Introducing DeepEL (Extended Abstract)

Alessandro Longato[1], Ignacio Huitzil[2] and Rafael Peñaloza[1]

[1]University of Milano-Bicocca, Milan, Italy
[2]University of Zaragoza, Aragon Institute of Engineering Research (I3A), Zaragoza, Spain

1. Introduction

Description logics (DLs) [1] have been successfully used to model and reason about many different knowledge domains. They provide formal guarantees for the reasoning conclusions, under the assumption that the knowledge is modelled correctly. While this assumption can be considered to hold for terminological knowledge expressing the relations between the used symbols, it is less clear with assertions, where the properties of individuals may depend on imperfect perceptions. For instance, a diagnosis for jaundice depends on an often subtle perception of the skin colour of a patient.

Neural systems have been shown to excel at perception-based classification (see [2] for one of many surveys), but need to be trained for a specific selection of classes, and are essentially static after construction; that is, when knowledge changes, even minimally, they need to be trained anew.

We present DeepEL, a neurosymbolic extension of the DL $\mathcal{EL}^\bot$ [3] which allows for provably correct reasoning modulo a perception uncertainty, while being able to robustly handle noisy input signals, or properties that cannot be formally defined but only grasped from data. The formalism presents a fully integrated neuro-symbolic view, where non-observable properties can be learned and deduced with the help of terminological knowledge, and where different perceptions may complement each other for more robust answers.

This paper is an abridged version of a paper accepted to KR 2026 [4]. Due to lack of space, here we focus on presenting the main language and the tasks that can be solved with the formalism, and show how to implement them with the help of a DeepProbLog implementation. We assume a basic understanding of DeepProbLog [5] and the underlying Prolog [6]. Full details and experimental results are available at the full version [4].

2. DeepEL

DeepEL is a neuro-symbolic variant of $\mathcal{EL}^\bot$ tailored towards the use of domain knowledge to complement neural perceptions of the properties of individuals. The basic idea is that general terminological knowledge (i.e., a TBox) has been formalised, but needs to be applied to understand the properties of objects perceived through fallible sub-symbolic (neural) classifiers. This allows us to extend the capabilities of the neural network without fine-tuning or retraining, but also to increase the confidence in the results in some cases through probabilistic reasoning and other related computations.

For example, consider a neural classifier trained to identify digits from hand-written documents. Suppose that for a given application we are only interested in the parity of the digit (whether it is even or odd). A standard neural approach would require a new training phase to include these possibilities. A TBox, instead, can define even and odd digits—e.g., through the GCIs Zero $\sqsubseteq$ Even, One $\sqsubseteq$ Odd, …, Nine $\sqsubseteq$ Odd—and reuse the classifier without modification. Assume that the neural classifier produces a

DL 2026: 39th International Workshop on Description Logics, July 17–19, 2026, Lisbon, Portugal
alessandro.longato01@universitadipavia.it (A. Longato); ihuitzil@unizar.es (I. Huitzil); rafael.penaloza@unimib.it (R. Peñaloza)
https://rpenalozan.github.io (R. Peñaloza)
0000-0002-7581-0345 (I. Huitzil); 0000-0002-2693-5790 (R. Peñaloza)

probability distribution expressing the uncertainty of a given input corresponding to each possible digit. Upon an input, the classifier returns that the image contains a One with probability 0.35; a Four with probability 0.2; and a Seven with probability 0.45 (all other digits have probability 0 in this example). The classifier is fairly uncertain about the specific digit it has read, but we have a good reason to believe (with probability 0.8) that it is an Odd number.

Similarly, through a different TBox which defines Small (between 0 and 4) and Large (between 5 and 9) digits, one could deduce that the input image contains a small digit with probability 0.55. Putting this knowledge together, we can conclude that the probability of it being a small odd number is 0.35. Note that (i) the classifier output is used without modification, but manipulated through the TBox and (ii) the probability of being a small odd number is not the product of the individual probabilities ($0.35 \neq 0.8 \cdot 0.55$), as these are not independent properties. This simple example motivates the formal architecture of DeepEL introduced next.

