What Does It Take to Reproduce Experiments? Evidences from the Neuro-Symbolic Domain

Luisa Sophie Werner, Nabil Layaïda, Pierre Genevès, Jérôme Euzenat

What Does It Take to Reproduce Experiments? Evidences from the Neuro-Symbolic Domain

Luisa Werner[1] , **Nabil Layaïda**[1] , **Pierre Genevès**[1] and **Jérôme Euzenat**[1]
[1]Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, 38000 Grenoble, France
{first.second}@inria.fr

Abstract

Reproducibility is a desirable property of scientific experiments, which is gaining relevance in computer science. Although guidelines exist, it is not immediately obvious what is necessary for making an experiment reproducible and what is necessary for reproducing it. In this work, we address the task of independently reproducing results published in the neuro-symbolic domain. We describe the methodology used for reproducing (using the available code) and replicating (using an alternative implementation) the experiments conducted with Knowledge Enhanced Neural Networks. We also extend them by applying the KENN to additional datasets. For each of these steps, we identify issues that may arise. This work shows that reproducibility should not be taken for granted. We discuss solutions that have proven effective in overcoming the encountered problems. This can be used as a guide for further reproducibility studies and generally improve reproducibility in machine learning.

1 Introduction

Experiment reproducibility is a cornerstone of scientific research. It is so for the experimenters because it allows them to be more confident in the results they claim. It is so for the other scientists because it allows them to stand on a solid ground in developing their own work. This leads to more reliable and superior research results, which is beneficial for the society as a whole.

In the machine learning community, a strong incentive emerges to provide the necessary information allowing to reproduce results [Pineau *et al.*, 2021]. More generally, this is a basic open science requirement (used for accountability as for reproducibility). However, guidelines are often limited to publishing code and data [IJCAI, 2022]. It is not obvious that such guidelines are sufficient to independently reproduce experiments in the current state of machine learning. Indeed, random operations may be introduced in several places, hyperparameters may change the behaviour of programs, components may exchange information in imprecise ways, the data processing pipeline may often be overlooked, etc.

Here, we report on our effort to reproduce previous experimental results in the fields of neuro-symbolic integration. Indeed, we base our work on that of Knowledge Enhanced Neural Networks (KENN) [Daniele and Serafini, 2020b]. Since KENN is well-suited to handle relational data, we re-implement it from TensorFlow to PyTorch to have access to the graph neural network library PyTorch Geometric [Fey and Lenssen, 2019] upon which we want to extend KENN in the future. To be sure that this re-implementation is reliable, we decide to reproduce and replicate the initial experiments.

Recently, there has been some hesitation on the terminology regarding reproducibility and in particular its application to computer science [Drummond, 2009; Goodman *et al.*, 2016; Rougier *et al.*, 2017; Plesser, 2018]. From here on, we use the 'standard' terminology adopted by ACM [Association for Computing Machinery, 2016]:

- **'repeat** means re-running an experiment with the same code, the same parameters, the same data by the same experimenter;
- **'reproduce'** means performing an experiment with the same code, the same parameters and the same data but by a different experimenter;
- **replicate'** means performing an experiment independently on the same data but using different software and by a different experimenter;
- to these definitions we add **'extend'** which means performing an experiment independently on different data.

We will use the word reproduction as the general term covering all these activities, as in the title of this paper.

We report on performing independent reproduction, replication and extension of a given experiment. This task is summarised in Table 1 and Figure 1. Given the purpose of this effort, namely re-implementing the same approach in a different framework, we identify two types of reproducibility goals.

The first one, as common in natural sciences, is to check

	Same software	Different software
Same datasets	**Reproduction** (§5)	**Replication** (§6)
Different datasets	**Extension** (§7)	

Table 1: Experiments and corresponding reproduction level.

whether the experimental results support the claims of the initial paper (**Type 1**). The second one, coming rather from engineering, aims at obtaining the same or approximately the same results as the initial experiment (**Type 2**).

Beyond that, the purpose of this work is to identify the obstacles that can be encountered when attempting to reproduce, replicate and extend experiments, and how they may be overcome. From this extensive experience, we draw lessons on what could be done in order to make experiments in neural networks and related fields more reproducible.

Outline We first briefly discuss the related work on reproduction methods (Section 2) before introducing the main concepts of KENN and providing necessary details on the reproduced experiment in this work (Section 3). We then illustrate the overall methodology deployed (Section 4) and report on the reproduction (Section 5), replication (Section 6), and extension (Section 7) of the initial experiments with KENN.

2 Related Work

Given the high degree of non-determinism of deep learning techniques in general, the problem of reproducing results has gained importance in this domain.

The reproducibility of experiments on graph neural network has been discussed in the context of their application to medical diagnostics [Nebli *et al.*, 2022]. This work resulted in guidelines that mandate making source code, data and parameters available, as it is now standard in open science [Pineau *et al.*, 2021; IJCAI, 2022].

