EM-NeSy: Expectation Maximization for Neurosymbolic Learning

Annegret Seibt
Department of Computer Science
KU Leuven
annegret.seibt@kuleuven.be

Luc De Raedt
Department of Computer Science
KU Leuven
luc.deraedt@kuleuven.be

Giuseppe Marra
Department of Computer Science
KU Leuven
giuseppe.marra@kuleuven.be

Abstract

Neurosymbolic (NeSy) models integrate neural networks and symbolic reasoning for robust and interpretable AI. State-of-the-art NeSy models require that the symbolic component is expressed in a differentiable way, often complicating the use of approximate inference. We propose EM-NeSy which casts predictive probabilistic NeSy learning as an instance of the Expectation-Maximization (EM) algorithm. In the expectation step, we compute the posterior over the neurally predicted symbols conditioned on the label via probabilistic inference. In the maximization step, we update the neural parameters based on this posterior using gradient descent only through the neural component. This formulation unlocks the full potential of the EM algorithm for predictive NeSy learning. It allows NeSy to extend naturally to approximate reasoning without any additional modifications or differentiability requirements of the symbolic component. Furthermore, it recovers the standard end-to-end gradient-based NeSy setting under exact inference. Our experimental results demonstrate the scalability and computational efficiency of EM-NeSy.

1 Introduction

Neurosymbolic (NeSy) models combine the perceptual strengths of neural networks with the reasoning capabilities of symbolic logic, enabling systems to process raw sensory data while maintaining structured, interpretable reasoning over high-level concepts. The integration of continuous neural networks and discrete symbolic logic can be realized through numeric [16, 19, 59, 37, 21, 20, 43, 44, 55] or fuzzy [47, 54, 3, 42] relaxations of logic statements. Alternatively, De Raedt [11] coined the formula

$$\text{Neurosymbolic} = \text{Neural} + \text{Logical} + \text{Probabilistic},$$

highlighting probabilistic semantics as a natural choice for linking neural and logical components, as they provide a principled way to model uncertainty over neural network outputs. This approach is now widely adopted by state-of-the-art NeSy models [58, 34, 60, 26, 36, 57], which typically implement it by performing probabilistic inference over the outputs of neural networks. This enables them to train their neural component in an end-to-end fashion, i.e. they backpropagate gradients first through the symbolic and then the neural component.

While probabilities are easily differentiable, exact probabilistic inference is #P-hard and therefore often computationally infeasible – and often not necessary during training since sufficient learn-

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

ing signals can be obtained by approximate inference. However, when resorting to approximate inference methods, differentiability can become challenging. The case of sampling, which is inherently non-differentiable, has been extensively studied and several specialized approaches have been proposed, including gradient estimators [48, 14, 33] and reparameterization-based techniques [29]. But differentiability issues might also arise in iterative approximate inference methods like Pearl's belief propagation algorithm [40] and dual decomposition and linear-programming-based relaxations for approximate maximum-a-posteriori (MAP) inference [28, 46] – which are theoretically differentiable but, in practice, backpropagating through all their iterations would introduce substantial computational overhead.

To overcome this reliance on differentiable symbolic reasoning, we identify a connection between standard end-to-end differentiable NeSy learning and the Expectation-Maximization (EM) algorithm: under exact inference, the former can be equivalently reformulated as an instance of the latter (see theorem 3.1). Following this insight, we propose a novel learning approach for predictive probabilistic NeSy systems decomposing into alternating expectation and maximization steps [17]. In every expectation step (E-step), a posterior distribution over the neurally predicted symbolic variables conditioned on the target variable label is computed. In every maximization step (M-step), the parameters of the neural component are then updated based on the posterior computed in the E-step via gradient descent solely through the neural component itself. Hence, irrespective of differentiability, the E-step can be carried out using any symbolic inference engine, whether exact or approximate. This is a major operational advantage, as any specialized and optimized software can be employed, regardless of the symbolic framework.

In summary, our work makes the following key contributions:

- We recast predictive probabilistic NeSy learning within the – traditionally fully probabilistic – Expectation-Maximization (EM) algorithm framework, enabling the use of advanced EM variants such as Generalized EM [17], Stochastic EM [6] and Hard EM [45] for improved flexibility and scalability;
- We show that, under exact inference, EM-NeSy learning yields the same gradients as end-to-end neuro-symbolic learning;
- We propose a general framework for integrating any probabilistic inference engine into a NeSy model without requiring differentiability of the symbolic inference process;
- We provide experimental evaluation demonstrating the effectiveness and generality of the proposed approach across diverse exact and approximate inference methods.

This paper is structured as follows: In section 2, we review the standard NeSy learning setting and the EM algorithm. Section 3 presents our EM-NeSy learning framework, including a proof of its equivalence to standard NeSy learning under exact inference and an EM-specific approximation to accelerate convergence. section 4 demonstrates how the framework generalizes to alternative inference schemes, with a particular focus on sampling-based and iterative approaches. In section 5 we discuss related work. Finally, section 6 reports experimental results.

2 Preliminaries

2.1 The Expectation-Maximization algorithm

The Expectation-Maximization (EM) algorithm, introduced by Dempster et al. [17], is an iterative algorithm for computing maximum likelihood estimates in parameterized probabilistic models with latent variables. Consider a latent variable model parameterized by $\boldsymbol{\theta}$ with observed variables $\mathbf{Y}$ and latent variables $\mathbf{Z}$. Let $\mathbf{y}$ denote an instantiation of $\mathbf{Y}$ and $R(\mathbf{Z})$ the range, i.e. the set of all instantiations, of $\mathbf{Z}$. Direct maximization of the marginal likelihoods $\log p(\mathbf{y} \mid \boldsymbol{\theta}) = \log \sum_{\mathbf{z} \in R(\mathbf{Z})} p(\mathbf{y}, \mathbf{z} \mid \boldsymbol{\theta})$ is often intractable due to the summation over the latent variables $\mathbf{Z}$. EM addresses this by alternating between two steps: The E-step computes the expected complete-data log-likelihood under the current parameter estimate $\boldsymbol{\theta}^{(t)}$

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = \mathbb{E}_{\mathbf{z} \sim \mathbf{Z} \mid \mathbf{Y}, \boldsymbol{\theta}^{(t)}} \big[\log p(\mathbf{Y}, \mathbf{z} \mid \boldsymbol{\theta}) \big]. \tag{1}$$

Note that $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$ is a function of $\boldsymbol{\theta}$; therefore, the actual computation in the E-step consists of determining the posterior distribution $p(\mathbf{Z} \mid \mathbf{Y}, \boldsymbol{\theta}^{(t)})$. The M-step updates the parameters by

maximizing the expectation (1):

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}). \tag{2}$$

These steps are repeated until convergence to a local optimum. The algorithm guarantees that the likelihood does not decrease at each iteration and is widely used in probabilistic models, such as mixture models and hidden Markov models.

A common extension is the Generalized EM (GEM) algorithm [17], which relaxes the M-step requirement: instead of fully maximizing the Q-function (1), GEM only requires an update that increases it. This flexibility enables variants such as

- **Hard EM** [45] which instead of computing the expectation in (1), only computes the MAP assignment $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})_{\text{hard}} = \max_{\mathbf{z} \in R(\mathbf{Z})} \log p(\mathbf{Y}, \mathbf{z} \mid \boldsymbol{\theta})$;
- **Online EM** [5] or **Stochastic EM** [6] which only use a single observation $\{\mathbf{Y} = \mathbf{y}\}$ or mini-batches instead of the entire dataset to compute the Q-function $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})_{\text{online}} = \mathbb{E}_{\mathbf{Z} \mid \mathbf{Y}=\mathbf{y}, \boldsymbol{\theta}^{(t)}} \big[\log p(\mathbf{Y} = \mathbf{y}, \mathbf{Z} \mid \boldsymbol{\theta}) \big]$ in the E-step. Consequently, it only performs a partial update $\boldsymbol{\theta}^{(t+1)}_{\text{online}} = \boldsymbol{\theta}^{(t)} + \gamma^{(t)} \big(\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)} \big)$ in the M-step for some learning rate $\gamma^{(t)}$, making the algorithm suitable for large-scale or streaming data – and also to NeSy learning;
- **Multi-M EM**, a variant of online EM that performs multiple partial M-steps per E-step.

2.2 Probabilistic NeSy Learning as a Latent Variable Model

State-of-the-art NeSy models combine a neural and a symbolic component by performing probabilistic inference over the outputs of neural networks to predict the distribution of a queried target variable [34, 26, 57].

