An Empirical Study on the Robustness of Knowledge Injection Techniques Against Data Degradation

Andrea Rafanelli[1,2,†], Matteo Magnini[3,*,†], Andrea Agiollo[3,†], Giovanni Ciatto[3] and Andrea Omicini[3]

[1]Department of Computer Science, University of Pisa, Pisa, Italy
[2]Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, L'Aquila, Italy
[3]Department of Computer Science and Engineering (DISI), University of Bologna, Bologna, Italy

Abstract

Symbolic knowledge injection (SKI) represents a promising paradigm for bridging symbolic knowledge and sub-symbolic predictors in intelligent autonomous agents. Given the wide availability of SKI methods from the literature, we observe that SKI effectiveness is commonly measured in terms of predictive performance variation – e.g., accuracy improvement – introduced by SKI. However, other aspects such as the injection mechanism's ability to maintain its performance and generalisation capability, despite encountering unexpected or anomalous input during the training process, are equally relevant. Accordingly, in this paper we propose a new metric to evaluate the robustness of SKI techniques, defined as a measure of performance degradation in response to systematic dataset variations. The proposed metric enables precise quantification of the robustness degree across injection approaches and different perturbations. Details on generating and quantifying perturbations are also provided. We evaluate the effectiveness of our metric through several experiments, where we apply multiple SKI techniques to three datasets and measure how robustness varies as perturbations increase.

Keywords

Symbolic Knowledge Injection, Robustness, Neural Networks

1. Introduction

In recent years, the success of sub-symbolic approaches such as neural networks (NNs) enabled computational agents with more and more smart behaviours ranging from speech recognition [1] to object detection [2], and many more [3]. However, while reaching considerable success in tasks requiring perception, sub-symbolic mechanisms still face challenges in tasks requiring reasoning and / or interpretability of decision processes and results [4]. Moreover, the autonomous agents' knowledge and behaviours are commonly represented *symbolically* both at the conceptual and technological level, hindering the integration of *sub*-symbolic components inside autonomous agents capable of complex reasoning. The integration of symbolic and sub-symbolic systems (a.k.a. *neuro-symbolic*) has emerged as a possible solution to the aforementioned problems gaining significant attention in the field of autonomous agents [5] and general artificial intelligence (AI) systems [6]. In this context, most recent efforts focus on the integration of NN with symbolic AI techniques aiming to augment NNs with some reasoning capability—or making them more interpretable, knowledgeable, predictable, or any combination of those. Among the many methods for mixing NN with symbolic AI [7], in this paper we focus on those involving the *injection* of symbolic knowledge into NN [8].

Symbolic knowledge injection (SKI) [8] techniques entail the incorporation of structured and explicit knowledge – available at the intelligent agent level – into sub-symbolic systems, such as NN, to improve their reasoning and decision-making capabilities. Symbolic knowledge can take the form of logic

WOA 2024: 25th Workshop "From Objects to Agents", 7th-10th July, 2024, Forte di Bard (Italy)
*Corresponding author.
†These authors contributed equally.
andrea.rafanelli@phd.unipi.it (A. Rafanelli); matteo.magnini@unibo.it (M. Magnini); andrea.agiollo@unibo.it (A. Agiollo); giovanni.ciatto@unibo.it (G. Ciatto); andrea.omicini@unibo.it (A. Omicini)
0000-0001-8626-2121 (A. Rafanelli); 0000-0001-9990-420X (M. Magnini); 0000-0003-0531-1978 (A. Agiollo); 0000-0002-1841-8996 (G. Ciatto); 0000-0002-6655-3869 (A. Omicini)

rules, as well as knowledge graphs, or expert knowledge in a specific domain. The available symbolic knowledge can represent the agent's beliefs, some well-known concepts, a societal norm or any desirable rule for the sub-symbolic system to be considered. In any case, the goal of SKI is altering the sub-symbolic predictor to make it compliant w.r.t. some given symbolic knowledge. Commonly, this is attained by extending either the structure or the training process of a sub-symbolic predictor. Letting a NN exploit prior knowledge may lead to several benefits, e.g., making the NN *(i)* more robust to data quality degradation (e.g., noisy or corrupted inputs, lack of data, etc.), *(ii)* more predictable – hence, more trustworthy –, *(iii)* less data-requiring, and, sometimes, *(iv)* less time-requiring to train.

SKI methods are often measured by predictive performance variation, such as accuracy improvements, but other aspects like *efficiency* [5], or the *robustness* of the injection mechanism also play a role. Robust mechanisms can effectively incorporate new knowledge into NN without causing errors or deteriorating their overall performance. In safety-critical applications, like autonomous vehicles or medical diagnosis systems, a non-robust mechanism may introduce errors, leading to poor performances. A robust mechanism can seamlessly adapt to new knowledge, ensuring the quality of performance and enhancing the trustworthiness of NN systems.

Robustness can be assessed by evaluating the injected model's ability to maintain its performance in the presence of data perturbations, such as noisy or corrupted training samples. Therefore, measuring the robustness of injection mechanisms is a crucial stage in developing neuro-symbolic agents, and it should be included in the process of evaluating any SKI approach. However, available metrics focus specifically on purely sub-symbolic mechanisms, such as NN [9], while trustworthiness measurements in the SKI realm are currently lacking [10]. Moreover, such metrics focus heavily on robustness under the adversarial attack paradigm, while ignoring the robustness of the learning algorithm itself when faced with bugged training data. Therefore, in this paper, we focus specifically on the SKI realm and consider robustness as the resilience of its training approach over imperfect, bugged or missing data. Accordingly, we present a comprehensive modelling of a new robustness metric for SKI, as well as an empirical evaluation of this metric through PSyKI [11]—a software library for SKI algorithms. Our findings provide compelling evidence that the introduction of a robustness metric for SKI is a crucial step towards enhancing the reliability and transparency of AI systems. To the best of our knowledge, this study represents the first attempt to introduce a robustness metric for SKI, along with the necessary software tools for its practical implementation and evaluation.

2. Background

Robustness Robustness is a crucial challenge for NN [12], and relates to their capability to maintain performance despite the presence of input perturbations. In general, the more a neural network is robust the more reliable and accurate its predictions are likely to be.

