the solution. NY Measles data do not have an exact annual periodicity, but it is visually smooth. For instance, although the peak values appear in every summer season, the difference between these values is significant, e.g., 1048 patients in week 25 versus 3771 in week 225 because of the impact of vaccination. Therefore, by taking the inappropriate reconstructed result of H-FUSE as basic in

the first phases to generate a dominant pattern, ARES may get a worse result.

Except for the particular scenarios we explained above, there are few cases in plots where ARES fails to over-perform the competitors. We claim that the failure of these cases is contributed by inaccurate approximation in the first phase. The pattern discovered by AF may mislead the final reconstruction when it cannot capture the critical information from the estimation from the first phase. However, it is noteworthy that the scale of the loss is much lower compared to the error improvement that ARES demonstrates in most cases.



(a) TURBOLIFT compared to H-FUSE on NY Measles

(b) TURBOLIFT compared to ARES on NY Measles

Figure 17: The error change rate of TURBOLIFT on Tycho NY Measles data.

Next, we investigate the performance of TURBOLIFT. As Figure 17 shows, for both cases, when *Shift* is less than 3, the improvement is larger for smaller RDs. For example, with $RD = 2, Shift = 1$, TURBOLIFT dramatically decreases the RMSE to reach almost 0, providing a very accurate reconstructed series. As the *Shift* increases, the effect of the *RD* value on the error change rate becomes less notable. We truncate the range of *Shift*, since for larger *Shift* (> 10), the error change rate of TURBOLIFT is nearly at the zero line. We conclude that, in those cases, the given aggregated samples are insufficient to provide more information for refining the initial reconstruction. We explain this phenomenon in the following: note that increasing the *RD* and/or *Shift* would decrease the number of reports (equations), and the number of equations in the linear system $\mathbf{O}\vec{x} = \vec{y}$ is essential in our method and the baseline methods used for initialization. For the proposed method, having more reports/equations in the system would constrain the solution

to be closer to the target sequence since the optimization problem is constrained by the linear system $\mathbf{O}\vec{x} = \vec{y}$. This is also the case for the initial methods (H-FUSE and ARES), which are based on the least-squares ($\|\vec{y} - \mathbf{O}\vec{x}\|_2^2$) of the linear system. As the number of equations increases, the least-squares solution is constrained to be closer to the actual sequence. In all methods, as the number of equations decreases (i.e., large *Shift* and/or *RD*), the degree of freedom of $\vec{x}$ in $\|\vec{y} - \mathbf{O}\vec{x}\|_2^2$ increase. Thus, the quality of the solution relies more on the imposed constraints, e.g., smoothness, or periodicity. In other words, for our proposed method, the solution relies more on the regularization term $\|\vec{x} - \vec{z}\|_2^2$ in (Eq.(9)). In this case, the solution provided by TURBOLIFT is closer to the initial solution obtained from the baseline. However, in all cases, the error change rate is non-negative, which demonstrates the steady recovery and "first, do no harm" principle of TURBOLIFT.

It is clear to observe that the improvement of TURBOLIFT initialized using H-FUSE is better than ARES. The reason is NY Measles does not follow precisely the smoothness and periodicity constraint (see explanation above), and one major advantage of TURBOLIFT is to "release" these inaccurate constraints imposed on the existing reconstruction solution from inaccurate constraints. Therefore, H-FUSE has more room to be improved.

In order to quantitatively measure the improvement of both methods, in Table 3 and 4, we present the average RMSE of ARES, TURBOLIFT and the baseline methods for several RDs (same range as used in the figures). Specifically, we gather the full range of *Shift* for Table 3, while consider the *Shift* values only in the range of 1 to 10 for Table 4, to avoid mitigating the impact of TURBOLIFT. In most cases, except when $RD = 52$, we have a significant improvement, given that the extra execution time added by ARES is 0.35 and by TURBOLIFT is only 0.05 seconds on average. However, the aggregation when $RD = 52$ is very aggressive (yearly samples), resulting in a small number of reports that can not capture the detailed structure of the series. Figure 18 shows three examples of the reconstructed series using H-FUSE with $RD = 52$ and $Shift = 1, 5$ and 10. As we observe, even with more comprehensive reports coverage ($Shift = 1$), the reconstructed series is very inaccurate. In these cases, neither of the proposed methods or the baseline methods have enough constraints in the linear system ($\mathbf{O}\vec{x} = \vec{v}$) to generate an accurate result.

Next, we show the error change rate when compared to the baselines using datasets from different domains in Figure 19 and Figure 20. The error change rate with all the datasets shows a

Table 3: The average RMSE of ARES compared to different baseline methods.

	RD = 2	RD=12	RD = 22	RD = 32	RD = 42	RD = 52
LSQ	954.5815	319.9840	288.7527	274.5531	378.5227	930.2356
H-FUSE	367.9551	221.1754	222.3000	226.2674	308.8957	987.0710
ARES	**181.5022**	**159.7902**	**181.6240**	**178.9682**	**237.5226**	**866.7910**

Table 4: The average RMSE of TURBOLIFT compared to different baseline methods.

	RD = 2	RD=12	RD = 22	RD = 32	RD = 42	RD = 52
H-FUSE	367.9551	221.1754	222.3000	226.2674	308.8957	987.0710
TURBOLIFT	**333.0311**	**206.4655**	**209.1427**	**212.3931**	**299.9737**	**986.0033**
ARES	181.5022	159.7902	181.6240	178.9682	237.5226	866.7910
TURBOLIFT	**172.9991**	**151.9155**	**176.1008**	**173.8196**	**235.0734**	**866.3280**

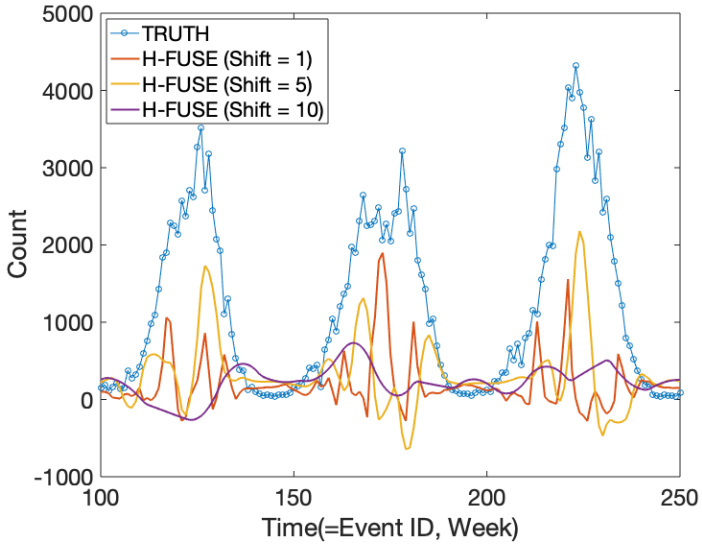
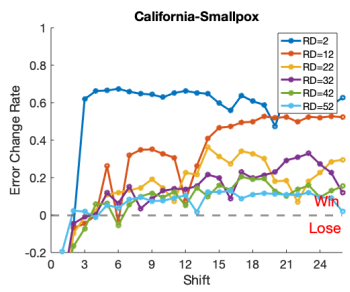
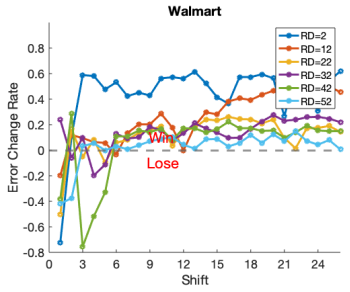


Figure 18: H-FUSE reconstruction results of NY Measles when $RD = 52$.

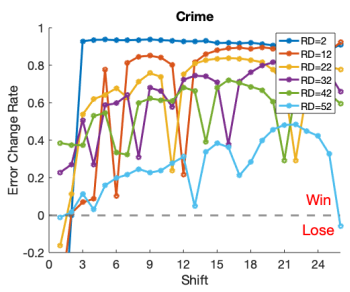
similar trend as with NY Measles in Figure 17. However, we notice a special phenomena on the



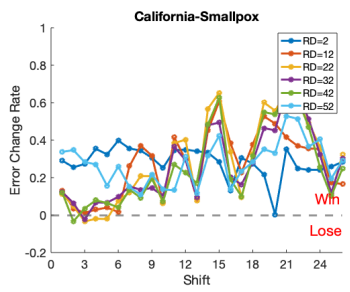
(a) ARES compared to **LSQ** on California Smallpox



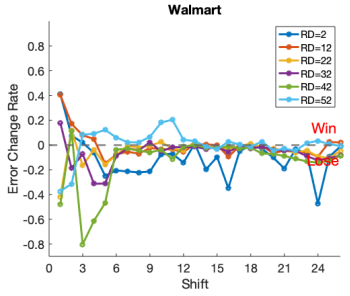
(b) ARES compared to **LSQ** on Walmart



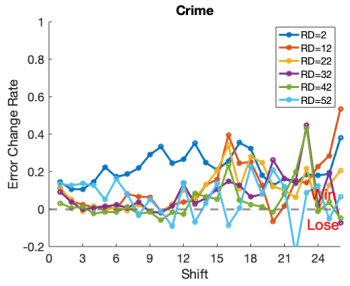
(c) ARES compared to **LSQ** on Crime



(d) ARES compared to **H-FUSE** on California Smallpox

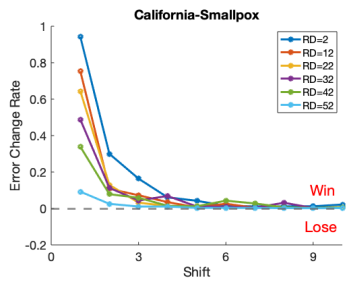


(e) ARES compared to **H-FUSE** on Walmart

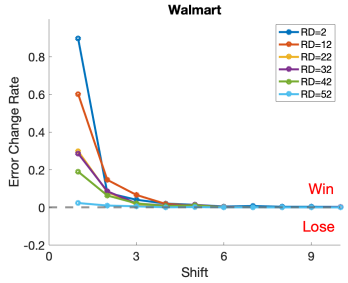


(f) ARES compared to **H-FUSE** on Crime

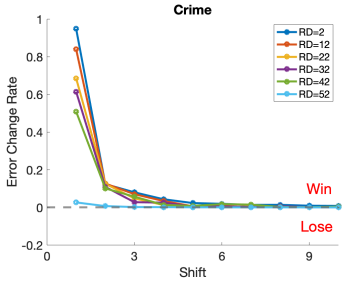
Figure 19: ARES error change rate on different datasets comparing to different baseline.



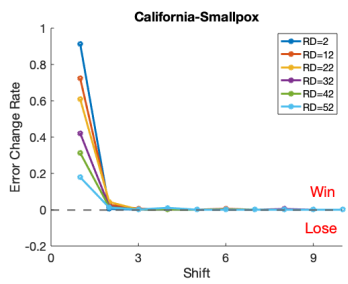
(a) TURBOLIFT compared to **H-FUSE** on California Smallpox



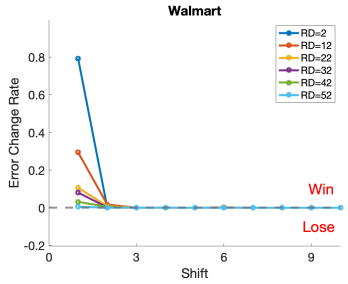
(b) TURBOLIFT compared to **H-FUSE** on Walmart Sales



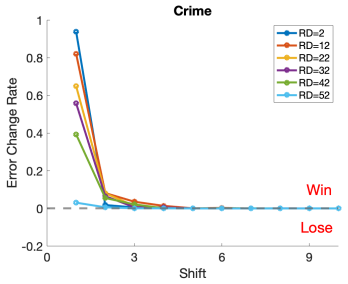
(c) TURBOLIFT compared to **H-FUSE** on Chicago Crime



(d) TURBOLIFT compared to **ARES** on California Smallpox



(e) TURBOLIFT compared to **ARES** on Walmart Sales



(f) TURBOLIFT compared to **ARES** on Chicago Crime

Figure 20: TURBOLIFT error change rate on different datasets with different initials.

Walmart data, especially when the result is associated with ARES: 1) when ARES is compared with H-FUSE (Figure 19 (e)), ARES performance even worse; and 2) when TURBOLIFT is compared with ARES (Figure 20 (e)), TURBOLIFT only improves the performance of ARES reconstruction at $Shift = 1$. This is because Walmart data has several spikes around weeks 50 and 100, making it difficult for the Annihilating Filter to discover a clear and dominant pattern. In this case, the constraint in ARES misleads the reconstruction process and thus affects the performance of TURBOLIFT further.

As explained above in the experiments and Section 3.2.3, both the estimation in the first phase of ARES and the initial vector of TURBOLIFT are associated with the final reconstruction performance. In other words, a reconstruction solution, which cannot reflect the nature of the real data, brings risks to the following step. Therefore, to understand and avoid this situation in practice, we discuss the impact of the first phase approximation on the reconstruction accuracy of ARES and selection of initial vector in TURBOLIFT in the following section.

3.4.2 Impact of the First Phase Approximation in ARES

In this section, we explore the impact of the first phase approximate sequence estimation on ARES performance.

We compare ARES with an alternative approach, called ARES+, that uses a large part of the original sequence to learn AFs at the first phase. Since the original sequence is commonly unavailable, ARES+ is not very practical, and we use it only to assess ARES.

The expectation is that ARES+ could learn more accurate AFs from the original sequence and perform a better reconstruction than ARES primed with H-FUSE approximation of the sequence. We also expect that the performance gain of ARES+ compared to ARES should not be very significant. This would prove that the first phase approximation recovers enough information about the original signal to perform an efficient second phase reconstruction.

Indeed, Figure 21 shows that ARES+ slightly outperforms ARES in recovering New York Measles data. However, the performance gain is much less compared to how ARES outperforms the competitors (H-FUSE and LSQ). These results confirm that the knowledge of the original sequence would help to improve the reconstruction quality. However, this improvement is not very

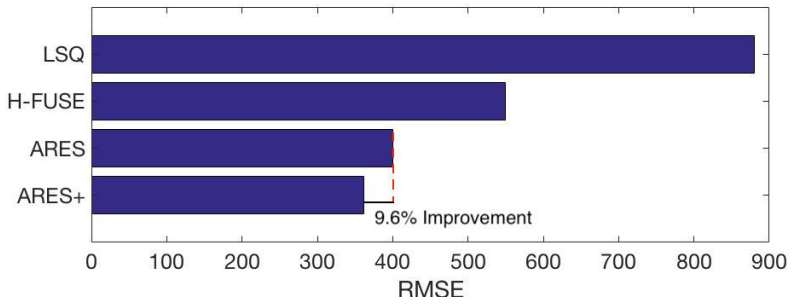


Figure 21: Performance of ARES and ARES+ is comparable.

significant compared to ARES. Meanwhile, in a realistic setting when the original data is not available, ARES is a robust and sustainable approach that ensures highly accurate data reconstruction.

3.4.3 Practitioner's Guide for Disaggregation methods selection

Due to the different performance of each method under different datasets, we provide some guidance based on the insights we gained from the previously discussed experiments to help the readers select the appropriate disaggregation method for different types of datasets. This guide could also be applied for selecting the initial vector of TURBOLIFT.

- **Periodic** data: If the time series has a clear periodical pattern, then H-FUSE is preferable, as the periodicity constraint penalizes the differences between successive periods. By clear periodicity pattern, we mean that there is a repeating pattern, for example, the weather observation time series.
- **No domain knowledge**: If no domain knowledge is provided, ARES is recommended as it discovers the hidden pattern automatically. However, note that in the case of the presence of spikes, ARES should be avoided because it tries to discover the pattern within a few neighboring time-ticks, and spikes could disturb the pattern estimation.
- **RD close to the intrinsic period**: If the RD value for the majority of the reports is close to the prominent period in the data, then all the methods may have large errors. The intuition is that it is hard to recover the data structure within a period (e.g., peak in the summer in NY Measles),

if only the periodical summations are given.

Last but not least, for any reconstruction method, TURBOLIFT is recommended to be used as the last step to refine the solution and improve its accuracy. The amount of improvement varies depending on multiple factors, as we analyzed above. Nevertheless, this step will not reduce the quality of the initial solution, and the computational complexity of this extra refinement step is very small.

3.4.4 Scalability

In this section, we examine the scalability for proposed reconstruction methods – ARES and TURBOLIFT.

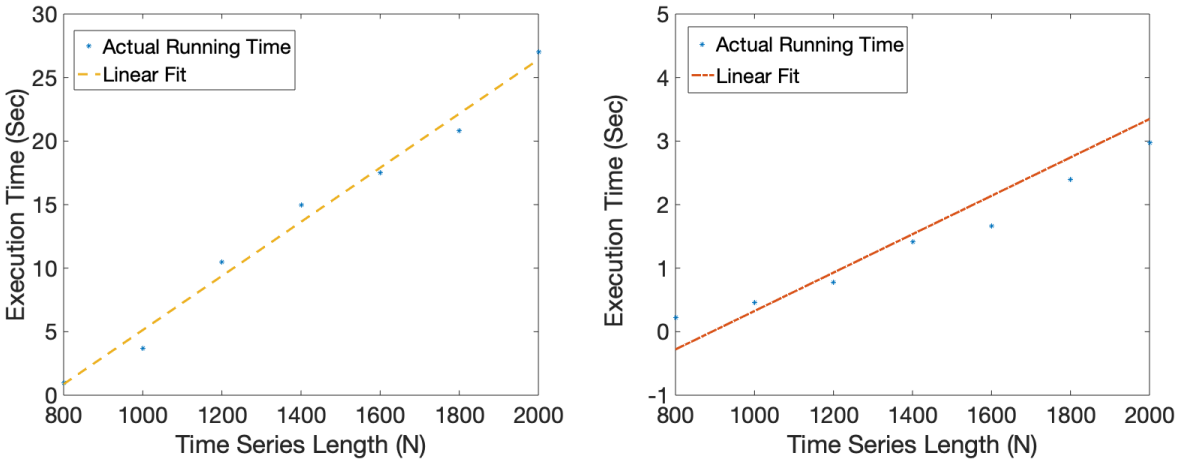
According to the complexity analysis in Section 3.1.3 and 3.2.2, the time complexity of ARES is almost $Nlog(N)$ and the complexity of TURBOLIFT depends on the SVD step in Eq.(12), which can be computed by Fast SVD [113] efficiently, especially with sparse matrices.

To better assess the scalability of each method, we show how the execution time varies with the length of the time series in Figure 22. The *x*-axis is the length of the time series (N), and the *y*-axis is the execution time in seconds. The execution time of both solutions shows a linear relation with respect to the length of the time series, which proves that both methods scale well.

3.5 Conclusion

In this chapter, we introduce two data disaggregation methods and demonstrate that the aggregated data could be reconstructed to finer resolution for further analysis. The contributions of our work are summarized as follows.

