Ambiguity-Aware Abductive Learning

Hao-Yuan He [1 2] **Hui Sun** [1 2] **Zheng Xie** [1] **Ming Li** [1 2]

Abstract

Abductive Learning (ABL) is a promising framework for integrating sub-symbolic perception and logical reasoning through abduction. In this case, the abduction process provides supervision for the perception model from the background knowledge. Nevertheless, this process naturally contains uncertainty, since the knowledge base may be satisfied by numerous potential candidates. This implies that the result of the abduction process, i.e., a set of candidates, is ambiguous; both correct and incorrect candidates are mixed in this set. The prior art of Abductive Learning selects the candidate that has the minimal inconsistency of the knowledge base. However, this method overlooks the ambiguity in the abduction process and is prone to error when it fails to identify the correct candidates. To address this, we propose Ambiguity-Aware Abductive Learning (A^3BL), which evaluates all potential candidates and their probabilities, thus preventing the model from falling into sub-optimal solutions. Both experimental results and theoretical analyses prove that A^3BL markedly enhances ABL by efficiently exploiting the ambiguous abduced supervision.

1. Introduction

Currently, machine learning methods are achieving significant success in perception (Krizhevsky et al., 2012; Vaswani et al., 2017). However, real-world learning tasks usually require not only the perception ability but also the logical reasoning ability (Kahneman, 2011). To address the limitations of current machine learning methods, the next generation of Artificial Intelligence calls for the integration of data-driven machine learning and knowledge-driven symbolic reasoning (Zhou, 2019).

[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China [2]School of Artificial Intelligence, Nanjing University, Nanjing, China. Correspondence to: Ming Li <lim@lamda.nju.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Abductive Learning (ABL) (Zhou, 2019; Dai et al., 2019) represents a novel framework seamlessly integrating machine learning systems with logical reasoning systems. Within this framework, the machine learning model is trained to transform raw input data, e.g., images and text, into sub-symbolic concepts; concurrently, the logical model is designed to conduct reasoning based on these concepts. Logical reasoning, facilitated through abductive reasoning (Magnani, 2009), is employed to identify the accurate concepts of unlabeled instances; these identified concepts are then utilized to update the machine learning model through supervised learning.

Consider the following illustrative example: the digit equation $\texttt{SUM}(\boxed{1}, \boxed{2}) = 3$ is presented, accompanied by the background knowledge that it represents a digit addition task. Initially, the machine learning model identifies $\boxed{1} = 1$ and $\boxed{2} = 1$. However, upon logical inference, it is deduced that $1 + 1 \neq 3$. Nonetheless, a plausible explanation is proposed: $\boxed{1} = 1, \boxed{2} = 2$, aligning with the background knowledge and thereby validating the equation.

Although the above procedure seems feasible, *ambiguity* still persists. The term *ambiguity* in this context means that the abduction process yields not only the correct result but also other validate candidates (Magnani, 2009), such as $[\boxed{1} = 0, \boxed{2} = 3], \cdots, [\boxed{1} = 3, \boxed{2} = 0]$, which are reasonable hypotheses given the existing background knowledge. The use of such ambiguous supervision derived from abductive reasoning brings significant challenges to model training processes. To address this issue, prior studies (Dai et al., 2019) have selected the nearest candidate (e.g., in terms of Hamming distance) to the model's prediction as pseudo-labels for learning. In this case, the accurate identification of candidates depends heavily on the performance of the perception model. When training the perception model from scratch, the initial recognitions are often unreliable, which means that the closest candidate of the abduction results may not be the correct one. Further training on these selected candidates as supervision data may lead to the perception model becoming entrapped in a sub-optimal state. Therefore, it still remains a challenge to properly utilize the ambiguity of abduction candidates in the ABL framework.

To address the ambiguity in the abduction results, this paper introduces the concept of Ambiguity-Aware Abductive

Learning, abbreviated as A^3BL. Rather than considering a single candidate at once, this work focuses on assessing all potential abduction candidates. Specifically, A^3BL employs an Expectation-Maximization (EM) algorithm for optimization purposes. Initially, A^3BL assigns a weight to each candidate, derived from the machine learning model. Utilizing these weights, A^3BL converts the ambiguous outcomes of abductive reasoning into instance-level class probability distributions. Then A^3BL utilizes a uniquely formulated ambiguity-aware abductive loss function. Following the optimization of this loss function, the machine learning model updates the weight of each candidate, thereby revising the instance-level probability distribution for subsequent optimization steps. Through the iterative optimization of the EM algorithm, which guarantees convergence, A^3BL effectively maximize the consistency between the perception model and the knowledge base. Both experimental outcomes and theoretical analyses substantiate that our approach significantly enhances ABL by efficiently leveraging ambiguous abduced supervision.

The contributions of this study are outlined as follows:

- Identification and articulation of the ambiguity issue within the ABL framework, a challenge that stems from the inherent ambiguity in the abduction process.
- Proposal of A^3BL as a novel solution to mitigate this issue, accompanied by a comprehensive theoretical analysis with the establishment of an error bound to substantiate the effectiveness of A^3BL.
- Empirical studies present the superior performance of the proposed method A^3BL.

