LENs: a Python library for Logic Explained Networks

A Preprint

Pietro Barbiero
Department of Computer Science
University of Cambridge
Cambridge, UK
pb737@cam.ac.uk

Gabriele Ciravegna
Department of Computer Science
Universita' di Siena
Siena, IT
gabriele.ciravegna@unifi.it

Dobrik Georgiev
Department of Computer Science
University of Cambridge
Cambridge, UK
dgg30@cam.ac.uk

Franscesco Giannini
Department of Computer Science
Universita' di Siena
Siena, IT
fgiannini@diism.unisi.it

May 26, 2021

Abstract

LENs is a Python module integrating a variety of state-of-the-art approaches to provide logic explanations from neural networks. This package focuses on bringing these methods to non-specialists. It has minimal dependencies and it is distributed under the Apache 2.0 licence allowing both academic and commercial use. Source code and documentation can be downloaded from the github repository: https://github.com/pietrobarbiero/logic_explainer_networks.

Keywords Python · logic · deep learning · XAI

1 Introduction

The lack of transparency in the decision process of some machine learning models, such as neural networks, limits their application in many safety-critical domains (Chander et al., 2018). Employing black-box[1] models may be unacceptable in contexts such as industry, medicine or courts, where the potential economical or ethical repercussions are calling for lawmakers to discourage from a reckless application of non-interpretable models (EUGDPR, 2017; Law, 10; Goddard, 2017; Gunning, 2017). As a consequence, research in Explainable Artificial Intelligence (XAI) has become strategic and has been massively encouraged, leading to the development of a variety of techniques that aim at explaining black-box models (Das and Rad, 2020; Brundage et al., 2020) or at developing effective interpretable models (Carvalho et al., 2019; Rudin, 2019).

The need for high-quality human-friendly explanations is one of the main reasons why concept-based explanations are receiving ever-growing consideration. Explanations are given in terms of human-understandable symbols (the *concepts*) rather than raw features such as pixels or characters (Kim et al., 2018; Ghorbani et al., 2019; Koh et al., 2020). As a consequence, they seem more suitable to serve many strategic human purposes such as decision making tasks. For instance, a concept-based explanation may describe a high-level category through its attributes as in "a *human* has *hands* and a *head*". While collecting high-quality evidences for an explanation is a common feature of concept-based techniques, there are very few approaches formulating hypothesis and even less providing synthetic descriptions whose validity can be quantitatively assessed (Das and Rad, 2020). A possible solution is to rely on a formal language that

[1]In the context of this paper, a black-box classifier is any classifier that cannot provide human understandable explanations about its decisions

is very expressive and closely related to natural language and reasoning, such as First-Order Logic (FOL). A FOL explanation can be considered a special kind of a concept-based explanation, where the description is given in terms of logical predicates, connectives and quantifiers, such as "$\forall x: \textit{is\_human}(x) \rightarrow \textit{has\_hands}(x) \wedge \textit{has\_head}(x)$". However, FOL formulas generally express much more complex relationships among the concepts involved in a certain explanation. Compared to other concept-based techniques, logic-based explanations provide many key advantages, that we briefly described in what follows. An explanation reported in FOL is a rigorous and unambiguous statement (CLARITY). This formal clarity may serve cognitive-behavioral purposes such as engendering trust, aiding bias identification, or taking actions/decisions. For instance, dropping quantifiers and variables for simplicity, the formula "*snow* $\wedge$ *tree* $\leftrightarrow$ *wolf*" may easily outline the presence of a bias in the collection of training data. Different logic-based explanations can be combined to describe groups of observations or global phenomena (MODULARITY). For instance, for an image showing only the face of a person, an explanation could be "(*nose* $\wedge$ *lips*) $\rightarrow$ *human*", while for another image showing a person from behind a valid explanation could be "(*feet* $\wedge$ *hair* $\wedge$ *ears*) $\rightarrow$ *human*". The two local explanations can be combined into "(*nose* $\wedge$ *lips*) $\vee$ (*feet* $\wedge$ *hair* $\wedge$ *ears*) $\rightarrow$ *human*". The quality of logic-based explanations can be quantitatively measured to check their correctness and completeness (MEASURABILITY). For instance, once the explanation "(*nose* $\wedge$ *lips*) $\vee$ (*feet* $\wedge$ *hair* $\wedge$ *ears*)" is extracted for the class *human*, this logic formula can be applied on a test set to check its generality in terms of quantitative metrics like accuracy, fidelity and consistency. Finally, FOL-based explanations can be rewritten in different equivalent forms such as in *Disjunctive Normal Form* (DNF) and *Conjunctive Normal Form* (CNF) (SIMPLIFIABILITY). Further, techniques such as the Quine–McCluskey algorithm can be used to compact and simplify logic explanations (McColl, 1878; Quine, 1952; McCluskey, 1956). For instance, the explanation "(*person* $\wedge$ *nose*) $\vee$ ($\neg$*person* $\wedge$ *nose*)" can be easily simplified in "*nose*".

