CEUR-WS.org/Vol-3432/paper19.pdf

Designing Logic Tensor Networks for Visual Sudoku puzzle classification

Lia Morra[1,*], Alberto Azzari[3], Letizia Bergamasco[1,2], Marco Braga[4], Luigi Capogrosso[3], Federico Delrio[1], Giuseppe Di Giacomo[1], Simone Eiraudo[1], Giorgia Ghione[1], Rocco Giudice[1], Alkis Koudounas[1], Luca Piano[1], Daniele Rege Cambrin[1], Matteo Risso[1], Marco Rondina[1], Alessandro Sebastien Russo[1], Marco Russo[1], Francesco Taioli[3], Lorenzo Vaiani[1] and Chiara Vercellino[1,2]

[1]*Politecnico di Torino, Torino, Italy*
[2]*LINKS Foundation, Torino, Italy*
[3]*University of Verona, Verona, Italy*
[4]*University of Milano-Bicocca, Milano, Italy*

Abstract

Given the increasing importance of the neurosymbolic (NeSy) approach in artificial intelligence, there is a growing interest in studying benchmarks specifically designed to emphasize the ability of AI systems to combine low-level representation learning with high-level symbolic reasoning. One such recent benchmark is Visual Sudoku Puzzle Classification, that combines visual perception with relational constraints. In this work, we investigate the application of Logic Tensork Networks (LTNs) to the Visual Sudoku Classification task and discuss various alternatives in terms of logical constraint formulation, integration with the perceptual module and training procedure.

Keywords

neuro–symbolic, logic tensor networks, visual reasoning, benchmarks

1. Introduction

In the past decade, deep learning (DL) has emerged as one of the most efficient ways to perform inductive tasks, showing an elevated accuracy in building models for different domains, such as computer vision, speech recognition, text understanding, etc. Purely data-driven strategies, however, are not without shortcomings. The most obvious limitation arises when the available data are not sufficient to build effective and appropriately generalizing models. Additionally, it is not possible to enforce compliance with limits imposed, for example, by natural laws, regulatory requirements, or safety regulations that are critical to reliable AI [1, 2, 3, 4].

To increase DL trustworthiness and generalizability, Neuro-Symbolic Artificial Intelligence (NeSy) techniques seek to combine the benefits of knowledge representation and reasoning with those of machine and deep learning [5, 6, 3]. Among such approaches, the Logic Tensor Network (LTN) paradigm combines deep neural networks with first-order logic knowledge representation.

NeSy 2023, 17th International Workshop on Neural-Symbolic Learning and Reasoning, Certosa di Pontignano, Siena, Italy

*Corresponding author.

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

Briefly, LTNs use an infinitely-valued fuzzy logical language called Real Logic as the underlying formalism, which consists of a first-order logic language whose signature consists of constant, function and predicate symbols. To apply the framework to real-world problems, where there is no complete certainty and formulas can be partially true, fuzzy semantics is adopted. In Real Logic, the word *grounding* is used to emphasize that symbols are concretely interpreted by tensors in the real field [7, 3].

This article aims to explore the application of LTNs to the recently proposed Visual Sudoku Puzzle Classification (ViSudo-PC) benchmark [8]. Briefly, given a Sudoku puzzle constructed from images as input, the classification task is to determine whether the puzzle is correctly solved, without access to the labels of individual digits. It was previously shown that the performance of a naive solution, that is classifying each digit using a Convolutional Neural Network (CNN) and then applying Sudoku rules to determine whether the solution is correct, rapidly degrades when the performance of the digit classifier is less than perfect [8]. Performing well on the ViSudo-PC benchmark thus requires systems that are able to reason about the perceptual information in the images as well as the additional information from Sudoku constraints. The implementation is available at https://github.com/MalumaDev/SymbolicSudoku.