Due to this increased awareness, there has been a surge in documented reproduction and replication attempts, particularly encouraged through 'reproducibility challenges' [Pineau *et al.*, 2021; Sinha *et al.*, 2022]. Such challenges ask interested individuals to try to replicate results from recent papers. Our approach is somewhat different from their motivations, as our goal is not to achieve reproducibility for its own sake. We reproduce experiments because we want to be sure that the system we re-implement retains the properties of the initial implementation. Therefore, we offer a thoughtful software engineering approach that integrates reproduction, replication and extension.

Most efforts aim at reproducibility according to the aforementioned definition and are faced with difficulties such as the lack of documentation of details, e.g. hyperparameters [Ankit *et al.*, 2022] or failure to reproduce [Blanco *et al.*, 2020].

Far closer to our work, [Holdijk *et al.*, 2021] report on the tentative to replicate an experiment comparing GNN explainers, as opposed to classifiers. Similar to us, the authors also re-implement the approach from TensorFlow to PyTorch. In this case, the replication highlights issues in the documentation (code-paper mismatch, missing architectural details) and evaluation procedures (choice of ground truth).

3 Experiments with Knowledge Enhanced Neural Networks

To overcome the limitations of Deep Learning such as data-hungriness or black-box properties, the emerging research field of neuro-symbolic integration aims at combining neural approaches with methods of symbolic AI [Garnelo and Shanahan, 2019].

As an approach from this domain, KENN [Daniele and Serafini, 2020b] has recently been designed. KENN consists of two components. First, a base neural network, in the following called NN, solves a classification task given numeric data. Second, a module of knowledge enhancement layers elaborates on a given set of prior knowledge expressed in first-order logic. The whole architecture of the NN and the knowledge enhancement layers is end-to-end differentiable.

Each knowledge enhancement layer implements a differentiable *t-conorm boost function* that revises the predictions y of the base neural network with respect to the satisfaction of the logical clauses and returns updated predictions y'.

To be incorporated in a neural network architecture, the knowledge has to be interpreted in the real-valued domain. Therefore, fuzzy logic [Marra *et al.*, 2020] is applied to map binary truth values to a continuous interval of $[0, 1]$. The preactivations of the NN are used as interpretations of unary predicates in the numeric domain and t-conorm functions map the truth values of atoms to the truth value of a clause. A knowledge enhancement layer contains *clause weights* as trainable parameters which are jointly optimized with the parameters of the NN at training stage. The clause weight quantifies the importance of a logical clause. A module called *clause enhancer* is instantiated for each clause and applies modifications to the preactivations of the NN. The changes introduced by all clause enhancers are aggregated, added to the preactivations and fed into a logistic function in order to obtain the final predictions of the entire model. By introducing binary predicates into its logic language, KENN is also applicable to relational data.

In [Daniele and Serafini, 2020b], KENN has been applied to a collective classification task on the Citeseer dataset [Lu and Getoor, 2003]. The Citeseer dataset consists of 3.312 scientific publications belonging to one of the six computer science research fields Agents (AG), Artificial Intelligence (AI), Human-Computer Interaction (HCI), Machine Learning (ML), Databases (DB) or Information Retrieval (IR). Each publication is described by a one-hot-encoded vector that encodes the abstract and the title of the paper. A total of 4.732 citations amongst documents exist. The dataset can be modelled as graph where scientific publications are represented by nodes and citations by edges. The one-hot-encoded word vectors are used as node features.

The prior knowledge provided to the knowledge enhancement layers is derived from the assumption that two publications that cite each other fall into the same document class. According to the following schema, one logical clause is defined for each document class to finally obtain a knowledge base of six clauses

$$\forall x \forall y : \neg Class(x) \vee \neg Cite(x; y) \vee Class(y)$$

While the preactivations of the NN are used as numeric interpretation of the unary predicates AG(x), AI(x), HCI(x), ML(x) DB(x) and IR(x), the citations between two publications are assumed to be known apriori. For this reason, the preactivations of the binary predicate Cite(x,y) are set to a

high constant value.

In [Daniele and Serafini, 2020b], the authors report the performance of KENN for experiments on multiple training set sizes (10%, 25%, 50%, 75%, 90%). For each experiment, 100 independent runs are conducted. KENN is compared to the standalone NN, as well as to the neuro-symbolic baselines Relational Neural Machines (RNM) [Marra *et al.*, 2020] and Semantic-Based Regularization (SBR) [Diligenti *et al.*, 2017]. From the experiments, the following conclusions are drawn:

H1 KENN consistently outperforms the NN for all training set sizes.

H2 The performance gain due to the knowledge enhancement is larger when training data is limited.

H3 KENN leads to comparable and even superior performance with respect to RNM and SBR.

Here, we focus on the two former hypotheses.

4 Methodology

This section gives an overview of the applied reproduction methodology in this work.