To increase the neural network robustness, a number of techniques have been proposed in the literature, such as regularisation techniques [13], data augmentation techniques [14], and also methods such as adversarial training [15]. The strong interest behind NN robustness is motivated by both *(i)* the need to address certain limitations of NN – such as be sensitive to random perturbations [16, 17] –, and *(ii)* the development of trustworthy AI (TAI).

Trustworthy AI TAI refers to the development and deployment of AI systems that are transparent, responsible, ethical, and safe. It gained momentum in recent years, also because of the growing popularity of AI, and the consequent attention of policymaker—cf. the guidelines set by the European Union [18].

Robustness is a key component of TAI, as it ensures that the AI system can function consistently and accurately in a variety of scenarios. Along this line, research about NN trustworthiness has greatly advanced, with more and more new possible solutions being sought. In this regard, neuro-symbolic contributions were proposed as means to achieve more trustworthy models. In practice, these methods aim at combining the strengths of NN and symbolic AI, mitigating the weaknesses of both—e.g., model

opacity for NN, or limited scalability for symbolic systems.

Neuro-symbolic integration Neuro-symbolic methods [19] can increase the robustness of NN, for instance by providing them with symbolic representations of knowledge. Symbolic knowledge is characterised by the use of facts and rules – commonly, in logic form –, which can facilitate the representation of knowledge in a structured – and, possibly, *intensional* – manner. This can be particularly useful for applications where training data may be noisy or incomplete. In fact, the lack of quality data may be compensated by concise – yet general – logic rules describing the phenomena for which data is missing. Therefore, by incorporating symbolic knowledge into NN, the general expectation is that they should improve their performance, even in presence of issues in the training data.

Symbolic Knowledge Injection This paper focuses on symbolic knowledge injection (SKI) as an approach for neuro-symbolic integration. SKI techniques, in general, are arbitrary algorithmic procedures that affect how sub-symbolic predictors draw their inferences, either so that predictions are computed as a function of or made consistent with some given symbolic knowledge. To enhance reasoning and decision-making capabilities in sub-symbolic systems, SKI requires incorporating structured knowledge, such as logical rules, knowledge graphs, or expert knowledge [8].

There are several methods for knowledge injection [8], including *constraining-based methods*, *structuring-based methods*, and *embedding* symbolic knowledge within NN—the latter laying outside the scope of this paper. In all such cases, SKI is performed by modifying the sub-symbolic predictor to conform with input knowledge, and then trained and exploited "as usual". Accordingly, in the remainder of this paper, we denote any injection procedure as $\hat{N} = \mathcal{I}(N, K, D)$, where N denotes the predictor *before* injection, K is the symbolic knowledge to be injected, D is the training dataset, $\mathcal{I}$ is the injection procedure, and $\hat{N}$ is the predictor with injected knowledge. Informally, we may refer to $\hat{N}$ as the *educated* predictor, and N as the *uneducated* one.

Performance scores Effectiveness of SKI is commonly measured by comparing the performance gain (e.g. accuracy, F1-score, MSE, etc.) achieved by the SKI predictor w.r.t. its uneducated counterpart. This metric indicates the quality of the SKI approach in predictive performance but does not capture aspects related to injection mechanism quality, such as robustness. Therefore, in the remainder of this paper, we identify a metric to measure the actual robustness of the SKI mechanism compared to its uneducated counterpart.

3. Robustness Score for SKI

In counterfactual analysis [20], it is a common practice to exploit controlled perturbation on the training data to assess the robustness of predictors. Taking inspiration from this technique, in this section we introduce the notion of *statistical robustness* of SKI procedure. Accordingly, we consider an injection mechanism as *robust* if the predictive performance of the educated predictors is poorly affected by *perturbations* of the training data—as long as the perturbation *magnitude* is small. The remainder of this section provides a general description of robustness, before delving into the details of which perturbations types can be applied to training data and how their magnitude may be measured.

Data perturbation We define *data perturbation* as altering a training dataset D by adding, removing, or editing its entries, and denote it by ΔD. Accordingly, we denote the perturbed dataset as $D' = D \circ \Delta D$, where $\circ$ is the perturbation *application* operator. We also denote by $\|\Delta D\| \in \mathbb{R}_{\geq 0}$ the *magnitude of the perturbation*—i.e., the scalar value quantifying the amount of changes induced by the perturbation.

Robustness score Let $\mathbf{D} = \{\Delta D_1, \ldots, \Delta D_n\}$ be a set of potential data perturbations to be applied to some dataset D, let N be a predictor of any sort – trained on D –, and let $N_{\Delta D}$ be the predictor having the same hyperparameters of N, yet being trained upon the perturbed dataset $D \circ \Delta D$. Under such hypotheses, we define the *robustness score* of N w.r.t. $\mathbf{D}$, as follows:

$$\rho_{N,D}(\mathbf{D}) = \frac{1}{n} \sum_{\Delta D \in \mathbf{D}} \|\Delta D\| \cdot \frac{\pi(N_{\Delta D}, D \circ \Delta D)}{\pi(N, D)} \tag{1}$$

where π is a performance metric of choice, such as accuracy, computing the performance of the input predictor w.r.t. the input dataset.

As the reader may notice, robustness is *directly* proportional to *(i)* the magnitude of the perturbations, and *(ii)* the ratio among the performance of the perturbed predictors and the performance of the unperturbed predictor. We would like to emphasise that, although it is true that every component in Equation (1) has a positive value, this does not imply that every data perturbation leads to a positive impact on performance. Indeed, our robustness measure incorporates a performance ratio, where both the numerator and denominator are bounded between 0 and 1 (as the performance metric itself is considered to range from 0 to 1). As a result, the performance ratio can be slightly above 1 when the perturbed model performs better – i.e., positive perturbation where $\rho_{\hat{N},D}(\mathbf{D}) > \rho_{N,D}(\mathbf{D})$ –, while it falls below 1 when the perturbed model performs worse—i.e. negative perturbation where $\rho_{\hat{N},D}(\mathbf{D}) < \rho_{N,D}(\mathbf{D})$. Practically speaking, when positive perturbations occur, our metric produces a ratio approaching 1. This indicates merely a marginal gain, as ratios higher than 1 denote performance improvements over the unperturbed model. Ratios below 1 point to decrements in performance. In other words, the robustness of a predictor increases when relatively big perturbations in the training data have a relatively small effect on the overall predictor's performance, i.e. the more robust the model, the less its performance is disrupted by significant modifications to its training inputs.

