Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning
KR meets Machine Learning and Explanation

DeepEL: Deep Learning and Formal Description Logic Reasoning

Alessandro Longato[1] , **Ignacio Huitzil**[2] , **Rafael Peñaloza**[1]
[1]University of Milano-Bicocca, Italy
[2]University of Zaragoza, Aragon Institute of Engineering Research (I3A), Spain
alessandro.longato01@universitadipavia.it, ihuitzil@unizar.es, rafael.penaloza@unimib.it

Abstract

Light-weight description logics like $\mathcal{EL}$ have been successfully used to represent the terminological knowledge of many application domains. Yet, identifying the characteristics of instances often requires sub-symbolic approaches. We introduce a neuro-symbolic extension of $\mathcal{EL}$ which allows for (neural) perception and classification of properties, accompanied by formal (symbolic) reasoning based on these perceptions. Through a prototypical implementation, based on a reduction to logic programming, we show that inferences and learning are effectively achievable.

1 Introduction

Description logics (Baader et al. 2007) are a family of knowledge representation formalisms characterised by clear syntax and formal semantics that allow for effective automated reasoning. In particular, the $\mathcal{EL}$ family of light-weight DLs (Baader, Brandt, and Lutz 2005), which allows for polynomial-time reasoning tasks, has been successfully used to represent knowledge from many domains, particularly within the areas of biology and medicine. For instance, $\mathcal{EL}$ is the language underlying the large medical ontology SNOMED CT (Spackman 2007; Spackman 2008). Despite all its successes, $\mathcal{EL}$ suffers from the same issue as all purely logical formalisms: it requires a classification of the properties of domain object which cannot always be perfectly defined or characterised formally; for instance, when these properties are extracted from an image or a signal.

Deep learning models have been shown to be effective in image and signal processing, as witnessed by their many applications developed over the last decade (see e.g. (Archana and Jeevaraj 2024) for one of many surveys). However, by their very nature, these models cannot provide any correctness guarantee in their classification, and cannot perform any kind of formal reasoning. This limitation is particularly important in critical applications like in medicine, where a simple classification error may have dire consequences. Another important limitation of deep learning models is that they require training data that is labelled according to the properties of interest. If the properties change (e.g., if one is interested in a different level of abstraction or the information is updated) the model must be trained anew, which can be very costly.

For example, the International Union for Conservation of Nature's Red List of Threatened Species[1] is updated every year with different species changing category. While a pure deep learning model that classifies animals according to their danger level would need yearly re-trainings, a system that uses a model only to identify the species and applies the rules from the Red List to obtain the danger level needs only a simple update in those rules, without affecting the classification component.

In this paper, we follow the current trend on neuro-symbolic AI (Sarker 2022; Hitzler et al. 2022) and propose a formalism (called DeepEL) which combines formal reasoning with correctness guarantees (based on $\mathcal{EL}$ knowledge bases) with perception and signal processing from a deep learning model. Our formalism separates both tasks in a way that allows each component to contain the disadvantages of the other. In practical terms, the $\mathcal{EL}$ component reasons over the confidence provided by the neural component at classification time.

To test the effectiveness of our formalism, we implement a simple prototype, which uses a known translation from $\mathcal{EL}$ to Prolog, and takes advantage of the inference and learning capabilities of DeepProbLog (Manhaeve et al. 2021), a neuro-symbolic tool extending Prolog with neural predicates associated to classifiers.

An empirical analysis shows that the approach is effective both, if a pre-trained classifier is used and if the classifier is trained over implicit inferences from the knowledge base. These results suggest that description logics can very well be used at the base of neuro-symbolic systems, thus taking advantage of existing knowledge bases that have been developed over the years.

2 Preliminaries

We start by briefly recalling the main notions of the description logic $\mathcal{EL}$ and DeepProbLog needed for understanding the rest of our contribution.

2.1 $\mathcal{EL}$

$\mathcal{EL}$ and $\mathcal{EL}^{\bot}$ (Baader, Brandt, and Lutz 2005) are light-weight description logics (DLs) (Baader et al. 2007) which have been successfully used to represent the knowledge

[1]https://www.iucnredlist.org/

of many different application domains, specially from the biomedical fields. As with all DLs, knowledge is represented by restricting the possible interpretation of *concepts* (corresponding to first-order unary predicates) and *roles* (binary predicates); and by associating properties to domain objects. This is formalised next.

Definition 1 ($\mathcal{EL}^\bot$ syntax)**.** *Let N_I, N_C, and N_R be three mutually disjoint sets, whose elements are called* individual names*,* concept names*, and* role names*, respectively. The class of $\mathcal{EL}^\bot$* concepts *is defined through the grammar rule $C ::= A \mid \top \mid \bot \mid C \sqcap C \mid \exists r.C$, where $A \in N_C$ and $r \in N_R$. A* general concept inclusion *(GCI) is an expression of the form $C \sqsubseteq D$, where C, D are concepts. A* TBox *is a finite set of GCIs. An* assertion *is an expression of the form $A(a)$ (concept assertion) or $r(a, b)$ (role assertion), where $a, b \in N_I$, $A \in N_C$, and $r \in N_R$. An* ABox *is a finite set of assertions. A* knowledge base *(KB)—or* ontology*—is composed of a TBox and an ABox. Given a KB $\mathcal{K}$, $N_I(\mathcal{K})$, $N_C(\mathcal{K})$, and $N_R(\mathcal{K})$ denote the class of individual, concept, and role names appearing in $\mathcal{K}$, respectively.*

A KB expresses restrictions on the class of interpretations of interest, as formalised through first-order logic semantics.

Definition 2 ($\mathcal{EL}^\bot$ semantics)**.** *An* interpretation *is a pair $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ where $\Delta^\mathcal{I}$ is a non-empty set called the* domain *and $\cdot^\mathcal{I}$ is the* interpretation function *which maps: every individual name $a \in N_I$ to an element $a^\mathcal{I} \in \Delta^\mathcal{I}$; every concept name $A \in N_C$ to a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$; and every role name $r \in N_R$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. This interpretation function is extended to arbitrary concepts by setting: $\top^\mathcal{I} := \Delta^\mathcal{I}$; $\bot^\mathcal{I} := \emptyset$; $(C \sqcap D)^\mathcal{I} := C^\mathcal{I} \cap D^\mathcal{I}$; and $(\exists r.C)^\mathcal{I} := \{\delta \in \Delta^\mathcal{I} \mid \exists \eta \in C^\mathcal{I}.(\delta, \eta) \in r^\mathcal{I}\}$.*

$\mathcal{I}$ satisfies *the GCI $C \sqsubseteq D$ iff $C^\mathcal{I} \subseteq D^\mathcal{I}$; it* satisfies *the concept assertion $A(a)$ iff $a^\mathcal{I} \in A^\mathcal{I}$ and the role assertion $r(a, b)$ iff $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$. $\mathcal{I}$ is a* model *of the KB $\mathcal{K}$ iff it satisfies all the GCIs and assertions in $\mathcal{K}$.*

$\mathcal{EL}$ is the sublogic of $\mathcal{EL}^\bot$ where the special concept $\bot$ is not allowed.

