Neuro-symbolic Computation for XAI: Towards a Unified Model

Giuseppe Pisano[a], Giovanni Ciatto[b], Roberta Calegari[a] and Andrea Omicini[b]

[a]*Alma AI – Alma Mater Research Institute for Human-Centered Artificial Intelligence, Alma Mater Studiorum—Università di Bologna, Italy*
[b]*Dipartimento di Informatica – Scienza e Ingegneria (DISI), Alma Mater Studiorum—Università di Bologna, Italy*

Abstract
The idea of integrating symbolic and sub-symbolic approaches to make intelligent systems (IS) understandable and explainable is at the core of new fields such as *neuro-symbolic computing* (NSC). This work lays under the umbrella of NSC, and aims at a twofold objective. First, we present a set of *guidelines* aimed at building explainable IS, which leverage on logic *induction* and *constraints* to integrate symbolic and sub-symbolic approaches. Then, we reify the proposed guidelines into a case study to show their effectiveness and potential, presenting a prototype built on the top of some NSC technologies.

Keywords
XAI, Hybrid Systems, Neural Networks, Logical Constraining

1. Introduction

In the last decade, we have witnessed an unprecedented spread of artificial intelligence (AI) and its related technologies [1]. The fields involved are manifold, ranging from autonomous driving systems and expert systems to computer vision and reasoning systems—just to mention a few. In all the aforementioned fields, AI is enabling artificial systems to act in a more intelligent, efficient, and effective way.

Besides the many impactful achievements, some factors have emerged that can slow down the further diffusion of AI technologies. A primary concern is related to the *trustability* of intelligent systems (IS) leveraging on sub-symbolic AI—i.e., exploiting approaches such as deep learning. Indeed, the resulting IS suffers from well-known problems of *opaqueness*, since humans typically find very difficult to understand how sub-symbolic systems work.

The need to overcome opaqueness is one of the main goal of the research in *eXplainable AI* (XAI) [2], essentially aimed at making AI systems more understandable and explainable. Some of the most interesting XAI techniques are deeply rooted in symbolic AI—as representatives of

21st Workshop "From Objects to Agents" (WOA), September 14–16, 2020, Bologna, Italy
g.pisano@unibo.it (G. Pisano); giovanni.ciatto@unibo.it (G. Ciatto); roberta.calegari@unibo.it (R. Calegari); andrea.omicini@unibo.it (A. Omicini)
https://about.me/gciatto (G. Ciatto); http://robertacalegari.apice.unibo.it (R. Calegari); http://andreaomicini.apice.unibo.it (A. Omicini)
0000-0003-0230-8212 (G. Pisano); 0000-0002-1841-8996 (G. Ciatto); 0000-0003-3794-2942 (R. Calegari); 0000-0002-6655-3869 (A. Omicini)

a natural choice for building human-intelligible IS [3]. In fact, symbolic systems offer a human-understandable representation of their internal knowledge and processes: so, integrating them into sub-symbolic models – to promote transparency of the resulting system – is the most prominent stimulus for new research fields such as *neuro-symbolic computing* (NSC) [4].

Our work lays under the umbrella of NSC, and its contribution is twofold. On the one hand, we present a set of *guidelines* aimed at building explainable IS, even when they exploit sub-symbolic techniques. The guidelines leverage on *logic induction* and *logic constraints* as the two main techniques integrating symbolic and sub-symbolic approaches. In particular, logic induction makes it possible to extract the knowledge from black-box ML-based predictors – typically, the sub-symbolic part of an IS – offering a corresponding symbolic, logical representation. Conversely, logic constraints are exploited to inject some logic knowledge into the black box, thus restricting the numerical underlying model.

On the other hand, we reify the proposed guidelines into a case study to show their effectiveness and potential. In particular, we present a prototype built over some promising NSC technologies. The resulting system is then assessed to verify its capability of being adjusted (i.e., debugged and fixed) in case some unexpected behaviour in the sub-symbolic part of the system is revealed. Accordingly, we show how the prototype is correctly performing w.r.t. the proposed guidelines.

The paper is structured as follows. Section 2 provides a brief overview of the field of symbolic and sub-symbolic integration. It also includes the main related works on the use of a hybrid system as a mean for explainability. In Section 3, we first present the guidelines for building explainable systems; then, in Section 4 we discuss a possible instantiation of the proposed guidelines. In Section 5, we proceed with the assessment of our prototype and the discussion of results. Finally, Section 6 concludes the work.

2. Background

In the last years, deep and machine (ML) learning methods have become largely popular and successful in real-world intelligent systems. However, their use raises the issue of understanding and explaining their behaviour to humans. Neural networks in particular – which are the most hyped and widely adopted approach within sub-symbolic AI – mostly suffer from the problem of opaqueness: the way they obtain their results and acquire experience from data is unintelligible to humans.

One of the proposed approaches addressing the explainability problem [2] is the hybridisation of symbolic and sub-symbolic techniques [3]. An increasing number of authors recognises that formal logic is capable of significantly improving humans' understanding of data [5]. In their view, in principle, an opaque system, combined with a symbolic model, can provide a significant result in terms of transparency. Many researches start from these assumptions.

Among the others, the neuro-symbolic computing (NSC) field is a very recent and promising research area whose ultimate goal is to make symbolic and sub-symbolic AI techniques effortlessly work together. Due to the freshness of the topic, however, a well-established and coherent theory of NSC is still missing. For this reason, a variety of methods have been proposed so far, focusing on a multitude of aspects not always addressing interpretability and explainability as their major concern. Nevertheless, some attempts to categorise XAI-related existing works under the NSC umbrella exist [4, 3], which provide for a helpful overview of the topic.