The robustness of some injection mechanism can be measured by applying Equation (1) to some educated predictor $\hat{N} = \mathcal{I}(N, K, D)$, attained by injecting the knowledge K, on some uneducated predictor N, then trained upon D—which we denote denote by $\rho_{\hat{N},D}(\mathbf{D})$. One may also be interested in understanding whether some injection mechanism makes the predictor more or less robust than its uneducated counterpart. To this end, we define the *robustness gain* score as follows: $R_{N,D}(\mathcal{I}) = \rho_{\hat{N},D}(\mathbf{D})/\rho_{N,D}(\mathbf{D})$. Here, the robustness gain is a positive measure that indicates if the injection mechanism $\mathcal{I}$ produces a more robust predictor ($R_{N,D}(\mathcal{I}) > 1$) w.r.t. its uneducated counterpart over data perturbation. Meanwhile, injection mechanisms suffering data perturbations result in $R_{N,D}(\mathcal{I}) < 1$, as they produce an educated model $\hat{N}$ less effective for dealing with perturbed data. Therefore, $R_{N,D}(\mathcal{I})$ is an easy-to-understand measure for analysing the robustness quality of a selected SKI mechanism.

3.1. Measuring Data Perturbations

Equation (1) defines robustness by relying on the possibility of measuring the magnitude $\|\Delta D\|$ of any given perturbation ΔD. While being easy to model in theory, the problem is hard to tackle in practice. As far as this paper is concerned, we focus on the simpler goal of measuring the *difference* among any two datasets A, B. To serve this purpose, we leverage on the Kullback-Leibler (KL) divergence [21]. In theory, the KL-divergence is a statistical operator aimed at measuring the difference among any two probability distributions α and β. In the particular case where *(i)* α and β are multivariate normal distributions having the same dimensionality, and *(ii)* two datasets A, B are sampled from α and β respectively, the KL-divergence can be computed as follows:

$$\psi(A, B) = \frac{1}{2}\Big[\mathrm{tr}(\boldsymbol{\sigma}_B^{-1}\boldsymbol{\sigma}_A) - \dim(A) \; + \ln\Big(\frac{\det \boldsymbol{\sigma}_B}{\det \boldsymbol{\sigma}_A}\Big) + (\boldsymbol{\mu}_B - \boldsymbol{\mu}_A)^\top \boldsymbol{\sigma}_B^{-1}(\boldsymbol{\mu}_B - \boldsymbol{\mu}_A)\Big] \tag{2}$$

where $\boldsymbol{\mu}_A$ (resp. $\boldsymbol{\mu}_B$) represents the mean of A (resp. B), and $\boldsymbol{\sigma}_A$ (resp. $\boldsymbol{\sigma}_B$) represents its covariance matrix, $\det \boldsymbol{\sigma}_A$ (resp. $\det \boldsymbol{\sigma}_B$) is its determinant, and $\mathrm{tr}(\boldsymbol{\sigma}_A)$ (resp. $\mathrm{tr}(\boldsymbol{\sigma}_B)$) its trace. Finally, $\dim(A)$ represents the dimensionality of A—i.e., the number of features characterising each of its entries.

The normality hypothesis may appear restrictive, as datasets A and B may, in the general case, be sampled from unknown distributions. However, any unknown probability distribution can be approximated by a *mixture* of normal distributions – as the latter are universal approximators for probability densities [22, Sec. 3.9.6] –, and the computation of the KL-divergence for a mixture of normal distributions can be approximated to that of a single normal distribution—as discussed in Joyce [21, Sec. 2.2.2]. So, in practice, Equation (2) can be exploited to estimate the difference among any two datasets A and B.

KL-divergence for classification problems. For classification problems, the KL divergence is commonly computed per-class, as different classes may come with different distributions. In case K classes are available, we consider datasets as partitioned into disjoint sub-sets, on a per-class basis: i.e., $A = A_1 \cup \ldots \cup A_K$ and $B = B_1 \cup \ldots \cup B_K$, where A_k (resp. B_k) is the set of samples belonging to class k in A (resp. B).

We here propose to define the *overall* divergence score between A and B as the weighted average of class-specific Kullback-Leibler divergences, namely: $\Psi(A, B) = \frac{1}{|A|} \sum_{k=1}^{K} \psi(A_k, B_k) \cdot |A_k|$. where $|A|$ (resp. $|A_i|$) represents the cardinality of A (resp. A_i). The weighted average is necessary to tackle very unbalanced datasets with diverging distributions over unpopular labels. Therefore, the proposed approach allows to quantify the distribution changes for each label, while also focusing on the points where the actual change occurred, handling cases where the perturbation may significantly affect only some classes.

Accordingly, we estimate the magnitude of a perturbation ΔD transforming D into $D' = D \circ \Delta D$ by means of the overall divergence among D and D': $\|\Delta D\| = \Psi(D, D')$. The proposed metric lays in $\mathbb{R}_{\geq 0}$: it is equal to zero for two identical datasets – i.e., when the perturbation ΔD has no effect –, and it gets higher in value as D' differs from D—i.e., proportionally to the effect of $\|\Delta D\|$.

It is worth highlighting that the proposed perturbation metric $\|\cdot\|$ can be applied to any sort of perturbation, as it does not take into account *how* the perturbation ΔD affects the dataset or its samples. For this reason, the metric can be exploited to measure the many sorts of perturbations discussed in Section 3.2.

3.2. Perturbation Strategies

Here we discuss three major perturbation strategies for altering a dataset, namely: *(i)* sample drop, *(ii)* noise addition, and *(iii)* label flipping. These strategies mimic common issues that can degrade a dataset's quality, as they involve randomly deleting samples, adding random noise, and randomly changing labels.

Sample Drop This perturbation strategy mimics the effect of an intelligent agent lacking training data. Hence, it aims at selectively mutilating a dataset in a *controlled* way. This strategy is commonly exploited to test if and to what extent neuro-symbolic approaches are effective to deal with data scarcity [23].

