SIGNAL 2022

The Seventh International Conference on Advances in Signal, Image and Video Processing

ISBN: 978-1-61208-970-6

May 22nd –26th, 2022

Venice, Italy

SIGNAL 2022 Editors

Pavel Loskot, ZJU-UIUC Institute, China

SIGNAL 2022

Foreword

The Seventh International Conference on Advances in Signal, Image and Video Processing (SIGNAL 2022), held between May 22 – 26, 2022, continued the inaugural event considering the challenges mentioned above. Having these motivations in mind, the goal of this conference was to bring together researchers and industry and form a forum for fruitful discussions, networking, and ideas.

Signal, video and image processing constitutes the basis of communications systems. With the proliferation of portable/implantable devices, embedded signal processing became widely used, despite that most of the common users are not aware of this issue. New signal, image and video processing algorithms and methods, in the context of a growing-wide range of domains (communications, medicine, finance, education, etc.) have been proposed, developed and deployed. Moreover, since the implementation platforms experience an exponential growth in terms of their performance, many signal processing techniques are reconsidered and adapted in the framework of new applications. Having these motivations in mind, the goal of this conference was to bring together researchers and industry and form a forum for fruitful discussions, networking, and ideas.

We take here the opportunity to warmly thank all the members of the SIGNAL 2022 Technical Program Committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to SIGNAL 2022. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the SIGNAL 2022 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that SIGNAL 2022 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of signal processing.

We are convinced that the participants found the event useful and communications very open. We also hope that Venice provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city

SIGNAL 2022 Chairs:

SIGNAL 2022 Steering Committee
Wilfried Uhring, Université de Strasbourg, France
Jérôme Gilles, San Diego State University, USA
Constantin Paleologu, Polytechnic University of Bucharest, Romania
Sergey Y. Yurish, Excelera, S. L. | IFSA, Spain
Pavel Loskot, ZJU-UIUC Institute, China

SIGNAL 2022 Publicity Chairs
Hannah Russell, Universitat Politècnica de València (UPV), Spain
Mar Parra, Universitat Politecnica de Valencia, Spain

SIGNAL 2022

Committee

SIGNAL 2022 Steering Committee

Wilfried Uhring, Université de Strasbourg, France
Jérôme Gilles, San Diego State University, USA
Constantin Paleologu, Polytechnic University of Bucharest, Romania
Sergey Y. Yurish, Excelera, S. L. | IFSA, Spain
Pavel Loskot, ZJU-UIUC Institute, China

SIGNAL 2022 Publicity Chairs

Hannah Russell, Universitat Politècnica de València (UPV), Spain
Mar Parra, Universitat Politecnica de Valencia, Spain

SIGNAL 2022 Technical Program Committee

Waleed H. Abdulla, The University of Auckland, New Zealand
Ahmed Al Hilli, Technical College of Najaf | Al-furat Al-Awsat Technical University, Iraq
Kiril Alexiev, Institute for Information and Communication Technologies -Bulgarian Academy of Sciences, Bulgaria
Djamila Aouada, SnT | University of Luxembourg, Luxembourg
Nadia Baaziz, Université du Québec en Outaouais, Canada
Junaid Baber, University of Balochistan, Pakistan
Vesh Raj Sharma Banjade, Intel Coporation, USA
Joan Bas, CTTC, Spain
Wassim Ben Chikha, Tunisia Polytechnic School, Tunisia
Amel Benazza-Benyahia, SUP'COM | COSIM lab. | University of Carthage, Tunisia
Anirban Bhowmick, Vellore Institute of Technology | Bhopal University, India
Lionel Bombrun, IMS - University of Bordeaux, France
Larbi Boubchir, LIASD - University of Paris 8, France
Moez Bouchouicha, LIS - Laboratoire d'Informatique et Systèmes | Toulon University, France
Salah Bourennane, Ecole Centrale de Marseille, France
Geraldo Braz, Federal University of Maranhão, Brazil
Paula María Castro Castro, University of A Coruña, Spain
Aniruddha Chandra, National Institute of Technology (NIT), Durgapur, India
M. Girish Chandra, TCS Research & Innovation, India
Zhuyun Chen, South China University of Technology,Guangzhou, China
Doru Florin Chiper, Technical University Gheorghe Asachi of Iasi, Romania
João Dallyson Sousa de Almeida, Federal University of Maranhão, São Luís, Brazil
Natasja M. S. de Groot, Erasmus Medical Center | Technical University Delft, Netherlands
Laura-Maria Dogariu, University Politehnica of Bucharest, Romania
António Dourado, University of Coimbra, Portugal
Konstantinos Drossos, Tampere University, Finland

Table of Contents

Cleaning Outdoor Activity Logs using Deep Learning

Davide Sbetti
Free University of Bozen–Bolzano
Email: davide.sbetti@gmail.com

Sergio Tessaris
Free University of Bozen–Bolzano
Email: tessaris@inf.unibz.it

***Abstract*—Nowadays, position recording personal tracking devices are ubiquitous and used by both athletes and outdoor enthusiasts to track and analyse their activities. These devices rely on Global Navigation Satellite Systems to obtain the position in real time. Although the nominal precisions of the different GNSS are high enough for analysis, there are several environmental factors that affect the precision of such devices. Most of the commercial services providing analysis of outdoor activities use techniques to "clean" the user-uploaded data (tracklogs). Most of these techniques require and exploit the huge amount of data that they collect and analyse, but the resulting logs still manifest outliers and recording errors. In this paper, we present a deep learning based technique to identify part of tracklogs that might be influenced by recording errors, in such a way that can be corrected using standard techniques. Our approach does not require geographical or crowdsourced data, and can be also used on low powered devices.**

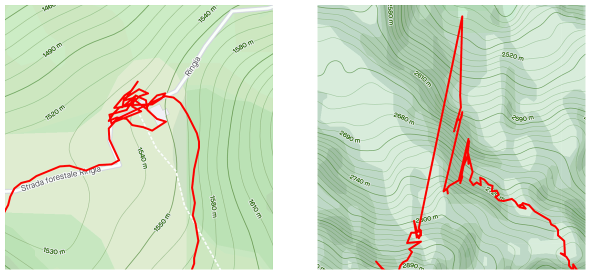
***Index Terms*—Data Cleaning, GPS Traces, Trajectory repairing, recurrent neural networks**

I. Introduction

Location tracking devices based on Global Navigation Satellite Systems (GNSS) are widely used by outdoor enthusiasts and athletes to track their activities for both analysis and social sharing purposes. Although the precision of GNSS-based geolocation is within a few metres under optimal conditions, there are several environmental and receiver-related factors that may introduce substantial errors [1].

Most commercial providers of services related to the analysis and sharing of outdoor activities employ techniques to correct imprecisions in the data provided by users (e.g. [2]). However, the results are not always satisfactory, even by exploiting the large amount of data collected by the big players in the field (see, e.g., Figure 1).

In our work, we focus on two types of error that are often present in recorded activity logs, which are demonstrated in Figure 1. Those are the errors that can be easily identified by users and appear in most outdoor activity recordings. Both these segments are extracted from the web interface of one of the biggest commercial service provider, after any correction that might have been applied to the raw data. The first segment shows a pause that has not been detected by the recording device (some devices feature pause detection algorithms, but often they do not provide a reliable outcome), while the second shows that some recorded points do not reflect the actual position of the receiver. Note that the nature of the two types of errors are different, since the behaviour of the first case is due to the intrinsic imprecision of the position pinpointing while the second is mostly due to environmental conditions, as rock formations. It might be argued that part of the tracking of the first case is due to inevitable small movements of the receiver; however, even in this case, it would be desirable to remove these segments from the recording.



(a) (b)

Fig. 1. Examples of recording errors

In our research, we explore the use of Deep Learning [3] (DL) techniques to improve the quality of recorded activities without resorting to big data techniques (e.g., heat maps) or background knowledge (e.g., network of roads and paths). The reason is twofold, from one side we would like to be able to apply our technique at the level of edge computing also on low powered devices; on the other hand, there are several outdoor activities that are not constrained by the network of paths or roads (e.g., ski touring or kayaking). We are also interested to develop techniques not relying on the behaviour of specific receivers or activities. In this way, it could be applied to data coming from different sources, or even when the information on the recording device or activity type has been stripped from the data.

The contributions of this work are as follows.

- Collection of an annotated dataset of outdoor activities for data repair.
- Evaluation of different DL architectures for GNSS data classification.
- Development and evaluation of a DL model for classifying GNSS receiver errors in activity logs.
- Evaluation and development of algorithms for repairing activity logs.

The next section will introduce the main concepts and related works, because of space restrictions we assume that the reader is familiar with Deep Learning techniques and architectures (otherwise the reader is referred to the relevant bibliographic entries). Section III describes the development and evaluation of the classifier, while Section IV describes the repair techniques. A more detailed description of this work is available in the M.Sc. thesis [4].

II. Backgrounds and Related Work

Modern tracking devices use combinations of different GNSS to identify the current location; moreover, some devices integrate data from other sensors – e.g., barometer, gyroscope, compass – to compensate for possible reception errors. In this paper, we focus on the *Global Positioning System* (GPS), but similar considerations apply also to the other systems [5].

GPS uses 24 satellites on different orbits to enable the reception of at least four of them, regardless of the location of the receiver. In addition, there are control stations that precisely monitor the position of each satellite and share this information using a common registry. To calculate its position, the receiver listens to the electromagnetic waves of four different satellites. It then monitors the time taken by the signal to reach the device starting from the satellites and, since it knows the exact position of each satellite, is able to calculate its position with respect to them. Four different satellites are required to measure the three different coordinates (latitude, longitude, and altitude), while the fourth signal is used to synchronise the internal clock of the device [6].

According to a study conducted in 2015 [7], the positioning error of a smartphone using GPS is within $5m$ when there are no obstacles. However, the GPS signal can be influenced by the surrounding environment. In particular, the signal can be reflected by some surfaces resulting in what is known as multipath reception [1], depicted in Figure 2.

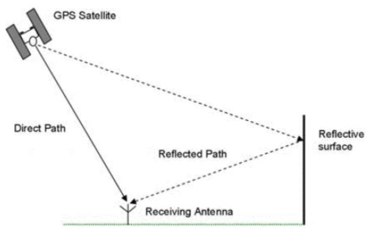


Fig. 2. Multipath effect. Reproduced from [1]

The reflected signal takes longer than the correct direct path, introducing a delay that could result in a potential error in the calculation of the position. Moreover, given that the satellites are constantly moving, the effect of multipath at a certain location changes over time, depending on the position of the satellites. Experiments have shown that the multipath effect can introduce an additional average of $8m$ error in the derived position [1]. This effect may occur in nature due to, for example, mountains, resulting in what is widely known as *canyon effect*. This effect is also occurring in urban contexts, where tall buildings substitute rock walls (urban canyons) [8].

Figure 3 shows the effects of the rocky environment on a recorded activity.

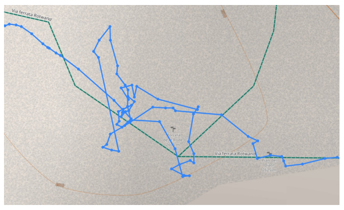


Fig. 3. Canyon effect on an activity log recorded during a via ferrata

In this paper, we use the term *tracklog* to denote a sequential record of geographic coordinates with associated timestamps (the *location points* or simply *points*) collected by a GNSS receiver. Timestamps are essential for most activity analysis tasks, and tracklogs are also referred as GNSS *traces* or *trajectories* in the literature. A *tracklog repair* is an editing of the sequence by removing or changing the geographic coordinates of some points. We assume the correctness of the timestamps; in particular, the relevant detail is the time interval between timestamps.

The task of cleaning GPS data, and as a more general case time series, has been widely studied in the literature. We identified six groups of related works based on the techniques they employ.

Smoothing-based techniques Moving average approaches or autoregressive algorithms are often used to smooth time series [9]. Kalman filters [9], which combine the observation and a prediction, based on the relation of the various components with external factors, were also applied, updating the external influence [10] or weighting the observations using the variance [11]. Smoothing techniques generally modify the entire time series, which may also lead to adjusting the correct points.

Data-driven techniques In [12], past observations are used to extract the main routes and areas in a region to correct the points falling into them. Predetermined trajectories, known a priori, were used to correct GPS readings [13]. Furthermore, in [14] the authors employed the detection of outlying subtrajectories, considering the distance of their characteristics. Data driven techniques heavily rely on the regional availability of data, generally lacking generalisation.

Constraint-based techniques Ordered, or sequential, relations are also often employed; such as speed constraints to identify outliers [9]. However, their definition can be problematic for dynamic activities.

Statistics-based techniques Markov Models have been used to predict and then compare observations as a cleaning strategy [9]. Dynamic probabilistic models, such as STPM, were used to calculate the conditional probabilities of attributes, applying a threshold based cleaning [9]. Moreover, in [15], using available data, the corrected sequence is calculated as close as possible to the original using the largest likelihood in terms of speed change between consecutive points.

Anomaly detection techniques In [16], the authors partitioned the original trajectories using significant direction or speed changes, and then they apply a consistency model to the partitions in order to identify anomalies. Thresholds have also been employed, for example in [17], to the reachability speed between points to remove outliers, then applying the underlying geographic information to adjust their positions, or in [8] using the altitude and distance between consecutive points. Smoothing techniques, e.g., a Gaussian kernel, are applied to the points. However, identifying the right thresholds can be challenging for dynamic activities.

Machine learning techniques Clustering algorithms were used to merge directly reachable clusters of points [9], using DL models adopted to perform anomaly detection [9]. In [18], decision trees and the SVM were combined to classify the GPS points into different categories. In [19], a neural network is used to predict the destination of taxi trips among selected destinations, which are discovered applying a mean shift algorithm on the training trajectories. Furthermore, in [20] a regression model, trained on points collected by smartphones, is employed to identify a radius used to restrict candidates for the correct position within a circle centred on the point to be corrected.

III. Building and Training the Classifier

The first problem we faced is the lack of (annotated) datasets to develop and train a classification model. Therefore, we built a publicly accessible web application that allows anonymous upload and annotation of logs. The description of the application is outside the scope of this paper, but its source is available on [21].

The annotation app enables graphical annotation by selecting (ranges of) points and assigning one of the available classes: *pause*, *outlier*, *correct*. We decided to include the *correct* class in order to mitigate the fact that the majority of points are not annotated, so the dataset would be heavily unbalanced if we were considering all the not annotated points as correct. The *correct* class enables the annotator to specify points that they consider to be placed correctly. The annotator has the possibility of including additional metadata, such as the type of activity and the recording device, but this is not required.

Over 5 months, we collected 61 logs distributed among 7 types of activities, and most of them were hiking, walking, and ski mountaineering (they correspond to 80% of the logs). The number of distinct logs is important to ensure diversity, but the dataset should be understood in terms of location points and annotations. The total number of points is $232,238$ with $13,514$ (0.06%) annotations; of those, the majority – more than 77% – are marked as correct, followed by pauses (15%) and outliers (8%). The majority of the collected logs are located in the Trentino - South Tyrol Italian region, while more in general we could observe how all of them were in the northern Italy.

The fact that a point is identified as not correct does not depend on its location and timestamp, but on the variation w.r.t. preceding and following points. In fact, several techniques in the literature just rely on identifying wrong points on the basis of their variation w.r.t. neighbours (see, e.g., [16], [17]). Because of this, we normalise the dataset, transforming each point into a tuple representing the variation of location and time from the previous point (the *deltas*). Each delta represents the variation of the coordinates on the three spatial axes and the time elapsed between each pair of consecutive points (Figure 4).

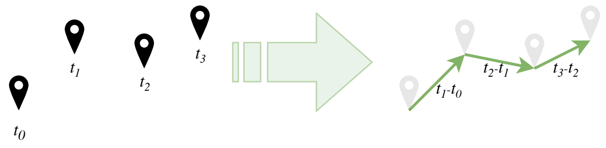


Fig. 4. Location points to deltas.

