PySpArX - A Python library for generating Sparse Argumentative eXplanations for neural networks

Ioana Mihailescu[*†] icm20@ic.ac.uk
Alan Weng[*†] aw920@ic.ac.uk
Siddharth Sharma[*†] ss2120@ic.ac.uk
Mihnea Ghitu[*†] mg2720@ic.ac.uk
Dilshaan Grewal[*†] dsg20@ic.ac.uk
Khemi Chew[*†] kjc20@ic.ac.uk
Hamed Ayoobi[‡†] h.ayoobi@ic.ac.uk
Nico Potyka[‡†] n.potyka@ic.ac.uk
Francesca Toni[‡†] f.toni@ic.ac.uk

1 Introduction

The increasing use of data-centric techniques in autonomous intelligent systems in areas such as healthcare, automotive, and finance raises concerns about their predictions' reliability, fairness, and transparency. In the case of systems empowered by black-box machine learning models, the user has no access to the inner mechanics of the systems, thus making it essential to obtain a human-readable interface providing explanations of the models' operations.

Explainable Artificial Intelligence (XAI) techniques aim to provide user-friendly explanations for models. SpArX (Sparse Argumentative Explanations for Neural Networks [2]) is an XAI method that faithfully explains the mechanics of neural networks in the form of Multi-Layer Perceptrons (MLPs) by exploiting the correspondence between MLPs and Quantitative Argumentation Frameworks (QAF) [11]. Intuitively, QAFs are a form of argumentation framework amounting to weighted graphs that hold arguments as their nodes and relations (of attack or support) between arguments as their edges. To provide an explainable interface of MLPs using QAFs, the MLPs are first sparsified, through a per-layer clustering of neurons based on output-similarity. The sparsified MLPs are then transformed into QAFs and then global (input-independent) and local (within the vicinity of a target input) explanations are generated, while faithfully preserving the mechanics of the original MLPs.

Building upon SpArX, we have provided users with tools that help them prove and understand how MLPs, trained on specific datasets, work. The tools allow users to customize explanations to their needs.

2 System Overview

The architecture comprises of three primary components, namely a Python library, a GUI for generating models, and a web-based application for visualization. We have provided technical users with a Python

[*]Equal contributors.
[†]Imperial College London, London, United Kingdom
[‡]Supervisors.

Submitted to:
ICLP 2023 System
Demonstrations Track
© I. Mihailescu, A. Weng, S. Sharma, M. Ghitu, D. Grewal, K. Chew, H. Ayoobi, N. Potyka & F. Toni
This work is licensed under the
Creative Commons Attribution License.

Pip package [7] in which a modular implementation of SpArX can be found, as well as a GUI [5] in which they can input their own dataset, and generate a customizable MLP. This is forwarded to our framework pipeline, which performs the aforementioned transformations (sparsification and mapping onto a QAF). The final QAF is then stored in a database and made available online to other users via a shareable link [10] where it can be visualised.

2.1 sparx-lib

The Python Pip package, sparx-lib [7], is a custom library developed to handle the heavy computations required for generating SpArX explanations. It is designed with a modular architecture that allows for the flexible integration of different components of the SpArX algorithm. It enables users to generate explanations for a variety of models, which can be either user-specified, with a dataset, or imported. Tensorflow and Keras are utilized to develop and train custom models. Users can generate two types of visualisations: pure JSON, which can be viewed as text or parsed, or a GUI using NetworkX and Bokeh. We used a documentation generation tool called Sphinx to create documentation for the sparx-lib pip package [6].

2.2 Model generation GUI

The model generation GUI [5] is a tool developed using the React framework. It provides a platform for administrators or technical users to design MLPs for a selected dataset, generate explanations, and store visualizations in a remote database for access by other users. The model generation process is currently performed locally due to the computational demands involved in training the model and calculating explanations. The requirements and instructions for its execution are provided in the **SpArX-GUI repository** [5].