Example 2.1 (Visual Sudoku [2]). Suppose, for instance, we have a dataset where the input is a 9×9 grid of handwritten digits and the label is whether they form a valid Sudoku but there is no direct supervision on single digits. A natural approach to train a neurosymbolic model on such a dataset is as follows (see fig. 1b): First we employ a neural component to classify each handwritten digit, producing a probability distribution over the possible values $\{0, \ldots, 8\}$ for each digit. Then a symbolic component that encodes the rules of Sudoku can use these probabilistic digit predictions to compute the likelihood that the entire grid constitutes a valid Sudoku configuration. Finally, this likelihood can be compared to the provided label (valid or invalid) to define a loss function, allowing the entire system to be trained end-to-end. In other words, the model learns to adjust the neural predictions so that the probability of satisfying the Sudoku constraints aligns with the ground truth.

Consider the general learning task of predicting a symbolic output variable $\mathbf{Y}$ from a subsymbolic input variable $\mathbf{X}$, given a dataset of observed samples and some background knowledge. Then we can interpret a NeSy model solving this task as a latent variable model (see fig. 1a) with a subsymbolic input variable $\mathbf{X}$, a hidden symbolic variable $\mathbf{Z}$ [1] representing the output of the neural component and a symbolic output variable $\mathbf{Y}$ that can be inferred from $\mathbf{Z}$ using the symbolic component that encodes the background knowledge.

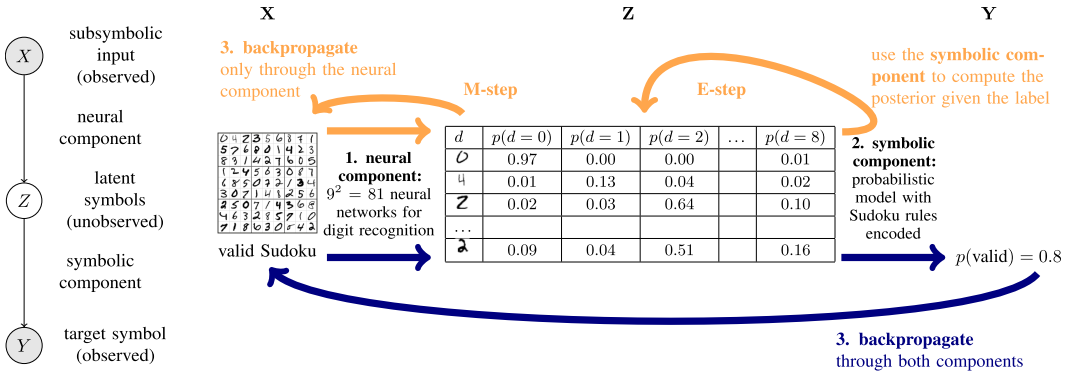
The symbolic component is a probabilistic model that predicts the target variable $\mathbf{Y}$ from the latent symbolic variable $\mathbf{Z}$ via probabilistic inference, i.e. by computing the distribution $p_{\text{Sy}}(\cdot|\mathbf{z})$ of the output variable $\mathbf{Y}$ conditioned on the latent variable $\mathbf{Z}$ taking the state $\mathbf{z}$. Meanwhile, the neural component serves as a conditional prior for the symbolic component, by predicting the distribution $p_{\text{Ne}}(\cdot|\mathbf{x}, \boldsymbol{\theta})$ of the latent variable $\mathbf{Z}$ given the subsymbolic input $\mathbf{x}$ via neural networks $\mathbf{N}$ parameterized by $\boldsymbol{\theta}$.

Thus, we can abstractly represent the neurosymbolic predictor as a conditional probability distribution

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = \sum_{\mathbf{z} \in R(\mathbf{Z})} p(\mathbf{y}, \mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = \sum_{\mathbf{z} \in R(\mathbf{Z})} p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) p_{\text{Sy}}(\mathbf{y}|\mathbf{z}), \tag{3}$$

where we denote by $R(\mathbf{Z}) = R(Z_1) \times \cdots \times R(Z_n)$ the range (i.e. the set of all possible values) of $\mathbf{Z}$. The loss function in the standard NeSy setting is defined on the target-variable level as the negative

[1] In the following, we assume $\mathbf{Z}$ to be discrete. The theory developed, however, extends naturally to the continuous case by replacing summations with integrals.



d	$p(d=0)$	$p(d=1)$	$p(d=2)$	…	$p(d=8)$
0	0.97	0.00	0.00		0.01
4	0.01	0.13	0.04		0.02
7	0.02	0.03	0.64		0.10
…					
2	0.09	0.04	0.51		0.16

(a) The probabilistic NeSy model is a latent variable model.

(b) To train on Visual Sudoku [2], an **end-to-end differentiable NeSy** model neurally predicts 81 digit distributions, computes the Sudoku-validity probability via the symbolic model, and backpropagates through both parts. **EM-NeSy** also neurally predicts prior distributions for all digit, but uses the symbolic model to obtain posterior digit distributions conditioned on the label and trains the neural predictors by matching priors to posteriors, avoiding gradients through the symbolic component. (Sudoku grid and digits from [2].)

Figure 1: Comparison of end-to-end differentiable NeSy learning and EM-NeSy learning.

log-likelihood

$$\mathcal{L}(\boldsymbol{\theta}) = -\log p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) = -\log \sum_{\mathbf{z}\in R(\mathbf{Z})} p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}) p_{\text{Sy}}(\mathbf{y}|\mathbf{z}). \tag{4}$$

It is typically minimized by backpropagating gradients through the symbolic and the neural component. Note that learnable parameters could also be introduced to the symbolic component and be jointly optimized with the parameters of the neural component.

3 EM-NeSy Learning

n In this section, we will follow the interpretation of predictive[2] probabilistic NeSy as a conditional latent variable model from section 2.2 and introduce our EM-NeSy learning framework. Compared to the standard NeSy setting, we will change the use of the symbolic component during learning: instead of letting it predict the log-likelihood of the label given the neurally predicted latent variables, we will use it to compute the posterior distribution of these neurally predicted latent variables given the label:

$$p(\mathbf{z}|\mathbf{y},\mathbf{x},\boldsymbol{\theta}) = \frac{p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}) p_{\text{Sy}}(\mathbf{y}|\mathbf{z})}{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}. \tag{5}$$

Hence, the symbolic component in EM-NeSy learning acts as a probabilistic corrector, refining and aligning the neural network outputs with the background knowledge. It can be thought of as abducing labels

$$\ell(\mathbf{z}) = p(\mathbf{z}|\mathbf{y},\mathbf{x},\boldsymbol{\theta}) \tag{6}$$

for the neurally predicted latent variables $\mathbf{Z}$ from the prior distribution $p_{\text{Ne}}(\cdot|\mathbf{x},\boldsymbol{\theta})$ and the original label $\mathbf{y}$ of the target variable, which is related to abductive reasoning [41, 10] in general. Note that these abduced labels can be soft probabilistic labels as in (6), or alternatively hard labels in the case of MAP inference: $\ell(\mathbf{z}) = \mathbb{I}(\mathbf{z} = \arg\max_{\mathbf{z}'} p(\mathbf{z}'|\mathbf{y},\mathbf{x},\boldsymbol{\theta}))$, where $\mathbb{I}$ denotes the indicator function. Such a MAP-based abduction is for example performed by Relational Neural Machines [35]. This symbolic correction step will be the E-step of our EM-NeSy learning algorithm. It can be carried out by any exact or approximate symbolic inference engine and regardless of its differentiability.

The labels abduced in the E-step can now serve to define a loss function on the latent-variable level: the cross entropy between the neurally predicted prior distributions $p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$ of the latent variables

[2] In this paper, we focus on *predictive* probabilistic NeSy systems. Nevertheless, EM-NeSy naturally extends to generative architectures (see section D).

and their posterior $\ell(\mathbf{z})$, yielding the expected complete-data log-likelihood of the conditional-prior model:

$$\mathcal{L}_{\text{EM-NeSy}}(\boldsymbol{\theta}) = -\sum_{\mathbf{z}\in R(\mathbf{Z})} \ell(\mathbf{z}) \log p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}). \tag{7}$$

The M-step of our EM-NeSy learning algorithm uses this loss function to updates the parameters $\boldsymbol{\theta}$ of the neural component by backpropagating only through the neural component itself, while treating the posterior $\ell(\mathbf{z})$ as a constant. The entire EM-NeSy algorithm then simply consists of alternating the expectation and maximization steps. For a high-level visualization, see fig. 1b; for a worked example of the E-step, see section A.

Under exact inference, EM-NeSy learning recovers the standard NeSy learning setting described in section 2.2, in the sense that it backpropagates the same gradients through the neural component.