To simplify the computations of deltas, we convert the original location points encoded in longitude, latitude, and altitude into the Earth-Centred Earth-Fixed Coordinate System (ECEF) [22]. The latter being a Cartesian system, the deltas are just the difference between values of the corresponding axes, and their values do not depend on the actual geodetic position. Moreover, given the initial location and timestamp, the original sequence of geodetic coordinates can be reconstructed without data loss. In the rest of the paper, we will use the term points to refer to the deltas rather than the geographical coordinates.

Before committing to a specific deep learning architecture, we performed a set of preliminary experiments in order to identify the one that would be best suited to identify properties of GNSS logs. Our hypothesis was that Recurrent Neural Networks are well suited for this task, being widely adopted for *sequence labelling*, in particular focusing on *Long Short Term Memory* (LSTM) networks [23]. However, we decided to compare it with two other different architectures used to classify and repair GNSS data: Multi-Layer Perceptron [24], and Convolutional [25]. Since we were still collecting our data, we decided to evaluate these architectures on a simple activity recognition task (that is, identifying the type of sport activity from the recorded task) using 150 publicly available tracklogs of walking, hiking, and running (downloaded from *Wikiloc* and *Garmin Connect* sites with the consent of users). Our preliminary experiments confirmed that LSTM networks provided the best performance among the selected architectures.

In [26] different variations of the LSTM architecture have been evaluated and shown that “forget gate and the output activation function to be its most critical components”, reinforcing our choice of adopting recurrent neural networks based on this cell type, considering also how some possible variations, tested in [26], did not show particular advantages in terms of accuracy. To select the right network topology and hyper-parameters we compared a common *vanilla LSTM* as baseline, with a three-layers LSTM architecture used in [27] to evaluate driving style, and our proposed architecture coupling three LSTM layers with dropout layers (see Figure 5). In our proposed architecture, the first (bidirectional) LSTM layers outline the nonlinear temporal relations between the data; while the remaining layers gradually reduce the number of nodes before the target output to avoid overfitting.

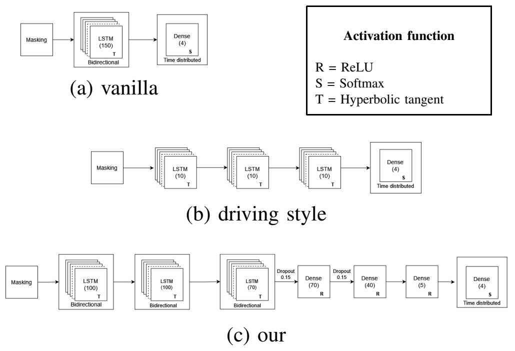


Fig. 5. The evaluated LSTM architectures

For all selected LSTM architectures, the input layer is a sequence of contiguous points (4-tuples) of a given fixed size (the *window*), while the output layer is composed of a sequence (the same size as the input) with an array of units, each corresponding to one of the classification classes, activated using softmax. Each array of units provide the classification of the corresponding input point.

To classify the points of a tracklog, we employ a *sliding window* approach where sequences are created by moving a fixed size window over the original tracklog, by a given sliding step (Fig. 6).

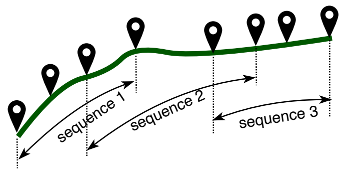


Fig. 6. Sliding window approach to segmenting (size 4, step 2)

When the classifier is used on a given tracklog, each point is classified multiple times (depending on the size of the window and sliding step), for selecting the class we decided to use a majority selection. This is used in the evaluation for training the network and in its deployment for cleaning new logs.

To select the best parameters we focused on the window size (3, 5, 10, 15), step size (1, 2, 3, 5), and number of training epochs (10, 20, 30, 40, 50). We performed a grid search on those combinations using a standard K-fold technique (3 folds). For the grid search, we used a subset of the final dataset (66% of the tracklogs), and the dataset was composed of all the sequences generated by the sliding-window approach over the (padded) annotated points. We ensured that training and testing folds did not share common points due to intersecting windows. Table I shows the best three combination across the different architectures. The *driving* architecture does not appear in the table because the best accuracy obtained among the different combinations of parameters was 0.83.

We performed a final evaluation of the selected architecture and parameters using the same k-fold procedure. The results are summarised using the confusion matrices of the folds in Figure 7. Finally, we trained the network on the entire dataset, generating the model to be deployed to classify the points in tracklogs.

TABLE I
TOP THREE COMBINATIONS

Epochs	Window	Step	Accuracy	LSTM Architecture
30	15	2	0.864243	our
40	15	2	0.860122	vanilla
10	15	2	0.859688	vanilla

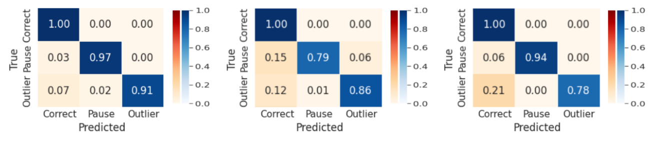


Fig. 7. Confusion matrices for model evaluation

Our goal is to be able to deploy the data repair on low-powered devices, so it is paramount to maximise the efficiency and minimise the resource consumption of the classifier. The standard *TensorFlow* library and models are not suitable for edge computing; however, *TensorFlow Lite* [28] is tailored for deployment on mobile, embedded, and IoT devices. Standard TensorFlow models cannot be used with the "lite" version of the library and need to be converted. Since the original model is not very large, we decided to convert it without any further optimisation; therefore, the properties of the "lite" model are the same as the original. This has been confirmed by comparing the results of the original and "lite" models over the entire training data set.

The code used for our experiments is available in [29].

IV. REPAIRING TRACKLOGS

The process we envisage for cleaning the activity logs is composed by two stages: in the first, location points are classified using the trained model; after, one of the various techniques presented in the literature can be applied to the points identified as non-correct.

As introduced in Section III the classification of points in a tracklog is performed by first converting them into a sequence *deltas* and then classifying them by majority vote using a sliding window to obtain multiple classifications for each point.

After the classification process, the correction focuses on the points assigned to either *pause* or *outlier* classes. The first kind of points are easy to deal with, since they represent a situation in which the device should have been stationary, so they should be all referring to the same position. Therefore, a sequence of "pause" points can be removed or replaced with a single point by averaging their data. In fact, there might be actual small movements, but they are irrelevant, or even misleading, for activity analysis purposes.

To correct "outliers" we considered different techniques applied in the relevant literature. Most techniques that do not exploit crowd-sourced data are based on the idea of "smoothing" segments in which points diverge w.r.t. a set of predefined parameters. The smoothing can be performed using linear or splines [30] interpolation. *Kalman filters* [31] have also been used to improve the quality of GNSS data [9]. This family of transformations join together the observed value and the predicted one to obtain the final observation.

We compared linear and spline interpolation (Figure 8) with Kalman filters (Fig 9) and we observed that the latter provided the best results. Note that the quality of the result is based on the visual inspection of the corrected tracklog; we reckon that more objective measures should be investigated, but to the best of our knowledge, there is no consensus in the research community on how to evaluate the quality of a tracklog.

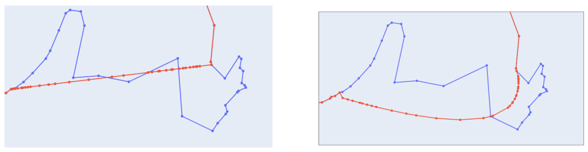

(a) linear (b) spline

Fig. 8. Example of interpolation (in red the changes)

There are different ways to apply the filter; we excluded the common approach of applying it to all data points because our classifier enables us to focus on only the points that are considered "outliers". We apply filtering only to the segments that include "outliers". We also noticed that the application of the filter along the direction of the timestamps introduces an unnatural joining of the corrected segment with the rest of the tracklog due to an "inertial" effect of the filter (it tends to maintain the current direction of the movement). To mitigate this effect, we applied the filter in both forward and backward directions, averaging their changes to the points (Figure 9).

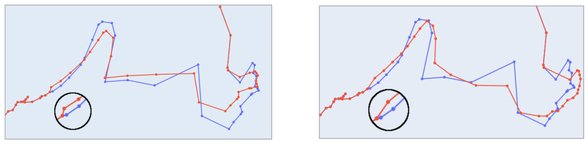

(a) unidirectional (b) bidirectional

Fig. 9. Example of Kalman filter (in red the changes)

The code we used for our repair experiments is available in [32].

V. Conclusions

In this paper, we described a Deep Learning approach to identify parts of the recorded activity logs that could be affected by GNSS receiver errors. We identified two types of error, namely those deriving from the so-called "canyon effect" (outliers) and those deriving from a stationary receiver (pauses). Moreover, we show how different techniques can be applied to modify the identified errors, and suggest the ones that provide good quality results. Note that the modularity of our approach enables the use of different techniques and/or preferences (e.g., pause segments might be left as they are).

Our empirical evaluation shows that an LSTM architecture is well suited to identify points affected by receiver errors, and the classifier can also be used on low-powered (edge) devices. Our work is also showing that the identification of errors can also be performed without a prior knowledge about the type of device and activity. The training process does not require a large amount of computational resources, so individual users could adapt the model to their kind of specific activities or devices.

Moreover, we developed a web application to enable the collection and annotation of tracklogs, and used it to create a dataset for our experiments. Clearly, the quality of the dataset is paramount to ensure the accuracy of the predictions. In the future we plan to extend the dataset, in particular w.r.t. the geographical area.

Another problem we identified is the evaluation of the quality of tracklog repairs. In this work, we adopted a subjective approach based on the appearance of the resulting tracklog, but we think that identifying a proper quality measure is an open and relevant problem which does not seem to be addressed in any of the works we reviewed. An approach could be to collect a "golden standard" dataset in a controlled environment; but it is difficult to cover the variety of outdoor activities and the environments in which they occur.

References

[1] T. Kos, I. Markezic, and J. Pokrajcic, "Effects of multipath reception on GPS positioning performance," in *Proceedings ELMAR-2010*, Sep. 2010, pp. 399–402, iSSN: 1334-2630.

[2] "Elevation on Strava FAQs," Mar 2022, [Online; accessed 30. Mar. 2022]. [Online]. Available: https://support.strava.com/hc/en-us/articles/115001294564-Elevation-on-Strava-FAQs

[3] S. Skansi, *Introduction to Deep Learning*. Cham, Switzerland: Springer, 2018.

[4] D. Sbetti, "Cleaning GPS trajectories using machine learning techniques," Master's thesis, Free University of Bozen-Bolzano – Computer Science Department, Bolzano, Italy, 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6369847

[5] X. Li, M. Ge, X. Dai, X. Ren, M. Fritsche, J. Wickert, and H. Schuh, "Accuracy and reliability of multi-GNSS real-time precise positioning: GPS, GLONASS, BeiDou, and Galileo," *Journal of Geodesy*, vol. 89, no. 6, pp. 607–635, Jun. 2015. [Online]. Available: https://doi.org/10.1007/s00190-015-0802-8

[6] N. Crato, "How GPS Works," in *Figuring It Out: Entertaining Encounters with Everyday Math*, N. Crato, Ed. Berlin, Heidelberg: Springer, 2010, pp. 49–52. [Online]. Available: https://doi.org/10.1007/978-3-642-04833-3_12

[7] F. van Diggelen and P. Enge, "The World's first GPS MOOC and Worldwide Laboratory using Smartphones," in *Proceedings of the 28th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2015)*, Sep. 2015, pp. 361–369, iSSN: 2331-5954. [Online]. Available: http://www.ion.org/publications/abstract.cfm?jp=p&articleID=13079

[8] N. Schuessler and K. W. Axhausen, "Processing Raw Data from Global Positioning Systems without Additional Information," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2105, no. 1, pp. 28–36, Jan. 2009. [Online]. Available: http://journals.sagepub.com/doi/10.3141/2105-04

[9] X. Wang and C. Wang, "Time Series Data Cleaning: A Survey," *IEEE Access*, vol. 8, pp. 1866–1881, 2020.

[10] A. H. Mohamed and K. P. Schwarz, "Adaptive Kalman Filtering for INS/GPS," *Journal of Geodesy*, vol. 73, no. 4, pp. 193–203, May 1999. [Online]. Available: http://link.springer.com/10.1007/s001900050236

[11] S. Shokri, N. Rahemi, and M. R. Mosavi, “Improving GPS positioning accuracy using weighted Kalman Filter and variance estimation methods,” *CEAS Aeronautical Journal*, vol. 11, no. 2, pp. 515–527, Jun. 2020. [Online]. Available: http://link.springer.com/10.1007/s13272-019-00433-x

[12] L. Li, X. Chen, Q. Liu, and Z. Bao, “A Data-Driven Approach for GPS Trajectory Data Cleaning,” in *Database Systems for Advanced Applications*, Y. Nah, B. Cui, S.-W. Lee, J. X. Yu, Y.-S. Moon, and S. E. Whang, Eds. Cham: Springer International Publishing, 2020, vol. 12112, pp. 3–19, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-030-59410-7_1

[13] R. Milton and A. Steed, “Correcting GPS Readings from a Tracked Mobile Sensor,” in *Location- and Context-Awareness*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, T. Strang, and C. Linnhoff-Popien, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3479, pp. 83–94, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/11426646_8

[14] J.-G. Lee, J. Han, and X. Li, “Trajectory Outlier Detection: A Partition-and-Detect Framework,” in *2008 IEEE 24th International Conference on Data Engineering*. Cancun, Mexico: IEEE, apr 2008, pp. 140–149. [Online]. Available: http://ieeexplore.ieee.org/document/4497422/

[15] A. Zhang, S. Song, and J. Wang, “Sequential Data Cleaning: A Statistical Approach,” in *Proceedings of the 2016 International Conference on Management of Data*. San Francisco California USA: ACM, jun 2016, pp. 909–924. [Online]. Available: https://dl.acm.org/doi/10.1145/2882903.2915233

[16] X. Yang, L. Tang, X. Zhang, and Q. Li, “A Data Cleaning Method for Big Trace Data Using Movement Consistency,” *Sensors*, vol. 18, no. 3, p. 824, Mar. 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/3/824

[17] A. Hendawi, S. S. Sabbineni, J. Shen, Y. Song, P. Cao, Z. Zhang, J. Krumm, and M. Ali, “An Interactive Map-based System for Visually Exploring and Cleaning GPS Traces,” in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. Chicago IL USA: ACM, Nov. 2019, pp. 572–575. [Online]. Available: https://dl.acm.org/doi/10.1145/3347146.3359105

[18] J. Zhang and Y. Sun, “An Automatic Data Cleaning Method for GPS Trajectory Data on Didi Chuxing GAIA Open Dataset Using Machine Learning Algorithms,” in *2019 6th International Conference on Systems and Informatics (ICSAI)*. Shanghai, China: IEEE, Nov. 2019, pp. 1522–1526. [Online]. Available: https://ieeexplore.ieee.org/document/9010403/

[19] A. De Brébisson, E. Simon, A. Auvolat, P. Vincent, and Y. Bengio, “Artificial neural networks applied to taxi destination prediction,” in *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge - Volume 1526*, ser. ECMLPKDDDC’15. Aachen, DEU: CEUR-WS.org, Sep. 2015, pp. 40–51.

[20] P. Zhao, A. Zhang, C. Zhang, J. Li, Q. Zhao, and W. Rao, “ATR: Automatic Trajectory Repairing With Movement Tendencies,” *IEEE Access*, vol. 8, pp. 4122–4132, 2020, conference Name: IEEE Access.

[21] D. Sbetti and S. Tessaris, “Track annotation app,” Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5977242

[22] I. Skog and P. Handel, “In-car positioning and navigation technologies - a survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 4–21, 2009.

[23] A. Graves, *Supervised sequence labelling with recurrent neural networks*, ser. Studies in computational intelligence. Berlin New York: Springer, 2012, no. v. 385.

