A Note on Methods for Explainable Malware Analysis

Martin Homola[1,*], Peter Anthony[1], Iveta Bečková[1], Ján Kľuka[1], Ján Mojžiš[5], Peter Švec[2], Štefan Balogh[2], Franco Alberto Cardillo[3], Franca Debole[4], Umberto Straccia[4], Martin Kenyeres[5], Francesco Giannini[6], Michelangelo Diligenti[7], Marco Gori[7], Tomáš Bisták[1], Daniel Trizna[8] and Zekeri Adams[1]

[1]*Comenius University in Bratislava, Mlynská dolina, 84248 Bratislava, Slovakia*
[5]*Institute of Informatics, Slovak Academy of Sciences, Dúbravská cesta 9, 84507 Bratislava, Slovakia*
[2]*Slovak Technical University, Ilkovičova 3, 84104 Bratislava, Slovakia*
[3]*Istituto di Linguistica Computazionale, CNR, Via Giuseppe Moruzzi 1, 56127 Pisa, Italy*
[4]*Istituto di Scienza e Tecnologie dell'Informazione, CNR, Via Giuseppe Moruzzi 1, 56127 Pisa, Italy*
[6]*Faculty of Sciences, Scuola Normale Superiore, Piazza dei Cavalieri 7, 56126 Pisa, Italy*
[7]*Department of Information Engineering and Mathematics, University of Siena, Via Roma 56, 53100, Siena, Italy*
[8]*CSIRT.SK, Ministry of Investment, Regional Development and Informatization, Pribinova 25, 81109 Bratislava, Slovakia*

Abstract
The inevitable rise of machine learning in malware analysis puts forward the need for human-understandable explanations of the learned results. We point out how the ontological representation of malware data provides a suitable language for the construction of such explanations. We then focus on possible methods that enable producing such explanations and we reflect on our experience with them in the context of the EMBER dataset.

Keywords
Malware analysis, explainable AI, ontology, EMBER dataset

1. Introduction

The essential role of machine learning (ML) in malware analysis has now become unquestionable [1, 2, 3, 4, 5, 6, 7, 8, 9]. But it has also become apparent [10, 11, 12] that such an instrument must be implemented in a trustworthy and interpretable fashion – so as to empower malware analysts not only with an efficient tool to classify samples into *malware* and *benign* but also to provide suitable justifications for such a classification. A key asset in this endeavour is the availability of suitable datasets collecting sufficient amounts of preclassified samples of quality data. In the malware domain, this role has been predominantly filled by EMBER [13] and SoReL-20M [14] datasets collecting samples extracted by static analysis of malware and benign PE files (i.e. Windows executable files). Building on the assumption that good justifications require a well-established, shared language to be expressed in, we have developed the *PE Malware Ontology* [15], which provides a suitable vocabulary for their construction and captures essential domain knowledge that may potentially aid both classification and justification tasks.

Proceedings of the Joint Ontology Workshops (JOWO) – Episode XI: The Sicilian Summer under the Etna, co-located with the 15th International Conference on Formal Ontology in Information Systems (FOIS 2025), September 8–9, 2025, Catania, Italy
*Corresponding author.
homola@fmph.uniba.sk (M. Homola); peter.anthony@fmph.uniba.sk (P. Anthony); beckova@fmph.uniba.sk (I. Bečková); kluka@fmph.uniba.sk (J. Kľuka); jan.mojzis@savba.sk (J. Mojžiš); peter.svec1@stuba.sk (P. Švec); stefan.balogh@stuba.sk (Š. Balogh); francoalberto.cardillo@ilc.cnr.it (F. A. Cardillo); franca.debole@isti.cnr.it (F. Debole); umberto.straccia@isti.cnr.it (U. Straccia); martin.kenyeres@savba.sk (M. Kenyeres); francesco.giannini@sns.it (F. Giannini); diligmic@diism.unisi.it (M. Diligenti); marco.gori@unisi.it (M. Gori); bistak5@uniba.sk (T. Bisták); daniel.trizna@csirt.sk (D. Trizna); zekeri.adams@fmph.uniba.sk (Z. Adams)
0000-0001-6384-9771 (M. Homola); 0000-0002-9010-3075 (P. Anthony); 0000-0002-6396-9770 (I. Bečková); 0000-0002-3406-3574 (J. Kľuka); 0000-0002-2196-2271 (J. Mojžiš); 0000-0002-8315-5301 (P. Švec); 0000-0003-0634-9476 (Š. Balogh); 0000-0003-0940-4768 (F. A. Cardillo); 0000-0002-0369-6045 (F. Debole); 0000-0001-5998-6757 (U. Straccia); 0000-0003-2430-1126 (M. Kenyeres); 0000-0001-5892-2049 (F. Giannini); 0000-0002-5805-8032 (M. Diligenti); 0000-0001-6337-5430 (M. Gori); 0009-0005-8715-0574 (T. Bisták); 0009-0006-3413-2409 (Z. Adams)

The explanations/justifications themselves may be obtained in diverse ways, and they may have different forms. We will focus predominantly on methods loosely falling under the umbrella of *structured machine learning* (SML) [16], which enables to find justifications in the form of symbolic expressions with two important properties – (i) they are, to a large extent, *human readable and interpretable* and (ii) they may be used as symbolic classifiers, i.e. they may be *evaluated* as true or false w.r.t. each data sample and possibly some background knowledge base $\mathscr{K}$. In a general SML setting, we assume a formal symbolic language $\mathscr{L}$ where we abstract from the particular syntax and semantics – we only assume that one can express equivalence ($\equiv$), a predicate being true for an individual ($P(e)$), and entailment: an expression being true in a knowledge base ($\mathscr{K} \models \phi$).

Given a knowledge base $\mathscr{K}$, the task of SML is to generate expressions ϕ, which characterize the input samples $E = E^+ \cup E^-$, where E^+ are positive examples and E^- are negative examples. Formally, the goal is to find a target expression ϕ, so that $\mathscr{K} \cup \{TARGET \equiv \phi\} \models TARGET(e)$ for all $e \in E^+$ and $\mathscr{K} \cup \{TARGET \equiv \phi\} \not\models TARGET(e)$ for all $e \in E^-$. Of course, such precise characterization across the whole sample is only possible in theory. In practice we will expect it to "hold" to a certain extent (hopefully as high as possible), and we will rely on standard information retrieval methodology to charaterize the success in terms of accuracy, F1 measure, and false-positive rate.

In this note, we summarize our experience with five different approaches that we have applied on the EMBER dataset with the aim of obtaining suitable justifications for the malware classification task: (1) post-hoc methods such as LIME, SHAP and Anchor, (2) decision tree learning, (3) description logics concept learning, (4) knowledge graph embedding, and (5) logic explained networks. They may be roughly classified into *post-hoc* (1) and *intrinsic* (2–5), while the latter all loosely fall under SML and may be further divided into purely *symbolic* (2–3) and *neuro-symbolic* (4–5) methods.