This work presents a Python library for XAI enabling neural networks to *solve and explain* a categorical learning problem integrating elements from deep learning and logic. Differently from vanilla neural architectures, these models can be directly interpreted by means of a set of FOL formulas. In order to implement such a property, such models require their inputs to represent the activation scores of human-understandable concepts. Then, specifically designed learning objectives allow them to make predictions in a way that is well suited for providing FOL-based explanations that involve the input concepts. In order to reach this goal, LENs exploit parsimony criteria aimed at keeping their structure simple as described in recent works (Koh et al., 2020; Ciravegna et al., 2020b,a).

2 Background

Classification is the problem of identifying a set of categories $y \in Y \subset [0,1]^r$ an observation $x \in X \subset \mathbb{R}^d$ belongs to. A standard neural network is a black-box model $f : X \mapsto Y$ predicting for any sample $x \in X$ the corresponding class membership $\hat{y} \in Y$. In case the input features x are not easily interpretable (low-level feature, image pixels) concept-based classifiers have been introduced to predict class memberships Y from human-understandable categories (a.k.a. *concepts*) $C \subset [0,1]^k$: $f : C \mapsto Y$ to improve the understanding of black boxes and their decision process. Concepts can either correspond to the predictions of a classifier (i.e. $g : X \mapsto C$) (Koh et al., 2020) or simply to a re-scaling of the inputs space from the unbounded $\mathbb{R}^d$ to the unit interval $[0,1]^k$ such that input features can be treated as logic predicates. Concept-based classifiers improve human understanding as their input and output spaces consists of interpretable symbols.

Recent work on concept-based neural networks has led to the development of models like the ψ network (Ciravegna et al., 2020a), a concept-based classifier *explaining its own decision process*. The ψ network leverages the intermediate symbolic layer C to distill First-Order Logic formulas representing the learned map from C to Y. The model consists of a sequence of fully connected layers with sigmoid activations only. An $L1$-regularization and strong pruning strategy is applied to each layer of weights. This allows to compute a logic formula representing the activation of each node and, recursively, to generate concise explanations of the network predictions. This library implements the aforementioned learning criteria allowing a customized implementation of neural models providing logic explanations.

3 Core API

3.1 High-level APIs

The code library is designed with intuitive APIs requiring only a few lines of code to train and get explanations from the neural network as shown in the following code snippet 1 illustrating a generic example. The library supports two out-of-the-box explainable neural networks: the ψ network described in Ciravegna et al. (2020b) and multi-layer neural networks with rectified linear units. The APIs allow the definition of a model by specifying: (i) the number of input concepts `n_features`, (ii) the number of target categories `n_classes`, (iii) the number of hidden neurons `hidden_neurons` as a list of integers, (iv) the form of the loss function `loss`, and (v) hyperparameters such as the