The goal of knowledge representation is not only to *represent* the knowledge of a domain (through a KB), but also to *reason* about this knowledge; that is, to understand what other properties are guaranteed under the constrains imposed by the KB. While many different reasoning services have been considered in the literature, here we focus on *instance checking*.

Definition 3 (instance checking)**.** *An individual $a \in N_I$ is an* instance *of the concept name $A \in N_C$ w.r.t. the KB $\mathcal{K}$ ($\mathcal{K} \models A(a)$) iff every model of $\mathcal{K}$ satisfies the assertion $A(a)$. The problem of* instance checking *consists in deciding, given $a \in N_I$, $A \in N_C$, and a KB $\mathcal{K}$, whether a is an instance of A w.r.t. $\mathcal{K}$.*

It is well known that instance checking in $\mathcal{EL}^\bot$ can be solved in polynomial time. One way to achieve this is through the completion algorithm proposed in (Baader, Brandt, and Lutz 2005), which makes a representative class of implicit consequences, including instances, explicit.[2] The algorithm takes as input an ontology in *normal form*; that is, where all GCIs are of one of the forms

$$A \sqsubseteq B, \quad A_1 \sqcap A_2 \sqsubseteq B, \quad \exists r.A \sqsubseteq B, \quad A \sqsubseteq \exists r.B$$

where $r \in N_R$, $A, A_1, A_2 \in N_C \cup \{\top\}$, $B \in N_C \cup \{\top, \bot\}$. This is not a limitation since every ontology can be rewritten into an equivalent one (w.r.t. the original vocabulary) in normal form. The algorithm starts with a class $\mathcal{C}$ of obvious consequences and explicit concept assertions

$$\mathcal{C} := \{\top(a), A \sqsubseteq A, A \sqsubseteq \top \mid a \in N_I(\mathcal{K}), A \in N_C(\mathcal{K})\} \cup \{A(a) \mid A(a) \in \mathcal{A}\},$$

which is expanded by applying the completion rules from Table 1, until no rule can be applied any more. When the algorithm terminates, the set $\mathcal{C}$ obtained is sound and complete w.r.t. atomic subsumption and instance checking; that is, $\mathcal{K} \models A(a)$ iff $A(a) \in \mathcal{C}$ or $\mathcal{K}$ is inconsistent. The latter can be verified by checking whether $\top \sqsubseteq \bot \in \mathcal{C}$ or there exists some $b \in N_I$ such that $\bot(b) \in \mathcal{C}$. We will use this result in the following sections.

if $A \sqsubseteq B \in \mathcal{K}$	**and** $X \sqsubseteq A \in \mathcal{C}$	**then add** $X \sqsubseteq B$	to $\mathcal{C}$
if $A(a) \in \mathcal{C}$	**and** $A \sqsubseteq B \in \mathcal{C}$	**then add** $B(a)$	to $\mathcal{C}$
if $r(a, b) \in \mathcal{K}$	**and** $A(b) \in \mathcal{C}$	**then add** $\exists r.A(a)$	to $\mathcal{C}$
if $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{K}$	**and** $X \sqsubseteq A_1, X \sqsubseteq A_2 \in \mathcal{C}$	**then add** $X \sqsubseteq B$	to $\mathcal{C}$
if $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{K}$	**and** $A_1(a), A_2(a) \in \mathcal{C}$	**then add** $B(a)$	to $\mathcal{C}$
if $A \sqsubseteq \exists r.B \in \mathcal{K}$	**and** $X \sqsubseteq A \in \mathcal{C}$	**then add** $X \sqsubseteq \exists r.B$	to $\mathcal{C}$
if $A \sqsubseteq \exists r.B \in \mathcal{C}$	**and** $A(a) \in \mathcal{C}$	**then add** $\exists r.B(a)$	to $\mathcal{C}$
if $\exists r.A \sqsubseteq B \in \mathcal{K}$	**and** $X \sqsubseteq \exists r.Y, Y \sqsubseteq A \in \mathcal{C}$	**then add** $X \sqsubseteq B$	to $\mathcal{C}$
if $\exists r.A \sqsubseteq B \in \mathcal{K}$	**and** $\exists r.Y(a), Y \sqsubseteq A \in \mathcal{C}$	**then add** $B(a)$	to $\mathcal{C}$

Table 1: $\mathcal{EL}^\bot$ completion rules.

2.2 DeepProbLog

DeepProbLog (Manhaeve et al. 2021) is a neuro-symbolic probabilistic extension of the well-known logic programming language Prolog (Robinson 1965).

Very briefly, a Prolog *rule* is a predicate Horn clause written as an implication

$$p_1(t_1^1, \ldots, t_{n1}^1) \wedge \cdots \wedge p_k(t_1^k, \ldots, t_{nk}^k) \rightarrow q(s_1, \ldots, s_m),$$

where p_i, q are predicate symbols and t_i^j, s_i are terms (that is, constants or variables).[3] The syntax of Prolog distinguishes predicate and constant symbols from variables through their case: the former use names starting with a lower case, while the latter start with an upper case. It also expresses the implication in a slightly different manner. For instance, the clause $p(x, y), q(y, a) \rightarrow r(a, y)$, where x, y are variables and a a constant, is expressed in Prolog syntax as the rule

```
r(a,Y) :- p(X,Y), q(Y,a).
```

[2] See http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/ for a slightly outdated list of reasoners, including for $\mathcal{EL}^\bot$, which follow this or other reasoning strategies.

[3] For the sake of simplicity, and since they are not relevant for the rest of this paper, we leave out functional symbols from this description.

A Prolog *program* is a set of rules, in which all the variables are (implicitly) universally quantified. Formally, a query `q(a)` is *entailed* by a program if it holds in all models of the program. Prolog uses unification and back tracing of rules to find answers efficiently.