Theorem 3.1. *The gradients with respect to the neural parameter θ of the loss $\mathcal{L}(\theta)$ in the standard NeSy learning setting of eq.* (4) *and the loss $\mathcal{L}_{\textit{EM-NeSy}}(\theta)$ in the EM-NeSy learning setting of eq.* (7) *are equal, i.e.*

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\textit{EM-NeSy}}(\boldsymbol{\theta}). \tag{8}$$

Proof. see Appendix B □

In practice, EM-NeSy learning follows an online-EM style approach, where each E-step processes a single training pair $(\mathbf{x}, \mathbf{y})$ (or a mini-batch) rather than the entire dataset, and the M-step performs a partial parameter update, scaled by the learning rate of the neural component. As noted earlier, the abduced labels (6) may be either soft (probabilistic) or hard (MAP), leading to a soft or hard variant of EM-NeSy learning.

A further approximation of EM-NeSy learning, besides the possible use of an approximate inference component can be achieved by a Multi-M EM approach (see Appendix C): Every additional back-propagation step for the same training example $(\mathbf{x}, \mathbf{y})$ with regard to the same loss function saves one potentially expensive re-computation of the posterior of the latent variables. Note that performing these multiple small steps leads to a slightly more stable learning process than just increasing the learning rate as the latter can cause instability due to overshooting the optimum. Finally, also in EM-NeSy, learnable parameters can be introduced into the symbolic component and optimized jointly with the neural component's parameters that predict the latent variables $\mathbf{Z}$ [35].

4 Generalization to alternative inference schemes

To update the neural parameters in the M-step, we only need the posteriors (5) of the latent variables, i.e. the outcome of the probabilistic inference in the E-step, but not the gradients of this computation. Hence, the E-step can be carried out by any symbolic inference engine that can compute posterior marginals or the MAP assignment, regardless of whether it performs exact or approximate inference in a differentiable or non-differentiable fashion. *Thus, in general, EM-NeSy learning is applicable to any inference scheme, whether differentiable or not*. In this section, we will explore when it is especially useful.

4.1 Exact inference

In NeSy models, exact inference is often performed by compiling the symbolic component into an arithmetic circuit in an offline knowledge-compilation step [7, 58, 34, 1]. When this is feasible, gradient backpropagation through the symbolic component becomes highly efficient, as it only requires a single backward pass through the circuit. Recall from section 3 that we established the equivalence between the gradients of the standard-NeSy loss and the EM-NeSy loss. This implies that differentiating through the symbolic component is effectively the same as computing posterior marginals. So, put differently, backpropagation through the circuit provides a fast and direct way to obtain the posterior marginals required for EM-NeSy learning. This link between differentiating and computing posterior marginals was already noted by Darwiche [9], who leveraged the same equivalence for the opposite purpose: proposing a differentiable approach to efficiently compute marginals in Bayesian networks. Consequently, when exact inference with arithmetic circuits is possible, EM-NeSy does not offer additional benefits in terms of differentiability, as there are no differentiability challenges in this setting.

However, knowledge compilation can be prohibitively expensive, and in some cases a circuit may fail to compile within a reasonable time even though other exact inference methods remain feasible. In such situations, EM-NeSy can be used to eliminate any potential differentiability issues. Furthermore, even if knowledge compilation is possible but perhaps yields a large arithmetic circuit, Multi-M EM-NeSy learning may still be useful to accelerate convergence without introducing instability issues, such as those caused by increasing the learning rate.

4.2 Sampling-based inference

Although sampling is inherently non-differentiable, several gradient estimators have been proposed for sampling-based inference in NeSy models. The most straightforward approach is the score function estimator, commonly known as REINFORCE [48]. This method relies on a simple application of the "delta-log trick" (10) to rewrite the gradient of the log-likelihood of the label as

$$\nabla_{\boldsymbol{\theta}} p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) \overset{(3)}{=} \sum_{\mathbf{z}\in R(\mathbf{Z})} p_{\text{Sy}}(\mathbf{y}|\mathbf{z}) \nabla_{\boldsymbol{\theta}} p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}) \overset{(10)}{=} \mathbb{E}_{\mathbf{z}\sim p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})}[p_{\text{Sy}}(\mathbf{y}|\mathbf{z}) \nabla_{\boldsymbol{\theta}} \log p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})], \quad (9)$$

thus enabling sampling from the neurally predicted distribution $p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$ of the latent variables. However, because each sample is weighted by $p_{\text{Sy}}(\mathbf{y}|\mathbf{z}) \nabla_{\boldsymbol{\theta}} \log p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$, the variance of REINFORCE can become large. Lower-variance alternatives include a Rao–Blackwellized version of REINFORCE [14] and model-sampling [33, 52], which reduce variance by exploiting conditional expectations or structured sampling strategies. Still, these methods fundamentally rely on reformulating the neurosymbolic gradient $\nabla_{\boldsymbol{\theta}} p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})$. In contrast, NeSy-EM learning avoids such case-specific rewritings. It only requires estimates of the posterior marginals $p(\mathbf{z}|\mathbf{y},\mathbf{x},\boldsymbol{\theta})$, which can be obtained using any sampling method – such as rejection sampling, importance sampling, Gibbs sampling and Metropolis-Hastings [30] – without additional adjustments to the NeSy setting.

Rejection sampling. The most straightforward way to approximate a posterior distribution by sampling is rejection sampling. It operates by drawing candidate samples from the prior distribution and then accepting or rejecting each sample based on whether it agrees with the observed data. Only the accepted samples are kept, and these form an empirical approximation of the posterior. If we employ rejection sampling to EM-NeSy, then we almost recover the REINFORCE gradient estimator:

$$\begin{aligned}\nabla_{\theta} \mathcal{L}_{\text{EM-NeSy}}(\boldsymbol{\theta}) &\overset{(7),(5)}{=} -\sum_{\mathbf{z}\in R(\mathbf{Z})} \frac{p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}) p_{\text{Sy}}(\mathbf{y}|\mathbf{z})}{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})} \nabla_{\theta} \log p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}) \\ &= \frac{1}{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})} \cdot \mathbb{E}_{\mathbf{z}\sim p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\theta)}[p_{\text{Sy}}(\mathbf{y}|\mathbf{z}) \nabla_{\boldsymbol{\theta}} \log p_{\text{Ne}}(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})].\end{aligned}$$

We would sample from the same distribution as in (9) and divide the estimates for all gradients by the likelihood of the observed label. This adjustment reflects the fact that REINFORCE approximates the gradient of the likelihood itself, whereas EM-NeSy targets the gradient of the log-likelihood which differs by the factor $p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})$ (see (10)).

ABC rejection sampling. Rejection sampling – and thus also REINFORCE – struggle in large sampling spaces because it rarely produces samples that satisfy the evidence. The same is the case for likelihood weighting [30] under determinism as the likelihood weights then become deterministic as well. Approximate Bayesian Computation (ABC) sampling [49, 4], which is closely related to Pseudo maximum-likelihood estimation [56, 22], provides a more flexible alternative to likelihood weighting. In likelihood weighting, we sample from the neurally predicted prior $p_{\mathbf{Ne}}(\cdot \mid \mathbf{x},\boldsymbol{\theta})$ and weigh each sample $\mathbf{z}$ with the likelihood value $p_{\text{Sy}}(\mathbf{y} \mid \mathbf{z})$. However, if the likelihood is intractable to compute or deterministic – as in the case of Sudoku – the likelihood weights become deterministic as well and tend to collapse to zero because the evidence is extremely sparse. (Sampling a valid 9×9 Sudoku configuration is highly unlikely.)

ABC sampling addresses this issue by introducing *summary statistics* $s_{\mathbf{Z}} : R(\mathbf{Z}) \to \mathcal{S}$ and $s_{\mathbf{Y}} : R(\mathbf{Y}) \to \mathcal{S}$, which map the potentially complex instantiations of $\mathbf{Z}$ and $\mathbf{Y}$ into a simpler comparison space $\mathcal{S}$ (e.g. $\mathcal{S} = \mathbb{R}$), and a *distance metric* $\rho : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, such as the L^1- and L^2-norm. In ABC sampling, the exact likelihood weight is replaced by the more flexible discrepancy measure $\frac{1}{C}\rho(s_{\mathbf{Z}}(\mathbf{z}),\ s_{\mathbf{Y}}(\mathbf{y}))$ (where C denotes a normalization constant), which allows inference to proceed even when the likelihood is unavailable or degenerate. A general strategy for selecting summary

statistics and discrepancy measures is as follows: We decompose the constraint[3] determining whether a sample is valid into a set of constraints M. For a given sample $\mathbf{z}$, let k denote the number of constraints in M that it satisfies, and let $s_{\mathbf{Z}}(\mathbf{z}) := \frac{k}{|M|}$. By choosing $s_{\mathbf{Y}}(\mathbf{y}) = 0$ and $\rho = L^1$, the weight assigned to the sample $\mathbf{z}$ is $\frac{1}{C}\rho(s_{\mathbf{Z}}(\mathbf{z}), s_{\mathbf{Y}}(\mathbf{y})) = \frac{1}{C}|\frac{k}{|M|} - 0| = \frac{k}{C|M|}$. Thus, the weight of a sample corresponds to the fraction of constraints it satisfies.

