PiShield: A PyTorch Package for Learning with Requirements

Mihaela Cătălina Stoian[1] , **Alex Tatomir**[1] , **Thomas Lukasiewicz**[2,1] and **Eleonora Giunchiglia**[2]
[1]Department of Computer Science, University of Oxford, UK
[2]Institute of Logic and Computation, Vienna University of Technology, Austria

Abstract

Deep learning models have shown their strengths in various application domains, however, they often struggle to meet safety requirements for their outputs. In this paper, we introduce PiShield, the first package ever allowing for the integration of the requirements into the neural networks' topology. PiShield guarantees compliance with these requirements, regardless of input. Additionally, it allows for integrating requirements both at inference and/or training time, depending on the practitioners' needs. Given the widespread application of deep learning, there is a growing need for frameworks allowing for the integration of the requirements across various domains. Here, we explore three application scenarios: functional genomics, autonomous driving, and tabular data generation.

1 Introduction

Deep neural networks (DNNs) have shown their strengths in various application domains. However, they often fail to comply with given requirements defining the safe output space of the model. To obviate this problem, neuro-symbolic AI methods were introduced, which can be broadly classified into two categories. The first comprises methods able to integrate the requirements in the loss function and penalize the models when they violated the requirements [Diligenti *et al.*, 2012; Diligenti *et al.*, 2017; Donadello *et al.*, 2017; Xu *et al.*, 2018; Fischer *et al.*, 2019; Nandwani *et al.*, 2019; Badreddine *et al.*, 2022; Li *et al.*, 2023; Ahmed *et al.*, 2022c; Stoian *et al.*, 2023]), and while these approaches help reduce the requirements' violation incidence, they cannot guarantee their satisfaction. Emerging later, the second category consists of methods able to incorporate a given set Π of requirements (also called constraints) directly in the topology of the network [Giunchiglia and Lukasiewicz, 2021; Hoernle *et al.*, 2022; Ahmed *et al.*, 2022b] and, thus, to guarantee their satisfaction. More recently, an alternative method [van Krieken *et al.*, 2023], also capable of guaranteeing compliance with the constraints, proposed using neural networks for performing approximate inference in polynomial time to address the scalability problem of probabilistic neuro-symbolic learning frameworks such as DeepProbLog [Manhaeve *et al.*, 2018]. For an in-depth survey of the methods combining deep learning with logical constraints, we refer to [Giunchiglia *et al.*, 2022], while for a broad survey on neuro-symbolic AI, we refer to [d'Avila Garcez *et al.*, 2019].

In this paper, we propose PiShield[1,2], a PyTorch-based package allowing for seamlessly integrating domain requirements into neural networks by means of new PyTorch layers that can be built on top of any neural network. These layers, which we call Shield Layers, adhere to the principles outlined in our most recent works [Stoian *et al.*, 2024; Giunchiglia *et al.*, 2024], advocating for a more requirements-driven machine learning as in [Giunchiglia *et al.*, 2023a], and guarantee the satisfaction of the requirements regardless of the input.

The Shield Layers can be applied during inference and/or training, depending on the practitioners' needs. For users that have restricted access to a model but require that the outputs of their models are compliant with a set of rules, PiShield offers an easy-to-use interface to meet this need. Alternatively, PiShield can guide model training, suiting also practitioners accustomed to modifying their models. As expected, its ability to ensure compliance makes it ideal for safety-critical scenarios. Furthermore, studies [Giunchiglia and Lukasiewicz, 2020; Giunchiglia *et al.*, 2024; Giunchiglia and Lukasiewicz, 2021; Stoian *et al.*, 2024] have repeatedly shown its efficacy in aiding model learning. We illustrate both of these aspects on three different application domains: functional genomics, autonomous driving, and tabular data generation domains.

Related Work. A closely related work is Pylon [Ahmed *et al.*, 2022a], a framework built on PyTorch which allows users to integrate constraints into a loss function. Similarly, the LTN package [Badreddine *et al.*, 2022] provides a TensorFlow implementation of the Logic Tensor Networks (LTNs), while LTNTorch [Carraro, 2022] provides its PyTorch implementation. However, unlike our framework, all the methods described above cannot guarantee the satisfaction of the constraints.