- **Proposed framework for reconstructing and refining the reconstruction accuracy**: We proposed two novel formulations to improve the accuracy of data disaggregation.
- **Scalability for large data**: We proved that both of these two methods scale well in terms of the length of time series.
- **Effective on real-world data**: We conducted comprehensive experiments on *real historical*



(a) Execution time of ARES.

(b) Execution time of of TURBOLIFT.

Figure 22: Execution time. Both methods scale well with a linear relation.

data from different domains. Our experimental results demonstrate the outstanding performance of the proposed approaches.

In addition to these two works, by collaborating with a group of people from the University of Minnesota, Carnegie Mellon University, and the University of Virginia, I worked on another algorithm for data disaggregation, named HOMERUN [5]. In this method, we formulate the problem as so-called *basis pursuit* using the Discrete Cosine Transform (DCT) as a sparsifying dictionary *and* impose non-negativity and smoothness constraints. This work has been published in Proceedings of the VLDB 2019.

4.0 DATA NAVIGATION: INTEGRATED DATA WAREHOUSE WITH INTELLIGENT DECISION NAVIGATOR

Although, as we analyzed in the previous chapter, data disaggregation methods could reconstruct time series with acceptable accuracy and provide an elaborate understanding of the original data, they inevitably request extra computational cost, and this cost is unignorable, especially for big data. Moreover, the performance of reconstructed data is not stable for all report configurations. For example, the spikes cannot be captured by large RD and thus result in a worse estimation of data. Therefore, a strategy for efficiently processing the aggregated big data for future analysis is urgent. Fortunately, we notice that, in some tasks, analyzing using the aggregation data has the ability to provide a compatible result as raw data, e.g., predicting the patients' status with continuous monitoring records. Therefore, in this chapter, we first present a novel time-series data warehouse to efficiently store the aggregated data and then propose a navigation method to bypass the reconstruction process and automatically suggest aggregation levels for specific tasks, which maximizes the analysis accuracy under different machine learning models, on top of the data warehouse. The main idea of navigation is to take advantage of the automatic machine learning (AutoML) techniques to select an appropriate sequential combination of an aggregation report and a machine learning algorithm. In the end, we expect to discover that the aggregated data can achieve competitive results as the raw data.

To better evaluate the performance of the proposed method, we introduce a well-defined and challenging problem in the field of medication: *prognostication comatose survivors of cardiac arrest.* The reasons for selecting this problem are: 1) it contains a vast amount of data (i.e., 4.5 $\times 10^8$ data points for a single patient in one day). In this case, data aggregation is recommended to avoid computation-cost and communication-cost. 2) the target of this problem is binary classification and may not strongly associate with the detail of data, which makes the coarse data desirable.

In Table 5, we summarize the symbols used frequently throughout this chapter.

Table 5: Notations used in Chapter 4

Notation	Description
$\mathbf{X} \in \mathbb{R}^{m \times n}$	The medical records of patients.
$\vec{y} \in \mathbb{R}^{m}$	The real status of patients, i.e., awake or unconscious.
m	The number of time-ticks covered by each medical record.
n	The number of patients.
$\mathcal{A}$	The set of all machine learning pipelines
A	The algorithms used in the machine learning pipeline.

4.1 BACKGROUND: PROGNOSIS AFTER CARDIAC ARREST

Cardiac arrest is the most common cause of death globally [89]. There are over 900,000 cases in the United States each year [12]. Thanks to improvements in care, increasing numbers of cardiac arrest victims are successfully resuscitated and survive to hospital admission. Most are initially comatose, and it is challenging to differentiate recoverable brain injury from irrecoverable injury for many days after cardiac arrest. Specifically, there is no combination of clinical findings and test results that preclude favorable recovery for at least 72 hours post-arrest, and the outcome is often uncertain much longer [20]. Moreover, an international data study shows most providers rely heavily on imprecise or debunked prognostic signs and thus influence the prognosis through a self-fulfilling prophecy bias [91]. For these reasons, many patients receiving aggressive care are then discovered to have never had a chance of recovery. This results in tremendous financial cost and emotional burden to families with no benefit. Conversely, withdrawing life-sustaining therapy from patients with the potential to recover results in avoidable deaths. Therefore, most healthcare providers and families choose a trial of aggressive care and delay decision-making for days until more data become available rather than make a hasty decision with limited information. Improving the speed and accuracy of post-arrest prognostication could save lives, allow appropriate resources to be directed to patients who are likely to benefit, avoid long and difficult care for patients who cannot recover, and spare families prolonged uncertainty.

4.1.1 Quantitative Electroencephalography (qEEG)

Electroencephalography (EEG) is a method to record the electrical activity of the brain. In the ICU, EEG is typically acquired across 22 channels at 256 Hz per channel [73]. Because EEG measures brain functions that are necessary for arousal and cognition, it is perhaps not surprising that characteristics of the EEG signal are strongly associated with severity of brain injury and potential for recovery [28, 42, 119, 138]. Past studies have explored quantitative EEG (qEEG) features like amplitude, fast Fourier transformed power spectra or suppression [139]. Despite growing research interest in qEEG, current standard clinical care is for an expert clinical provider to provide a summary interpretation of a daily EEG recording. This approach is costly, experience-dependent, and has limited between-expert reliability. To address these weaknesses, several recent studies have used ML models of qEEG to predict outcome [48, 72, 116]. Because there are infinite ways to generate quantitative features from EEG, strategies for these biological signals selection are important. Nagaraj et al. [95] propose to apply the least absolute shrinkage and selection operator (LASSO) method, which penalizes the small coefficients prior to a classifier (in this case, random forest). Amorim et al. [6] propose a penalized logistic regression model that penalizes features with a large regression coefficient to reduce the size of feature sets. However, regularization may hurt the model performance by removing useful information, especially when multiple features are correlated [141].

4.1.2 Major Challenges

Although several machine learning methods have been proposed as a potential tool to provide clinicians with useful clinical predictions, however, according to a recent survey[1] from the Society of Actuaries (SOA), only 60% of hospitals use ML in practice. Key technical challenges have limited the translation of ML in clinical medicine [90]. We summarize them as follows:

- **Data storage**: Vast amounts of clinical data are generated daily and mostly ignored or discarded because of limited infrastructure and the human capacity for data integration. In most cases, historical data are important for decision support. The ability to preserve and analyze all available data would benefit ML models, but a single day's clinical recording on one pa-

[1] https://www.soa.org/globalassets/assets/Files/programs/predictive-analytics/2019-health-care-trend.pdf

tient, which often includes multiple waveforms that are continuously recorded, results in many billion data points. Although our focus is prognostication after cardiac arrest, this problem is ubiquitous in modern medicine. Many patients undergo continuous cardiorespiratory monitoring in the hospital. Temporal trends in these data can predict deterioration but are not routinely preserved or available for retrospective review and analysis [107].

- **Data aggregation & Clinical records selection**: Heterogeneous clinical data are measured continuously in acute care settings such as the intensive care unit (ICU), from which an infinite variety of features can be summarized. Only some of these data and features predict the outcome. Appropriately selecting and aggregating the clinical records can significantly reduce the computational cost with minimal effect on the model performance.
- **Model & Hyperparameters selection**: According to the "no free lunch" theorem proposed by Wolpert and Macready [140], it is impossible to find a universal model that fits all problems. A large amount of time and effort is needed to find an acceptable good model and tune the hyperparameters.
- **Model transferability**: A good model for one medical problem usually needs to be re-designed and re-trained from scratch for different patients, practice patterns, and collected attributes, which may affect model selection. Being able to leverage the existing models with fine-tuning saves the effort of machine learning experts.

Next, we present the proposed method in the following order: 1) to address the challenge of *data storage*, we first present a dynamic health data warehouse, named BrainFlux [40], which efficiently stores and summarizes huge time series at different aggregation levels, 2) then by leveraging this scalable infrastructure as the data provider, we propose SMARTPROGNOSIS to tackle the other challenges.

4.2 BRAINFLUX: INTEGRATED DATA WAREHOUSING INFRASTRUCTURE FOR DYNAMIC HEALTH DATA

In this section, we present BrainFlux - a novel data warehousing technology that implements a holistic paradigm for the preservation, aggregation, and discovery of trends in large amounts of

data from continuously measured physiological processes. A unique feature of the BrainFlux is its efficient and information-preserving summarization of huge time series at different aggregation levels, which makes it highly scalable and applicable for a wide array of health monitoring tasks.

4.2.1 BrainFlux Architecture

A generalized BrainFlux architecture is shown in Figure 23. It has three major layers. At the bottom is a data collection and storage layer, where multiple clinically derived data streams are transferred for future aggregation and processing. Some data are EEG-based, while others are patient-level metadata, timestamped medication administration data, etc.. Note that data are heterogeneous and distributed in multiple files for each patient.

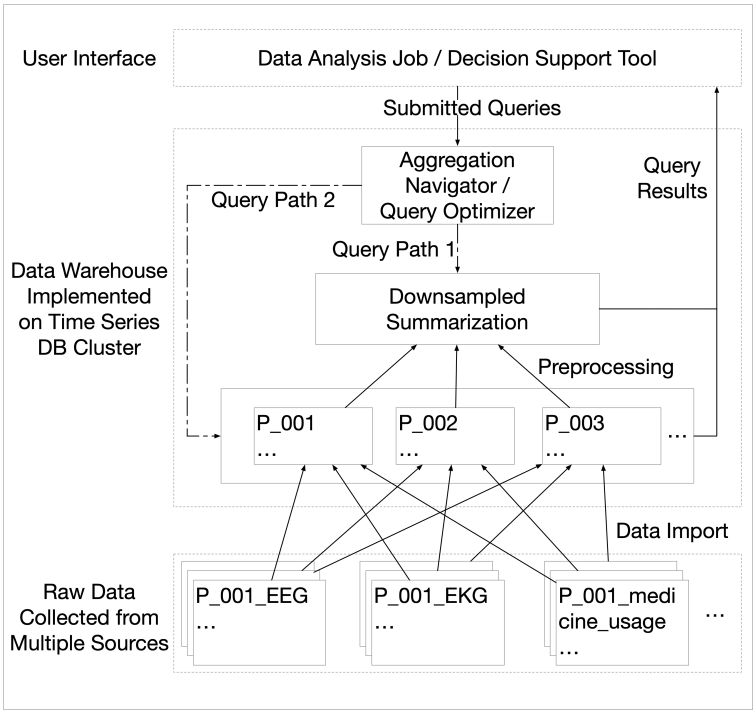


Figure 23: General BrainFlux Architecture

The BrainFlux imports raw data from the bottom layer and organizes them based on record timestamps. While imported data are theoretically suitable for data analytics, a considerable amount of data makes any meaningful analysis of these raw data challenging. Instead, time-varying summary measures of aggregated data are often of greater clinical and research interest. It is not

uncommon for BrainFlux to handle hundreds of terabytes of raw data on a regular basis, which risks creating a barrier between the data and end-user when the data are imported, organized, and aggregated. With a data transfer rate of 12MB/sec, for example, a user would have to wait a full day only for the import process on 1TB of data to be completed. To address this problem, BrainFlux takes advantage of system idle time to prepare aggregated data for future analyses based on user-defined parameters. For example, BrainFlux may generate 10-second, 60-second, and 120-second summaries (e.g., mean, median, variance, and between-electrode correlations) from the original imported data. One set of such summary measures may require a full week of processing time but is then added to the data warehouse as a new sub-layer. The top layer of BrainFlux is an aggregation navigator that intelligently selects the most appropriate data aggregation sub-layer in the warehouse to which to direct new user-defined queries.

The navigator is designed to maximize new query efficiency while preserving information accuracy for data-driven decision-making. For user-defined problems, the aggregation navigator directs queries to the most appropriate sub-layer to maximize efficiency while preserving information. The performance boost is considerable. For example, if we wished to explore 1-hour mean values of two qEEG trends averaged across all electrodes, the system would direct this aggregation task to the lowest resolution data that support this query without information loss. Using 120-second means instead of 1Hz data provides a linear (120x) boost to data processing time.

4.3 SMARTPROGNOSIS: INTELLIGENT DECISION NAVIGATOR

In this section, we utilize the aggregated data in BrainFlux and propose an intelligent decision navigator, which suggests sequential combinations of an appropriate data aggregation level with a machine learning model for this prognosis comatose survivors problem. Similar to the aggregation navigator mentioned above, this decision navigator can be embedded at the top layer of BrainFlux as well. In general, most ML models in medicine share a common sequential process: selecting important biological signals, creating and summarizing features from these signals, then using these features for prediction. We refer to this as an ML pipeline. Building on this approach, we propose an end-to-end automatic ensemble classification framework, named SMARTPROGNOSIS,

which automatically generates candidate pipelines tuned to maximize sensitivity for prediction of poor outcomes (i.e., unrecoverable) at a user-defined near-zero false positive rate for misclassification of patients with good outcomes (i.e., recoverable). This performance metric is informed by surveys of clinical providers, who view the withdrawal of life-sustaining therapy from patients incorrectly identified as likely to have a poor outcome as unacceptable [128]. We will explain this in detail later. SMARTPROGNOSIS applies ensemble learning on candidate pipelines to enhance performance. The intuitive idea is to optimize and select high-performing ML pipelines without involving experts then further improve prognostic performance by ensembling the outputs, the results of which are then provided to experts. Figure 24 illustrates the workflow of SMARTPROGNOSIS– the upper part shows the automatic candidate pipeline selection, and the lower part shows the prognosis with ensemble learning. We demonstrate the effectiveness of SMARTPROGNOSIS on real data by comparing it to commonly used alternative approaches that are manually tuned by experts.

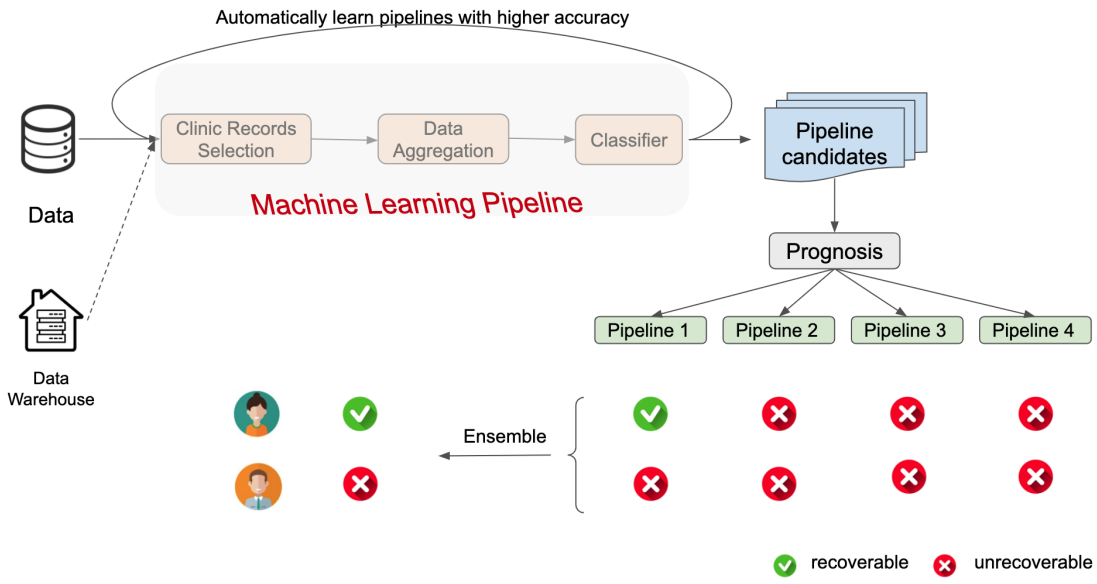


Figure 24: The framework of SMARTPROGNOSIS.

4.3.1 Proposed Framework

In this section, we provide a detailed description of SMARTPROGNOSIS. We start by introducing the implementation of AutoML and then explaining the cost function that helps for machine learning pipeline selection. In the end, we present how ensemble learning enhances accuracy.

4.3.1.1 Automated Machine Learning

Automated Machine Learning (AutoML) has recently emerged as a sub-field of ML, which aims to make ML techniques easier to apply and reduce the demand for expertise [109]. AutoML can automatically select pipelines that maximize a particular cost function that is relevant for a given content domain. The definition of the ML pipeline is as follows.

Definition 4.3.1 (Machine Learning Pipeline)**.** An ML pipeline is a combination of various algorithms that takes an input vector $\vec{x} \in \mathrm{R}^{n \times m}$ and outputs a target value $\vec{y} \in \mathrm{R}^{n}$. The set of all algorithm combinations can be represented as $\mathcal{A} = \{\mathbf{A}_{Data} \cup ... \cup \mathbf{A}_{Model}\}$, where $\mathbf{A}_{Data}$ and $\mathbf{A}_{Model}$ represent a set of pre-processing algorithms and ML models respectively. The pipeline can include more kinds of algorithms than just these two. Each algorithm is defined by a hyper-parameter vector $\vec{\lambda} \in \mathrm{R}^{k}$.

Specific to our clinical setting, the ML pipelines takes patients' biological record ($\vec{x}$), such as qEEG and demographics, as inputs to predict the patients' outcome $\vec{y}$. As mentioned above, the pipeline includes three steps, biological signals selection ($\mathbf{A}_{Select}$), data aggregation ($\mathbf{A}_{Aggregation}$), and classification ($\mathbf{A}_{Model}$).

AutoML algorithms have been successfully implemented using techniques such as Sequential Model-Based Optimization(SMBO) [45, 77], Genetic Programming [97], and Gradient Descent [104] etc. AutoML performs well compared to ML models manually tuned by experts.

4.3.1.2 Genetic Programming for Pipeline Selection

To automate pipeline selection, we use the Tree-Based Pipeline Optimization Tool (TPOT) [97]. We chose TPOT because 1) it over-performs other AutoML algorithms [150], and 2) it has a more flexible structure and looser constraints [97], which makes it easier to customize for our needs. For

example, in TPOT, the cost function does not have to be differentiable.