These methods vary largely in their design, character, computational effectiveness, and potential types of output. It is not our goal in this note to conduct a rigorous comparison; instead, we summarize our experience, highlight their strengths and limitations and focus on understanding and comparing especially the types of output these methods may potentially provide. In the consecutive discussion we focus on further questions open in light of our research, potential other methods to investigate in this context, the question of the quality of the data and how it can be improved by dynamic malware analysis, and finally also on the issues related to presentation of the obtained explanations to human users.

2. Data and Ontology

The experiments described in the following sections have been performed on datasets derived from EMBER [13], a benchmark dataset for training static malware detection models. EMBER contains structured entries describing 1.1 million Windows Portable Executable files (PE files). We have only used samples labelled as malware or benign (400,000 each) ignoring unlabelled samples.

Each EMBER sample is described by a JSON object in terms of properties of the respective PE file determined by static analysis (i.e., without actually running the sample). These include general information (file size, presence of a digital signature, the numbers of imported and exported functions, etc.), header information (the target architecture, linker version, various timestamps), descriptions of the PE file's sections (name, content type, access rights, entropy), lists of imported functions per DLL, and the list of exported functions. There are multiple numerical properties, such as a byte histogram, byte-entropy histogram, or simple statistics for various kinds of strings found in the file.

2.1. PE Malware Ontology and Ontological Datasets

We have designed the PE Malware Ontology [15] to semantically describe EMBER and similar data sources on static malware analysis of PE file samples with focus on interpretability. This OWL 2 ontology comprises 195 classes, 6 object properties, and 10 data properties. It is expressible in the light-weight DL-Lite$_{\text{core}}(D)$ logic underlying the OWL 2 QL profile. The core classes and properties are depicted in Fig. 1.

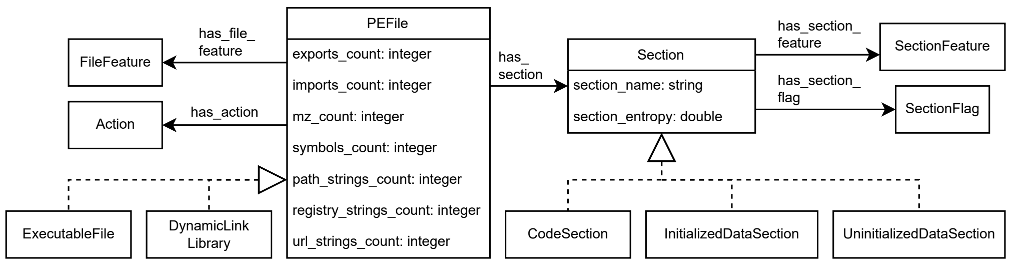

Figure 1: PE malware ontology core classes and properties

A sample is described as an instance of the central `PEFile` class and related to its constituent sections, instances of the `Section` class. Files and sections are further classified by their type and described by relationships to instances of the `FileFeature`, `SectionFeature`, and `SectionFlags` classes. The former two subsume, respectively, 15 and 3 feature classes relevant to malware detection from the domain experts' point of view (e.g., entry point located in a non-executable section, presence of often exploited kinds of sections, high section content entropy). The `SectionFlags` class subsumes classes of access-control flags whose unusual combination may also indicate malware. The richest characterization of the sample comes from relating it to `Action` instances, representing the actions that the sample can perform. These abstract and standardize the lists of functions imported by the sample from system DLLs. Each action is classified in one of 139 classes derived from the standard MAEC vocabulary of malware actions [17]. Further abstraction is provided by higher-level action classes that partition the lower-level classes into 17 categories (networking, access management, system manipulation, ...).

We have produced and published a collection of 31 ontological datasets combining the PE malware ontology with transformed EMBER data, each with an equal proportion of malware and benign samples. Aiming towards computationally intensive symbolic methods and *k*-fold cross-validation, the collection contains datasets of 1,000, 10,000, and 100,000 samples (10 datasets per size), and the full dataset describing all 800,000 labelled EMBER samples.

2.2. Vectorized Dataset

For experiments with ML methods requiring traditional datasets with numerical feature vectors, we have created a dataset reflecting the binary nature of class membership and feature relationships in the ontological dataset, as well as the inability of many ontology-based methods to work with numeric values. The samples in this dataset are thus mapped to a 0/1-valued feature space. Its dimensions correspond to selected concept expressions being true of the sample. These expressions are either classes (e.g., the feature `is_dll` corresponds to `DynamicLinkLibrary`) or (chains of) existential restrictions expressing the presence of particular features in the sample or one of its sections (e.g., `has_nonstandard_mz` stands for ∃`has_feature`.`NonstandardMZ`; `act_lf_execute_file` for ∃`has_action`.`ExecuteFile`; and `sect_rdata_has_CNT_INITIALIZED_DATA` for ∃`has_section`.(∃`name`.{"`rdata`"} ⊓ `InitializedDataSection`)). As the targeted methods can handle large amounts of data, a single vectorized dataset based on all 800,000 labelled EMBER samples was produced.

3. Explainable Malware Classification Methods

3.1. Post-Hoc Methods

Post-hoc explanation methods try to extract and unravel the rationale behind black-box models' decisions after training, hence the term post-hoc explainers. To apply a post-hoc explainer, a black-box model is first trained to perform the classification. Then the post-hoc explainer is applied to extract the explanations for their decisions, providing insights and enhancing trust in the system. Popular in literature is the use of LIME (Local Interpretable Model-Agnostic Explanations) [18], SHAP (SHapley

Additive exPlanations) [19], and Anchor [20] explainers, due to their model-agnostic nature.

LIME explains individual predictions by approximating the black-box model locally with a simple interpretable model (e.g., linear regression). It perturbs a given instance to obtain modified inputs and observe changes in model's output to determine which features contribute most to the prediction.

SHAP assigns each feature an importance value for a specific prediction using principles from cooperative game theory. It considers all possible feature combinations to compute how much each feature contributes on average, making it well-suited for capturing both local and global feature importance.

Anchor produces high-precision rule-based explanations by identifying conditions (anchors) that are sufficient to "lock in" a model's prediction. These rules are instance-specific but are designed to be precise and easy to understand, typically in the form of IF–THEN logic.