[24] S. R. Bashir and V. Misic, “Detecting Fake Points of Interest from Location Data,” in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, Dec 2021, pp. 5347–5356.

[25] S. Dabiri, N. Marković, K. Heaslip, and C. K. Reddy, “A deep convolutional neural network based approach for vehicle classification using large-scale GPS trajectory data,” *Transportation Research Part C: Emerging Technologies*, vol. 116, p. 102644, Jul 2020.

[26] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A Search Space Odyssey,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[27] S. Wrtz and U. Ghner, “Driving Style Analysis Using Recurrent Neural Networks with LSTM Cells,” *Journal of Advances in Information Technology*, vol. 11, p. 1, Jan. 2020.

[28] “TensorFlow Lite | ML for Mobile and Edge Devices,” Dec 2021, [Online; accessed 4. Feb. 2022]. [Online]. Available: https://www.tensorflow.org/lite

[29] D. Sbetti and S. Tessaris, “Gps tracks: transform and model,” Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5978571

[30] S. Dyer and J. Dyer, “Cubic-spline interpolation. 1,” *IEEE Instrumentation Measurement Magazine*, vol. 4, no. 1, pp. 44–46, Mar. 2001, conference Name: IEEE Instrumentation Measurement Magazine.

[31] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[32] D. Sbetti and S. Tessaris, “Gpsclean,” Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5978000

STP-Net: Semi-Tensor Product Neural Network for Image Compressive Sensing

Youhao Yu[1,2], Richard M. Dansereau[1]

Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada[1]
School of Information Engineering, Putian University, Fujian, China[2]
e-mail: youhaoyu@sce.carleton.ca e-mail: rdanse@sce.carleton.ca

***Abstract*—Semi-tensor product (STP) is developed into a neural network in this paper and applied to image compressive sensing (CS). Large matrix computation for fully connected layers results in a large number of weight coefficients that need long training times. Instead of using an *M*×*N* measurement matrix, according to the theory of STP a smaller measurement matrix of size *M/t*×*N/t* can be applied, where *t* is a shrinkage factor. STP only needs *N/t* elements of the original signal for one measurement and the measurement matrix is shrunk to 1/*t*² that of traditional CS. The shrinkage factor *t* is adjustable. To demonstrate the effectiveness of the STP-based neural network, we apply it to image reconstruction. The goal is to sample and recover larger images, without partitioning into smaller blocks that introduces block artifacts, and provide good initial reconstruction for subsequent networks.**

Keywords-compressive sensing; convolutional neural network; semi-tensor product; image reconstruction.

I. INTRODUCTION

Compressive sensing (CS) has become a significant research field in analog information processing, image compression, machine learning and so on [1]–[3]. The fundamental issue in CS is to reconstruct a sparse signal from few measurements. Since natural images are intrinsically sparse in some domain, they can be restored efficiently from CS measurements. Sparse signal reconstruction is an inverse problem that can be solved by techniques like basis pursuit (BP) [4], matching pursuit (MP) [5], orthogonal matching pursuit (OMP) [6], approximate message passing (AMP) [7], and least absolute shrinkage and selection operator (LASSO) [8], to name a few, but these tend to be time-consuming.

CS acquires signals that are sparse in a certain basis in a compressed form. The sparsifying basis and the measurement matrix should be incoherent [3]. According to CS theory, a high dimension sparse signal x is sampled by a measurement matrix Φ resulting in a low dimension measurement y as

$$y = \Phi x \tag{1}$$

where x is an $N\times 1$ vector, y is an $M\times 1$ vector and Φ is an $M\times N$ matrix ($M<<N$). The measurement matrix should satisfy the restricted isometry property (RIP) [9].

Each measurement y_i is the linear combination of the elements in x through a row of Φ as

$$y_i = \phi_{i,1}x_1 + \cdots + \phi_{i,j}x_j + \cdots + \phi_{i,N}x_N \tag{2}$$

where x_j, y_i, and $\phi_{i,j}$ are elements of x, y, and Φ. In (2), all N elements in x are used to obtain one measurement y_i, which causes large computational cost when x is long since the measurement matrix Φ will be large.

There are numerous data reconstruction approaches for compressive sensing. In [10][11], STP is adopted and an iterative optimization approach used for image CS reconstruction. While the approach in [10][11] produces good results at a high measurement rate, it is time consuming, needs many iterations, and a wavelet transform is used before CS and an inverse wavelet transform after CS reconstruction.

In this paper, we propose developing the Semi-Tensor product into a neural network (NN). Such a network uses fewer parameters to train and provides theoretical foundation for a layer to be designed. The proposed NN needs fewer layers and less training time for efficient measurement and a good initial reconstruction compared to others, performing better than other full convolutional NN systems. Given the efficiency, the developed NN is used for whole image CS reconstruction using no block partitioning.

The rest of this paper is organized as follows. Section II is an overview of previously proposed CS reconstruction methods. Section III describes the measurement and the initial reconstruction of CS based on STP. Section IV gives a detailed description of STP-Net. Our experiments are set up to demonstrate the outstanding performance of STP-Net in Section V. Finally, we conclude with a discussion of our findings in Section VI.

II. RELATED WORK

Many NN algorithms have been studied for CS measurement reconstruction, such as [12]–[18]. Among them, [12] develops a good framework for sensing and recovering structured signals, but a few full connection layers make it less efficient. [13] and [14] use convolutional layers or residual blocks to refine the initial reconstruction of every image block. After that, they use block-matching and 3-D filtering (BM3D) [15] to remove block artifacts, which is an image denoising strategy based on an enhanced sparse representation in the transform domain. [16][17] give novel methods that measure an image using convolutional layers. However, our experiments show that when the sampling rate is 1% the reconstructed image is affected by block artifacts, especially at the edges of the image, even with a residual network (ResNet) after initial reconstruction as they did to improve performance. [18] introduces an interpretable optimization-inspired deep network for image compressive sensing. They cast the iterative shrinkage-threshold algorithm (ISTA) into a deep network that produces good performance.

To overcome the limitation of GPU memory, existing methods usually divide an image into blocks and vectorize the blocks before measurement. Then, the image is reconstructed block-by-block, making block artifacts inevitable [16] and requiring denoising to remove block artifacts.

Our method is different from others since we process an image as a whole rather than block-by-block, which avoids the block artifacts because the structure information of an image is preserved. Normally, operating on the full image would be computationally costly but the proposed STP-Net helps reduce that computational cost.

III. STP Approach for CS

STP approach can be applied for CS measurement and its initial reconstruction of 1D and 2D signal.

A. *Measurement*

According to the theory of Semi-Tensor product [19], a smaller measurement matrix $\Phi(t)$ can be obtained with dimensions $M/t \times N/t$. Here, *t* is a shrinkage factor which is a common divisor of M and *N*. ($M, N, t, M/t$ and N/t are all positive integers) [11]. Only N/t elements of the signal are needed for one measurement. The dimension of the measurement matrix is shrunk by $1/t^2$ of that for traditional CS. Using the left product operator $\ltimes$ for STP, (1) is rewritten as

$$y = \Phi(t) \ltimes x. \quad (3)$$

For clarity, the signal x and its measurements y can be segmented into groups and every group only has t points. So, x is divided into N/t fragments and y is divided into M/t fragments. Reshaping the vectors $x \in \mathbb{R}^{N\times 1}$ and $y \in \mathbb{R}^{M\times 1}$ into matrix form $X \in \mathbb{R}^{t\times \frac{N}{t}}$ and $Y \in \mathbb{R}^{t\times \frac{M}{t}}$, we rewrite (3) to maintain column-wise order as

$$Y = X \cdot \Phi(t)^T. \quad (4)$$

Considering X as an image, this means an image can be sampled directly by matrix multiplication.

For our work, t is set to the number of rows the image. Our method takes an image as a whole without dividing it into blocks or vectorizing the image.

B. *Initial Reconstruction*

Typically, the rows of the measurement matrix Φ are chosen to be orthonormal and the least-squares solution (minimum energy reconstruction) as the initial estimate for *x* [20]. Let Φ be an $M\times N$ matrix and let *y* be a vector in R^M. The least-squares solution of $\Phi x = y$ is the solution of $\Phi^T\Phi x = \Phi^T y$ [21]. If Φ has orthonormal rows, $\Phi\Phi^T$ is an $M\times M$ identity matrix. Using $\Phi^T y$ is a common initial reconstruction.

STP maintains the above-mentioned property [11]. If $\Phi(t)$ is a matrix with orthonormal rows and $y = \Phi(t) \ltimes x$, the least-squares solution for *x* is the solution of $[\Phi(t)]^T \ltimes y = [\Phi(t)]^T \ltimes \Phi(t) \ltimes x$. Since $\Phi(t) \ltimes [\Phi(t)]^T = \Phi(t)[\Phi(t)]^T = I$, $\tilde{x} = [\Phi(t)]^T \ltimes y$ is an initial estimate. The result can be written in matrix form as $\tilde{X} = Y \cdot \Phi(t)$.

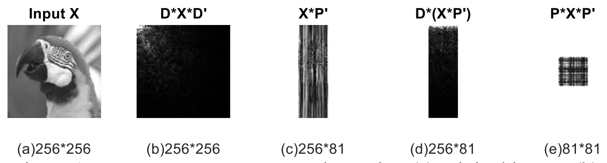


Figure 1. Image measurements and sparsity: (a) original image, (b) 1D sparsifying basis *D* applied along rows and columns, (c) measurement of (a) using random matrix *P*, (d) result showing measurement is still sparse, and (e) image sampled along rows and columns. Sampling rate = $(81/256)^2 \approx 10\%$.

C. *2D Compressive Sensing*

According to compressive sensing theory [1]–[3], a signal that is sparse in a certain domain can be sampled at a rate less than the Nyquist sampling rate. In Fig. 1, (a) is an image with size 256×256. Usually, a natural image is sparse in the frequency domain, as in Fig. 1(b) where a 2D discrete cosine transform (DCT) has been applied using matrix *D*. Most energy is concentrated on top-left corner. If measurement matrix $\Phi(t) = P$, where *P* is a random matrix, then the image measured by (4) produces Fig. 1(c) with smaller size.

Since t is set to the number of rows, the measurement matrix has effectively measured each column separately. As shown in Fig. 1(d), the measurement results are still sparse with the right choice of sparsifying basis *D*; most energy concentrates in the top area. Following CS theory, the measurements can be sampled and compressed again using

$$Y = \Phi_1(t) \cdot X \cdot \Phi_2(t)^T. \quad (5)$$

In Fig. 1, since the image is square, we suppose $\Phi_1(t) = \Phi_2(t) = P$, where P is a random matrix. Let us now map image X and its measurements Y into vectors x and y by column ordering, it is equivalent to (1) when $\Phi = \Phi_1(t) \otimes \Phi_2(t)$ where $\otimes$ is the Kronecker product. In this case, the 1D operation of (1) is expressed as the separable 2D operation that reduces the computational complexity [22].

It has been shown that a sparse matrix $\hat{X}$ (i.e, $\hat{X} = DXD^T$) can be recovered from its matrix sketching $Y = \Phi_1 \cdot \hat{X} \cdot \Phi_2^T$ [23][24][25]. Here, we assume X has size $t \times t$ where $t^2 = N$. The dimension of Y is $m \times m$ with $m = M/t$ and $m \ll t$. Φ_1 and Φ_2 are two $m \times t$ matrices that can be seen as measurement matrices. It is equivalent to $y = (\Phi_1 \otimes \Phi_2) \cdot x$. The initial estimate of the image can then be $\tilde{X} = \Phi_1^T \cdot Y \cdot \Phi_2$.

As shown in the dashed box of Fig. 2, an image is effectively measured and compressed twice and its initial reconstruction uses two corresponding steps.

IV. STP-Net: Neural Network Layer

Inspired by the flexibility of STP, we build a neural network layer to implement it. The sampling and initial recovery process of the proposed STP neural network (STP-Net) for compressive sensing can be implemented by building layers of an NN in two ways. One is defining a custom deep learning layer with learnable parameters in forward and backward propagation [26], and the other is by means of ready-made convolutional layers, like in Matlab,

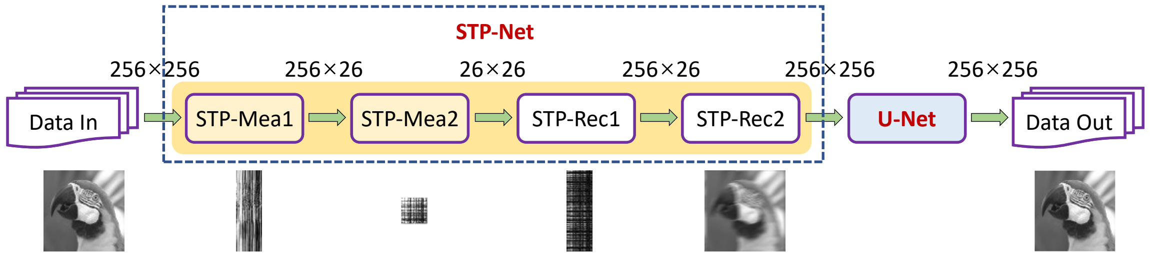


Figure 2. STP-Net connected with U-Net for deblurring (Sampling rate:1%).

TensorFlow, or PyTorch, followed by a custom reshape layer.

A. *Defining a Custom Deep Learning Layer*

A custom STP layer has a forward pass, a backward pass, and learnable parameters including weights and biases. It can be used as a measurement matrix by setting the biases to zero and the learning rates to zero so that the set weights remain static. It also can be used in initial reconstruction through a least-squares like solution.

Reconstruction of a signal is a regression problem, so the output layer is a regression layer. The loss function of the regression layer is the half-mean-square-error (HMSE) [27]

$$L = \frac{1}{2S}\sum_{i=1}^{S}\sum_{j=1}^{R}\frac{(t_{ij}-y_{ij})^2}{R} \quad (6)$$

where S is the number of observations in a mini-batch, R is the number of responses, t_{ij} is the target output, and y_{ij} is the network's prediction for the response variable corresponding to observation i.

For a semi-tensor product layer, the derivative of the loss function with respect to the input data of the custom layer are

$$\frac{\partial L}{\partial x_{i,j}} = \sum_{a=1}^{M/t}\frac{\partial L}{\partial y_{i,a}}\phi_{a,j} \quad (7)$$

where $i = 1,\dots,t,\ a = 1,\dots,M/t$, and $j = 1,\dots,N/t$.

The derivatives of the loss with respect to the weights are

$$\frac{\partial L}{\partial \phi_{a,j}} = \sum_{i=1}^{t}\frac{\partial L}{\partial y_{i,a}}x_{i,j} \quad (8)$$

where $i = 1,\dots,t,\ a = 1,\dots,M/t$, and $j = 1,\dots,N/t$.

For better performance, it could have biases B added to the STP layer. Since $Y = X\cdot\Phi(t)^T + B$, the derivatives of the loss with respect to the biases B have the same size with output Y and the values are

$$\frac{\partial L}{\partial B_{ij}} = \frac{\partial L}{\partial y_{ij}} \quad (9)$$

where B_{ij} and y_{ij} are the elements of B and Y ($i = 1,\dots,t$ and $j = 1,\dots,M/t$).

Obtaining the derivative of the loss with respect to the measurement matrix and the biases, the learnable parameters are updated with

$$\phi_{a,b}^{k} = \phi_{a,b}^{k-1} - \eta\frac{\partial L}{\partial \phi_{a,b}} \quad (10)$$

$$B_{ij}^{k} = B_{ij}^{k-1} - \eta\frac{\partial L}{\partial B_{ij}} \quad (11)$$

where k is the iteration number, η is the learning rate, $a = 1,\dots,M/t,\ b = 1,\dots,N/t,\ i = 1,\dots,t$ and $j = 1,\dots,M/t$.

The initial reconstruction would be $\tilde{X} = Y\cdot\Phi(t) + B$. The traditional compressive sensing measurement paradigm applies fixed linear measurement [12], which is easy to implement in practical applications. Biases are not added with the measurement results, but they are added during initial reconstruction to have better results.