2 PiShield Overview

PiShield is built on top of PyTorch, allowing for a seamless integration of Shield Layers into neural networks. Figure 1

[1]Code: https://github.com/mihaela-stoian/PiShield
[2]Website: https://sites.google.com/view/pishield

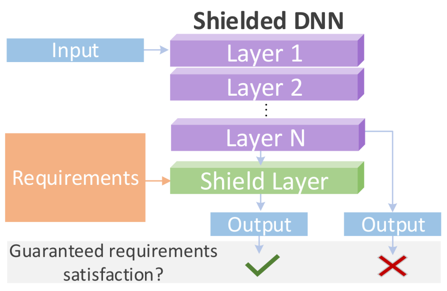

Figure 1: PiShield overview.

gives an overview of how to apply PiShield to deep neural networks. As shown in the Figure, to constrain a DNN during training, the architecture of the DNN needs to be changed by appending a Shield Layer immediately after the output layer. To use PiShield only at inference time, it is enough to simply build a Shield Layer outside the neural network and apply it on the outputs to make them compliant with the requirements. Regardless of its use, each Shield Layer requires only two elements to be instantiated: (i) the dimension of the input to the layer (hence in most cases the dimension of the output of the network), and (ii) the path to the file which contains the requirements.

Requirements. The requirements can be expressed either as a propositional logic formula in conjunctive normal form (CNF) or as linear inequalities. In the file, each line should contain a single requirement, which will thus be either a clause in the first case or a single linear inequality in the latter.

Example 1. *Suppose we have a simple multi-label classification problem where we are given as input images taken from an autonomous vehicle, and we have to identify whether a traffic light appears in the image, and whether it is green, yellow, or red. For each image, a standard neural network for this task will output a 4-dimensional vector where each element corresponds to one of the concepts of interest. Suppose, for example, the outputs of the neural networks are ordered in the following way:* `TrafficLight`, `Red`, `Yellow`, `Green`. *Then the file, containing the knowledge that a traffic light is always associated with one of the colors and that the colors associated with a traffic light are mutually exclusive, will have the format below:*

```
not y_0 or y_1 or y_2 or y_3
not y_0 or not y_1 or not y_2
not y_0 or not y_1 or not y_3
not y_0 or not y_2 or not y_3
```

Example 2. *Suppose we have a tabular data generation problem, where we have to generate a synthetic dataset for a clinical trial. Further, suppose that we have the following knowledge about the problem available: (i) the maximum hemoglobin recorded per patient should be always higher or equal than the minimum, and (ii) the maximum temperature recorded should be at least as high as the minimum. Then, assuming that the outputs of the neural network are ordered as* `MaxHemoglobin`, `MinHemoglobin`, `MaxTemp`, `MinTemp`, *the input file should have the following format:*

Listing 1: Correcting predictions with PiShield at inference time.

```python
1 from pishield.shield_layer import build_shield_layer
2
3 def correct_predictions(predictions,requirements_path):
4     num_variables = predictions.shape[-1]
5     shield_layer = build_shield_layer(num_variables,
          requirements_path)
6     corrected_predictions = shield_layer(predictions)
7     return corrected_predictions
```

Listing 2: Building a Shield Layer into a DDN to train with it.

```python
1 from pishield.shield_layer import build_shield_layer
2
3 class Shielded_DNN(torch.nn.Module):
4     def __init__(self, num_dim, requirements_path, ..):
5         self.model = torch.nn.Sequential(...)
6         self.shield_layer = build_shield_layer(num_dim,
               requirements_path)
7         ...
8
9     def forward(self, input):
10        output = self.model(input)
11        corrected_output = self.shield_layer(output)
12        return corrected_output
```

```
y_0 - y_1 >= 0
y_2 - y_3 >= 0
```

Usage. As it can be seen from both Listings 1 and 2, to build our layer, we only need (lines 5 and 6, respectively) to call the function `build_shield_layer`, which takes as input two parameters: (i) the dimension of the input to the layer, and (ii) the path to the requirements file. Once instantiated, the Shield Layer receives as input a tensor of predictions p (possibly violating the requirements) and returns a tensor $\hat{p}$ (of the same dimension as p), which is now guaranteed to be compliant with the requirements. PiShield benefits from an easy-to-use interface, and so, correcting p with a Shield Layer is a one-step operation consisting of a forward call of the Shield Layer on p. Listing 1, in particular, shows how to do all these steps at inference time. On the other hand, Listing 2 shows how to integrate the layer at training time for which the layer's forward call needs to be done before the backpropagation step. As shown in the Listing, the easiest way to match this condition is to instantiate the Shield Layer in the constructor of the DNN and then call the layer inside the usual forward method (as shown in line 11).

3 Example Scenarios

The need for a package like PiShield naturally raises in many application domains. Here, we discuss three example scenarios to which we applied the layers implemented in PiShield.