TPOT is based on Genetic Programming (GP) [79], an iterative, domain-agnostic optimization method inspired by biological evolution. In the beginning, a list of candidate ML pipelines, called a population, is randomly initialized, evaluated on the input data set via a cost function, and ranked in order. The initialization is done by randomly selecting algorithms from a pool, then randomly selecting hyperparameters for the chosen algorithms. For example, the logistic regression is initialized with the L1 penalty by giving the selection of L1, L2, elastic-net, and none. Then a predefined proportion of the ordered population is selected to transfer to the next generation while poorly performing candidate pipelines are removed. Descendant pipelines are generated in one of three ways: 1) *crossover*: a pair of parents randomly exchange a subset of the pipeline (e.g., parent pipelines $P_1 = \mathbf{A}_{Data_1} \cup \mathbf{A}_{Model_1}$ and $P_2 = \mathbf{A}_{Data_2} \cup \mathbf{A}_{Model_2}$ exchange their data preprocessing algorithm to generate two new child pipelines $C_1 = \mathbf{A}_{Data_2} \cup \mathbf{A}_{Model_1}$ and $C_2 = \mathbf{A}_{Data_1} \cup \mathbf{A}_{Model_2}$), 2) *mutation*: randomly mutate (add/remove/replace) a part of the pipeline, which can be either hyperparameter or algorithm (e.g., mutating a logistic regression classifier with L1 penalty to L2 penalty.), 3) *copy*: a predefined proportion of pipelines, ordered by the performance, is carried forward to the next generation without modification. This procedure is repeated for several iterations while maintaining a fixed population size (i.e., number of pipelines) across generations, then stops by either reaching the maximum iteration or until the pipeline with the highest performance remains constant for several iterations. One advantage of GP is that all the candidate pipelines in a generation are independent so that the process can be sped up by running in parallel.

Usually, the cost function used in GP is either accuracy, F1 score, or Area Under the Curve (AUC). Specific to this clinical domain, these performance metrics are less useful. Instead, an optimal model should be extremely conservative when it predicts a poor prognosis. The decision to continue life-sustaining therapy while additional information is gathered is viewed by clinicians as acceptable, even when additional data demonstrate no recovery potential. By contrast, the withdrawal of life-sustaining therapy for perceived poor neurological prognosis (or based on model output suggesting no recovery potential) invariably leads to the death of the patient. This type of error is unacceptable. Thus, higher overall accuracy does not necessarily mean a better model. Rather, a clinically useful model is identified as the greatest possible number of poor outcomes with near-zero misclassification of patients who actually recover. Formally, we consider, **False**

Negatives(*FN*, a patient has bad outcome but is not identified by the model) is much lower risk than **False Positives**(*FP*, a patient has good outcome but is misidentified, which could lead to a withdrawal of life-sustaining therapy based on the incorrectly predicted poor outcome, despite the true potential for recovery, i.e., an avoidable death). That said, a model with zero False Positives Rate (*FPR*) but low **Sensitivity** (True Positive Rate, *TPR*) is applicable in fewer cases than the one with high *Sensitivity* and zero *FPR*. Thus, we design a cost function to reflect this requirement. The intuitive idea is to fix the *FPR* at a conservative threshold (e.g., 0.01 or 0.05) and then maximize the *Sensitivity*. The customized cost function is as follows:

$$\mathcal{C} = Sensitivity(X, y \mid Pipeline, FPR), \tag{14}$$

where $X \in \mathrm{R}^{m \times n}, y \in \mathrm{R}^{m}$ is the input clinical data and output target value respectively, $Pipeline$ is the candidate machine learning pipeline generated by GP, *FPR* is the conservative requirement of *FPR* level, and the *Sensitivity* is defined as

$$Sensitivity = \frac{TP}{TP + FN}.$$

In this paper, we refer to this sensitivity as the pipeline performance.

The *FPR* and *Sensitivity* of a given pipeline are determined not only by that pipeline's performance (i.e., the model probability estimate) but also the threshold value used to dichotomize a continuous (0-1) patient-level outcome probability into a binary treatment recommendation (continue life-sustaining therapies or withdraw life-sustaining therapies). The optimal threshold that maximizes *Sensitivity* while satisfying the target *FPR* is likely to vary for different pipelines. Therefore, in GP training, we record the two relevant values (*Sensitivity* and *threshold*) computed based on the training data set. *Sensitivity* is then used for pipeline selection, while the pipeline-specific *threshold* is carried forward with the final selected models to evaluate performance in the test data set.

The approach described so far addresses *model and hyperparameter selection*. Next, we consider other challenges in these complex time series data. Specifically, we customize the ML pipeline by adding two steps for data preparation that are jointly optimized with other model parameters across subsequent generations: *summarization of clinical records* and *data aggregation*. For each patient, k clinical records are recorded, covering m timestamps. Often, individual

records are closely related (for example, simultaneously sampled FFT frequency spectra of two anatomically adjacent electrodes). Many ML classifiers (e.g., Logistic Regression) cannot handle time-series data ($\in \mathrm{R}^{n \times k \times m}$, where n is the number of patients). One common way to solve this is fusing clinical records into a single record ($\in \mathrm{R}^{n \times m}$) with a user-defined operator (e.g., min, max) and treating each timestamp as a single feature without preserving the temporal relationship. The functionality of these two stages are as follows:

- **Clinical records summarization**: Select a subset of clinical records (i.e., related qEEG features) and fuse by an operator, such as sum, mean, or max, etc.
  $Hyperparameter\ set = \{\lambda_{records\_set},\ \lambda_{combine\_method}\}$

- **Data aggregation**: Generate the aggregation, such as sum, mean, or max, etc., of time series data over sequential time interval of various lengths. We name the time interval as aggregation level.
  $Hyperparameter\ set = \{\lambda_{aggregation\_level},\ \lambda_{aggregation\_method}\}$

Incorporating these steps in clinical record selection and data preparation, the set of all candidate pipelines can be represented as $\mathcal{A} = \{\mathbf{A}_{Selection} \cup \mathbf{A}_{Aggregation} \cup \mathbf{A}_{Model}\}$. In plain words, we first fuse a subset of qEEG clinical records as a single time series and then generate the aggregation of this new time series as features for an ML classifier. The hyperparameters in these two stages are tuned by GP automatically without expert engagement. We will show the detailed hyperparameters selection range in Table 7.

As discussed previously, SMARTPROGNOSIS is built on top of the BrainFlux data warehouse [43], which preserves all clinical records at different aggregation levels. Thus, during the data aggregation process, we can easily import the corresponding aggregated data from BrainFlux instead of directly applying the data aggregation to reduce the computational cost. Although aggregation dramatically reduces execution time, it has the potential to reduce classification performance due to information loss. It is noteworthy that different methods have different susceptibility to information loss from data aggregation. For example, for some classifiers, performance fluctuates slowly with increasing aggregation levels but drops dramatically for others. Therefore, we apply GP to select the appropriate combination of aggregation level and classifier.

4.3.1.3 Ensemble Learning for Prediction

In this section, we introduce the method of generating predictions for new patients. In general, model performance in test data is lower than that demonstrated in training. In other words, even if we constrain $FPR <= 0.01$ in the GP training process, we cannot guarantee comparable performance at a given classification threshold in the test set. Based on our experimental results, on average, the *FPR* is around 0.1 for the test data when the training requirement is $FPR <= 0.01$, which makes the model non-usable. In order to reduce this misclassification error, previous works suggest to re-learning the threshold based on the test data. This approach is obviously data-sensitive and cannot be used prospectively. Instead, we use ensemble learning [34] to improve performance in the test set by combining output from multiple different pipelines that were optimized using GP. The number of the combined pipelines is set empirically. In addition, to ensure adequate performance in the test set, we train models at an excessively conservative *FPR*, such as an extreme condition $FPR <= 0$. We report results from both these two approaches, using SMARTPROGNOSIS$_{simple}$ when we adjust the threshold based on the test data with the best-performed pipeline, and SMARTPROGNOSIS$_{ensemble}$ to represent the ensemble method.

As Buza et al. [18] mentioned, closely-related classification models often make similar errors when trained on the same data set, which can limit the usefulness of ensemble learning. For instance, two logistic regression models with slightly different hyperparameters tend to misclassify the same patients in a given data set. Therefore, to support ensemble learning, we modify the GP framework by recording the top l performance candidate pipelines among all generations, including the structure, *Sensitivity* score, and *threshold*, instead of just outputting the top one. Particularly, these l candidates share no common model structure (i.e., none of these candidates have the same prediction model, aggregation level, and clinical recordsets) to ensure the diversity of results. Recording multiple pipelines also facilitates model transferability. We can easily and quickly fine-tune the GP process with new data by initializing the first generation with the recorded pipelines.

In the testing process, the new patient is predicted by all l pipelines with their recorded thresholds. Only if all pipelines agree that the patient is extremely likely to have a poor outcome would the medical provider receive this prognostic estimate from the decision support system.

We show the pseudo-code of SMARTPROGNOSIS in Algorithm 3.

Algorithm 3 SMARTPROGNOSIS

Input: $X = \{X_{train}, X_{test}\}, \vec{y}_{train}, \mathcal{A} = \{\mathbf{A}_{Selection} \cup \mathbf{A}_{Aggregation} \cup \mathbf{A}_{Model}\}$

Output: $\hat{\vec{y}}_{test}$

function SMARTPROGNOSIS$_{train}$($X_{train}, \vec{y}_{train}, \mathcal{A}$)
    Initialize ▷ randomly select m pipelines from $\mathcal{A}$
    while $i < n$ **do**
        Evaluate pipelines on X_{train}. ▷ cost function in Eq. 14
        Rank pipelines by performance.
        Generate a new list of pipelines using high performed pipelines. ▷ crossover, mutation and copy
    Record the top l performance candidate pipelines $\mathcal{A}_l$.

function SMARTPROGNOSIS$_{test}$(X_{test}, $\mathcal{A}_l$)
    for $P \in \mathcal{A}_l$ **do**
        $\hat{\vec{y}}_{test_p} = P(X_{test})$
    Ensemble $\hat{\vec{y}}_{test_p}$ where $P \in \mathcal{A}_l$

4.4 EXPERIMENTS

In this section, we present the experimental study of SMARTPROGNOSIS on real data and provide a comprehensive analysis of the results.

4.4.1 Experimental Setup

4.4.1.1 Dataset

We test SMARTPROGNOSIS on 1039 patients who underwent post-arrest EEG monitoring. These patients were admitted to a single academic medical center from May 2010 to March 2018. We excluded patients who awakened within 6 hours of arrival, patients with advanced directives inconsistent with ICU care, patients with < 6 hours of EEG, and patients admitted > 12 hours post-arrest. For each patient, we summarized ten qEEG characteristics at every 5 minutes for total 48 hours, including the amplitude-integrated EEG (aEEG), peak-to-peak based measure of EEG amplitude (Peak Envelope), FFT (Fast Fourier Transformation) Spectrogram, Rhythmicity Spectrogram, and burst suppression ratio (SR) on both left and right hemisphere. Each hemisphere record is an average of all 11 channels on that hemisphere. We normalized the data to range 0 to 1. Besides the qEEG records, we also included clinical characteristics including age and Pittsburgh Cardiac Arrest Category (PCAC), a four-level ordinal measure of early post-arrest illness severity [30].

We show the clinical characteristics of these patients in Table 6. Only 261 patients (around $1/4$) awakened from a coma while in the hospital (our primary outcome of interest), which makes the dataset imbalanced. Consequently, training the model on this data would create a bias toward the majority class. Therefore, we applied the Synthetic Minority Over-sampling Technique (SMOTE) [22] to create synthetic samples of the underrepresented class to balance the training set. SMOTE generates synthetic cases for the minority class along the line segments joining any/all of the k nearest neighbors in this class. In this paper, we empirically choose $k = 5$.

4.4.1.2 Training and Test setting

We outline the general setting for GP in Table 7. In the experiments, we evaluate the performance of SMARTPROGNOSIS using 5-fold cross-validation to avoid over-fitting. All the following results are computed by averaging over 5-folds. During the GP training, we further split the data into training (80%) and validation set (20%). The ranking of pipelines is based on the performance of the validation set. We report performance of prediction results via three criteria: Area Under the

Table 6: Clinical characteristics of patients

Characteristic / Outcome	Count	Age	Gender		PCAC			
			Male	Female	II	III	IV	Unknown
Did not recover	778	58± 16	470	303	112	80	516	65
Recovered	261	55 ± 16	170	90	152	43	33	32

Curve (AUC), *Sensitivity* at $FPR \leq 0.01$ and *Sensitivity* at $FPR \leq 0.05$.

4.4.2 Experimental Results

In this section, we evaluate the performance of SMARTPROGNOSIS and compare it with common alternative approaches. Specifically, we consider as alternatives logistic regression without regularization, random forest, penalized logistic regression [6] and random forest with LASSO [95] (see details in Section 4.1.1). We also consider general time-series classification methods, including Fully Convolutional Networks (FCN) [137] and PROCESS [17], which classifies data based on the dynamic time warping (DTW) distance. Results from the pooled test-set data are shown in Table 8. SMARTPROGNOSIS outperforms the baseline methods on all evaluation metrics.

For all baseline methods and SMARTPROGNOSIS$_{simple}$, the threshold that satisfies the *FPR* requirement is selected based on the test set. SMARTPROGNOSIS$_{ensemble}$ bypasses this limitation and achieves higher *Sensitivity* at a low *FPR* level.

4.4.3 Discussion

In this section, we provide analysis in terms of 1) the pipeline hyperparameter selection, including data preparation via summarization of clinic records selection and data aggregation, 2) the number of pipelines in ensemble learning, 3) false-positive cases, 4) limitations, *and* 5) future works.

Table 7: Parameterization

Parameter	Value
Number of generations / individuals	50 / 50
Cost function	$\mathcal{C} = Sensitivity(Pipeline, X, y, FPR)$ (Eq.14)
FPR training requirement	0
Clinical recording selection	*Records set*: all subsets of the 10 qEEG records *Combine method:* min, max, avg, sum
Data aggregation	*Aggregation level*: 1, 3, 6, 12, 30, 60, 120, 300 (level 1 represents 5 mins aggregation) *Combine method:* min, max, avg, sum
Classifier	All available classifiers from Python Scikit-learn [103]

4.4.3.1 Pipeline hyperparameter

As mentioned before, appropriately summarizing and aggregating the clinical records could significantly reduce the computational cost with minimal effect on performance. In this case, the frequency with which a particular clinical record appears in candidate pipelines indirectly reflects record importance and the extent to which the clinical record can be summarized across time with less information loss. Thus, considering the frequency of occurrence for selected clinical records, aggregation levels, and machine learning classifiers among all pipelines assembled for $FPR \leq 0.01$ is clinically interesting.

- *Clinic records:* SR is chosen by almost all pipelines, meaning it is highly correlated with the outcome after cardiac arrest. This observation is consistent with current research [41, 122, 38], which domesticates that the SR has high predictive power.
- *Aggregation level:* Among the aggregation levels, the top three are aggregated by 150 mins, 5 mins (finest level), and 15 mins. That means highly aggregated data can be used to improve efficiency without sacrificing performance. For example, a single model requires 13.5s to

Table 8: Experimental results

Model	Feature	AUC	Sensitivity $FPR \leq 0.01$	Sensitivity $FPR \leq 0.05$
Logistic Regression (L_2 penalty, $\lambda = 1$)	Age, PCAC, average of **all** qEEGs	84.71%	30.29%	52.31 %
Random Forest ($number\ of\ trees = 100$, $criterion = gini$)		84.16%	8.12%	50.74 %
Penalized Logistic Regression [6] (L_1 penalty, $\lambda = 1$)	Age, PCAC, average of **selected** qEEGs	88.11%	35.15%	59.96 %
Random Forest with LASSO [95]		88.96%	11.44%	60.95%
FCN [137] (three FC layers with dimension 128-256-128)	Age, PCAC, average of **all** qEEGs	78.73%	13.89%	33.79%
PROCESS [17]		70.09%	7.86%	24.67%
SMARTPROGNOSIS$_{simple}$ ($Aggregation\ level = 30min$, $classifier = ExtraTreesClassifier$)	Age, PCAC, average of **selected** qEEGs	89.68%	49.03%	68.18%
SMARTPROGNOSIS$_{ensemble}$	Age, PCAC, average of **selected** qEEGs	**90.64%**	**59.74%**	**69.04%**

converge using 5 min data v.s. 1.8s for 150 mins data.

- *Classifier:* Random Forest Classifier and Extra-trees Classifier are the top two selected classifiers with even distribution. Essentially, these two models have a common structure, a bagged decision tree, which can provide a better result with less likely over-fitting. However, that also makes these models less interpretable. Our framework's target is to identify patients with a near-zero probability of recovery. Therefore, performance is most important in our case.

4.4.3.2 Ensemble Learning

In this part, we discuss the effect of the number of combined pipelines in ensemble learning.

As demonstrated in Figure 2, assembling more pipelines decreases to reduce not only *FPR* but also *Sensitivity*. In Figure 25, we show the relationship between the number of assembled pipelines and the *FPR* and *Sensitivity* on test samples. The x-axis represents the number of pipelines, and the y-axis shows the corresponding *FPR* (left, red) and *Sensitivity* (right, blue). The pipelines

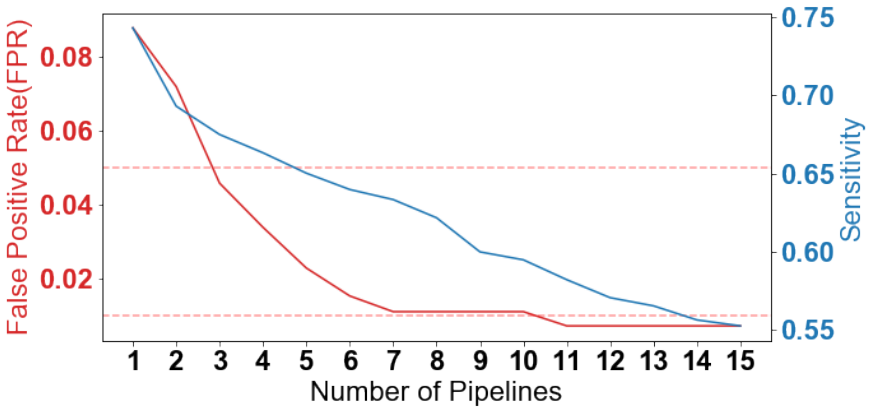


Figure 25: The impact of number of pipelines in ensemble learning.

are added by performance order. In this plot, we train the GP process with *FPR* requirement ≤ 0 (Eq.14), which is much lower than the practical requirement. The actual target *FPRs* (0.01 and 0.05) are highlighted with red dash horizontal lines. Both *FPR* and *Sensitivity* have a negative relationship with the number of assembled pipelines. Based on these results, we chose an ensemble of 3 pipelines for $FPR <= 0.05$ and 11 for $FPR <= 0.01$.

4.4.3.3 Misclassification

Although we achieved better performance at a low FPR level compared to baseline methods, we found two interesting cases that were misclassified as the poor outcome by all the pipelines even when we pushed the FPR training requirement down to ≤ 0. We explored the clinical details of these repeatedly misclassified patients in detail. In both cases, there were readily apparent confounders that affected the qEEG data. Specifically, extremely high doses of sedative medications or general anesthetics were administered to these patients as part of clinical care. These medications cause generalized EEG similar to that observed after severe or irrecoverable brain injury.

These cases highlight two important aspects of future modeling applications. 1) First, in most of the current work, only qEEGs and simple clinical characteristics are included as predictors. A strong but infrequently observed confounder may mislead the model. One way to deal with it is by using the multi-view data analysis [99], which incorporates data from multiple sources. In our case,

we could integrate other clinical data, such as vital signs, medications, etc. 2) Second, prognostic estimates from any predictive algorithm must be interpreted by a clinical provider with access to the full context of available measurable and immeasurable (e.g., patients' values and preferences) data.