Despite NSC does not explicitly include XAI among its primary goals, in this paper we borrow some ideas from the NSC field to show how the explainability of modern IS may benefit from the integration of symbolic and sub-symbolic AI. In particular, our work focuses on two main NSC sub-fields: namely, the *logic as a constraint*, for the constraining module, and *differentiable programming* for the induction module—since the proposed prototype exploits logic induction via differentiable programming [6]. In short, the former line of research aims at constraining the training process of a sub-symbolic predictor in such a way that it cannot violate the superimposed constraint at runtime. About the latter, differentiable programming is the combination of neural networks approaches with algorithmic modules in an end-to-end differentiable model, often exploiting optimisation algorithms like gradient descent [7].

Within the scope of *logic as a constraint*, most approaches exploit some sort of *logic* formulæ to constrain the behaviour of the sub-symbolic predictor—in most cases, a neural network. This formula is then *vectorised* – i.e. translated into a continuous function over vectors of real numbers – and exploited as a regularisation term in the loss function used for training the sub-symbolic predictor [8, 9, 10]. Different strategies have been proposed to this end. For example, in [11] the symbolic constraints are used to modify the network structure incorporating them into the training process. In the general case, however, logic constraining can be used to fix bias or bugs in the behaviour of a sub-symbolic system, or, it can mitigate the situation in which poor training data is available to correctly train a black-box system on a specific aspect.

With respect to the second research area, some works laying under the umbrella of differentiable programming fruitfully intertwine ML and inductive logic programming (ILP) [12] to provide logic induction capabilities on top of sub-symbolic predictors. ILP is a well established research area, laying at the intersection of ML and logic programming, which is strongly interrelated with NSC. An ILP system is a tool able to induce (i.e., derive) – given an encoding of some background knowledge and a set of positive and negative examples represented logic facts –, a logic program that entails all the positive and none of the negative examples. While traditionally these systems base their operation on the use of complex algorithms as their core component [13] – deeply undermining their efficiency and usability – hybrid approaches exist leveraging NSC to make the induction process more efficient [6]. Furthermore, as we show in this paper, induced logic rules can be used as a means to *inspect* what a black-box predictor has learned—as induction makes the predictor knowledge explicit in symbolic form.

2.1. Related Works

As far as the intersection of logical systems and numerical models is concerned, the main contributions come from [14] and [15]. Their work can be summarised in:

- usage of a knowledge base filled automatically from training data to reason about what has been learned and to provide explanations;
- adoption of logic rules to constrain the network and to correct its biases.

Although these works offer a good starting point in the search of a solution to the transparency problem, some remarks should be pointed out. First, exploiting a knowledge base obtained only from the training data is not sufficient to acquire the knowledge required to explain the entire network behaviour. That would lead to a system giving explanations according to the network optimal functioning, not accounting for the training errors. Moreover, according to these models, also the constraining part should be driven by the rules inferred from the training data, hardly limiting the potential of those techniques. Indeed, the possibility for users to impose their own rules would also give them the ability to mitigate the errors derived from an incomplete or incorrect training set.

The work presented here aims at building a model overcoming both these limitations. As for the explanations' coherence problem, the use of the black box as a data source in the logic induction process should guarantee the correct correlation between the black box itself and the derived logic theory. Furthermore, logic can be leveraged so as to combine the IS with the user experience and knowledge, thus exploiting all the advantages of the constraining techniques.

3. A NSC model for XAI

In this paper, we present a general model for explainable data-driven IS, and a set of guidelines supporting their construction. The novelty of our approach lays in the fruitful intertwining of symbolic and sub-symbolic AI, which aims at providing both predictive accuracy – through the exploitation of state-of-the-art machine learning (ML) techniques – and transparency—through the exploitation of computational logic and logic programming (LP). The proposed model, in particular, aims at overcoming the well-known limitations of ML-based AI w.r.t. interpretability. Accordingly, it leverages on a number of contributions from the NSC and LP research field, as well as two basic techniques—namely, induction and constraining.

The main idea behind our work is that IS should feature both predictive precision and interpretability features. To preserve predictive precision, IS should keep exploiting high-performance, data-driven, black-box predictors such as (deep) neural networks. To overcome the interpretability-related issues, IS should couple sub-symbolic approaches with logic theories obtained by automatically extract the sub-symbolic knowledge of black boxes into symbolic form. This would in turn enable a number of XAI-related features for IS, providing human users with the capabilities of *(i)* inspecting a black box – also for debugging purposes –, and *(ii)* correcting the system behaviour by providing novel symbolic specifications.

Accordingly, in the reminder of this section, we provide further details about the *desiderata* which led to the definition of our model. We then provide an abstract description of our model and a set of guidelines for software engineers and developers. Finally, we provide a technological architecture to assess both the model and the guidelines.