Functional Genomics. In functional genomics, the task is to predict a set of (hierarchical) functions that genes may possess [Vens *et al.*, 2008; Wehrmann *et al.*, 2018; Clare, 2003]. As such, it belongs to the broad category of Hierarchical Multi-label Classification (HMC) problems, which are multi-label classification problems whose labels are organized in a hierarchical structure. The hierarchy can be captured by propositional logic rules, where the atoms correspond to the labels. In addition to functional genomics,

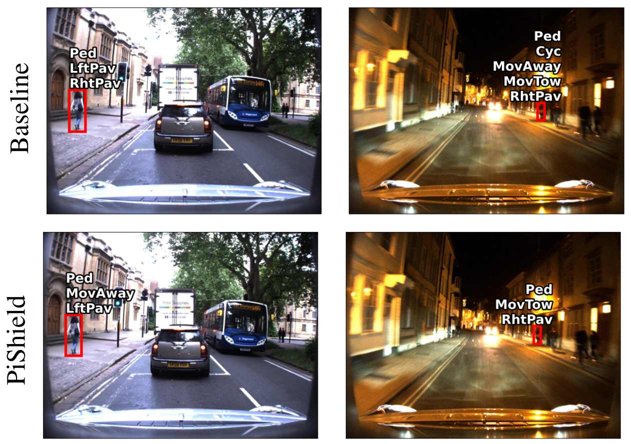

Figure 2: The unconstrained models (top row) violate simple background knowledge rules, predicting (i) that a pedestrian is both on the right and on the left pavement, and (ii) that a person is both a pedestrian and a cyclist, and is moving towards and away from the self-driving vehicle at the same time. On the other hand, the predictions made using PiShield (bottom row) are guaranteed to be compliant with the background knowledge.

HMC problems find applications in many real-world domains, such as document (see, e.g., [Klimt and Yang, 2004; Lewis *et al.*, 2004]) and image classification (see, e.g., [Deng *et al.*, 2009]), or medical diagnosis (see, e.g., [Dimitrovski *et al.*, 2008]). In our demo video[3], we show on a simple task that applying PiShield to a model for this task guarantees that the outputs preserve the hierarchical structure.

Road Events Detection. Generalizing the above problem, we have the task of assigning sets of labels to instances of various types, including images, documents etc. Given available domain knowledge, constraints are placed on the possible outputs, eliminating impossible (i.e., contradicting background knowledge) outcomes. In our recent work [Giunchiglia *et al.*, 2024], we considered the task of multi-label classification for an autonomous driving scenario using the ROAD-R dataset [Giunchiglia *et al.*, 2023b], the first real-world dataset for autonomous driving with manually annotated logical constraints. ROAD-R is build upon the road event detection dataset ROAD [Singh *et al.*, 2023] by adding 243 propositional logic requirements written in CNF. We showed that applying Shield Layers on the outputs is not only able to guarantee that these requirements are satisfied, but also results in increased performance, as seen in Table 2 of [Giunchiglia *et al.*, 2024]. Figure 2 shows examples of how using PiShield can impact the predictions' quality, which is intrinsically coupled with the safety guarantees.

Tabular Data Generation. Another application where we utilized PiShield is the task of synthesizing tabular data such that the outputs of a model for this task satisfies background knowledge rules provided as linear inequalities, which capture relations between the features of the tabular data. Applying Shield Layers to standard deep generative models (DGMs) resulted in more realistic outputs, which are compliant with the background knowledge. For example, consider the constraint *MaxHaemoglobin* $\geq$ *MinHaemoglobin*, which is a real constraint that we encountered in [Stoian *et al.*, 2024] in one of the datasets. In Figure 3, following the qualitative illustrations in Figures 3-6 of our previous work [Stoian *et al.*, 2024], we show how the constrained DGMs match the real data more closely in the distribution of their outputs. Additionally, through an extensive empirical analysis, we showed (in Table 2 of [Stoian *et al.*, 2024]) that integrating Shield Layers into DGMs yields an increased performance across two standard metrics used in tabular data generation.

[3]Video: http://tinyurl.com/pdv9eafa

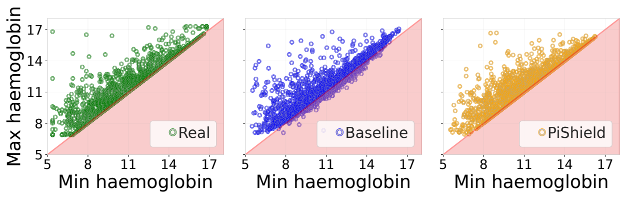

Figure 3: Real data (left) and samples generated by an unconstrained neural network (middle) and a neural network constrained with PiShield (right).