4.1 Material

In the study of reproducibility, we refer to the following publicly available material. The reproduction and replication is based on the results reported in the paper [Daniele and Serafini, 2020b] (that we call it the *initial paper*), and on the reported experiments[1] (that we call the *initial experiments*). The experiments make use of the Python package KENN2[2]. In addition, we also extract information from other related papers [Daniele and Serafini, 2019; Daniele and Serafini, 2020a; Daniele and Serafini, 2022].

4.2 Process

In life sciences, reproducibility refers to the use of different instruments, lab routine, or test subject. In computer science, replicability requires the use of a different implementation, eventually in a different programming language or (in this case) in a different framework. We to extend KENN in the deep learning framework PyTorch [Paszke *et al.*, 2019] in conjunction with the graph learning library PyTorch Geometric [Fey and Lenssen, 2019]. To this end, the re-implementation of KENN together with a comprehensive reproducibility study is the best way to ensure that KENN is extended on a reliable basis.

With this objective in mind, we conduct the following steps:

1. We reproduce the results obtained with the initial implementation (Section 5);
2. We re-implement the initial approach and the initial experiments and replicate them. (Section 6);
3. We extend the experiments to additional datasets (Section 7).

[1]https://github.com/rmazzier/KENN-Citeseer-Experiments

[2]https://github.com/DanieleAlessandro/KENN2

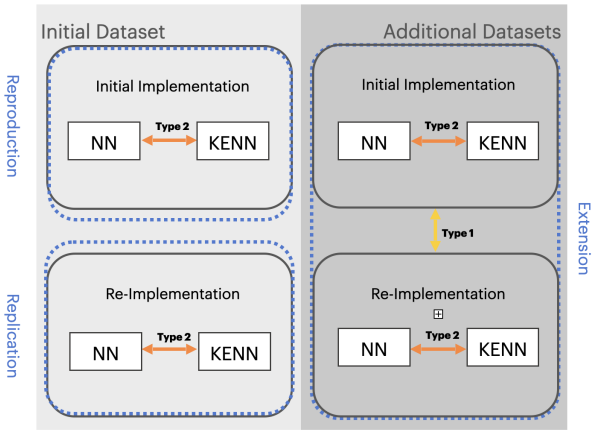

Figure 1: Overview of the definitions of reproduction (§5), replication (§6) and extension (§7) as well as type 1 and 2 reproducibility.

In the following we refer to our implementation as *re-implementation*. For each step, we provide a detailed description of the required information, the hurdles we had to overcome, how we managed to overcome them. Further, we report whether the statements of [Daniele and Serafini, 2020b] could be confirmed and the lesson learned from the effort. We tried to preserve the input (dataset) and output (results) format of the initial experiment as close as possible. This allows us to interpret the results in the same way with the provided jupyter notebook. In this work, we focus on the transductive part of the initial experiments with KENN.

4.3 Evaluation Criteria

When it comes to the evaluation of the reproduction the experiments in [Daniele and Serafini, 2020b], two aspects regarding Type 1 and Type 2 reproducibility objectives (as defined in Section 1 are considered:

- Do the results support hypotheses $H1$ and $H2$ (see Section 3)? (Type 1)
- Do the obtained results correspond (to some extent) to those reported in [Daniele and Serafini, 2020b]? (Type 2)

Concerning the evaluation of $H1$ and $H2$, we adopt the criteria from the initial paper. To test $H1$, a one-sided independent Student t-test is employed to assess the superiority of the mean accuracy of KENN over NN. We retain 0.01 as significance threshold. Regarding hypothesis $H1$, which establishes a relation between the training set size and the delta of NN and KENN, no precise evaluation procedure is proposed in [Daniele and Serafini, 2020b]. The absolute difference between the mean test accuracies for the experiments is nevertheless observed and interpreted.

Secondly, we evaluate our replication and extension by examining the distributions of the reported test results. It is not reasonable to expect absolute equality of the results, since even re-executing the initial implementation twice does not lead to equal results. This point is discussed in more detail in Section 5. This implies the need to measure and assess the similarity of the distributions. Hence, we evaluate the similarity between the different probability distributions of the

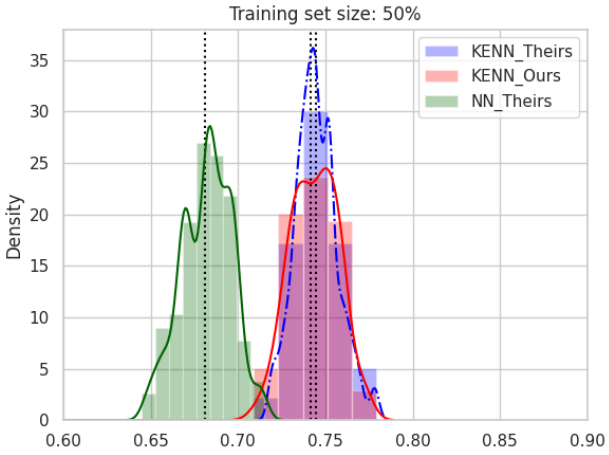

Figure 2: NN vs. reproduced and replicated results on Citeseer, with means (dotted), histograms and kernel density estimate plots.

observed results, as displayed in Figure 2. To this end, we computed the two-sided Kolmogorov-Smirnov goodness-of-fit test (KS-Test) [Massey, 1951], which checks whether two samples are drawn from the same distribution. If the p-value is below the significance threshold, there is evidence that the results of the experiments are not drawn from the same distribution and consequently diverge. While the test does not provide significance for equality, it indicates that there is no statistical evidence that the distributions are not equal.