The remainder of this paper is organized as follows: Section 2 introduces the majority of the related works pertinent to this paper. Then, Section 3 presents the preliminaries and the proposed method A^3BL, including the theoretical analysis. After that, Section 4 details empirical studies conducted to support our claims and verify the performance of A^3BL. Finally, Section 5 provides the conclusion and additional discussion of this paper.

2. Related Works

Neuro-Symbolic Learning Researchers have made attempts to combine neural networks and symbolic reasoning in order to achieve a more comprehensive form of Artificial Intelligence several decades ago (Towell & Shavlik, 1994; Sun, 1994; Garcez et al., 2002). Pioneering efforts have been made to convert logic rules into loss functions, such as the development of Semantic Loss (Xu et al., 2018). This approach utilizes probabilistic logic to transform logic rules into loss functions. Another notable method is Semantic-Based Regularization (Roychowdhury et al., 2021), which employs fuzzy logic to achieve the transformation of logic rules into loss functions. However, approximating logical reasoning cannot replace a true logical engine, and problems may arise when attempting to approximate discrete logical calculations (van Krieken et al., 2022; He et al., 2024). Recently, there have been advancements in this area by employing hybrid-system, including DeepProbLog (Manhaeve et al., 2018; 2021), which integrates deep neural networks with problog (De Raedt et al., 2007) using a probabilistic logic approach. Another approach is NeurASP (Yang et al., 2020), which is similar to DeepProbLog but employs Answer Set Programming (ASP) (Dimopoulos et al., 1997) instead of problog. Additionally, DeepStochLog (Winters et al., 2022) is a related approach to DeepProbLog, but it enhances computation speed by utilizing a stochastic logic approach.

Abductive Learning Abduction (Magnani, 2009) refers to the process of selectively inferring certain facts and hypotheses that explain phenomena and observations based on background knowledge. It has been a recurring topic of interest in the field of AI, as attempts have been made to integrate it with symbolic induction (Muggleton & Bryant, 2000; Mooney, 2000). Abductive Learning (ABL) (Dai et al., 2019; Zhou, 2019) aims to leverage learning and abduction in a mutually beneficial loop, presenting a novel paradigm for integrating machine learning and logical reasoning within a unified framework. The ABL framework is renowned for its expressive and flexible nature, as it can be applied to both labeled and unlabeled data with an appropriate knowledge base. Dai & Muggleton (2021) enhance the ABL framework's ability to induce knowledge from the raw data, the optimization builds upon the EM algorithm. Furthermore, ABL has been applied in various practical tasks, including theft judicial sentencing (Huang et al., 2020), stroke evaluation (Wang et al., 2021), optical character recognition (Cai et al., 2021), and historical document segmentation and recognition (Gao et al., 2024). Recently, Huang et al. released an open library for ABL, which is easy to use. Yang et al. (2024) worked on scenarios where the given knowledge base does not fit well with the ground-truth background knowledge. Despite its achievements in multiple applications, ABL still faces the challenge of the cold-start problem (Tao et al., 2024). This problem arises when it becomes difficult to abduce the correct candidate, particularly when the machine learning model is trained from scratch. This ambiguity stems from the the abduction process (Magnani, 2009), especially during the initial learning phase of the learning model.

Weakly Supervised Learning Our method draws inspiration from commonly employed weighting techniques in the field of weakly supervised learning (WSL) (Zhou, 2017). WSL aims to learn from imperfect supervision, which en-

compasses various approaches such as learning from noisy labels (Natarajan et al., 2013), multi-instance learning (Dietterich et al., 1997), multi-label learning (Zhang & Zhou, 2014), and partial label learning (Cour et al., 2011). WSL has also achieved considerable success in other applications, such as natural language process (Artzi & Zettlemoyer, 2013); object detection (Zhang et al., 2022); AUC optimization (Xie et al., 2024); and so on. The ABL framework can be viewed as expanding the domain of WSL (Zhou & Huang, 2022), where the supervision information can come from knowledge reasoning. Within the ABL framework, it is possible to get an effective model even when there is an insufficient amount of labeled or unlabeled data, as long as high-quality knowledge is available. Recently, Wang et al. (2023) investigated a typical case of neuro-symbolic system, from a multi-instance weak supervision perspective. Tao et al. (2024) analyzed the cold-start problem of ABL by adopting a perspective rooted in noisy-label learning. Both of these works focused on the theoretical aspects of the field without proposing any specific algorithms. Distinguishing itself from previous studies, our work specifically focuses on the ambiguity of abduction within the ABL framework. A^{3}BL extend the ABL by enhancing stable training and promoting fast convergence, while also providing a promising theoretical analysis.

3. Ambiguity-Aware Abductive Learning

In this section, we begin by introducing the preliminaries of the problem setting. Subsequently, we discuss the optimization objectives of the previous ABL methods and examine their limitations. Following this, we presented the primary contributions of this paper, namely the Ambiguity-Aware Abductive Learning method, short for A^{3}BL. Finally, we provide a theoretical analysis of A^{3}BL, establishing an error bound and drawing connections between our optimization process and the Expectation-Maximization (EM) algorithm. To the best of our knowledge, this is the first attempt to provide an error bound analysis within the ABL framework.