The basic idea behind sample drop is to randomly remove some samples from D to obtain D'. The whole process is stochastic and controlled by a single parameter, namely the *dropping probability* – denoted by $p \in]0, 1[$ –, which is shared among all data entries in D. Generally speaking, $p = \mathbb{E}[X_d]$ represents the mean of the Bernoulli-distributed random variable $X_d \sim \mathcal{B}(p)$, dictating whether the data entry $d \in D$ should ($\mathbb{P}(X_d = 0)$) or should not ($\mathbb{P}(X_d = 1)$) be included in D'. Under such hypotheses, the perturbed dataset D' is such that $D' = \{d \mid \forall d \in D \text{ s.t. } X_d = 0\}$.

We consider constructing the set of data perturbations $\mathbf{D} = \{\Delta D_1, \ldots, \Delta D_n\}$ by applying the sample drop process on D multiple times – namely, n times – with increasing dropping probabilities $0 < p_1 < \ldots < p_n < 1$.

Noise Addition This perturbation strategy mimics the situation where the data sampling / acquisition process of the autonomous agent is affected by error—e.g., due to sensor noise. Hence, it aims at

degrading a dataset in a *controlled* way. This strategy is commonly exploited to test if and to what extent neuro-symbolic approaches are effective to deal with unreliable (e.g., corrupted, inconsistent) data [24].

The basic idea behind noise addition is to add some random noise to the data entries in D, to obtain D'. The whole process is stochastic, and it is controlled by a single parameter, namely the *noise intensity* – denoted by $v \in \mathbb{R}_{\geq 0}$ –, which is shared among all data entries in D. Generally speaking[1], $v \cdot \mathbf{1} = \text{cov}(V_d)$ represents the covariance matrix of the 0-mean, multi-variate, normally distributed random variable $V_d \sim \mathcal{N}(\mathbf{0}, v \cdot \mathbf{1})$, representing the random noise to be applied to each entry $d \in D$. Under such hypotheses, the perturbed dataset D' can be described as $D' = \{d + V_d \mid \forall d \in D\}$. As the reader may notice, the perturbed dataset is characterised by the same number of samples of D—meaning that $|D| = |D'|$.

We consider constructing the set of data perturbations $\mathbf{D} = \{\Delta D_1, \ldots, \Delta D_n\}$ by applying the noise addition process on D multiple times – namely, n times – with increasing noise intensities $0 < v_1 < \ldots < v_n$.

About discrete features. Normal noise can be applied to any *continuous* feature in a dataset. However, when dealing with datasets with ordinal or categorical features, it is necessary to adjust the noise addition process to account for discontinuity. Accordingly, to deal with discrete features, the additive normal noise is discretised by replacing the distribution $\mathcal{N}(\mathbf{0}, v \cdot \mathbf{1})$ with its discrete counterpart $\bar{\mathcal{N}}$ [25]. Overall, this means that for *ordinal* features, additive noise is more likely to choose an additional ordinal value close to the original one, whereas for *categorical* features, additive noise may choose an additional ordinal value with uniform probability—analogously to the label flipping case described below.

Label Flipping This perturbation strategy mimics the situation where some data labelling process is affected by error—e.g., human operator mislabelling some data entries. Hence, it aims at selectively flipping the labels of some entries in a dataset. The underlying assumption is that the dataset consists of input and output features, and the output ones are either binary or categorical. Therefore, this perturbation strategy mostly makes sense for classification tasks.

The basic idea behind label flipping is to randomly alter the output features of some data entries in D, to produce D'. Flipping, in particular, requires each output feature to be randomly re-assigned with some value in the output domain—of course different from the original one.

Without loss of generality, we consider the case of a single output feature having K admissible values—i.e., K classes represented as a single categorical attribute. Hence, for an m-dimensional dataset D, we let $d \equiv (x_{d,1}, \ldots, x_{d,m-1}, y_d) \in D$ denote the generic data entry in D, where the many $x_{d,j}$ are the values of the input features of d, and $y_d \in \{1, \ldots, K\}$ is the value of its output feature.

Similarly to the cases above, we model label flipping as a stochastic process controlled by the *flipping probability* parameter – denoted by $f \in]0,\ 1[$ –, representing the likelihood that the label of any given data entry may flip. Parameter f controls the probability distribution of the random variable Y_d, which represents the novel value of the output feature for data entry $d \in D$, after flipping. In particular, we define the probability density of Y_d as follows: $\mathbb{P}(Y_d = k) = f/(K-1)$ if $k \not\equiv y_d$; otherwise $\mathbb{P}(Y_d = y_d) = (1 - f)$. In other words, the value of the output feature may flip with probability f, and in that case each value different than the original one is equally likely to be selected. Under such hypotheses, the perturbed dataset D' is such that $D' = \{(x_{d,1}, \ldots, x_{d,m-1}, y'_d) \mid \forall d \in D : Y_d = y'_d\}$.

We consider constructing the set of data perturbations $\mathbf{D} = \{\Delta D_1, \ldots, \Delta D_n\}$ by applying the label flipping process on D multiple (i.e., n) times, with increasing flipping probability $0 < f_1 < \ldots < f_n < 1$.

About regression problems. Label flipping is not directly applicable to regression problems, as the output feature is not categorical. However, it may be modelled as a particular sort of noise addition

[1] $\mathbf{1}$ is the $m \times m$ identity matrix, $\mathbf{0}$ is the $m \times 1$ zero vector, where m is dimensionality of the dataset.

affecting only output feature(s). Nevertheless, the application of label flipping to regression problems is beyond the scope of this paper.

4. Experiments

In this section we present the setups and results of the experiments for evaluating the effectiveness of the proposed robustness metric. For sake of reproducibility, the source code of our experiments is public and available.[2]

Datasets We rely on three different datasets from the UCI repository,[3] namely the "Breast Cancer Wisconsin" (BCW) dataset,[4] the "Primate Splice Junction Gene Sequences" (PSJGS) dataset,[5] and "Census Income" (CI) dataset.[6] Subsequently, we present in detail each of the datasets.