Scenario	Base.	PiShield
Functional genomics(AU($\overline{\text{PRC}}$))	0.225	**0.241**
Autonomous driving (f-mAP)	0.288	**0.303**
Tabular data generation (Utility-F1)	0.430	**0.458**

Table 1: Aggregated performance. The best results are in **bold**.

Performance. In Table 1, we compare baseline models with their constrained versions, which use PiShield during training. We report results for three real-world applications: (i) functional genomics, based on Table 3 of [Giunchiglia and Lukasiewicz, 2020], reporting the area under the average precision and recall curve (AU($\overline{\text{PRC}}$)) over 8 datasets, (ii) autonomous driving, based on Table 2 of [Giunchiglia *et al.*, 2024], reporting the frame-wise mean average precision (f-mAP) on the ROAD-R dataset, annotated with 243 propositional requirements; (iii) tabular data generation, based on Table 2 of [Stoian *et al.*, 2024], reporting the F1-score for the utility performance (Utility-F1) averaged over 5 deep generative model types and 5 different datasets, annotated with up to 31 linear inequality constraints. As we can see, using PiShield during training provides major performance improvements over the unconstrained baselines.

4 Conclusions

In this paper, we introduced PiShield, the first package that allows for injecting requirements into neural networks' topology by building Shield Layers, which correct the outputs so that they are guaranteed to satisfy the rules. PiShield can be applied (i) during training by building Shield Layers into the neural networks' architecture, or (ii) at inference time by correcting the neural networks' outputs as a post-processing step. We envision our package will be of use to practitioners working on real-world applications where domain knowledge can be expressed as propositional or linear constraints.

Ethical Statement

There are no ethical issues directly associating with the framework's specification and available methods. Any ethical issues would only arise from the context in which the framework is planned to be used and on the constraints provided by the users. For example, when generating tabular data, the users could create a set of constraints that would facilitate (partially) recovering specific records in the training data, thus posing problems in cases where the data generated in this way can be made publicly available, but the training data cannot.

Acknowledgments

Mihaela Cătălina Stoian is supported by the EPSRC under the grant EP/T517811/1. This work was also supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1, by the AXA Research Fund, by the EPSRC grant EP/R013667/1, and by the EU TAILOR grant. We would like to thank Salijona Dyrmishi, Thibault Simonetto, Andrew Ryzhikov, Salah Ghamizi, and Maxime Cordy for the useful discussions. We also acknowledge the use of the EPSRC-funded Tier 2 facility JADE (EP/P020275/1), GPU computing support by Scan Computers International Ltd.

References

[Ahmed *et al.*, 2022a] Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck, and Sameer Singh. Pylon: A PyTorch Framework for Learning with Constraints. In *Proceedings of Neural Information Processing Systems 2021 Competitions and Demonstrations Track*. PMLR, 2022.

[Ahmed *et al.*, 2022b] Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, and Antonio Vergari. Semantic probabilistic layers for neuro-symbolic learning. In *Proceedings of Neural Information Processing Systems*, 2022.

[Ahmed *et al.*, 2022c] Kareem Ahmed, Eric Wang, Kai-Wei Chang, and Guy Van den Broeck. Neuro-symbolic entropy regularization. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2022.

[Badreddine *et al.*, 2022] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303, 2022.

[Carraro, 2022] Tommaso Carraro. LTNtorch: PyTorch implementation of Logic Tensor Networks, 2022.

[Clare, 2003] Amanda Clare. *Machine Learning and Data Mining for Yeast Functional Genomics*. PhD thesis, University of Wales, 2003.

[d'Avila Garcez *et al.*, 2019] Artur S. d'Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *CoRR*, abs/1905.06088, 2019.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A large-scale hierarchical image database. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2009.

[Diligenti *et al.*, 2012] Michelangelo Diligenti, Marco Gori, Marco Maggini, and Leonardo Rigutini. Bridging logic and kernel machines. *Machine Learning*, 2012.

[Diligenti *et al.*, 2017] Michelangelo Diligenti, Soumali Roychowdhury, and Marco Gori. Integrating prior knowledge into deep learning. In *Proceedings of International Conference on Machine Learning and Applications*, 2017.

[Dimitrovski *et al.*, 2008] Ivica Dimitrovski, Dragi Kocev, Suzana Loskovska, and Sašo Džeroski. Hierchical annotation of medical images. In *Proc. of IS*. IJS Ljubljana, 2008.

[Donadello *et al.*, 2017] Ivan Donadello, Luciano Serafini, and Artur d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2017.

[Fischer *et al.*, 2019] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. DL2: Training and querying neural networks with logic. In *Proceedings of International Conference on Machine Learning*, 2019.