3.1. Problem Setting

This study follows the common setting of the ABL. The ABL framework consists of a perception model and a reasoning model, e.g., a knowledge base. The perception model, denoted as $f : \mathcal{X} \mapsto \mathcal{Z}$, maps an instance x from the input space $\mathcal{X}$ to a label z in the symbol space $\mathcal{Z} = \{1, \cdots, L\}$. The knowledge base, denoted as KB, is comprised of rules defined over a sequence of instances $\boldsymbol{x} = (x_1, \cdots, x_m) \in \mathcal{X}^m$, where m denotes the sequence length. The corresponding labels of the instance sequence are denoted by $\boldsymbol{z} = (z_1, \cdots, z_m) \in \mathcal{Z}^m$. To clarify, in some cases, f may represent a mapping from $\mathcal{X}$ to a distribution over $\mathcal{Z}$. Additionally, when f processes a sequence input $\boldsymbol{x}$, it correspondingly outputs a sequence. Though not knowing the sequence of labels $\boldsymbol{z}$ of $\boldsymbol{x}$, we have some indirect information $y \in \mathcal{Y}$, the target label, such that $\boldsymbol{z} \wedge \text{KB} \models y$. If the target label is determined by $\boldsymbol{z}$ given the knowledge base KB, this process can be referred as a logical forward function $\sigma(\cdot)$, indicating that $\sigma(\boldsymbol{z}) = y$. The overall training set with N sequences can be denoted as $(\hat{\mathcal{X}}^m, \hat{\mathcal{Y}})$, which is drawn from the distribution $(\mathcal{X}^m, \mathcal{Y})$. Here, $\hat{\mathcal{X}}^m = \{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(N)}\}$, and $\hat{\mathcal{Y}} = \{y^{(1)}, \cdots, y^{(N)}\}$.

In a typical Abductive Learning process, the machine learning model can not access the sequence labels $\boldsymbol{z}$ during training, but it can access the knowledge base for abductive reasoning. When provided with the input $\boldsymbol{x}$, the model outputs $\tilde{\boldsymbol{z}} = (\tilde{z}_1, \cdots, \tilde{z}_m)$. If the prediction $\tilde{\boldsymbol{z}}$ with the knowledge base KB entails a wrong target label $\tilde{y} \neq y$, the resaoner immediately knows that $\tilde{\boldsymbol{z}}$ is incorrect. Following the abduction process, the resaoner could restrict the possible concept sequence $\boldsymbol{z}$ within a candidate set $\boldsymbol{s} \subseteq \mathcal{Z}^m$. We provide an example below to facilitate a better understanding of the notations mentioned above.

Example 3.1. *The input data $\boldsymbol{x} = (x_1, x_2) \in \mathcal{X}^2$, where $\mathcal{X}$ is a digit image space (e.g., MNIST). Symbol space $\mathcal{Z} = \{0, 1, \cdots, 9\}$, target label space $\mathcal{Y} = \{0, 1, \cdots, 18\}$. The logical forward function $\sigma(\cdot, \cdot) : \mathcal{X}^2 \mapsto \mathcal{Y} = \textit{Sum}(\cdot, \cdot)$. If target label $y = 2$ and the logical forward result $\sigma(\tilde{\boldsymbol{z}}) \neq 2$, then the knowledge base will abduce candidate set $\boldsymbol{s} = \{(0, 2), (1, 1), (2, 0)\}$.*

The learning system of ABL is designed for the perception tasks. In this study, we specifically focus on the classification task, which involves minimizing the risk associated with classification errors. For convenience, we can equivalently rewrite the risk at the sequence level as follows:

Definition 3.1 (Classification risk)**.** The objective of multi-class classification is to train a multi-class classifier that minimizes the classification risk defined as follows:

$$R(f; \mathcal{L}) = \mathbb{E}_{p(\boldsymbol{x}, \boldsymbol{z})}[\mathcal{L}(f(\boldsymbol{x}), \boldsymbol{z})]. \tag{1}$$

Here, the loss function $\mathcal{L}(\cdot, \cdot)$ represents the aggregation of classification errors for each instance in the sequence $\boldsymbol{x}$. Mathematically, it can be expressed as:

$$\mathbb{E}_{p(\boldsymbol{x}, \boldsymbol{z})} \left[\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_{\text{cls}}(f(x_i), z_i) \right], \tag{2}$$

where $\mathcal{L}_{\text{cls}}$ typically denotes a classification loss function.

3.2. Previous Abductive Learning

To minimize the aforementioned classification risk, prior Abductive Learning methods basically solve an empirical

risk minimization problem which can be formalized as:

$$\begin{aligned} &\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\boldsymbol{x}^{(i)}), \boldsymbol{z}^{(i)}) \\ \text{s.t.} \quad &\boldsymbol{z}^{(i)} = \arg\min_{\boldsymbol{c} \in \boldsymbol{s}^{(i)}} \text{Score}(\boldsymbol{c}, f(\boldsymbol{x}^{(i)})), i \in [N], \end{aligned} \tag{3}$$

where $\boldsymbol{s}^{(i)}$ is the abduced candidate set of $f(\boldsymbol{x}^{(i)})$, that is to say $\boldsymbol{s}^{(i)} = \{\boldsymbol{c} | \boldsymbol{c} \wedge \text{KB} \models y^{(i)}\}$. The term $\text{Score}(\boldsymbol{c}, f(\boldsymbol{x}^{(i)}))$ is used to quantify how likely the candidate $\boldsymbol{c}$ is incorrect based on the model's prediction $f(\boldsymbol{x}^{(i)})$.