We explored the effectiveness of these explainers in the malware domain. We first employed the entire 800,000 labelled samples of the vectorized version of the EMBER dataset described in Section 2.2, split into 60:20:20 as training, validation and test set. Training set was used to train a multi-layer perceptron (MLP) with two hidden layers (512 and 256 units) using ReLU activation and dropout rates of 0.4 and 0.2 for regularization. The model was optimized using the Adam optimizer (learning rate = 0.001) with binary cross-entropy as the loss function, and batch size of 100. Validation set was used to compute the validation loss, which served for the early-stopping criterion. The model achieved F1-score of 91% and false positive rate of 7% on the test set. Using the test set, we then extracted the explanations using SHAP and Anchor. Figure 2 shows the resulting feature importance scores produced by SHAP, highlighting `has_debug`, `is_dll`, and `has_section_high_entropy` as the top three influential features. Formula (1), a sample rule obtained from Anchor, rewritten in DL syntax, shows that the simultaneous presence of high-entropy sections, a write-execute memory region, and a file-writing behaviour indicates a strong likelihood of malware. Similarly, Formula (2) shows that the absence of DLL characteristics, relocation information, and high entropy in the UPX1 section—combined with the presence of a write-execute section—indicates a potential malware instance.

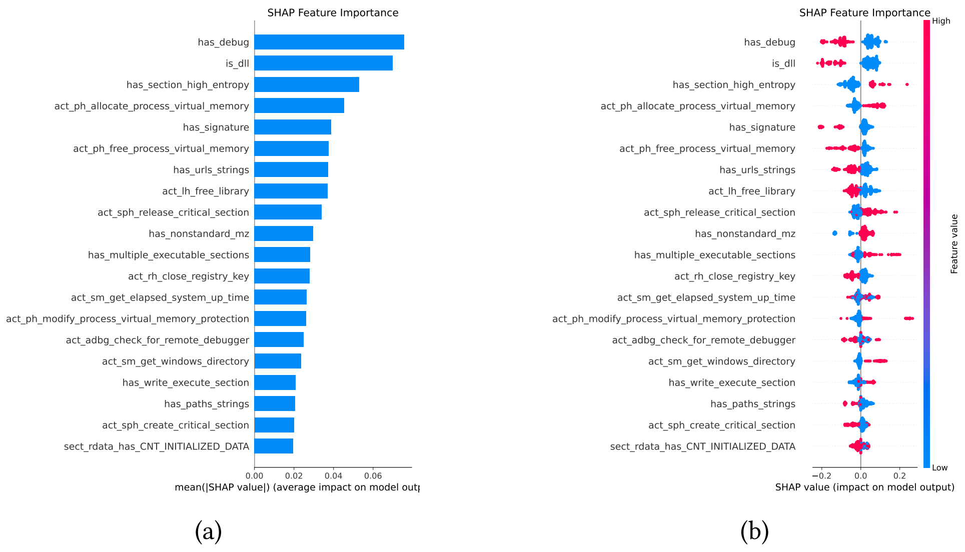

Figure 2: SHAP explanation visualizations: (a) Summary plot, (b) Boxplot.

$$\begin{aligned}&\exists\mathtt{has\_section}.\exists\mathtt{has\_section\_feature}.\mathtt{HighEntropy} \sqcap \exists\mathtt{has\_file\_feature}.\mathtt{WriteExecuteSection}\\&\quad\sqcap \exists\mathtt{has\_action}.\mathtt{WriteToFile}\end{aligned} \tag{1}$$

$$\begin{aligned}&\neg\mathtt{DynamicLinkLibrary} \sqcap \exists\mathtt{has\_file\_feature}.\mathtt{Relocations}\\&\quad\sqcap \exists\mathtt{has\_section}.(\exists\mathtt{name}.\{\texttt{"upx1"}\} \sqcap \exists\mathtt{has\_section\_feature}.\mathtt{HighEntropy})\end{aligned} \tag{2}$$

Furthermore, we propose a hybrid explainability framework that extends SHAP and Anchor. While

SHAP and LIME provide first-order or instance-specific explanations, our method enhances SHAP to provide second-order pairwise feature interactions and extends Anchor to extract symbolic decision rules characterizing malware behaviour across samples. These global rules support not only interpretability but also classification of previously unseen data. Figure 3 shows the result of the extended SHAP framework on the trained MLP model. The plot shows the average change in the predicted probability for a number of possible feature combinations. Positive values indicate a shift toward the model predicting malware, while negative values indicate a shift toward benign classification.

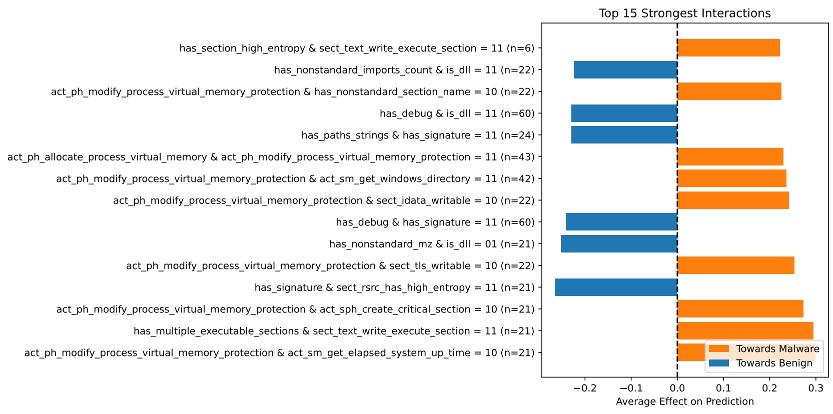

Figure 3: Interaction effects between the binary features `has_section_high_entropy` and `is_dll` on the model's prediction scores.

We applied the framework to the vectorized EMBER dataset, the extracted rules by the extended Anchor retained 80% F1-score with 12% false positive rate. The resulting rules offered actionable insights and contributed a transparent reasoning layer over high-performing black-box models.

In conclusion, while the application of the post-hoc methods enhances interpretability, it is important to note that the fidelity of the generated explanations has been questioned in the literature [21, 22].