2. Methodology

2.1. Puzzle construction

A Sudoku "puzzle" or "board" consists of 9×9 grid, in which each cell is populated with digits 1–9. A puzzle is correctly solved if no row, column, or non-overlapping 3×3 subgrid (or "square") contains all the possible numbers without repetitions. The ViSudo-PC benchmark generalizes the concept of Sudoku puzzles assuming that the board is a square of dimension $m = n \times n$ (specifically, 4×4 and 9×9 grids are provided), and that symbols can come from arbitrary domains. It includes four domains of increasing complexity: digits (MNIST), English letters (EMNIST), fashion items (FashionMNIST), and Japanese characters (KMNIST). In the following, we will refer for simplicity and without loss of generality to the symbols as *digits*.

2.2. Common definitions

In this section, definitions of all domains, variables, and predicates are provided. The function $\mathbf{D}(\cdot)$ associates each variable with the corresponding domain, and $\mathbf{D_{in}}(\cdot)$ each predicate with the input domain.

Domains:

- **images**, denoting the images that form the individual cells of the Sudoku puzzle, encoded as 28×28 pixel maps,
- **sudoku**, denoting the image representing the Sudoku puzzle,
- **results**, denoting the Boolean value that indicates the validity of the Sudoku board,
- **digits**, denoting the digits from 0 to 9,
- **coordinates**, denoting the coordinates of each cell in a Sudoku puzzle.

Variables:

- **x** ranging over the list of images $[x_{0,0}, ..., x_{i,j}, ..., x_{m,m}]$ associated to each Sudoku board, where $x_{i,j}$ with $i \in [0, m-1]$, $j \in [0, m-1]$ are the images at position (i, j), with $\mathbf{D}(x_{i,j}) =$ images;
- **d** ranging over the list of digits $[d_{0,0}, ..., d_{i,j}, ..., d_{m,m}]$ associated to each Sudoku board, where $d_{i,j}$, with $i \in [0, m]$, $j \in [0, m]$ represents the digit at position (i, j), with $\mathbf{D}(d_{i,j}) =$ digits;
- **se** ranging over all sub-elements of each Sudoku, that is rows, columns, and squares, each formed by a sequence of coordinates $\{i, j\}$;
- **S** for the $m \times m$ Sudoku puzzles in the data, with $\mathbf{D}(S) =$ sudoku;
- $\mathbf{S}_+$ and $\mathbf{S}_-$ for the correct and incorrect Sudoku puzzles, respectively;
- **l** for the labels, i.e., the validity of the Sudoku, $\mathbf{D}(l) =$ results.

Predicates:

- **digit**(x, d) is a digit classifier, where d is a term denoting a digit constant or a digit variable. The classifier should return the probability of an image x being of digit d, with $\mathbf{D_{in}}(\text{digit}) = \text{images, digits}$;
- **equal**$(x_{i,j}, x_{k,l})$ is a predicate indicating whether two images $x_{i,j}$ and $x_{k,l}$ contain the same digit, with $\mathbf{D_{in}}(\text{equal}) = \text{images, images}$;
- **valid**(S) denotes whether a Sudoku board S is valid. The predicate should return the probability that the image represents a valid Sudoku solution, with $\mathbf{D_{in}}(\text{valid}) = \text{sudoku}$;
- **validElement**(se) denotes whether a sub-element of a Sudoku board S is valid.

Fuzzy operators:

- **Diagonal quantification** $\text{Diag}(x, \ldots, l)$ quantifies over tuples that combine the i-th instance of each of the variables in the argument of Diag [3]. For instance, given a data set with samples x and target labels y, $\forall \text{Diag}(x, y)$ quantifies over each label, sample pair.