To implement this function, different measures can be used. For instance, Dai et al. (2019) use the Hamming distance (Hamming, 1950) as the score function:

$$\text{Score}(\boldsymbol{c}, f(\boldsymbol{x}^{(i)})) = \text{Hamming}(\boldsymbol{c}, f(\boldsymbol{x}^{(i)})). \tag{4}$$

This scoring function tries to use the candidate $\boldsymbol{c}^*$ that has most of the same labels as the prediction $f(\boldsymbol{x}^{(i)})$. Additionally, it is feasible to extend this approach to include the confidence provided by the model's prediction. This extension is implemented in the official package[1] as follows:

$$\text{Score}(\boldsymbol{c}, f(\boldsymbol{x}^{(i)})) = 1 - \prod_{j=1}^{m} f(\boldsymbol{x}^{(i)})_{c_j}. \tag{5}$$

In this context, $f(\cdot)$ represents the model's predicted distribution over $\mathcal{Z}$. Additionally, $f(\boldsymbol{x}^{(i)})_{c_j}$ is the model's estimated probability of the class c_j given the input $x_j^{(i)}$.

3.3. Ambiguity-Aware Abductive Loss

As discussed above, the abductive reasoning process cannot eliminate all incorrect predictions, hence the candidate set cannot be used as an accurate supervision. Previous research attempts to select one assignment of the sequence labels with minimal difference to the predictions (Equation 4) or the maximal probability according to the model's confidence (Equation 5). However, the selected assignment is prone to error and could lead the model astray. To prevent this from happening, such methods have to require a number of instance-level supervision, i.e., Semi-Supervised Abductive Learning (Huang et al., 2020) or assume an initial model performance, i.e., pretrain the model (Dai et al., 2019). Such requirements further limit the application of the methods. The root cause of the learner being trapped in an incorrect situation stems from the ambiguity of the abduced result: once the model selects an incorrect label to learn, its incorrect perception can be strengthened in the subsequent training process, leading to a vicious cycle. Thus, it is critical to be aware that any possible candidate could be the correct one.

[1] https://github.com/AbductiveLearning/ABLKit

To achieve this, we propose Ambiguity-Aware Abductive Learning, or A^{3}BL for short. A^{3}BL initially transforms the ambiguous abduction results into instance-level class probability distributions. Subsequently, it utilizes a novel ambiguity-aware abductive loss to enable the model to learn classification from these class probability distributions. By doing this, A^{3}BL can fully leverage the ambiguous abductive results to facilitate learning.

Suppose the model predicts the labels of an instance sequence $\boldsymbol{x} = (x_1, \cdots, x_m)$ to be $\tilde{\boldsymbol{z}} = (\tilde{z}_1, \cdots, \tilde{z}_m)$. Through abductive reasoning, it is realized that the candidate set of potentially correct label sequences should be $\boldsymbol{s} = \{\boldsymbol{c}_i | \boldsymbol{c}_i \wedge \text{KB} \models y\}$. The probability of a candidate label sequence to be true can be formalized as:

$$p(\boldsymbol{z} = \boldsymbol{c}_i | \boldsymbol{x}) = \prod_{1 \le j \le m} p(z_j = c_{ij} | x_j)\,. \tag{6}$$

Due to the fact that all impossible label sequences are ruled out by the abduction, the posterior probability of a candidate label sequence $\boldsymbol{c}_i$ to be all correct, given the candidate set $\boldsymbol{s}$, can be obtained by:

$$p(\boldsymbol{z} = \boldsymbol{c}_i | \boldsymbol{x}, \boldsymbol{s}) = \frac{p(\boldsymbol{z} = \boldsymbol{c}_i | \boldsymbol{x})}{\sum_{\boldsymbol{c} \in \boldsymbol{s}} p(\boldsymbol{z} = \boldsymbol{c} | \boldsymbol{x})}\,. \tag{7}$$

Then, for an instance x_j appeared in the sequence $\boldsymbol{x}$, its probability of being k-th class should be:

$$p(z = k | x_j, \boldsymbol{s}) = \sum_{\boldsymbol{c}_i \in \boldsymbol{s}} \mathbb{I}[k = c_{ij}] p(\boldsymbol{z} = \boldsymbol{c}_i | \boldsymbol{x}, \boldsymbol{s})\,. \tag{8}$$

By this equation, we obtain the class probability distribution of any instance x_j that occurred in the sequence $\boldsymbol{x}$, which can be used as a supervision for the model training.

The next key step is to train a classifier based on the class distribution $p(z|x, \boldsymbol{s}) = [p(z = 1|x, \boldsymbol{s}), \cdots, p(z = L|x, \boldsymbol{s})]^\top$. To make the model learn from this ambiguous supervision, we minimize the difference between the model's output and the probability distribution:

$$\frac{1}{m \cdot N} \sum_{i=1}^{N} \sum_{j=1}^{m} \mathcal{L}_{\text{cls}} \left(f(x_j^{(i)}), p(z | x_j^{(i)}, \boldsymbol{s}^{(i)}) \right), \tag{9}$$

where the $\mathcal{L}_{\text{cls}}$ is the cross entropy loss. Here we denote $\boldsymbol{s}^{(i)}$ as the candidate set abduced from $\boldsymbol{x}^{(i)}$. In fact, the above optimization object is equivalent to the empirical risk below:

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{\boldsymbol{c} \in \boldsymbol{s}^{(i)}} p(\boldsymbol{c} | \boldsymbol{x}^{(i)}, \boldsymbol{s}^{(i)}) \cdot \mathcal{L}(f(\boldsymbol{x}^{(i)}), \boldsymbol{c}). \tag{10}$$

The term $p(\boldsymbol{c} | \boldsymbol{x}^{(i)}, \boldsymbol{s}^{(i)})$ is provided by the perception model f. This term represents the weight assigned to candidate

$\boldsymbol{c}$, considering the candidate set $\boldsymbol{s}^{(i)}$ and the sequence $\boldsymbol{x}^{(i)}$. The equivalence of the above two equations can be easily validated by expanding both of them, and the proof can be seen in Appendix A. We name Equation (10) as $\hat{R}_{\mathrm{A}^3}$, representing empirical ambiguity-aware abductive risk. By optimizing this risk, each potential candidate is evaluated with distinct weights, enabling A^{3}BL to more effectively utilize ambiguous abduction results.

3.4. Theoretical Analysis

In this section, we provide a theoretical analysis of A^{3}BL. Initially, we demonstrate that optimizing Equation (10) can be interpreted as a process of maximizing the log-likelihood $\sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}^{(i)}, y^{(i)})$, via the Expectation-Maximization (EM) algorithm (Dempster et al., 1977), where θ denotes the parameters to be optimized. Additionally, we present an estimation error bound for Equation (10). Detailed proofs are available in Appendix A.

Theorem 3.2. *The optimization of Equation (10) is actually optimizing the log-likelihood $\sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}^{(i)}, y^{(i)})$, via the EM algorithm.*

Proof Sketch. For simplicity, consider a single sample $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$. By Jensen's inequality, the log-likelihood $\log p(\boldsymbol{x}^{(i)}, y^{(i)})$ is lower bounded by:

$$\log p_\theta(\boldsymbol{x}^{(i)}, y^{(i)}) \geq \sum_{\boldsymbol{c} \in \boldsymbol{s}^{(i)}} w_{\boldsymbol{c}} \log\left(\frac{p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c})}{w_{\boldsymbol{c}}}\right), \tag{11}$$

where the $w_{\boldsymbol{c}} \in [0, 1]$ and $\sum_{\boldsymbol{c} \in \boldsymbol{s}^{(i)}} w_{\boldsymbol{c}} = 1$ is a coefficient. The equality holds when

$$\forall \boldsymbol{c}_k, \boldsymbol{c}_j \in \boldsymbol{s}^{(i)}, \quad \frac{p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c}_k)}{w_{\boldsymbol{c}_k}} = \frac{p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c}_j)}{w_{\boldsymbol{c}_j}}. \tag{12}$$

It turns out that this equality holds when $w_{\boldsymbol{c}} = p(\boldsymbol{z} = \boldsymbol{c} | \boldsymbol{x}^{(i)}, \boldsymbol{s}^{(i)})$. Thus, the E-step corresponds to setting $w_{\boldsymbol{c}} = p(\boldsymbol{z} = \boldsymbol{c} | \boldsymbol{x}^{(i)}, \boldsymbol{s}^{(i)})$, and then the M-step involves optimizing the ambiguity-aware abductive risk. □

The log-likelihood $\sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}^{(i)}, y^{(i)})$ quantifies the consistency between the model's predictions and the underlying knowledge base. In this context, optimizing Equation (10) is to *maximize the consistency* between the machine learning model and the knowledge base. Next, we will establish an estimation error bound to analyze the difference between the estimation error of $\hat{f}_{\mathrm{A}^3}$ and the optimal classifier $f^\star$. Here, $f^\star$ is defined as the classifier that minimizes the risk $R(f)$ over the function space $\mathcal{F}$, while $\hat{f}_{\mathrm{A}^3}$ is the empirical risk minimizer of $\hat{R}_{\mathrm{A}^3}(f)$ over the same function space $\mathcal{F}$. To accomplish this, we need to define a class of real functions $\mathcal{F}_i$ (Maurer, 2016), and then $\mathcal{F} = \oplus_{i \in [K]} \mathcal{F}_i$ represents the K-valued function space, where $K = L^m$. Thus, it can be observed that as m increases, the complexity of the learning task increases too.

Theorem 3.3 (Error bound)**.** *Suppose that $\mathcal{L}(f(\boldsymbol{x}), \boldsymbol{z})$ is ρ-Lipschitz with respect to $f(\boldsymbol{x})$ for all $\boldsymbol{z} \in \mathcal{Z}^m$ and upper-bounded by $M = \sup_{f \in \mathcal{F}, \boldsymbol{x} \in \mathcal{X}^m, \boldsymbol{z} \in \mathcal{Z}^m} \mathcal{L}(f(\boldsymbol{x}), \boldsymbol{z})$. Let $\mathfrak{R}_n(\mathcal{F}_i)$ be the Rademacher complexity of $\mathcal{F}_i$ with sample size n. Then for any $\delta > 0$, with probability at least $1 - \delta$,*

$$R(\hat{f}_{A^3}) - R(f^\star) \leq 4\sqrt{2}\rho \sum_{i=1}^{K} \mathfrak{R}_n(\mathcal{F}_i) + 2M\sqrt{\frac{\log(2/\delta)}{2n}}. \tag{13}$$

As $n \to \infty$, it follows that $\mathfrak{R}_n(\mathcal{F}_i) \to 0$ for all parametric models with a bounded norm, such as deep networks trained with weight decay (Lu et al., 2019). Therefore, Theorem 3.3 demonstrates that $\hat{f}_{\mathrm{A}^3}$ converges to $f^\star$ as the number of training data tends to infinity, indicating that the learning process is consistent in an asymptotic sense.