3.1. Desiderata

Regardless of the architectural and technological choices performed by designers and developers, the IS adhering to our model are characterised by several common *desiderata* ($\mathbf{D_i}$) w.r.t. their overall functioning and behaviour. Generally speaking, these are aimed at making IS both prediction-effective and explainable. Indeed, following this purpose, IS should

$\mathbf{D_1}$ attain high predictive performances by leveraging ML and data-driven AI

$\mathbf{D_2}$ provide human-intelligible outcomes / suggestions / recommendations

$\mathbf{D_3}$ acquire knowledge from both data and from high-level specifications

$\mathbf{D_4}$ make their knowledge base inspectable by human experts

$\mathbf{D_5}$ let human experts override / modify their knowledge base

A key enabling point in satisfying these *desiderata* is knowledge representation. While ML and data-driven AI are certainly required to mine effective information from data efficiently, they soon fall short when it comes to satisfying *desiderata* $\mathbf{D_2}$–$\mathbf{D_5}$. This happens because they mostly leverage on a *distributed*, sub-symbolic representation of knowledge which is hard to interpret for human beings. Therefore, to support $\mathbf{D_2}$ and $\mathbf{D_4}$, we need an alternative human-intelligible representation of the sub-symbolic model and a procedure to perform such a representation transformation. Furthermore, to support $\mathbf{D_3}$ and $\mathbf{D_5}$, we also need to link symbolic and sub-symbolic representations in a bidirectional way—meaning that an inverse procedure aimed at converting symbolic information back into sub-symbolic form is needed as well.

Accordingly, the focus of our model is both on the extraction of symbolic representation from the black-box predictor and, vice-versa, on the injection of symbolic representation (constraints) in the corresponding black-box predictor. The purpose of this dichotomy is twofold: guaranteeing the comprehensibility of the black-box model for humans – as symbolic representations are to some extent inherently intelligible –, and enabling debugging and correction of the black-box behaviour, in case some issue is found through inspection.

3.2. Modelling

Generally speaking, we model an explainable, hybrid IS as composed by a black box, a knowledge base (KB) and an automatic logic reasoner, as depicted in Figure 1. Explainable recommendations or suggestions are provided to the end-user via the logic reasoner – based on symbolic rules and facts included in the KB by domain experts – and via black-box predictions—based on data. Accordingly, the reasoner combines several sorts of inference mechanisms (e.g. deduction and induction). Furthermore, to balance the knowledge coming from data with the domain experts knowledge, the model exploits induction and constraining techniques to improve the black box with the logical knowledge and vice-versa.

The black-box module is the core of the IS, making it capable of mining effective information from data. Any sub-symbolic model providing high predictive performances – e.g. neural networks, SVM, generalised linear models, etc. – can be used for the module implementation. This, however, may bring opaqueness-related issues. Thus, the black-box module is complemented with the other two modules to provide explanation and debugging facilities.

The reasoner module is aimed at providing explainable outcomes to the end-users. In particular, explanations are given in terms of logic KB, capable of approximating the black box. The construction of the logic KB relies on the *induction* capabilities offered by this module. More precisely, the outcomes generated by the black box are exploited to build a logic theory, mimicking the work of the black-box predictor with the highest possible fidelity. An inductive

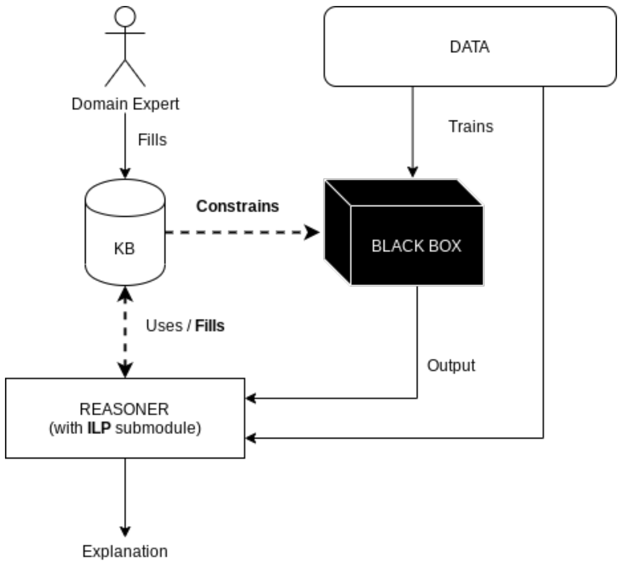


Figure 1: Model schema.

process is then fed with the resulting extended theory. This leads to a theory containing a number of general relations, which can be exploited to provide intelligible information to the end-users. The described workflow supports explainability in two ways: *(i)* it provides a global explanation of how the black box works in terms of general relations that must always hold; *(ii)* it provides local explanations, justifying each black-box conclusion by enabling the use of deductive reasoning on the connected logical knowledge. In other words, this mechanism is what enables IS to be *interpretable* and *debuggable* by users. Finally, the KB module aims at storing the logical knowledge approximating the black-box behaviour. The knowledge can be modified by domain experts. When this is the case, it becomes of paramount importance to keep the black-box module coherent with the human-proposed edits. To this end, constraining is performed to align the black-box behaviour with the KB. This mechanism is what enables IS to be *fixed* by users.

In the reminder of this section, we delve deeper into the details of these mechanisms.

3.2.1. Logic induction

While deductive reasoning moves from universal principles that are certainly true to specific conclusions that can be mechanically derived, inductive reasoning moves from specific instances to general conclusions. Induction is a key mechanism of our model. Assuming IS can transform the data they leverage upon for training black boxes into theories of logic facts, inductive reasoning can be used to extract the rules that best explain that data.

The induction procedure is not meant to replace the black box as a learning or data-mining tool—as it would be quite difficult to obtain the same performance in terms of accuracy and ability to scale over huge amounts of data. Conversely, it aims at “opening the black box”, letting humans understand how it is performing and why. In other words, following the abstract framework presented in [16, 17], logic induction is a means for *explaining* the functioning of a black-box predictor via symbolic rules. More precisely, induction is fundamental for the debuggability of our model. For example, it enables the discovery of fallacies in the learning process of a black box even for people not used to the specific domain.