2.3. Knowledge base definition

At its core, an LTN-based NeSy approach to the ViSudo-PC task can be constructed by combining one or more CNNs, that recognize digits and/or classify the whole Sudoku board, with a LTN that enforces the logical constraints given by the rules of Sudoku. Multiple solutions are available depending on which predicates are defined and how they are combined in the knowledge base. The first group of solutions (denoted in the following as *indirect* solutions) verifies whether the detected digits constitute a valid Sudoku solution by means of a non-trainable predicate that enforces the rules of Sudoku. The network learns to detect the digits via a learnable digit(x, d) predicate. At inference time a fuzzy or crisp validity score can be computed applying the rules of Sudoku using real or standard logic. The second group of solutions (denoted in the following as *direct* solutions) computes instead the validity of the entire Sudoku board via the valid(S) predicate. An auxiliary digit(x, d) predicate is used to detect digits and enforce the rules of Sudoku. Both predicates *digit*(x, d) and *valid*(S) are grounded by CNNs. In all cases, we assume that labels for each individual digit are not available, and that digit classification must be trained in a semi-supervised fashion exclusively by enforcing the rules of Sudoku, as mandated by the ViSudo-PC task [8].

2.3.1. Indirect solution #1

This approach is based on the observation that every sub-element (row, column, and square) belonging to a correct Sudoku is also correct. Conversely, if all the sub-elements are correct, the Sudoku is correct as well. On the other hand, if the Sudoku is not correct, at least one wrong sub-element must exist. We further observe that a Sudoku sub-element is correct if, and only if, all available digits are present. Indeed, since the length of a sub-element is equal to the number of digits, if one is repeated twice, then another must be missing. Hence, the problem can be reduced to the simpler problem of detecting whether a sub-element is correct or not.

Axioms:

$$\forall \mathbf{se} \left(\left(\forall d \, \exists x_{i,j} \; : \; \{i,j\} \in \mathbf{se} \; (\text{digit}(x_{i,j}, d)) \right) \iff \text{validElement}(\mathbf{se}) \right) \tag{1}$$

$$\forall \text{Diag}(S,l) \left(l \iff (\forall \mathbf{se} \; : \; \mathbf{se} \in s \; \text{validElement}(\mathbf{se})) \right) \tag{2}$$

$$\forall \text{Diag}(S,l) \left(\neg l \iff (\exists \mathbf{se} \; : \; \mathbf{se} \in s \; \neg\text{validElement}(\mathbf{se})) \right) \tag{3}$$

Grounding:

The digit predicate is grounded by a CNN taking as input each digit image. The validElement predicate is a non-trainable predicate that can be computed directly using the axiom in Eq. 1.

2.3.2. Indirect solution #2

This solution is based on the observation that a given sub-element (row, column, or square) cannot contain the same symbol twice. More generally, given all possible image pairs within a Sudoku board, it is possible to define an additional predicate sameSubelement($[p_1, p_2]$) that determines whether the image pair at positions $p_1 = (i, j)$ and $p_2 = (l, k)]$ belong to the same sub-element and hence cannot contain the same digit (or, more generally, the same symbol). Based on this principle, two axioms can be formulated, one for correct Sudoku boards and one for incorrect Sudoku boards. In the experiments, only correct examples were used.

Axioms:

$$\forall S_+ \; \forall([p_1, p_2]) \left(\text{sameSubelement}(p_1, p_2) \implies \neg\text{equal}(x_{p_1}, x_{p_2}) \right) \tag{4}$$

$$\forall S_- \; \exists([p_1, p_2]) \left(\text{sameSubelement}(p_1, p_2) \implies \text{equal}(x_{p_1}, x_{p_2}) \right) \tag{5}$$

Grounding:

The equal(x_1, x_2) predicate is grounded by a function of the probabilities that x_1 and x_2 belong to each digit, computed by the digit(x) predicate::

$$\text{equal}(x_1, x_2) = exp\Big(-ReLU(\|\text{digit}(x_1) - \text{digit}(x_2)\| - c)\Big) \tag{6}$$

where c is a constant added for numerical stability. Alternative grounding formulations could compute the distance i) based only on the most probable digits or ii) directly from the image embedding, e.g., before computing the softmax.