Lagrangian multiplier `l1_weight`. The `fit` method receives two Pytorch (Paszke et al., 2019) `DataLoader` objects containing train and test data and fitting parameters such as the number of epochs `epochs` and the learning rate `l_r`. The `predict` method can be used to get class membership predictions for the test set. The `get_global_explanation` is used to extract logic formulas in disjunctive normal form. Once extracted, formulas can be tested on an unseen set of test samples using the `test_explanation` method providing the accuracy of the logic formula.

```python
import deep_logic as dl

# import train, validation and test data loaders
[...]

# instantiate a mu network
model = dl.models.PsiNet(n_classes=n_classes, n_features=n_features,
                         hidden_neurons=[200], loss=torch.nn.CrossEntropyLoss(),
                         l1_weight=0.001, fan_in=10)

# fit the model
model.fit(train_data, val_data, epochs=100, l_r=0.0001)

# get predictions on test samples
outputs, labels = model.predict(test_data)

# get first-order logic explanations for a specific target class
target_class = 1
formula = model.get_global_explanation(x_val, y_val, target_class)
# compute explanation accuracy
accuracy = dl.logic.test_explanation(formula, target_class, x_test, y_test)
```

Listing 1: Example on how to use high-level APIs.

3.2 Low-level APIs

The library supports a set of low-level APIs allowing a fine-grained customization of the neural networks as shown in the following code example 2. The architecture of the model and the training loop is defined by means of standard Pytorch APIs (Paszke et al., 2019). The `l1_regularization` combined with the `prune_features` method allow the network to get rid of less relevant input concepts to solve the classification problem. This allows the generation of concise explanations for the neural network predictions (Ciravegna et al., 2020b). The `combine_local_explanations` method allows the extraction of logic formulas from the trained model. Once extracted, formulas can be tested on an unseen set of test samples using the `test_explanation` method providing the accuracy of the logic formula.

```python
import deep_logic as dl

# import train, validation and test data loaders
[...]

# architecture
layers = [
    torch.nn.Linear(n_features, 100),
    torch.nn.ReLU(),
    torch.nn.Linear(100, 10),
    torch.nn.ReLU(),
    torch.nn.Linear(10, n_classes),
]
model = torch.nn.Sequential(*layers).to(device)

prune_epoch = epochs // 2
optimizer = torch.optim.Adam(model.parameters(), lr=l_r)
loss_form = torch.nn.CrossEntropyLoss()
model.train()
for epoch in range(epochs):
    # forward pass
    optimizer.zero_grad()
    y_pred = model(x_train)
    # Compute Loss with L1-regularization
    loss = loss_form(y_pred, y_train) + l1 * dl.utils.l1_regularization(model)

    # backward pass
    loss.backward()
    optimizer.step()

    # prune weights associated to concepts with low relevance
    if prune_epoch == epoch:
        dl.utils.prune_features(model, n_classes=1, device=device)

# get predictions on test samples
y_test_pred = model(x_test)

# get first-order logic explanations for a specific target class
target_class=1
formula, _, _ = dl.logic.relu_nn.combine_local_explanations(model, x_val, y_val,
                                                            target_class)
# compute explanation accuracy
accuracy = dl.logic.test_explanation(formula, target_class, x_test, y_test)
```

Listing 2: Example on how to use low-level APIs.

4 Software and documentation availability

In order to make state-of-the-art approaches accessible to the whole community, we released this library as a Python package on PyPI: https://pypi.org/project/deep-logic/.An extensive documentation on methods is available on read the docs[2] and unit tests results on TravisCI[3]. The Python code and the scripts used for the experiments, including parameter values and documentation, is freely available under Apache 2.0 Public License from a GitHub repository[4].