ProbLog (Raedt, Kimmig, and Toivonen 2007) extends Prolog by allowing *probabilistic facts* of the form `p::a.` where $p \in (0, 1)$ and a is a ground predicate; i.e., a predicate without any variables. It uses a mutiple-world semantics, where, for each subset of probabilistic facts, there is one world consisting of a (classical) Prolog program, including the chosen probabilistic facts stripped off their probabilities. Each world gets assigned a probability given by the product of the probabilities of the probabilistic facts it contains. The probability of a query entailment `q(a)` is then computed as the sum of the probabilities of all worlds which classically entail `q(a)`. While this computation may seem to require exponential time, ProbLog uses techniques from formula compilation and weighted model counting (Kimmig et al. 2011; Bryant 1986) to find these probabilities more efficiently, which is fundamental given that the problem is PP-hard (Ceylan and Peñaloza 2017).

DeepProbLog further extends ProbLog through an interface which allows for calls to neural classifiers. Specifically, DeepProbLog includes a new reserved predicate symbol `nn` which is used to call the external classifier over an instance and populate a ProbLog program with the answers given by the classifier. This is achieved through so-called *neural annotated disjunctions* (nADs) and *neural facts*. Syntactically, a nAD is an expression of the form

```
nn(m,[X1,...,Xk],O,[y1,...,yn])::
                          r(X1,...,Xk,O).
```

where `m` is the name of the neural classifier, `Xi` are its input variables, `O` is its output variable, `yi` are the different classes of the distribution, and `r` is the predicate symbol used in the probabilistic facts generated. When triggered, this neural annotated disjunction adds n probabilistic facts of the form p_i `::r(x1,...,xk,yi)` for each $i, 1 \leq i \leq n$, where p_i is the probability of output y_i under the input values $x_1, \ldots, x_k$. A neural fact is of the form `nn(m,[X1,...,Xk])::r(X1,...,Xk).` This has the same behaviour, assuming that the classifier is binary, and hence returns only the probability of the input having the property identified by `m`.

Intuitively, one can think of nADs and neural facts as perceiving the signals $x_1, \ldots, x_n$ and returning its belief on each of the available classes $y_1, \ldots, y_k$.

Importantly, the connection between the neural classifier `m` and the ProbLog program is bidirectional: not only can the resulting program be used to make inferences about the perceived input, but the results of the inference can be used to train or fine-tune the classifier. The full details of this task fall beyond the scope of this work, but we refer the interested reader to (Manhaeve et al. 2021).

3 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 behind DeepEL is that general terminological knowledge (i.e., a TBox) has been formalised, but needs to be applied to understand the properties of objects which are perceived through fallible sub-symbolic methods such as a neural classifier. 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.

As a simple motivating example, consider a neural classifier which has been trained to identify digits from a handwritten document. Suppose now 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. With a TBox, instead, one can simply express which digits are even and which are odd—e.g., through the GCIs $\mathsf{Zero} \sqsubseteq \mathsf{Even}$, $\mathsf{One} \sqsubseteq \mathsf{Odd}$, …, $\mathsf{Nine} \sqsubseteq \mathsf{Odd}$—and reuse the classifier without modification.

Suppose, for the sake of the example, that the neural classifier produces a probability distribution expressing the uncertainty of a given input corresponding to each possible digit.[4] 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 serves as a motivation for the formal architecture of DeepEL which we will introduce next. We consider first an idealised scenario where the classifier output corresponds to a multinomial distribution, and then discuss the variants required to cover other scenarios.

For the rest of this section, we assume that we have a fixed TBox $\mathcal{T}$, a fixed classical ABox $\mathcal{A}$, and consider an ABox which is populated through one or more neural classifiers. More formally, consider 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 defined as $\mathcal{A}_\alpha^\Gamma := \{p_1 :: A_1^\Gamma(\alpha), \ldots, p_{n_\Gamma} :: A_{n_\Gamma}^\Gamma(\alpha)\}$, where p_i is the confidence level of Γ for the class A_i^Γ on input α, henceforth denoted as $P(A_i^\Gamma(\alpha))$. Hence, in our running example, if Γ is the digit classifier and α is the input image,

[4] This can be achieved e.g., by normalising the classification output probabilities.

we get[5]

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

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. As neural classifiers may provide different interpretations for the confidence level in their output, we divide this semantics in two cases.

3.1 Multinomial Distributions

Suppose first that the output of the classifiers is normalised and their output represents a multinomial distribution of the classes, expressing the probability that the input belongs to each of them. 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 possibilities 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 will correspond 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. Note that each $\mathcal{K}_\omega$ is a classical $\mathcal{EL}^\bot$ KB. The probability of each outcome is $P(\omega) = \prod_{A^\Gamma_i(\alpha)\in\omega} P(A^\Gamma_i(\alpha))$

Continuing our example, suppose that we have an additional input image β, which is connected to α through the (classical) assertion $\mathsf{same\_parity}(\alpha, \beta)$, stating that α and β have the same parity, as formalised through the GCIs $\exists\mathsf{same\_parity}.\mathsf{Even} \sqsubseteq \mathsf{Even}$, $\exists\mathsf{same\_parity}.\mathsf{Odd} \sqsubseteq \mathsf{Odd}$; and that $\mathcal{A}^\Gamma_\beta = \{0.6 :: \mathsf{Eight}(\beta), 0.4 :: \mathsf{Three}(\beta)\}$. 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 2).[6]

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.

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

Table 2: Six worlds and their probability from the running example. All worlds are assumed to include the classical axioms; only the neural assertions are depicted.

In our example, we see that 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. Note, however, that our knowledge allows us to derive, in some worlds, that the input α is Odd and Even at the same time (for instance, in the world where $\mathsf{Four}(\alpha)$ 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 that these two classes must be disjoint. Under this new knowledge, some worlds (those where α or β was simultaneously odd and even) become inconsistent, and hence describe situations which are impossible to happen given our knowledge. As such, they must be assigned probability 0. To preserve a probability distribution, we proportionally scale up the probabilities of all remaining (consistent) worlds.

Formally, let $\Omega_\bot := \{\omega \in \Omega \mid \mathcal{K}_\omega \text{ is inconsistent}\}$ be the set of all inconsistent worlds and $P(\Omega_\bot) := \sum_{\omega\in\Omega_\bot} P(\omega)$ where P is the probability distribution defined previously. We construct a new probability distribution

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

which is equivalent to the probability distribution conditioned on the event of having a consistent world.