Breast Cancer The BCW dataset contains 699 instances of breast cancer clinical exams, each with 9 features summarising biological characteristics. The feature $BareNuclei$ has 16 missing values, which are replaced with the value zero. The dataset is divided into benign and malignant classes, with 458 (65.5%) benign samples and 241 (34.5%) malignant samples.

Splice Junction The PSJGS dataset contains 3,190 gene sequence instances, each described by 60 nucleotides. Values are adenine (`a`), cytosine (`c`), guanine (`g`), and thymine (`t`). However, other symbols may be used to indicate that more than one specific nucleotide can appear in the same position. For this reason, we perform a multi-hot encoding of the features, resulting in 240 boolean features. The dataset has three classes, exon-intron (`ei`), intron-exon (`ie`), and none of the previous (`n`), with 767 (24%), 768 (24.1%), and 1655 (51.9%) instances respectively.

Census Income The CI dataset contains 48,842 instances representing individuals, with 14 features representing different attributes. We opt to remove $Education$ as $EducationNumeric$ is an ordered representation of the same feature. $Fnlwgt$ and $Race$ features are avoided to avoid social bias. The remaining nominal features ($WorkingClass$, $MaritalStatus$, $Occupation$, $Relationship$, $NativeCountry$) are one-hot encoded. Classes are binary and represent the yearly income of a person, less/equal than/to $50,000$ USD and more than $50,000$ USD, with $37,155$ (76.1%) and $11,687$ (23.9%) instances respectively.

Injected Knowledge For our experiments we exploit specific knowledge for each dataset, using Prolog syntax [26] for classification logic rules The PSJGS dataset already has a knowledge base in the literature, with biological rules provided by human domain experts [27]. We use this very knowledge as well, with the addition of an explicit third rule to classify the `n` class: $class(\bar{X}, \texttt{n}) \leftarrow \neg class(\bar{X}, \texttt{ei}) \wedge \neg class(\bar{X}, \texttt{ie})$. where $\bar{X}$ is an abbreviation for the full sequence of variables $X_{-30}, \ldots, X_{+30}$ that represent the input data of a 60-basis long DNA sequence. The rule simply states that if a record is not classified as `ei` or `ie` then it is classified as `n`. For the remaining datasets – BCW and CI – there are no predefined rules in the literature. Therefore, we define a custom set of logic rules for both of them.

[2] https://anonymous.4open.science/r/ski-data-robustness-experiments
[3] https://archive.ics.uci.edu/ml/index.php
[4] https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original
[5] https://archive.ics.uci.edu/dataset/69/molecular+biology+splice+junction+gene+sequences
[6] https://archive.ics.uci.edu/dataset/20/census+income

Injectors We rely on PSyKI[7] (Platform for Symbolic Knowledge Injection) for the SKI algorithms in the experiments. PSyKI enables sub-symbolic predictors incorporate prior symbolic knowledge. It is compatible with Tensorflow-deployed[8] predictors and allows for various SKI algorithms to be used:

- **KINS**, Knowledge Injection via Network Structuring [28] is a structuring-based injection mechanism where an ordinary multi-layer neural network is extended with ad-hoc neural modules aimed at mimicking the provided symbolic knowledge. The weights of the neural modules are trained together with the weights of the original neural network.
- **KILL**, Knowledge Injection via Lambda Layer [29] is a guided-learning-based approach that constrains the training phase as follows. Before each back-propagation step, a penalty value that represents the degree of violation w.r.t. the input knowledge is added to the loss function. Therefore, the training phase is forced to minimise the loss function and the penalty value at the same time—hence maximising the compliance w.r.t. the knowledge to be injected.
- **KBANN**, Knowledge-Based Artificial Neural Network [27] is one of the first structuring-based SKI algorithms proposed in the literature. It creates a NN from a set of propositional rules: each piece of each formula is converted into partial neural structures, to be then composed in a single network. The main difference w.r.t. to KINS is that the network created by KBANN is *entirely* constructed from the knowledge, while KINS extends an existing network with ad-hoc modules.

4.1. Experimental Setup

For our experiments, we leverage three datasets, three input knowledge bases, three different injectors, and three perturbation strategies. For each dataset we train an uneducated model, which is then applied to the three injectors using the corresponding knowledge base. The uneducated model has two hidden layers, one input, and one output layer, with rectified linear units (ReLU) as the activation function, and softmax for the output layer. The number of neurons per hidden layer changes depending on the dataset—namely, $[16, 8]$ for BCW, $[32, 16]$ for CI, and $[64, 32]$ for PSJGS. Training involves *categorical cross-entropy* loss function, a batch size of 32 elements, and an epoch limit of 100.

Data perturbation is applied multiple times for each dataset-model configuration, using sample drop, noise addition, and label flipping. More precisely, the drop probability (d) parameter varies from 0.0 to 0.95 with a step of 0.05, the flipping probability (f) parameter varies from 0.0 to 0.90 with a step of 0.08, and the noise intensity (v) parameter varies from 0.0 to 1.0 with a step of 0.1. Each experiment is repeated 30 times to draw statistically relevant comparisons.

We point out that our robustness analysis exclusively focuses on symbolic knowledge injection (SKI) approaches. In this context, we did not find any existing previous measures allowing a direct comparison. The only related available metric [30] focuses on generalisation capability rather than training robustness. Therefore, we here consider to solely analyse the goodness of the proposed robustness metric, aiming at checking if it allows to identify robust SKI mechanisms.

4.2. Results and Discussion

Here we present the results of our experiments (Figure 1) over different perturbation strategies.

To better understand the real differences in performance between educated and uneducated predictors—which are fairly comparable—we compare the average accuracy of each predictor w.r.t. the uneducated one. For this purpose, we employ the *Mann-Whitney U Test* [31], a non-parametric test for differences between two groups. In this setting, a *p-value* ≥ 0.05 indicates that there are no significant differences between the two average accuracy distributions, whereas a *p-value* < 0.05 indicates the opposite.