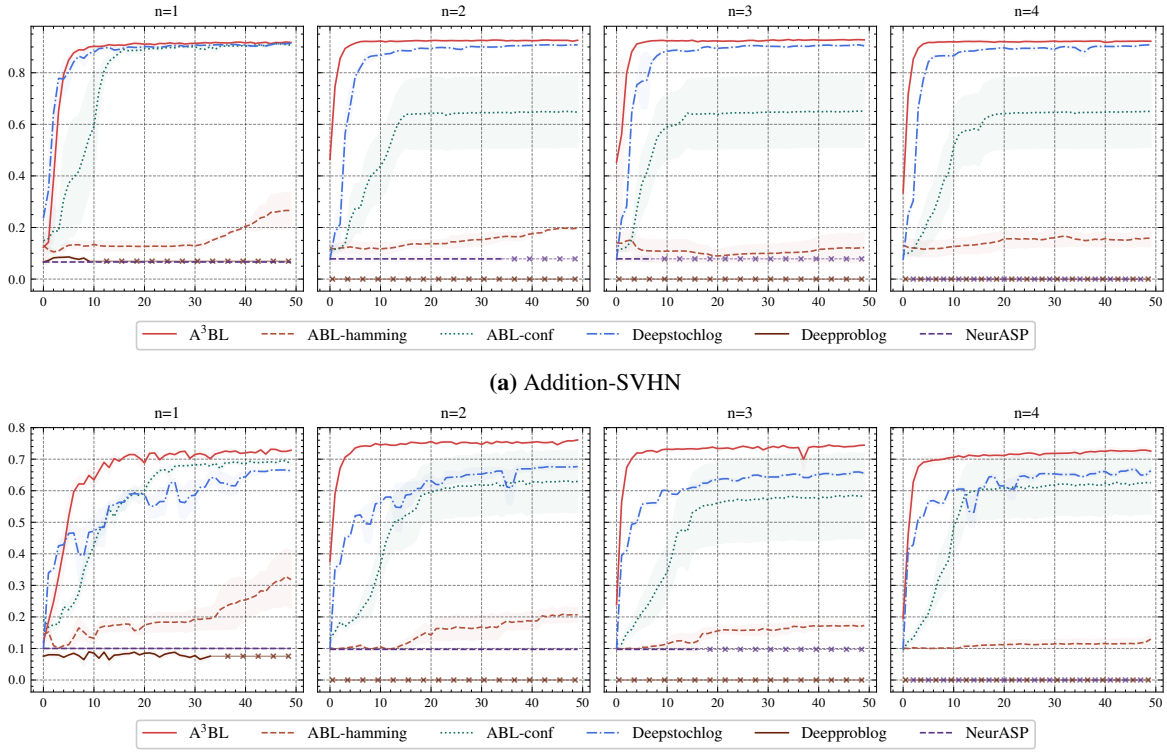
4. Empirical Study

In this section, we conduct experiments to verify our claims and validate the superior performance of A^{3}BL. Specifically, the experiments focus on two challenging tasks in the neuro-symbolic field: *Digit Addition* and *Handwritten Formula Recognition*. To ensure reproducibility, all experiments are repeated five times, each with a different random seed. The same backbone is used for all baselines; for further details, please refer to the appendix.

4.1. Settings of Digit Addition

Manhaeve et al. (2018) proposed the *Digit Addition* task, which is based on standard addition rules. The training data for this task is presented in the form SUM(,) = 3. Building upon this concept, we have expanded the task to incorporate four distinct datasets: MNIST (Deng, 2012), KMNIST (Clanuwat et al., 2018), CIFAR10 (Krizhevsky, 2009), and SVHN (Netzer et al., 2011). Each dataset consists of 10 classes, with the class indices representing digits from 0 to 9. In line with the work of Winters et al. (2022), to increase the task's complexity, we have extended the range of digit-size n from 1 to 4, e.g., SUM( ) = 33. When the size of the digits increases, the size of the abduced candidates set correspondingly expands, and so does the complexity of the background knowledge.

Compared Methods We conducted a comparative analysis of our method by contrasting it with several prominent hybrid system approaches. These methods include DeepProbLog (Manhaeve et al., 2021), which integrates deep neural networks with ProbLog (De Raedt et al., 2007) through a probabilistic logic framework. Another noteworthy method is NeurASP (Yang et al., 2020), akin



(a) Addition-SVHN

(b) Addition-CIFAR10

Figure 1. Performance curves (accuracy vs. epochs) of compared methods on digit addition tasks. The shaded area represents the standard error of the methods over 5 repetitions. Cross marks denote experiments that were not finished within 48 hours.