We can now define the *probability of the assertion* $A(a)$ w.r.t. the neural KB $\mathcal{K}$ as $P(A(a)) := \sum_{\mathcal{K}_\omega \models A(a)} P_\bot(\omega)$. In words, the probability of an assertion is the sum of the probabilities of all consistent worlds that entail this assertion. Computing this probability is the natural probabilistic variant of the problem of instance checking.

Returning to our example, we can see that three of the six worlds from Table 2 are inconsistent; namely those where α and β have different parity, 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.

[5]For brevity, from now on we will disregard all expressions of the form $0 :: A(a)$. The reason should be clear when the semantics is introduced.

[6]Formally, there would be 100 worlds, but all others have probability 0, so we do not consider them here.

Before presenting a method for computing this probability, we discuss how to deal with other kinds of classifiers.

3.2 Confidence

While typical neural classifiers associate a score between 0 and 1 to each of the classes on a given input, this score seldom reflects a probability, for two main reasons: 1) the scores might not be normalised to sum 1; and 2) even when normalised, there is no theoretical reason to call these numbers probabilities; the results of the classifier do not necessarily behave as a probability distribution.[7] These scores are thus often rather called *confidence*: a higher score represents a higher "trust" on the answer of the classifier.

In general, as confidence is an underspecified term, there is no specific operation which correctly computes the confidence of an implicit result given the outputs of the classifier. Yet, some natural operations have been used in the literature, based on different intuitions. Most of these options follow a pattern which can be thought of as a generalisation of the method for dealing with multinomial distributions from Section 3.1: consider two different binary operations $\otimes$ and $\oplus$ on $[0, 1]$. Given a world $\omega \in \Omega$, we can define the *confidence* of ω as $\mathsf{conf}(\omega) := \bigotimes_{p::\alpha\in\omega} p$. In words, the confidence of an outcome is the "product" ($\otimes$) of the confidences of the assertions in it. Given an assertion $\alpha := A(a)$, let $\Omega_{A(a)}$ be the set of all worlds $\omega \in \Omega$ such that $\mathcal{K}_\omega \models A(a)$. Then, the *confidence* of $A(a)$ is

$$\mathsf{conf}(A(a)) := \bigoplus_{\omega\in\Omega_{A(a)}} \mathsf{conf}(\omega).$$

Note that interpreting $\otimes$ as the product of real numbers and $\oplus$ as the addition yields exactly the probability computed in the previous subsection, if all the worlds with positive probability are consistent. Yet, other operators can also be considered. For instance, the confidence on the assertion might be the *maximum* of the confidences of the worlds that entail it, where the latter may be computer either by multiplying or by minimising the individual confidences (Mohamed et al. 2022; Qi, Pan, and Ji 2007).

Under some minor assumptions on the operators (mainly, associativity, existence of neutrals, distributivity, and absorption—known as a semiring), this corresponds to the computation of the provenance, a task which has been studied for many languages, including $\mathcal{EL}$ and $\mathcal{EL}^\bot$ (Bourgaux, Ozaki, and Peñaloza 2023; Calautti et al. 2024). The general computation of the confidence of a consequence in DeepEL is described in Algorithm 1. Assuming that the semiring operations are *cheap*, this algorithm runs in exponential time, by making an exponential number of (polynomial-time) entailment tests. While this bound cannot be improved in general, there are special kinds of semirings for which this confidence can be computed more efficiently. For instance, if $\otimes$ and $\oplus$ are $\min$ and $\max$, respectively, then computing the confidence requires only polynomial time (Qi, Pan, and Ji 2007). More in general, if $\oplus = \max$, then this problem is in Δ_2^p: one simply needs to guess (in NP) the world that maximises $\mathsf{conf}(\omega)$ and verify (in co-NP) that indeed no other world has a higher confidence. Note moreover that if some worlds are inconsistent, the computation can easily be adapted by simply changing the condition in line 3 to "**if** $\mathcal{K}_\omega \models A(a)$ and $\mathcal{K}_\omega$ is consistent **then**."

[7] Some work focuses on *calibrating* the scores to reflect a probability; see e.g. (Guo et al. 2017).

Algorithm 1: Computing entailment confidence

Data: $\mathcal{K}$ neural KB, $A(a)$ assertion
Result: Confidence of $A(a)$ w.r.t. $\mathcal{K}$

```
1 c ← 0
2 for ω ∈ Ω do
3 |   if K_ω ⊨ A(a) then
4 |   |   c ← c ⊕ conf(ω)
5 return (p_0, q_0)
```

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: Axioms as facts

In the next section we take advantage of some known connections between $\mathcal{EL}^\bot$ and ProbLog to implement a DeepEL reasoner based on multinomial distributions. This approach has the additional advantage of allowing learning and fine-tuning of the neural classifiers through the answers of the reasoner as well.

4 Implementing a Reasoner

Rather than implementing a DeepEL reasoner from scratch, we follow the strategy of transforming the ontology into a logic programming formalism, and taking advantage of the efficient implementations available for those formalisms, as done in the past (Carral, Dragoste, and Krötzsch 2020; Carral and Krötzsch 2020; Ceylan, Mendez, and Peñaloza 2015; Huitzil et al. 2023a). Specifically, we use OWL2ASP (Huitzil et al. 2023b), which translates $\mathcal{EL}^\bot$ KBs into sets of facts corresponding to GCIs in normal form and simple assertions (see Table 3).[8] The reasoning steps of the completion algorithm are simulated through the Prolog rules from Table 4, where `inst(x,a)` and `inste(a,b,r)` are obtained from the neural classifiers through the DeepProbLog `nn` predicate, as explained later. Note that the rules are slightly different from those by Huitzil et al., as they include handling of 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

[8] Since Pro(b)log sees capital letters as variables, concept names need to be transformed to lower case. Hence, here we use x to refer to individual names.

```
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).
```

Table 4: Completion rules as Prolog rules

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 discrete distribution that the classifier assigns to the instance. In our example from Section 3, the call to Γ over the input image α yields the probabilistic facts 0.35 :: `inst(x,one)`, 0.4 :: `inst(x,four)`, and 0.45 :: `inst(x,seven)`, along with seven other facts with probability 0. The probabilistic engine of DeepProbLog is then used to compute the probability of a query. The correctness of this approach 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 probability of an answer as the sum of the probabilities of the worlds where the answer holds. This is exactly the definition of $P(A(a))$ from Section 3.1.