In the following, we assume a fixed TBox $\mathcal{T}$ and a fixed classical ABox $\mathcal{A}$, and consider an ABox which is populated through one or more neural classifiers, as follows. Given a multi-class classifier Γ over the classes $A_1^\Gamma, \ldots, A_{n_\Gamma}^\Gamma$, the *neural ABox* of Γ over input α is $\mathcal{A}_\alpha^\Gamma := \{p_1 :: A_1^\Gamma(\alpha), \ldots, p_{n_\Gamma} :: A_{n_\Gamma}^\Gamma(\alpha)\}$, where p_i is the confidence level of Γ for the class A_i^Γ on input α, henceforth denoted as $P(A_i^\Gamma(\alpha))$. In our running example, if Γ is the digit classifier and α is the input image, we get[1]

$$\mathcal{A}_\alpha^\Gamma = \{0.35{::}\mathsf{One}(\alpha), 0.2{::}\mathsf{Four}(\alpha), 0.45{::}\mathsf{Seven}(\alpha)\} \cup \{0 :: A(\alpha) \mid A \notin \{\mathsf{One}, \mathsf{Four}, \mathsf{Seven}\}\}.$$

The information about the (perceived) objects is populated by a family of neural classifiers, each processing some objects. The *global neural ABox* is the union of all the neural ABoxes obtained through this processing. The *neural KB* is the union of the (classical) TBox $\mathcal{T}$ and ABox $\mathcal{A}$, and the global neural ABox. What remains is to explain the semantics of these expressions.

The semantics of neural KBs follows a multiple-world approach. The output of one classifier Γ over one input defines n_Γ worlds, which express the different cases that may arise depending on the actual class of α. Informally, the classifier is not certain about the class to which α belongs, but assigns a probability to each possibility, which corresponds to the probability of the associated world. For simplicity, we assume that all classifications made (either by different neural classifiers or by the same classifier over different inputs) are probabilistically independent. Hence, the set of worlds is the Cartesian product of all possible outputs on all the classifiers used, and the probability of each of these worlds is the product of the probabilities of the classes it contains. More formally, we construct the joint distribution over all classifier instantiations under the assumption of independence of models and instantiations. The classical part of the KB is assumed to hold in all worlds.

Formally, let Ω be the class of all possible outcomes in this probability distribution P (i.e., all the worlds in the Cartesian product). For each $\omega \in \Omega$, $\mathcal{K}_\omega$ is the KB containing $\mathcal{T}$, $\mathcal{A}$, and the assertions associated to the outcome ω. Each $\mathcal{K}_\omega$ is a classical $\mathcal{EL}^\bot$ KB. Its probability is $P(\omega) = \prod_{A_i^\Gamma(\alpha) \in \omega} P(A_i^\Gamma(\alpha))$

For our example, consider an additional input β, connected to α via the assertion $\mathsf{same\_parity}(\alpha, \beta)$, stating that α and β have the same parity (e.g. through the GCIs $\exists \mathsf{same\_parity}.\mathsf{Even} \sqsubseteq \mathsf{Even}$, $\exists \mathsf{same\_parity}.\mathsf{Odd} \sqsubseteq \mathsf{Odd}$) and let $\mathcal{A}_\beta^\Gamma = \{0.6 :: \mathsf{Eight}(\beta), 0.4 :: \mathsf{Three}(\beta)\}$. The neural KB obtained this way defines six worlds with positive probability, depending on the value that the classifier assigns to the two inputs (see Table 1).[2]