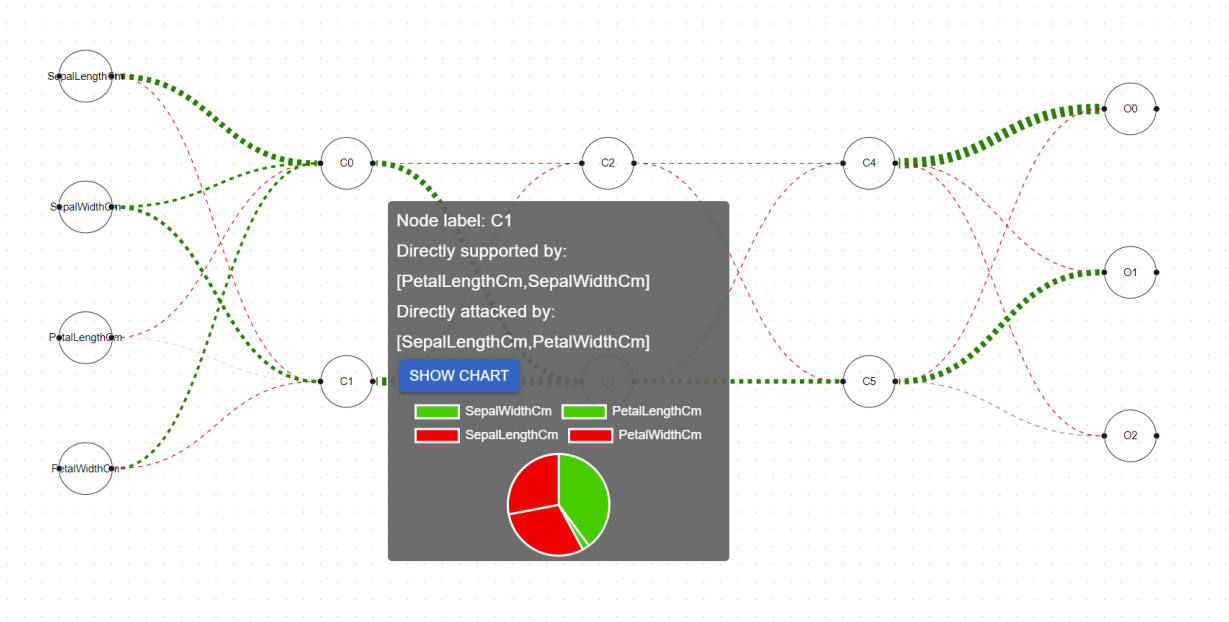


Figure 1: Web app visualisation [10] for the Iris dataset [3]

2.3 Visualisation

The customizable visualization component [9] provides features such as:

- displaying node and edge information such as attackers/supporters and their weight using hover-cards, and visualising node dependencies using charts;
- filtering graph information by progressively rendering layers and restricting the nodes shown;
- sharing generated visualisations via a link/QR code.

We have utilised React with React Flow to create interactive graphs that can be easily modified (See Figure 1). Our server has been deployed on the web to enhance accessibility to the app. Administrators can simply share a link to the dedicated visualisation page with other non-technical users.

3 Demo requirements

To run the project, follow the instructions in the README file from the SpArX-GUI repository [5]. The model generation GUI accepts datasets in CSV format where the first line represents the name of the parameter/column. A test dataset is provided in the SpArX-GUI repository under `data/iris.data`. For demo purposes, an already generated visualisation of a SpArX explanation can be found online [10]. A demonstration video can be found at [4].

References

[1] H. Ayoobi, N. Potyka & F. Toni (2022): *SpArX legacy code*. Available at `https://github.com/H-Ayoobi/SpArX`.

[2] H. Ayoobi, N. Potyka & F. Toni (2023): *SpArX: Sparse Argumentative Explanations for Neural Networks*, doi:10.48550/ARXIV.2301.09559. Available at `https://arxiv.org/abs/2301.09559`.

[3] Dheeru Dua & Casey Graff (2017): *UCI Machine Learning Repository*. Available at `http://archive.ics.uci.edu/ml`.

[4] I. Mihailescu (2023): *PySpArX - A library for SpArX explanations of Neural Networks - Demo*. Available at `https://youtu.be/OJHl4zegFvU`.

[5] I. Mihailescu, K. Chew, M. Ghitu, S. Sharma, D. Grewal & A. Weng (2022): *SpArX-GUI*. Available at `https://github.com/SpArX-Group-10/SpArX-GUI`.

[6] I. Mihailescu, K. Chew, M. Ghitu, S. Sharma, D. Grewal & A. Weng (2022): *Sparx-lib documentation*. Available at `https://sparx-group-10.github.io/SpArX/`.

[7] I. Mihailescu, K. Chew, M. Ghitu, S. Sharma, D. Grewal & A. Weng (2022): *Sparx-lib pip package*. Available at `https://pypi.org/project/sparx-lib/`.

[8] I. Mihailescu, K. Chew, M. Ghitu, S. Sharma, D. Grewal & A. Weng (2022): *SpArX-lib Repository*. Available at `https://github.com/SpArX-Group-10/SpArX`.

[9] I. Mihailescu, K. Chew, M. Ghitu, S. Sharma, D. Grewal & A. Weng (2022): *SpArX-Visualisation*. Available at `https://github.com/SpArX-Group-10/SpArX-Visualisation`.

[10] I. Mihailescu, K. Chew, M. Ghitu, S. Sharma, D. Grewal & A. Weng (2022): *SpArX-Visualisation Web app example*. Available at `https://sparx-vis.herokuapp.com/63ac3bd37963d9af247d8102`.

[11] Nico Potyka (2021): *Interpreting Neural Networks as Quantitative Argumentation Frameworks*. In: *AAAI Conference on Artificial Intelligence, (AAAI)*, AAAI Press, pp. 6463–6470.