3.2. Decision-Tree Learning

Decision trees originating in classical machine learning can be used to construct a hierarchical tree model of conditions composed of the root, nodes, and leaves. The most decisive condition of such a tree is the condition located in the root of the tree. In our study [23], we trained a decision tree model C4.5 on the vectorized dataset (cf. Sect. 2.2) in order to construct a hierarchy tree. The tree can be used to extract rules as conjunctions of the conditions on the paths from the root to the leaves. The leaves contain information on the number of samples classified by the respective path. The rules can be extracted based on 1) the leaves with the highest proportion of successfully classified records or 2) the shortest paths from the root to the leaves, resulting in the most compact rules. In [23], we demonstrated the extracted rules based on the shortest paths for benign and malware samples. With the C4.5 model, our own proprietary ESFS feature selection method and 200 features, we created a tree model which classifies malware samples with accuracy of 91.61%, true positive rate of 92.3%, false positive rate of 9.1%, and macro F1 of 91.65%. Formulas (3) and (4) show examples of rules, rewritten in DL syntax, for benign and malware samples respectively, which correspond to subtrees shown in Fig. 4.

$$\begin{aligned}\neg\texttt{DynamicLinkLibrary} \sqcap\ & \neg\exists\texttt{has\_section}.(\exists\texttt{name}.\{\texttt{"pdata"}\} \sqcap \texttt{InitializedDataSection})\\ & \sqcap \exists\texttt{has\_section}.(\exists\texttt{name}.\{\texttt{"didat"}\} \sqcap \exists\texttt{has\_section\_flag}.\texttt{Writable})\\ & \sqcap \neg\exists\texttt{has\_section\_feature}.\texttt{WriteExecuteSection}\end{aligned} \tag{3}$$

$$\begin{aligned}\neg\texttt{DynamicLinkLibrary} \sqcap\ & \neg\exists\texttt{has\_section}.(\exists\texttt{name}.\{\texttt{"pdata"}\} \sqcap \texttt{InitializedDataSection})\\ & \sqcap \neg\exists\texttt{has\_section}.(\exists\texttt{name}.\{\texttt{"didat"}\} \sqcap \exists\texttt{has\_section\_flag}.\texttt{Writable})\\ & \sqcap \exists\texttt{has\_section}.(\exists\texttt{name}.\{\texttt{"coderpub"}\} \sqcap \exists\texttt{has\_section\_flag}.\texttt{Readable})\end{aligned} \tag{4}$$

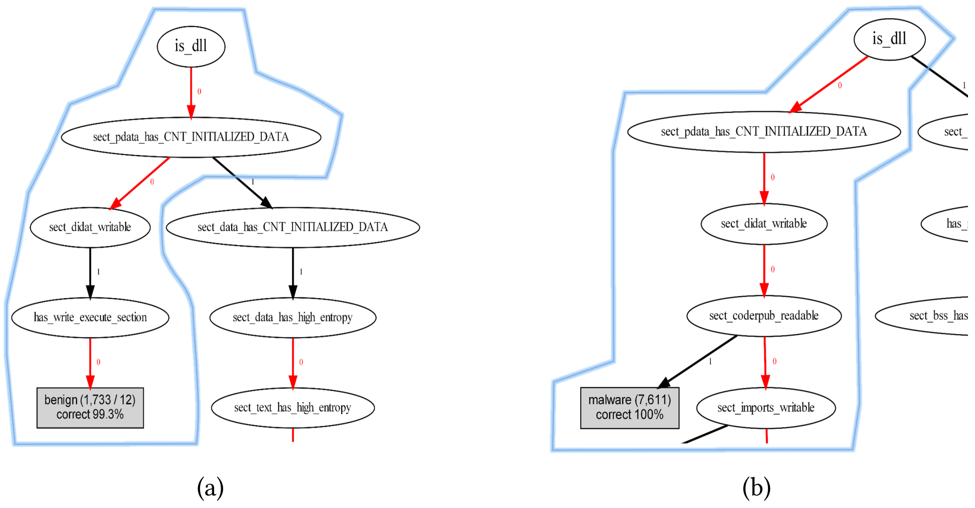
is_dll
sect_pdata_has_CNT_INITIALIZED_DATA
sect_didat_writable
sect_data_has_CNT_INITIALIZED_DATA
has_write_execute_section
sect_data_has_high_entropy
benign (1,733 / 12)
correct 99.3%
sect_text_has_high_entropy
(a)
is_dll
sect_pdata_has_CNT_INITIALIZED_DATA
sect_didat_writable
sect_coderpub_readable
malware (7,611)
correct 100%
sect_imports_writable
sect_bss_has
(b)

Figure 4: Two subtrees generated from the model: (a) Benign subtree, (b) Malware subtree. To ease the navigation, red colored edges represent the absence of a feature and black colored edges the presence. See Sect. 2.2 for feature explanation.

3.3. Concept Learning

Concept learning [24] is a method that enables the learning of a class expression that matches a set of examples in a knowledge base. That is, it is an SML method in which the learned expression is a complex concept expression in a description logic [25]. Generally, concept learning algorithms search through a space of all possible concepts using *refinement operator* [26], which for a given concept returns a set of refined concepts and *heuristics*, that controls how the search space is traversed. In our work, we investigated four main concept learning algorithms that are available in *DL-Learner* [27] framework: OCEL [26], CELOE [28], PARCEL [29] and SPACEL [30].

We performed multiple experiments using previously mentioned concept learning algorithms and *PE Malware* ontology [15, 31]. In these experiments, we used datasets containing 1000 samples (of which 500 were positive examples and 500 negative examples). For evaluation, we used k-fold cross validation, with $k = 5$. We achieved an F1 score of 74% and an FP rate of 22% for the OCEL algorithm. This algorithm produces a single class expression that characterizes the whole dataset. The expression of the class for OCEL can be seen in Formula (5). For PARCEL algorithm, which on the other hand produces partial definitions (that are combined together using disjunction), we achieved F1 score of 77% and FP rate of 15%. An example of a partial definition can be seen in Formula (6). Parallel algorithms such as PARCEL or SPACEL usually produce significant amounts of shorter class expressions.

$$\begin{aligned}\texttt{ExecutableFile} \sqcap \exists \texttt{has\_file\_feature}.(&\texttt{MultipleExecutableSections} \sqcup \texttt{NonstandardMZ})\\ &\sqcap \exists \texttt{has\_section}.\exists \texttt{has\_section\_flag}.\texttt{Writable}\\ &\sqcap \leqslant 1 \texttt{has\_action}.(\texttt{AcceptSocketConnection} \sqcup \texttt{DirectoryHandling}\\ &\qquad \sqcup \texttt{EnumerateThreads} \sqcup \texttt{GetProcessCurrentDirectory} \sqcup \texttt{OpenMutex})\end{aligned} \tag{5}$$

$$\geqslant 3 \texttt{has\_section}.(\exists \texttt{has\_section\_feature}.\texttt{NonstandardSectionName} \sqcap \exists \texttt{has\_section\_flag}.\texttt{Writable}) \tag{6}$$

In [32], we explored the possibility of improving the efficiency of concept learning by focusing on individual malware families. Previous results showed that concept learning lags significantly behind standard machine learning algorithms. Through these experiments, we concluded that concept learning struggles to detect malware in general, as it involves a large search space. For this reason, we decided to train models for individual malware families, since a set of malware samples from a specific family should contain multiple common features (i.e., a smaller search space) compared to all malware samples in the dataset. As part of these experiments, we trained the classifiers separately for five families and combined the results using disjunction. We were able to obtain models with an F1 score of 91% and FP rate of 0.24% for the OCEL algorithm, and an F1 score of 93% and an FP rate of 11% for the PARCEL algorithm.