[Giunchiglia and Lukasiewicz, 2020] Eleonora Giunchiglia and Thomas Lukasiewicz. Coherent hierarchical multi-label classification networks. In *Proceedings of Neural Information Processing Systems*, 2020.

[Giunchiglia and Lukasiewicz, 2021] Eleonora Giunchiglia and Thomas Lukasiewicz. Multi-label classification neural networks with hard logical constraints. *Journal of Artificial Intelligence Research*, 72, 2021.

[Giunchiglia *et al.*, 2022] Eleonora Giunchiglia, Mihaela Catalina Stoian, and Thomas Lukasiewicz. Deep learning with logical constraints. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2022.

[Giunchiglia *et al.*, 2023a] Eleonora Giunchiglia, Fergus Imrie, Mihaela van der Schaar, and Thomas Lukasiewicz. Machine Learning with Requirements: a Manifesto. *CoRR*, abs/2304.03674, 2023.

[Giunchiglia *et al.*, 2023b] Eleonora Giunchiglia, Mihaela Cătălina Stoian, Salman Khan, Fabio Cuzzolin, and Thomas Lukasiewicz. ROAD-R: The autonomous driving dataset for learning with requirements. *Machine Learning Journal*, 2023.

[Giunchiglia *et al.*, 2024] Eleonora Giunchiglia, Alex Tatomir, Mihaela Catalina Stoian, and Thomas Lukasiewicz. CCN+: A neuro-symbolic framework for deep learning with requirements. *International Journal of Approximate Reasoning*, 2024. In press.

[Hoernle *et al.*, 2022] Nicholas Hoernle, Rafael-Michael Karampatsis, Vaishak Belle, and Kobi Gal. MultiplexNet:

Towards fully satisfied logical constraints in neural networks. In *Proceedings of Association for the Advancement of Artificial Intelligence*, 2022.

[Klimt and Yang, 2004] Bryan Klimt and Yiming Yang. The Enron Corpus: A new dataset for email classification research. In *Proceedings of European Conference on Machine Learning*, 2004.

[Lewis *et al.*, 2004] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5, 2004.

[Li *et al.*, 2023] Zenan Li, Zehua Liu, Yuan Yao, Jingwei Xu, Taolue Chen, Xiaoxing Ma, and Jian Lü. Learning with logical constraints but without shortcut satisfaction. In *Proceedings of International Conference on Learning Representations*, 2023.

[Manhaeve *et al.*, 2018] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Proceedings of Neural Information Processing Systems*, 2018.

[Nandwani *et al.*, 2019] Yatin Nandwani, Abhishek Pathak, Mausam, and Parag Singla. A primal dual formulation for deep learning with constraints. In *Proceedings of Neural Information Processing Systems*, 2019.

[Singh *et al.*, 2023] Gurkirt Singh, Stephen Akrigg, Manuele Di Maio, Valentina Fontana, Reza Javanmard Alitappeh, Salman Khan, Suman Saha, Kossar Jeddi Saravi, Farzad Yousefi, Jacob Culley, Tom Nicholson, Jordan Omokeowa, Stanislao Grazioso, Andrew Bradley, Giuseppe Di Gironimo, and Fabio Cuzzolin. ROAD: The ROad event Awareness Dataset for autonomous driving. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.

[Stoian *et al.*, 2023] Mihaela C. Stoian, Eleonora Giunchiglia, and Thomas Lukasiewicz. Exploiting T-norms for Deep Learning in Autonomous Driving. In *Proceedings of International Workshop on Neural-Symbolic Learning and Reasoning*, 2023.

[Stoian *et al.*, 2024] Mihaela C. Stoian, Salijona Dyrmishi, Maxime Cordy, Thomas Lukasiewicz, and Eleonora Giunchiglia. How Realistic Is Your Synthetic Data? Constraining Deep Generative Models for Tabular Data. In *Proceedings of International Conference on Learning Representations*, 2024.

[van Krieken *et al.*, 2023] Emile van Krieken, Thiviyan Thanapalasingam, Jakub M. Tomczak, Frank Van Harmelen, and Annette Ten Teije. A-neSI: A scalable approximate method for probabilistic neurosymbolic inference. In *Proceedings of Neural Information Processing Systems*, 2023.

[Vens *et al.*, 2008] Celine Vens, Jan Struyf, Leander Schietgat, Saso Dzeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Mach. Learn.*, 73(2), 2008.

[Wehrmann *et al.*, 2018] Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. Hierarchical multi-label classification networks. In *Proceedings of International Conference on Machine Learning*, 2018.

[Xu *et al.*, 2018] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of International Conference on Machine Learning*, 2018.