A Short Tutorial for Multivariate Time Series Explanation using tsCaptum

Davide Italo Serramazza, Thach Le Nguyen, Georgiana Ifrim
School of Computer Science, University College Dublin, Ireland

Abstract

tsCaptum is a Python library that enables explainability for time series classification and regression using saliency maps (i.e., attribution-based explanation). It bridges the gap between popular time series frameworks (e.g., aeon, sktime, sklearn) and explanation libraries like Captum. tsCaptum tackles the computational complexity of explaining long time series by employing chunking techniques, significantly reducing the number of model evaluations required. This allows users to easily apply Captum explainers to any univariate or multivariate time series model or pipeline built using the aforementioned frameworks. tsCaptum is readily available on pypi.org and can be installed with a simple "pip install tsCaptum" command.

Keywords

Time Series Classification; Time Series Regression; Explanation; Attribution; Saliency Maps; XAI; Python

Code metadata

Nr.	Code metadata description	Please fill in this column
C1	Current code version	*v0.0.5*
C2	Permanent link to code/repository used for this code version	*https://github.com/mlgig/tscaptum*
C3	Permanent link to Reproducible Capsule	
C4	Legal Code License	*MIT License*
C5	Code versioning system used	*git*
C6	Software code languages, tools, and services used	*Python pip*
C7	Compilation requirements, operating environments & dependencies	torch $\geq$ 2.0.0, captum, aeon $\geq$ 0.10.0, scikit-learn, tqdm, openpyxl, matplotlib
C8	If available Link to developer documentation/manual	
C9	Support email for questions	davide.serrramazza@ucdconnect.ie

1. Introduction

A time series (TS) is a sequence of values recorded over time. It is *univariate* (UTS) when only one value is recorded at each time point or *multivariate* (MTS) when multiple values are recorded at the same time. Even though recent TS models achieve good performance regarding accuracy, they are often challenging to explain due to their intrinsic complexity. This highlights the necessity for the Explainable Artificial Intelligence (XAI) field. One very common approach is to compute *saliency maps* or *attributions*, i.e. a vector in the UTS case or a matrix (for MTS) of weights where each entry represents the importance played in the prediction by the corresponding element of the TS. A common visual representation technique for a saliency map is a heat map, i.e., a plot of the raw TS that uses colors to visually represent the importance of each time point.

Recently, many *model-specific* explanations methods have been developed for TS, with most of them tailored for CNN models. Nevertheless, CNNs often do not match the accuracy of other SOTA classifiers (e.g., ROCKET) [7]. As opposed to model-specific explanations, *perturbation-based* explanations are model agnostic thus they can be applied to any model. Our library **tsCaptum** facilitates the usage of these explainers for TS tasks dealing with some of the main problems due to why they are not commonly used for TS. It also implements some computational steps shown to be important for effectively and efficiently using time series explanation [10, 11].

2. Description

tsCaptum bridges the gap between the explanation framework Captum [4] and TS downstream frameworks, such as sktime [5] or aeon [6]. Captum, meant for PyTorch models [9], is arguably one of the more complete frameworks for computing explanations. **tsCaptum** handles all necessary steps for connecting the two frameworks, such as:

- Automatically detecting whether the current task is a classification or regression one.
- Extracting probabilities for the classification case while using the raw predicted output in the regression case, as required by Captum.
- Extracting probabilities even in the case of a pipeline transformation followed by a Ridge Classifier (relying on the distance of the current sample(s) from the decision hyperplane).

- Checking the correctness of provided data, e.g. dimensions of input, presence of labels for classification and absence for regression tasks.
- Converting back and forth from numpy or pandas representation to pytorch tensor.
- Internally initializing a torch DataLoader to divide the samples to be explained into mini-batches achieving a better computational efficiency.
- Normalizing the attribution in the $[-1, 1]$ range (optional).
- Plotting functionality: a heatmap plot that uses the computed attribution values and a color map to highlight the importance of each time series point (the color varies from red which means positive importance, e.g., this point contributes to increasing the predicted value, to blue which means negative importance, e.g., this point contributes to decreasing the predicted value).

In addition to these steps, tsCaptum can segment the data into equal-length **chunks** or segments (by default this number is set to 10 for each channel). Segmenting has the effect of lowering the number of features to be explained and is key for speeding up the attribution computation. Previous works [10] estimated a gain of 2-3 orders of magnitude due to chunking, which is crucial to make a computationally demanding algorithm such as SHAP feasible to run for long time series.