Example 4.1. In the case of Sudoku, M may consist of the constraints that certain pairs of digits – namely those sharing a row, column, or box – must be different. For an addition task, M could contain constraints specifying that each individual digit of the sum is correct. In a path-finding task on a grid, M could consist of constraints indicating whether each grid cell lies on the path or not.

A slightly more refined variant of ABC sampling can be obtained by decomposing the sample $\mathbf{z}$ into its individual components $(z_1, \ldots, z_n)$. For each component z_i, we then consider only the subset $M_i \subset M$ of constraints that depend most on Z_i. Let k denote the number of constraints in M_i) that are satisfied by z_i. We can then assign to each component z_i the weight $\frac{1}{C}\rho\big(s_{Z_i}(z_i), s_{\mathbf{Y}}(\mathbf{y})\big) = \frac{1}{C}|\frac{k}{|M_i|} - 0| = \frac{k}{C|M_i|}$, which corresponds to the fraction of *relevant* constraints that z_i satisfies.

For more details, see section E.3.

4.3 Iterative inference algorithms

Differentiability challenges can also arise in iterative approximate inference methods such as belief propagation [40], and dual decomposition and linear-programming-based relaxations for MAP inference [28, 46]. While these algorithms are differentiable in theory, backpropagating through their iterations in practice often requires maintaining a large computation graph or resorting to expensive techniques such as gradient checkpointing and implicit differentiation. In contrast, EM-NeSy learning removes this overhead entirely, as it only requires posterior marginals rather than differentiating through the inference procedure. Conveniently, belief propagation computes all posterior marginals or MAP assignments during inference, making them immediately available for use without any additional steps.

5 Related Work

While the equivalence of computing gradients and posterior marginals on which the proof in section 3 is based, has already been exploited by Darwiche to substitute exact inference with differentiation in Bayesian networks [9], we use it to do the converse: to replace differentiation by (approximate) inference.

A notable special case of EM-NeSy learning is found in Relational Neural Machines (RNMs) [35], a neurosymbolic framework that jointly learns neural and symbolic parameters while performing relaxation-based approximate MAP inference via gradient descent. Although RNMs employ the EM algorithm to avoid the computational overhead of differentiating through this iterative inference procedure, they do not introduce EM as a general framework for NeSy learning.

A closely related framework to EM-NeSy learning is Implicit Maximum Likelihood Estimation (I-MLE) [39], a general framework for backpropagating through discrete exponential-family distributions. I-MLE operates by correcting the neurally predicted distribution $p_{\boldsymbol{\theta}}(\cdot)$ by inferring from the label $\mathbf{y}$ a *target distribution* $q_{\boldsymbol{\theta}'}(\cdot)$, and then minimizing the KL-divergence between $p_{\boldsymbol{\theta}}(\cdot)$ and $q_{\boldsymbol{\theta}'}(\cdot)$ – an approach conceptually similar to EM-NeSy learning. However, EM-NeSy offers an actual general-purpose choice for the target distribution $q_{\boldsymbol{\theta}}$: the posterior $p(\mathbf{z}|\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})$, thereby establishing a direct connection to the well-studied EM algorithm. Moreover, EM-NeSy provides a more general framework: while I-MLE prescribes a specific perturb-and-MAP inference strategy which relies on the availability of a MAP-solver, EM-NeSy simply requires an approximation of the posterior, a problem that has been extensively studied. This flexibility allows EM-NeSy to integrate any black-box inference engine.

Another related neurosymbolic framework is EXAL [52], which is based on sampling and follows an approach similar to I-MLE. Given the label $\mathbf{y}$, the method samples a set of explanations, namely

[3] A similar idea is explored in NeSyDM [50], which derives an RLOO optimization objective from a NELBO, using the fraction of satisfied constraints as the reward signal.

instantiations of the latent variables $\mathbf{Z}$. Reweighting these samples using the neurally predicted distribution $p_{\text{Ne}}(\mathbf{z}|\boldsymbol{\theta})$ yields an *auxiliary distribution* $q_{\boldsymbol{\theta}}(\mathbf{x})$. The model parameters are then optimized by minimizing the KL divergence between p and q. EM-NeSy replaces this auxiliary distribution with the full posterior, making EXAL an instance of the EM-NeSy framework that approximates the posterior via the auxiliary distribution.

While state-of-the-art NeSy models realize the integration of neural perception and symbolic reasoning by backpropagating gradients through the symbolic component [58, 34, 26, 1], Abductive Learning (ABL) [8, 27, 25] sidesteps the need for a differentiable symbolic component in a similar way like EM-NeSy learning: The symbolic component corrects neural predictions by abductive logic reasoning. While the neural predictions can be probabilities like in EM-NeSy, ABL is not a probabilistic model as it only uses these probabilities or weights to determine which neural outputs will be corrected, but the reasoning itself remains deterministic [27], while in EM-NeSy the abduction itself is probabilistic. The closest ABL-based method to EM-NeSy is Ambiguity-Aware ABL (A^3BL) [23], which computes posterior probabilities over outputs consistent with deterministic background knowledge in an EM framework. However, it can only handle deterministic knowledge and depends on exact abductive inference, limiting scalability.

6 Experiments

To assess the scalability and flexibility of EM-NeSy, we conduct a set of experiments[4] addressing the following questions: **Q1**: Under *exact inference*, does EM-NeSy recover the same learning signal as end-to-end differentiable NeSy while reducing memory and runtime? **Q2**: How does EM-NeSy behave when the E-step is performed with *approximate or non-differentiable inference*?

6.1 Benchmarks

To answer our questions, we conduct experiments on three widely used NeSy benchmarks that all present weakly supervised tasks combining visual perception with symbolic reasoning.

Multi-Digit MNISTAdd [34]. Each input consists of a sequence of $2n$ MNIST [18] images, for some $n \geq 1$, representing an addition task where two numbers of equal length are summed (e.g., $495 + 037$ for $n = 3$). The dataset provides only the final sum as the label for each sequence, without individual digit annotations.
Visual Sudoku. [2] Each input corresponds to a 4×4- or 9×9-grid of MNIST images representing a Sudoku puzzle. Supervision is limited to a single binary label indicating whether the grid constitutes a valid Sudoku configuration, without explicit guidance on individual cell values.
Warcraft path-planning [53]. Each input consists of a visual grid of 12×12- or 30×30 cells of five different terrain types with different costs of traversing them. Supervision is a binary mask indicating the shortest path from the top-left to the bottom-right corner.

6.2 Baselines

We compare EM-NeSy against state-of-the-art NeSy models[5], namely A-NeSI [51], I-MLE [39], EXAL [52], Scallop [26] and exact inference based on knowledge compilation [57, 58, 1], noting that not all methods are evaluated on every benchmark. The exact inference methods we use (namely DeepStochLog (DSL) [57], Semantic Loss (SL) [58] and Semantic probabilistic Layer (SPL) [1]) as well as Scallop which employs a top-k approximaization are end-to-end differentiable and rely on knowledge compilation, which limits scalability. They are ill-suited for the path-planning task, where shortest-path computation is expressible but becomes a prohibitively large combinatorial argmin problem for logic-based differentiable inference. A-NeSI replaces symbolic inference with a neural approximation, but on MNIST-Addition we observe overflow when scaling to 100 digits. I-MLE and EXAL are more closely related to EM-NeSy in that they rely on sampling-based training objectives. However, neither I-MLE nor EXAL is applicable to the Visual-Sudoku task, as both require either a MAP solver or an efficient sampler over valid Sudoku solutions conditioned on predicted digit probabilities – which is prohibitively expensive. In EM-NeSy, we use multiple inference approaches:

[4]The code for reproducing the experiments is available at: https://github.com/ASeibt/emnesy.

[5]DeepSoftLog [32] is not included as a baseline because its contribution (soft unification as an interface between the neural and symbolic component) is orthogonal to EM-NeSy, which focuses on non-differentiable inference.