In our model, the induction process is fed with both the raw data and the outcomes of the black box. In the former case, induction leads to the unveiling of latent relations possibly buried in the original data: we refer to the logic theory obtained as the *reference theory*. In the latter case, induction leads to a symbolic approximation of the knowledge the black box has acquired from data via ML: we refer to the logic theory obtained as *explanation theory*. Assuming the soundness of the induction process, discrepancies between these two theories could reveal some possible rules that have not been correctly learned by the sub-symbolic model. It is then possible to fix it by enforcing the correct relations via logic constraining.

Finally, one last point is worth to be discussed. Unless the induction process possesses the same learning capabilities of the black box, it is impossible to detect all its learning errors. In this case, in fact, the reference theory would be the optimal solution itself, and the sub-symbolic model would be useless. As the induction process aims at opening the box, its use on the raw data aims at providing insights on the accuracy of the training phase. Thus, it does not aim at providing an optimal solution.

3.2.2. Logic constraining

While induction is the mechanism aimed at translating sub-symbolic knowledge into symbolic form, constraining is the inverse process aimed at injecting some symbolic knowledge into a black box. In this way, both the induced rules and those coming from domain experts are used to constrain the black box and its outcomes. This is another key mechanism of our model.

In particular, the ability to encode some prior knowledge within a model is interesting for two main reasons. On the one side, it enables a reduction in the data needed to train the black box. In fact, handmade rules may be exploited to model a portion of the domain not included in the training data. So, rather than creating a more exhaustive training set, an expert may directly encode his/her knowledge into rule and train a constrained black box. We call this procedure *domain augmentation*. On the other side, one may exploit the constraining process to guide and support a black-box learning, e.g., helping it to avoid biases. We call this procedure *bias correction*.

3.3. Guidelines

In the following, we discuss the main aspects to be considered when designing a system conforming to our model. As a first step in this direction, we first handle some important aspects concerning data representation.

3.3.1. Type of logic

The first aspect concerns the type of logic used by the IS. Logic, in fact, plays a fundamental role: it heavily affects the explainability properties of the final system. For this reason, the choice of the most appropriate logic formalism is a primary concern.

For our model to be effective, the selected logic formalism should provide high flexibility in the description of the domain. At the same time, to keep a system as simple as possible, it should be coherent in every part of it: from the constraining module to the induction one, every part should share the same representation of the logical knowledge.

3.3.2. Sort of data

The second aspect concerns the sort of data used to feed the IS. For the logical induction process to be carried out on data, the first thing to do is transforming the data itself. In other words, the available data should be translated into a logical theory according to the chosen formalism. For the translation to be effective, the resulting theory should preserve as much as possible the information contained in the original data. Anyway, when dealing with some particular type of data some problems have to be taken into account.

When unstructured data – e.g., images, audio, time-series, etc – come into play, the transformation process may become considerably harder. In that case, a two-step transformation must be performed, involving:

1. extraction of semantically meaningful, structured features from unstructured data;
2. translation of extracted features into the desired logical formalism.

Of course, this procedure adds a whole range of new problems related to the accuracy of the features extracted. In fact, the explanation process is mainly linked to the reliability of the data used as the basis for the induction process. The exploitation of data generated through an automatic process makes a discrepancy between the data extrapolated by the inductive process and the behaviour of the black box more likely.

3.3.3. Sorts of black box

In the general case, we require the architectures adhering our model to be as agnostic as possible w.r.t. the particular sort of black box to be adopted. In fact, the choice of the most adequate sort of black box is strongly affected by *(i)* the nature of the data at hand, *(ii)* the availability of some symbolic extraction technique making the black box less opaque, *(iii)* the possibility of constraining the black box with logic formulæ.

Strictly speaking, the choice of the black box should be delayed to the implementation phase. Here we describe a number of criteria to be taken into account when performing this choice. We recall, however, that the other guidelines described so far are agnostic w.r.t. the sub-symbolic model to be used.

As far as the nature of the data is concerned, traditional ML techniques [18] – e.g., decision trees, generalised linear models, etc. – are usually exploited on structured datasets, whereas deep learning techniques are better suited to deal with unstructured data. However, this is not due to an inadequacy of neural networks for structured data, but rather to the greater simplicity of the learning algorithms used by traditional ML techniques. Structured data delivers a relatively-smaller complexity to deal with, making simpler ML algorithms a more suitable choice for their comprehensibility and usability.

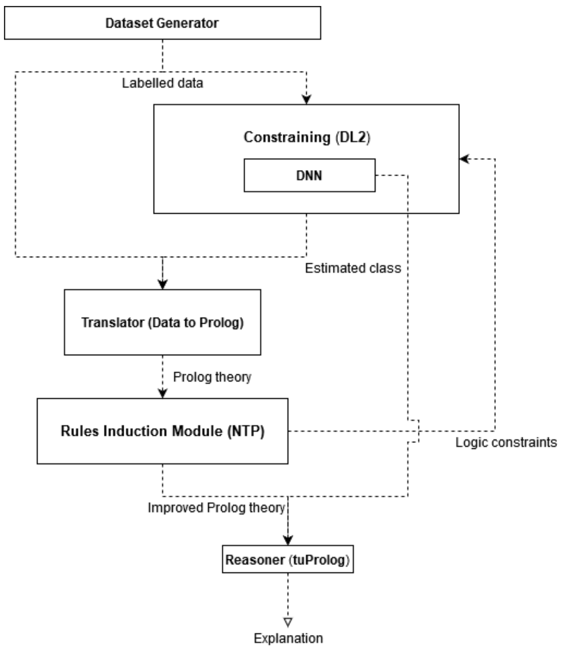


Figure 2: Technologies schema.