Since the different classes assigned by each neural classifier are disjoint (by assumption), the worlds represent a discrete probability distribution over the possible combined outcomes. As standard in multiple-world semantics, the probability of a consequence is the sum of the probabilities of the worlds that entail this consequence. In our example, the probabilities of α and β being a small odd numbers are 0.35 (the sum of the two worlds in the first column) and 0.4 (the sum of the three worlds in the second row), respectively. However, our knowledge allows us to derive, in some worlds, that the input α is Odd and Even at the same time (e.g., when $\mathsf{Four}(\alpha)$ and $\mathsf{Three}(\beta)$ hold), which is an undesired behaviour. This can be fixed by adding the GCI $\mathsf{Even} \sqcap \mathsf{Odd} \sqsubseteq \bot$ expressing disjointness of the two classes, making

[1] For brevity, we disregard all expressions of the form $0 :: A(a)$. The reason should be clear when the semantics is introduced.
[2] Formally, there would be 100 worlds, but all others have probability 0, and we ignore them here.

Table 1
Six worlds and their probability from the running example. All worlds are assumed to include the classical axioms; only the neural assertions are depicted.

	$\mathsf{One}(\alpha)$	$\mathsf{Four}(\alpha)$	$\mathsf{Seven}(\alpha)$
$\mathsf{Eight}(\beta)$	$0.21 = 0.6 \cdot 0.35$	0.12	0.27
$\mathsf{Three}(\beta)$	0.14	0.08	0.18

some worlds (those where α or β was simultaneously odd and even) inconsistent—describing situations which are impossible given our knowledge. As such, they are assigned probability 0. To preserve a probability distribution, we proportionally scale up the probabilities of all remaining (consistent) worlds.

Let $\Omega_\bot$ be the set of all inconsistent worlds and $P(\Omega_\bot) := \sum_{\omega \in \Omega_\bot} P(\omega)$ where P is the original probability distribution. We construct a new distribution

$$P_\bot(\omega) := \begin{cases} 0 & \omega \in \Omega_\bot \\ \frac{P(\omega)}{1-P(\Omega_\bot)} & o.w. \end{cases}$$

equivalent to the probability distribution conditioned on the event of having a consistent world. We define the *probability of the assertion* $A(a)$ w.r.t. the neural KB $\mathcal{K}$ as $P(A(a)) := \sum_{\mathcal{K}_\omega \models A(a)} P_\bot(\omega)$. That is, the probability of an assertion is the sum of the probabilities of all consistent worlds that entail it. Computing this value is the natural probabilistic variant of the problem of instance checking.

In our example, we see that three of the six worlds from Table 1 are inconsistent, and $P(\Omega_\bot) = 0.56$. Hence, the probability of α (or β) being a small odd number is now approximately 0.32. Importantly, the knowledge about the shared parity of the objects affects the resulting probability (as it should be): the perceptions that contradict this knowledge are discarded as erroneous. In fact, the KB now entails that the probability of α being Odd is now approximately 0.73 (down from 0.8), given the evidence that β is more likely to be Even than Odd. In other words, knowledge allows the system to perform a *double-check* and update its probabilistic confidence on the observations *on-the-fly*, without any further fine-tuning or processing of the classifier.

3. Implementing a Reasoner

Rather than implementing a DeepEL reasoner from scratch, we follow the strategy of transforming the ontology into a logic program, and using the efficient implementations available for that formalism, as done e.g., in [7, 8, 9, 10]. Specifically, we use OWL2ASP [11], to translate $\mathcal{EL}^\bot$ KBs into sets of facts corresponding to GCIs in normal form and simple assertions (see Table 2).[3] The reasoning steps of the completion algorithm are simulated by the Prolog rules from Table 3, where `inst(x,a)` and `inste(a,b,r)` are obtained from the neural classifiers through the DeepProbLog nn predicate, as explained later. These rules are slightly different from those by Huitzil et al., as they include the ABox and manage transitivity in a different way. Yet, they encode the same abstract behaviour. This translation to Prolog simulates the classical reasoning over an $\mathcal{EL}$ KB. In particular, to verify whether $A(x)$ is an instance of $\mathcal{K}$, we ask this program the query `d(x,a)`.