3.4. Fuzzy Concept Learning

One disadvantage of regular DLs is their inability to work on top of numeric data values. For example, in the original dataset the imports count ≥ 0 is given for each sample and the entropy value 0−1 is given for each section. To be able to include these important features in the concept learning process, we had to pre-process the input data and generate a "crisp" (i.e. binary) derived file feature `low_imports_count` and section feature `high_entropy` based on a suitable threshold. Fuzzy DL-learner [33, 34] is able to learn fuzzy concept descriptions in $\mathscr{EL}$(D), which also supports automatic learning of "fuzzy datatypes" such as `imports_count_veryLow` or `entropy_high` based on actual numeric values in the dataset [35]. The final learned expressions are fuzzy classification rules $C \sqsubseteq_c$ TARGET with the meaning that the learned fuzzy concept C classifies under the target concept TARGET with confidence $0 \leq c \leq 1$. The Fuzzy DL-Learner system implements various learning algorithms including fuzzy DL-FOIL [36, 37], pFOIL-DL [38], Fuzzy OWL Boost [33] and fuzzy PN-OWL [34]. The latter, currently the most effective one, uses a novel learning strategy, alternating a P-stage in which the algorithm tries to learn positive fuzzy classification rules covering as much of the sample as possible with high precision, and a N-stage in which the algorithm tries to learn fuzzy classification rules that rule out as many false positives as possible that are covered by the rules learned in the P-stage. The rules are then combined via an aggregation function.

In the experiments both algorithms were compared on one of the EMBER_1k datasets (which was one of the two largest datasets included in the study). PN-OWL outperformed fuzzy DL-FOIL by achieving the F1 of 73.4% vs. 70.4%. One of the learned fuzzy DL classification rules can be seen in Formula (7).

$$\begin{gathered}\texttt{ExecutableFile} \sqcap \exists \texttt{has\_action.CreateWindow} \sqcap \exists \texttt{imports\_count.imports\_count\_veryLow} \\ \sqcap \exists \texttt{urls\_strings\_count.urls\_strings\_count\_low} \sqsubseteq_{0.986} \texttt{Malware}\end{gathered} \tag{7}$$

Malware experts particularly evaluated the expressions featuring the learned fuzzy datatypes as useful and insightful.

3.5. KG Embedding

Knowledge base embedding (KBE), particularly knowledge graph embedding (KGE), enables embedding of structured semantic information from malware ontologies into continuous vector spaces, preserving relationships among malware attributes and behaviours. In our study [39], we used the RotatE embedding model [40], which maps entities and relations into vector space using rotational transformations, allowing it to model complex relational patterns with high fidelity. We utilized one of the 1000-sample datasets of the ontology built on the EMBER dataset as described in Section 2.1 for our experiment. Triples were extracted from the ontology via SPARQL queries, and these were then embedded using the PyKEEN library.

We evaluated three KGE methods–TransE, BoxE, and RotatE–with RotatE model, trained with 128-dimensional embeddings, performing best, with classification accuracy of 80% and 16% false positive rate. Hence, it was selected for explainable rule generation. To extract interpretable rules from the RotatE model, we first retrieved top-scoring triples associated with the 'Malware' class. A recursive algorithm then generated conjunctions of these triples that showed high classification performance. Finally, a disjunction of the top conjunctions—limited to six for complexity control—was selected as the final rule. This rule was expressed in DL syntax as shown in Formula 8, and evaluated on the test set achieving accuracy of 76% with 22% false positive rate. The explanation generation approach employed here is model-specific, as opposed to methods such as the fuzzy concept learning or decision trees which are intrinsically explainable, or post-hoc methods such as LIME, SHAP and Anchor which are model-agnostic.

$$\begin{aligned}\texttt{Malware} \sqsubseteq\ & (\neg\exists \texttt{has\_file\_feature.Debug} \sqcap \exists \texttt{has\_action.ProcessHandling}) \\ & \sqcup (\texttt{ExecutableFile} \sqcap \exists \texttt{has\_file\_feature.MultipleExecutableSections})\end{aligned} \tag{8}$$

3.6. Logic Explained Networks

Motivated by the need to retain the high performance of deep learning models while ensuring interpretability and explanation fidelity, we employ Logic Explained Networks (LENs) [21], which offer transparent decision-making without compromising predictive accuracy. Logic Explained Networks (LENs) are interpretable-by-design neural architectures that provide human-understandable First-Order Logic (FOL) explanations for their decisions, aiming to combine the high performance of black-box deep learning models with interpretability. Unlike traditional interpretable models such as decision trees, linear regression, or k-Nearest Neighbours, which often struggle with performance or scalability, Logic Explained Networks (LENs) achieve high predictive performance while producing explanations in the form of logical rules. In contrast to post-hoc explanation methods typically applied to deep learning models, LENs are intrinsically interpretable, ensuring that the generated explanations are inherently faithful to the model's decision-making process.

LENs require human-understandable predicates as inputs, such as tabular data or concepts extracted from raw data. They determine the relevant subset of the input concepts that account for their decisions through an ad-hoc pruning and regularization technique. Formally, a LEN f is a function, mapping input concepts represented in $[0, 1]^d$ to one or more output classes ($c \geq 1$), simultaneously supporting classification and explanations framed as first-order logic (FOL) rules built upon these concepts.

In our study [41], we employed a tailored variant called Tailored-LENs, which introduces a threshold-based optimization strategy to refine the global explanations, thereby reducing false positives and improving explanation fidelity. This method enables LENs to scale to large datasets while maintaining readable and compact explanations, distinguishing them from both standard LENs and other interpretable models. We applied Tailored-LENs to the vectorized EMBER dataset. The dataset was used with varying numbers of selected features, with experiments run on 600,000 samples for training and 200,000 for testing using our proprietary feature selection pipeline. The LENsmodel achieved an accuracy of 92.32%, precision of 93.35%, and false positive rate of 6.52%. Generated local rules, such as an example in DL syntax in Formula (9), were validated by domain experts as indicative of packed or unsigned malware. Compared to other interpretable approaches, Tailored-LENs outperformed both concept learning methods and decision trees, while achieving performance on par with state-of-the-art deep learning models.

$$\exists \texttt{has\_section}.\exists \texttt{has\_section\_feature.HighEntropy} \sqcap \exists \texttt{has\_feature.Signature} \tag{9}$$

4. Comparison

While the majority of machine learning research is focused on achieving the best possible performance regarding the task being solved, in explainable AI it is necessary to also factor in the desirable properties of explanation methods themselves, such as their applicability, efficiency or expressivity of the produced explanations. Therefore, classifiers and explainers need to be evaluated with respect to each other. In this section, we discuss our experience with applying mentioned methods on the EMBER dataset. A concise summary of their properties can be found in Table 1.