As far as opaqueness is concerned, virtually any ML techniques is affected by that to some extent: yet, neural networks remain the most critical from this point of view. Nevertheless, the vast majority of rule induction procedures are either black-box agnostic – a.k.a. pedagogical [19] – or neural-network-specific [3]. In terms of the support for constraining, it should be possible to guide the learning activity trough some regulariser terms attained from the constraints to be enforced.

4. Technological architecture

This subsection provides a technological description of an IS prototype adhering to our model. It is based on the concrete architecture detailed in Figure 2, which specialises the abstract model from Figure 1 through a number of technological commitments.

First, we present our choices in relation to the points examined in Subsection 3.3. As for the type of logic, within the scope of this paper, we adopt first-order logic (FOL) as our formalism of choice, and Prolog as our reference concrete language. FOL is likely to offer the best trade-off between flexibility and expressiveness of the representation language. Furthermore, the choice of FOL enables the exploitation of many existing approaches supporting both conditioning and induction. Finally, the choice of the Prolog syntax enables the direct exploitation of the induced theories within existing automatic reasoners—e.g. tuProlog, SWI-Prolog, etc.

As far as the sort of the data is concerned, in this paper we only describe a prototype based on structured data. In fact, the feature extraction module can be omitted without hindering the generality of our approach. Moreover, the greater simplicity in the data helps to avoid the problems related to the possible information loss due to their transformation. In the future, however, we plan to extend our prototype to support arbitrary datasets including any sort of data. This requires the creation of a module for feature extraction.

In terms of the sort of the black-box used, we focus on a prototype based on neural networks, being them the most critical from the opacity perspective. The remaining technological choices derive consequently.

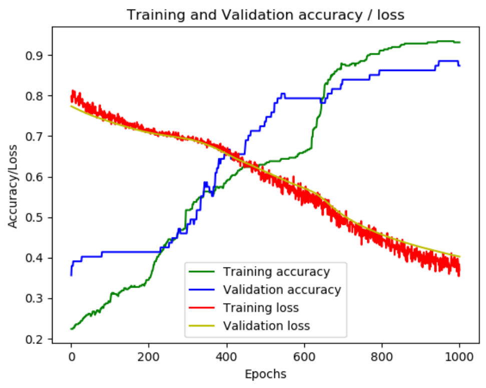
The prototype exploits two main technologies: DL2 [9] for the conditioning part, and NTP [6][20] for the induction part. DL2 is one of those models leveraging on symbolic rules vectorisation as means to constrain the target neural network. We choose DL2 as the most mature and user-friendly technology supporting neural-network constraining through logic rules.

The choice of NTP as the induction engine is more straightforward. Indeed, NTP is among the few ready-to-use technologies offering both deductive and inductive logic capabilities in a differentiable way. On the one hand, *differentiability* is what makes induction more computationally efficient w.r.t. similar technologies—and this is why we choose it. On the other hand, NTP deductive capabilities are not mature enough. In fact, training a model correctly approximating a logic theory in a reasonable amount of time is very challenging—especially when the complexity of the theory grows. While this could be acceptable for the induction process, it is still very limiting for deductive reasoning. Moreover, traditional logic engines combine a very large ecosystem of supporting tools – IDEs, debuggers, libraries – that have potential to hugely improve the effectiveness of the reasoning phase. For this reason, we adopt tuProlog (2P) [21] – a Java-based logic engine built for use in ubiquitous contexts – as the main tool for the manipulation of logic theories, as well for automated reasoning. This choice is motivated by its reliability, modularity, and flexibility.

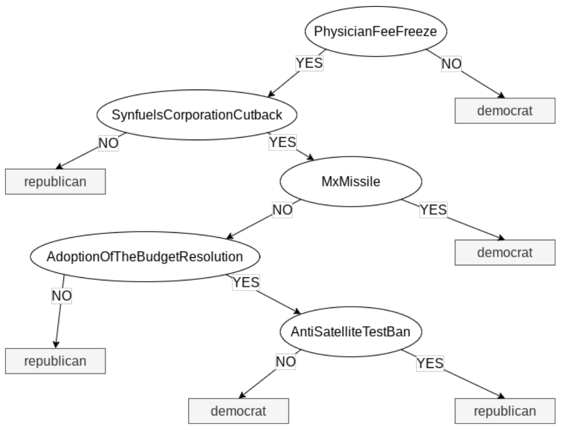
While the entire system is built around the aforementioned technologies, Python is used as the glue language to keep modules together. In particular, the sub-symbolic module is implemented via the PyTorch learning framework [22], thus ensuring an easy integration with DL2—which is natively built to work with this framework. The modules responsible for the extraction of the Prolog theory from the input dataset (i.e. the Translator block in Figure 2) are created using Python as well. The NTP integration takes place through the Prolog logic language. In fact, NTP easily allows the use of Prolog theories as input for the induction process. The result of the induction phase is also delivered in a logical form, allowing for an easy consultation and analysis through the 2P technology.