2.1 Code Snippet

As a result of all the steps listed in the previous section which are automatically taken into account by the library, to use tsCaptum we only need a few lines of code as shown in the following pseudo-code snippet[1]:

```
from tsCaptum.explainers import Shapley_Value_Sampling as SHAP
from tsCaptum.visualization import plot_saliency_map_uni, plot_saliency_map_multi
# Assuming you already trained your model on (X_train, y_train)
# couple SHAP with the model
shap_explainer = SHAP(model)
#get the explanations from the model, labels are needed only for classification
saliency_maps = shap_explainer.explain(X_test, labels=y_test)
# plot saliency_maps as heat maps
for (specimen,explanation) in (X_test,saliency_maps):
    plot_saliency_map_uni/multi(sample=specimen, attribution=explanation)
```

3. Use Cases

In this section we provide two different use cases computed using a laptop having an Intel Core i7-12700H processor and 32GB of RAM. The first use case focuses on explanation for **Multivariate Time Series Classification** and uses the Military Press (MP) dataset[2] [12] which tracks different body parts positions in a video while executing this exercise. The data has 8 channels (161 length) corresponding to tracking the upper body parts and the class labels are the type of correct/incorrect execution (e.g., Normal or Reduced-Range categorical targets). We trained a standard ROCKET pipeline using the aeon and sklearn libraries (ROCKET transform, standard scaler and Logistic Regression) achieving an accuracy of 0.82. We then used tsCaptum to explain the test set samples using chunking for faster explanation computation and FeatureAblation. The runtime was less than $4s$ for explaining one sample, while using a larger batch size, specifically of 8 samples, the efficiency increases requiring a total time of $8s$ (without chunking running time is respectively $56s$ and $151s$ for a single sample and a batch of 8). Figure 1 shows a heat map for one correctly predicted sample of the *Reduced-Range* class. According to the domain knowledge, the first and second channels, i.e., Right Shoulder and Right Elbow are the most important ones, while the last two channels Left and Right Hips are the least important.

[1] More details on code use with examples: https://github.com/mlgig/tscaptum/blob/main/examples_tscaptum.ipynb
[2] https://github.com/mlgig/Video_vs_Shimmer_ECML_2023