Note, however, 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; however, 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 using this translation to DeepProbLog, we can take advantage also of the learning capabilities of the system. Hence, 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.

5 Experiments

We performed an empirical evaluation over the implementation of a DeepEL reasoner described in the previous section.[9] The main goal of the experiments is to verify the feasibility of the prototype and quantify its capabilities in tasks that require both perceptual learning and logical reasoning. We divide this goal into three subgoals:

1. The first goal is to evaluate the performance of the prototype with pre-trained neural networks for attribute classification; that is, evaluate the *inferential* component of the system, assuming that an adequate classifier exists. The hypothesis is that the model should already perform well in symbolic tasks when reasoning on the output of a fairly accurate neural component, given the logical guarantees of the reasoner.
2. The second goal is to demonstrate the feasibility of DeepProbLog end-to-end training within the DeepEL framework. In principle, it should be possible to train the neural component from the symbolic feedback of the logical reasoner. We thus train a classifier based on implicit consequences.
3. As a third goal, we want to understand the quality of a classifier trained on implicit consequences against one trained on the original features populated by the neural ABox. Hence, we analyse whether the difference in performance obtained through these strategies is significant.

We describe our approach towards those goals next.

5.1 Dataset

To evaluate the effectiveness of the proposed approach and to investigate the research questions described above, a dataset containing both sub-symbolic data and symbolic information needs to be employed. We hence consider the Animals With Attributes 2 (AWA2) dataset,[10] which was originally designed as a benchmark for zero-shot learning (Xian et al. 2019). The dataset contains 37,322 images belonging to 50 distinct mammal classes. However, its most relevant feature for the scope of our experiments is the set of symbolic annotations it contains. Each of the 50 mammal classes is attached to a vector consisting of 85 semantic human-readable attributes which are shared between classes and describe, for example, the animal colour, pattern, or behaviour. These attributes are provided both as binary values, representing presence or absence, and as continuous values, representing the degree of presence. For simplicity and to maintain a clear connection to logical predicates, which are intrinsically true or false, we used the binary attributes.

These characteristics make of AWA2 a well suited dataset to test the capabilities of our prototypical implementation of a DeepEL system. The semantic attributes provide the foundations to build the knowledge base, while the images

[9] The system and full evaluation are publicly available at the repository https://github.com/a-longato/deepel/tree/main.

[10] https://cvml.ista.ac.at/AwA2/

provide the data to be fed to the neural networks. The use of both the data modalities allows to ground the raw image data into the semantic knowledge. The attributes can be combined to define more complex concepts to test reasoning tasks beyond neural classification. To simplify the computational side of the experiments, the 2048-dimensional feature vectors extracted from the top-layer pooling units of a ResNet-101 model pre-trained on ImageNet were used instead of the original images as inputs for the neural networks. These vectors are provided together with the images in the dataset. This simplification enables to focus on the correct neuro-symbolic integration, rather than on the expensive task of training a competent image classifier from scratch.

5.2 DeepEL Program Structure

The DeepEL programs written for the experiments follow a specific five-part structure:

Neural Predicates `nn`, one for each attribute, linking the image features to probabilistic facts as in Section 2.2.

Bridging Predicates translate the probabilistic outputs of the neural predicates into deterministic logical assertions for the reasoner. For an attribute 'attr,' they take the form `inst(Image,attr):-nn_attr(Image,1)`, where `nn_attr` is the NN that verifies attr.

Knowledge Base: $\mathcal{EL}$ GCIs encoded as ProbLog facts.

Reasoning Rules encode the completion rules in Prolog, as presented in Table 4.

Query Predicate: The final high-level predicate that represents the object of the query for each task.

5.3 Task Design

For each experiment, we construct two components: (i) a symbolic component consisting of an $\mathcal{EL}$ TBox, constructed from the semantic attributes of AWA2. The knowledge is constructed to describe a (high-level) class from the attributes associated to it; and (ii) a neural component built as a set of multi-layer perceptrons (MLPs). One MLP is instantiated for each attribute required by the knowledge base. Each of them takes as input a 2048-dimensional vector and it is composed of two hidden layers, with 512 and 128 units respectively, with GeLU activation, normalization, and dropout. The output layer is constituted by two units with softmax activation that represent the probability for presence or absence of the attribute. For example, as we will see in Section 5.4, the TBox can express that Ground, Quadrupedal animals are Terrestrial. Two MLPs will be used to detect whether an image contains a ground animal and a quadrupedal, respectively, and this information is propagated to infer whether it is terrestrial or not.

Each experiment is structured in two phases. To ensure result comparability, the exact same dataset split is used across all experiments and phases. The 37322 samples of the AWA2 dataset were split between 80%, 29857 samples, for the training set and 20%, 7465 samples, for the test set. The first phase is the inference evaluation using pre-trained neural networks. Each MLP was trained on the training set with a standard PyTorch training loop for 50 epochs and then tested with the test set. Each individual MLP reached an accuracy greater than 0.95 with some of them surpassing a 0.99 accuracy. The weights of these pre-trained MLPs corresponding to the attributes utilized in the program were loaded in the DeepEL system. Lastly, the model was run in inference mode on the test set. Quality is measured according to the implicit attribute of interest.

The second phase is end-to-end training from scratch guided by the logic program. As in the first phase, a DeepEL model is assembled, but now with *untrained* MLPs. The learning does not concern the individual base attributes, but it aims to satisfy the high-level query. The model is trained on the training set for 10 epochs using DeepProbLog learning pipeline. After that, the model is evaluated the exact same way as the model from the first phase. The quality of the results is then directly comparable between the two approaches.

5.4 Reasoning Tasks

We construct five experiments of differing difficulty aimed at evaluating different properties of the system, as described next.

Terrestrial This experiment is focused on the binary classification of samples into the classes of terrestrial and non-terrestrial mammals. The experiment uses two attributes, ground and quadrupedal, to define the membership to to positive or negative terrestrial class. The TBox is very simple using only conjunctions and two GCIs:

$$\mathsf{Ground} \sqcap \mathsf{Quadrupedal} \sqsubseteq \mathsf{Terrestrial}$$
$$\mathsf{Not\_Ground} \sqsubseteq \mathsf{Not\_Terrestrial}$$

Note that since $\mathcal{EL}^{\bot}$ does not allow negations, Not_Terrestrial is not the logical complement of Terrestrial. While we could add the GCI $\mathsf{Terrestrial} \sqcap \mathsf{Not\_Terrestrial} \sqsubseteq \bot$ to guarantee disjointness, we decided not to do this and rather treat them as separate classes.