Table 1: Accuracy and training time on the MNISTAdd task.

	accuracy in $\%$			**training time in** min		
number of digits	4	15	100	4	15	100
A-NeSI	93.28 ± 0.25	55.88 ± 9.30	–	25.75	57.55	–
EXAL	90.71 ± 1.01	62.62 ± 4.24	6.67 ± 7.83	4.17	42.44	372.20
DSL[6](exact)	92.70 ± 0.60	T/O	T/O	–	T/O	T/O
BP-std (exact)	92.16 ± 0.23	$\mathbf{72.79 \pm 1.62}$	$\mathbf{11.60 \pm 3.85}$	2.21	7.93	36.26
BP-EM (ours)	92.53 ± 0.62	$\mathbf{74.85 \pm 1.83}$	$\mathbf{9.20 \pm 9.76}$	**1.33**	**1.95**	**8.51**
BP-EM-M (ours)	33.44 ± 33.49	30.36 ± 40.09	0.00 ± 0.00	**1.33**	**2.38**	13.09

Table 2: Accuracy and training time on the Visual Sudoku task.

	accuracy in $\%$		**training time in** min	
Sudoku size	4×4	9×9	4×4	9×9
A-NeSI	87.20 ± 2.18	$\mathbf{59.20 \pm 2.06}$	46.08	73.33
Scallop	75.00 ± 0.45	T/O	1.52	T/O
SL[6] (exact)	86.70 ± 0.50	T/O	–	T/O
BP-std	87.80 ± 2.05	0.50 ± 0.00	38.23	–
BP-EM (ours)	89.70 ± 3.05	0.50 ± 0.00	21.23	–
RBP-EM (ours)	87.10 ± 2.68	$\mathbf{59.50 \pm 2.67}$	**0.69**	**2.55**
ABC-EM (ours)	86.30 ± 1.30	53.30 ± 3.15	1.40	8.24

sum–product belief propagation (BP-EM), exact for MNIST-Addition and approximate for Visual Sudoku due to loopy underlying graphs; a relaxed version of belief propagation for Visual Sudoku (RBP-EM); max–product belief propagation (BP-EM-M), yielding hard EM; and ABC sampling (ABC-EM). We additionally include an end-to-end differentiable baseline with the same sum–product belief propagation inference (BP-std).

6.3 Results

In this section, we present our experimental results; for more details, see Appendix section E.

Q1: To evaluate EM-NeSy under exact inference, we use the MNISTAdd benchmark and compare EM-NeSy (BP-EM) with the standard NeSy learning setting (BP-std), both using sum-product belief propagation[7], and with DeepStochLog. The corresponding results are reported in table 1. As predicted by theorem 3.1, accuracies are nearly identical, with minor numerical differences resulting in increased variance for EM-NeSy at 100 digits. Note that in the 100-digit MNIST addition task the neural component has to classify 200 digit images. Hence minor differences in its performance (say accuracy 0.99 vs. 0.985) result in large differences of task accuracy ($0.99^{200} \approx 0.13$ vs. $0.985^{200} \approx 0.05$). Knowledge compilation prevents DeepStochLog from scaling to 15 digits, whereas EM-NeSy scales to 100 digits and is faster than end-to-end-differentiable NeSy since it avoids backpropagation through iterative updates and gradient-checkpointing overhead. This also leads to a substantial reduction in peak memory usage, from 90.61 GB to 35.52 GB at 100 digits. Hence, the answer to Q1 is that

[6]Accuracies for A-NeSI and Semantic Probabilistic Layer on Warcraft path-planning, and Semantic Loss on Visual Sudoku are taken from [51], accuracies for DeepStochLog on MNISTAdd and EXAL on Warcraft path-planning from [52].

[7]Belief propagation operates on a tree in this task and is therefore exact.

Table 3: Accuracy and training time on the Warcraft path-planning task.

	accuracy in $\%$		**training time in** min	
grid size	12×12	30×30	12×12	30×30
A-NeSI[6]	$\mathbf{98.96 \pm 1.33}$	67.57 ± 36.76	439.10	1596.51
I-MLE	95.34 ± 0.20	$\mathbf{93.4 \pm 0.64}$	26.77	227.82
EXAL[6]	94.19 ± 1.74	80.85 ± 3.83	**7.75**	58.75
SPL[6] (exact)	78.2	T/O	–	T/O
ABC-EM (ours)	$\mathbf{98.80 \pm 0.41}$	69.60 ± 35.64	**7.26**	**8.37**

under exact inference, EM-NeSy matches the accuracy of the standard end-to-end NeSy setting while offering faster training and decreased memory usage.

Q2: To evaluate EM-NeSy beyond exact inference, we instantiate the E-step with several approximate or non-differentiable inference procedures. The results are reported in table 2 and table 3. Unlike many other methods, EM-NeSy can seamlessly integrate different inference procedures without requiring differentiation through the symbolic component. This results in EM-NeSy scaling to all experimental settings where exact inference times out. Furthermore, it enables a direct comparison between different inference methods under the same learning algorithm. As expected, learning performance largely reflects the quality of the underlying inference procedure.

Hard EM (BP-EM-M), which uses MAP inference in the E-step, often converges to poor local optima on MNISTAdd. Because MAP inference only considers the single most probable explanation under the current neural prior, it can select incorrect explanations early in training when the prior is still largely uninformative. These assignments are then reinforced during learning, creating a feedback loop that biases the model toward its initial guesses. In the 100-digits MNISTAdd task, for example, BP-EM-M learns to reliably distinguish only nine digit classes, effectively merging the tenth class with another. Annealing may mitigate this issue by delaying hard decisions and preserving alternative explanations until the prior becomes more informative.

While ABC-EM achieves very fast runtimes and state-of-the-art accuracies on Visual Sudoku and 12×12 Warcraft maps, the variance increases substantially on 30×30 maps. This is likely due to the enormous sampling space and the small number of samples used: There are $5^{(30^2)}$ possible instantiations of the neural prior, while the approximation is based on only 20 samples, as each sample requires executing the relatively expensive Dijkstra algorithm. Hence the quality of the approximation, and therefore the quality of the supervision signal provided to the neural component, depends heavily on obtaining representative samples. ABC-EM therefore provides an attractive accuracy–efficiency trade-off, achieving very fast runtimes while incurring increased variance on large-scale problems.

Loopy belief propagation (BP-EM) fails on the 9×9 Sudoku grid because of the highly loopy factor graph structure. The relaxed variant of belief propagation (RBP-EM) however achieves state-of-the-art accuracy while keeping computation time minimal. This is possibly explained by the fact that a smaller number of iterations mitigates overconfident beliefs caused by cyclic dependencies.

With respect to Q2, these results show that EM-NeSy can be applied in settings where end-to-end differentiation is unavailable or impractical, while remaining computationally efficient. Overall, EM-NeSy is robust to moderately approximate E-steps, but increasing inference noise leads to higher variance and may eventually lead training to break down.

7 Conclusion

EM-NeSy shows that NeSy learning can be viewed as an instance of the EM algorithm. In contrast to state-of-the-art NeSy approaches – which typically require inference-specific modifications to ensure differentiability under approximate inference – EM-NeSy provides a unified learning framework that supports arbitrary approximate probabilistic inference methods by eliminating the need for differentiable symbolic inference. This substantially simplifies model design and enables the development of more scalable and flexible NeSy systems.

Limitations and future work While EM-NeSy accommodates both exact and approximate inference without differentiability requirements, it does not address how to select effective approximations. Further, EM-NeSy is developed and evaluated for predictive neuro-symbolic learning, as this represents the most common setting in probabilistic NeSy. We do not investigate architectures combining predictive and generative neural components [38, 12, 15]. Evaluating EM-NeSy in such settings is left for future work, although section D outlines a possible extension of the framework to these architectures. Finally, our experimental results depend on task-specific hyperparameter tuning.

Acknowledgements

This research received funding from the Flemish Government (AI Research Program), from the Flanders Research Foundation (FWO) under project G047124N and from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programm (grant agreement No. 101142702).

References

[1] Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, and Antonio Vergari. Semantic probabilistic layers for neuro-symbolic learning. *Advances in Neural Information Processing Systems*, 35:29944–29959, 2022.

[2] Eriq Augustine, Connor Pryor, Charles Dickens, Jay Pujara, William Wang, and Lise Getoor. Visual sudoku puzzle classification: A suite of collective neuro-symbolic tasks. In *International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*, 2022.

[3] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022.

[4] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[5] Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 71(3):593–613, 2009.

[6] Gilles Celeux, Didier Chauveau, and Jean Diebolt. Stochastic versions of the EM algorithm: an experimental study in the mixture case. *Journal of statistical computation and simulation*, 55(4):287–314, 1996.

[7] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799, 2008.

[8] Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. Bridging machine learning and logical reasoning by abductive learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[9] Adnan Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM (JACM)*, 2003.

[10] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. *IJCAI*, 2007.