5. Assessment

In this section we assess our model from the explainability perspective leveraging on the prototype implementation described in Section 4. Particularly, we train the sub-symbolic module through a real-world ML problem, and we test whether and to what extent our architecture actually provides *(i)* the capability of building a logical representation of sub-symbolic knowledge acquired from data via induction, *(ii)* the capability of altering or fixing the system behaviour via conditioning, and, ultimately, *(iii)* the inspectability and debuggability of the system as a whole. In the following subsections we set up an *ad-hoc* experiment aimed at performing these



(a) Network default training



(b) Decision tree built on test data

Figure 3: Network default training (a) and decision tree built on test data (b).

tests. A detailed presentation of the experiments and their results follows.

5.1. Experiment Design

The proposed experiment aims at comprehensively testing the operation of the prototype against some real classification problem. In particular, we consider a binary classification task on structured data. Roughly speaking, the experiment works by artificially constructing a buggy neural network for the classification task, and then showing how our architecture makes it possible to reveal the bug. More precisely, the experiment is structured as follows:

1. a neural network is trained until reaching maximum validation-set accuracy for the classification task;
2. by combining the training data and the predictions of the network, a coherent Prolog theory is extracted;
3. the induction module is used to extract the latent relations from the theory, until at least one relation which properly divides the data is found;
4. through constraining, we inject an error in the network, in such a way that it misclassify some instances;
5. by repeating Item 2 and Item 3, we show that the approximated logic theory reveals the injected error.

This workflow lets us verify the behaviour of our prototype and of all its components. In particular, Item 3 and Item 5 aim at demonstrating that a logic theory that optimally approximates a neural network is actually attainable. In terms of the ability of debugging and correcting a network, the whole procedure aims at demonstrating their feasibility. The extraction of the correct theory in the initial part of the experiment is comparable to the one extracted from a malfunctioning classifier. The inclusion of the fake constraint, as well as its recognition in the theory extracted at the conclusion of the experiment, shows the feasibility of the network correction process.

Two fundamental points are worth to be taken into account for the experiment to be meaningful. The first point is the ability to accurately evaluate the accuracy of the logic theory recovered. In fact, in order to verify that the theory extracted from the neural network is actually the correct one, it is either necessary *(i)* to have an optimal knowledge of the domain, or *(ii)* to use easily analysable and verifiable datasets. As for the first case – being an expert of the analysed domain – it turns out to be very easy, by verifying the correctness of the logical relations. The only alternative comes from the usage of an easily verifiable dataset as the base for the experiment. Indeed, it should be possible also for a domain novice to understand the ratio behind the data. For instance, a simple way to verify the correctness of the rules is to use an alternative ML tool – one that can guarantee a high level of transparency – to analyse the data. In the case of a classifier, the best choice could be a decision tree (DT). In fact, DTs training produces a symbolic model that can be efficiently verified by the user. Hence, DT output can be used as a reference in the evaluation of the induction results.

The second point is the exploitation of constraining as a means to inject bugs into a network in a controlled way. In fact, in order to verify the prototype capability of revealing and correcting bugs in the black-box module, it is first necessary to construct a buggy network. However, in case of a poorly training, as in that case, it can not be clear where the bug is. Hence, to evaluate the actual capabilities of the prototype, the bugs to be spotted must be *a priori* known in order to have a reference.

5.2. Experiment Setup

We base our experiment on the *Congressional Voting Records* data set[1] from the UCI Repository [23]. It consists of table registering the voting intentions – namely, favourable, contrary or unknown – of 435 members of the Congress of the USA on 15 main topics, as well as their political orientation—namely, either Democrat or Republican. The goal of the classification task is to sort a member of the Congress as either democratic or republican depending on his/her intention on those 15 topics.

Given the relatively small amount of instances in the dataset, along with the small number of attributes, an intuitive understanding of the classification problem can be assumed. However, to further simplify the analysis of the results, we use a DT trained over the data as a reference.

A neural network capable of distinguishing between Democrats and Republicans based on voting intentions is trained and assessed as described above. The experiments code is available at the Github repository[2]. We now proceed discussing the results we obtained in each step of the experiment.

[1]https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records
[2]https://github.com/Gilbocc/NSC4ExplainableAI

5.3. Results

The training of the aforementioned neural network is performed via ordinary stochastic gradient descent until reaching a 95% accuracy score on the validation set (containing a randomly selected 20% of whole data). The training process is depicted in Figure 3a.
Through the induction process on the data generated by the network it was possible to recover the following relationships:

$$\begin{aligned}
&democrat(X_1, \dots, X_N) \ \texttt{:-}\\
&\qquad opposePhysicianFeeFreeze(X_1, \dots, X_N),\\
&\qquad supportMxMissile(X_1, \dots, X_N),\\
&\qquad supportBudgetResolution(X_1, \dots, X_N).\\
&democrat(X_1, \dots, X_N) \ \texttt{:-}\\
&\qquad opposePhysicianFeeFreeze(X_1, \dots, X_N),\\
&\qquad supportBudgetResolution(X_1, \dots, X_N).\\
&republican(X_1, \dots, X_N) \ \texttt{:-}\\
&\qquad supportPhysicianFeeFreeze(X_1, \dots, X_N),\\
&\qquad opposeAntiSatelliteTestBan(X_1, \dots, X_N).\\
&republican(X_1, \dots, X_N) \ \texttt{:-}\\
&\qquad supportPhysicianFeeFreeze(X_1, \dots, X_N),\\
&\qquad opposeAntiSatelliteTestBan(X_1, \dots, X_N),\\
&\qquad opposeBudgetResolution(X_1, \dots, X_N),\\
&\qquad opposeSynfuelsCutback(X_1, \dots, X_N).
\end{aligned}$$

To verify their validity we can examine the DT generated using the original data as source (Figure 3b)—the C4.5 algorithm has been used. As expected, the inductive process managed to recover the relationship that most discriminates between Democrats and Republicans: the tendency of the Democrats to be against the freezing of the cost of medical expenses.