2.3.3. Indirect solution #3

This solution stems from the observation that the same number can appear only once for each sub-element (row, column, and square). Hence, if we denote as $x_0, ..., x_k, ..., x_n$ the list of images in sub-element *se*, the following knowledge base can be defined:

Axioms:

$$\forall S_+ \ \forall \mathbf{se} \ \forall x_1, x_2, d \ : \ (x1 \in \mathbf{se} \wedge x2 \in \mathbf{se}) \Big(x_1 \neq x_2 \wedge \text{digit}(x_1, d) \implies \neg\text{digit}(x_2, d)\Big) \tag{7}$$

$$\forall S_- \ \forall \mathbf{se} \ \exists x_1, x_2, d \ : \ (x1 \in \mathbf{se} \wedge x2 \in \mathbf{se}) \Big(x_1 \neq x_2 \wedge \text{digit}(x_1, d) \implies \text{digit}(x_2, d)\Big) \tag{8}$$

2.3.4. Direct solution

The direct solutions involve two predicates, valid to compute the validity of the Sudoku board and digit to compute the probability that each cell contains a given digit. The latter predicate is used only to enforce Sudoku rules. At inference time, predictions are directly calculated from the valid predicate. While the digit predicate is trained in a semi-supervised fashion, to comply with the training setting specified by the ViSudo-PC benchmark, the valid predicate can be trained in a supervised fashion. In addition, additional axioms are specified to encode prior knowledge about the rules of Sudoku. Any of the axioms that were defined in previous solutions can be used for this purpose: only one possible solution is shown here.

Axioms:

$$\forall S_+ \text{valid}(S), \forall S_- \neg\text{valid}(S) \tag{9}$$

$$\forall S \left(\left(\forall([p_1, p_2])(\text{sameSubelement}(p_1, p_2) \implies \neg\text{equal}(x_{p_1}, x_{p_2}))\right) \Leftrightarrow \ \text{valid}(S)\right) \tag{10}$$

$$\forall S \left(\left(\exists([p_1, p_2])(\text{sameSubelement}(p_1, p_2) \implies \text{equal}(x_{p_1}, x_{p_2}))\right) \Leftrightarrow \ \neg\text{valid}(S)\right) \tag{11}$$

Grounding: The equal(x_1, x_2) predicate is grounded as in Eq. 6. The two predicates valid(S) and digit(x, d) are grounded by either two separate CNNs or by one CNNs with a common backbone and two different prediction heads.

2.3.5. Potential pitfalls

Alternative solutions could be designed by extending the LTN for the semi-supervised MNIST classification task proposed in [3], in which MNIST classification is learned by solving single- and multi-digit additions: $\forall Diag(x_1, x_2, y_1, y_2, n)\ (\exists d_1, d_2, d_3, d_4 : 10d_1 + d_2 + 10d_3 + d_4 = n.$ $(digit(x_1, d_1) \wedge digit(x_2, d_2) \wedge digit(y_1, d_3) \wedge digit(y_2, d_4)))$ A similar approach, in this case, would yield the following solution: $\forall Diag(x_{0,0}, ..., x_{i,j}, ..., , x_{m,m}, l)\ (\exists d_{0,0}, ..., d_{i,j}, ..., , d_{m,m}$: validPuzzle$(d_{0,0}, ..., d_{i,j}, ..., , d_{m,m}) = l.(digit(x_{0,0}, d_{0,0}) \wedge ... \wedge digit(x_{i,j}, d_{i,j}) \wedge ... \wedge digit(x_{m,m}, d_{m,m})))$, where validPuzzle determines if the specific sequence of digits yields a Sudoku board consistent with the label (i.e., valid or invalid). Empirically, we found that solutions involving up to $m \times m$ multiple $\wedge$ operations in a single axiom led to exploding memory issues, possibly due to the LTNtorch implementation [9]. The proposed solutions are based on simpler formulas, each implementing a separate logical constraint for pairs of digits at a time, aggregated using existential and universal quantifiers. However, given a $m \times m$ board, with m rows, m columns, and $\sqrt{m}$ squares, the number of possible digit pairs with each sub-element is $\binom{m}{2} = \frac{N(N-1)}{2}$, hence the number of constraints increases as m^3.