4.4.3.4 Limitations

Besides the specific situations discussed above, we found SMARTPROGNOSIS has some limitations which could restrict its performance. First, as currently implemented, inputs must be clean and predictably structured prior to storage and aggregation. This limitation is conceptually easy to overcome by integrating various pre-processing steps into our pipeline. Second, SMARTPROGNOSIS performs best when model performance can be improved by incorporating additional incremental information gleaned from diverse modeling strategies. In the case of strong collinearity between model predictions, it may be more computationally intensive without much improvement in predictive performance. Finally, although our data are time series, we do not allow the best model ensembling to vary over time as currently implemented. Recent work has demonstrated the feasibility and potential benefits of this approach [84]. Despite these limitations, as noted above, SMARTPROGNOSIS implements an efficient methodological approach that addresses several modern medicine problems, coupling data storage, data aggregation, feature selection, model selection, and hyperparameter tuning.

4.4.4 Future works

In this part, we provide some potential ideas to enhance the performance of SMARTPROGNOSIS in the future.

- As we discussed above, aggregated data can be used to achieve a competitive result while reducing the computational cost. In order to encourage the framework to pick up more efficient pipelines in the future, we can involve the consumed time as a factor in the cost function (Eq. 14).
- Currently, SMARTPROGNOSIS only considers simple classification methods, such as logistic regression and random forest, as they are the most frequently used methods in the medical

domain. However, these methods do not consider the time dependency inside each feature, which may contain crucial information. We plan to include more complex models, such as LSTM and Attention to discover the time information in the data.

- The current ensemble learning assumes that all l candidate pipelines share equal weight, while in practice, some pipelines should play more vital roles. Therefore, assigning different weights, such as the normalized cost function value, to each pipeline may benefit the model performance.

4.5 CONCLUSION

In conclusion, we propose an end-to-end automatic ensemble classification framework, SMARTPROGNOSIS. In our specific test case, we used this tool to predict poor neurological outcomes in comatose patients resuscitated from cardiac arrest. SMARTPROGNOSIS automatically generates and assembles candidate machine learning pipelines which are selected by maximizing the *Sensitivity* at a very low *FPR* level. We evaluate SMARTPROGNOSIS on real patient data and demonstrate that it over-performs conventional methods on all evaluation criteria. We also provide suggestions for future works based on our experimental results. Beyond our test case, this approach has broad relevance to analyzing other complex large datasets amenable to ML analysis, such as images, cardiorespiratory waveforms, or even more conventional electronic health record data.

5.0 DATA SUMMARIZATION: TARGET-SENSITIVE DATA SUMMARIZATION FOR TIME SERIES DATA

In the previous chapter, the automatic selection framework experiments show that aggregated data could provide enough information to get a comparable analytic result with less computational resources and storage. However, it is noteworthy that most aggregation methods apply only simple mathematics operators, such as sum, average, and min, etc., which would result in information loss, especially when the aggregation level increases (i.e., lossy aggregated data). For example, annual counts of Walmart data (see Figure 14) are hard to capture the detailed trade, such as the spikes during the end year. Predicting the peak season with the yearly counts of data or even reconstructed data may lead to an inaccurate result. Therefore, in this chapter, we propose a novel summarization method to aggregate data, I-AGG, which emphasizes the critical information of the original data by applying different aggregation frequencies over different segments. The idea is that preserving more details in the essential parts while reducing the particulars at other parts can improve the summarization quality and thus lead to better analysis results. For example, aggregating the Walmart data weekly at the end of the year and monthly otherwise reveals the spike at the Christmas week. In detail, I-AGG first learns an element-wised important score for the entire input data, which indicates the importance of each time ticks, and then decides the aggregation granularity based on the score and segment length. The score is learned using a Variational Autoencoder- Generative Adversarial Network (VAE-GAN) framework [82], which enforces the weighted time series (i.e., multiplying the score with the time series element-wisely), preserving enough information to reconstruct the original one.

We evaluate the effectiveness of the proposed method on different time-series datasets from UEA multivariate time series classification archive [8] and access the performance via three aspects – the ability of analysis, reconstruction, and imputation. Our summarization result overperforms all the baseline methods. Detailed analysis and discussion are provided in Section 5.3.

5.1 BACKGROUND

In this section, we provide a background of the techniques used in the proposed method, including Generative Adversarial Networks (GANs) and Variational AutoEncoders (VAEs).

5.1.1 Generative Adversarial Network (GAN)

Generative Adversarial Networks (GANs) was first proposed by Goodfellow, et al. [51] and has been applied successfully in different domains, such as image super-resolution [83], image generation [32], image-to-image translation [66], and text-to-image translation [110]. GANs are composed of two components: Generator (G) and Discriminator (D). The generator is used to generate synthetic data from a random noise vector, which follows Gaussian distribution (in most cases). The discriminator is designed to distinguish between synthetic data and real data. By competing with each other, the generator finally produces synthetic data that follows a similar distribution compared to the real data and can "deceive" the discriminator. The general architecture of GANs is shown in Figure 26, and it has been illustrated under an application of face generation. As shown in the figure, a synthetic image, which has never appeared in the original dataset, is generated from random noise and is hard to distinguish by a human. Mathematically, given a random input $\vec{z}$ with distribution $\vec{z} \sim p_z$, the target of GANs is to train the probability distribution (p_g) of the generator's output ($G(\vec{z})$) to approximate the real data distribution p_r. Optimization of a GAN is performed with respect to a joint loss function for D and G [136]

$$\min_G \max_D \mathbb{E}_{\vec{x}\sim p_r} \log[D(\vec{x})] + \mathbb{E}_{\vec{z}\sim p_z} \log[1 - D(G(\vec{x}))] \tag{15}$$

where $D(\vec{x})$ is a binary classification, the expected output of $D(\vec{x})$ is 1 for real data and $D(G(\vec{x}))$ is 0 for synthetic data.

5.1.2 Variational AutoEncoder (VAE)

Other than the GANs introduced above, another set of deep generative models, named Variational AutoEncoders (VAEs) [75] has also shown a strong ability to produce synthetic data. VAEs are a member of AutoEncoders (AEs) [50], which aims to encode input data ($\vec{x}$) to a latent space

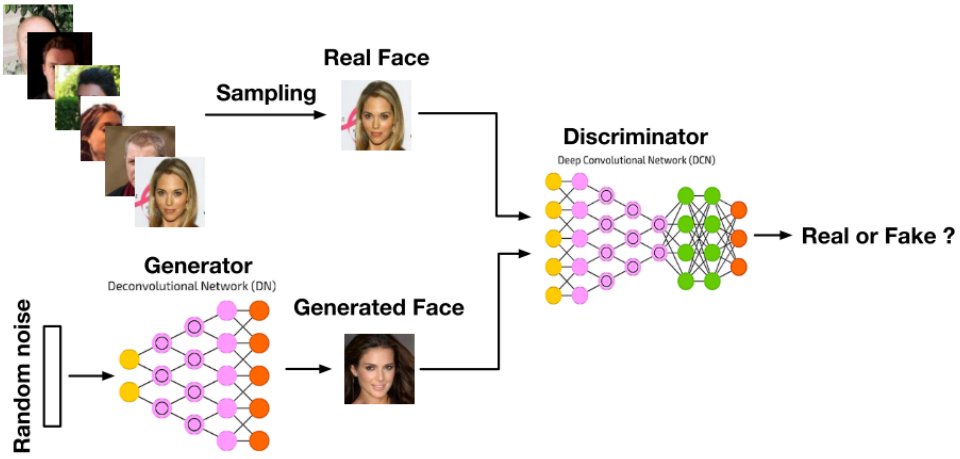


Figure 26: Architecture of GANs [136]

(i.e., representation, $\vec{z}$) and then decode the representation back to the original space. Considering the limitation of AEs, such as over-fitting, VAEs include regularity in the latent space. That also ensures that the latent space has good properties allowing us to generate new unseen data. Formally, encoder encodes input data ($\vec{x} \sim P(\vec{x})$) to a latent distribution $P(\vec{z}|\vec{x})$ and then decode a sample from $P(\vec{z}|\vec{x})$ back to the input space, the new generated data follows a distribution $P(\vec{x}|\vec{z})$. The loss function is as follows.

$$E(log(P(\vec{x}|\vec{z})) - D_{KL}(Q(\vec{z}|\vec{x})||P(\vec{z})) \tag{16}$$

where $Q(\vec{z}|\vec{x})$ is a distribution that is inferred from $P(\vec{z}|\vec{x})$ using Variational Inference (VI) method [61]. D_{KL} is the Kullback–Leibler divergence [81]. The first term is the reconstruction loss, which forces the decode output close to the input data. The second term is the regularization loss to mitigate the information loss when using the VI to represent $P(\vec{z}|\vec{x})$.

5.1.3 Hybrid of GANs and VAEs

Despite that GANs and VAEs both belong to the deep generative models, they are suitable for different tasks. Technically, GANs perform better in term of new data generation, while VAEs is more useful in compressing data. The reason is the strategy of VAEs is to generate new data

from a hidden representation learned from the original data, and the representation has to follow a predefined distribution (i.e., single Gaussian distribution, for most cases). However, given multi-modal distributed input data, the hidden representation is hard to capture all the details and thus results in low-quality generated data with blurry details [19].

In order to tackle this blurry issue, Larsen et al. [82] proposed to measure the similarity of VAEs' decoder output with the original data using a higher-level and sufficiently invariant representation of the data instead of element-wise similarity (i.e., squared error). Specifically, they combine the VAEs' decoder and the GANs' generator into one unit and train them jointly. Figure 27 shows the structure of the proposed model VAE/GAN. It first learns synthetic data via VAE generator and then distinguishes it with the real data using GAN discriminator.

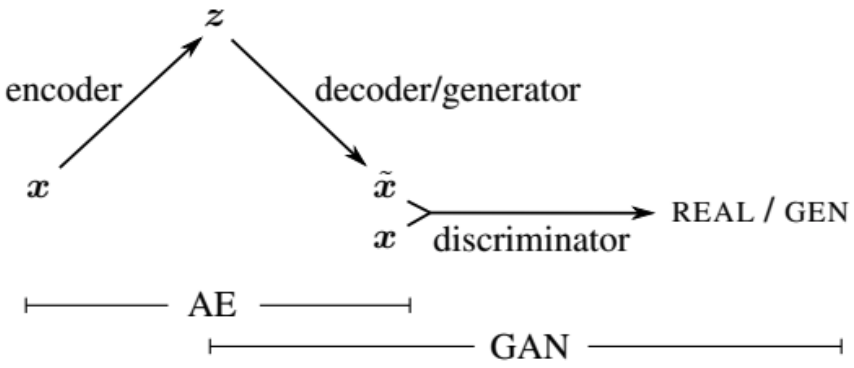


Figure 27: Structure of VAE/GAN [82]

Following this direction, more and more researchers have explored the hybridization of VAEs with GANs and shown the effectiveness of this new structure [11, 93, 94]. They claim that the hybrid of VAEs and GANs would improve the stability and diversity of GANs as well as the sample quality of VAEs [115]. Until now, these models have been applied in different domains. One application is the video summarization, which represents a video by a few video frames [92, 145]. This is similar to the problem we tackle in this chapter – sequential data summarization. However, unlike video, which is an image time series, individual time ticks or small segments of a historical (numerical) time series data cannot convey enough information without context. Therefore we proposed a new summarization method for historical time series, I-AGG, following the existing video

summarization framework SUM-GAN [92]. Instead of representing the sequence with a subset of video frames (i.e., time-ticks) with high important score, I-AGG aggregates time-ticks within different segments using different frequencies determined by the important score and segment length (i.e., higher granularity on critical segments).

5.2 Time Series Data Summarization Framework

In this section, we introduce the architecture of I-AGG in detail. We begin by presenting the score generation, which identifies the crucial data points in the input time series. Then we explain how to integrate the score information for information-sensitive summarization generation.

The input time series for our problem is a multi-variable time series $\mathbf{X}^{org} \in \mathbb{R}^{n \times k \times m}$, where n, k, m are the number of instances, the number of features, and the length of timestamps, respectively. We denote the ith time series as $\mathbf{X}_i \in \mathbb{R}^{k \times m}$.

5.2.1 Score generation

As mentioned above, the SUM-GAN structure has been adopted to learn the critical score for time ticks. It mainly consists of two parts: 1) a Bidirectional Long Short-Term Memory (Bi-LSTM) [64] *Selector* which learns the importance score of each time-ticks and then generates a summarized time series by emphasizing those critical time ticks (i.e., weighted time series by scores), and 2) a VAE-GAN *Evaluator* [82] which is used to evaluate the effectiveness and compactness of learned score by enforcing the reconstructed data from summarized data ($\mathbf{X}^{sum}$) close to the real data ($\mathbf{X}^{org}$). The overall objective of the proposed architecture is to maximize information preserved in the summarized data. Figure 28 shows the proposed architecture for time series data summarization in detail.

At the beginning, a Bi-LSTM network (**selector**) takes the input time series data $\mathbf{X}^{org}$ and learns a same length score vector ($\vec{s} \in \mathbb{R}^m$). We choose the Bi-LSTM network in the selector because it can take both the past and future data into account and provide a better understanding of the data. The value of $\vec{s}$ is normalized between $[0, 1]$ with a sigmoid layer after the Bi-LSTM,

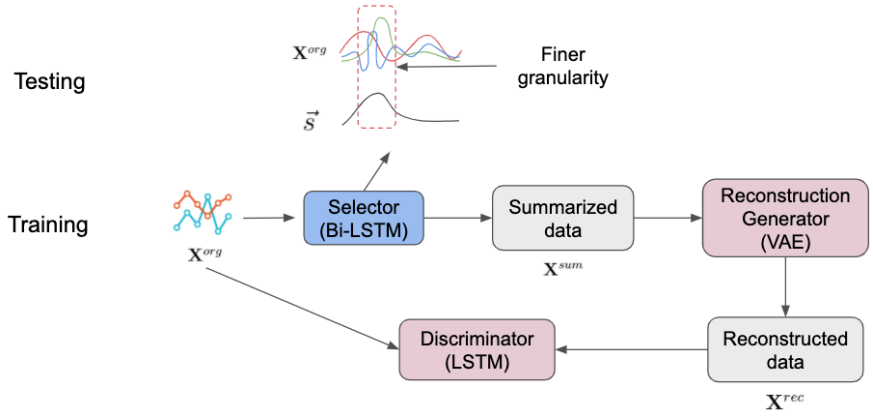


Figure 28: Architecture of proposed method for data aggregation.

and a higher score means more important. Then we multiply the score vector with the input vector ($\mathbf{X}^{org} \odot \vec{s}$, $\odot$ is the element-wise multiplication) to get a summarized data ($\mathbf{X}^{sum}$). Ideally, the summarized data should emphasize more on the informative data points. However, another extreme case may happen when the selector assigns equal weights to all the time ticks and results in worthless summarized data. To avoid preserving redundant information in the summarized data, we design a sparsity loss function to restrict the score vector with a regularization factor p.

$$L_{Sparsity} = \|\frac{1}{m}\sum_{t=1}^{m}\vec{s}_t - p\|_2 \tag{17}$$

where p is a predefined summarized ratio. This would penalize the score vector when a large number of time-ticks has a higher score.

Next, we assess the quality of the summarized data ($\mathbf{X}^{sum}$) through a VAE-GAN evaluator in terms of the ability to restore the original data. It first maps the $\mathbf{X}^{sum}$ back to the original time series data through a VAE and then differentiates it with the real original data. Specifically, $\mathbf{X}^{sum}$ is encoded by an LTSM layer (VAE$_E$) into a low dimensional hidden space and then decoded back to the original space through another LSTM layer (VAE$_D$). The expectation is that the summarized data carries enough information that can be captured by the latent vector to reconstruct the original data. After generating the recovered time-series data ($\mathbf{X}^{rec}$), an LSTM discriminator (D) tries to distinguish it with the real input data ($\mathbf{X}^{org}$). In a word, the VAE-GAN evaluator aims to train

a reconstruction generator to recover the original data from the summarized data and ensure the reconstructed one follows the same distribution as the real data.

Similar to VAE-GAN, we learn this score generator by alternatively optimizing the generator with the discriminator. The parameters are updated based on the following steps:

- $\theta_{Selector}, \theta_{VAE_E} \longleftarrow - \bigtriangledown_{\theta_{Selector}, \theta_{VAE_E}} (\mathcal{L}_{Sparsity} + \mathcal{L}_{Prior} + \mathcal{L}_{Reconstruct})$
- $\theta_{VAE_D} \longleftarrow - \bigtriangledown_{\theta_{VAE_D}} (\mathcal{L}_{Reconstruct} + \mathcal{L}_{GAN_{sum'}})$
- $\theta_D \longleftarrow - \bigtriangledown_{\theta_D} (\mathcal{L}_{GAN_{sum}} + \mathcal{L}_{GAN_{org}})$

where $\theta_{Selector}, \theta_{VAE_E}, \theta_{VAE_D}$ and θ_D are the parameters of Selector, VAE encoder, VAE decoder and Discriminator, respectively. $\mathcal{L}_{Sparsity}$ and is defined in Eq.17, which penalizes big number of important time ticks. $\mathcal{L}_{Prior}$ is the general VAE loss which is the same as the definition in Eq 16. $\mathcal{L}_{GAN_{sum}}$ and $\mathcal{L}_{GAN_{org}}$ target to maximize the capability of the discriminator to differentiate the reconstructed time series ($\mathbf{X}^{rec}$) with the original time series ($\mathbf{X}^{org}$). In detail, it forces the discriminator to classify the $\mathbf{X}^{rec}$ as synthetic data, which has probability zero, while optimizing the classification probability of $\mathbf{X}^{org}$ close to one. On the contrary, the $\mathcal{L}_{GAN_{sum'}}$ enhances the VAE generator output ($\mathbf{X}^{rec}$) to fool the discriminator (i.e., the probability is close to 1). We train this loss by minimizing the Mean Square Error (MSE) of the discriminator output and the label (i.e., 1 is real and 0 is synthetic). Mathematically, the equation is defined as follows.

$$\begin{aligned}
\mathcal{L}_{GAN_{sum'}} &= \|D(VAE(\mathbf{X}^{sum}))\|^2 \\
\mathcal{L}_{GAN_{sum}} &= \|1 - D(VAE(\mathbf{X}^{sum}))\|^2 \\
\mathcal{L}_{GAN_{org}} &= \|1 - D(\mathbf{X}^{org})\|^2
\end{aligned}$$

Here we define the loss function with MSE loss instead of Binary Cross Entropy (BCE) loss because the authors in [7] demonstrate that the MSE loss is more suitable for training the VAE to reconstruct the input.