Malware Classification Accuracy. As we can observe, many of the methods achieved both accuracy and F1 91-92% on par with what was showed earlier in the literature [42]. Less performing methods included fuzzy concept learning and KG embedding, however here the experiments were limited to the smallest 1k datasets, limiting the potential generalization power. The best FP rates were around 6–7% which is not low but may be expected when working with real-world datasets. All in all a variety of

Table 1
Comparison of previously mentioned methods in terms of classifiers' performance and explainers' properties.

	Performance			Properties			
Method	accuracy	F1	FP	# of data	data representation	explanation type	coverage
LIME + MLP	91.30%	91%	7%	$\approx 10^6$	vectorized	feature importance	local
SHAP + MLP	91.30%	91%	7%	$\approx 10^6$	vectorized	feature importance	local and global
Anchor + MLP	91.30%	91%	7%	$\approx 10^6$	vectorized	rule-based	local and global (ours)
Decision Tree	91.61%	91.65%	9.1%	$\approx 10^6$	vectorized	rule-based	global (partial)
Concept Learning	83–92%	84–93%	0–18%	$\approx 10^4$	knowledge-base	rule-based	global
Fuzzy CL		70–73%		$\approx 10^3$	knowledge-base + numeric	fuzzy rule-based	global
KG embedding	80%		16%	$\approx 10^3$	knowledge-graph	rule-based	global
LEN	92.32%	92%	6.52%	$\approx 10^6$	vectorized	rule-based	local and global

methods achieved good enough performance and the most suitable method may be selected among these based on other criteria

Data Processing Efficiency. From the point of view of data efficiency, concept learning methods were performing the worst; fuzzy concept learning further added complexity, only being able to process thousands of inputs in a reasonable time. It may be useful to investigate what target expressivity of DL concepts is necessary for the task and consecutively to explore reducing the expressivity in the configuration of DL-Learner or even trying out specific systems designed for tractable DLs, such as SPELL [43].

Required Data Representation. We have developed an ontological representation for the datasets, and thus it is natural to apply methods that can directly take knowledge-graph data in RDF format as inputs. This has additional advantages in no data loss on the input (by required feature selection), potential exploitation of ontological domain knowledge, and rich direct representation for the explanations. Not all methods however support this and for some of them the inputs ned to be vectorized. As the authors note [21], LENs require vectorized data, however, individual features must correspond to meaningful, interpretable concepts. To certain extent, this is also true for all other methods with vectorized inputs – to be able to construct the explanations. In our use case, this meaningful interpretation is derived from the ontology.

Expressivity of Explanations. From the expressivity point of view, post-hoc methods are the least powerful. LIME and SHAP explain their predictions via importance scores. Anchor is better thanks to explanations being rule-based, however, the "vanilla" version provides only local explanations. Our extended version alleviates this problem by generating global rules. The downside of these three methods is the fact that, as they are post-hoc, they may suffer from fidelity issues.

Other used methods all provide rule-based explanations. The highest expressivity is offered by concept learning, depending on the system and particular algorithm used. The target expressivity of the searched expressions may also be configured by the user of concept learning systems. One relevant factor shows to be, whether disjunctions are supported or allowed. For example the fuzzy DL leaner does not support it, which may be one of the reasons of relatively low achieved classification permanence. The issue needs to be further investigated. However, its capability of learning fuzzy datatypes and subsequently generating fuzzy rules is beneficial for human users. While KG embedding is capable of generating rules with disjunctions. Rules from individual branches of decision trees are represented as conjunctions, but it is possible to combine rules from multiple branches using disjunctions. LENs generate local rules in the form of conjunctions and these are then combined into disjunctions, which represent the global rules. However, the resulting global rules are often too complex. We tried to tackle this problem in [41].

Fidelity. The fidelity of an explanation is the measure of how faithful it is to the model that it is explaining, in the sense that if the explanations were to be used for classification, how well would the outputs match the original model's outputs. Methods such as decision trees, LENs, and symbolic approaches like concept learning are intrinsically interpretable, so their explanations should, theoretically, have 100% fidelity. While for some methods (e.g., concept learning) it is true, some level of fidelity can be lost in other methods due to the process of simplifying the explanations (for example in LENs), or due to explanations being only partial (individual branches of decision trees). On the other hand, post-hoc methods, especially the model-agnostic methods like SHAP, LIME and Anchor have lower fidelity as they use approximations of black-box models and, therefore, are not able to fully capture their rationale.

5. Discussion and Conclusions

5.1. Summary

We have focused on the problem of classification-based malware characterization under the assumption that it must be complemented by some form of suitable justification. We have argued in favour of a symbolic treatment of such justifications, rooted in a suitable ontological vocabulary for the domain. To this end, we have reported about the proposed *PE Malware Ontology* and on our experience with applying multiple approaches mainly on semantically treated EMBER data to achieve such semantically-justified classification that could improve the overall utility and especially the trust of malware experts in the classification results.

While this note is not intended as a rigorous formal comparison, we were able to observe and point out distinct properties of each approach. In summary, on the one hand, symbolic methods, most notably concept learning, yield very expressive justifications with high fidelity, but they are computationally very demanding, limiting the overall amount of data feasible to be processed. We were able to partially mitigate this problem by breaking down the data based on malware families. In the future we plan to try exploiting methods such as clustering to decrease the datasets with the goal to decrease the necessary size of the input datasets. We also plan to apply newer, potentially more efficient concept learning tools [43, 44] based on SAT solvers. Other approaches are computationally more efficient, but the yielded justifications are less expressive and/or practical, or even somewhat experimental in nature as in the case of KB embedding. Neural symbolic methods such as LENsare promising w.r.t. the handled amounts of data – on the other hand, the expression extraction (as much as in the case of e.g. decision-tree learning) does not take the ontology into account.

5.2. Other Potential Methods

Neural networks and deep learning have now gained undisputed prominence in machine learning applications with many novel research streams that are worth exploring. Notably, a large effort is currently dedicated to alleviating their black-box nature and enabling their applications even in mission-critical areas such as malware analysis. This is already underscored by the result we were able to obtain by LEN.

Graph Neural Networks (GNNs) [45] are a promising method to apply directly over knowledge-graph inputs as they eliminate the need for feature extraction and vectorization. Different approaches to achieve explainability of GNNs were studied [46]. Among the most promising appears to be XGNNs [47] which enables to identify sub-graph patterns relevant for certain decisions. Similarly MGNNs [48] enables to extract explanations in the form of datalog rules.