2.3.6. Extension to digit localization

A near-perfect accuracy in digit classification is paramount to determining whether a given Sudoku puzzle is correctly solved. The solutions proposed in the previous section rely heavily on the $digit(x, d)$ and $equal(x_{p_1}, x_{p_2})$ predicates, and assume that the image grid is known. This limitation could be overcome by integrating an Object Detection method within the NeSy framework. In this way, the grounding of the sub-images can be extended by including the bounding box coordinates, which would be used to localize the digits within the board. The digit(S, b, d) predicate would take as input the Sudoku Board S and a bounding box b to predict the probability that the bounding box b contains the digit d, and would be grounded by an object detector. In a preliminary implementation, we used the You Only Look Once (YOLO) [10] algorithm, specifically the YOLOv1 architecture. YOLOv1 consists of two main parts: *i)* the feature extractor and the *ii)* detection network. The latter is a fully connected layer that, from the output of the feature extraction, generates a set of bounding boxes that contain the detected objects in the image. The feasibility of training an object detector and a LTN end-to-end has been established in [11].

3. Exploratory assessment

3.1. Dataset

Preliminary experiments were conducted on the basic dataset provided by the ViSudo-PC benchmark, which includes 10 splits per dataset to be used for scoring [8][1]. We performed experiments on the 4×4 Sudoku with data sources MNIST, EMNIST, FMNIST, and KMNIST. Each split contains 50/100/100 puzzles for training/validation/test, respectively. Performances (Area under the ROC curve) are reported in cross-validation by averaging across the 10 splits.

[1]The dataset was downloaded from https://github.com/linqs/visual-sudoku-puzzle-classification

3.2. Experimental settings

A total of three configurations were tested, three variants of the indirect solution #2 introduced in Section 2.3.2, denoted in the following as LTN_IND_A, LTN_IND_B and LTN_IND_C. Preliminary experiments were also conducted on the direct solution (LTN_DIR). The indirect solution was tested with and without negative examples (Eq. 11) in the knowledge base. At inference time, the probability of a correct Sudoku is computed using the same axiom for positive example (Eq. 10) for the indirect solutions, whereas for the direct solution it is directly computed by the valid predicate.

The digit CNN consists of 2 convolutional layers with a kernel size of 5, each followed by a max pooling layer of size 2 with stride 2. The latter feeds into fully connected (FC) dense layers of sizes 320, 50 with ReLU activation and a final softmax layer. For the direct solution, two output branches of sizes 320, 50 and 5120, 320 computes the Valid and digit predicate, respectively.

We approximate the universal quantifier with the generalized mean w.r.t. the error aggregator $A_{pME} = 1 - \left(\frac{1}{n}\sum_{i=1}^{n}(1-a_i)^{p_\forall}\right)^{\frac{1}{p_\forall}}$ and the existential quantifier with the generalized mean aggregator $A_{pM} = \left(\frac{1}{n}\sum_{i=1}^{n} a_i^{p_\exists}\right)^{\frac{1}{p_\exists}}$, as suggested by Badreddine et al. [3]. We set $p_\exists = 1.5$, whereas for the universal quantifier both fixed $p_\forall = 2$ (LTN_IND_A and LTN_IND_C) and scheduled values (LTN_IND_B, LTN_DIR) were evaluated. In the latter, $p_\forall$ is gradually increased as follows: 1 (epochs 0–20), 2 (20–120), 6 (120–170), 8 (170–200), and 10 (200–500). Higher p values at the beginning of training prevented the network from converging, as previously reported [3]. For inference $p_\forall$ was set to 1000. All networks were trained with batch size 8 and the Adam optimizer. Learning rate and number of epochs were set experimentally for each configuration (LTN_IND_A: 0.001/200; LTN_IND_B: 0.0005/300; LTN_IND_C: 0.0005/500; LTN_DIR:0.0005/300). The LTN was implemented in PyTorch using the LTNtorch package [9].