The neural ABox is handled through the nn predicate of DeepProbLog. Specifically, a call to nn with input a neural classifier and an instance, returns a list of probabilistic facts describing the distribution that the classifier assigns to the instance. In our example, the call to Γ over the input α yields the probabilistic facts 0.35 :: `inst(x,one)`, 0.4 :: `inst(x,four)`, and 0.45 :: `inst(x,seven)`, and seven other facts with probability 0. The probabilistic engine of DeepProbLog is used to compute the probability of a query. Correctness follows from the multiple-world semantics of ProbLog, which overall assigns the probability of each world as the product of the probabilities of the instances it includes, and the

[3] Since Problog sees capital letters as variables, concept names are transformed to lower case. We hence use x to refer to individual names.

Table 2
Axioms as facts

axiom	fact
$A \sqsubseteq C$	`s0(a,c).`
$A_1 \sqcap A_2 \sqsubseteq C$	`s1(a1,a2,c).`
$A \sqsubseteq \exists r.B$	`s2(a,r,b).`
$\exists r.A \sqsubseteq B$	`s3(r,a,b).`
$A(x)$	`inst(x,a).`
$r(x,y)$	`inste(x,y,r).`

Table 3
Completion rules as Prolog rules

```
t0(X,Y)  :-s0(X,Y).
t0(X,Y)  :-t0(X,Z), s0(Z,Y).
t0(X,Y)  :-t0(X,Z1), t0(X,Z2), s1(Z1,Z2,Y).
t2(X,R,Y):-t0(X,Z), s2(Z,R,Y).
t0(X,Y)  :-t2(X,R,Z1), t0(Z1,Z2), s3(Z2,R,Y).
```

```
d(X,Y)   :-inst(X,Y).
d(X,Y)   :-d(X,Z), t0(Z,Y).
d(X,Y)   :-d(X,Z1), d(X,Z2), s1(Z1,Z2,Y).
de(X,R,Y):-d(X,Z), t2(Z,R,Y).
de(X,R,Y):-inste(X,Z,R), d(Z,Y).
de(X,R,Y):-de(X,R,Z), t0(Z,Y).
d(X,Y)   :-de(X,R,Z), s3(R,Z,Y).
```

probability of an answer as the sum of the probabilities of the worlds where the answer holds. This is exactly the definition of $P(A(a))$. Note that the same instance may belong to different concept names associated to one or more neural classifiers. The semantics of DeepProbLog assumes that all probabilistic facts are probabilistically independent; yet, the many facts obtained through the multi-classifier are not. Thus, to obtain the adequate behaviour, one must add constraints to the program that guarantee that worlds in which an instance belongs to two disjoint classes are impossible.

As a side benefit of this translation, we can exploit the learning capabilities of DeepProbLog. Rather than assuming that the classifier for the basic neural concepts has been already trained elsewhere, it is possible to use the results of (implicit) consequences to train (or fine-tune) them. In our digit example, under the knowledge that α is indeed a small odd number, the classifier can be updated to increase the probability of α being `One` or `Three`, while decreasing that of being `Four` or `Seven`. Importantly, in this case, as should be clear from the example, the training cannot guarantee that the fine-grain features of the instances are learned, but only to the point where the implicit consequence is deduced. On the other hand, different implicit consequences can be combined to improve the quality of the classifier; e.g., if the training data can classify instances which are (i) odd, (ii) multiples of three, and (iii) small, separate fine-tuning will eventually yield a classifier separating most digits.

4. Conclusions

We introduced a neuro-symbolic extension of $\mathcal{EL}^{\bot}$, where in addition to a classical knowledge base, some concept and role assertions may be perceived by one or more neural classifiers. The combination of symbolic knowledge and neural classification allows for formal reasoning, with correctness guarantees, even in the presence of uncertain, imperfect perceptions. As we argue, our formalism allows for correcting potentially erroneous perceptions, in the presence of new information. We show how to use an existing DeepProbLog implementation to reason about, and learn concepts w.r.t. DeepEL ontologies.