[11] Luc De Raedt, Robin Manhaeve, Sebastijan Dumancic, Thomas Demeester, and Angelika Kimmig. Neuro-Symbolic= Neural+ Logical+ Probabilistic. In *NeSy@ IJCAI*, 2019.

[12] Lennert De Smet, Pedro Zuidberg Dos Martires, Robin Manhaeve, Giuseppe Marra, Angelika Kimmig, and Luc De Raedt. Neural probabilistic logic programming in discrete-continuous domains. In *Uncertainty in Artificial Intelligence*, pages 529–538. PMLR, 2023.

[13] Lennert De Smet, Pedro Zuidberg Dos Martires, Robin Manhaeve, Giuseppe Marra, Angelika Kimmig, and Luc De Readt. Neural probabilistic logic programming in discrete-continuous domains. In *Uncertainty in Artificial Intelligence*, pages 529–538. PMLR, 2023.

[14] Lennert De Smet, Emanuele Sansone, and Pedro Zuidberg Dos Martires. Differentiable sampling of categorical distributions using the catlog-derivative trick. *Advances in Neural Information Processing Systems*, 36:30416–30428, 2023.

[15] Lennert De Smet, Gabriele Venturato, Luc De Raedt, and Giuseppe Marra. Relational neurosymbolic markov models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 16181–16189, 2025.

[16] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1389–1399, 2016.

[17] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

[18] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[19] Michelangelo Diligenti, Marco Gori, and Claudio Sacca. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017.

[20] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. *arXiv preprint arXiv:1904.11694*, 2019.

[21] Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

[22] Christian Gourieroux, Alain Monfort, and Alain Trognon. Pseudo maximum likelihood methods: Theory. *Econometrica: journal of the Econometric Society*, pages 681–700, 1984.

[23] Hao-Yuan He, Hui Sun, Zheng Xie, and Ming Li. Ambiguity-aware abductive learning. In *Forty-first International Conference on Machine Learning*, 2024.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] Wen-Chao Hu, Wang-Zhou Dai, Yuan Jiang, and Zhi-Hua Zhou. Efficient rectification of neuro-symbolic reasoning inconsistencies by abductive reflection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17333–17341, 2025.

[26] Jiani Huang, Ziyang Li, Binghong Chen, Karan Samel, Mayur Naik, Le Song, and Xujie Si. Scallop: From probabilistic deductive databases to scalable differentiable reasoning. *NeurIPS*, 2021.

[27] Lin-Han Jia, Si-Yu Han, Lan-Zhe Guo, Zhi Zhou, Zhao-Long Li, Yu-Feng Li, and Zhi-Hua Zhou. A smooth transition between induction and deduction: Fast abductive learning based on probabilistic symbol perception. *arXiv preprint arXiv:2502.12919*, 2025.

[28] Vladimir Jojic, Stephen Gould, and Daphne Koller. Accelerated dual decomposition for MAP inference. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 503–510, 2010.

[29] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[30] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[32] Jaron Maene and Luc De Raedt. Soft-unification in deep probabilistic logic. *Advances in Neural Information Processing Systems*, 36:60804–60820, 2023.

[33] Jaron Maene, Vincent Derkinderen, and Luc De Raedt. On the hardness of probabilistic neurosymbolic learning. *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[34] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *advances in neural information processing systems*, 31, 2018.

[35] Giuseppe Marra, Michelangelo Diligenti, Francesco Giannini, Marco Gori, and Marco Maggini. Relational neural machines. *Proceedings of the European Conference on Machine Learning*, 2020.

[36] Giuseppe Marra and Ondřej Kuželka. Neural markov logic networks. In *Uncertainty in Artificial Intelligence*, pages 908–917. PMLR, 2021.

[37] Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. *33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

[38] Eleonora Misino, Giuseppe Marra, and Emanuele Sansone. Vael: Bridging variational autoencoders and probabilistic logic programming. *Advances in Neural Information Processing Systems*, 35:4667–4679, 2022.

[39] Mathias Niepert, Pasquale Minervini, and Luca Franceschi. Implicit MLE: backpropagating through discrete exponential family distributions. *NeurIPS*, 2021.

[40] Judea Pearl. Belief networks revisited. *Artificial intelligence in perspective*, pages 49–56, 1994.

[41] David Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial intelligence*, 64(1):81–129, 1993.

[42] Connor Pryor, Charles Dickens, Eriq Augustine, Alon Albalak, William Wang, and Lise Getoor. Neupsl: Neural probabilistic soft logic. *32nd International Joint Conference on Artificial*, 2023.

[43] Mukund Raghothaman, Xujie Si, Kihong Heo, and Mayur Naik. Difflog: Learning datalog programs by continuous optimization, 2019.

[44] Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. *Advances in neural information processing systems*, 30, 2017.

[45] Andrea Ruggieri, Francesco Stranieri, Fabio Stella, and Marco Scutari. Hard and soft EM in Bayesian network learning from incomplete data. *Algorithms*, 13(12):329, 2020.

[46] David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual decomposition for inference. 2011.

[47] Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezny, Steven Schockaert, and Ondrej Kuzelka. Lifted relational neural networks: Efficient learning of latent relational structures. *Journal of Artificial Intelligence Research*, 62:69–100, 2018.

[48] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[49] Simon Tavaré, David J Balding, Robert C Griffiths, and Peter Donnelly. Inferring coalescence times from DNA sequence data. *Genetics*, 145(2):505–518, 1997.

[50] Emile van Krieken, Pasquale Minervini, Edoardo Ponti, and Antonio Vergari. Neurosymbolic Diffusion Models. *arXiv preprint arXiv:2505.13138*, 2025.

[51] Emile van Krieken, Thiviyan Thanapalasingam, Jakub Tomczak, Frank Van Harmelen, and Annette Ten Teije. A-nesi: A scalable approximate method for probabilistic neurosymbolic inference. *Advances in Neural Information Processing Systems*, 36:24586–24609, 2023.

[52] Victor Verreet, Lennert De Smet, Luc De Raedt, and Emanuele Sansone. Explain, Agree, Learn: scaling learning for neural probabilistic logic. In *ECAI 2024: 27th European Conference on Artificial Intelligence*, pages 1349–1356, 2024.

[53] Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. *8th International Conference on Learning Representations*, 2020.

[54] Wenya Wang and Sinno Jialin Pan. Integrating deep learning with logic fusion for information extraction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9225–9232, 2020.

[55] Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. Nlprolog: Reasoning with weak unification for question answering in natural language. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[56] Halbert White. Maximum likelihood estimation of misspecified models. *Econometrica: Journal of the econometric society*, pages 1–25, 1982.

[57] Thomas Winters, Giuseppe Marra, Robin Manhaeve, and Luc De Raedt. Deepstochlog: Neural stochastic logic programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10090–10100, 2022.

[58] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International conference on machine learning*, pages 5502–5511. PMLR, 2018.

[59] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems*, 30, 2017.

[60] Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming. *IJCAI*, 2020.

A Worked example

We illustrate one iteration of EM-NeSy using the 1-digit MNIST-Add task [34]. Suppose the observed sum label is $y = 1$, and the neural component predicts the following prior distributions over the symbolic variables Z_1 and Z_2, representing the two summand digits:

z_1	0	1	2	3	4	5	6	7	8	9
$p_{\text{Ne}}(z_1 \mid x)$	0.91	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

z_2	0	1	2	3	4	5	6	7	8	9
$p_{\text{Ne}}(z_2 \mid x)$	0.025	0.4	0.025	0.025	0.025	0.025	0.025	0.4	0.025	0.025

Thus, the neural component assigns high probability to $z_1 = 0$, while predicting either $z_2 = 1$ or $z_2 = 7$ as likely values.

In the E-step, the symbolic component computes the posterior distribution

$$p_{\text{NeSy}}(z_1, z_2 \mid x, y) = \frac{p(z_1, z_2, y \mid x)}{p(y \mid x)}.$$

Since the symbolic constraint requires $z_1 + z_2 = y$, there are exactly two possible explanations for the observed sum $y = 1$: $(z_1, z_2) = (0, 1)$ and $(z_1, z_2) = (1, 0)$. Assuming that the neural predictions for the two digits are independent, we obtain

$$\begin{aligned} p(y = 1 \mid x) &= p_{\text{Ne}}(z_1 = 0 \mid x) p_{\text{Ne}}(z_2 = 1 \mid x) \\ &\quad + p_{\text{Ne}}(z_1 = 1 \mid x) p_{\text{Ne}}(z_2 = 0 \mid x) \\ &= 0.91 \cdot 0.4 + 0.01 \cdot 0.025 \\ &= 0.36425. \end{aligned}$$

The unnormalized probabilities of the two explanations are therefore

$$\begin{aligned} p(z_1 = 0, z_2 = 1, y = 1 \mid x) &= 0.91 \cdot 0.4 = 0.364, \\ p(z_1 = 1, z_2 = 0, y = 1 \mid x) &= 0.01 \cdot 0.025 = 0.00025. \end{aligned}$$

All other assignments have probability zero under the symbolic constraint. Normalizing these probabilities yields the posterior marginals:

z_1	0	1	2	3	4	5	6	7	8	9
$p_{\text{Sy}}(z_1 \mid x, y = 1)$	0.9993	0.0007	0	0	0	0	0	0	0	0

z_2	0	1	2	3	4	5	6	7	8	9
$p_{\text{Sy}}(z_2 \mid x, y = 1)$	0.0007	0.9993	0	0	0	0	0	0	0	0

Thus, the symbolic component strongly favors the explanation $(z_1, z_2) = (0, 1)$, effectively resolving the ambiguity in the neural prediction for z_2.