[7] https://github.com/psykei/psyki-python
[8] https://www.tensorflow.org/?hl=it

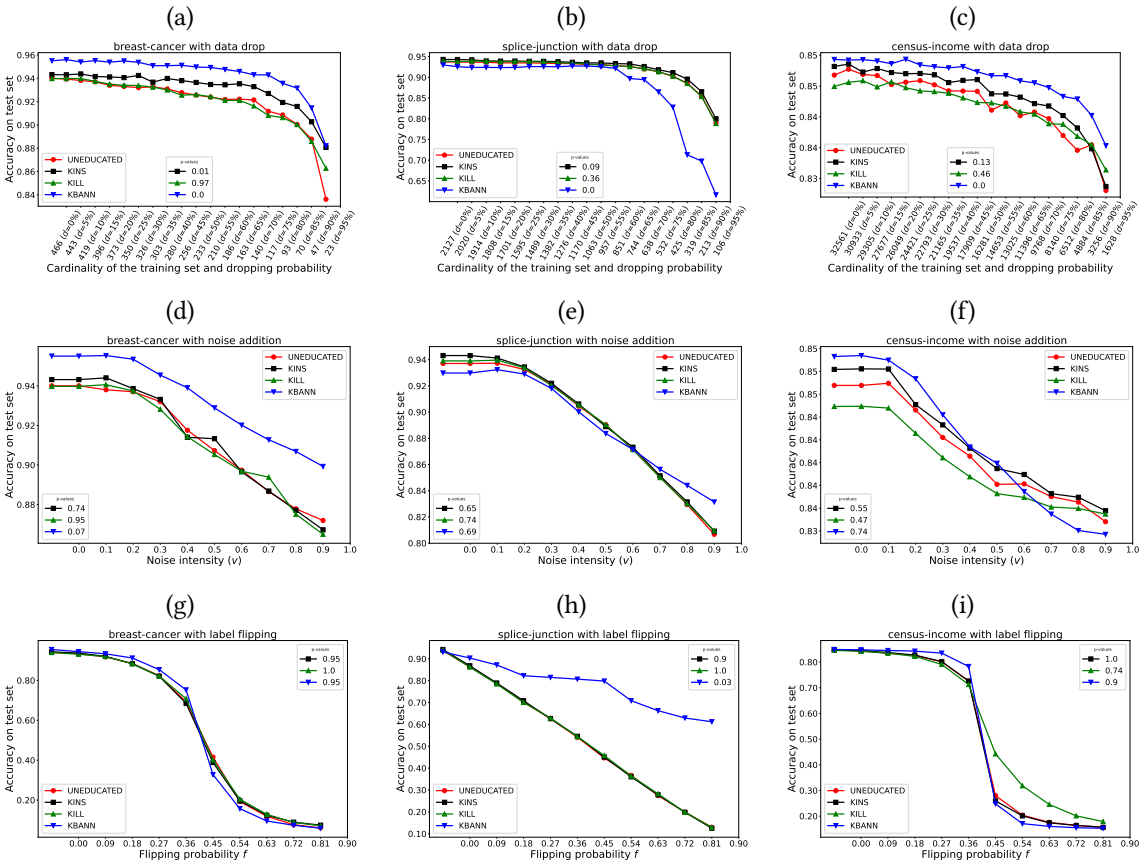

Figure 1: Average accuracy over different datasets with different perturbation strategies

Table 1
Robustness relative scores. Bold numbers are the ones greater than 1 (i.e., the educated model is more robust than the uneducated one).

Dataset	$R_{N,D}(\mathcal{I})$ drop			$R_{N,D}(\mathcal{I})$ noise			$R_{N,D}(\mathcal{I})$ flip		
	KINS	KILL	KBANN	KINS	KILL	KBANN	KINS	KILL	KBANN
BCW	**1.0493**	**1.0318**	**1.0382**	0.9960	0.9985	**1.0109**	0.9994	**1.0184**	0.9520
PSJGS	**1.0045**	0.9968	0.8425	0.9950	0.9984	**1.0145**	0.9962	**1.0026**	**1.6749**
CI	0.9998	**1.0039**	**1.0043**	0.9992	**1.0012**	0.9965	0.9897	**1.1703**	0.9815

Drop In the data drop experiments (Figure 1 a-c), KBANN is the only predictor that shows significant differences w.r.t. the uneducated one. Specifically, KBANN demonstrates improved performance on the BCW (Figure 1a) and CI (Figure 1c) datasets, but its performance on the PSJGS (Figure 1b) dataset rapidly declined when 60% of the data are dropped ($d = 0.6$). For the other educated predictors, KINS shows slightly better performances than the uneducated model. In the BCW dataset its *p-value* $= 0.01$ indicates that there are significant differences in terms of performance compared to the uneducated model.

Looking at Table 1, one can see the influence of the performance values on the robustness score, e.g. for KBANN in the PSJGS dataset. Apart from this, the educated models improve robustness in 6 out of 9 experiments.

Noise The noise experiments (Figure 1 d-f) reveal that SKI mechanisms are more sensitive to noisy data than missing data. In fact, the average accuracy of predictors trained over noisy data drops more quickly than the data drop scenario. This is due to the downward trends exhibited by all predictors

since the early stages of the curves. Indeed, Table 1 confirms that educated models enhance robustness in only 3 experiments out of 9.

KBANN outperforms other SKI methods and the uneducated model in the BCW (Figure 1d), but performs poorly in the CI (Figure 1f) dataset. In PSJGS (Figure 1e) dataset, it has low performances too w.r.t. the uneducated predictor —except for high noise levels, where it is subject to relatively slower performance decline. This explains why in Table 1 KBANN's robustness score shows a gain over the uneducated one. The opposite trend can be observed for the CI scenario.

Along this line, it is worth of notice observing that KINS – despite being a SKI approach based on neural structuring like KBANN – performs noticeably lower than KBANN in terms of robustness, regardless of the dataset. The discrepancy may be caused by the trainable nature of KINS' neural modules, which makes them more prone to overfitting noisy data. It is noticeable that KILL consistently exhibits similar (or worse) behaviour across all datasets than the uneducated model. This may imply that the penalties imposed by KILL during training for performing injection are not compensating for noise perturbations.