5 Experiment Reproduction on Citeseer

As a first step, we reproduce (as defined in Section 1) the initial experiments with KENN.

5.1 Method

To reproduce the experiments, we need four elements: (*a*) the executable, (*b*) the dataset, (*c*) the instructions and environment information to run the executable, (*d*) the procedure to collect and interpret the results. With this, it is possible to process the instructions to reproduce the experiment.

5.2 Pitfalls and Workarounds

The first encountered problem is that the paper [Daniele and Serafini, 2020a] only provides the KENN2[2] without material on the experiments. Through an exchange with the authors, we gained access to the source code and data of the experiments, respectively. Later, the authors made this information publicly available on Github[1], but pointed to another paper [Daniele and Serafini, 2020b] which further documents the experiment. Parameters in the public implementation correspond to the results reported in [Daniele and Serafini, 2022] but not to the results found in [Daniele and Serafini, 2020a] and [Daniele and Serafini, 2020b].

The provided repository contains (*a*) a reference to the KENN2 package, (*b*) a link to the data set, (*c*) a README file with instructions to run the code, and (*d*) a Jupyter notebook to analyse the results. The link to the dataset was incorrect. Instead of referring to an external, widely available version of the dataset, the initial experiment refers to a dataset included in the repository. No additional information concerning the provenance of this dataset is given. Furthermore, no indication is mentionned on whether the initial dataset has been modified, for instance in a preprocessing or filtering stage.

The instructions to run the experiments are reasonably complete, including a description of the required Python modules (`requirement.txt` file) as well as the full command line to be run. However, the `requirement.txt` file does only contain a list of packages without their version number. This can be a critical concern as Python packages evolve frequently, sometimes compromising compatibility. The Python version is not defined either. For the successful execution of the experiments, the Python version is crucial. We deduced it from the reference of the KENN package.

In both papers [Daniele and Serafini, 2020a; Daniele and Serafini, 2020b], experimental results are reported in numeric tables. No explanation is given regarding the values in brackets. We initially assumed they were standard deviations. However, the authors clarified them as differences between the NN and KENN.

Even though random operations are seeded, we note a subtle variation in the results. We identified a non-deterministic set operation which may introduce variation. Hence, according to [PyTorch, 2022], exact reproducibility cannot be guaranteed.

5.3 Results

After having reached a fully functional setup of the environment, we successfully ran the experiments. The obtained results are reported in Table 2. In comparison to the results initially provided in [Daniele and Serafini, 2020b], we add additional information such as the standard deviation of the reported test accuracies over all runs, and the p-values for a statistical t-test. In terms of the evaluation of $H1$, KENN significantly outperforms the NN for all training set sizes ($p \ll 0.01$) which is consistent with the authors' claims. Regarding $H2$, the reported difference (column delta) of the mean accuracy with KENN in comparison to NN is observed larger when the training set size is small. Note that, since the initial experiments use paired samples, the average of deltas can be reported, though we can only report the difference of the means. The reported and reproduced results cannot be compared statistically, since we full sample of the reported results is unknown.

5.4 Lessons Learned

Several conferences, including IJCAI [IJCAI, 2022], promote reproduction by encouraging authors to provide source code, experimental descriptions and datasets of their papers.

A first lesson learned is that the complete description of the software environment including version numbers of all modules should be provided. Furthermore, the datasets used should also be described precisely, including information on their origin and on any preprocessing that has been applied to them. Both are critical information to reproduction.

A second lesson learned is that the reproduction should be made as automatic as possible by the initial authors. Such an

	Reported results			Initial Implementation		
train	NN	KENN	Delta	NN	KENN	Delta
10%	0.544	0.652	0.108	0.540 (0.067)	0.651 (0.017)	0.110 (0.067)
25%	0.629	0.702	0.073	0.630 (0.018)	0.702 (0.012)	0.072 (0.012)
50%	0.680	0.744	0.065	0.681 (0.187)	0.745 (0.012)	0.064 (0.021)
75%	0.733	0.788	0.055	0.733 (0.025)	0.791 (0.016)	0.058 (0.026)
90%	0.759	0.808	0.049	0.758 (0.027)	0.807 (0.022)	0.049 (0.028)

Table 2: Reproduction results: average accuracy on the test set reported in [Daniele and Serafini, 2020b] (left) vs. reproduced results with the initial implementation (right). The standard deviation of the test accuracies over 100 runs are reported between parentheses.

automation would even be useful for the initial authors to be able to rerun their experiments routinely.