In the M-step, the neural component is optimized to align its predictions with the posterior computed by the symbolic component. This is achieved by minimizing the KL divergence between the posterior and the neural prior:

$$\mathcal{L}_{\text{NeSy}} = D_{\text{KL}} \left(p_{\text{NeSy}}(z \mid x, y) \parallel p_{\text{Ne}}(z \mid x) \right).$$

This encourages the neural component to assign higher probability to digit assignments that are both compatible with the observed label and supported by the symbolic constraints.

B Proof of theorem 3.1

Proof. Recall that, by the chain rule, we have

$$\nabla_{\boldsymbol{\theta}} \log f(\boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})}{f(\boldsymbol{\theta})} \tag{10}$$

for any differentiable function f. Hence, the claim follows from the following sequence of equalities:

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) &\overset{(4)}{=} -\nabla_{\boldsymbol{\theta}} \log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) \\
&\overset{(10)}{=} -\frac{\nabla_{\boldsymbol{\theta}} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})} \\
&\overset{(4)}{=} -\frac{\nabla_{\boldsymbol{\theta}} \sum_{\mathbf{z} \in R(\mathbf{Z})} p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) p_{\text{Sy}}(\mathbf{y}|\mathbf{z})}{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})} \\
&= -\frac{\sum_{\mathbf{z} \in R(\mathbf{Z})} p_{\text{Sy}}(\mathbf{y}|\mathbf{z}) \nabla_{\boldsymbol{\theta}} p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})} \\
&\overset{(10)}{=} -\frac{\sum_{\mathbf{z} \in R(\mathbf{Z})} p_{\text{Sy}}(\mathbf{y}|\mathbf{z}) p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})} \\
&\overset{(3)}{=} -\frac{\sum_{\mathbf{z} \in R(\mathbf{Z})} p(\mathbf{z}, \mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})} \\
&\overset{(5)}{=} -\frac{\sum_{\mathbf{z} \in R(\mathbf{Z})} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) p(\mathbf{z}|\mathbf{y}, \mathbf{x}, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})} \\
&= -\sum_{\mathbf{z} \in R(\mathbf{Z})} p(\mathbf{z}|\mathbf{y}, \mathbf{x}, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) \\
&\overset{(6)}{=} -\sum_{\mathbf{z} \in R(\mathbf{Z})} \ell(\mathbf{z}) \nabla_{\boldsymbol{\theta}} \log p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \theta) \\
&= \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{EM-NeSy}}(\theta).
\end{aligned}
$$

□

C Multi-M EM

Algorithm 1 One training step of Multi-M EM

Require: training sample $(\mathbf{x}, \mathbf{y})$, number of M steps m
Compute $p_{\text{Ne}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$
Compute $p_{\text{Sy}}(\mathbf{z}|\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})$
For $i = 1$ to m: Backpropagate w.r.t. $\mathcal{L}_{\text{EM-NeSy}}(\boldsymbol{\theta})$

D Generative EM-NeSy

Although EM-NeSy is developed and evaluated for predictive neuro-symbolic learning settings, where the neural component acts as a conditional prior for the symbolic component, the framework naturally extends to generative NeSy learning. Existing generative NeSy systems combine perception, symbolic reasoning, and generative components [38, 12, 15]. EM-NeSy extends naturally to such architectures by alternating between inference over latent symbolic variables and optimization of the neural components.

We illustrate this by describing how EM-NeSy can be instantiated for the CVAE-based architecture proposed in [13]. For a training pair (x, y) of a subsymbolic input x and symbolic label y, training of the NeSy-CVAE from [13] proceeds as follows:

1. A neural encoder predicts subsymbolic latent variables S_1 and S_2 from the input x;
2. A neural perception model predicts a distribution of a symbolic latent variable Z given S_2;
3. The symbolic component predicts the likelihood of the label y given the neurally predicted prior of Z;
4. A neural decoder generates a reconstruction g of the input conditioned on the latent variables S_1 and Z.

The loss function consists of two terms:

- L_1: the negative log-likelihood of the label y given the input x;
- L_2: a reconstruction loss comparing the input x to the generated reconstruction g.

EM-NeSy can be applied to this architecture in essentially the same way as in the purely predictive setting. The only necessary modification concerns the role of the symbolic component and the corresponding supervised loss term L_1. Rather than using the symbolic model to predict the likelihood of the label y from the latent symbolic variables Z, we use it to compute the posterior distribution over Z given the observed label y. Consequently, the negative log-likelihood term L_1 of the original objective is replaced by a new loss L_1': the KL-divergence between the neurally predicted prior distribution over Z and the posterior distribution over Z computed by the symbolic component, which is the complete-data log-likelihood of the neural prior.

theorem 3.1 shows that L_1 and L_1' induce identical gradients under exact inference, ensuring that the optimization behavior remains unchanged. Importantly, the reconstruction loss L_2 is completely unaffected by this modification, as it depends only on the neurally predicted variables S_1 and Z. It therefore continues to train the encoder-decoder pair exactly as in the original CVAE framework. In summary, extending EM-NeSy to a NeSy-CVAE does not require redesigning the generative architecture or its reconstruction objective. Only the symbolic supervision term is reformulated, while the remainder of the training procedure stays unchanged.

An optional additional modification of the NeSy-CVAE is to condition the decoder on S_1 and the *posterior* distribution over Z given the label y, rather than on the neurally predicted prior over Z. Intuitively, this may lead to faster convergence, since the decoder is provided with a more accurate estimate of the latent symbolic state that is directly informed by the observed label. However, this modification does change the reconstruction loss L_2. Because the posterior distribution over Z given y is computed by the symbolic component and treated as fixed, no gradients can flow through it. Nevertheless, gradients still flow through S_1, allowing the encoder and decoder to continue learning from the reconstruction objective.

E Experimental details

All experiments were performed using a single NVIDIA L40S GPU with 48 GB of memory on a machine equipped with a 128-thread CPU and 256 GB of RAM. For all experiments, we report the mean and standard deviation of the accuracies over random seeds $0, \ldots, 4$. We compute training times by

E.1 Benchmarks

MNISTAdd The MNIST dataset [18] contains $60{,}000$ training and $10{,}000$ test examples. We further split the original training set into a training and a validation split of sizes $50{,}000$ and $10{,}000$, respectively. From each of the resulting three MNIST splits and for a given number of digits n, we construct MNISTAdd datasets of size $\frac{50{,}000}{2n}$. Each MNIST digit is used in at most one example, where every example consists of $2n$ MNIST images representing two n-digit numbers, and the corresponding label is their sum.

Visual Sudoku The predefined Visual Sudoku datasets [2] consist of 500 Sudoku grids of size either 4×4 or 9×9, split into 100 training examples, 200 validation examples, and 200 test examples. A total of 11 splits are provided. We use the first one for all our experiments. Within each split, there is no overlap between the MNIST images used in the training, validation, and test sets. All training puzzles are valid Sudokus, whereas in the validation and test sets half of the puzzles are invalid. Invalid puzzles are generated by corrupting valid ones, either by replacing a correct digit with an incorrect one or by swapping two valid digits.

Warcraft path-planning The predefined Warcraft path-planning datasets [53] consist of grid-based maps of size 12×12 or 30×30. The maps are generated using a custom random sampling procedure based on 142 terrain tiles from the *Warcraft II* game environment. Each tile belongs to one of five terrain types, each associated with a fixed traversal cost ranging from 0.8 to 9.2. For each map, the

supervision is given as a 12×12 or 30×30 binary matrix indicating the shortest path from the top-left to the bottom-right corner. The training, validation, and test splits contain $10{,}000$, $1{,}000$, and $1{,}000$ examples, respectively.