$\mathcal{L}_{Reconstruct}$ is to minimize the distance between the reconstructed time series ($\mathbf{X}^{rec}$) and the summarized time series ($\mathbf{X}^{sum}$). The standard VEA method uses Euclidean distance to measure the element-wise distance. However, Larsen, Anders Boesen Lindbo, et al. [82] claims that this error is not adequate for image and other signals with invariance. They propose to replace the element-wise error with a feature-wise metric expressed in the discriminator. The detailed $\mathcal{L}_{Reconstruct}$ is

defined as

$$\|\vec{h}^{rec} - \vec{h}^{org}\|^2,$$

where is $\vec{h}$ is the output of the last hidden layer of discriminator.

5.2.2 Aggregation generation

Now for any new time series, we can obtain an important score vector through the selector layer. However, as we mentioned above, unlike the video time series, a single time tick or a short period of a numerical time series does not contain enough information to describe the whole data. For example, a large number of sales of Apple Inc. stock in June doesn't indicate it will keep the same level in July. Therefore, instead of selecting a subset of time ticks or several segments using the importance score as the summarized data, in I-AGG, we propose to aggregate the whole time series under different resolutions, i.e., higher resolution in a more important part. We first break the multivariate time series into several segments by the Greedy Gaussian Segmentation (GGS) method [58]. The basic idea of GGS is to make sure the data in each segment can be well explained as independent samples from a Gaussian distribution. Empirically, unless otherwise stated, the number of segments is predefined to 15 for most of the datasets in the experiment. The segments' important score is then computed by averaging the score over all time ticks within the segment. Considering that the segments may have different lengths, we also take it into consideration for the resolution determination. The basic rule is that the segments with more significant information or longer interval need more data to represent. The detail aggregation level for each segment (AL_k) is calculated as follow:

- *Step 1 (Information score of segment k)*:

$$\vec{I_k} = \frac{\vec{s_k} \odot \vec{nl}_k}{\vec{s_k} + \vec{nl}_k}$$

  Where $\vec{s}_k$, $\vec{nl}_k$, and $\vec{l}_k$ are the importance score, normalized length, and actual length of the kth segment, respectively. The informative score ($\vec{I_k}$) is computed by averaging the segment's important score and the normalized length ($\vec{nl}_k$) to represent the amount of information contained in each segment.

- *Step 2 (Summarization length of segment k)*:

$$\vec{SL}_k = \vec{nI}_k \times m \times r$$

Where m is the total length of the time series data, and r is a predefined ratio indicating the portion of the data to be preserved in the summarized data. We assign a corresponding ratio of summarization length ($\vec{SL}_k$) to segments based on their normalized information score ($\vec{nI}_k$).

- *Step 3 (Aggregation level of segment k)*:

$$\vec{AL}_k = ceil(\frac{\vec{l}_k}{\vec{SL}_k})$$

The final aggregation level ($\vec{AL}_k$) equals the actual length divided by the assigned summarization length.

Figure 29 shows an example of the different aggregation levels over segments. The red dashed vertical lines split the segmentation, and the yellow vertical lines represent the aggregation resolution. Specifically, we generate the summarized data by aggregating time ticks between every two yellow lines with an operator (i.e., sum, mean, etc., we use mean in the proposed algorithm.). As observed, two parts with higher variation are aggregated with finer granularity to preserve more detailed information, while the flat part at the end is summarized with coarser resolution. Conceptually, it is similar to Discrete Wavelet Transform (DWT) which enforces the heterogeneous signal in the time series. However, our method is more flexible in discovering and preserving these signals. We will show more examples and comparisons in Section 5.3.

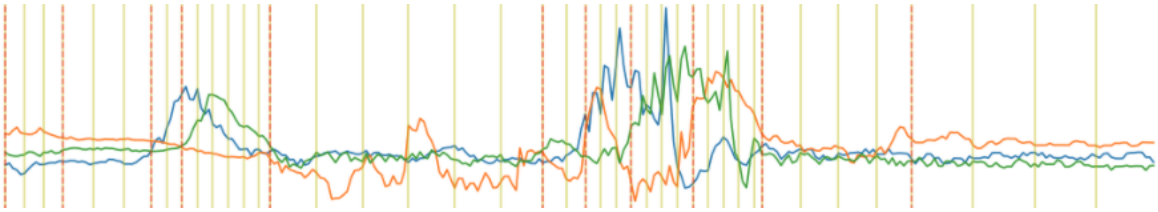


Figure 29: Example of the aggregation frequency over different segments.

5.3 Experiments

In this section, we evaluate the performance of the proposed summarization method from different aspects. We first introduce the experimental setting, including datasets, evaluation metrics, and baseline methods. Then we evaluate the effectiveness of the proposed method by comparing the performance with the baseline methods on different metrics. After that, comprehensive analysis in terms of the interpretability and a practitioners' guide regarding the applicability are elaborated.

5.3.1 Experimental Setting

- **Dataset**: We evaluate our proposed method on datasets from UEA multivariate time series classification archive [8], which include 30 datasets with a wide range of applications, dimensions, and time-series length. Since the method aims to summarize the sequence data over the time dimension, datasets with 3-dimensions and lengths larger or equal to 300 are selected. For all datasets, we truncate the original time series to retain only 15% of the data (i.e., $r = 0.15$) and assess the information preserved in the shorted sequence.
- **Evaluation metrics** Since the time series data summarization problem does not have ground truth, we propose to assess the performance in terms of three aspects.

  The first one is *analysis (i.e., classification) capability*, which can be evaluated by the classification accuracy. In the experiment, different classification approaches have been applied to the original data, such as 1NN-DTW [125], FCN [137]. We choose the one with the best performance to evaluate the summarized data.

  The second one is *summarization quality*. Although most summarization methods are lossy aggregation, a good summarization should have the ability to approximate the original data by preserving enough information. We evaluate this quality by measuring the distance between the original and the approximated time series (i.e., RMSE). The approximated time series is reconstructed from the summarized data using H-FUSE-S [87], which assumes the data is smooth. Other reconstruction methods, such as ARES [143], HOMERUN [5], and TURBOLIFT [142], are also applicable.

  The last one is *Imputation capability*. In practice, due to unexpected accidents, such as

equipment damage or communication error, it is common that some values of the time series data are missing and thus result in worse performance in the downstream applications. In general, aggregating or downsampling would ignore the missing parts in the time series by considering only the existing part and providing an approximated summarization. That may make the summarized data miss some critical information. Therefore, we consider the imputation capability as one criterion to assess the *summarization quality*. We evaluate this property by reconstructing the summarized data with missing values in the original data. The RMSE is measured on both existing and missing time ticks. We create the missing data by randomly removing 10%, 20%, 30%, and 40% of the data points in the dataset HeartBeat and generate summarization based on these incompleted data. To specially handle the missing information, we slightly modify the proposed framework. In detail, we follow the strategy in GAIN [144] to replace the missing value with random noise and change the reconstruction loss to measure only the distance between existing values. The new reconstruction loss is $\mathcal{L}_{Reconstruction} = \|\mathbf{X}^{rec} \odot M - \mathbf{X}^{org} \odot M\|^2$, where M is a binary mask matrix that represents the position of existing values (i.e., '1' means existing and '0' means missing.). The intuitive idea is that the VAE generator can synthesize the missing part using the noise data and deceive the discriminator.

- We compare I-AGG method with two baseline methods, Piecewise Aggregate Approximation (PAA) and Discrete Wavelet Transform (DWT), which are frequently used for the representation of time series data. A detailed description of these two baseline methods is provided in section 2.3.2. In order to preserve a similar length summarization (i.e., 15% of data), we generate the PAA aggregation by averaging every seven time-ticks. In DWT, we decompose the time series with ten levels Haar wavelet [26] and use the first eight levels of coefficients to represent the original data. For the missing data experiments, since DWT cannot handle missing values, we interpolate those time-ticks with linear regression and then do the aggregation.

5.3.2 Experimental result

The performance of I-AGG and baseline methods regarding the three criteria on different datasets are shown in Table 10, 11, and 12. The best performance for each dataset is denoted with

boldface. Overall I-AGG has higher performance in terms of analysis capability, summarization quality, and imputation capability.

It is noteworthy that I-AGG has higher classification accuracy on most datasets except the StandWalkJump and even achieves better performance than the original long data in some cases. This is because the original data contains some undesired noise or outliers, impacting the performance of classification. However, the summarization process, working as a low pass filter, mitigates the impact of those parts by aggregating those time-ticks with surrounding data. Similarly, the DWT approximation over-performs the original data, as it filters out these noisy parts and keeps only the higher frequency signals. We show the processing time on both the original data and the summarized data (averaged by all methods). Since we apply different classification methods on different datasets, it is meaningless to compare cross rows. Apparently, the summarized data can save up to 98% of computational time (712.48s v.s. 14.2s) without accuracy loss.

Dataset	Length (m)	Time of original data (s)	Length of PAA	Length of DWT	Length of I-AGG	Time of I-AGG (s)
AtrialFibrillation	640	6.18	92	84	93	0.11
Cricket	1197	712.48	171	154	172	14.20
HeartBeat	405	12.59	58	55	60	2.0
StandWalkJump	2500	2.5	358	317	352	0.59

Table 9: Analysis capability comparison on different datasets (Length and Time spend)

For the summarization quality, the proposed summarized data preserves more information and provides a better-reconstructed result (lower RMSE) on most datasets. Based on the experimental results, most of the significant parts detected by I-AGG have complex dynamic trends, such as higher variation (e.g., the middle part in the time series in Figure.29). Therefore, compared to the less complex parts (e.g., flat pattern), aggregating those parts with higher resolution would benefit the reconstruction process.

Finally, we demonstrate the imputation capability of our proposed method. Even if a large amount of time-ticks is missing (i.e., 40%), the selector can correctly locate the important information using the data inferred by the VAE generator. Figure 30 shows a score output example of a

Dataset	Accuracy of original data	Accuracy of PAA	Accuracy of DWT	Accuracy of I-AGG
AtrialFibrillation	0.2	0.2	0.2	**0.267**
Cricket	1.0	**1.0**	**1.0**	**1.0**
HeartBeat	0.722	0.702	0.6	**0.712**
StandWalkJump	0.4	0.333	**0.6**	0.467

Table 10: Analysis capability comparison on different datasets (Classification accuracy)

time series with 30% missing values. The top figure is the original sequence with different colors means different dimensions. The middle one is the data with missing value, and the last one is the output of the selector (i.e., score $\vec{s}$). Clearly, the selector successfully detects these two bumpy areas and assigns a higher score on these parts. In Table 12, we summarized the reconstruction result of different methods on missing data and demonstrate that I-AGG has better performance.

5.3.3 Analysis

- **Shapelet preserving:** In this section, we take one dataset, Cricket, as an example to interpret our summarized result. Generally, summarization is expected to preserve some important information for analysis while reducing computational cost and storage requirements. Then a question is raised – what is the golden rule of the important information for this unsupervised problem. Although the three evaluation metrics above implicitly indicate that the proposed method can preserve important information, it is hard to understand how it takes advantage. We propose to utilize the shapelet as one crucial piece of information to visualize the benefit explicitly. Shapelet is a subsequence of the time series data which can be used to identify the class membership. It is a powerful tool to measure the phase-independent similarity between time series. In order to keep the classification accuracy, the summarized results are expected to preserve this information. Classic summarization methods, such as PAA, may flatten the shapelet due to averaging within a wide range of neighborhoods. In contrast, under

Dataset	RMSE of PAA	RMSE of DWT	RMSE of I-AGG
AtrialFibrillation	0.532	0.773	**0.516**
Cricket	0.146	0.44	**0.144**
HeartBeat	0.293	0.525	**0.290**
StandWalkJump	**0.124**	0.507	0.134

Table 11: Summarization quality comparison on different datasets

the assumption that the shapelet subsequences contain critical information meaning they have higher informative scores, our proposed method is able to preserve this information with a higher aggregation resolution.

Figure 31 shows three examples to explain the preserved shapelet information by different summarization methods visually. These examples are generated using the subsequence of Cricket datasets[1]. The bold red line in the original time series highlights the shapelets discovered by sktime [88], and the bold red line in the PAA and I-AGG summarized data are the corresponding subsequence. In these experiments, PAA and I-AGG summarized data are truncated by around 85%. Specifically, PAA is generated by averaging every seven time-ticks, and I-AGG preserves 15% of the data. It is evident that the spikes in Figure 31 (row one and row two) have been counteracted with surrounded regular value time ticks in PAA, while I-AGG can capture it with higher resolution. Although I-AGG summarized data in (row one) does not preserve all three spikes, it successfully shows a dramatic change at that part. In Figure 31 (row three), PAA summarization lowers the value of the bump at the beginning of the shapelet, which should have a similar value as the highest point, while I-AGG preserves this property.

- **Irregular sampled and unaligned time series:** It is noteworthy that the I-AGG learns a specific aggregation resolution for each time-series data. Therefore, the summarized data within the same dataset are irregular sampled and unable to align. This irregular-sampled data also

[1] http://www.timeseriesclassification.com/description.php?Dataset=CricketX

Missing Ratio	RMSE of PAA	RMSE of DWT	RMSE of I-AGG
10%	0.297	0.532	**0.289**
20%	0.302	0.532	**0.296**
30%	0.308	0.545	**0.304**
40%	0.319	0.558	**0.309**

Table 12: Imputation capability on Cricket dataset with different missing ratios.

frequently happens in the real world. For example, the clinic data are usually recorded at irregular time intervals due to the patients' available time. Although this brings challenges for classical machine learning methods, such as Logistic Regression, many recent works have been proposed to tackle this problem with high performance. For example, P-VAE and P-BiGAN [85] infer the latent space distribution of the irregular-sample time series and then use it for classification. ODE-RNN [118] proposes to model continuous dynamics by latent ordinary differential equation (ODE) models with a neural network.

5.3.4 Practitioner's Guide

Due to the different performance of I-AGG under different datasets, we provide some guidance based on the applicability we gained from the previously discussed experiments to help the readers select the appropriate summarization method for different types of datasets.

- **Complex subsequences**: If the time series data contains some complex subsequences, which cannot be captured by coarser granularity, such as the examples in Figure 31 (row one)(row two), I-AGG is preferable. However, it should be noticed that I-AGG may not work if the data has several sparse spikes, which require the finest level of granularity.
- **Unsmooth data**: I-AGG is hard to take advantage of smooth time series data. If the data is

smooth enough, such as the gunpoint dataset[2] (Fig 32), our summarized data is expected to return a similar result as PAA.

- **Data with noise**: If the data contains too much noise that shields the dominant information, I-AGG cannot be used for denoising. It may consider the noisy part as an informative sequence and assign it to a high informative score. In figure 33 we show the effectiveness of the noise on the summarization quality (i.e., reconstruction accuracy). We randomly add noise following normal distribution on simulated data and perform both I-AGG and PAA on it. Y-axis is the RMSE difference between I-AGG and PAA, the x-axis is the level of noise added to the data. Postive means I-AGG over-performs PAA. It is not surprising that I-AGG works better than the PAA when the noise level is less then certain threshold. While the noise increases, it starts to disturb the information discovery processing and results in bad summarization.

5.4 Conclusion

In conclusion, we propose a novel data summarization method, I-AGG, which aggregates different segments using different resolutions. The resolutions are determinate by a VAE-GAN framework which tends to preserve more information in the crucial part. We evaluate the proposed summarization on several time-series data from UEA multivariate time series classification archive. The results show that our aggregated data are able to remain a similar classification accuracy as the original data, preserve enough information to reconstruct back to the original data, and impute the missing value of data. We also provide analysis and practitioners' guides for the use cases of our proposed method.

[2] http://www.timeseriesclassification.com/description.php?Dataset=GunPoint

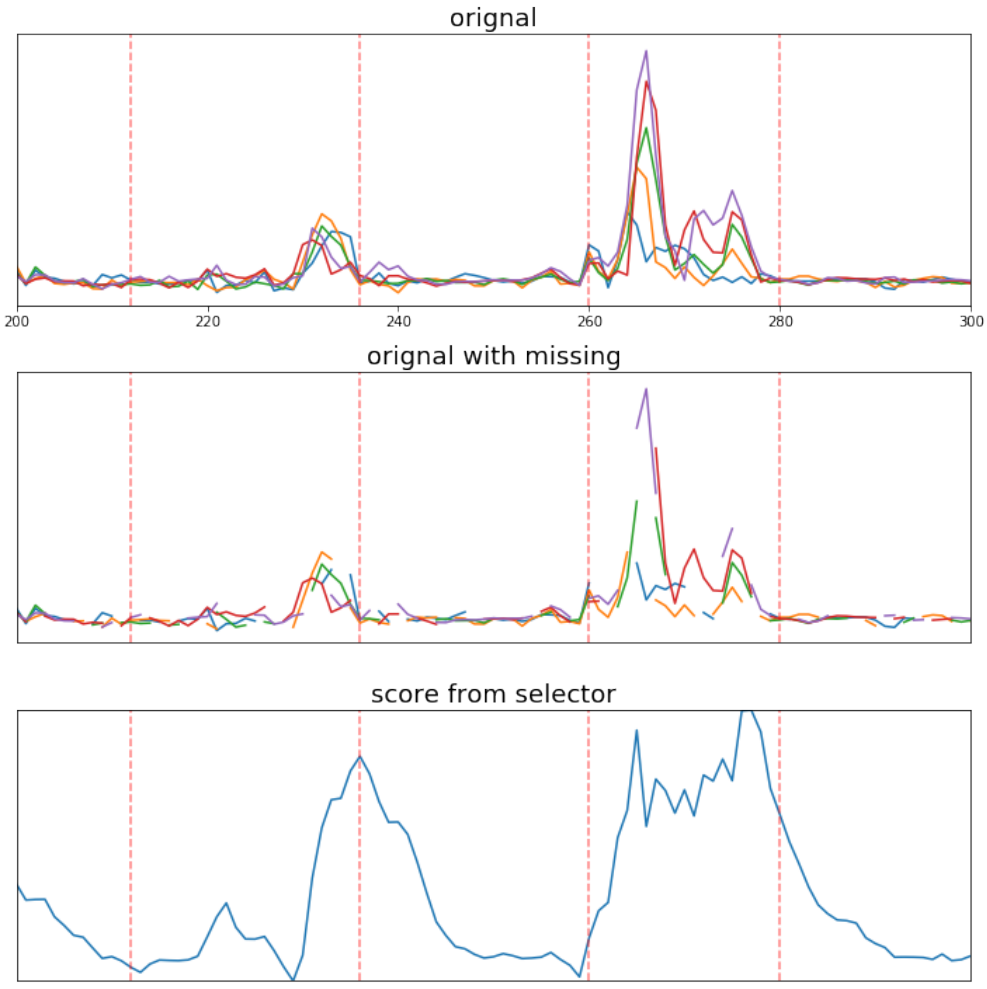


Figure 30: Example of the score output for data with 30% of missing.