The verification of the correction process is based on the reversal of the relationship retrieved above. Formally, the conditioning of the network occurred on this rule:

$$\begin{aligned}
&republican(X_1, \dots, X_N) \ \texttt{:-}\\
&\qquad opposePhysicianFeesFreeze(X_1, \dots, X_N).
\end{aligned}$$

In natural language, this Prolog rule expresses that, given a set of votes ($X_1, \dots, X_N$), if these share the opposition to the freezing of medical expenses, then the voter is likely from the Republican Party. This constraint translates into a very simple result: all the votes should belong to Republicans. Indeed, if initially only the Democrats were against the freezing, by forcing the network to consider them as Republicans, we reach a situation where the whole set of voters is Republican.

More in detail, constraining rules are codified through the DSL offered by the D2L implementation. For example, the above rule can be expressed in the form:

```
dl2.Implication(
    dl2.EQ(x[FeesFreeze], 0),
    dl2.LT(y[Dem], y[Rep]))
```

Table 1
Training with fake constraint results.

	precision	recall	f1-score	support
repubblican	0.42	1.00	0.59	35
democrat	1.00	0.06	0.11	32
accuracy			0.44	87

This rule is then automatically converted in a numerical function. Its evaluation contributes to the final loss function adopted by the target neural network.

For the experiment, a completely-new network is trained considering the new constraint. The results in Table 1 show the effect of the constraint on the network. The Democrats/Republican imbalance – which is made evident by Table 1 – reflects in the result of the induction process on the data generated by the constrained network:

$$\begin{aligned}
&republican(X_1, \ldots, X_N) \;\texttt{:-}\\
&\qquad supportMxMissile(X_1, \ldots, X_N),\\
&\qquad opposePhysicianFeeFreeze(X_1, \ldots, X_N).\\
&republican(X_1, \ldots, X_N) \;\texttt{:-}\\
&\qquad supportPhysicianFeeFreeze(X_1, \ldots, X_N),\\
&\qquad opposeBudgetResolution(X_1, \ldots, X_N),\\
&\qquad opposeMxMissile(X_1, \ldots, X_N).\\
&republican(X_1, \ldots, X_N) \;\texttt{:-}\\
&\qquad opposePhysicianFeeFreeze(X_1, \ldots, X_N),\\
&\qquad supportBudgetResolution(X_1, \ldots, X_N),\\
&\qquad supportAntiSatelliteTestBan(X_1, \ldots, X_N),\\
&\qquad supportMxMissile(X_1, \ldots, X_N).
\end{aligned}$$

The inducted knowledge base only contains rules concerning the Republican wing, thus confirming the footprint of the relation imposed during the conditioning process.

Summarising, the results of the assessment are positive. It was first possible to obtain the relationships implied in the operation of the classifier in a logic form. By these rules has been possible *(i)* to identify the more discriminant features in the dataset; *(ii)* to enable the correction of the black box using the user knowledge. The constraining part has also demonstrated effective. Indeed, the imposition of the user-crafted rule has led to a coherent change in the black-box behaviour—enabling its correction. The results demonstrate how the presented guidelines lead to an IS giving explanations about its functioning, thus allowing the user intervention on its behaviour.

6. Conclusions

The solutions proposed so far in the literature to the opaqueness issue – one of the main problem of today AI technologies – are disparate. In this work, we showed how symbolic logic can be a crucial element in this panorama.

On the trails of the NSC models, we presented a series of guidelines aimed at correctly integrating a ML-based predictor – i.e., a black box – with a logic-based subsystem. In particular, our guidelines support the creation of IS exposing clear insights about their own functioning, thus enabling end users to intervene on the IS behaviour via a logical interface. We then tested our guidelines against a prototype IS, in order to study if and to what extent our approach is feasible and useful. Notably, the prototype assessment confirms our approach is feasible exploiting technologies already available in the research scene. Nevertheless, the prototype has been tested only on a single scenario. In order to confirm the efficacy of our approach, we need to perform a more exhaustive range of experiments. Moreover, we plan to extend the prototype assessment with more complex use-cases. For example, as anticipated in Section 4, we intend to enhance the prototype with the support for unstructured data. This extension would considerably improve the applicability of the studied approach, allowing its assessment also on the more complex area of image classification.

Through the above experimental investigation – in the case of more positive results – we aim at introducing a rigorously formalised version of the proposed model—presented in this paper in a more intuitive and preliminary shape. Consequently, we should also better investigate and verify the preliminary guidelines provided. We aim at obtaining an accurate and comprehensive guide that would allow developers efficiently integrating opaque AI systems and logic.

References

[1] J. Bughin, E. Hazan, S. Ramaswamy, M. Chui, T. Allas, P. Dahlstrom, N. Henke, M. Trench, Artificial intelligence: the next digital frontier?, Discussion paper, McKinsey Global Institute, 2017. URL: https://www.mckinsey.com/~/media/mckinsey/industries/advancedelectronics/ourinsights/howartificialintelligencecandeliverrealvaluetocompanies/mgi-artificial-intelligence-discussion-paper.ashx.