The most important lesson learned is, that we miss a unique reference to the experiments. In fact, the authors published a first version of a paper [Daniele and Serafini, 2019] and then several others [Daniele and Serafini, 2020b; Daniele and Serafini, 2020a; Daniele and Serafini, 2022]. These versions contain improvements and come with additional experimental settings. Still, all papers point to the same repository. This complicates to reproduce the experiments of a given paper. A good practice would be to provide one repository, or one way to reconstruct the experimental setting, per paper. The scientific unit is the paper. If the unit should be reproducible, a dedicated repository is required.

6 Experiments Replication

In this section, we now refer to the re-implementation of KENN into PyTorch and the replication of the experiments with it. We compare the obtained results to the results of the published and reproduced experiments, using the criteria defined in Section 4.3.

6.1 Method

To re-implement KENN, we first identify the main components of the approach by examining, on the one hand, the concepts of KENN introduced in the initial paper. On the other hand, since the paper does not contain all required information, we also examine the code of the initial implementation.

In principle, the main components are (*a*) the data preprocessing, (*b*) the model definition, (*c*) the training loop, and (*d*) the hyperparameters definition. Given the information on these components, we can re-implement them as follows.

Data preprocessing. We use the same Citeseer dataset and prior knowledge as provided for the initial experiments. We preserve the data splitting procedure, as well as the rebalancing.

Model definition. The KENN model consists of two stacked components: The NN and the knowledge enhancement layers, as described in Section 3. In the initial implementation, the NN is implemented with Keras [Chollet and others, 2015] and the knowledge enhancement layers are referred from the KENN2[2] package written in Tensorflow. We re-implement both, the NN as well as the layers from the KENN2 package, in PyTorch. We integrate optional boost functions proposed in [van Krieken and Daniele, 2022].

Training loop. In the training loop, we replace the optimizer and the loss function implemented in Tensorflow by their PyTorch equivalent.

Hyperparameters definition. We identify the defined hyperparameters in the experiments and the values assigned to them. To keep track of hyperparameters in a clean manner, we decide to connect the re-implementation to an experiment tracking tool [Biewald, 2020].

The results were evaluated according to Section 4.3.

6.2 Pitfalls and Workarounds

At first sight, the re-implementation in PyTorch seemed straightforward. However, we struggled in identifying the hyperparameters used in the initial implementation. The relevant information had to be gathered from various sources or recovered from default values. The set of revealed hyperparameters, their assignments and how we recovered them is given in Table 3.

A first set of values for hyperparameters is explicitly named in the initial paper including the architecture of the NN, the initialisation of the clause weights and the binary preactivation value. An additional set of parameters is defined in a user modifiable Python script in the GitHub repository of the initial implementation[1]. These script contains information on early stopping (and related parameters), the number of epochs, as well as the size of the validation set. Additional parameter values are required that are not mentioned in the paper, nor in the code documentation. By reviewing the entire source code, we recovered the number of knowledge enhancement layers and the batch size, for example. An additional set of hyperparameters is implicitly introduced and defined by framework-specific functions and their default assignments, such as the weight initialization of the dense layers in the NN and the optimizer-specific parameters [Kingma and Ba, 2014]. These default values are found in the software documentation for these functions.

Determining the values of some additional parameters turned out to be even more challenging. To get useful insights, we examined the isolated behaviour of each implementation component to ensure that, given some input, they produce the same output. When training neural networks, many of these components are non-deterministic and rely on random numbers. While randomness is essential to learning, it complicates the analysis of numerical behaviour. For the inspection of some components, we temporarily simplified operations and replaced random numbers by fixed values. This step made it possible to identify that linear layers are by default differently initialized in Tensorflow and in PyTorch. In Tensorflow, they are initialized randomly following a Glorot uniform distribution and the bias is initialized with zeroes. However, in PyTorch, the weights and biases are randomly initialized following the uniform distribution.

Parameter	Value	In Paper	In Code	Defaults
NN - Number of Hidden Layers	3	✓	✓	
NN - Number of Hidden Neurons	50	✓	✓	
NN - Hidden Layer Activation	ReLu	✓	✓	
NN - Output Layer Activation	Linear	✓	✓	
KENN - Clause Weight Initialization	Constant, 0.5	✓	✓	
KENN - Binary Preactivations	500	✓	✓	
KENN - Number of KE Layers	3		✓	
KENN - Range Constraint	$[0.0, 500.0]$		✓	
Epochs	300		✓	
Batch Size	Full-batch		✓	
Loss function	Categorical Cross-entropy		✓	
Early Stopping - Patience	10		✓	
Early Stopping - Min Delta	0.001		✓	
Learning Rate	0.001			✓
NN - Weight Initialization	Random, Glorot uniform			✓
NN - Bias Initialization	Constant, Zeroes			✓
Optimizer	Adam, $\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-7}$			✓
Dropout Rate	0.0, no dropout			

Table 3: The set of hyperparameters used in the initial experiments.