B. *Ready-made Convolutional Layer*

The CS measurement process based on STP is similar to dilated convolution (also known as "à trous" convolution). STP can be implemented by means of a dilated convolutional layer with factor *t* used to increase the receptive field (the area of the input signal which the layer can detect) of the layer without increasing the number of parameters and computation [28]. Since the number of rows of the measurement matrix correspond to the number of neurons in the convolutional layer and every filter produces one channel output, this layer should be followed by a reshaping, which makes the measurements in the same channel.

V. Experiments

In this section, we conduct a series of experiments to test the measurement and reconstruction performance of STP-Net.

A. *Implementation Details*

The experiments are conducted on MATLAB R2019a. The computer is equipped with Intel i7-8700K, GeForce GTX 1080 CPU with frequency of 3.7 GHz and 16 GB RAM. Natural images from the ILSVRC2014 ImageNet dataset are adopted. Here, 20k images are chosen: 14k (70%) for training, 3k (15%) for validation, and 3k (15%) for testing. We extracted the central 256×256 part of each image and converted them to 8-bit grayscale. Training used stochastic gradient descent with momentum, minibatch size of 64, maximum epoch of 40, learning rate of 2e-03, drop factor of 0.10, and drop period of 15. For comparison, we also utilize the widely used benchmark dataset Set11 [13] during testing.

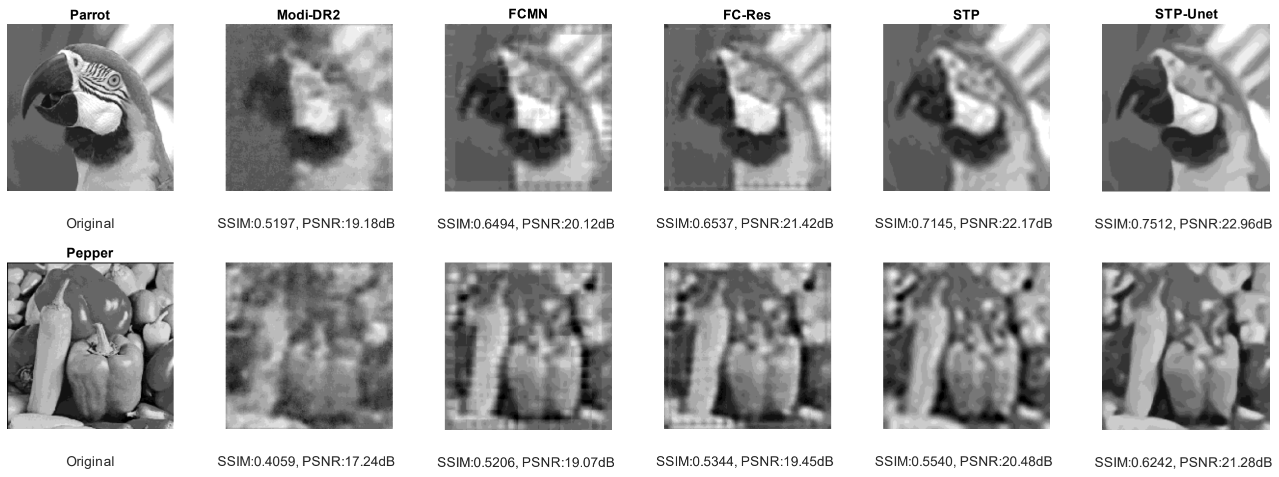

Parrot
Modi-DR2
FCMN
FC-Res
STP
STP-Unet
Original
SSIM:0.5197, PSNR:19.18dB
SSIM:0.6494, PSNR:20.12dB
SSIM:0.6537, PSNR:21.42dB
SSIM:0.7145, PSNR:22.17dB
SSIM:0.7512, PSNR:22.96dB
Pepper
Original
SSIM:0.4059, PSNR:17.24dB
SSIM:0.5206, PSNR:19.07dB
SSIM:0.5344, PSNR:19.45dB
SSIM:0.5540, PSNR:20.48dB
SSIM:0.6242, PSNR:21.28dB

Figure 3. Reconstruction results for parrot and pepper from noiseless CS measurements at measurement rate of 1%. It is evident that STP based method restores more visually appealing images than the competitors.

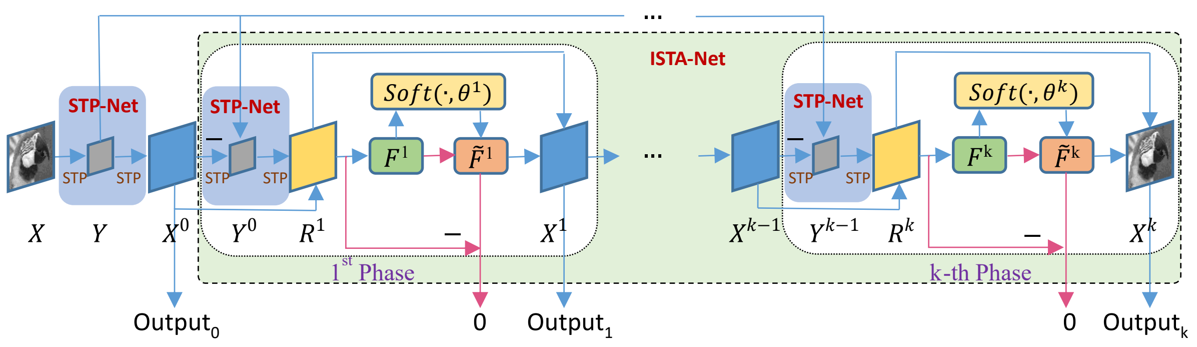


Figure 4. Connect STP-Net with ISTA-Net and use STP layers for measurement and reconstruction in every phase.

B. Measurement Matrix Based on STP

Every image has 65536 pixels. A sampling rate of 1% will produce 655 measurements. The size of such a measurement matrix is 655×65536, which needs more than 300 MB of memory to store the measurement matrix (with double precision matrix elements). According to the STP method, if the parameter t is 256, then a measurement matrix with size equal to *ceiling*(256×1%)×256 can be used that satisfies RIP [9]. It needs 6 kB of memory and is much smaller than the full measurement matrix. Too small of a measurement matrix leads to excessive information loss, so we implement CS measurement in two steps. As mentioned in Section III, we can use two 1D projections for the 2D image, which can be done by applying STP twice. For a target of 1%, one option is that the first pass compresses the signal by 10% and the second pass compresses by another 10%. The size of measurement matrix would be *ceiling*(256×10%)×256 with t equal to 256.

C. Proposed Structure

Fig. 2 shows the structure of the neural network. STP-Net samples the image and provides initial reconstruction that can be passed to subsequent networks. U-Net [29] has proved to work well with semantic segmentation, which motivates us to apply it for reconstruction as a deblurring step. To set up an image regression network, we remove the last two layers (soft max and segmentation layers) of a U-Net with an encoder-decoder depth of 3 and add a regression layer. In Fig. 4, STP-Net is connected with ISTA-Net and STP layers are used repeatedly as measurement and reconstruction for each iteration. Our experiments adopt five ISTA iterations (phases) and every convolutional layer uses 16 kernels.

D. Experimental Results

For the sampling process, we applied different sampling rate combinations (20%+5%, 10%+10% and 5%+20%) in the two measurement layers to reach total measurement rate of 1%. The mean peak signal-to-noise ratio (PSNR) of the 3000 test images with these sampling rate combinations are 20.55, 21.50 and 20.64 dB, respectively, without U-Net. It seems that the square root (10%+10%) of the total measurement rate could be a better choice. Tables I to III show that the proposed method has better performance than other methods and has higher PSNR and structural similarity measure (SSIM).

From Fig. 3, we see that other methods have block artifacts. For DR^2-Net [14], the image is measured block-by-block and reconstructed with 4 residual blocks. Then, they use BM3D [15] to remove the artifacts caused by block-wise processing. We modified the process by composing an intermediate reconstructed image with their initial reconstruction and then using the 4 residual blocks to remove block artifacts. It has better performance than their original method even without

BM3D. Full convolutional measurement network (FCMN) has block artifacts too, especially at the four edges of a reconstructed image. We apply one residual block after it as the authors did in [16]. The block artifacts are alleviated, but the four corners of restored image are darker. From Tables I to IV, it shows that STP-Net provides an attractive initial reconstruction quality for another network. Table III shows that STP-Net works well in less time. In Table IV, ISTA-Net and ISTA-Net+ adopt nine ISTA iterations with every convolutional layer having 32 kernels. STP-Net provides ISTA-Net with information-rich measurements and reasonable initial reconstruction, that enable it to simplify its structure with fewer ISTA iterations and kernels to have better performance.

VI. CONCLUSION

We have presented an STP-based neural network to the problem of CS image reconstruction. The measurement and initial reconstruction process are efficiently implemented through STP without dividing the image into blocks and vectorizing. At different measurement rates, our algorithm yields superior quality reconstructions than other methods. The method does not have block artifacts that many people try to solve. It makes the sampling process convenient and provides good initial reconstruction for subsequent network, such as U-Net or ISTA-Net. In future work, we are going to apply it in deep equilibrium architecture to develop an efficient, high performance fixed-point iteration layer [30].

TABLE I. PSNR VALUES IN dB ON SET11 WITH DIFFERENT ALGORITHMS AT 1% MEASUREMENT RATE

Image Name	**ReconNet +BM3D** [13]	**DR^2-Net** [14]	**DR^2+BM3D** [14]	**Modified DR^2-Net** [14]	**FCMN** [16]	**FC-Res** [16]	**STP-Net**	**STP-UNet**
Barbara	19.08	18.65	19.10	19.02	20.38	20.97	21.83	**22.10**
Boat	18.83	18.67	18.95	18.82	19.96	20.57	21.46	**22.23**
Cameraman	17.49	17.08	17.34	17.72	19.16	19.68	20.14	**21.25**
Fingerprint	14.88	14.73	14.95	14.92	15.56	15.83	**16.16**	**16.16**
Flintstones	14.08	14.01	14.18	13.29	14.46	14.77	15.28	**15.37**
Foreman	20.33	20.59	21.08	22.54	21.08	23.72	**27.15**	27.00
House	19.52	19.61	19.99	20.61	20.93	22.38	23.16	**24.47**
Lena	18.05	17.97	18.40	18.51	20.49	21.15	21.95	**22.72**
Monarch	15.49	15.33	15.50	15.52	17.20	17.58	18.28	**18.79**
Parrot	18.30	18.01	18.41	19.18	20.12	21.42	22.17	**22.96**
Pepper	16.96	16.90	17.11	17.24	19.07	19.45	20.48	**21.28**

(For ReconNet, we use the results reported in [13]. For DR^2-Net and DR^2+BM3D, we use the results reported in [14]. For the other algorithms, the experiments use MATLAB with networks trained from the same dataset with the same images.)

TABLE II. SSIM VALUE FOR 11 EXTRA IMAGES

Image Name	**ReconNet** [13]	**Modified DR^2-Net** [14]	**FCMN** [16]	**FC-Res** [16]	**STP-Net**	**STP-UNet**
Barbara	0.3730	0.3578	0.4555	0.4575	0.5024	**0.5271**
Boat	0.4140	0.3838	0.4729	0.4771	0.4950	**0.5587**
Cameraman	0.4517	0.4391	0.4998	0.5389	0.5503	**0.6565**
Fingerprint	0.1641	0.0708	0.0853	0.0858	0.0884	**0.0886**
Flintstones	0.2733	0.1789	0.2386	0.2429	0.2580	**0.2871**
Foreman	0.5647	0.6078	0.6680	0.6849	0.7536	**0.7869**
House	0.5278	0.5282	0.5809	0.5948	0.6291	**0.7056**
Lena	0.4418	0.4344	0.5364	0.5489	0.5765	**0.6324**
Monarch	0.3802	0.3427	0.4683	0.4816	0.5003	**0.5578**
Parrot	0.5329	0.5197	0.6494	0.6537	0.7145	**0.7512**
Pepper	0.4002	0.4059	0.5206	0.5344	0.5540	**0.6242**
Mean SSIM	0.4112	0.3881	0.4705	0.4819	0.5111	**0.5615**

(For ReconNet, we calculate the values of SSIM from the images the authors provide.)

TABLE III. RESULTS OF 3000 TEST IMAGES

Evaluation index	**Modified DR^2-Net** [14]	**FCMN** [16]	**FC-Res** [16]	**STP-Net**	**STP-UNet**
Mean SSIM	0.3696	0.4347	0.4563	0.4911	**0.5301**
Mean PSNR	18.72	19.26	20.45	21.50	**22.06**
Elapsed Time(s)	56.62	10.70	22.02	**9.36**	41.33

(The number in the table are the mean of 10 times experiments.)

TABLE IV. AVERAGE PSNR (dB) PERFORMANCE COMPARISONS ON SET11 WITH DIFFERENT CS RATIOS

Sampling rate	**ReconNet +BM3D** [13]	**DR^2-Net** [14]	**DR^2+BM3D** [14]	**FCMN** [16]	**FC-Res** [16]	**ISTA-Net** [18]	**ISTA-Net+** [18]	**STP-Net**	**STP-ISTA-Net**
1%	17.55	17.44	17.73	18.95	19.77	17.30	17.34	20.65	**21.30**
4%	20.44	20.80	21.29	23.14	24.22	21.23	21.31	23.39	**24.92**
10%	23.23	24.32	24.71	25.36	27.30	25.80	26.64	26.02	**28.65**
25%	25.92	28.66	29.06	28.69	31.15	31.53	32.57	30.06	**33.54**

(The best performance is labeled in bold.)

ACKNOWLEDGEMENT

This research was partially funded through a grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

REFERENCES

[1] E. J. Candes and J. K. Romberg, “Signal recovery from random projections,” *Comput. Imaging III*, vol. 5674, pp. 76–86, 2005.

[2] R. G. Baraniuk, V. Cevher, and M. F. Duarte, “Model-based compressive sensing,” *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1982–2001, 2010.

[3] D. L. Donoho, “Compressed sensing,” *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[4] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM*, vol. 43, pp. 129–159, 2001.

[5] S. G. Mallat and Z. Zhang, “Matching pursuits with time-frequency dictionaries,” *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, 1993.

[6] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, “Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition,” *Conf. Rec. Asilomar Conf. Signals, Syst. Comput.*, vol. 1, pp. 40–44, 1993.

[7] J. Zammit and I. J. Wassell, “Adaptive block compressive sensing: Toward a real-time and low-complexity implementation,” *IEEE Access*, vol. 8, pp. 120999–121013, 2020.

[8] R. Tibshirani, “Regression shrinkage and selection via the Lasso,” *J. R. Stat. Soc. Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.

[9] E. J. Candès, J. K. Romberg, and T. Tao, “Stable signal recovery from incomplete and inaccurate measurements,” *Commun. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, 2006.

[10] J. Wang, S. Ye, Y. Ruan, and C. Chen, “Low storage space for compressive sensing: Semi-tensor product approach,” *Eurasip J. Image Video Process.*, vol. 2017, no. 1, pp. 1-13, 2017.

[11] J. Wang, Z. Xu, Z. Wang, S. Xu, and J. Jiang, “Rapid compressed sensing reconstruction: A semi-tensor product approach,” *Inf. Sci. (Ny).*, vol. 512, pp. 693–707, 2020.

[12] A. Mousavi, A. Patel, and R. Baraniuk, “A deep learning approach to structured signal recovery,” *53rd Annu. Allert. Conf. Commun. Control. Comput. Allert. 2015*, pp. 1336–1343, 2016.

[13] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, “ReconNet: Non-iterative reconstruction of images from compressively sensed measurements,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016, pp. 449–458, 2016.

[14] H. Yao, F. Dai, S. Zhang, Y. Zhang, Q. Tian, and C. Xu, “DR2-Net: Deep residual reconstruction network for image compressive sensing,” *Neurocomputing*, vol. 359, pp. 483–493, 2019.