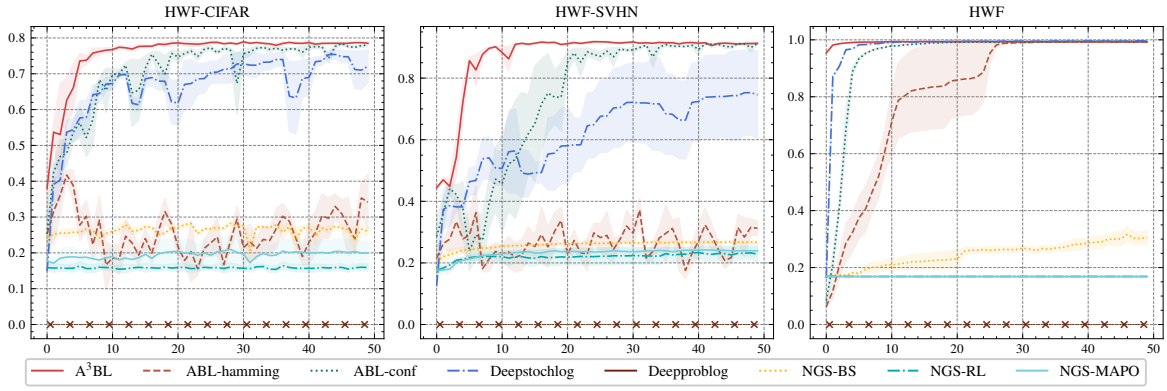


Figure 2. Performance curves (accuracy vs. epochs) of compared methods on handwritten formula recognition tasks. The shaded area represents the standard error of the methods over 5 repetitions. Cross marks denote experiments that were not finished within 48 hours.

to DeepProbLog but leveraging answer set programming (ASP) (Dimopoulos et al., 1997) in place of ProbLog. DeepStochLog (Winters et al., 2022) represents a related strategy to DeepProbLog, distinguishing itself by enhancing computational speed via a stochastic logic methodology. Additionally, our analysis encompassed various implementations of ABL. The ABL version utilizing the score function Equation (4) was referred to as ABL-hamming, and the one employing Equation (5) was termed ABL-conf. In our experimental setup, all methods utilize a consistent perception model for fairness in comparison. For the MNIST and KMNIST datasets, the selected perception model is LeNet (LeCun & Bengio, 1998). For the CIFAR10 and SVHN datasets, we employ ResNet50 (He et al., 2016). All these perception models are trained from scratch, and all compared methods were evaluated based on their official

implementations. The results for the Addition-CIFAR10 and Addition-SVHN tasks are depicted in Figure 1. Owing to space limitations, the results for the Addition-MNIST and Addition-KMNIST tasks are provided in Appendix C.

4.2. Settings of Handwritten Formula Recognition

Li et al. (2020) introduced the Handwritten Formula Recognition (HWF) task, which is based on the CROHME 2019 Offline Handwritten Formula Recognition Task[2]. The HWF dataset, derived from this task, encompasses training data composed of equations with various lengths and their corresponding evaluation results. The equations in the dataset have lengths in the set $\{1, 3, 5, 7\}$. In contrast to prior approaches, we enhance the task's difficulty by exclusively considering equations whose lengths are greater than or equal to 5. Furthermore, to augment the task's perceptual difficulty, we incorporate CIFAR10 (Krizhevsky, 2009) and SVHN (Netzer et al., 2011), thereby introducing the HWF-CIFAR and HWF-SVHN variants, respectively.

Compared Methods This study's selected comparative methods include the Neural-Grammar-Symbolic model (NGS) (Li et al., 2020), which adopts an approach akin to the ABL model by integrating symbolic reasoning with neural networks. NGS utilizes context-free grammar as its knowledge base and employs Markov Chain Monte Carlo (MCMC) sampling for candidate exploration in the solution space. We implemented various NGS variants as proposed by Li et al. (2020), including NGS-BS (employing back-search), NGS-RL (utilizing REINFORCE algorithm), and NGS-MAPO (with Memory Augmented Policy Optimization). Methods including DeepProbLog, DeepStochLog, ABL-hamming, and ABL-conf were also compared. In our experiments, all methods utilize a consistent perception model for fairness in comparison. For the HWF task, the selected perception model is LeNet (LeCun & Bengio, 1998). For the HWF-CIFAR and HWF-SVHN, we employ ResNet50 (He et al., 2016). All these perception models are trained from scratch, and all compared methods were evaluated based on their official implementations. It is important to recall that in this study, only equations with lengths greater than or equal to 5 are considered, a criterion set to augment the task's difficulty. Consequently, the results obtained in this research are expected to differ from those reported by Li et al. (2020).

4.3. Empirical Analysis

(a) ABL Suffers from Ambiguity The ambiguity of abduction can significantly impair the performance of ABL. To support this assertion, the following analysis is presented.