6.3 Results

After recovering the full set of required parameter values and completing the re-implementation of KENN, we now assess the replicability of experiments. The results of the replication are summarized in Table 4.

First, we check whether $H1$ and $H2$ hold in the replicated experiment. We observe that the p-values for all training set sizes fall below the significance level, in favor of $H1$ which is thus supported by the replication.

For $H2$, considering the deltas of the mean test accuracies across the training set sizes, no clear relationship between training set size and its effect on the knowledge enhancement is apparent.

The average accuracies of both implementations are quite close (largest difference observed 0.02). However, their distributions is not identical. The similarity of accuracy distributions is computed against the reproduced results, since we do not have the initial results from the authors of KENN. The p-values of the Kolmogorov-Smirnov test are listed in the righthand column. For the 10% training size, a significant ($p < 0.01$) difference between the distributions of the reproduced and replicated results is detected. For the remaining training set sizes, no difference reveals to be significant. Hence, we consider our results as reasonably similar to those of the initial implementation, at least for large sample sizes. It seems that the lower accuracy returned with 10% training set is less variable than that obtained with the larger ones.

In conclusion, we have not been able to replicate the results concerning $H2$. Our inability to confirm $H2$, which was assessed in the initial paper without further test, may be due to our inability to re-implement KENN or to the lack of a precise evaluation metric in the initial experiment.

	Re-implementation			
train	NN	KENN	Delta of means	p-values (KS Test)
10%	0.550 (0.042)	0.629 (0.076)	0.079	0.001
25%	0.632 (0.016)	0.676 (0.088)	0.044	0.024
50%	0.681 (0.014)	0.741 (0.028)	0.060	0.583
75%	0.729 (0.022)	0.785 (0.039)	0.056	0.054
90%	0.748 (0.026)	0.806 (0.020)	0.058	0.702

Table 4: Replication results of KENN on the Citeseer dataset.

6.4 Lessons Learned

We draw the following lessons from the replicated experiments.

First, the identification of important hyperparameters together with their values is critical for reproducibility. They have the potential to considerably impact the results and thus the conclusions drawn from experiments. While some hyperparameters such as learning rate and batch size are standard in deep learning approaches, custom models often define their own hyperparameters, such as KENN's clause weights. Since parameters refer to different components, they are often declared at different steps in the experiments, which makes their identification even harder. A complete configuration file with an exhaustive list of all critical parameters, their description, and their assigned values should be made mandatory.

7 Results on Additional Datasets

After reproduction and replication of KENN, we now extend the experiments with two new datasets: Cora and Pubmed [Yang *et al.*, 2016]. The goal of this extension is to evaluate

the generalisation of KENN, and compare the behaviour of the two implementations on other datasets.

7.1 Material and Method

The Cora and Pubmed datasets are third-party citation networks similar to Citeseer. Nodes and edges represent scientific publications and citations between them. Cora contains 2.708 publications, 10.556 citations and 7 document classes. Pubmed is significantly larger as it contains 19.717 nodes, 88.648 citations and 3 document classes. The knowledge enhancement is obtained as described in Section 3: the knowledge for Cora contains 7 clauses and for Pubmed 3 clauses.

7.2 Results

The results for Cora and Pubmed are shown in Tables 5 and 6.

When applying KENN to Pubmed, we detected a performance issue in the initial implementation, specifically in the data preprocessing. We had to modify the initial preprocessing in order to scale from a performance perspective, while keeping it functionally equivalent.

For Cora, $H1$ is supported in both implementations ($p \ll 0.01$). Concerning $H2$, the situation is less clear. Given the deltas in the initial implementation, a relationship between decreasing training set size and knowledge enhancement can be discovered. However, this difference is rather uncertain for the second implementation.

Comparing the distribution of test accuracies in both experiments, we observe that the two implementations exhibit similar behaviour. Specifically, when comparing the distribution of the test accuracies in the both experiments, the p-value of the KS-test is considered. Their p-value does not lie below the significance level. Therefore, no significant difference between the behaviour of both implementations is observed.

For Pubmed, the p-values of the KS-test are not below the significance threshold across all training set sizes. (In the initial implementation only for training set sizes 10% and 25% and for the re-implementation for 10%, 50%, 75% and 90%). This rejects $H1$ on Pubmed for both implementations. Hence, no significant additional accuracy was gained with KENN. Furthermore, no monotonically increasing benefit of knowledge enhancement for smaller training set sizes can be observed with both implementations, therefore $H2$ is also rejected for Pubmed.

7.3 Lessons Learned

Again, we observed that hyperparameter tuning significantly affects the results of the experiment. The hyperparameter values considered earlier do not provide the best results with these new datasets. Using hyperparameter tuning tools, it is possible to find other sets of hyperparameter values that produce more accurate results.