While capable of learning expressive ontology-based characterizations of malware samples, one of the main drawbacks of symbolic methods, such as concept learning, lies in its inherently high computational demand. This greatly limits the amount of data that these methods may possibly process. Here, a promising approach could be in ontology-based concept recognition from neural networks' activation patterns [49]. This method would enable the use of one of the highly effective neural classifiers for the malware classification task and recognition of the ontology concepts that are relevant for a given decision. Such an approach would not reach the expressivity of complex concept descriptions outputted by concept learners that can be used in classification rules; to some extent this can be addressed by integration of concept learning into the process [50].

Completely different type of explanations is provided by *counterfactual examples* [51, 52]. Counterfactual examples are local explanations in the form of what minimal change would need to be applied to a certain input in order for the model's output to be changed. This method offers good flexibility; it has been applied in intrinsic, hybrid, and even model-agnostic scenarios. Moreover, the contrastive nature of the explanations could be beneficial for human users [53]. One of the most promising tools to obtain counterfactual, contrastive, or even more general *argumentative* explanations are argumentation frameworks [54].

5.3. Quality of Data

All the works above focused on EMBER dataset and thus on data derived from static malware analysis. Powerful black-box classifiers are known to achieve accuracies as high as 92 % on this data [42]. This is partly due to their high efficiency but partly also due to the inclusion of features with low interpretability, e.g. byte histograms, which are unsuitable for inclusion in justifications and therefore we have excluded them from the semantically-treated data [15].

Also, the feedback we obtained from malware experts suggests that many of our learned justifications are still rather simple and less insightful than they would hope for. Given that these concept expressions do approximate best the available data, we conjecture that more detailed data could potentially lead to more informative concept expressions.

It is well known that *dynamic malware analysis* [55], which requires running the evaluated samples in a protected environment and observing their behaviour, allows for more efficient malware detection, exactly for this reason. Unfortunately, suitable datasets based on dynamic malware analysis are not yet as readily available. A suitable ontological representation is part if our ongoing work.

5.4. Usefulness to Human Users

Throughout this work we have presented a number of examples of different types of explanations we were able to obtain by the diverse methods. Some are feature importance values, but most of them have the form of logical expressions (formulas) in some formal language. Their expressivity and complexity vary, and the question is apparent – what type and size of explanations is actually the most useful to the human users. Higher expressivity and complexity of these expressions may offer better classification power over the sample, and to some extent they may be more informative than overly simple expressions. However they may be also more expensive to compute and some of the expressive description logic constructs may be harder to comprehend. We were able to obtain some preliminary feedback from domain experts as discussed above, but it is apparent that wider and more rigorous studies are required to better understand these issues in order to steer the research on explanation extraction methods and also to answer how the explanations are to be presented to the human users (e.g. if they should be translated into natural language or otherwise accommodated).

Acknowledgments

Z. Adams, P. Anthony, Š. Balogh, I. Bečková, M. Homola, J. Kľuka, and P. Švec are funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I05-03-V02-00064. F. Giannini is funded by the Partnership Extended PE00000013 – "FAIR - Future Artificial Intelligence Research" – Spoke 1 "Human-centered AI", and ERC-2018-ADG G.A. 834756 "XAI: Science and technology for the eXplanation of AI decision making".

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] D. Ucci, L. Aniello, R. Baldoni, Survey of machine learning techniques for malware analysis, Computers & Security 81 (2019) 123–147. doi:10.1016/j.cose.2018.11.001.

[2] S. Pramanik, H. Teja, EMBER – Analysis of malware dataset using convolutional neural networks, in: 2019 Third International Conference on Inventive Systems and Control (ICISC), 2019, pp. 286–291.

[3] D. Gibert, C. Mateu, J. Planes, The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, Journal of Network and Computer Applications 153 (2020) 102526.

[4] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, M. Xu, A survey on machine learning techniques for cyber security in the last decade, IEEE Access 8 (2020) 222310–222354.

[5] Y. Supriya, G. Kumar, D. Sowjanya, D. Yadav, D. L. Kameshwari, Malware detection techniques: A survey, in: 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), 2020, pp. 25–30.

[6] J. Singh, J. Singh, A survey on machine learning-based malware detection in executable files, J. Syst. Archit. 112 (2021) 101861.

[7] E. Raff, W. Fleshman, R. Zak, H. S. Anderson, B. Filar, M. McLean, Classifying sequences of extreme length with constant memory applied to malware detection, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 9386–9394.

[8] P. Aggarwal, S. F. Ahamed, S. Shetty, L. J. Freeman, Selective targeted transfer learning for malware classification, in: 2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), 2021, pp. 114–120.

[9] U. Tayyab, F. B. Khan, M. H. Durad, A. Khan, Y. S. Lee, A survey of the recent trends in deep learning based malware detection, J. Cybersecur. Priv. 2 (2022) 800–829.

[10] A. Mills, T. Spyridopoulos, P. Legg, Efficient and interpretable real-time malware detection using random-forest, in: 2019 International conference on cyber situational awareness, data analytics and assessment (Cyber SA), 2019, pp. 1–8.

[11] B. Marais, T. Quertier, C. Chesneau, Malware analysis with artificial intelligence and a particular attention on results interpretability, in: Distributed Computing and Artificial Intelligence, Volume 1: 18th International Conference, DCAI 2021, Salamanca, Spain, 6-8 October 2021, volume 327 of *LNNS*, Springer, 2021, pp. 43–55.

[12] J. Dolejš, M. Jureček, Interpretability of machine learning-based results of malware detection using a set of rules, in: Artificial Intelligence for Cybersecurity, Springer International Publishing, 2022, pp. 107–136.

[13] H. S. Anderson, P. Roth, EMBER: an open dataset for training static PE malware machine learning models, 2018. arXiv:1804.04637.

[14] R. Harang, E. M. Rudd, SOREL-20M: A large scale benchmark dataset for malicious PE detection, arXiv preprint arXiv:2012.07634 (2020).

[15] P. Švec, Š. Balogh, M. Homola, J. Kľuka, T. Bisták, Semantic data representation for explainable Windows malware detection models, arXiv preprint arXiv:2403.11669 (2024).

[16] P. Westphal, L. Bühmann, S. Bin, H. Jabeen, J. Lehmann, SML-Bench – A benchmarking framework for structured machine learning, Semantic Web 10 (2019) 231–245.

[17] MITRE Corp., Malware attribute enumeration and characterization, 2020. URL: https://maecproject.github.io/, [Online; accessed 2022-05-15].

[18] M. T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": Explaining the predictions of any classifier, 2016.

[19] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in Neural Information Processing Systems 30 (2017) 4765–4774.

[20] M. T. Ribeiro, S. Singh, C. Guestrin, Anchors: High-precision model-agnostic explanations, Proceedings of the AAAI Conference on Artificial Intelligence 32 (2018).

