Deep Symbolic Learning: Discovering Symbols and Rules from Perceptions*

Alessandro Daniele[1], Tommaso Campari[1], Sagar Malhotra[1,2] and Luciano Serafini[1]

[1]Fondazione Bruno Kessler (FBK), Trento, Italy.
[2]Research Unit of Machine Learning, TU Wien, Vienna, Austria.

Neuro-Symbolic (NeSy) integration combines symbolic reasoning with Neural Networks (NNs) for tasks requiring both perception and reasoning. Most NeSy systems rely on continuous relaxation of logical knowledge, and they assume the symbolic rules to be given. We propose *Deep Symbolic Learning* (DSL) [1], a NeSy system that simultaneously learns the perception and symbolic functions while being trained only on their composition. The key idea is to adapt reinforcement learning policies to the NeSy context: given the NN predictions $\mathbf{t}$ with $\sum_i t_i = 1$, we use the greedy policy $\pi(\mathbf{t}) = argmax_i t_i$ to select a single discrete symbol, and the function $\mu(\mathbf{t}) = max_i t_i$ to select the corresponding value in $\mathbf{t}$, interpreted as a truth value under a fuzzy logic semantics. When performing multiple discrete choices within the model, each truth value is sent to an aggregation operator, which returns the truth value of the final output.

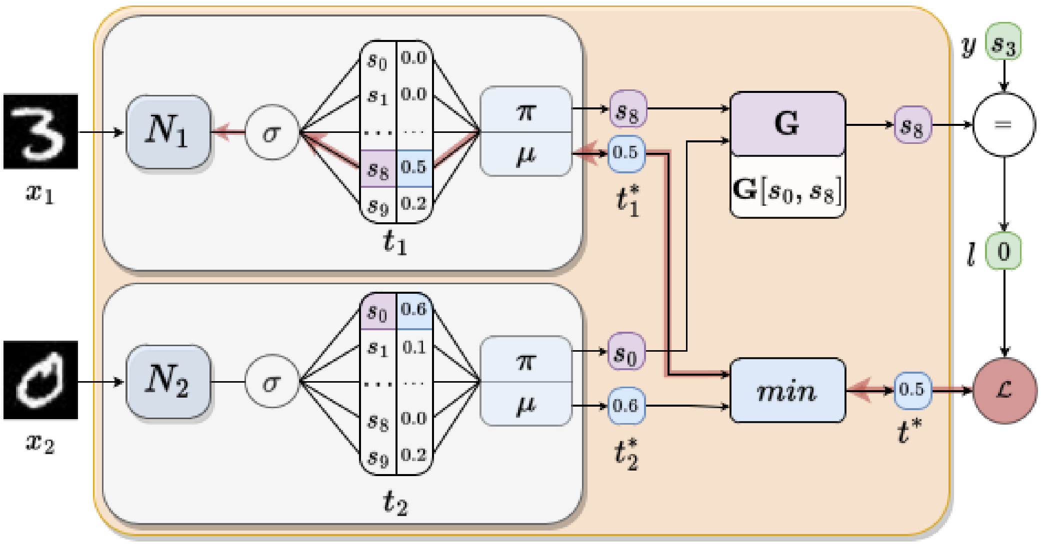

Figure 1: DSL architecture

NeSy2023, 17th international workshop on neural-symbolic learning and reasoning, Certosa di Pontignano, Siena, Italy
daniele@fbk.eu (A. Daniele)

In Fig. 1, DSL architecture for the MNIST-Addition task [2] is provided. Two NNs, N_1 and N_2, are used to classify the images, and the softmax function σ converts their predictions into fuzzy truth values $\mathbf{t_1}$ and $\mathbf{t_2}$. The greedy policy function π, takes in $\mathbf{t_1}$ and $\mathbf{t_2}$, and returns discrete symbolic predictions (s_8 and s_0, resp.). Their corresponding fuzzy truth values ($t_1^* = 0.5$ and $t_2^* = 0.6$ resp.) are given by the max operator μ. These symbols (s_8 and s_0) are then passed to the symbolic function, which is represented as a lookup table $\mathbf{G}$, returning the final output. The confidence in the final output is given by the confidence in the predicted symbols being simultaneously correct, using the Gödel semantics of conjunction, i.e., $\mathbf{t}^* = min(\mathbf{t_1^*}, \mathbf{t_2^*})$. In Fig. 1, the NNs predict symbols s_8 and s_0, while the function defined by $\mathbf{G}$ corresponds to the sum. As a consequence, the final output of DSL is s_8 $(8 + 0 = 8)$. The framework considers the correctness of the final output, producing a label $l = 1$ if the prediction is correct and $l = 0$ otherwise. Such a label is given as supervision for $\mathbf{t}^*$. In the example, the prediction is wrong since the first digit has been classified as s_8 instead of s_3. Since the min function admits only one non-zero partial derivative (corresponding to the minimum value), the back-propagation changes the weights of a single network (N_1 in Fig. 1). If the prediction is wrong, the effect of this change is to reduce the truth value of the currently selected symbol (s_8), increasing the chances of choosing a different symbol in the next iteration. Finally, DSL can also learn the table $\mathbf{G}$ from the data by applying π and μ to a learnable tensor $\mathbf{W}$.
In Tab. 1 we report the accuracy on the MNIST MultiDigitSum (MDS). For further experiments/analysis, see [1].

	2	4	15
NAP [3]	93.9 ± 0.7	T/O	T/O
DPL [2]	95.2 ± 1.7	T/O	T/O
DStL[4]	96.4 ± 0.1	92.7 ± 0.6	T/O
DSL	95.0 ± 0.7	88.9 ± 0.5	64.1 ± 1.5

Table 1: Accuracy obtained on the MNIST MDS task. T/O stands for timeout. **NB:** DSL is the only method which learns the knowledge, the other methods assume it to be given.

Acknowledgements TC and LS acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU.

References

[1] A. Daniele, T. Campari, S. Malhotra, L. Serafini, Deep symbolic learning: Discovering symbols and rules from perceptions, arXiv preprint arXiv:2208.11561 (Accepted at IJCAI2023) (2023).
[2] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. De Raedt, Deepproblog: Neural probabilistic logic programming, NIPS (2018).
[3] Z. Yang, A. Ishay, J. Lee, Neurasp: Embracing neural networks into answer set programming, in: IJCAI, 2020.
[4] T. Winters, G. Marra, R. Manhaeve, L. De Raedt, Deepstochlog: Neural stochastic logic programming, in: AAAI, 2022.