Overall, the extended experiments show that KENN improves the accuracy on the Cora dataset, but not on the Pubmed dataset. As a lesson learned, it can be derived that the application of a model to various datasets can either strengthen result by showing robustness or reveal weaknesses of approaches with respect to specific datasets.

	Initial Implementation			**Re-implementation**			
train	NN	KENN	Delta	NN	KENN	Delta of means	p-values (KS-Test)
10%	0.530 (0.029)	0.750 (0.017)	0.220 (0.030)	0.576 (0.016)	0.766 (0.010)	0.190	$8.9\ 10^{-4}$
25%	0.606 (0.018)	0.800 (0.012)	0.193 (0.016)	0.647 (0.009)	0.819 (0.012)	0.173	$8.4\ 10^{-10}$
50%	0.652 (0.013)	0.833 (0.009)	0.187 (0.017)	0.678 (0.009)	0.831 (0.013)	0.152	$5.9\ 10^{-1}$
75%	0.691 (0.016)	0.850 (0.014)	0.159 (0.018)	0.686 (0.013)	0.833 (0.015)	0.147	$2.9\ 10^{-4}$
90%	0.715 (0.027)	0.871 (0.016)	0.156 (0.028)	0.743 (0.012)	0.913 (0.017)	0.170	$9.2\ 10^{-11}$

Table 5: Extension results on the Cora dataset.

	Initial Implementation			**Re-implementation**			
train	NN	KENN	Delta	NN	KENN	Delta of means	p-values (KS-Test)
10%	0.333 (0.096)	0.405 (0.002)	0.071 (0.096)	0.326 (0.098)	0.404 (0.003)	0.077	$3.9\ 10^{-1}$
25%	0.341 (0.108)	0.416 (0.002)	0.075 (0.108)	0.380 (0.080)	0.414 (0.004)	0.034	$7.1\ 10^{-2}$
50%	0.409 (0.095)	0.443 (0.004)	0.033 (0.096)	0.364 (0.133)	0.441 (0.005)	0.077	$1.3\ 10^{-1}$
75%	0.447 (0.143)	0.498 (0.004)	0.051 (0.143)	0.414 (0.176)	0.495 (0.007)	0.081	$1.3\ 10^{-1}$
90%	0.505 (0.011)	0.510 (0.008)	0.004 (0.011)	0.504 (0.015)	0.504 (0.015)	-0.0001	$5.9\ 10^{-1}$

Table 6: Extension results on the Pubmed dataset.

8 Conclusion and Perspective

In this paper, we presented our efforts to reproduce KENN as an approach from the field of neurosymbolics so that it can be extended with confidence. After having re-implemented it, we investigated step by step the points of reproduction, replication and extension. We judged our success with defined metrics, related to type 1 reproducibility and type 2 reproducibility, which refer to the validity of the hypotheses and the equality of the distribution of results, respectively. For each stage, we recap on our lessons learned.

Regarding the reproduction, we observed that the availability of a paper's source code together with a more complete documentation of it (e.g. package version numbers) is essential to conduct the experiments.

In terms of replication, the clarity about hyperparameters and the specification of the experimental design is critical. Publishing this information in a standardised way should become an established practice in machine learning.

Referring to the extension, we remark that the evaluation of approaches on multiple datasets can confirm the robustness of an approach or reveal potential weaknesses.

In conclusion, we can derive the following statements on KENN and its reproducibility. In nearly all experiments across the three tasks, $H1$, the fact that KENN improves the NN, can be approved. Thus, this study confirms the main thesis of KENN and supports its reliability in both implementations. With respect to the type 2 reproducibility goal, we were unable to obtain clear results on all three tasks. It should be noted that this does not necessarily speak against the achievement of reproducibility, but may also be a question of measurability. Further research is needed in this direction in order to be able to perform reproducibility studies in the future.

Supplementary Material

As supplementary material, we provide our re-implementation of the experiments including the re-implementation of the KENN layers in PyTorch. In detail, we provide

- the source code
- the execution instructions
- the software requirements
- the hyperparameters
- the result evaluation script
- the raw files of the experiment results

We also attach the adapted source code for the initial experiments to the additional datasets.

Ethical Statement

The authors declare that they have no conflicts of interests.

Acknowledgements

We thanks the authors of KENN for their help.

This work has been partially supported by MIAI @ Grenoble Alpes (ANR-19-P3IA-0003).

References

[Ankit *et al.*, 2022] Ankit Ankit, Sameer Ambekar, Baradwaj Varadharajan, and Mark Alence. [re] counterfactual generative networks. *ReScience C*, 8(#2), 2022.

[Association for Computing Machinery, 2016] Association for Computing Machinery. Artifact review and badging, 1.1, 2016. https://www.acm.org/publications/policies/artifact-review-badging.

[Biewald, 2020] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[Blanco *et al.*, 2020] Wilfredo Blanco, Paulo Lopes, Anderson Abner de Souza, and Michael Mascagni. Non-replicability circumstances in a neural network model with hodgkin-huxley-type neurons. *Journal of computational neuroscience*, 48(3):357–363, 2020.