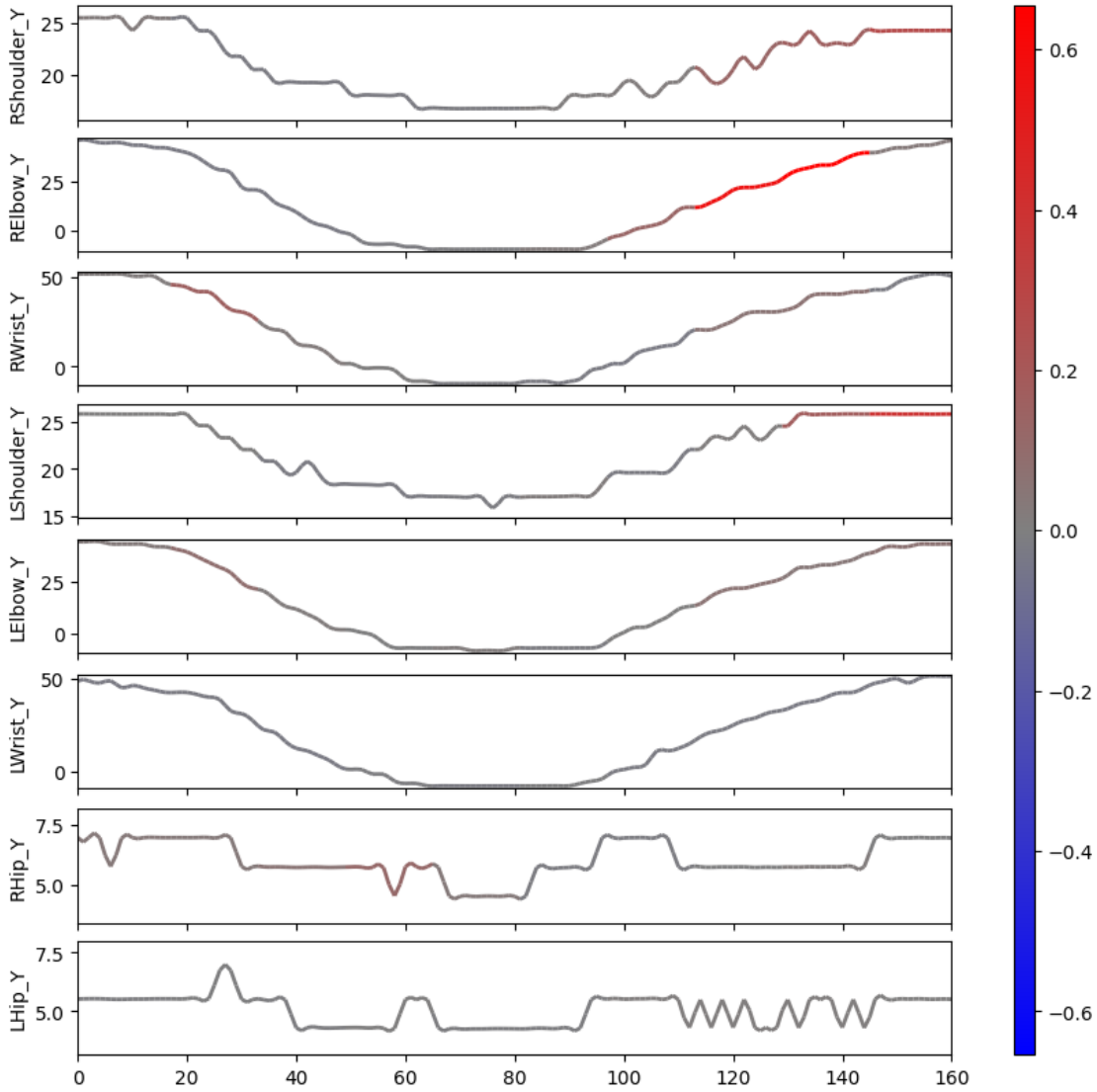

Figure 1: Use Case1: Explanation of Multivariate Time Series Classification using tsCaptum. Explanation of a sample from the Military Press dataset, correctly classified as Reduced Range. Saliency map computed using the Feature Ablation algorithm.

The second use case focuses on explanation for **Multivariate Time Series Regression** using the Vista Milk data challenge dataset[3] 2024 [2]. This dataset was used in an international competition and contains 300 channels representing the spectrum of pixels extracted from hyperspectral images of samples of cow milk. The task here is to predict the lactose content of each sample expressed as mg/ml (numeric target). For this use case we took the approach of the competition's winner which was to shrink the 300 channels to 3 channels (3424 length) representing the $25th, 50th, 75th$ percentiles of the original datasets. We trained a pipeline using ROCKET and feature selection steps from the aeon and sklearn libraries. We then used tsCaptum to explain the 8 test samples. In particular, we computed the Shapely Value Sampling (SHAP) explanation using chunking (running time of 7 and 15 minutes respectively for one sample and a batch of 8 samples). In agreement with what was verified during the competition, for each test sample the SHAP explainer pointed out the $75th$ percentile as the most important channel. Figure 2 shows a heat map for a sample with a correct prediction of lactose content equal to $60mg/ml$. The highlighted segments are mostly observed from the 75th-percentile channel (at the top).

[3] https://github.com/mlgig/VM-challenge-lactose-prediction-2024

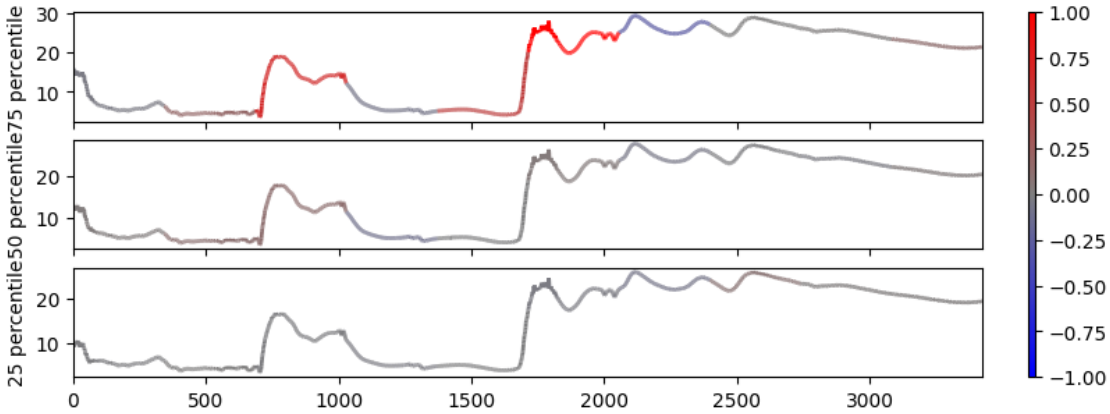

Figure 2: Use Case2: Explanation of Multivariate Time Series Regression using tsCaptum. Explanation of a sample from the Vista Milk challenge dataset, correctly predicted as 60 mg/ml of lactose content. Saliency map computed using the Shapley Value Sampling algorithm.

4. Software Impact

The tsCaptum library has the potential to have an impact on the (M)TS XAI field: its **ease of usage** allows the user of the library to focus on the final results rather than the technical details about how to compute a saliency map. On top of that, the **chunking** functionality massively reduces the computation time, allowing very long TS with potentially many channels, to be explained in a reasonable amount of time. Additionally, the user can change the baseline for computing the explanation, which is an important consideration for many attribution-based methods [13].