[21] G. Ciravegna, P. Barbiero, F. Giannini, M. Gori, P. Liò, M. Maggini, S. Melacci, Logic explained networks, Artificial Intelligence 314 (2023) 103822.

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

[23] J. Mojžiš, M. Kenyeres, Interpretable rules with a simplified data representation-a case study with the ember dataset, in: Proceedings of the Computational Methods in Systems and Software, Springer, 2023, pp. 1–10.

[24] J. Lehmann, P. Hitzler, Concept learning in description logics using refinement operators, Machine Learning 78 (2010) 203–250.

[25] F. Baader, I. Horrocks, C. Lutz, U. Sattler, An Introduction to Description Logic, Cambridge University Press, 2017.

[26] J. Lehmann, Learning OWL class expressions, volume 22, IOS Press, 2010.

[27] J. Lehmann, Dl-learner: learning concepts in description logics, The Journal of Machine Learning Research 10 (2009) 2639–2642.

[28] L. Bühmann, J. Lehmann, P. Westphal, S. Bin, Dl-learner structured machine learning on semantic web data, in: Companion Proceedings of the The Web Conference 2018, 2018, pp. 467–471.

[29] A. C. Tran, J. Dietrich, H. W. Guesgen, S. Marsland, An approach to parallel class expression learning, in: Rules on the Web: Research and Applications: 6th International Symposium, RuleML 2012, Montpellier, France, August 27-29, 2012. Proceedings 6, Springer, 2012, pp. 302–316.

[30] A. C. Tran, J. Dietrich, H. W. Guesgen, S. Marsl, Parallel symmetric class expression learning, Journal of Machine Learning Research 18 (2017) 1–34.

[31] T. Bisták, P. Švec, J. Kľuka, A. Šimko, Š. Balogh, M. Homola, Improving DL-Learner on a malware detection use case., in: Description Logics, 2023.

[32] P. Švec, Ontologická reprezentácia pre bezpečnosť informačných systémov [Ontological representation for security of information systems], Phd thesis, Slovak University of Technology in Bratislava, Faculty of Electrical Engineering and Informatics, Bratislava, Slovakia, 2024. URL: https://uim.fei.stuba.sk/wp-content/uploads/2024/10/2024.svec_.dizp_.pdf.

[33] F. A. Cardillo, U. Straccia, Fuzzy owl-boost: Learning fuzzy concept inclusions via real-valued boosting, Fuzzy Sets Syst. 438 (2022) 164–186.

[34] F. A. Cardillo, F. Debole, U. Straccia, PN-OWL: A two-stage algorithm to learn fuzzy concept inclusions from OWL 2 ontologies, Fuzzy Sets Syst. 490 (2024) 109048.

[35] I. Huitzil, U. Straccia, N. Díaz-Rodríguez, F. Bobillo, Datil: Learning fuzzy ontology datatypes, in: Proceedings of the 17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2018), Part II, volume 854 of *Communications in Computer and Information Science*, Springer, 2018, pp. 100–112.

[36] F. A. Lisi, U. Straccia, A FOIL-like method for learning under incompleteness and vagueness, in: 23rd International Conference on Inductive Logic Programming, volume 8812 of *LNAI*, Springer Verlag, Berlin, 2014, pp. 123–139.

[37] F. A. Lisi, U. Straccia, Learning in description logics with fuzzy concrete domains, Fundamenta Informaticae 140 (2015) 373–391.

[38] U. Straccia, M. Mucci, pFOIL-DL: Learning (fuzzy) $\mathcal{EL}$ concept descriptions from crisp owl data using a probabilistic ensemble estimation, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC-15), ACM, Salamanca, Spain, 2015, pp. 345–352.

[39] D. Trizna, P. Anthony, M. Homola, Z. Adams, Š. Balogh, Learning explainable malware characterization using knowledge base embedding, in: 2024 IEEE 5th International Conference on Electro-Computing Technologies for Humanity (NIGERCON), IEEE, 2024, pp. 1–8.

[40] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, RotatE: Knowledge graph embedding by relational rotation in complex space, 2019. arXiv:1902.10197.

[41] P. Anthony, F. Giannini, M. Diligenti, M. Homola, M. Gori, Štefan Balogh, J. Mojžiš, Explainable malware detection with tailored logic explained networks, 2024. arXiv:2405.03009.

[42] Y. Oyama, T. Miyashita, H. Kokubo, Identifying useful features for malware detection in the ember dataset, in: 2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW), 2019, pp. 360–366.

[43] B. ten Cate, M. Funk, J. C. Jung, C. Lutz, SAT-based PAC learning of description logic concepts, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China, ijcai.org, 2023, pp. 3347–3355.

[44] M. Funk, J. C. Jung, T. Voellmer, SAT-based bounded fitting for the description logic $\mathcal{ALC}$ (extended abstract), in: 38th International Workshop On Description Logics, Opole, Poland, 2025.

[45] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Networks 20 (2009) 61–80.

[46] H. Yuan, H. Yu, S. Gui, S. Ji, Explainability in graph neural networks: A taxonomic survey, IEEE Trans. Pattern Anal. Mach. Intell. 45 (2023) 5782–5799.

[47] H. Yuan, J. Tang, X. Hu, S. Ji, XGNN: towards model-level explanations of graph neural networks, in: R. Gupta, Y. Liu, J. Tang, B. A. Prakash (Eds.), KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, ACM, 2020, pp. 430–438.

[48] D. J. T. Cucala, B. C. Grau, E. V. Kostylev, B. Motik, Explainable gnn-based models over knowledge graphs, in: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022, OpenReview.net, 2022.

[49] M. de Sousa Ribeiro, J. Leite, Aligning artificial neural networks and ontologies towards explainable AI, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 4932–4940.

[50] J. Ferreira, M. de Sousa Ribeiro, R. Gonçalves, J. Leite, Looking inside the black-box: Logic-based explanations for neural networks, in: G. Kern-Isberner, G. Lakemeyer, T. Meyer (Eds.), Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel, July 31 - August 5, 2022, 2022.

[51] S. Wachter, B. Mittelstadt, C. Russell, Counterfactual explanations without opening the black box: Automated decisions and the gdpr, Harv. JL & Tech. 31 (2017) 841.

[52] I. Stepin, J. M. Alonso, A. Catalá, M. Pereira-Fariña, A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence, IEEE Access 9 (2021) 11974–12001.

[53] P. Lipton, Contrastive explanation, Royal Institute of Philosophy Supplements 27 (1990) 247–266.

[54] K. Cyras, A. Rago, E. Albini, P. Baroni, F. Toni, Argumentative XAI: A survey, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 4392–4399.

[55] O. Or-Meir, N. Nissim, Y. Elovici, L. Rokach, Dynamic malware analysis in the modern era – A state of the art survey, ACM Comput. Surv. 52 (2019) 88:1–88:48.