Label-flipping The label flipping experiments (Figure 1 g-i) show that predictors behave similarly in both BCW (Figure 1g) and CI (Figure 1i) datasets. This similarity between all predictors can also be observed when looking at *p-values* which are all close to 1—i.e. no significant difference between educated and uneducated predictors. In both, BCW and CI, there is a quick degradation in performance starting from 54% of flipped labels ($f = 0.54$). Conversely, for the PSJGS dataset (Figure 1h), the performance decline pattern is linear. One notable exception is KBANN, that exhibits an impressive performance, retaining nearly 70% accuracy even for highest flipping probability values ($f = 0.9$)—as opposed to other models whose accuracy drop to 20%. Such behaviour is further emphasised by Table 1, where KBANN demonstrates a remarkable improvement in robustness w.r.t. the uneducated model.

Except for KBANN, the other predictors exhibit similar performance trends w.r.t. the uneducated model across all three scenarios. Looking at Table 1, KBANN generally experiences a slight decrease in robustness compared to the uneducated model—except for the PSJGS case. On the other hand, KILL emerges as the predictor that best withstands the perturbations caused by label flipping.

Discussion and take-home message These experiments show that among the perturbations analysed, data drop elicits the greatest robustness gains from SKI methods, indicating their ability to compensate for missing information through knowledge integration. The PSJGS dataset is an exception, indeed the injected knowledge is far from optimal, being good for classifying only one class out of three. In this case, poor prior knowledge combined with missing data may hinder the model from learning underlying patterns. Regarding noise perturbations, the SKI methods employed prove insufficient to improve robustness. For label flipping perturbations, the constraining method—KILL—demonstrates good robustness across all three datasets, suggesting that penalty added during training is sufficient to compensate for label flipping. The exception to KBANN on PSJGS likely stems from its strong adherence to the injected knowledge, which provides useful joint information on splice junctions and mitigates the influence of flipped labels.

In summary, the experimental findings are: *(i)* SKI approaches demonstrate heightened robustness when data is limited, owed to the alternate guidance of integrated knowledge. *(ii)* Loss-manipulating SKI techniques like KILL better tolerate label corruptions because their algorithm de-emphasises possibly-flawed labels during backpropagation. *(iii)* SKI structuring methods like KBANN are more robust than constrained-layer types like KINS since injected knowledge is maintained intact.

5. Conclusions

In this paper we propose a novel metric to assess the robustness of SKI mechanisms. We define three different types of data perturbation strategies along with a way to assess their intensity. We also provide a formula to calculate the robustness score of a predictor, gauging its robustness when the data is

perturbed. Furthermore, we introduce a comparison to understand whether the SKI predictor is better (or not) than its uneducated counterpart from a robustness perspective.

Overall, our experiments indicate that our metric can be used to effectively determine the robustness of the performance variations obtained through SKI. These findings serve as a solid foundation that paves the way for a range of potential evaluations of model robustness, including those in the realms of symbolic and sub-symbolic integration. By evaluating the robustness of educated predictors, we gain a better understanding of their performance under varying circumstances and enhance their ability to manage uncertainty and complexity inherent in real-world scenarios, thus leading to trustworthy and effective AI systems.

Acknowledgements

This work has been partially supported by: *(i)* "FAIR–Future Artificial Intelligence Research", Spoke 8 "Pervasive AI" (PNRR, M4C2, Investimento 1.3, Partenariato Esteso PE00000013), funded by the EC under the NextGenerationEU programme; and by *(ii)* "ENGINES — ENGineering INtElligent Systems around intelligent agent technologies" project funded by the Italian MUR program "PRIN 2022" (G.A. 20229ZXBZM).

References

[1] A. B. Nassif, I. Shahin, I. B. Attili, M. Azzeh, K. Shaalan, Speech Recognition Using Deep Neural Networks: A Systematic Review, IEEE Access 7 (2019) 19143–19165. doi:10.1109/ACCESS.2019.2896880.

[2] Z.-Q. Zhao, P. Zheng, S.-t. Xu, X. Wu, Object Detection With Deep Learning: A Review, IEEE Transactions on Neural Networks and Learning Systems 30 (2019) 3212–3232. doi:10.1109/TNNLS.2018.2876865.

[3] A. Agiollo, A. Omicini, GNN2GNN: Graph neural networks to generate neural networks, in: J. Cussens, K. Zhang (Eds.), Uncertainty in Artificial Intelligence, volume 180 of *Proceedings of Machine Learning Research*, ML Research Press, Maastricht, The Netherlands, 2022, pp. 32–42. URL: https://proceedings.mlr.press/v180/agiollo22a.html, proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands.

[4] J. Zhang, B. Chen, L. Zhang, X. Ke, H. Ding, Neural, symbolic and neural-symbolic reasoning on knowledge graphs, AI Open 2 (2021) 14–35. doi:10.1016/J.AIOPEN.2021.03.001.

[5] A. Agiollo, A. Rafanelli, M. Magnini, G. Ciatto, A. Omicini, Symbolic knowledge injection meets intelligent agents: QoS metrics and experiments, Autonomous Agents and Multi-Agent Systems 37 (2023) 27:1–27:30. doi:10.1007/s10458-023-09609-6.

[6] Y. Xie, Z. Xu, K. S. Meel, M. S. Kankanhalli, H. Soh, Embedding symbolic knowledge into deep networks, in: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 4235–4245. URL: https://proceedings.neurips.cc/paper/2019/hash/7b66b4fd401a271a1c7224027ce111bc-Abstract.html.

[7] L. d. Raedt, S. Dumančić, R. Manhaeve, G. Marra, From statistical relational to neuro-symbolic artificial intelligence, in: C. Bessiere (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, 2020, pp. 4943–4950. doi:10.24963/ijcai.2020/688.

[8] G. Ciatto, F. Sabbatini, A. Agiollo, M. Magnini, A. Omicini, Symbolic knowledge extraction and injection with sub-symbolic predictors: A systematic literature review, ACM Computing Surveys 56 (2024) 161:1–161:35. doi:10.1145/3645103.

[9] D. Hendrycks, T. G. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019. URL: https://openreview.net/forum?id=HJz6tiCqYm.

[10] A. Agiollo, A. Omicini, Measuring trustworthiness in neuro-symbolic integration, in: M. Ganzha, L. Maciaszek, M. Paprzycki, D. Ślęzak (Eds.), Proceedings of the 18th Conference on Computer Science and Intelligence Systems, volume 35 of *Annals of Computer Sciences and Information Systems*, 2023, pp. 1–10. doi:10.15439/2023F6019.