[15] K. Dabov, A. Foi, and K. Egiazarian, “Image denoising by sparse 3-D transform-domain collaborative filtering,” *IEEE Trans. IMAGE Process.*, vol. 16, no. 8, pp. 2080–2095, 2007.

[16] J. Du, X. Xie, C. Wang, G. Shi, X. Xu, and Y. Wang, “Fully convolutional measurement network for compressive sensing image reconstruction,” *Neurocomputing*, vol. 328, pp. 105–112, 2019.

[17] J. Du, X. Xie, C. Wang, and G. Shi, “Perceptual compressive sensing,” *Lect. Notes Comput. Sci*, vol. 11258 LNCS, pp. 268–279, 2018.

[18] J. Zhang and B. Ghanem, “ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing,” *IEEE/CVF Conf. Computer Vision and Pattern Recognition*, pp. 1828–1837, 2018.

[19] Qi, Hongsheng and Daizhan Cheng. "Analysis and control of boolean networks: A semi-tensor product approach." *2009 7th Asian Control Conference*. IEEE, pp. 1352–1356, 2009.

[20] Candes E, Romberg J. “L1-magic : Recovery of sparse signals via convex programming,” April 2022. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.212.9120&rep=rep1&type=pdf.

[21] D. Margalit and J. Rabinoff, *Interactive Linear Algebra*. Georgia Institute of Technology, 2019.

[22] A. Akansu, R. Haddad, and P. Haddad, *Multiresolution signal decomposition: transforms, subbands, and wavelets*. Academic Press, 2001.

[23] T. Wimalajeewa, Y. C. Eldar, and P. K. Varshney, “Recovery of sparse matrices via matrix sketching,” *CoRR*, vol. 2, no. 5, pp. 1–5, 2013.

[24] G. Dasarathy, P. Shah, B. Bhaskar, and R. Nowak, “Sketching sparse matrices,” *arXiv Prepr. arXiv1303.6544*, pp. 1–33, 2013.

[25] G. Dasarathy, P. Shah, B. Bhaskar, and R. Nowak, “Sketching sparse matrices, covariances, and graphs via tensor products,” *IEEE Trans. Inf. Theory*, vol. 61, no. 3, pp. 1373–1388, 2015.

[26] MathWorks, “Define Custom Deep Learning Layer with Learnable Parameters,” April 2022. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/define-custom-deep-learning-layer.html.

[27] MathWorks, “Specify Layers of Convolutional Neural Network,” April 2022. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html.

[28] MathWorks, “2-D Convolutional Layer,” April 2022. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.convolution2dlayer.html.

[29] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” *Int. Conf. Med. Image Comput. Comput. Interv.*, pp. 234–241, 2015.

[30] D. Gilton, G. Ongie, and R. Willett, “Deep equilibrium architectures for inverse problems in imaging,” *arXiv Prepr. arXiv2102.07944*, pp. 1–21, 2021.

Alcohol Detection over Long Periods Using Smartphone Accelerometer Data

Manuel Gil-Martín, Rubén San-Segundo, Cristina Luna-Jiménez
Speech Technology and Machine Learning Group, Information Processing and Telecommunications Center,
E.T.S.I. Telecomunicación, Universidad Politécnica de Madrid
Madrid, Spain
Email: manuel.gilmartin@upm.es; ruben.sansegundo@upm.es; cristina.lunaj@upm.es

***Abstract*—This paper proposes a motion biomarker for alcohol detection using a deep learning approach that processes inertial signals recorded with a smartphone. The deep learning architecture is composed of a Convolutional Neural Network, including three convolutional layers for learning features from the inertial signal spectrum, and several fully connected layers to perform classification and regression tasks. The motion biomarker is computed in two steps. Firstly, the inertial signals are segmented in short sub-windows (3-6 seconds) and the system generates a score for each sub-window. Secondly, the scores in consecutive sub-windows are combined to provide a motion biomarker over in longer periods of time (30 seconds). This paper compares the proposed approach to previous works using the same experimental dataset and setup: Bar Crawl Detecting Heavy Drinking Data Set, K-fold cross-validation methodology and two tasks (classification and regression). The proposed deep learning approach overperformed previous reported results: the accuracy increased 4 % (absolute) when classifying between intoxicated and sober participants and the Mean Squared Error relatively decreased 9 % when estimating the Transdermal Alcohol Content of the participants by averaging the scores from consecutive sub-windows.**

Keywords-Alcohol Detection; Motion Wearable Sensors; Convolutional Neural Networks; Sub-windows Combination.

I. INTRODUCTION

High-frequency alcohol consumption could become a serious threat for people's health. In fact, physicians and social workers are interested in reducing the alcohol consumption in young adults. For example, measuring the Transdermal Alcohol Content (TAC) is useful to recommend the person to stop drinking in real time. In addition, wearable technology could be used for developing an alcohol detection system based on inertial signals from accelerometers included in smartphones.

This paper evaluates a strategy to combine sub-windows information for an alcohol detection system based on wearable technology and deep learning, obtaining important improvements for long windows (over 10 seconds). This study was performed over the public dataset Bar Crawl: Detecting Heavy Drinking Data Set. It contains acceleration recordings from 13 subjects during a university event and measurements from a TAC sensor. The results significantly outperform the performance reported in previous works over the same dataset.

This paper is organized as follows. Section 2 reviews the related work. Section 3 describes the material and methods, including the dataset, the signal processing and deep learning modules. Section 4 details the evaluation metrics and the experiments performed in this work. Finally, Section 5 summarizes the main conclusions of this work.

II. RELATED WORK

Alcohol detection through mobile sensing has gained popularity in the last years. Researchers have combined different sources of information from smartphones and wearable devices, such as acceleration signals, location, keystroke speed, or sent/received calls to predict intoxication levels of alcohol consumers. Moreover, inertial signals from wearables and smart devices have been used for motion modelling in other areas, like activity classification [1][2] or biometrics [3]. This section describes several previous works on alcohol detection based on mobile sensing.

Kao et al. [4] developed controlled laboratory experiments to classify alcohol intoxication through smartphone accelerometer signals. Arnold et al. [5] compared several machine learning algorithms (Naïve Bayes, Decision Tree, Support Vector Machines and Random Forest) using acceleration data from the smartphone to classify alcohol intoxication levels through the number of drinks consumed by a user. They proved that Random Forest was the most accurate classifier, reaching 56% and 70% accuracy on the training and validation sets, respectively, classifying the number of drinks into ranges of 0-2 drinks (sober), 3-6 drinks (tipsy) or >6 drinks (drunk). This work reached encouraging results, but they used potentially biased self-reports to measure ground-truth intoxication levels, which could limit the reliability of the results.

Santani et al. [6] characterized youth drinking behavior using smartphones involving 241 participants during a weekend night using a Random Forest classification algorithm to infer whether an individual consumed alcohol (over a threshold). This work also used self-reports on individual alcoholic drinks consumed on Friday and Saturday nights over a three-month period. They concluded that accelerometer data was the most informative single signal, reaching an accuracy of 75.8%.

McAfee et al. [7] used drunk busters goggles to distort vision and simulate the effects of alcohol consumption on the body and rate at four BAC levels [0.00-0.08), [0.08-0.15), [0.15-0.25), [0.25+). They used accelerometer and gyroscope features from smartphone, height, weight, and gender reached to classify 33 subjects into these BAC levels. This previous work used 5-second segments and reached 89.45%

of accuracy when detecting the BAC level using a decision tree classifier when using 99% as training data and 1% as testing data and 73.74% using a Random Forest algorithm when using a 10-fold cross-validation setup.

Killian et al. [8] measured accelerometer signals with a smartphone and TAC data during a drinking event in a non-intrusively way. This work used 10-second windows of acceleration recordings and the authors randomized the data using 75% for training and 25% for testing. They compared machine and deep learning algorithms, concluding that Random Forest approach outperformed the classification between sober (TAC < 0.08) and intoxicated (TAC >= 0.08) participants and with a 77.48% of accuracy. Another previous work [9] used the same dataset and performed both classification and regression task using a K-fold cross-validation strategy. They used a Convolutional Neural Network (CNN) architecture and obtained an accuracy of 80.43 ± 0.21 % using 2-second windows for the classification task and a MSE of alcohol content estimation of 0.001559 ± 0.000011 g/dl for the regression task.

In addition, a previous work [10] analyzed the effect performance saturation in activity recognition when increasing the analysis window size. They proposed several strategies to combine the information from consecutive sub-windows, obtaining significant improvements compared to directly using long windows. This paper combines several sub-windows at the end of a CNN architecture, obtaining significant improvements compared to previous works using the same dataset.

III. MATERIAL AND METHODS

This section describes the dataset used for the experiments, the signal processing, the CNN, and the post-processing module for combining the scores from sub-windows.

A. Dataset

We used the Bar Crawl: Detecting Heavy Drinking Data Set" [8]. It includes recordings from 13 undergraduate students in a drinking event. The dataset includes acceleration signals from a sensor embedded in smartphones sampled at 40 Hz and TAC measurements collected with an ankle bracelet. A TAC=0.08 g/dl was used as the level to discriminate between intoxicated participants (TAC >= 0.08) and sober participants (TAC < 0.08). Participants joined in drinking activities without any instruction. For the classification task, we considered two classes: intoxicated and sober participants. For the regression task, our target was to estimate the TAC. The total duration of the dataset is 77 hours approximately. The acceleration values mostly vary between -4 and 4g, and Table 1 summarizes the acceleration and TAC signals statistics.

B. Signal Processing

We divided the accelerometer signals into non-overlapped consecutive sub-windows using a Hanning function (other functions were evaluated like Hamming or Blackman without significant differences). In this paper, the system provided a consumption score per sub-window, and we integrated consecutive scores to evaluate longer periods.

TABLE I. ACCELERATION AND TAC SIGNALS STATISTICS

Signal	Units	Min	Mean	Max
X	g	-43.335	-0.009	39.23
Y	g	-33.475	0.001	27.311
Z	g	-49.023	0.056	42.313
TAC	g/dl	0	0.065	0.443

For each sub-window, we computed the Fast Fourier Transform (FFT). For example, in case of using 3-second sub-windows, we used 60 bins in the frequency domain per example as inputs to the CNN corresponding to the FFT magnitude from 0 to 20 Hz. As in previous works using this dataset, we only considered temporal windows whose estimated energy was higher than zero at 2 Hz (average human walking activity frequency). We used GNU Octave for the signal processing step (windowing and computing the FFT).

C. Deep Learning Architecture

We used a deep learning approach composed of a feature learning subnet and a classification subnet. Figure 1 represents the architecture that models and classifies participants between intoxicated (TAC >= 0.08) and sober (TAC < 0.08) using 3-second sub-windows. The first part of the structure learns features from the spectra using three convolutional layers and one intermediate max-pooling layer. The second part of architecture contains fully connected layers that classify the sub-windows as intoxicated or sober subjects. The last layer has one neuron and uses the sigmoid activation function, and the binary cross-entropy loss metric for classification problem. In case of regression problem, this last layer has a linear activation function and uses the mean squared error loss metric. In intermediate layers, ReLU is used as activation function to reduce the impact of gradient vanishing effect. Both tasks used the root-mean-square propagation optimizer [11], with learning rates of 0.001 and 0.00005 for classification and regression tasks, respectively. It was discovered that to achieve better results on the regression task, a lower learning rate was required. Before reporting testing results, the validation subset (10 % of training subset) was used to tune the number of epochs (10) and the batch size (200) of the architecture. Each This architecture, which uses 3-second sub-windows, has 137,665 parameters. We used Python distribution with Tensorflow and Keras libraries to create the deep neural network architecture.

D. Post-processing Decision Module

Combining the information along several consecutive sub-windows allows increasing the decision robustness, by

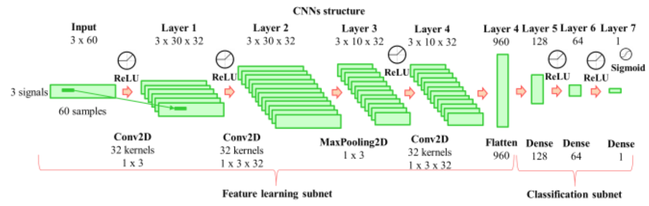


Figure 1. Deep learning architecture including convolutional and fully connected layers for classification. This is the deep learning module which obtains a score sub-window by sub-window.

evaluating long periods of time that keep a uniform behavior. The mean filtering approach used in this work consisted of running a non-overlapped filter through a specific signal and computing the mean of N sub-windows. In this sense, this mean filtering was used as a post-processing technique that allowed the integration of information from consecutive windows using the final scores of the CNN output. After computing the mean score of N consecutive windows, we obtained a single value which integrates the information along these windows. When the prediction is completely filtered, the final number of examples would be divided by N. We used N=1, 2, 4, 5, 6, 8, 10, 12, and 14, being N=1 the lack of filtering. Figure 2 shows an example of mean filtering of final prediction using N=4 with 3-second sub-windows, where the prediction (between 0 and 1) is modified after applying the filtering technique and integrating more time (12 s) and some isolated errors are corrected through this integration of temporal information.

IV. Experiments and Discussion

This section defines the evaluation metrics used in this work and shows the results in the experiments.

A. *Evaluation Metrics and Validation*

This paper performs two tasks: TAC classification and regression. For the classification task, we used accuracy: the ratio between the number of correctly classified examples and the number of total examples. In our case, every analysis window is considered as an example. This metric is presented with confidence intervals of 95%, obtained with (1), given M examples (windows) and a specific value of accuracy. Two results are considered significantly different when there is no overlap in these confidence intervals. We also used the Area Under the Curve (AUC) to evaluate this binary classification problem.

$$\text{acc (95\%)} = \text{acc} \pm 1.96 * \sqrt{((\text{acc}(100-\text{acc}))/M)} \qquad (1)$$

Regarding regression task, Mean Square Error (MSE) was considered as the average squared difference between the estimated values and the actual values. This error is presented with confidence intervals of 95%, obtained with (2), given M examples (windows) and an error standard deviation s. We also used the Pearson correlation coefficient between the estimated and the actual TAC measurements to evaluate the regression problem.

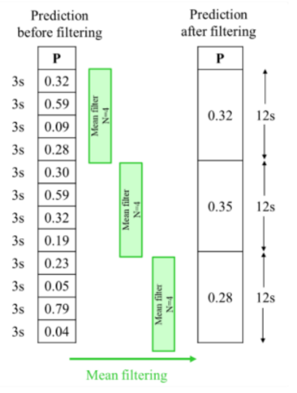


Figure 2. Mean filtering of predictions using N=4 and 3-second sub-windows.

$$\text{MSE (95\%)} = \text{MSE} \pm 1.96 * s / \sqrt{M} \qquad (2)$$

In this work, we used K-fold cross-validation for comparison to previous works: data is divided into K folds (13 in this work) to divide data in training, validation, and testing subsets. A different fold is used for testing in each iteration, with the remaining folds used for training (10 % of training subset was used for validation). This methodology allows to evaluate the system over all available data using different data distributions. The reported results are the average along all iterations. For example, in case of using 3-second sub-windows, a total of 92,000 examples approximately are considered. For each fold, 7,000 examples for testing and 85,000 examples for training (8,500 for validation) approximately. Training the model with a 90% of data, guarantees a well-trained model: reducing this amount could have a negative impact over the performance. We observed these classification and regression tasks are high-user dependent, so a leave-one-out approach is considered as future work.

B. *Experiments*

We analyzed the influence of the window size and the combining information technique averaging the predictions from sub-windows after the deep learning architecture over the classification and regression tasks. As baseline system [9], we performed lack of combination experiments using long windows directly to observe the performance saturation when increasing the analysis window length. After that, we compared these results to our approach: averaging the scores from sub-windows after the CNN.

Related work section mentioned previous works [8] [9] that obtained 77.48 % and 80.43 ± 0.21 % of accuracy for the classification task and a MSE of alcohol content estimation of 0.001559 ± 0.000011 g/dl for the regression task.