3.3. Results

Results for the indirect solutions are presented in Table 1. LTN–based solutions perform best for the simpler dataset (MNIST). As observed in [3], LTNs appear sensitive to random initialization and may occasionally fail to converge to a non-trivial solution, which explains high standard deviations for most configurations. Compared to the baseline LTN (LTN_IND_A), stability and performance are enhanced by scheduling p in the universal aggregator [3], as well as by including axioms for negative examples (LTN_IND_C). Compared to NeuPSL [8], LTN–based solution appear to perform better on the KMNIST domain, and worse on the EMNIST domain; it is possible that, when samples are more difficult to classify, predicting whether two digits are equal is a more robust alternative than comparing the actual classifications, and thus the different may be also attributed to the choice of knowledge base.

The direct solution does not achieve performance above random guess on MNIST4 (0.52 ± 0.02) and was not tested on other domains. However, When computing predictions using the digit predicate and rules of Sudoku, performance increases (0.85 ± 0.02). Hence, the LTN learns to distinguish digits, but fails to predict the validity directly from the input image, possibly due

Data source	NeuPSL [8]	LTN_IND_A	LTN_IND_B	LTN_IND_C
MNIST4	0.88 ± 0.02	0.83 ± 0.18	0.84 ± 0.14	0.94 ± 0.10
EMNIST4	0.79 ± 0.09	0.58 ± 0.04	0.58 ± 0.06	0.65 ± 0.14
FMNIST4	0.74 ± 0.04	0.67 ± 0.11	0.76 ± 0.15	0.83 ± 0.11
KMNIST4	0.65± 0.12	0.83 ± 0.09	0.85 ± 0.11	0.87 ± 0.09

Table 1
Performance of the indirect LTN solution on the 4×4 Sudoku. The mean area under the receiver operating characteristic curve (AuROC) along with the standard deviation over 10 splits is reported. The tested LTN versions differ as follows: LTN_IND_A: positive examples only and fixed $p_\forall$;LTN_IND_B: positive examples only and varying $p_\forall$; LTN_IND_C: all axioms and fixed $p_\forall$.

to the specific choice of grounding.

4. Visual Sudoku as a teaching tool

Fostering research in Neuro-symbolic Artificial Intelligence requires training a new generation of scientists on NeSy approaches and related topics. The solutions described in this paper were developed during a graduate level course in Neuro-Symbolic Artificial Intelligence designed primarily for a PhD programme in Computer Engineering. Further details on the course organization are given in Appendix B. Project-based and task-based learning is an integral part of many machine learning and deep learning teaching curricula [12, 13]. Benchmarks such as ViSudo-PC provide an approachable yet challenging learning experience as they allow newcomers to explore two related, yet complementary aspects of NeSy approaches: (i) how to reformulate the learning setting and define axiomatic prior knowledge, and (ii) how different grounding choices may affect training.

A few practical hurdles could be tackled in future versions of the benchmark. The standardized split provides a very challenging benchmark by construction, as domains are challenging, few training samples are provided, and the semi-supervised setting assumes that labels for the digits are not available. Many configurations resulted in severe overfitting, and the difficulty of the optimization problem distracts from the problem formulation. Much higher performance can be obtained in simpler settings in which, e.g., labels were provided for some digits, or the learning task was reduced to learning the correctness of a single row/column. Providing standardized settings of low and medium difficulty would facilitate developing new solutions for this task, and extending beyond the 4×4 board.