Research: The tsCaptum library can be easily used by many researchers as a tool to ease their work due to its simplicity of use as done for example by Aderinola et al. [1]. In this work the authors used tsCaptum to explain fall detection classifications on synthetic as well as real-fall datasets. In addition, the winning team of the VistaMilk 2024 data challenge[4] has used tsCaptum to provide explanations for the winning time series regression model. By using tsCaptum the participants were able to efficiently compute SHAP-based attributions for a ROCKET-pipeline on long time series (3424 length).

Furthermore, the library is designed such that it can be easily extendable with other features, e.g., different backgrounds/baselines to perturb the data or different segmentation methods beyond equal-length segmentation. This is an opportunity for the community to easily adapt the library for their purposes and to further contribute to its development. Lastly, we can also measure impact through the early interest in this library by using *PyPI Stats*, a website that monitors the amount of downloads from PyPI of each library. Despite its recent release, the community downloaded tsCaptum more than 180 times in the last month. This number will likely increase once members of the community become more aware of the capabilities of this library.

Industry: tsCaptum can be easily applied post-training to any large-scale model or pipeline to get explanations for its predictions. This is a key necessity in the industrial sector as a system cannot be fully automated without a complete understanding of how it works. This is also a legal requirement according to recent legislation such as the recent EU AI act [3]. Moreover, some companies have a huge stream of MTS, e.g., coming from different sensors that are then processed using downstream tasks such as regression or classification. For example cloud or satellite data providers [8] work with very large scale data sampled at high frequency, and require tools to reduce the data and process it in close to real-time, as well as provide explanations for the data and model predictions. As shown in [10] it is possible to use attribution-based methods for effective channel selection. Applying tsCaptum for such a task with such huge amount of data has the potential to reduce both the data storage and the model prediction costs.

[4] https://github.com/mlgig/VM-challenge-lactose-prediction-2024

Acknowledgements

This work was funded by Science Foundation Ireland through the SFI Centre for Research Training in Machine Learning (18/CRT/6183) and the Insight Centre for Data Analytics (12/RC/2289_P2). For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] T. B. Aderinola, L. Palmerini, I. D'Ascanio, L. Chiari, J. Klenk, C. Becker, B. Caulfield, and G. Ifrim. Accurate and efficient real-world fall detection using time series techniques. *International Workshop on Advanced Analytics and Learning on Temporal Data*, 2024.

[2] V. Caponigro, F. Marini, A. G. Scannell, and A. A. Gowen. Single-drop technique for lactose prediction in dry milk on metallic surfaces: Comparison of raman, ft – nir, and ft – mir spectral imaging. *Food Control*, 144:109351, 2023.

[3] S. G. J, T. S, H. T. I, F. L. D, C. V, G. G. E, J. H, H. R, F. Y. D, and P. C. Ai watch: Artificial intelligence standardisation landscape update. (KJ-NA-31-343-EN-N (online)), 2023.

[4] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020.

[5] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király. sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*, 2019.

[6] M. Middlehurst, A. Ismail-Fawaz, A. Guillaume, C. Holder, D. Guijo-Rubio, G. Bulatova, L. Tsaprounis, L. Mentel, M. Walter, P. Schäfer, et al. aeon: a python toolkit for learning from time series. *Journal of Machine Learning Research*, 25(289):1–10, 2024.

[7] M. Middlehurst, P. Schäfer, and A. Bagnall. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, 2024.

[8] J. Murphy, F. Wang, M. D. Mazhar Qureshi, and B. M. Namee. From ground to orbit: Enhancing satellite autonomy with ai-powered anomaly detection. *Small Satellite Conference*, 2024.

[9] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[10] D. I. Serramazza, T. L. Nguyen, and G. Ifrim. Improving the evaluation and actionability of explanation methods for multivariate time series classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 177–195. Springer, 2024.

[11] D. I. Serramazza, T. T. Nguyen, T. Le Nguyen, and G. Ifrim. Evaluating explanation methods for multivariate time series classification. In *International Workshop on Advanced Analytics and Learning on Temporal Data*, pages 159–175. Springer, 2023.

[12] A. Singh, A. Bevilacqua, T. L. Nguyen, F. Hu, K. McGuinness, M. O'Reilly, D. Whelan, B. Caulfield, and G. Ifrim. Fast and robust video-based exercise classification via body pose tracking and scalable multivariate time series classifiers. *Data Mining and Knowledge Discovery*, 37(2):873–912, 2023.

[13] P. Sturmfels, S. Lundberg, and S.-I. Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1), Jan. 2020.