Carnivore This experiment aims at classifying mammals in carnivores and non-carnivores, defining the positive class as animals that are both predators and eat fish or meat so that it can test the capacity of the system where more than one option is available for deducing a class. The TBox contains the GCIs:

$$\mathsf{Hunter} \sqcap \mathsf{Meat} \sqsubseteq \mathsf{Predator}$$
$$\mathsf{Hunter} \sqcap \mathsf{Fish} \sqsubseteq \mathsf{Predator}$$
$$\mathsf{Not\_Hunter} \sqcap \mathsf{Meat} \sqsubseteq \mathsf{Not\_Predator}$$
$$\mathsf{Not\_Hunter} \sqcap \mathsf{Fish} \sqsubseteq \mathsf{Not\_Predator}$$
$$\mathsf{Not\_Hunter} \sqcap \mathsf{Not\_Meat} \sqsubseteq \mathsf{Not\_Predator}$$
$$\mathsf{Not\_Hunter} \sqcap \mathsf{Not\_Fish} \sqsubseteq \mathsf{Not\_Predator}$$

	Test	Accuracy	95% CI
pretrained	aquatic	0.9898	(0.9873–0.9919)
	carnivore	0.9680	(0.9637–0.9717)
	terrestrial	0.9908	(0.9883–0.9927)
	ungulate	0.9775	(0.9739–0.9806)
	combined	0.9762	(0.9724–0.9794)
end-to-end	aquatic	0.9908	(0.9883–0.9927)
	carnivore	0.9673	(0.9630–0.9711)
	terrestrial	0.9905	(0.9880–0.9925)
	ungulate	0.9796	(0.9762–0.9826)
	combined	0.9783	(0.9747–0.9814)

Table 5: Overall accuracy and 95% confidence interval of the accuracy for all experiments.

Aquatic This experiment uses multi-class classification problem. The task is to differentiate between aquatic and non-aquatic mammals, and additionally discriminate the aquatic animals between furry and hairless.

$$\begin{aligned} \text{Swims} \sqcap \text{Water} &\sqsubseteq \text{Aquatic} \\ \text{Not\_Swims} \sqcap \text{Not\_Water} &\sqsubseteq \text{Not\_Aquatic} \\ \text{Aquatic} \sqcap \text{Hairless} &\sqsubseteq \text{Hairless\_Aquatic} \\ \text{Aquatic} \sqcap \text{Furry} &\sqsubseteq \text{Furry\_Aquatic} \end{aligned}$$

Ungulate This experiment focuses in distinguishing between non-ungulates, non-horned ungulates, and horned ungulates with a longer reasoning chain using several intermediate concepts.

$$\begin{aligned} \text{Ground} \sqcap \text{Quadrupedal} &\sqsubseteq \text{Terrestrial} \\ \text{Vegetation} &\sqsubseteq \text{Herbivore} \\ \text{Terrestrial} \sqcap \text{Herbivore} &\sqsubseteq \text{Prey} \\ \text{Prey} \sqcap \text{Hooves} &\sqsubseteq \text{Ungulate} \\ \text{Ungulate} \sqcap \text{Horns} &\sqsubseteq \text{Horned\_Ungulate} \\ \text{Ungulate} \sqcap \text{Not\_Horns} &\sqsubseteq \text{Not\_Horned\_Ungulate} \end{aligned}$$

Combined The last experiment uses the full expressivity of $\mathcal{EL}$ and considers more limit cases, classifying instances in *primate*, *aquatic*, *ungulate*, and *other*. The KB has the 19 GCIs shown in Figure 1. This experiment evaluates the possible use of existential quantifiers, and classes which may interact in different manners.

5.5 Results

The overall accuracy of the method in the different experiments, along with a 95% confidence interval for this accuracy are presented in Table 5, divided into the pre-trained classifier and the classifier trained from scratch (end-to-end).[11] As it is clear from these intervals, the prototype behaviour has no significant difference between experiments and training approach. The detailed results per class are shown in Table 6 (for the pre-trained classifier) and Table 7 (end-to-end).

[11] As all point estimators lie close to 1, we use Wilson confidence intervals (Brown, Cai, and DasGupta 2001) in all cases.

	Precision		Recall		
Class	est.	95% CI	est.	CI	F1
not_aquatic	0.9983	(0.9969–0.9990)	0.9938	(0.9916–0.9955)	0.9960
hairless_aquatic	0.9688	(0.9544–0.9788)	0.9712	(0.9572–0.9808)	0.9700
furry_aquatic	0.8901	(0.8538–0.9183)	0.9586	(0.9317–0.9752)	0.9231
not_carnivore	0.9826	(0.9786–0.9858)	0.9710	(0.9660–0.9752)	0.9767
carnivore	0.9364	(0.9258–0.9455)	0.9612	(0.9525–0.9684)	0.9486
not_terrestrial	0.9474	(0.9314–0.9599)	0.9793	(0.9680–0.9867)	0.9631
terrestrial	0.9971	(0.9954–0.9981)	0.9924	(0.9899–0.9942)	0.9947
not_ungulate	0.9957	(0.9934–0.9972)	0.9808	(0.9766–0.9843)	0.9882
not_horned_ung.	0.9434	(0.9273–0.9562)	0.9511	(0.9358–0.9629)	0.9473
horned_ungulate	0.9428	(0.9303–0.9532)	0.9836	(0.9759–0.9889)	0.9628
primate	0.9735	(0.9520–0.9856)	0.9583	(0.9334–0.9742)	0.9659
aquatic	0.9657	(0.9536–0.9747)	0.9912	(0.9839–0.9952)	0.9783
ungulate	0.9640	(0.9561–0.9705)	0.9908	(0.9863–0.9939)	0.9772
other	0.9895	(0.9855–0.9925)	0.9624	(0.9555–0.9683)	0.9758

Table 6: Precision, recall (point estimate and 95% confidence intervals), and F1 for all the classes in the experiments executed, when using a pre-trained classifier to populate the neural ABox.