[11] M. Magnini, G. Ciatto, A. Omicini, On the design of PSyKI: a platform for symbolic knowledge injection into sub-symbolic predictors, in: D. Calvaresi, A. Najjar, M. Winikoff, K. Främling (Eds.), Explainable and Transparent AI and Multi-Agent Systems, volume 13283 of *Lecture Notes in Computer Science*, Springer, Cham, Switzerland, 2022, pp. 90–108. doi:10.1007/978-3-031-15565-9_6, 4th International Workshop, EXTRAAMAS 2022, Virtual Event, May 9–10, 2022, Revised Selected Papers.

[12] X. Liu, M. Cheng, H. Zhang, C. Hsieh, Towards robust neural networks via random self-ensemble, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII, volume 11211 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 381–397. doi:10.1007/978-3-030-01234-2_23.

[13] G. Montavon, G. B. Orr, K. Müller (Eds.), Neural Networks: Tricks of the Trade - Second Edition, volume 7700 of *Lecture Notes in Computer Science*, Springer, 2012. doi:10.1007/978-3-642-35289-8.

[14] C. Shorten, T. M. Khoshgoftaar, A survey on image data augmentation for deep learning, Journal of Big Data 6 (2019) 60:1–60:48. doi:10.1186/s40537-019-0197-0.

[15] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, R. Webb, Learning from simulated and unsupervised images through adversarial training, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, IEEE Computer Society, 2017, pp. 2242–2251. doi:10.1109/CVPR.2017.241.

[16] J. Franceschi, A. Fawzi, O. Fawzi, Robustness of classifiers to uniform l_p and Gaussian noise, in: A. Storkey, F. Perez-Cruz (Eds.), International Conference on Artificial Intelligence and Statistics (AISTATS 2018), volume 84 of *Proceedings of Machine Learning Research*, ML Research Press, Playa Blanca, Lanzarote, Spain, 2018, pp. 1280–1288. URL: http://proceedings.mlr.press/v84/franceschi18a/franceschi18a.pdf.

[17] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: Y. Bengio, Y. LeCun (Eds.), 2nd International Conference on Learning Representations (ICLR 2014), Conference Track Proceedings, Banff, AB, Canada, 2014. URL: https://openreview.net/forum?id=kklr_MTHMRQjG.

[18] European Commission, Ethics guidelines for trustworthy AI, https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai, 2019.

[19] T. R. Besold, A. S. d'Avila Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K. Kühnberger, L. C. Lamb, P. M. V. Lima, L. de Penning, G. Pinkas, H. Poon, G. Zaverucha, Neural-symbolic learning and reasoning: A survey and interpretation, in: P. Hitzler, M. K. Sarker (Eds.), Neuro-Symbolic Artificial Intelligence: The State of the Art, volume 342 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2021, pp. 1–51. doi:10.3233/FAIA210348.

[20] S. Verma, J. P. Dickerson, K. Hines, Counterfactual explanations for machine learning: A review, CoRR abs/2010.10596 (2020). URL: https://arxiv.org/abs/2010.10596. arXiv:2010.10596.

[21] J. M. Joyce, Kullback-Leibler divergence, in: M. Lovric (Ed.), International Encyclopedia of Statistical Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 720–722. doi:10.1007/978-3-642-04898-2_327.

[22] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org.

[23] 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.), Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of *Proceedings of Machine Learning Research*, PMLR, 2018, pp. 5498–5507. URL: http://proceedings.mlr.press/v80/xu18h.html.

[24] A. Yazdani, L. Lu, M. Raissi, G. E. Karniadakis, Systems biology informed deep learning for inferring parameters and hidden dynamics, PLoS Computational Biology 16 (2020). doi:10.1371/journal.pcbi.1007575.

[25] P. Kairouz, Z. Liu, T. Steinke, The distributed discrete gaussian mechanism for federated learning with secure aggregation, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, volume 139 of *Proceedings of Machine Learning Research*, PMLR, 2021, pp. 5201–5212. URL: http://proceedings.mlr.press/v139/kairouz21a.html.

[26] P. Körner, M. Beuschel, J. Barbosa, V. S. Costa, V. Dahl, M. V. Hermenegildo, J. F. Morales, J. Wielemaker, D. Diaz, S. Abreu, G. Ciatto, Fifty years of Prolog and beyond, Theory and Practice of Logic Programming 22 (2022) 776–858. doi:10.1017/S1471068422000102.

[27] G. G. Towell, J. W. Shavlik, M. O. Noordewier, Refinement of approximate domain theories by knowledge-based neural networks, Proceedings of the 8th National Conference on Artificial Intelligence (1990) 861–866. URL: http://www.aaai.org/Library/AAAI/1990/aaai90-129.php.

[28] M. Magnini, G. Ciatto, A. Omicini, KINS: Knowledge injection via network structuring, in: R. Calegari, G. Ciatto, A. Omicini (Eds.), CILC 2022 – Italian Conference on Computational Logic, volume 3204 of *CEUR Workshop Proceedings*, 2022, pp. 254–267. URL: http://ceur-ws.org/Vol-3204/paper_25.pdf.

[29] M. Magnini, G. Ciatto, A. Omicini, A view to a KILL: Knowledge injection via lambda layer, in: A. Ferrando, V. Mascardi (Eds.), WOA 2022 – 23rd Workshop "From Objects to Agents", volume 3261 of *CEUR Workshop Proceedings*, Sun SITE Central Europe, RWTH Aachen University, 2022, pp. 61–76. URL: http://ceur-ws.org/Vol-3261/paper5.pdf.

[30] L. von Rüden, J. Garcke, C. Bauckhage, How does knowledge injection help in informed machine learning?, in: International Joint Conference on Neural Networks, IJCNN 2023, Gold Coast, Australia, June 18-23, 2023, IEEE, 2023, pp. 1–8. doi:10.1109/IJCNN54540.2023.10191994.

[31] P. E. McKnight, J. Najab, Mann-Whitney U Test, John Wiley and Sons, Ltd, 2010, pp. 1–1. doi:10.1002/9780470479216.corpsy0524.