5 Conclusion

5.1 Limitations and future research directions

The extraction of a first-order logic explanation requires symbolic input and output spaces. This constraint is the main limitation of our framework, as narrows the range of applications down to symbolic I/O problems. In some contexts, such as computer vision, the use of LENs may require additional annotations and attribute labels to get a consistent symbolic layer of concepts. However, recent work on automatic concept extraction may partially solve this issue leading to more cost-effective concept annotations (Ghorbani et al., 2019; Kazhdan et al., 2020).

The improvement of LENs models is an open research area. The efficiency and the classification performances of fully interpretable LENs, i.e. ψ network, is still quite limited due to the extreme pruning strategy.

5.2 Broader impact

Current legislation in US and Europe binds AI to provide explanations especially when the economical, ethical, or financial impact is significant (EUGDPR, 2017; Law, 10). This work contributes to a lawful and safer adoption of some of the most powerful AI technologies allowing deep neural networks to have a greater impact on society. Extracting first-order logic explanations from deep neural networks enables satisficing (Simon, 1956) knowledge distillation while achieving performances comparable with the state of the art. The formal language of logic provides clear and synthetic explanations, suitable for laypeople, managers, and in general for decision makers outside the AI research field.

Thanks to their explainable nature, LENs can be effectively used to understand the behavior of an existing algorithm, to reverse engineer products, to find vulnerabilities, or to improve system design. From a scientific perspective, formal knowledge distillation from state-of-the-art networks may enable scientific discoveries or confirmation of existing theories.

[2]https://deep-logic.readthedocs.io/en/latest/
[3]https://travis-ci.org/github/pietrobarbiero/logic_explainer_networks
[4]https://github.com/pietrobarbiero/logic_explainer_networks

Acknowledgments and Disclosure of Funding

We thank Stefano Melacci, Pietro Lió, and Marco Gori for useful feedback and suggestions.

This work was partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 848077.

References

Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., Khlaaf, H., Yang, J., Toner, H., Fong, R., et al. (2020). Toward trustworthy ai development: mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*.

Carvalho, D. V., Pereira, E. M., and Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832.

Chander, A., Srinivasan, R., Chelian, S., Wang, J., and Uchino, K. (2018). Working with beliefs: Ai transparency in the enterprise. In *IUI Workshops*.

Ciravegna, G., Giannini, F., Gori, M., Maggini, M., and Melacci, S. (2020a). Human-driven fol explanations of deep learning. In *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence* {*IJCAI-PRICAI-20*}, pages 2234–2240. International Joint Conferences on Artificial Intelligence Organization.

Ciravegna, G., Giannini, F., Melacci, S., Maggini, M., and Gori, M. (2020b). A constraint-based approach to learning and explanation. In *AAAI*, pages 3658–3665.

Das, A. and Rad, P. (2020). Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*.

EUGDPR (2017). Gdpr. general data protection regulation.

Ghorbani, A., Wexler, J., Zou, J., and Kim, B. (2019). Towards automatic concept-based explanations. *arXiv preprint arXiv:1902.03129*.

Goddard, M. (2017). The eu general data protection regulation (gdpr): European regulation that has a global impact. *International Journal of Market Research*, 59(6):703–705.

Gunning, D. (2017). Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2(2).

Kazhdan, D., Dimanov, B., Jamnik, M., Liò, P., and Weller, A. (2020). Now you see me (cme): Concept-based model extraction. *arXiv preprint arXiv:2010.13233*.

Kim, B., Gilmer, J., Wattenberg, M., and Viégas, F. (2018). Tcav: Relative concept importance testing with linear concept activation vectors.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. (2020). Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR.

Law, P. A. (10). Code of federal regulations. *Wash.: Gov. print. off.*

McCluskey, E. J. (1956). Minimization of boolean functions. *The Bell System Technical Journal*, 35(6):1417–1444.

McColl, H. (1878). The calculus of equivalent statements (third paper). *Proceedings of the London Mathematical Society*, 1(1):16–28.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Quine, W. V. (1952). The problem of simplifying truth functions. *The American mathematical monthly*, 59(8):521–531.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological review*, 63(2):129.