Due to lack of space, and to emphasise the reduction to DeepProbLog, we focused on classifiers which are calibrated or, at least, return a probability distribution over the possible perceptions. Many classifiers do not behave in this way, though. In the full version [4], we argue how to handle more general *confidence* degrees.

As future work, we want to develop a native and more efficient implementation of the learner/reasoner, capable of dealing with larger KBs and a wider class of perceptions. We are also interested in answering complex queries beyond simple instance checking.

Acknowledgments

I. Huitzil supported by projects PID2024-159530OB-I00 (funded by MICIU/AEI/10.13039/501100011033)

and UZ2024-IyA-02 (funded by University of Zaragoza)..

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, 2nd ed., Cambridge University Press, 2007.

[2] R. Archana, P. S. E. Jeevaraj, Deep learning models for digital image processing: a review, Artif. Intell. Rev. 57 (2024) 11. URL: https://doi.org/10.1007/s10462-023-10631-z. doi:10.1007/S10462-023-10631-Z.

[3] F. Baader, S. Brandt, C. Lutz, Pushing the $\mathcal{EL}$ envelope, in: Proc. of 18th Int. Joint Conference on Artificial Intelligence (IJCAI 2005), Professional Book Center, 2005, pp. 364–369.

[4] A. Longato, I. Huitzil, R. Peñaloza, DeepEL: Deep learning and formal description logic reasoning, in: Proceedings of KR 2026, 2026. To appear.

[5] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. D. Raedt, Neural probabilistic logic programming in DeepProbLog, Artif. Intell. 298 (2021) 103504. URL: https://doi.org/10.1016/j.artint.2021.103504. doi:10.1016/J.ARTINT.2021.103504.

[6] J. A. Robinson, A machine-oriented logic based on the resolution principle, J. ACM 12 (1965) 23–41. URL: https://doi.org/10.1145/321250.321253. doi:10.1145/321250.321253.

[7] D. Carral, I. Dragoste, M. Krötzsch, Reasoner = logical calculus + rule engine, Künstliche Intell. 34 (2020) 453–463. URL: https://doi.org/10.1007/s13218-020-00667-6. doi:10.1007/S13218-020-00667-6.

[8] D. Carral, M. Krötzsch, Rewriting the description logic ALCHIQ to disjunctive existential rules, in: C. Bessiere (Ed.), Proc. of 29th Int. Joint Conference on Artificial Intelligence (IJCAI 2020), ijcai.org, 2020, pp. 1777–1783. URL: https://doi.org/10.24963/ijcai.2020/246. doi:10.24963/IJCAI.2020/246.

[9] İ. İ. Ceylan, J. Mendez, R. Peñaloza, The Bayesian ontology reasoner is BORN!, in: M. Dumontier, B. Glimm, R. S. Gonçalves, M. Horridge, E. Jiménez-Ruiz, N. Matentzoglu, B. Parsia, G. B. Stamou, G. Stoilos (Eds.), Proc. of the 4th International Workshop on OWL Reasoner Evaluation (ORE-2015), volume 1387 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015, pp. 8–14. URL: https://ceur-ws.org/Vol-1387/paper_5.pdf.

[10] I. Huitzil, G. Mazzotta, R. Peñaloza, F. Ricca, ASP-based axiom pinpointing for description logics, in: O. Kutz, C. Lutz, A. Ozaki (Eds.), Proc. of the 36th International Workshop on Description Logics (DL 2023), volume 3515 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3515/paper-14.pdf.

[11] I. Huitzil, G. Mazzotta, R. Peñaloza, F. Ricca, OWL2ASP tool, 2023. URL: https://doi.org/10.17632/r6xcggwvjp.1.