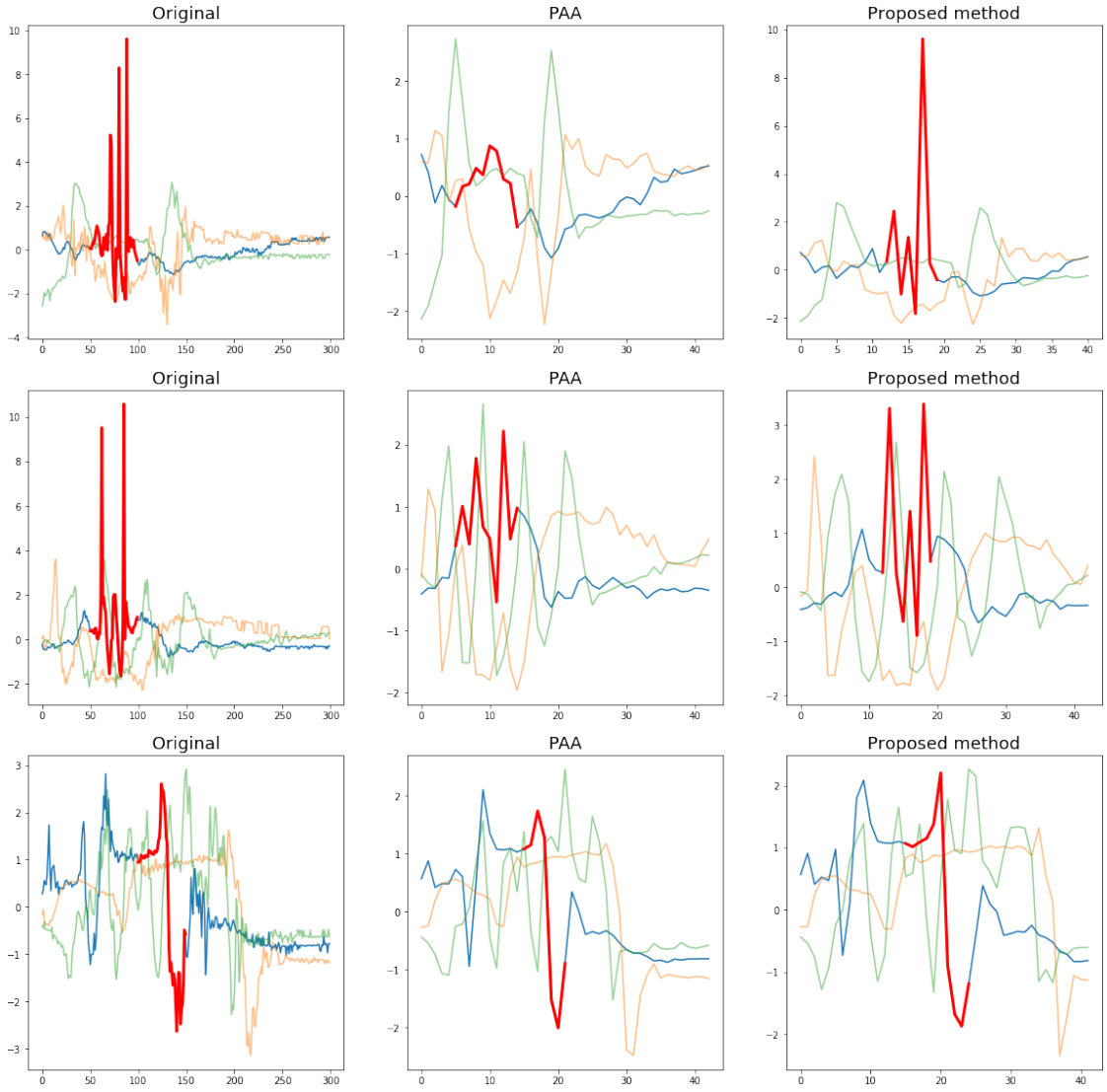


Figure 31: Illustration of the effectiveness of I-AGG over PAA with three examples.

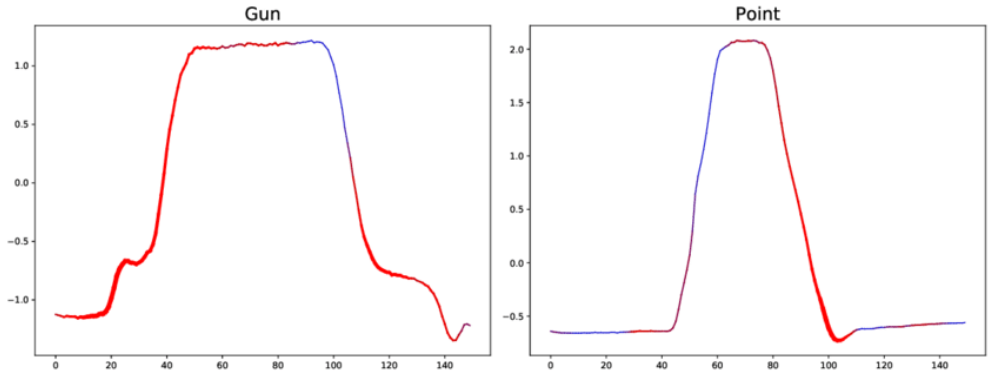


Figure 32: Example of gunpoint dataset.

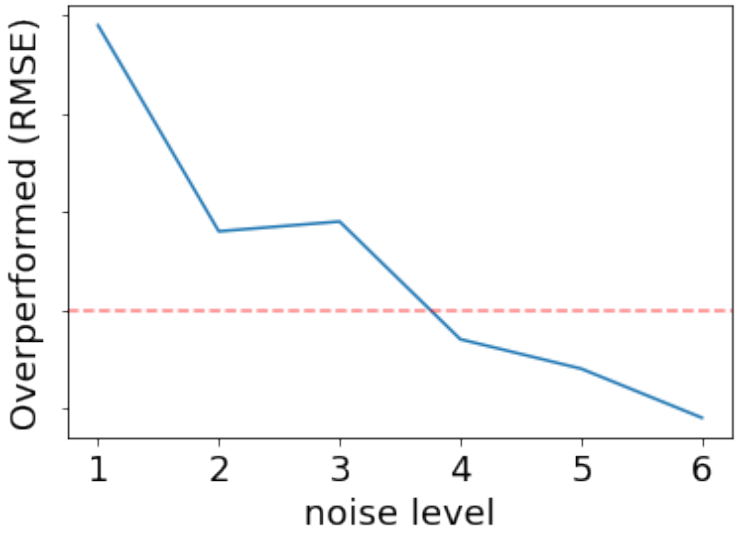


Figure 33: Reconstruction performance under different levels of noise.

6.0 CONCLUSION AND FUTURE DIRECTION

6.1 Concluding

In the thesis, I discussed our works to efficiently process and utilize the aggregated data while preserving the performance of analysis from three research areas, including:

- Data disaggregation: I present ARES which reconstructs historical data by automatically discover a dominant pattern of the target series. I present TURBOLIFT that refines the solutions provided by existing disaggregation methods.
- Data navigation: I propose SMARTPROGNOSIS to automatically select appropriate aggregation records and machine learning methods for a medical task. the trade-off between time complexity and performance.
- Data summarization: We propose I-AGG to summarize time series data with a different aggregation frequency identified by a VAE-GAN framework.

For each area, we showcase a detailed application scenario to demonstrate the benefit of our works. We also include comprehensive experiment results and practitioner's guidelines to provide insight and appropriate use-cases of our works.

6.2 Future Direction

In this part, we list some future directions that can be done in aggregated data.

- Disaggregate data **without explicit domain knowledge or dominate patterns**: In Chapter 3, we introduced two methods for disaggregation (ARES and TURBOLIFT). Both methods utilize the dominant pattern of the time series to facilitate the disaggregation, e.g., smoothness, Annihilating Filters based pattern. But sometimes, it is crucial to find an appropriate pattern for historical data. Therefore, a more intelligent way to deal with this problem is necessary.

- Navigating data **with more complexity algorithm**: in SMARTPROGNOSIS, we only considered some simple classification algorithms, such as Logistic Regression, KNN, etc. This is limited by the capability of the Genetic Program. However, with more complicated and accurate algorithms proposed for classification, it is crucial to discover a more efficient data navigation method.
- Summarizing data **with high variance**: it is not uncommon that numerical time series data contains undesired noise. But as mentioned in Chapter 5, I-AGG will trade the noise as dominant information and assign higher resolution, which results in worse performance. Therefore, it is important that future algorithms can consider decoupling the noise and the underlying information.

Appendix DATA DISAGGREGATION ADDITIONAL RESOURCES

A.1 Formal Justification of the TURBOLIFT Analytical Solution

In this part, we justify our analytical solution by proving the Lemma 3.2.1.

A.1.1 Preliminaries

First, we introduce some mathematics background. Given a real Toeplitz matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, we have the following properties [53]:

- $\mathbf{A}^T\mathbf{A} \in \mathbb{R}^{N \times N}$ is a positive semi-definite matrix.

- The Singular Value Decomposition (SVD) of $\mathbf{A}^T\mathbf{A}$ is $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$, where $\mathbf{U} \in \mathbb{R}^{N \times N}$ contains a set of singular vectors of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{\Sigma} \in \mathbb{R}^{N \times N}$ is a diagonal matrix with non-negative singular values $\lambda_i >= 0$ $(i = 1, 2, ..., N)$ on the diagonal.

- The singular value matrix $\mathbf{\Sigma}$ can be written as a partitioned matrix $\begin{bmatrix} \mathbf{\Sigma}_1 & 0 \\ 0 & 0 \end{bmatrix}$ where $\mathbf{\Sigma}_1 \in \mathbb{R}^{M \times M}$ contains all non-zero singular values. Correspondingly the matrix $\mathbf{U}$ can split into two parts $[\mathbf{U}_1, \mathbf{U}_2]$, where $\mathbf{U}_1 \in \mathbb{R}^{N \times M}$ contains the singular vectors corresponding to the non-zero singular values, while $\mathbf{U}_2 \in \mathbb{R}^{N \times (N-M)}$ contains the remaining part. Based on the property of a singular vector, $\mathbf{U}_1 \perp \mathbf{U}_2$ $(\mathbf{U}_1^T\mathbf{U}_2 = 0)$ and $\mathbf{U}_2 \perp \mathbf{A}^T$ $(\mathbf{U}_2^T\mathbf{A}^T = 0)$

- Suppose we have

$$\begin{aligned}\mathbf{\Lambda} &= \mathbf{\Sigma} + \mathbf{I}_n \\ &= \begin{bmatrix} \mathbf{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{I}_M & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N-M} \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 + 1 & ... & 0 & . & . & 0 \\ ... & . & . & . & . & ... \\ 0 & ... & \lambda_M + 1 & . & . & 0 \\ 0 & ... & 0 & 1 & ... & 0 \\ ... & . & . & . & . & ... \\ 0 & ... & 0 & 0 & ... & 1 \end{bmatrix}\end{aligned}$$

where $\lambda_i + 1 > 1$ $(i = 1, 2, ..., M)$ and $\mathbf{I}_n$ is the n-dimension identity matrix.

From the geometric series,

$$\begin{aligned}\frac{1}{\beta} + (\frac{1}{\beta})^2 + ... + (\frac{1}{\beta})^n & \\ &= \frac{1}{\beta}(1 + ... + (\frac{1}{\beta})^{n-1}) \\ &= \frac{1 - \beta^{-n}}{\beta - 1}\end{aligned}$$

With $n \to \infty$, if $\beta > 1$, the equation converge to $\frac{1}{\beta - 1}$, while if $\beta = 1$, the equation equal to n.

Then

$$\lim_{n\to\infty} \sum_{k=1}^{n} \mathbf{\Lambda}^{-k} = n \begin{bmatrix} \frac{1}{\lambda_1} & ... & 0 & . & . & 0 \\ ... & . & . & . & . & ... \\ 0 & ... & \frac{1}{\lambda_M} & . & . & 0 \\ 0 & ... & 0 & n & ... & 0 \\ ... & . & . & . & . & ... \\ 0 & ... & 0 & 0 & ... & n \end{bmatrix}$$

- Since $\lim_{n\to\infty} \frac{1}{\beta^n} = 0$, for $\beta > 1$,

$$\lim_{n\to\infty} \mathbf{\Lambda}^{-n} =$$

$$\lim_{n\to\infty} \begin{bmatrix} \frac{1}{(\lambda_1+1)^n} & \dots & 0 & . & . & 0 \\ \dots & . & . & . & . & \dots \\ 0 & \dots & \frac{1}{(\lambda_M+1)^n} & . & . & 0 \\ 0 & \dots & 0 & 1 & \dots & 0 \\ \dots & . & . & . & . & \dots \\ 0 & \dots & 0 & 0 & \dots & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \dots & 0 & . & . & 0 \\ \dots & . & . & . & . & \dots \\ 0 & \dots & 0 & . & . & 0 \\ 0 & \dots & 0 & 1 & \dots & 0 \\ \dots & . & . & . & . & \dots \\ 0 & \dots & 0 & 0 & \dots & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_{N-M} \end{bmatrix}$$

A.1.2 Proof of Lemma 3.2.1

Next we prove the Lemma 3.2.1 step by step.

Proof. First, we fix the $\vec{z}$ with any vector $\vec{z}_0$ as the initial vector

$$\min_{\vec{x}} \quad \|\mathbf{O}\vec{x} - \vec{v}\|_2^2 + \|\vec{x} - \vec{z}_0\|_2^2$$

or equivalently

$$\min_{\vec{x}} \quad \| \begin{bmatrix} \mathbf{O} \\ \mathbf{I} \end{bmatrix} \vec{x} - \begin{bmatrix} \vec{v} \\ \vec{z}_0 \end{bmatrix} \|_2^2 \tag{18}$$

We can get the unique solution of this problem by using the Moore-Penrose pseudo-inverse

$$\begin{aligned}\vec{x}_1 &= (\begin{bmatrix}\mathbf{O}^T & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{O}\\ \mathbf{I}\end{bmatrix})^{-1}\begin{bmatrix}\mathbf{O}^T & \mathbf{I}\end{bmatrix}\begin{bmatrix}\vec{v}\\ \vec{z}_0\end{bmatrix}\\ &= (\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-1}(\mathbf{O}^T\vec{v}+\vec{z}_0)\\ &= (\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-1}\mathbf{O}^T\vec{v}+(\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-1}\vec{z}_0\end{aligned}$$

Now we update the $\vec{z}$ by making it equals to the $\vec{x}_1$. So the problem, in the second iteration, changes to

$$\min_{\vec{x}} \quad \| \begin{bmatrix}\mathbf{O}\\ \mathbf{I}\end{bmatrix}\vec{x} - \begin{bmatrix}\vec{v}\\ \vec{x}_1\end{bmatrix} \|_2^2$$

Then the $\vec{x}$ can be updated using the same method as above.

$$\begin{aligned}\vec{x}_2 &= (\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-1}(\mathbf{O}^T\vec{v}+\vec{x}_1)\\ &= (\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-1}\mathbf{O}^T\vec{v}+(\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-1}[(\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-1}\mathbf{O}^T\vec{v}\\ &+ (\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-1}\vec{z}_0]\\ &= (\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-1}\mathbf{O}^T\vec{v}+(\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-2}\mathbf{O}^T\vec{v}+(\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-2}\mathbf{z}_0\end{aligned}$$

With the similar step, we have:

$$\begin{aligned}\vec{x}_3 = (\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-1}\mathbf{O}^T\vec{v}+(\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-2}\mathbf{O}^T\vec{v}\\ +(\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-3}\mathbf{O}^T\vec{v}+(\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-3}\vec{z}_0\end{aligned}$$

Therefore, with the Mathematical Induction, we can infer the formula of $\vec{x}$ after n iterations:

$$\vec{x}_n = (\sum_{k=1}^{n}(\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-k}\mathbf{O}^T\vec{v})+(\mathbf{O}^T\mathbf{O}+\mathbf{I})^{-n}\vec{z}_0 \tag{19}$$

Next we infer the convergent solution. Based on the preliminaries in the Section A.1.1, the Singular Value Decomposition of $\mathbf{O}^T\mathbf{O}$ is $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ and all the singular values are non-negative. The $\mathbf{\Sigma}$ can be

partitioned as $\begin{bmatrix} \mathbf{\Sigma}_1 & 0 \\ 0 & 0 \end{bmatrix}$ and the matrix $\mathbf{U}$ can be split into two parts $[\mathbf{U}_1, \mathbf{U}_2]$, where $\mathbf{U}_1 \in \mathbb{R}^{N\times M}$ contains the singular vectors corresponding to non-zero singular values, while $\mathbf{U}_2 \in \mathbb{R}^{N\times(N-M)}$ contains the rest part. Then the $\mathbf{O}^T\mathbf{O} + \mathbf{I}$ term in the derivation of $\vec{x}_n$ (see Eq. (19)) can be decomposed as

$$\mathbf{O}^T\mathbf{O} + \mathbf{I} = \underbrace{\begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix}}_{\mathbf{U}} \underbrace{(\begin{bmatrix} \mathbf{\Sigma}_1 & 0 \\ 0 & 0 \end{bmatrix} + \mathbf{I})}_{\mathbf{\Lambda}} \underbrace{\begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix}^T}_{\mathbf{U}^T}$$

So, we have

$$\begin{aligned}
(\mathbf{O}^T\mathbf{O} + \mathbf{I})^{-1} &= \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^T \\
(\mathbf{O}^T\mathbf{O} + \mathbf{I})^{-2} &= (\mathbf{O}^T\mathbf{O} + \mathbf{I})^{-1}(\mathbf{O}^T\mathbf{O} + \mathbf{I})^{-1} = \mathbf{U}\mathbf{\Lambda}^{-2}\mathbf{U}^T \\
&\vdots \\
(\mathbf{O}^T\mathbf{O} + \mathbf{I})^{-n} &= \mathbf{U}\mathbf{\Lambda}^{-n}\mathbf{U}^T
\end{aligned}$$

Then the derivation of $\vec{x}_n$ can be rewritten as,

$$\begin{aligned}
\vec{x}_n &= (\sum_{k=1}^{n} \mathbf{U}\mathbf{\Lambda}^{-k}\mathbf{U}^T)\mathbf{O}^T\vec{v} + \mathbf{U}\mathbf{\Lambda}^{-n}\mathbf{U}^T\vec{z}_0 \\
&= \mathbf{U}(\sum_{k=1}^{n} \mathbf{\Lambda}^{-k})\mathbf{U}^T\mathbf{O}^T\vec{v} + \mathbf{U}\mathbf{\Lambda}^{-n}\mathbf{U}^T\vec{z}_0
\end{aligned}$$