[2] A. B. Arrieta, N. D. Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion 58 (2020) 82–115. doi:10.1016/j.inffus.2019.12.012.

[3] R. Calegari, G. Ciatto, A. Omicini, On the integration of symbolic and sub-symbolic techniques for XAI: A survey, Intelligenza Artificiale 14 (2020) 7–32. doi:10.3233/IA-190036.

[4] A. S. d'Avila Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, S. N. Tran, Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning, FLAP 6 (2019) 611–632. URL: https://arxiv.org/abs/1905.06088.

[5] S. H. Muggleton, U. Schmid, C. Zeller, A. Tamaddoni-Nezhad, T. Besold, Ultra-strong

machine learning: comprehensibility of programs learned with ILP, Machine Learning 107 (2018) 1119–1140. doi:10.1007/s10994-018-5707-3.

[6] T. Rocktäschel, S. Riedel, End-to-end differentiable proving, in: I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 2017, pp. 3788–3800. URL: http://papers.nips.cc/paper/6969-end-to-end-differentiable-proving.

[7] A. Hernández, J. M. Amigó, Differentiable programming and its applications to dynamical systems, CoRR abs/1912.0 (2019). URL: http://arxiv.org/abs/1912.08168.

[8] L. Serafini, A. S. d'Avila Garcez, Logic tensor networks: Deep learning and logical reasoning from data and knowledge, in: T. R. Besold, L. C. Lamb, L. Serafini, W. Tabor (Eds.), 11th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy'16) co-located with the Joint Multi-Conference on Human-Level Artificial Intelligence (HLAI 2016), volume 1768 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016, pp. 23–34. URL: http://ceur-ws.org/Vol-1768/NESY16_paper3.pdf.

[9] M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, M. T. Vechev, DL2: training and querying neural networks with logic, in: K. Chaudhuri, R. Salakhutdinov (Eds.), 36th International Conference on Machine Learning, (ICML 2019), volume 97 of *Proceedings of Machine Learning Research*, 2019, pp. 1931–1941. URL: http://proceedings.mlr.press/v97/fischer19a.html.

[10] J. Xu, Z. Zhang, T. Friedman, Y. Liang, G. V. den Broeck, A semantic loss function for deep learning with symbolic knowledge, in: J. G. Dy, A. Krause (Eds.), 35th International Conference on Machine Learning (ICML 2018), volume 80 of *Proceedings of Machine Learning Research*, 2018, pp. 5498–5507. URL: http://proceedings.mlr.press/v80/xu18h.html.

[11] T. Li, V. Srikumar, Augmenting neural networks with first-order logic, in: 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019), Association for Computational Linguistics (ACL), 2020, pp. 292–302. doi:10.18653/v1/p19-1028.

[12] S. Muggleton, Inductive logic programming, New Generation Computing 8 (1991) 295–318. doi:10.1007/BF03037089.

[13] A. Cropper, S. Dumancic, S. H. Muggleton, Turning 30: New ideas in inductive logic programming, in: C. Bessiere (Ed.), 29th International Joint Conference on Artificial Intelligence (IJCAI 2020), IJCAI, 2020, pp. 4833–4839. doi:10.24963/ijcai.2020/673.

[14] D. Doran, S. Schulz, T. R. Besold, What does explainable AI really mean? A new conceptualization of perspectives, in: T. R. Besold, O. Kutz (Eds.), 1st International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), volume 2071 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017, pp. 15–22. URL: http://ceur-ws.org/Vol-2071/CExAIIA_2017_paper_2.pdf.

[15] A. Bennetot, J.-L. Laurent, R. Chatila, N. Díaz-Rodríguez, Highlighting bias with explainable neural-symbolic visual reasoning, CoRR abs/1909.0 (2019). URL: http://arxiv.org/abs/1909.09065.

[16] G. Ciatto, M. I. Schumacher, A. Omicini, D. Calvaresi, Agent-based explanations in AI: Towards an abstract framework, in: D. Calvaresi, A. Najjar, M. Winikoff, K. Främling (Eds.), Explainable, Transparent Autonomous Agents and Multi-Agent Systems, volume

12175 of *Lecture Notes in Computer Science*, Springer, Cham, 2020, pp. 3–20. doi:10.1007/978-3-030-51924-7_1, Second International Workshop, EXTRAAMAS 2020, Auckland, New Zealand, May 9–13, 2020, Revised Selected Papers.

[17] G. Ciatto, D. Calvaresi, M. I. Schumacher, A. Omicini, An abstract framework for agent-based explanations in AI, in: 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2019), International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1816–1818. Extended Abstract.

[18] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. F. M. Ng, B. Liu, P. S. Yu, Z. Zhou, M. S. Steinbach, D. J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowledge and Information Systems 14 (2008) 1–37. doi:10.1007/s10115-007-0114-2.

[19] R. Andrews, J. Diederich, A. B. Tickle, Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge-Based Systems 8 (1995) 373–389. doi:10.1016/0950-7051(96)81920-4.

[20] M. de Jong, F. Sha, Neural theorem provers do not learn rules without exploration, CoRR abs/1906.0 (2019). URL: http://arxiv.org/abs/1906.06805.

[21] E. Denti, A. Omicini, A. Ricci, tuProlog: A light-weight Prolog for internet applications and infrastructures, in: Lecture Notes in Computer Science, volume 1990, Springer Verlag, 2001, pp. 184–198. doi:10.1007/3-540-45241-9_13.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 2019, pp. 8024–8035. URL: http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.

[23] D. Dua, C. Graff, UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.