[Chollet and others, 2015] Francois Chollet et al. Keras, 2015.

[Daniele and Serafini, 2019] Alessandro Daniele and Luciano Serafini. Knowledge enhanced neural networks. In *Proc. Pacific Rim Conference on Artificial Intelligence*, number 11670 in Lecture notes in computer science, pages 542–554, 08 2019.

[Daniele and Serafini, 2020a] Alessandro Daniele and Luciano Serafini. Neural networks enhancement through prior logical knowledge. *ArXiv*, abs/2009.06087, 2020.

[Daniele and Serafini, 2020b] Alessandro Daniele and Luciano Serafini. Neural networks enhancement with logical knowledge, 2020.

[Daniele and Serafini, 2022] Alessandro Daniele and Luciano Serafini. Knowledge enhanced neural networks for relational domains, 2022. https://arxiv.org/abs/2205.15762.

[Diligenti *et al.*, 2017] Michelangelo Diligenti, Marco Gori, and Claudio Saccà. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017. Combining Constraint Solving with Mining and Learning.

[Drummond, 2009] Christopher Drummond. Replicability is not reproducibility: nor is it good science, 2009. http://cogprints.org/7691/.

[Fey and Lenssen, 2019] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[Garnelo and Shanahan, 2019] Marta Garnelo and Murray Shanahan. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29:17–23, 2019. Artificial Intelligence.

[Goodman *et al.*, 2016] Steven Goodman, Daniele Fanelli, and John Ioannidis. What does research reproducibility mean? *Science translational medicine*, 8(341), 2016.

[Holdijk *et al.*, 2021] Lars Holdijk, Maarten Boon, Stijn Henckens, and Lysander de Jong. [re] parameterized explainer for graph neural network. *ReScience C*, 7(#7), 2021.

[IJCAI, 2022] IJCAI. Reproducibility guideline, 2022. https://ijcai-22.org/reproducibility/.

[Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[Lu and Getoor, 2003] Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 496–503. AAAI Press, 2003.

[Marra *et al.*, 2020] Giuseppe Marra, Michelangelo Diligenti, Francesco Giannini, Marco Gori, and Marco Maggini. Relational neural machines, 2020.

[Massey, 1951] Frank Massey. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[Nebli *et al.*, 2022] Ahmed Nebli, Mohammed Amine Gharsallaoui, Zeynep Gürler, and Islem Rekik. Quantifying the reproducibility of graph neural networks using multigraph data representation. *Neural networks*, 148(3):254–265, 2022.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[Pineau *et al.*, 2021] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Lariviere, Alina Beygelzimer, Florence d'Alche-Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *Journal of machine learning research*, 22(164):1–20, 2021.

[Plesser, 2018] Hans Plesser. Reproducibility vs. replicability: a brief history of a confused terminology. *Frontiers in neuroinformatics*, 11, 2018.

[PyTorch, 2022] PyTorch. Reproducibility, 2022. https://pytorch.org/docs/stable/notes/randomness.html.

[Rougier *et al.*, 2017] Nicolas P. Rougier, Konrad Hinsen, Frédéric Alexandre, Thomas Arildsen, Lorena A. Barba, Fabien C.Y. Benureau, C. Titus Brown, Pierre de Buyl, Ozan Caglayan, Andrew P. Davison, Marc-André Delsuc, Georgios Detorakis, Alexandra K. Diem, Damien Drix, Pierre Enel, Benoît Girard, Olivia Guest, Matt G. Hall, Rafael N. Henriques, Xavier Hinaut, Kamil S. Jaron, Mehdi Khamassi, Almar Klein, Tiina Manninen, Pietro Marchesi, Daniel McGlinn, Christoph Metzner, Owen Petchey, Hans Ekkehard Plesser, Timothée Poisot, Karthik Ram, Yoav Ram, Etienne Roesch, Cyrille Rossant, Vahid Rostami, Aaron Shifman, Jemma Stachelek, Marcel Stimberg, Frank Stollmeier, Federico Vaggi, Guillaume Viejo, Julien Vitay, Anya E. Vostinar, Roman Yurchak, and Tiziano Zito. Sustainable computational science: the ReScience initiative. *PeerJ Computer Science*, 3:e142, dec 2017.

[Sinha *et al.*, 2022] Koustuv Sinha, Jesse Dodge, Sasha Luccioni, Jessica Zosa Forde, Sharath Chandra Raparthy, Joelle Pineau, and Robert Stojnic. Ml reproducibility challenge 2021. *ReScience C*, 8(#48), 2022.

[van Krieken and Daniele, 2022] Emile van Krieken and Alessandro Daniele. KENN: Knowledge enhanced neural networks., 2022. https://github.com/HEmile/KENN-PyTorch.

[Yang *et al.*, 2016] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings, 2016.