E.2 Neural component

For MNIST-digit recognition, we use a LeNet-style convolutional neural network [31] consisting of two convolution–max-pooling blocks with 5×5 kernels and ReLU activations, followed by three fully connected layers. Given inputs of size 28×28, the network produces N-dimensional class probabilities for each digit. For inputs containing multiple digits, the same network is applied independently to each digit via weight sharing.

To extract the latent costs of the terrain tiles in the Warcraft path-finding task, we use the first five layers of ResNet18 [24] followed by a max-pooling operation. The resulting continuous value for each grid cell is then mapped to a probability distribution over the five terrain types by computing the L^2 distances to learnable class centers (initialized with $0.0, 1.0, 2.0, 3.0, 4.0$) and applying a softmax transformation with temperature 3.

E.3 Symbolic component

We use different variants of EM-NeSy depending on the task, as different inference methods are suited to different problem structures. We do not evaluate ABC-EM on MNISTAdd because BP-EM already performs fast exact inference on this task. Conversely, we do not evaluate BP-EM on Warcraft because we are not aware of an efficient factor-graph encoding of Dijkstra's shortest-path algorithm.

Belief propagation We use our own implementation of Pearl's loopy belief propagation algorithm [40]. It employs a synchronized message-passing scheme in which all messages are updated in parallel.

For MNISTAdd, the underlying factor graph contains $2n$ categorical variables for each input digit and n categorical variables for each sum digit, as well as $n+1$ binary carry variables. The resulting graph is a tree, so belief propagation yields exact marginals. We perform $2n$ iterations of belief propagation.

For Visual Sudoku, the factor graph contains n^2 categorical variables for ($n \in \{4, 9\}$) for every cell of the Sudoku grid and $m = 2 \cdot n \cdot \binom{n}{2} + \frac{n^2 \cdot (\sqrt{n}-1)}{2} \in \{56, 810\}$ binary variables for each of the m pairs of cells that share a row, column, or box. In addition, $m-1$ binary variables encode constraints that combine subsets of 2 to m of the m pair constraints. We perform m iterations of belief propagation, using a damping parameter of 10^{-2} for the message updates. To avoid numerical degeneracies in deterministic factors where message entries may collapse to zero we apply a clamping parameter of 10^{-5}.

Relaxed belief propagation for Visual Sudoku For Visual Sudoku, we additionally employ a relaxed variant of belief propagation in which only two message-passing iterations are performed, rather than propagating beliefs through the entire factor graph. Furthermore, incoming factor-to-variable messages are combined additively instead of multiplicatively, followed by normalization. Concretely, for a Sudoku cell Z we update the probability $p(Z = i)$ that Z contains digit i to

$$\frac{1}{C}(p(Z = i) = \sum_{Z' \in M(Z)} (1 - p(Z' = i))),$$

where C is a normalization constant and $M(Z)$ denotes the set of all Sudoku cells that share a row, column or box with Z. This probability update coincides with the sampling objective we use in ABC rejection sampling.

ABC rejection sampling Both in the Visual Sudoku and Warcraft path-planning task, we follow the approach described in section 4.2. We sample uniformly from the neurally predicted distribution $p_{\text{Ne}}(\cdot|x, \theta)$ for the Sudoku digits or Warcraft grid cells, which factorizes into the distributions for the single digits/grid cells Z_i for $i = 1, \ldots, n$.

For the Visual Sudoku experiment, let m_{ij} denote the constraint $Z_i \neq Z_j$. As described in theorem 4.1, we define

$$M := \{m_{ij} : Z_i, Z_j \text{ share a row column or box in the Sudoku}\}$$

to be the set of Sudoku constraints. For every fixed i let

$$M_i := \{m_{ij} \in M : j \in \{1, \dots, n^2\}\}$$

be the set of all constraints that involve Z_i. We further choose the distance measure $\rho = L^1$ and the summary statistics $s_{\mathbf{Y}}(\mathbf{y}) = 0$ and $s_{Z_i}(z_i) = \frac{k}{|M_i|}$ where C is a normalization constant and k denotes the number of constraints from M_i that are satisfied by the sample z_i. Then we assign the weight

$$\frac{1}{C}\rho\big(s_{Z_i}(z_i), s_{\mathbf{Y}}(\mathbf{y})\big) = \frac{1}{C}\left|\frac{k}{|M_i|} - 0\right| = \frac{k}{C|M_i|},$$

to each sample z_i where C denotes a normalization constant.

In the Warcraft experiment, we employ a Dijkstra solver that predicts the shortest path for each sample. We let $s_{Z_i}(z_i) = 1$ if the predicted shortest path for the sample $\mathbf{z}$ matches the true shortest path on the grid cell Z_i (i.e. both the predicted and true shortest path contain Z_i or they both don't). We let $s_{\mathbf{Y}(\mathbf{y})} = 0$ and assign the weight $\frac{1}{C}\rho\big(s_{Z_i}(z_i), s_{\mathbf{Y}}(\mathbf{y})\big)$ to the sample z_i.

E.4 Hyperparameters

We used the validation splits for hyperparameter tuning. At test time, we report results using the checkpoint that achieves the highest validation accuracy. We employ the Adam optimizer for all experiments.

For MNISTAdd, we use a cosine-annealing learning-rate schedule updated after each epoch, beginning at 10^{-3} and decaying to 10^{-4}. We train for 30 epochs with batch sizes of 50, 10 and 2 for 4, 15 and 100 digits, respectively.

For Visual Sudoku, we use a cosine-annealing learning-rate schedule updated after each epoch, beginning at 10^{-3} and decaying to 10^{-6}. We train with a batchsize of 5 for 500 and 200 epochs for 4×4 and 9×9 Sudokus, respectively. For ABC sampling, we use 1000 samples on 4×4 Sudokus and 10,000 for 9×9 Sudokus.

For the Warcraft path-planning experiment, we use a constant learning-rate of $3 \cdot 10^{-4}$ on the 12×12 grid and 10^{-4} on the 30×30 grid. We train with a batchsize of 50 for 5 and 10 epochs, respectively. We use an entropy regularization parameter of 0.03 for training on the 12×12 grid. For ABC sampling, we used 50 and 20 samples on 12×12 and 30×30 grids, respectively.

E.5 Baselines

We ran the experiments for A-NeSI [51], I-MLE [39], EXAL [52] and Scallop [26] with the hyperparameters specified in the original papers. For A-NeSI we use the symbolic prediction variant. For A-NeSI on the Warcraft path-planning benchmark, we were not able to reproduce the results due to high variance and training times. Therefore we cite the accuracies from the original paper and report training time on our machines for a single run. For EXAL on the Warcraft path-planning benchmark, we use the accuracies reported in the original paper. Training times are obtained from runs on our own hardware using synthetic data, which has a negligible impact on training duration. We adopted this setup because the EXAL implementation depends on a text-based conversion of the original Warcraft dataset that was unavailable to us.

Note that EXAL relies on iterative Gibbs sampling with a burn-in period of 100 resampling steps followed by 300 sampling steps. This may explain why for the 12×12 Warcraft maps, the runtime of EXAL is comparable to ABC-EM, whereas on the 30×30 dataset, ABC-EM which uses only 20 samples on the 30×30 maps is faster.

E.6 Timeouts

The reported timeouts reflect the inherent computational cost of exact inference, whose search spaces grow exponentially with the problem size. The exact-inference results for DeepStochLog, Semantic Loss, and Semantic Probabilistic Layers are taken from [51]. Note that DeepStochLog considers all possible explanations of a label. For n-digit MNIST addition, this corresponds to 10^n explanations, making timeouts for $n = 15$ and $n = 100$ digits unsurprising. Similarly, Semantic Probabilistic Layers on Warcraft must compile the set of valid paths on a 3030 grid into an SDD. This set is already

lower-bounded by the number of monotone corner-to-corner paths, which is greater than $3 \cdot 10^{16}$. The Scallop baseline was implemented using the Python package scallopy. For the 9×9 Visual Sudoku task, the system attempted to compile the Sudoku constraints but failed after approximately two hours, terminating with an error before compilation could be completed.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We indeed introduce the EM-NeSy framework, show the properties that we claim and provide empirical results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a paragraph about limitations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The only theoretical result of the paper is 3.1 and we provide the proof of it in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all the experimental details in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The benchmarks we use are open access and we provide our code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all the experimental details in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide the standard deviation of all results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information on the types of CPUs and GPUs used, their available memory, and performance timings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [N/A]

   Justification: The paper is about foundational research about neurosymbolic learning and uses no human participants.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper provides foundational research about neurosymbolic AI. So, we do not see direct positive or negative societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We only use models for synthetic neurosymbolic tasks and do not see any risks for their misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all original papers that produce the datasets or code that we use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.