Since $\mathbf{U}_2 \perp \mathbf{O}^T$,

$$\begin{aligned}
\mathbf{U}^T\mathbf{O}^T &= \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix}^T \mathbf{O}^T = \begin{bmatrix} \mathbf{U}_1^T \\ \mathbf{U}_2^T \end{bmatrix} \mathbf{O}^T \\
&= \begin{bmatrix} \mathbf{U}_1^T\mathbf{O}^T \\ \mathbf{U}_2^T\mathbf{O}^T \end{bmatrix} = \begin{bmatrix} \mathbf{U}_1^T\mathbf{O}^T \\ 0 \end{bmatrix}
\end{aligned}$$

We have

$$\vec{x}_n = \underbrace{\mathbf{U}\,\underbrace{(\sum_{k=1}^{n}\mathbf{\Lambda}^{-k})}_{\mathbf{D}}\underbrace{\begin{bmatrix}\mathbf{U}_1^T\mathbf{O}^T\vec{v}\\ 0\end{bmatrix}}_{\varphi}}_{\mathbf{\Psi}} + \mathbf{U}\mathbf{\Lambda}^{-n}\mathbf{U}^T\vec{z}_0 \tag{20}$$

As all the singular values of $\mathbf{O}^T\mathbf{O}$ are non-negative, the diagonal elements of $\mathbf{\Lambda} = \mathbf{\Sigma} + \mathbf{I}$ are larger than 1, showing as

$$\begin{bmatrix}\lambda_1+1 & ... & 0 & . & . & 0\\ ... & . & . & . & . & ...\\ 0 & ... & \lambda_M+1 & . & . & 0\\ 0 & ... & 0 & 1 & ... & 0\\ ... & . & . & . & . & ...\\ 0 & ... & 0 & 0 & ... & 1\end{bmatrix}$$

where $\lambda_i > 0$ $(i = 1, 2, ...M)$. As $n \to \infty$, we have

$$\mathbf{D} = \begin{bmatrix}\frac{1}{\lambda_1} & ... & 0 & . & . & 0\\ ... & . & . & . & . & ...\\ 0 & ... & \frac{1}{\lambda_M} & . & . & 0\\ 0 & ... & 0 & n & ... & 0\\ ... & . & . & . & . & ...\\ 0 & ... & 0 & 0 & ... & n\end{bmatrix}$$

After multiplying with φ (especially the lower zero part), the lower right part of D vanishes. So, as $n \to \infty$,

$$\mathbf{\Psi} = \mathbf{U}_1\begin{bmatrix}\frac{1}{\lambda_1} & ... & 0\\ . & ... & .\\ 0 & ... & \frac{1}{\lambda_M}\end{bmatrix}\mathbf{U}_1^T\mathbf{O}^T\vec{v}$$

On the other hand, for the term $\mathbf{U}\mathbf{\Lambda}^{-n}\mathbf{U}^T\vec{z}_0$ in Eq. (20)

$$\lim_{n\to\infty}\mathbf{\Lambda}^{-n} = \lim_{n\to\infty}\begin{bmatrix} (\frac{1}{\lambda_1+1})^n & \dots & 0 & . & . & 0 \\ \dots & . & . & . & . & \dots \\ 0 & \dots & (\frac{1}{\lambda_M+1})^n & . & . & 0 \\ 0 & \dots & 0 & 1 & \dots & 0 \\ \dots & . & . & . & . & \dots \\ 0 & \dots & 0 & 0 & \dots & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I} \end{bmatrix}$$

Then we have:

$$\lim_{n\to\infty}\mathbf{U}\mathbf{\Lambda}^{-n}\mathbf{U}^T\vec{z}_0 = \begin{bmatrix}\mathbf{U}_1 & \mathbf{U}_2\end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{U}_1^T \\ \mathbf{U}_2^T\end{bmatrix}\vec{z}_0$$
$$= \mathbf{U}_2\mathbf{U}_2^T\vec{z}_0$$

Hence, the limit of $\vec{x}_n$ as $n \to \infty$ can be directly computed as

$$x_\infty = \mathbf{U}_1\begin{bmatrix}\frac{1}{\lambda_1} & \dots & 0 \\ . & \dots & . \\ 0 & \dots & \frac{1}{\lambda_M}\end{bmatrix}\mathbf{U}_1^T\mathbf{O}^T\vec{v} + \mathbf{U}_2\mathbf{U}_2^T\vec{z}_0$$
$$= \mathbf{U}_1\mathbf{\Sigma}^{-1}\mathbf{U}_1^T\mathbf{O}^T\vec{v} + \mathbf{U}_2\mathbf{U}_2^T\vec{z}_0$$

□

A.2 Additional Experiments of TURBOLIFT

A.2.1 Data

In addition to the data described in Section 3.3.1, we evaluate TURBOLIFT using scenarios, spiky and sparse, that appear in practice. We simulate the data using the real time series data as

follows:

- **Spiky:** We create spikes at 20% randomly chosen time-ticks in the data, the value of the spikes is equal to the maximum value in the time series.
- **Sparse:** We set the values of 80% randomly chosen time-ticks in the original data to be zeros.

The reason for choosing the spiky and sparse types of data, in addition to the regular time series (i.e., real data), is that they are ubiquitous in various applications. Using the accurate reconstruction of spiky data, analysts can detect anomalous events. On the other hand, the accurate reconstruction of sparse data is very appreciated in compressed sensing [37]. The spiky and sparse scenarios are created using NY Measles data.

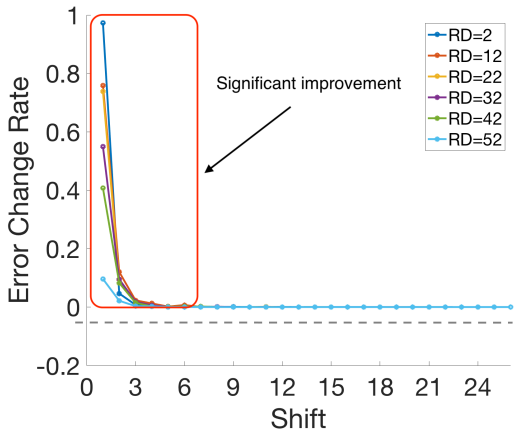
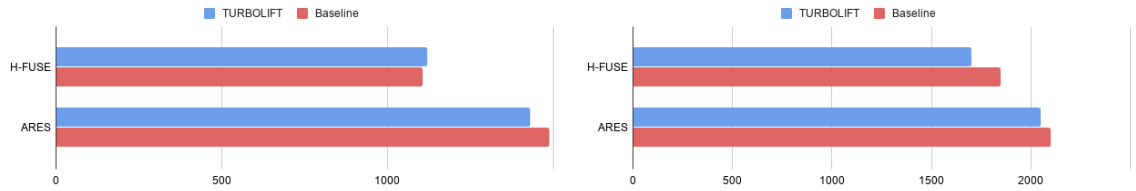


Figure 34: TURBOLIFT error change rate on spiky NY Measles data.

A.2.2 Spiky historical data

Next, we evaluate TURBOLIFT on the case when the series has some spikes, with values that are much larger than their neighboring time-ticks. Figure 34 shows the error change rate using the spiky NY Measles data when the solution of H-FUSE is used as the initial vector in TURBO-LIFT. We can see that with *Shift* less than 3, the error improvement is larger than 10%, except with $RD = 52$. Note that the error change rate measures the improvement in the RMSE. However,

one of the main practical goals with spiky data is to evaluate the accuracy of detecting anomalous events (i.e., spikes).



(a) TURBOLIFT compared to the baseline methods for **non-spiky time-ticks**

(b) TURBOLIFT compared to the baseline methods for **spiky time-ticks**

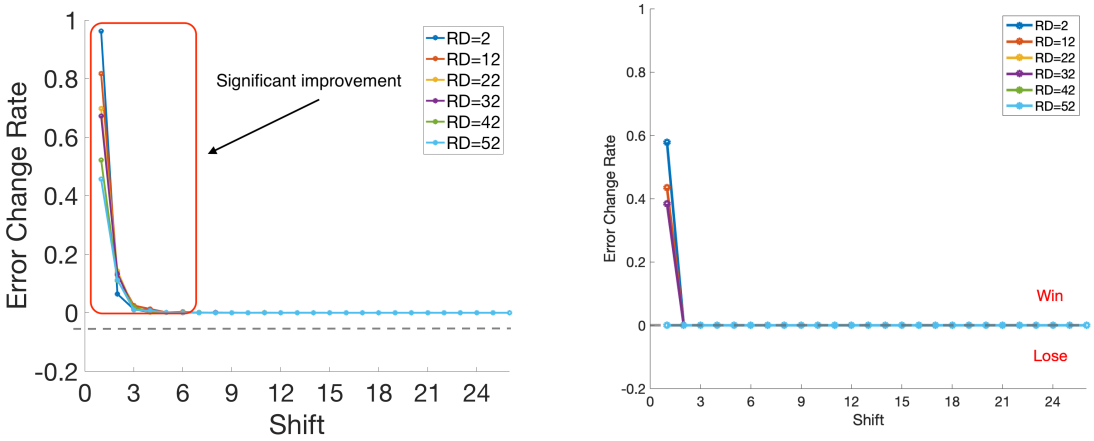
Figure 35: Comparison of the TURBOLIFT and the baseline methods.

We measure the improvement in the estimation accuracy at the time-ticks with spikes and the rest of time-ticks separately. We show the RMSE of TURBOLIFT and the baselines measured at the subset of "normal" time-ticks in Figure 35 (a), and the subset of time-ticks with spikes in Figure 35 (b). The RMSE is averaged over the scenarios when RD is changing from 2 to 52 with increments of 10, and $Shift$ is spanning from 1 to 10 with an increment of 1. There is a clear improvement in estimating the values at the spiky time-ticks, which shows the advantage of TURBOLIFT in anomaly detection from aggregated data. Moreover, TURBOLIFT preserves the accuracy of the estimation at the remaining time-ticks. Overall, the RMSE of TURBOLIFT is smaller than the baselines over the entire time-ticks (by combining the error in both Figure 35 (a) and Figure 35 (b)).

A.2.3 Sparse historical data

In this section, we show the error change rate of TURBOLIFT on sparse data with initial solutions provided by H-FUSE and BP baselines. As mentioned in Section 2.1.2, BP method assumes the data is sparse, which is favorable in this case. TURBOLIFT improves the reconstruction accuracy, especially with *Shift* less than 7, when H-FUSE used for initialization as shown in Figure 36 (a). On the other hand, when TURBOLIFT is initialized using BP, the error is improved only with $Shift = 1$ as shown in Figure 36 (b). H-FUSE penalizes the larger jumps between any two

successive time-ticks; therefore, it is not well suited to find a sparse solution. Whereas, BP is designed specifically to find the sparsest solution to an under-determined linear system.



(a) TURBOLIFT compared to **H-FUSE-S**

(b) TURBOLIFT compared to **BP**

Figure 36: TURBOLIFT error change rate on sparse NY Measles data.

In some detection applications, e.g., outlier detection, a binary answer is more beneficial (i.e., whether or not an event is detected). In our ground-truth sparse data, there are 80 non-zero entries (20% of the entire time-ticks). In the reconstructed series, we pick the time-ticks with the largest 80 values and consider them as the "non-empty" time-ticks. We count how many of the "non-empty" time-ticks meet with the true non-zeros in the ground-truth series, i.e., *number of hits*. Figure 37 shows the histogram of the number of hits and its fitted normal distribution in the solutions given by H-FUSE and TURBOLIFT initialized with H-FUSE. The y-axis in Figure 37 represents the number of aggregation configurations where the disaggregation method achieved a specific number of hits. We include 60 configurations: the report duration ranges from $RD = 2$ to $RD = 52$ with increments of 10, and the shift varies from $Shift = 1$ to $Shift = 10$ with increment of 1. With all the considered configurations, H-FUSE detects only half of the non-empty time-ticks or less, and the mean of the normal distribution of its histogram is around 30. While TURBOLIFT shifts the mean of the distribution to the right, and remarkably detects more than 60 non-zero time-ticks in some cases.

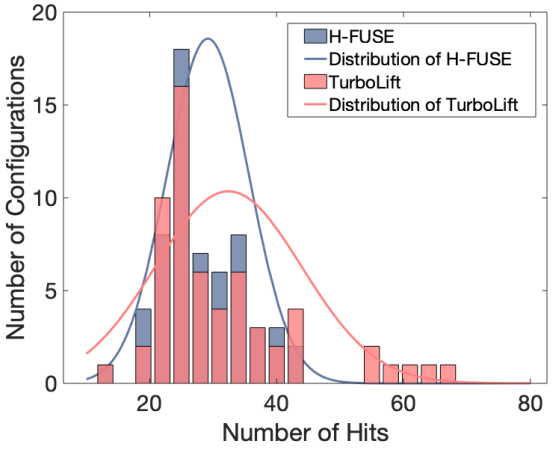


Figure 37: Histogram of number of hits (detected non-zero time-ticks).

Bibliography

[1] Christopher Adamson. *Mastering data warehouse aggregates: solutions for star schema performance*. John Wiley & Sons, 2012.

[2] Mohiuddin Ahmed. Data summarization: a survey. *Knowledge and Information Systems*, 58(2):249–273, 2019.

[3] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*, 2017.

[4] Faisal M Almutairi, Charilaos I Kanatsoulis, and Nicholas D Sidiropoulos. Prema: Principled tensor data recovery from multiple aggregated views. *arXiv preprint arXiv:1910.12001*, 2019.

[5] Faisal M Almutairi, Fan Yang, Hyun Ah Song, Christos Faloutsos, Nicholas Sidiropoulos, and Vladimir Zadorozhny. Homerun: scalable sparse-spectrum reconstruction of aggregated historical data. *Proceedings of the VLDB Endowment*, 11(11):1496–1508, 2018.

[6] Edilberto Amorim, Michelle Van der Stoel, Sunil B Nagaraj, Mohammad M Ghassemi, Jin Jing, Una-May O'Reilly, Benjamin M Scirica, Jong Woo Lee, Sydney S Cash, and M Brandon Westover. Quantitative eeg reactivity and machine learning for prognostication in hypoxic-ischemic brain injury. *Clinical Neurophysiology*, 130(10):1908–1916, 2019.

[7] Evlampios Apostolidis, Alexandros I Metsai, Eleni Adamantidou, Vasileios Mezaris, and Ioannis Patras. A stepwise, label-based approach for improving the adversarial training in unsupervised video summarization. In *Proceedings of the 1st International Workshop on AI for Smart TV Content Production, Access and Delivery*, pages 17–25, 2019.

[8] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

[9] Seyed-Ali Bahrainian and Andreas Dengel. Sentiment analysis and summarization of twitter data. In *2013 IEEE 16th International Conference on Computational Science and Engineering*, pages 227–234. IEEE, 2013.

[10] Martin Eduardo Baltazar-Lopez. *Applications of TAP-NDE technique to non-contact ultrasonic inspection in tubulars*. PhD thesis, Texas A&M University, 2005.

[11] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: fine-grained image generation through asymmetric training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2745–2754, 2017.

[12] Emelia J Benjamin, Paul Muntner, and Márcio Sommer Bittencourt. Heart disease and stroke statistics-2019 update: a report from the american heart association. *Circulation*, 139(10):e56–e528, 2019.

[13] Garrett Bernstein and Daniel Sheldon. Consistently estimating markov chains with noisy aggregate data. In *Artificial Intelligence and Statistics*, pages 1142–1150, 2016.

[14] Jens Bleiholder and Felix Naumann. Data fusion. *ACM Computing Surveys (CSUR)*, 41(1):1, 2009.

[15] Edgar F Borgatta and David J Jackson. *Aggregate data: Analysis and interpretation*. Sage Publications Beverly Hills/London, 1980.

[16] David Boulton and Martyn Hammersley. Analysis of unstructured data. *Data collection and analysis*, pages 282–297, 1996.

[17] Krisztian Buza, Júlia Koller, and Kristóf Marussy. Process: Projection-based classification of electroencephalograph signals. In *International Conference on Artificial Intelligence and Soft Computing*, pages 91–100. Springer, 2015.

[18] Krisztian Buza, Alexandros Nanopoulos, Tomáš Horváth, and Lars Schmidt-Thieme. Gramofon: General model-selection framework based on networks. *Neurocomputing*, 75(1):163–170, 2012.

[19] Lei Cai, Hongyang Gao, and Shuiwang Ji. Multi-stage variational auto-encoders for coarse-to-fine image generation. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 630–638. SIAM, 2019.

[20] Clifton W Callaway, Michael W Donnino, Ericka L Fink, Romergryko G Geocadin, Eyal Golan, Karl B Kern, Marion Leary, William J Meurer, Mary Ann Peberdy, Trevonne M Thompson, et al. Part 8: post–cardiac arrest care: 2015 american heart association guidelines update for cardiopulmonary resuscitation and emergency cardiovascular care. *circulation*, 132(18_suppl_2):S465–S482, 2015.

[21] Varun Chandola and Vipin Kumar. Summarization–compressing data into an informative representation. *Knowledge and Information Systems*, 12(3):355–378, 2007.

[22] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[23] Shaobing Chen and David Donoho. Basis pursuit. In *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 41–44. IEEE, 1994.

[24] Yunfan Chen, Lei Chen, and Chen Jason Zhang. Crowdfusion: A crowdsourced approach on data fusion refinement. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, pages 127–130. IEEE, 2017.

[25] Gregory C Chow and An-loh Lin. Best linear unbiased interpolation, distribution, and extrapolation of time series by related series. *The review of Economics and Statistics*, pages 372–375, 1971.

[26] Charles K Chui. *An introduction to wavelets*. Elsevier, 2016.

[27] Liu Chun-Lin. A tutorial of the wavelet transform. *NTUEE, Taiwan*, 2010.

[28] Jan Claassen, Kevin Doyle, Adu Matory, Caroline Couch, Kelly M Burger, Angela Velazquez, Joshua U Okonkwo, Jean-Rémi King, Soojin Park, Sachin Agarwal, et al. Detection of brain activation in unresponsive patients with acute brain injury. *New England Journal of Medicine*, 380(26):2497–2505, 2019.

[29] William AV Clark and Karen L Avery. The effects of data aggregation in statistical analysis. *Geographical Analysis*, 8(4):428–438, 1976.

[30] Patrick J Coppler, Jonathan Elmer, Luis Calderon, Alexa Sabedra, Ankur A Doshi, Clifton W Callaway, Jon C Rittenberger, Cameron Dezfulian, et al. Validation of the pittsburgh cardiac arrest category illness severity score. *Resuscitation*, 89:86–92, 2015.

[31] Graham Cormode and S Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *latin american symposium on theoretical informatics*, pages 29–38. Springer, 2004.

[32] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[33] Tommaso Di Fonzo. The estimation of m disaggregate time series when contemporaneous and temporal aggregates are known. *The Rev. of Econ. and Stats.*, pages 178–182, 1990.