	Precision		Recall		
Class	est.	95% CI	est.	CI	F1
not_aquatic	0.9960	(0.9941–0.9973)	0.9956	(0.9936–0.9969)	0.9958
hairless_aquatic	0.9776	(0.9648–0.9858)	0.9776	(0.9648–0.9858)	0.9776
furry_aquatic	0.9286	(0.8974–0.9508)	0.9363	(0.9062–0.9572)	0.9324
not_carnivore	0.9824	(0.9784–0.9856)	0.9702	(0.9652–0.9745)	0.9763
carnivore	0.9347	(0.9240–0.9439)	0.9608	(0.9520–0.9680)	0.9475
not_terrestrial	0.9695	(0.9565–0.9787)	0.9564	(0.9416–0.9676)	0.9629
terrestrial	0.9936	(0.9913–0.9952)	0.9955	(0.9936–0.9969)	0.9945
not_ungulate	0.9877	(0.9842–0.9904)	0.9934	(0.9907–0.9953)	0.9906
not_horned_ung.	0.9636	(0.9501–0.9736)	0.9436	(0.9277–0.9562)	0.9535
horned_ungulate	0.9648	(0.9546–0.9728)	0.9606	(0.9499–0.9691)	0.9627
primate	0.9683	(0.9453–0.9817)	0.9683	(0.9453–0.9817)	0.9683
aquatic	0.9503	(0.9362–0.9613)	0.9919	(0.9848–0.9958)	0.9707
ungulate	0.9857	(0.9803–0.9896)	0.9770	(0.9705–0.9821)	0.9813
other	0.9835	(0.9786–0.9873)	0.9759	(0.9702–0.9806)	0.9797

Table 7: Precision, recall (point estimate and 95% confidence intervals), and F1 for all the classes in the experiments executed, when training end-to-end for the classification of the low-level concepts.

The results demonstrate that the proposed framework does indeed work, realizing a strong and reliable integration of neural perception and $\mathcal{EL}$ reasoning. Loading individually pre-trained neural networks into the models yields very high performance, with F1 scores well over 0.95 across all experiments. This demonstrates that when equipped with accurate perceptual modules, the logic program can effectively use their outputs to solve logical queries, realising the aim of a powerful neuro-symbolic system.

The second notable result is that training DeepEL models from scratch not only is possible, but the experiments have shown that they reach performances comparable to the models which use pre-trained MLPs, even when the training was over a lower number of epochs. It confirms the feasibility of using the DeepProblog learning pipeline with $\mathcal{EL}$ KBs.

While metrics like accuracy are similar between the two modalities, looking at finer grain metrics like per-class precision, recall, and F1 reveals more nuanced differences. In the simpler Terrestrial and Carnivore experiments, the differences are negligible. The benefits of end-to-end training become clearer in more complex tasks. In the Aquatic experiment, both the recall and F1 score for the minority classes improve notably (F1 from 0.970 to 0.976 for *Hair-*

Ground ⊓ Quadrupedal	⊑ Terrestrial	Vegetation	⊑ ∃sustained_by.Plants	∃sustained_by.Plants	⊑ Herbivore
Terrestrial ⊓ Herbivore	⊑ Grazer	Hooves	⊑ ∃has_part.Hoof_part	∃has_part.Hoof_part	⊑ Ungulate_form
Terrestrial ⊓ Ungulate_form	⊑ Ungulate	Not_hooves	⊑ Not_ung_form	Swims ⊓ Water	⊑ Water_mobile
∃moves_via.Swim_mode	⊑ Aquatic_nature	Water_mobile	⊑ ∃moves_via.Swim_mode	Aquatic_nature ⊓ Not_ung_form	⊑ Aquatic
Not_swims ⊓ Not_water	⊑ Not_water_mobile	Not_water_mobile	⊑ ∃moves_via.Not_swim_mode	∃moves_via.Not_swim_mode	⊑ Not_aquatic_nat
∃has_part.Hand_part	⊑ Primate_1	Hand	⊑ ∃has_part.Hand_part	Not_ung_form ⊓ Not_aquatic_nat	⊑ Primate_2
Primate_1 ⊓ Primate_2	⊑ Primate				

Figure 1: The 19 GCIs from the **Combined** experiment.

less_Aquatic, and 0.935 to 0.939 for *Furry_Aquatic*). This shows that end-to-end training helped the model learn to better discriminate between the more nuanced subclasses. The most significant improvement happened in the Ungulate experiment. The model trained from scratch shows a considerable gain in performances in the *Horned_Ungulate* class, with its F1 score boosting from 0.949 to 0.970 and its recall from 0.943 to 0.965; the lack of intersection of the confidence intervals confirms that this difference is indeed significant. Overall, the models trained end-to-end in DeepEL seem to learn visual features that better adhere to the logical rules, a characteristic not encouraged in the individual pre-trained MLPs.

When looking at misclassifications, there may be two explanations. First, a class may satisfy some but not all logical requirements (e.g., in the Carnivore experiment, collies eat meat but are not hunters). Second, some classes are visually similar to classes in other categories (e.g., in the Terrestrial experiment, bats are visually similar to mice and rats). The most commonly misclassified classes were:

- **Terrestrial**: otter, spider monkey, and bat.
- **Carnivore**: rat, collie, and otter.
- **Aquatic**: otter, hippopotamus, and beaver.
- **Ungulate**: moose, cow, and pig.

This suggests the perceptual error is propagated through the logical rules to an incorrect final conclusion. It is clear that in this case the bottleneck for DeepEL is the perceptual component, which is expected in a neuro-symbolic system. Yet, as these examples suggest, we can more easily explain errors, thus being able to separate perception (classification) errors, from reasoning issues, and adopt adequate correction strategies given the case.

To conclude, DeepEL is able to successfully integrate neural perception and reasoning over $\mathcal{EL}$ ontologies. Consequently, the underlying upper bound for the performances of any DeepEL model is given by the expressiveness of the sub-symbolic data used, how well engineered the knowledge base is, and the capabilities of the neural networks employed.

6 Conclusions

We have introduced DeepEL, a neuro-symbolic extension of the light-weight description logic $\mathcal{EL}^{\perp}$. The goal of this new language is to allow for formal symbolic reasoning (with correctness guarantees) over instances whose properties are perceived and analysed by a neural classifier. In order to test these ideas, we proposed an implementation based on a reduction to DeepProbLog, which allowed us to include, beyond normal inferences, also a learning module.