5. Conclusions

In this paper, multiple LTN-based solutions to the ViSudo-PC benchmark were discussed. While experimental validation is still preliminary, we observe that several axioms could be used to encode the rule of Sudoku. Although logically equivalent, different solutions could result in different numerical properties, hindering comparison across different implementations and scenarios. Further experiments are needed to fully characterize all possible LTN-based solutions, as well as to optimize their grounding and increase stability to random initialization [3].

References

[1] M. Brundage, S. Avin, J. Wang, H. Belfield, G. Krueger, G. Hadfield, H. Khlaaf, J. Yang, H. Toner, R. Fong, et al., Toward trustworthy AI development: mechanisms for supporting verifiable claims, arXiv preprint arXiv:2004.07213 (2020).

[2] L. Von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, et al., Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems, IEEE Transactions on Knowledge and Data Engineering 35 (2021) 614–633.

[3] S. Badreddine, A. d'Avila Garcez, L. Serafini, M. Spranger, Logic tensor networks, Artificial Intelligence 303 (2022) 103649.

[4] E. Giunchiglia, M. C. Stoian, T. Lukasiewicz, Deep learning with logical constraints, in: IJCAI, 2022.

[5] L. De Raedt, S. Dumančić, R. Manhaeve, G. Marra, From statistical relational to neural-symbolic artificial intelligence, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 4943–4950.

[6] P. Hitzler, Neuro-symbolic artificial intelligence: The state of the art, IOS Press, 2022.

[7] L. Serafini, A. S. d'Avila Garcez, Logic tensor networks: Deep learning and logical reasoning from data and knowledge, ArXiv abs/1606.04422 (2016).

[8] E. Augustine, C. Pryor, C. Dickens, J. Pujara, W. Y. Wang, L. Getoor, Visual sudoku puzzle classification: A suite of collective neuro-symbolic tasks, in: International Workshop on Neural-Symbolic Learning and Reasoning (NeSy), Windsor, United Kingdom, 2022.

[9] T. Carraro, LTNtorch: PyTorch implementation of Logic Tensor Networks, 2022. URL: https://doi.org/10.5281/zenodo.6394282. doi:10.5281/zenodo.6394282.

[10] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[11] F. Manigrasso, F. D. Miro, L. Morra, F. Lamberti, Faster-LTN: a neuro-symbolic, end-to-end object detection architecture, in: Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II 30, Springer, 2021, pp. 40–52.

[12] T. Elstner, F. Loebe, Y. Ajjour, C. Akiki, A. Bondarenko, M. Fröbe, L. Gienapp, N. Kolyada, J. Mohr, S. Sandfuchs, et al., Shared tasks as tutorials: A methodical approach, in: 37th AAAI Conference on Artificial Intelligence (AAAI 2023). AAAI, 2023.

[13] S. Raschka, Deeper learning by doing: Integrating hands-on research projects into a machine learning course, in: Proceedings of the Second Teaching Machine Learning and Artificial Intelligence Workshop, PMLR, 2022, pp. 46–50.

Appendix

A. Visual Sudoku as a teaching tool: context information

The solutions described in this paper were developed during a graduate level course in Neuro-Symbolic Artificial Intelligence designed primarily for the PhD programme in Computer Engineering at Politecnico di Torino. The course was structured in frontal lessons (12 hours) followed by an introductory laboratory on Logic Tensor Networks (4 hours) loosely based on the tutorials and examples available in [9]. Students, divided into small groups of at most 3 members, were then invited to work on the ViSudo-PC benchmark as a case study and, optionally, provide an implementation, which was then reviewed, compared, and combined as appropriate.

Approximately 40 PhD candidates initially enrolled in the course. Based on a pre-course survey, most of the students had previous training in deep learning (30/34) or had already published in the field (18/34). On the other hand, few had prior training or research experience in first-order logic languages or similar topics (6/34). This distribution reflects the shift towards deep and machine learning in many engineering and data science-oriented undergraduate curricula.