[34] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[35] Xin Luna Dong and Felix Naumann. Data fusion: resolving data conflicts for integration. *Proceedings of the VLDB Endowment*, 2(2):1654–1655, 2009.

[36] Xin Luna Dong and Divesh Srivastava. Big data integration. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 1245–1248. IEEE, 2013.

[37] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[38] Callie M Drohan, Alessandra I Cardi, Jon C Rittenberger, Alexandra Popescu, Clifton W Callaway, Maria E Baldwin, and Jonathan Elmer. Effect of sedation on quantitative electroencephalography after cardiac arrest. *Resuscitation*, 124:132–137, 2018.

[39] Eric Eaton. Multi-resolution learning for knowledge transfer. In *AAAI*, pages 1908–1909, 2006.

[40] Jonathan Elmer and Clifton W Callaway. The brain after cardiac arrest. In *Seminars in neurology*, volume 37, pages 019–024. Thieme Medical Publishers, 2017.

[41] Jonathan Elmer, John J Gianakas, Jon C Rittenberger, Maria E Baldwin, John Faro, Cheryl Plummer, Lori A Shutter, Christina L Wassel, Clifton W Callaway, Anthony Fabio, et al. Group-based trajectory modeling of suppression ratio after cardiac arrest. *Neurocritical care*, 25(3):415–423, 2016.

[42] Jonathan Elmer, Bobby L Jones, Vladimir I Zadorozhny, Juan Carlos Puyana, Kate L Flickinger, Clifton W Callaway, and Daniel Nagin. A novel methodological framework for multimodality, trajectory model-based prognostication. *Resuscitation*, 137:197–204, 2019.

[43] Jonathan Elmer, Quan Zhou, Yichi Zhang, Fan Yang, and Vladimir I Zadorozhny. Brainflux: An integrated data warehousing infrastructure for dynamic health data. In *European Conference on Advances in Databases and Information Systems*, pages 135–143. Springer, 2019.

[44] Christos Faloutsos, H. V. Jagadish, and Nikolaos D. Sidiropoulos. Recovering information from summary data. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 36–45, 1997.

[45] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970, 2015.

[46] Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

[47] Thomas A Garrett. Aggregated versus disaggregated data in regression analysis: implications for inference. *Economics Letters*, 81(1):61–65, 2003.

[48] Mohammad Mahdi Ghassemi. *Life after death: techniques for the prognostication of coma outcomes after cardiac arrest*. PhD thesis, Massachusetts Institute of Technology, Cambridge, USA, 2018.

[49] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[50] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[51] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[52] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery*, 1(1):29–53, 1997.

[53] Robert M Gray et al. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2006.

[54] HJG Gundersen and EB Jensen. The efficiency of systematic sampling in stereology and its prediction. *Journal of microscopy*, 147(3):229–263, 1987.

[55] Viet Ha-Thuc, Duc-Cuong Nguyen, and Padmini Srinivasan. A quality-threshold data summarization algorithm. In *2008 IEEE International Conference on Research, Innovation and Vision for the Future in Computing and Communication Technologies*, pages 240–246. IEEE, 2008.

[56] David Lee Hall and John M Jordan. *Human-centered information fusion*. Artech House, 2010.

[57] David Lee Hall and Sonya AH McMullen. *Mathematical techniques in multisensor data fusion*. Artech House, 2004.

[58] David Hallac, Peter Nystrup, and Stephen Boyd. Greedy gaussian segmentation of multivariate time series. *Advances in Data Analysis and Classification*, 13(3):727–751, 2019.

[59] Georg Heinig et al. *Algebraic methods for Toeplitz-like matrices and operators*, volume 13. Birkhäuser, 2013.

[60] ZR Hesabi, Zahir Tari, A Goscinski, Adil Fahad, Ibrahim Khalil, and Carlos Queiroz. Data summarization techniques for big data—a survey. In *Handbook on Data Centers*, pages 1109–1152. Springer, 2015.

[61] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[62] Michael P. Holmes, Jr. Isbell, Charles Lee, and Alexander G. Gray. Quic-svd: Fast svd using cosine trees. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 673–680. Curran Associates, Inc., 2009.

[63] Ali Hormati and Martin Vetterli. Annihilating filter-based decoding in the compressed sensing framework. In *Optical Engineering+ Applications*, pages 670121–670121. International Society for Optics and Photonics, 2007.

[64] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[65] William H Inmon. *Building the data warehouse*. John wiley & sons, 2005.

[66] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[67] A Jain. Fast inversion of banded toeplitz matrices by circular decompositions. *IEEE transactions on acoustics, speech, and signal processing*, 26(2):121–126, 1978.

[68] Edwin T Jaynes. The well-posed problem. *Foundations of Physics*, 3(4):477–492, 1973.

[69] Yao "Henry" Jin, Brent D Williams, Travis Tokar, and Matthew A Waller. Forecasting with temporally aggregated demand signals in a retail supply chain. *Journal of Business Logistics*, 36(2):199–211, 2015.

[70] John David Kalbfleisch and Jerald F Lawless. Least-squares estimation of transition probabilities from aggregate data. *Canadian Journal of Statistics*, 12(3):169–182, 1984.

[71] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. Ieee, 2005.

[72] Sudha Kilaru Kessler, Alexis A Topjian, Ana M Gutierrez-Colina, Rebecca N Ichord, Maureen Donnelly, Vinay M Nadkarni, Robert A Berg, Dennis J Dlugos, Robert R Clancy, and Nicholas S Abend. Short-term outcome prediction by electroencephalographic features in children treated with therapeutic hypothermia after cardiac arrest. *Neurocritical care*, 14(1):37–43, 2011.

[73] Leslie Gordon Kiloh and John Walkinshaw Osselton. *Clinical electroencephalography*. Springer, 1966.

[74] Ralph Kimball. Aggregate navigation with (almost) no metadata. *DBMS*, 9(9):S15–22, 1996.

[75] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[76] J Zico Kolter and Matthew J Johnson. Redd: A public data set for energy disaggregation research. In *Workshop on data mining applications in sustainability (SIGKDD), San Diego, CA*, volume 25, pages 59–62, 2011.

[77] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *The Journal of Machine Learning Research*, 18(1):826–830, 2017.

[78] Nikolaos Kourentzes, Fotios Petropoulos, and Juan R Trapero. Improving forecasting by estimating time series structural components across multiple frequencies. *International Journal of Forecasting*, 30(2):291–302, 2014.

[79] John R Koza. Genetic programming. 1997.

[80] Sunil Kripalani, Frank LeFevre, Christopher O Phillips, Mark V Williams, Preetha Basaviah, and David W Baker. Deficits in communication and information transfer between hospital-based and primary care physicians: implications for patient safety and continuity of care. *Jama*, 297(8):831–841, 2007.

[81] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[82] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[83] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[84] Changhee Lee, William Zame, Ahmed Alaa, and Mihaela Schaar. Temporal quilting for survival analysis. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 596–605, 2019.

[85] Steven Cheng-Xian Li and Benjamin M Marlin. Learning from irregularly-sampled time series: A missing data perspective.

[86] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, 2003.

[87] Zongge Liu, Hyun Ah Song, Vladimir Zadorozhny, Christos Faloutsos, and Nicholas Sidiropoulos. H-fuse: Efficient fusion of aggregated historical data. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 786–794. SIAM, 2017.

[88] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J Király. sktime: A Unified Interface for Machine Learning with Time Series. In *Workshop on Systems for ML at NeurIPS 2019*.

[89] Rafael Lozano, Mohsen Naghavi, Kyle Foreman, Stephen Lim, Kenji Shibuya, Victor Aboyans, Jerry Abraham, Timothy Adair, Rakesh Aggarwal, Stephanie Y Ahn, et al. Global and regional mortality from 235 causes of death for 20 age groups in 1990 and 2010: a systematic analysis for the global burden of disease study 2010. *The lancet*, 380(9859):2095–2128, 2012.

[90] Gang Luo, Bryan L Stone, Michael D Johnson, Peter Tarczy-Hornoch, Adam B Wilcox, Sean D Mooney, Xiaoming Sheng, Peter J Haug, and Flory L Nkoy. Automating construction of machine learning models with clinical big data: proposal rationale and methods. *JMIR research protocols*, 6(8):e175, 2017.

[91] Carolina B Maciel, Mary M Barden, Teddy S Youn, Monica B Dhakar, and David M Greer. Neuroprognostication practices in postcardiac arrest patients: an international survey of critical care providers. *Critical care medicine*, 48(2):e107–e114, 2020.

[92] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. Unsupervised video summarization with adversarial lstm networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 202–211, 2017.

[93] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[94] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings*

of the 34th International Conference on Machine Learning-Volume 70, pages 2391–2400. JMLR. org, 2017.

[95] Sunil B Nagaraj, Marleen C Tjepkema-Cloostermans, Barry J Ruijter, Jeannette Hofmeijer, and Michel JAM van Putten. The revised cerebral recovery index improves predictions of neurological outcome after cardiac arrest. *Clinical neurophysiology*, 129(12):2557–2566, 2018.

[96] Frank Olken and Doron Rotem. Simple random sampling from relational databases. 1986.

[97] Randal S Olson, Nathan Bartley, Ryan J Urbanowicz, and Jason H Moore. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 485–492. ACM, 2016.

[98] Antigoni Panagiotopoulou and Vassilis Anastassopoulos. Super-resolution image reconstruction techniques: Trade-offs between the data-fidelity and regularization terms. *Information Fusion*, 13(3):185–195, 2012.

[99] Evangelos E Papalexakis. Mining large multi-aspect data: Algorithms and applications. 2016.

[100] Parth D Patel, Pranav B Lapsiwala, and Ravindra V Kshirsagar. Data aggregation in wireless sensor network. *International Journal of Managment, IT and Engineering*, 2(7):457–472, 2012.

[101] Jose Manuel Pavía-Miralles. A survey of methods to interpolate, distribute and extra-polate time series. *Journal of Service Science and Management*, 3(04):449, 2010.

[102] Jose Manuel Pavía-Miralles and Bernardí Cabrer-Borrás. On estimating contemporaneous quarterly regional gdp. *Journal of Forecasting*, 26(3):155–170, 2007.

[103] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[104] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. *arXiv preprint arXiv:1602.02355*, 2016.

[105] Daniël M Pelt and James A Sethian. A mixed-scale dense convolutional neural network for image analysis. *Proceedings of the National Academy of Sciences*, 115(2):254–259, 2018.

[106] Fotios Petropoulos and Nikolaos Kourentzes. Improving forecasting via multiple temporal aggregation. *Foresight: The International Journal of Applied Forecasting*, 34:12–17, 2014.

[107] Michael R Pinsky, Gilles Clermont, and Marilyn Hravnak. Predicting cardiorespiratory instability. *Critical care*, 20(1):1–8, 2016.

[108] Rimma Pivovarov and Noémie Elhadad. Automated methods for the summarization of electronic health records. *Journal of the American Medical Informatics Association*, 22(5):938–947, 2015.

[109] Yao Quanming, Wang Mengshuo, Jair Escalante Hugo, Guyon Isabelle, Hu Yi-Qi, Li Yu-Feng, Tu Wei-Wei, Yang Qiang, and Yu Yang. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*, 2018.

[110] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[111] Theodoros Rekatsinas, Manas Joglekar, Hector Garcia-Molina, Aditya Parameswaran, and Christopher Ré. Slimfast: Guaranteed results for data fusion and source reliability. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1399–1414. ACM, 2017.

[112] Huorong Ren, Mingming Liu, Zhiwu Li, and Witold Pedrycz. A piecewise aggregate pattern representation approach for anomaly detection in time series. *knowledge-based Systems*, 135:29–39, 2017.

[113] Facebook Research. Fast randomized svd. `https://research.fb.com/fast-randomized-svd/`, 2014.

[114] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[115] Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. Distribution matching in variational inference. *arXiv preprint arXiv:1802.06847*, 2018.

[116] Andrea O Rossetti, Mauro Oddo, Giancarlo Logroscino, and Peter W Kaplan. Prognostication after cardiac arrest and hypothermia: a prospective study. *Annals of neurology*, 67(3):301–307, 2010.

[117] Nicola Rossi et al. A note on the estimation of disaggregate time series when the aggregate is known. *The Review of Econ. and Stats.*, 64(4):695–696, 1982.

[118] Yulia Rubanova, Ricky TQ Chen, and David Duvenaud. Latent odes for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907*, 2019.

[119] Malin Rundgren, Ingmar Rosén, and Hans Friberg. Amplitude-integrated eeg (aeeg) predicts outcome after cardiac arrest and induced hypothermia. *Intensive care medicine*, 32(6):836, 2006.

[120] R Alberto Salinas, Christopher Richardson, Mongi A Abidi, and Ralph C Gonzalez. Data fusion: Color edge detection and surface reconstruction through regularization. *IEEE Transactions on Industrial Electronics*, 43(3):355–363, 1996.

[121] Christoph Sax and Peter Steiner. Temporal disaggregation of time series. 2013.

[122] David Seder, Sadie Denico, Teresa May, John Dziodzio, Lyn Ackert-Smith, Francis Lucas, Christine Lord, Ashley Eldridge, Barbara McCrum, and Richard Riker. 1464: Eeg suppression ratio predicts outcome 1-4 hours after resuscitation from cardiac arrest. *Critical Care Medicine*, 48(1):708, 2020.

[123] Ohad Shamir. Fast stochastic algorithms for svd and pca: Convergence properties and convexity. In *International Conference on Machine Learning*, pages 248–256, 2016.

[124] Elaine Shi, TH Hubert Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *Proc. NDSS*, volume 2, pages 1–17. Citeseer, 2011.

[125] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In *Proceedings of the 2015 SIAM international conference on data mining*, pages 289–297. SIAM, 2015.

[126] Cedric M Smith. Origin and uses of primum non nocere—above all, do no harm! *The Journal of Clinical Pharmacology*, 45(4):371–377, 2005.

[127] Steve Stehman. Estimating the kappa coefficient and its variance under stratified random sampling. *Photogrammetric Engineering and Remote Sensing*, 62(4):401–407, 1996.

[128] Alexis Steinberg, Clifton W Callaway, Robert M Arnold, Tobias Cronberg, Hiromichi Naito, Koral Dadon, Minjung Kathy Chae, and Jonathan Elmer. Prognostication after cardiac arrest: Results of an international, multi-professional survey. *Resuscitation*, 138:190–197, 2019.

[129] Petre Stoica and Randolph L Moses. *Introduction to spectral analysis*, volume 1. Prentice hall Upper Saddle River, 1997.

[130] Steven K Thompson. Adaptive cluster sampling. *Journal of the American Statistical Association*, 85(412):1050–1059, 1990.

[131] Tycho. https://www.tycho.pitt.edu.

[132] Willem G Van Panhuis, John Grefenstette, Su Yon Jung, Nian Shong Chok, Anne Cross, Heather Eng, Bruce Y Lee, Vladimir Zadorozhny, Shawn Brown, Derek Cummings, et al. Contagious diseases in the united states from 1888 to the present. *The New England journal of medicine*, 369(22):2152, 2013.

[133] Tielman T Van Vleck, Daniel M Stein, Peter D Stetson, and Stephen B Johnson. Assessing data relevance for automated generation of a clinical summary. In *AMIA annual symposium proceedings*, volume 2007, page 761. American Medical Informatics Association, 2007.

[134] Martin Vetterli, Pina Marziliano, and Thierry Blu. Sampling signals with finite rate of innovation. *IEEE transactions on Signal Processing*, 50(6):1417–1428, 2002.

[135] Yisen Wang, Bo Dai, Lingkai Kong, Sarah Monazam Erfani, James Bailey, and Hongyuan Zha. Learning deep hidden nonlinear dynamics from aggregate data. *arXiv preprint arXiv:1807.08237*, 2018.

[136] Zhengwei Wang, Qi She, and Tomas E Ward. Generative adversarial networks: A survey and taxonomy. *arXiv preprint arXiv:1906.01529*, 2019.

[137] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[138] Erik Westhall, Andrea O Rossetti, Anne-Fleur van Rootselaar, Troels Wesenberg Kjaer, Janneke Horn, Susann Ullén, Hans Friberg, Niklas Nielsen, Ingmar Rosén, Anders Åneman, et al. Standardized eeg interpretation accurately predicts prognosis after cardiac arrest. *Neurology*, 86(16):1482–1490, 2016.

[139] Sara Leingang Wiley, Babak Razavi, Prashanth Krishnamohan, Michael Mlynash, Irina Eyngorn, Kimford J Meador, and Karen G Hirsch. Quantitative eeg metrics differ between outcome groups and change over the first 72 h in comatose cardiac arrest patients. *Neurocritical care*, 28(1):51–59, 2018.

[140] David H Wolpert, William G Macready, et al. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[141] Huan Xu, Constantine Caramanis, and Shie Mannor. Robust regression and lasso. In *Advances in Neural Information Processing Systems*, pages 1801–1808, 2009.

[142] Fan Yang, Faisal M Almutairi, Hyun Ah Song, Christos Faloutsos, Nicholas D Sidiropoulos, and Vladimir Zadorozhny. Turbolift: fast accuracy lifting for historical data recovery. *The VLDB Journal*, pages 1–20, 2020.

[143] Fan Yang, Hyun Ah Song, Zongge Liu, Christos Faloutsos, Vladimir Zadorozhny, and Nicholas Sidiropoulos. Ares: Automatic disaggregation of historical data. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 65–76. IEEE, 2018.

[144] Jinsung Yoon, James Jordon, and Mihaela Van Der Schaar. Gain: Missing data imputation using generative adversarial nets. *arXiv preprint arXiv:1806.02920*, 2018.

[145] Li Yuan, Francis EH Tay, Ping Li, Li Zhou, and Jiashi Feng. Cycle-sum: Cycle-consistent adversarial lstm networks for unsupervised video summarization. *arXiv preprint arXiv:1904.08265*, 2019.

[146] Vladimir Zadorozhny and John Grant. A systematic approach to reliability assessment in integrated databases. *Journal of Intelligent Information Systems*, 46(3):409–424, 2016.

[147] Vladimir Zadorozhny and Michael Lewis. Information fusion for usar operations based on crowdsourcing. In *Information Fusion (FUSION), 2013 16th International Conference on*, pages 1450–1457. IEEE, 2013.

[148] Chunkai Zhang, Yingyang Chen, Ao Yin, Zhen Qin, Xing Zhang, Keli Zhang, and Zoe L Jiang. An improvement of paa on trend-based approximation for time series. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 248–262. Springer, 2018.

[149] Ying Zhu, Dorin Comaniciu, Martin Pellkofer, and Thorsten Koehler. Reliable detection of overtaking vehicles using robust information fusion. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):401–414, 2006.

[150] Marc-André Zöller and Marco F Huber. Survey on automated machine learning. *arXiv preprint arXiv:1904.12054*, 2019.