Figure 3 and Figure 4 show the test accuracy and AUC, respectively, using the baseline approach (lack of combination), 3-second sub-windows, and 6-second sub-windows combination for the alcohol classification task. Figure 5 and Figure 6 show the test MSE and correlation,

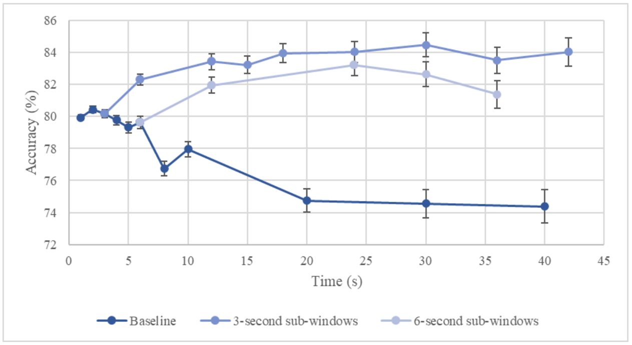


Figure 3. Accuracy evolution including the baseline results and the performance results obtained when integrating 3-second and 6-second sub-windows for the alcohol classification task.

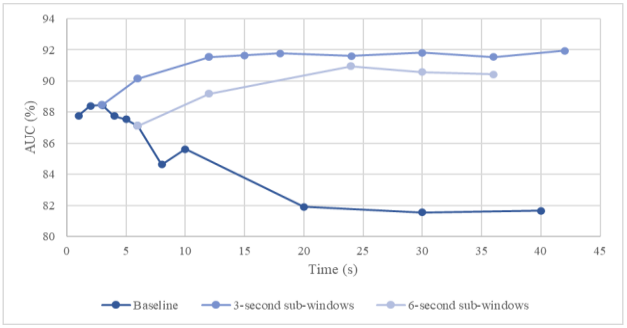


Figure 4. AUC evolution including the baseline results and the performance results obtained when integrating 3-second and 6-second sub-windows for the alcohol classification task.

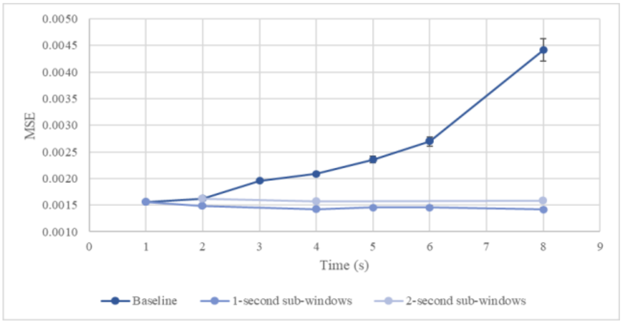


Figure 5. MSE evolution including the baseline results and the performance results obtained when integrating 1-second and 2-second sub-windows for the alcohol estimation task.

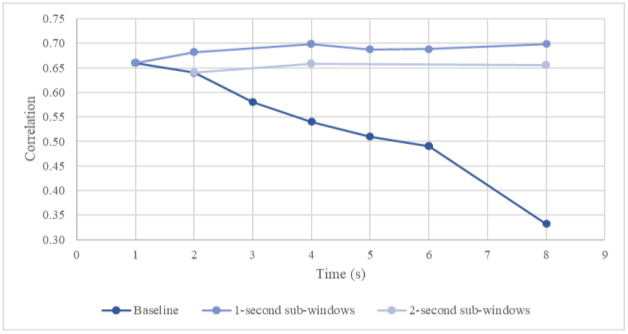


Figure 6. Correlation evolution including the baseline results and the performance results obtained when integrating 1-second and 2-second sub-windows for the alcohol estimation task.

respectively, using the baseline approach (lack of combination), 1-second sub-windows, and 2-second sub-windows combination for the alcohol estimation task. These figures show that the baseline approach achieves a performance saturation when increasing the analysis window length because we raise the number of parameters to be trained in the CNN and the spectral resolution, that increases the overfitting risk. However, integrating the scores after the CNN allows to boost the classification and regression performances, reaching a maximum in accuracy (84.47 ± 0.74 %) and AUC (91.82 %) evaluation metrics for 30 s evaluation using 3-second sub-windows. In the case of alcohol content estimation, the best result was an MSE of 0.00142 ± 0.00002 and a correlation of 0.7, obtained when combining four 1-second sub-windows.

V. CONCLUSION AND FUTURE WORK

Detecting alcohol consumption through wearable technology and deep learning is very interesting to avoid health risks in the future. This paper contributes to the supervision of alcohol consumption from acceleration signals by proposing a method to evaluate long periods of time. The system leverages that alcohol content is quite stable in time to integrate information from short sub-windows and boost the classification and regression performances. Using these short sub-windows, it is possible to decrease the number of parameters to be trained in the CNN and reduce the overfitting risk that occurs when increasing the spectral resolution. This work used the Bar Crawl: Detecting Heavy Drinking Data Set, obtaining better performance than previous works that used the same dataset.

As future work, it would be interesting to leverage the sequential information from sub-windows using Long Short-Term Memory (LSTM) layers to analyze the evolution of the alcohol content. In addition, we observed that the current approach has the limitation of generalizing to unseen subjects, so it would be useful to apply adaptation techniques and focus on specific characteristics of subjects in a Leave-One-Subject-Out CV scenario.

ACKNOWLEDGMENT

The work leading to these results is part of the projects AMIC (MINECO, TIN2017-85854-C4-4-R), AMIC-PoC (PDC2021-120846-C42), CAVIAR (MINECO, TEC2017-84593-C2-1-R) and GOMINOLA (PID2020-118112RB-C22) funded by MINECO/AEI/10.13039/501100011033 and by “the European Union “NextGenerationEU/PRTR”. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

REFERENCES

[1] M. Gil-Martin, R. San-Segundo, F. Fernandez-Martinez, and J. Ferreiros-Lopez, "Improving physical activity recognition using a new deep learning architecture and post-processing techniques," *Engineering Applications of Artificial Intelligence*, vol. 92, Jun 2020, pp. 103679, doi: 10.1016/j.engappai.2020.103679.

[2] M. Gil-Martín, R. San-Segundo, F. Fernández-Martínez, and R. de Córdoba, "Human activity recognition adapted to the type of movement," *Computers & Electrical Engineering*, vol. 88, pp. 106822, 2020/12/01/ 2020, doi: https://doi.org/10.1016/j.compeleceng.2020.106822.

[3] M. Gil-Martin, R. San-Segundo, R. de Cordoba, and J. Manuel Pardo, "Robust Biometrics from Motion Wearable Sensors Using a D-vector Approach," *Neural Processing Letters,* pp 2109-2125, 2020, doi: 10.1007/s11063-020-10339-z.

[4] H.-L. Kao, B.-J. Ho, A. C. Lin, H.-H. Chu, and M. Assoc Comp, "Phone-based Gait Analysis to Detect Alcohol Usage," *Ubicomp'12: Proceedings of the 2012 Acm International Conference on Ubiquitous Computing*, pp. 661-662, 2012 2012.

[5] Z. Arnold, D. LaRose, and E. Agu, "Smartphone Inference of Alcohol Consumption Levels from Gait," (in English), *2015 Ieee International Conference on Healthcare Informatics (Ichi 2015)*, Proceedings Paper pp. 417-426, 2015, doi: 10.1109/ichi.2015.59.

[6] D. Santani, T. M. T. Do, F. Labhart, S. Landolt, E. Kuntsche, and D. Gatica-Perez, "DrinkSense: Characterizing Youth Drinking Behavior Using Smartphones," (in English), *Ieee Transactions on Mobile Computing*, Article vol. 17, no. 10, pp. 2279-2292, Oct 2018, doi: 10.1109/tmc.2018.2797901.

[7] A. McAfee, J. Watson, B. Bianchi, C. Aiello, and E. Agu, "AlcoWear: Detecting Blood Alcohol Levels from Wearables," *2017 Ieee Smartworld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (Smartworld/Scalcom/Uic/Atc/Cbdcom/Iop/Sci)*, pp. 1-8, 2017, doi: 10.1109/UIC-ATC.2017.8397486.

[8] J. A. Killian, K. M. Passino, A. Nandi, D. R. Madden, and J. D. Clapp, "Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data," in *KHD@IJCAI*, pp. 35-42, 2019. Available from: https://archive.ics.uci.edu/ml/datasets/Bar+Crawl%3A+Detecting+Heavy+Drinking, 2022.04.04

[9] M. Gil-Martin, R. San-Segundo, L. Fernando D'Haro, and J. Manuel Montero, "Robust Motion Biomarker for Alcohol Consumption," *Ieee Instrumentation & Measurement Magazine*, vol. 25, no. 1, pp. 83-87, Feb 2022, doi: 10.1109/mim.2022.9693446.

[10] M. Gil-Martin, R. San-Segundo, F. Fernandez-Martinez, and J. Ferreiros-Lopez, "Time Analysis in Human Activity Recognition," *Neural Processing Letters*, vol. 53, pp. 4507–4525 2021, doi: 10.1007/s11063-021-10611-w.

[11] N. A. Weiss, *Introductory statistics*. Pearson, 2017.

Discovering Causality in Event Time-Series

Pavel Loskot
ZJU-UIUC Institute
Haining, China
e-mail: *pavelloskot@intl.zju.edu.cn*

***Abstract*—The event time-series can accurately describe the behavior of many dynamic systems. The challenge is that the events are categorical variables, so they cannot be analyzed by the existing statistical methods developed for numerical time-series. In order to infer the causally related events, in this paper, it is proposed to assume the empirical conditional probabilities of nearly certain and nearly uncertain events. Moreover, since the event ordering is usually locally irrelevant, the event sequences can be transformed into the event sets or multi-sets with appropriately defined distance metrics. The event sequences having a zero distance can be then assumed to be causally equivalent. The distance metrics are also used in matrix profile analysis of event time-series. Numerical examples are studied for chemical reaction events generated in stochastic simulations of biochemical molecular systems. Even though the proposed framework for discovering the causally related event sequences can be readily fully automated, they still need to be properly interpreted in the context of relevant domain knowledge.**

***Keywords*—causal; dynamic system; event; matrix profile; state-space; time-series**

I. INTRODUCTION

Traditional signal processing and machine learning mainly exploit statistical associations within data. However, it is well known that a strong association is neither necessary nor sufficient for causality, for example, due to confounding. At the same time, a weak association cannot rule out the causality. Recently, there has been a great interest in developing data models and processing methods, which are interpretable [1].

The cause and effect are central to scientific hypotheses testing, experiment design, and to generate the prescriptive analytics of engineering systems. It is possible to only consider whether the cause-effect exists without determining its direction or strength. The causal relationships can be represented as Structural Causal Models (SCM) [2]. The SCM can be created from a prior knowledge, or inferred by performing statistical independence tests on the data. An important question is whether the SCM can be determined from the observed data, and whether such a SCM is unique. The SCM can be converted into a Bayesian network using do-calculus [2].

Different approaches were adopted in the literature to obtain causal models of time-series data [3]–[11]. Granger causality decides whether the past values of a time-series can improve the prediction of future values of another time-series. However, this type of causality cannot be used for time-series with instantaneous effects, or when sub-sampling of time-series may mask the causal relationships. The intervention causality enforces a change in the time-series value at a particular time instant, and then the change can be evaluated as an Average Causal Effect (ACE). In supervised and semi-supervised machine learning, the labels of data can be assumed to be a cause of data features (i.e., the effects). It enables to automatically label data as well as to repair incorrect labels. However, all these methods normally assume numerical data.

In [12], the causality is induced by changes in the interaction covariances. The methods for evaluating causal intervention of non-randomized, small-size treatments are surveyed in [13]. A state-space SCM for causal inference in time-series data was studied in [14]. A causal graph discovery over multiple related datasets was proposed in [15]. The limitations of convergent cross-mapping in performing the causal inference were investigated in [16]. The temporal trends in data need to be identified before performing the causal inference as shown in [17]. The causal analysis of small sample sizes was performed in [18] by studying state-space attractors of non-linear dynamical systems. However, none of these works seem to have considered the causal inference for categorical data.

In this paper, our goal is to discover causal relationships within categorical time-series. Such series may represent the events occurring in control and monitoring of dynamic systems. The events cannot be often directly detected, but must be indirectly inferred except in computer simulations of dynamic systems. The events usually incur changes in the system internal states. It is extremely useful to understand what caused these changes, and to make more robust predictions about the anticipated future changes (effects). Moreover, the event time-series can be partitioned into shorter sequences. The task is then to determine the causality between the pairs of the event sequences. In addition, since the event ordering is locally irrelevant, it is proposed to transform the event sequences into the event sets or multi-sets.

More importantly, the cause-effect relationship is newly defined here assuming the conditional probability of nearly certain and nearly uncertain events. This probability is estimated empirically as a relative frequency of occurrence of particular event sequences. Even though such a notion of causality is incomplete, as many event sequences are conditionally neither certain nor uncertain, this approach has the advantage of its implementation simplicity, and it can be fully automated. Moreover, various distance measures [19]–[22] can be used to define equivalences among the event sequences, which can increase the number of these sequences classified as being causally related by our definition of causality. The distance metrics also enable the matrix profile analysis, a versatile framework used for the pattern discovery in time-series data.

Numerical examples are obtained for a biochemical reaction network, where the events represent chemical reactions. The history of chemical reactions are recorded by modifying the downloaded open-source simulation. The event time-series processing and visualization pipeline is implemented using a C++ code and the custom scripts in Python and Bash.

The rest of this paper is organized as follows. Section II describes a common model of dynamic systems with event-driven changes of observations and internal states. The proposed causal framework for analyzing the event time-series is introduced in Section III. Numerical examples are briefly presented in Section IV. The limitations of our work are discussed and the paper is concluded in Section V.

II. System Model

Consider a dynamic system described by transitions between the consecutive stationary states $z_{t-1} \in \mathcal{Z}$ and $z_t \in \mathcal{Z}$ due to periodically occurring events $e_t \in \mathcal{E}$, i.e.,

$$z_t = e_t(z_{t-1}, z_{t-2}, \ldots)$$

where t denotes a discrete time index. The system observations are defined by a generally non-linear function,

$$\boldsymbol{y}_t = O(z_t, z_{t-1}, \ldots)$$

of the current and the previous system states including any intrinsic and extrinsic noises (the latter not shown explicitly). The values $\boldsymbol{y}_t$ may not be available for all indexes t due to practical measurement constraints; this corresponds to uniform or non-uniform sub-sampling of the observations $\boldsymbol{y}_t$.

In this paper, it is assumed that, (1) the system model is memoryless, i.e., $z_{t+1} = e_t(z_t)$, and, $\boldsymbol{y}_t = O(z_t)$, and, (2) the observations are perfect, i.e., $\boldsymbol{y}_t = z_t$, and available for all t. Such Markovian and noise-free observation assumptions greatly simplify our reasoning, and they are satisfied for the system studied in Section IV. Furthermore, the events e_t are represented as categorical variables, such that, $e_t \in \{0,1,2,\ldots\}$. Under these assumptions, the transitions between the observations $\boldsymbol{y}_t$ and the events $\boldsymbol{e}_t$ are depicted in Figure 1. In particular, Figure 1 indicates that different events affect different components in the observed vector, $\boldsymbol{y}_t$. However, the practical constraints on the observations $\boldsymbol{y}_t$, for example, to ensure that, $\boldsymbol{y}_t \geq 0$ for $\forall t$, may enforce a dependency (i.e., a memory) among the successive events e_t. Note also that both the events and the observations are normally dependent on a number of other parameters, which is not explicitly considered in our model description.

The memoryless assumption implies the following property of the event-based modeling of dynamic systems.