We tested this prototypical implementation using the AWA2 dataset and five simple knowledge bases, which allow us to gauge the overall behaviour of the system. Our results show that, whether one starts with a pre-trained classifier, or trains the classifier from the implicit conclusions which can be obtained from reasoning, the neuro-symbolic system tends to behave well overall. We envision, still, the use of pre-trained classifiers as one of the main applications to our approach, as there is no need to re-train or re-label datasets if the knowledge of a domain is updated (as often happens for live KBs).

One advantage of using this disentangled neuro-symbolic approach is that it allows to interpret the results and trace back the reasons for an incorrect inference, so that the logical component is corrected, or the classifier fine-tuned, if needed. For this, we plan to study efficient methods to derive the most probable explanation (from the class of probabilistic facts) responsible for the error, and as hinted in Section 3 the posterior probability of observations given further evidence.

As future work, we intend to further optimise the approach and implementation by using modularity approaches to reduce the KB before inference (Suntisrivaraporn 2008) and understand the bottlenecks that can be opened through a more specific implementation. We also plan to evaluate the benefits of training over high-level classes to extract the low-level properties of instances.

Acknowledgments

Ignacio Huitzil received support from the Projects PID2024-159530OB-I00 (funded by MICIU/AEI/10.13039/501100011033/FEDER, UE) and UZ2024-IyA-02 (funded by University of Zaragoza).

AI Declaration

The authors have not employed any Generative AI tools.

References

Archana, R., and Jeevaraj, P. S. E. 2024. Deep learning models for digital image processing: a review. *Artif. Intell. Rev.* 57(1):11.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2nd edition.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. of 18th Int. Joint Conference on Artificial Intelligence (IJCAI 2005)*, 364–369. Professional Book Center.

Bourgaux, C.; Ozaki, A.; and Peñaloza, R. 2023. Semiring provenance for lightweight description logics. *CoRR* abs/2310.16472.

Brown, L. D.; Cai, T. T.; and DasGupta, A. 2001. Interval estimation for a binomial proportion. *Statistical Science* 16(2):101–117.

Bryant, R. E. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Computers* 35(8):677–691.

Calautti, M.; Livshits, E.; Pieris, A.; and Schneider, M. 2024. The complexity of why-provenance for datalog queries. *Proc. ACM Manag. Data* 2(2):83.

Carral, D., and Krötzsch, M. 2020. Rewriting the description logic ALCHIQ to disjunctive existential rules. In Bessiere, C., ed., *Proc. of 29th Int. Joint Conference on Artificial Intelligence (IJCAI 2020)*, 1777–1783. ijcai.org.

Carral, D.; Dragoste, I.; and Krötzsch, M. 2020. Reasoner = logical calculus + rule engine. *Künstliche Intell.* 34(4):453–463.

Ceylan, İ. İ., and Peñaloza, R. 2017. The Bayesian ontology language BEL. *J. Autom. Reason.* 58(1):67–95.

Ceylan, İ. İ.; Mendez, J.; and Peñaloza, R. 2015. The Bayesian ontology reasoner is BORN! In Dumontier, M.; Glimm, B.; Gonçalves, R. S.; Horridge, M.; Jiménez-Ruiz, E.; Matentzoglu, N.; Parsia, B.; Stamou, G. B.; and Stoilos, G., eds., *Proc. of the 4th International Workshop on OWL Reasoner Evaluation (ORE-2015)*, volume 1387 of *CEUR Workshop Proceedings*, 8–14. CEUR-WS.org.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 1321–1330. JMLR.org.

Hitzler, P.; Eberhart, A.; Ebrahimi, M.; Sarker, M. K.; and Zhou, L. 2022. Neuro-symbolic approaches in artificial intelligence. *National Science Review* 9(6):nwac035.

Huitzil, I.; Mazzotta, G.; Peñaloza, R.; and Ricca, F. 2023a. ASP-based axiom pinpointing for description logics. In Kutz, O.; Lutz, C.; and Ozaki, A., eds., *Proc. of the 36th International Workshop on Description Logics (DL 2023)*, volume 3515 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Huitzil, I.; Mazzotta, G.; Peñaloza, R.; and Ricca, F. 2023b. OWL2ASP tool.

Kimmig, A.; Demoen, B.; Raedt, L. D.; Costa, V. S.; and Rocha, R. 2011. On the implementation of the probabilistic logic programming language ProbLog. *Theory Pract. Log. Program.* 11(2-3):235–262.

Manhaeve, R.; Dumancic, S.; Kimmig, A.; Demeester, T.; and Raedt, L. D. 2021. Neural probabilistic logic programming in DeepProbLog. *Artif. Intell.* 298:103504.

Mohamed, R.; Bouraoui, Z.; Loukil, Z.; and Gargouri, F. 2022. Min-based conditioning of possibilistic EL ontology. In Barták, R.; Keshtkar, F.; and Franklin, M., eds., *Proceedings of the Thirty-Fifth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2022*. Florida Online Journals.

Qi, G.; Pan, J. Z.; and Ji, Q. 2007. Extending description logics with uncertainty reasoning in possibilistic logic. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, 828–839. Springer.

Raedt, L. D.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A probabilistic prolog and its application in link discovery. In Veloso, M. M., ed., *Proc. of 20th Int. Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2462–2467.

Robinson, J. A. 1965. A machine-oriented logic based on the resolution principle. *J. ACM* 12(1):23–41.

Sarker, M. K. 2022. *Neuro-symbolic artificial intelligence: The state of the art*. SAGE Publications Limited.

Spackman, K. A. 2007. An examination of OWL and the requirements of a large health care terminology. In Golbreich, C.; Kalyanpur, A.; and Parsia, B., eds., *Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions*, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Spackman, K. A. 2008. Using DL to support a very large healthcare terminology: Successes and challenges. In Baader, F.; Lutz, C.; and Motik, B., eds., *Proceedings of the 21st International Workshop on Description Logics (DL2008)*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Suntisrivaraporn, B. 2008. Module extraction and incremental classification: A pragmatic approach for ontologies. In Bechhofer, S.; Hauswirth, M.; Hoffmann, J.; and Koubarakis, M., eds., *Proc. of 5th European Semantic Web Conference, ESWC 2008*, volume 5021 of *Lecture Notes in Computer Science*, 230–244. Springer.

Xian, Y.; Lampert, C. H.; Schiele, B.; and Akata, Z. 2019. Zero-shot learning - A comprehensive evaluation of the good, the bad and the ugly. *IEEE Trans. Pattern Anal. Mach. Intell.* 41(9):2251–2265.