Theorem 1: Given the observation $\boldsymbol{y}_t$ at time t, the ordering of $(m+1) > 0$ events in the sequence, $(e_t, e_{t+1}, \ldots, e_{t+m})$, does not affect the observation $\boldsymbol{y}_{t+m}$ at time $(t+m)$.
Theorem 1 asserts that the same observation $\boldsymbol{y}_{t+m}$ is produced for any arbitrary ordering of a particular sequence of events. However, this does not guarantee that all the event orderings satisfy all the observation constraints; for instance, the natural

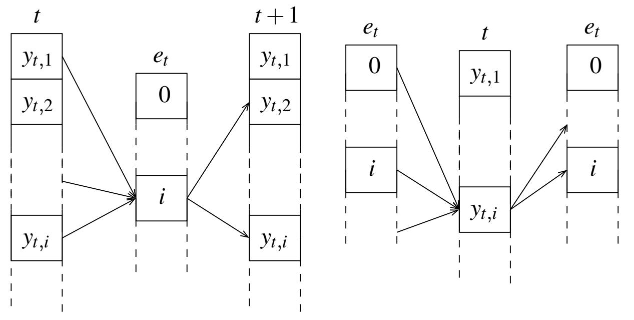


Figure 1. The changes in different components of observations $\boldsymbol{y}_t$ affected by different events $e_t \in \mathcal{E}$.

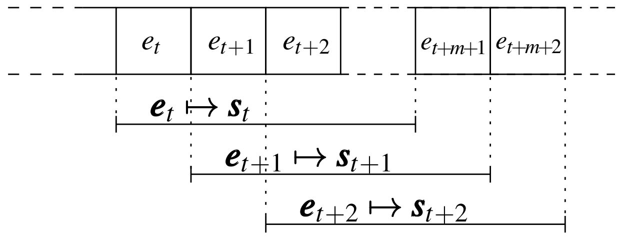

e_t
e_{t+1}
e_{t+2}
e_{t+m+1}
e_{t+m+2}
$\boldsymbol{e}_t \mapsto \boldsymbol{s}_t$
$\boldsymbol{e}_{t+1} \mapsto \boldsymbol{s}_{t+1}$
$\boldsymbol{e}_{t+2} \mapsto \boldsymbol{s}_{t+2}$

Figure 2. The sequences $\boldsymbol{e}_t$ of $(m+1)$ events mapped to (multi-) sets $\boldsymbol{s}_t$.

constraints that the observations are always non-negative, or do not exceed a certain value, may be temporarily violated.

Consequently, the sequences of events, $\boldsymbol{e}_t = (e_t, e_{t+1}, \ldots, e_{t+m})$, can be assumed to be multi-sets (i.e., the same events can appear multiple times, but their ordering is irrelevant), or ordinary sets (the repeated elements are removed). The corresponding (multi-) sets are denoted as, $\boldsymbol{s}_t$, and they can be created by sliding-window partitioning of the original event time-series as shown in Figure 2.

III. Analysis of Event Time-Series

Recall that the events e_t are categorical variables, which can be mapped to non-negative integers $\mathcal{E}$. Since such a mapping is rather arbitrary, and assumed purely for a representation convenience, it cannot be used for evaluating statistical properties of the time-series, $\{e_t\}_t$. Assuming instead the sequence of (multi-) sets, $\boldsymbol{s}_t \in \mathcal{S}$, where $\mathcal{S} \subseteq \mathcal{E} \times \cdots \times \mathcal{E} = \mathcal{E}^{m+1}$, we can examine the probability mass function as well as define various distance measures involving $\boldsymbol{s}_t$. The former approach will be used to identify the causal relationships between pairs of event sequences. The latter approach enables a flexible matrix profile analysis of the event time-series.

A. Causality Between Event Sequences

Our objective is to determine a possible causal relationship between pairs of consecutive but non-overlapping event sequences. Thus, given $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$, $j = i+m+1$, i.e., $\boldsymbol{e}_i \cap \boldsymbol{e}_j = \emptyset$ (empty set), decide, whether the event sequence $\boldsymbol{e}_i$ causes the event sequence $\boldsymbol{e}_j$ (causal learning), or whether the event sequence $\boldsymbol{e}_j$ is an effect of the event sequence $\boldsymbol{e}_i$ (anti-causal learning). One plausible and commonly used strategy is to construct a SCM, and fit it to the data (the event sequences).

The SCM analysis can be combined with interventions and do-calculus to find the causes and effects of specific event sequences. In this paper, we instead propose the following strategy to identify some, but not all pairs of causally related event sequences.

Definition 1: The event sequences $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$, $j > i$, have a cause-effect relationship, provided that their conditional probability,

$$\Pr(\boldsymbol{e}_j|\boldsymbol{e}_i) \to 1. \tag{1}$$

In such a case, the event sequence $\boldsymbol{e}_i$ is said to be a cause of $\boldsymbol{e}_j$. Equivalently, the event sequence $\boldsymbol{e}_j$ is an effect of $\boldsymbol{e}_i$.

The conditional probability (1) represents the likelihood of the event sequence $\boldsymbol{e}_i$, given the event sequence $\boldsymbol{e}_j$. Thus, the prior distribution of $\boldsymbol{e}_i$ is ignored in Definition 1, and so is the joint distribution $\Pr(\boldsymbol{e}_i, \boldsymbol{e}_j)$. This can be justified by noting that a cause-effect relationship is generally asymmetric. Even though there are also cases where $\boldsymbol{e}_i$ causes $\boldsymbol{e}_j$, and at the same time, $\boldsymbol{e}_j$ causes $\boldsymbol{e}_i$, in general, $\Pr(\boldsymbol{e}_j|\boldsymbol{e}_i) \neq \Pr(\boldsymbol{e}_i|\boldsymbol{e}_j)$.

The reason for assuming the conditional probability in Definition 1 to be nearly but not exactly equal to 1 is that the equality is too restrictive and almost never achievable in practice. Moreover, the conditional probability $\Pr(\boldsymbol{e}_j|\boldsymbol{e}_i)$ close to 1 indicates that, given $\boldsymbol{e}_i$, there are only a few possible event sequences $\boldsymbol{e}_j$ following $\boldsymbol{e}_i$ (i.e., the number of such sequences $\boldsymbol{e}_j$ is significantly smaller than the number of all possible event sequences observed).

More importantly, the size of the event space, $\mathcal{E}$, expressed as the cardinality, $|\mathcal{E}|^{(m+1)}$, is often much smaller than the volume of the corresponding observations $\boldsymbol{y}$ in a $(m+1)$-dimensional Euclidean or some other space. This is the key reason for analyzing the sequences of events for inferring the causality rather than directly processing the sequences of observations. Although in most practical scenarios, it is the observations that are available, whereas the internal events can only be inferred from these observations, which is prone to decision errors. However, computer simulations are a notable exception, and they will be utilized in Section IV.

The conditional probability of the event sequences in Definition 1 close to 1 is only one significant case, which can be readily causally interpreted. The other such significant case is represented by the conditional probability being close to 0. This leads to the following definition of causality between two event sequences.

Definition 2: The event sequences $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$, $j > i$, have no cause-effect relationship, provided that their conditional probability,

$$\Pr(\boldsymbol{e}_j|\boldsymbol{e}_i) \to 0 \quad \text{and} \quad \Pr(\boldsymbol{e}_i|\boldsymbol{e}_j) \to 0. \tag{2}$$

The sequences $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are then said to be causally unrelated.

Assuming both conditional probabilities in Definition 2 is necessary in order to ensure that neither $\boldsymbol{e}_i$ nor $\boldsymbol{e}_j$ can be a cause of the other. Moreover, unlike Definition 1, it is much more likely to find the pairs of event sequences having very small or even exactly zero conditional probabilities.

The conditional probabilities of event sequences may sometimes be available from the analysis of a Bayesian model derived from the SCM. However, in many practical scenarios, these probabilities must be empirically estimated from the event time-series data. In such a case, the event sequences $\boldsymbol{e}_t$ of $N = (m+1)$ events are first created by a sliding-window partitioning of the original event time-series. In order to enable Definitions 1 and 2 of causality, the sequences $\boldsymbol{e}_t$ are further subdivided into two disjoint sub-sequences (omitting the time index for brevity), $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ of N_1 and N_2 events, respectively, so that,

$$\boldsymbol{e}_t = \boldsymbol{e}_i \cup \boldsymbol{e}_j, \quad \boldsymbol{e}_i \cap \boldsymbol{e}_j = \emptyset$$

and $N = N_1 + N_2$, $N_1 = |\boldsymbol{e}_i|$, $N_2 = |\boldsymbol{e}_j|$, and importantly, all the events in $\boldsymbol{e}_i$ precede the events in $\boldsymbol{e}_j$. The corresponding (multi-) sets are denoted as $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, and they are referred to as the left and the right event (multi-) sets, respectively.

Define a 2D counter (matrix), $C_{i,j}$, of the number of the unique left and right sub-sequences, $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$, composing the event sequences, $\boldsymbol{e}_t$. The conditional probabilities (1) and (2) can be then estimated as,

$$\Pr(\boldsymbol{e}_j|\boldsymbol{e}_i) \approx C_{i,j}/K_i \tag{3}$$

where K_i denotes the number of times a specific event sub-sequence $\boldsymbol{e}_i$ was observed, i.e., $K_i = \sum_j C_{i,j}$.

There are, however, two issues with the causality in Definitions 1 and 2. The first problem is that the number of sub-sequences $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ satisfying (2) and especially (1) can be rather small in comparison to the total number of all observed event sub-sequences. This leaves out most other pairs of event sub-sequences $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$, for which their causal relationship cannot be determined using Definitions 1 and 2, since their conditional probability, $0 < \Pr(\boldsymbol{e}_j|\boldsymbol{e}_i) < 1$. The second problem is that identifying the rarely occurring, causally related sub-sequences using the estimator (3) becomes less accurate, unless sufficiently long event time-series are available.

In order to overcome these issues, we can exploit the mapping of event sequences to event (multi-) sets, as discussed in Section II. It allows us to define various notions of distances between the event sequences $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ as follows. Let d_0 be the Hamming distance between $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$. Then, any of the following expressions can be assumed as a distance metric between the event sequences $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$.

$$\begin{aligned}
d(\boldsymbol{e}_i,\boldsymbol{e}_j) &= d_0 - |\boldsymbol{s}_i \cup \boldsymbol{s}_j| && \text{(4a)}\\
d(\boldsymbol{e}_i,\boldsymbol{e}_j) &= d_0 - |\boldsymbol{s}_i \cap \boldsymbol{s}_j| && \text{(4b)}\\
d(\boldsymbol{e}_i,\boldsymbol{e}_j) &= d_0 - (|\boldsymbol{s}_i| + |\boldsymbol{s}_j|) && \text{(4c)}\\
d(\boldsymbol{e}_i,\boldsymbol{e}_j) &= d_0 - \max(|\boldsymbol{s}_i|, |\boldsymbol{s}_j|) && \text{(4d)}\\
d(\boldsymbol{e}_i,\boldsymbol{e}_j) &= \max(|\boldsymbol{s}_i|, |\boldsymbol{s}_j|) - \min(|\boldsymbol{s}_i|, |\boldsymbol{s}_j|) && \text{(4e)}\\
d(\boldsymbol{e}_i,\boldsymbol{e}_j) &= \min(|\boldsymbol{s}_i \setminus \boldsymbol{s}_j|, |\boldsymbol{s}_j \setminus \boldsymbol{s}_i|). && \text{(4f)}
\end{aligned}$$

Thus, always, $d(\boldsymbol{e}_i,\boldsymbol{e}_j) \geq 0$, $d(\boldsymbol{e}_i,\boldsymbol{e}_j) = d(\boldsymbol{e}_j,\boldsymbol{e}_i)$, and $|\boldsymbol{s}_i| \leq |\boldsymbol{e}_i|$.

Furthermore, in order to increase the number of occurrences of event sequences which are either causally related by Definition 1, or causally unrelated by Definition 2, we can assume any of the distance metrics (4a)-(4f) to introduce the following notion of equivalent event sequences.

Definition 3: The event (sub-) sequences $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are said to be equivalent, provided that their distance, $d(\boldsymbol{e}_i, \boldsymbol{e}_j) = 0$.

It is clear that by discarding the event ordering and keeping only the unique events in event sets as assumed in the distance metrics (4a)-(4f), the number of equivalent event sub-sequences can grow substantially, and so does the number of event sub-sequence pairs satisfying either Definition 1 or Definition 2.

The effect of assuming the equivalent event sub-sequence is illustrated in Figure 3. In the first step, the unique event sub-sequence pairs $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are identified, and their multiplicities, $C_{i,j}$, are counted. Using Definition 3, the equivalent event sub-sequences can be identified among the right or the left sub-sequences representing either the potential event causes, potential event effects, or both. The equivalent event sub-sequences are then merged (blue boxes in Figure 3), and the counters $C_{i,j}$ used in (3) are updated accordingly.

More specifically, let I_u, $u = 1, 2, \ldots$, be the sets of indices of the equivalent left event sub-sequences $\boldsymbol{e}_i$ representing possible causes, and $\mathcal{J}_v$, $v = 1, 2, \ldots$, are similar such index sets for the right event sub-sequences $\boldsymbol{e}_j$, representing the possible effects. The event sub-sequence counters in (3) are updated due to the left and the right merges as,

$$C'_{i,j} = \sum_{i \in I_u} C_{i,j}, \quad C'_{i,j} = \sum_{j \in \mathcal{J}_v} C_{i,j}.$$

Consequently, it then becomes much more likely that some pairs of the equivalent event sub-sequences have their conditional probability close to 1 (as estimated by their relative occurrences), so they can be assumed to be causally related by Definition 1. On the other hand, merging the equivalent event sub-sequences and aggregating the counters make it somewhat less likely that the condition of non-causal relationship in Definition 2 would be satisfied. These causal decisions are also greatly affected by a specific choice of the distance metric.

B. Matrix Profile Analysis of Event Time-Series

The canonical matrix profile effectively shows the minimum distances between constant length sequences, which are created by a sliding-window partitioning of the original time-series data. The distance calculations in the matrix profile are greatly optimized to allow processing of very long sequences of data. These calculations can be readily parallelized, for example, using a MapReduce algorithm. The matrix profile is mainly used to identify common patterns (motifs) as well as rare patterns (discords), and also to identify time instances when the distance-based sequence statistics have changed.

Even though the events are represented as categorical rather than numerical variables, the distance metrics (4a)-(4f) can be directly used in calculating the matrix profile of the event time-series. The choice of the actual distance metric strongly affects the resulting matrix profile, although less than one might expect. However, it is still useful to compare the matrix profiles for different values of the sequence lengths.

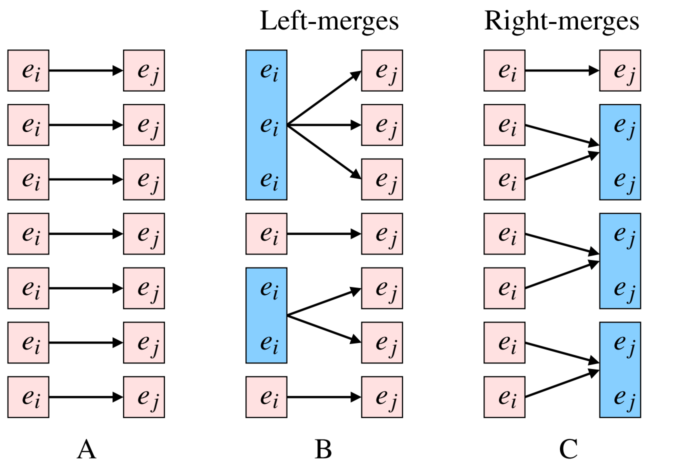


Figure 3. A: Original pairs of the unique event sub-sequences. B: Left-mergers of event sub-sequences as potential causes. C: Right-mergers of event sub-sequences as potential effects.

More importantly, the original matrix profile only displays the minimum distance values for each sliding-window sub-sequence. However, other statistics may also be useful. For example, the distribution of sub-sequence mutual distances provides a more global view, whereas the multiplicity of the smallest distances for each sub-sequence may be as informative in some applications as the actual smallest distance values.

IV. Case Study: Biochemical Reaction Networks

Biochemical reaction networks represent dynamic systems that undergo changes in copy counts (or, equivalently, concentrations) of chemical species due to chemical reaction events [23]. The number of chemical reactions is often much larger than the number of chemical species. The corresponding chemical kinetics can be stochastically described using a Chemical Master Equation (CME) [24]. The CME is usually solved by a Monte Carlo simulation [25], which tracks the time-evolution of the chemical species counts. More importantly, we assume a so-called well-stirred system, i.e., the spatial distribution and the diffusion of chemical molecules are ignored.

The models of chemical reaction systems may involve chemical species containing multiple binding sites [26]. Enumerating all chemical reactions for every binding site is impractical due to the combinatorial complexity of the resulting chemical reaction network. The network-free algorithms exploit the reaction (meta-) rules to effectively describe the groups of reactions without a need to enumerate all the reactions explicitly [27].

A. Numerical Experiments

Numerical experiments were obtained for an antigen receptor signaling regulating the activity and fate of the B-cells [28]. The corresponding model (referred to as BCR model) consists of 32 molecule types, 158 reaction rules, and 129 model parameters. The extracted full model contains 1,124 chemical species and 24,390 chemical reactions. The model was simulated in BioNetGen software [29], [30].

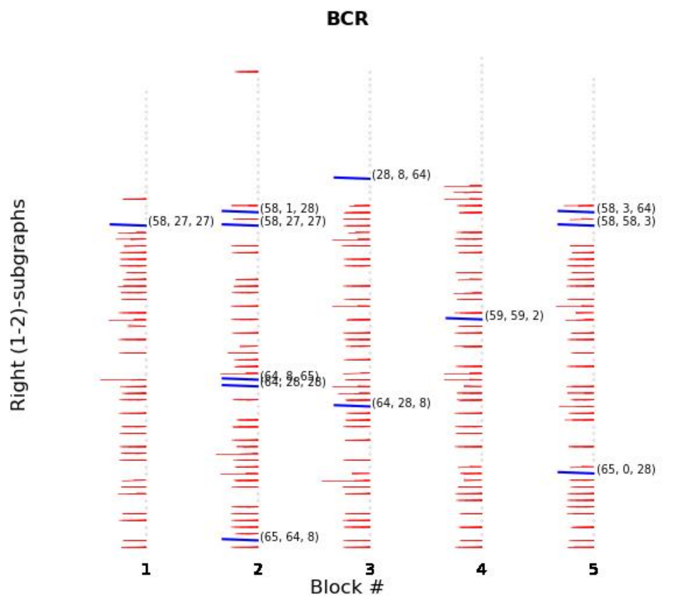


Figure 4. The estimated conditional probabilities $P_{j,i}$ of $N = (1+2)$ event sub-sequences. Black: $P_{j,i} < 0.1$, Red: $P_{j,i} < 0.9$, Blue: $P_{j,i} \geq 0.9$.

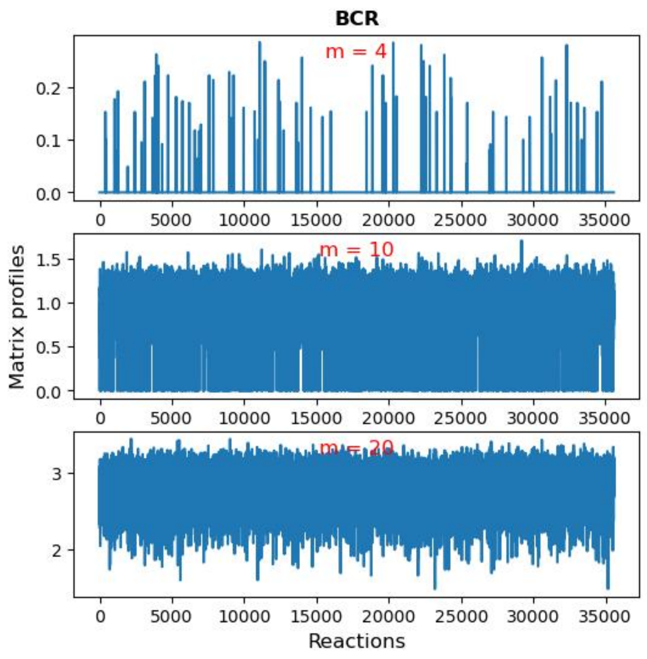


Figure 5. A canonical matrix profile assuming the minimum distances (4a), for sub-sequences of 4, 10 and 20 reaction events.

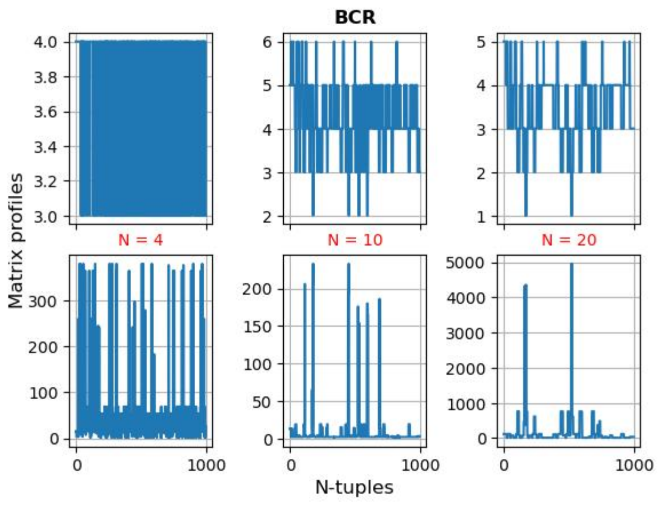


Figure 6. The matrix profile of the maximum distances (4f) (top-row) and their multiplicity (bottom-row) for sub-sequences of 4, 10 and 20 events.

BioNetGen is an open source software offering its own model description language to specify chemical reaction systems. The model description file is processed by a Perl script in order to generate the more complete system model given in System Biology Markup Language (SBML). The SBML file is then simulated in NFSim [31]; an open source software written in C++ [32]. We have modified NFSim to enable recording of the history of all reaction events in the course of the simulations. The trajectories of chemical species counts were simply discarded. The generated event time-series were processed, and visualized by the custom-made scripts written in Python. The overall process of performing the simulations, processing the event time-series, and generating the plots was fully automated mainly using the Bash scripts.

Simulating the BCR model over 100 simulation seconds resulted in 3,634,390 reaction events involving 35 reaction types. The reaction events can be naturally divided into 100 blocks over one second intervals. The sliding-window event sub-sequences were then formed and processed. The distinct frequencies of occurrence of the event N-tuples were observed, and they allow their clustering into multiple distinct classes.

Due to space limitations, only the following three plots are shown. Figure 4 visualizes the estimated reverse conditional probabilities, $\Pr(\boldsymbol{e}_i|\boldsymbol{e}_j)$ for the first five blocks (i.e., for the events $\boldsymbol{e}_i$ occurring before $\boldsymbol{e}_j$), assuming $|\boldsymbol{e}_j| = 2$ and $|\boldsymbol{e}_i| = 1$, i.e., $N = 3$. The reactions in each column in Figure 4 have the same ordering to indicate that some event patterns can be considered causal (according to our Definitions 1 and 2) in some blocks, but not in other blocks. Furthermore, in Figure 4, the right event sub-sequences were combined assuming the sub-sequence equivalences with the distance metric (4a). The line coloring is described in the caption of Figure 4.

Figure 5 depicts a canonical matrix profile assuming sub-sequences of 4, 10 and 20 events. The profile was calculated using the Python module stumpy [33]. It can be seen that the profile becomes more dense, and its mean moves away from zero with the increasing sub-sequence length. Finally, Figure 6 shows the matrix profile assuming the maximum (instead of minimum) values of the distance metric (4f) between any pair of sub-sequences of 4, 10 and 20 events, respectively (top-row), and the multiplicity of these maximum distance values (bottom-row). This illustrates how the choice of the distance metric greatly affects the shape of the matrix profile.

V. Discussion and Conclusion

The sliding-window event sequences were split into the left and the right event sub-sequences. The causality has been defined here as the pairs of nearly certain or nearly uncertain event sequences. The level of certainty can be evaluated empirically by measuring the corresponding conditional probabilities. Since ordering of events is locally irrelevant, it is useful to transform the event sub-sequences into event (multi-) sets, for which various distance metrics can be defined.

The distance metrics can be utilized to obtain the matrix profiles of event time-series. Our numerical experiments demonstrate that matrix profile is a rather general and flexible framework for analyzing numerical as well as categorical time-series, and conveniently visualizing their statistics.

Even though this paper focuses on analyzing the short sequences of consecutive events, the events neither have to be consecutive, nor short. However, assuming non-consecutive events make the pattern space to be combinatorially much larger, and the longer the event sequences, the less likely it is to identify those that can be considered to be statistically certain or uncertain. The causality analysis may also involve both the events and the observations. This can lead to explainable Monte Carlo simulations, provided that causally related (or unrelated) sub-sequences are identified and properly interpreted in a given domain [34], which can be challenging.

The simulation software adopted and the programming scripts produced to analyze state-space models of biochemical reaction networks allow fully automated processing of the recorded event time-series. It allows generating a large number of diverse plots for different models across different numerical experiments, although automated interpretation of the identified causal events may again be rather challenging.

Acknowledgment

This research was funded by a start-up research grant provided by Zhejiang University.

References

[1] J. Pearl, "The seven tools of causal inference, with reflections on machine learning," *Communications of the ACM*, vol. 62, no. 3, pp. 54–60, Mar. 2019.

[2] ——, "Causal inference in statistics: An overview," *Statistics Surveys*, vol. 3, pp. 96–146, 2009.

[3] K. Hlaváčková-Schindlera, M. Paluš, M. Vejmelka, and J. Bhattacharya, "Causality detection based on information-theoretic approaches in time series analysis," *Physics Reports*, vol. 441, pp. 1–46, 2007.

[4] M. Eichler, *Causal Inference in Time Series Analysis*. Wiley, 2012, ch. In Causality (eds W.A. Shewhart et al.).

[5] A. Papana, C. Kyrtsou, D. Kugiumtzis, and C. Diks, "Simulation study of direct causality measures in multivariate time series," *Entropy*, vol. 15, pp. 2635–2661, 2013.

[6] D. Lopez-Paz, K. Muandet, B. Schölkopf, and I. Tolstikhin, "Towards a learning theory of cause-effect inference," in *ICML'15*, vol. 37, 2015, pp. 1452–1461.

[7] Y. Chikahara and A. Fujino, "Causal inference in time series via supervised learning," in *IJCAI'18*, 2018, pp. 2042–2048.

[8] R. M. et al., "Causal inference for time series analysis: problems, methods and evaluation," *Knowledge and Information Systems*, vol. 63, no. 12, pp. 3041–3085, Dec. 2021.

[9] Y. Hmamouche, A. Casali, and L. Lakhal, "A causality based feature selection approach for multivariate time series forecasting," in *DBKDA*, 2017, pp. 1–6.

[10] Y. Huang and S. Kleinberg, "Fast and accurate causal inference from time series data," in *Int. Florida AI Research Soc. Conf.*, 2015, pp. 1–6.

[11] J. R. at al., "Inferring causation from time series in Earth system sciences," *Nature Communications*, vol. 10, no. 2553, pp. 1–13, Jun. 2019.

[12] R. J. Prill, R. Vogel, G. A. Cecchi, G. Altan-Bonnet, and G. Stolovitzky, "Noise-driven causal inference in biomolecular networks," *PLoS ONE*, vol. 10, no. 6, e0125777, pp. 1–16, Jun. 2015.

[13] P. Samartsidis, S. R. Seaman, A. M. Presanis, M. Hickman, and D. D. Angelis, "Assessing the causal effect of binary interventions from observational panel data with few treated units," *Statistical Science*, vol. 34, no. 3, pp. 486–503, Aug. 2019.

[14] K. H. Brodersen, F. Gallusser, J. Koehler, N. Remy, and S. L. Scott, "Inferring causal impact using Bayesian structural time-series models," *The Annals of Applied Statistics*, vol. 9, no. 1, pp. 247–274, 2015.

[15] S. Löwe, D. Madras, R. Zemel, and M. Welling, "Amortized causal discovery: Learning to infer causal graphs from time-series data," Feb. 2022, arXiv:2006.10833 [cs.LG].

[16] D. Mønster, R. Fusaroli, K. Tylén, A. Roepstorff, and J. F. Sherson, "Causal inference from noisy time-series data – testing the convergent cross-mapping algorithm in the presence of noise and external influence," *Future Generation Computer Systems*, vol. 73, pp. 52–62, Aug. 2017.

[17] K. W. Soo and B. M. Rottman, "Causal strength induction from time series data," *Journal of Experimental Psychology: General*, vol. 147, no. 4, pp. 485–513, 2018.

[18] B. gong Zhang, W. Li, Y. Shi, X. Liu, and L. Chen, "Detecting causality from short time-series data based on prediction of topologically equivalent attractors," *BMC Systems Biology*, vol. 11, no. 128, pp. 141–150, 2017.

[19] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures," in *KDD'03*, 2003, pp. 216–225.

[20] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, and A. Pulvirenti, *Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining*. IntechOpen, London, UK, 2012, pp. 71–96.

[21] G. E. Batista, E. J. Keogh, O. M. Tataw, and V. M. Souza, "CID: an efficient complexity-invariant distance for time series," *Data Mining and Knowledge Discovery*, vol. 28, no. 3, pp. 634–669, May 2014.

[22] A. Zielezinski, S. Vinga, J. Almeida, and W. M. Karlowski, "Alignment-free sequence comparison: benefits, applications, and tools," *Genome Biology*, vol. 18, no. 186, pp. 1–17, 2017.

[23] P. Loskot, K. Atitey, and L. Mihaylova, "Comprehensive review of models and methods for inferences in bio-chemical reaction networks," *Frontiers in Genetics*, vol. 10, no. 549, pp. 1–29, 2019.

[24] V. Wolf, R. Goel, M. Mateescu, and T. A. Henzinger, "Solving research article the chemical master equation using sliding windows," *BMC Systems Biology*, vol. 4, no. 42, pp. 1–19, 2010.

[25] D. T. Gillespie, "Stochastic simulation of chemical kinetics," *Annual Review of Physical Chemistry*, vol. 58, pp. 35–55, 2007.

[26] D. J. Klinke II and S. D. Finley, "Timescale analysis of rule-based biochemical reaction networks," *Biotechnology Prog.*, vol. 28, no. 1, pp. 33–44, 2012.

[27] A. Gupta and P. Mendes, "An overview of network-based and -free approaches for stochastic simulation of biochemical systems," *Computation*, vol. 6, no. 1, pp. 1–20, 2018.

[28] D. Barua, W. S. Hlavacek, and T. Lipniacki, "A computational model for early events in b cell antigen receptor signaling: analysis of the roles of lyn and fyn," *J. Immunology*, vol. 189, no. 2, pp. 646–658, Jul. 2012.

[29] L. A. H. et al., "Bionetgen 2.2: advances in rule-based modeling," *Bioinformatics*, vol. 32, no. 21, pp. 3366–3368, Nov. 2016.

[30] "BioNetGen source code," https://github.com/RuleWorld/bionetgen, accessed: January 2022.

[31] M. W. Sneddon, J. R. Faeder, and T. Emonet, "Efficient modeling, simulation and coarse-graining of biological complexity with NFsim," *Nature Methods*, vol. 8, no. 2, pp. 177–183, Feb. 2011.

[32] "NFsim source code," https://github.com/msneddon/nfsim, accessed: January 2022.

[33] "Stumpy python library," https://stumpy.readthedocs.io/, accessed: January 2022.

[34] Y. P. Park and M. Kellis, "CoCoA-diff: counterfactual inference for single-cell gene expression," *Genome Biology*, vol. 22, no. 228, pp. 1–23, 2021.