[2]https://www.cs.rit.edu/~crohme2019/task.html

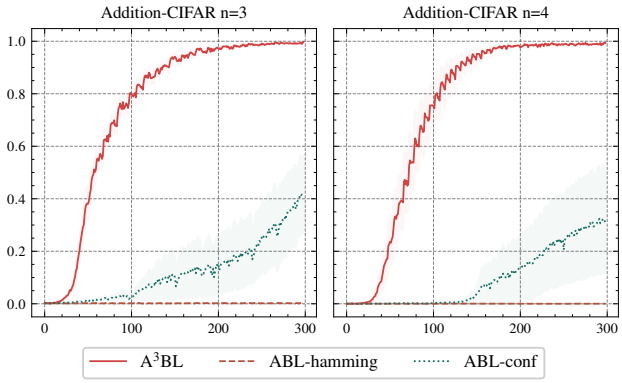
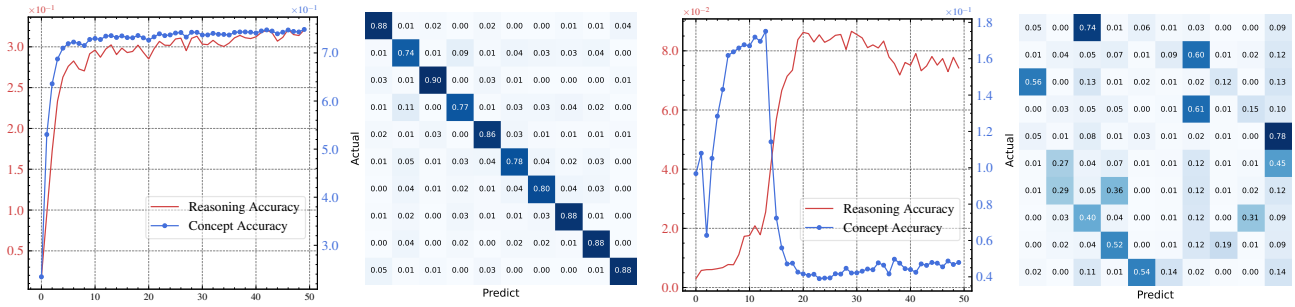


Figure 3. Abduction process (abduction accuracies vs. steps) of A^{3}BL and ABL on digit addition tasks. The shaded area represents the standard error of the methods over 5 repetitions.

- *Analysis of Performance Curve*. We investigate the performance of variants of ABL on two tasks: digit addition and handwritten formula recognition, as illustrated in Figure 1 and Figure 2, respectively. For the digit addition task, both ABL-conf and ABL-hamming encounter difficulties related to the inherent ambiguity in the abduction process. ABL-hamming, in particular, struggles to converge with increasing digit size and in complex perception scenarios, such as Addition-SVHN and Addition-CIFAR10. In contrast, ABL-conf is characterized by a high standard error, indicating an unstable training process. Similarly, in the handwritten formula recognition task, these patterns persist. ABL-hamming fails to converge effectively on the HWF-CIFAR and HWF-SVHN datasets, while ABL-conf again demonstrates a high standard error, underscoring the instability in its training process. However, in both tasks, A^{3}BL emerges as significantly superior in terms of performance. This consistency across different challenges highlights its robustness and adaptability in handling complex tasks.

- *Analysis of Worst Case.* The worst case, i.e., the worst performance run in repeated times. As demonstrated in Figure 4, the performance curves of A^{3}BL and ABL-conf exhibit divergent trends. A^{3}BL exhibits consistency between reasoning and concept accuracy, with both measures improving over time. Reasoning accuracy refers to the consistency between the model's prediction and the background knowledge, and concept accuracy refers to the instance-level classification. However, this is not the case for ABL-conf. In fact, the confusion matrix of ABL-conf reveals a *shortcut* pattern. This pattern may slightly improve the reasoning accuracy, but it negatively impacts the concept accuracy. This finding clearly highlights that ABL-conf encounters difficulties in handling ambiguity, resulting in the adoption of shortcuts.



(a) A^{3}BL **(b)** ABL-conf

Figure 4. Worst case analysis between A^{3}BL and ABL-conf. The concept accuracy of A^{3}BL increases along with reasoning accuracy, while the concept accuracy of ABL-conf could worsen as reasoning accuracy increases. Confusion matrices indicate that ABL-conf falls into a *shortcut*, i.e., the model *mistakenly* categorizes instances to enhance reasoning accuracy, while leading to bad concept accuracy.

Table 1. Test accuracies (mean $\pm$ standard deviation) on digit addition tasks with varying digit sizes. Each experiment was conducted five times. The best performance is denoted in boldface. 'N/A' indicates that the method failed to complete a single epoch within 48 hours.

	Addition-SVHN				Addition-CIFAR			
Method	$n=1$	$n=2$	$n=3$	$n=4$	$n=1$	$n=2$	$n=3$	$n=4$
NeurASP (Yang et al., 2020)	$6.63_{\pm 0.23}$	$7.84_{\pm 0.15}$	$7.84_{\pm 0.15}$	N/A	$10.00_{\pm 0.00}$	$9.73_{\pm 0.46}$	$9.73_{\pm 0.46}$	N/A
Deepproblog (Manhaeve et al., 2021)	$6.53_{\pm 1.12}$	N/A	N/A	N/A	$7.55_{\pm 1.26}$	N/A	N/A	N/A
DeepStochLog (Winters et al., 2022)	$91.61_{\pm 0.27}$	$90.81_{\pm 0.52}$	$90.28_{\pm 0.35}$	$90.92_{\pm 0.75}$	$66.32_{\pm 0.72}$	$67.63_{\pm 1.45}$	$65.55_{\pm 0.87}$	$66.38_{\pm 1.28}$
ABL-hamming (Dai et al., 2019)	$26.56_{\pm 11.67}$	$19.76_{\pm 0.14}$	$12.27_{\pm 9.97}$	$15.88_{\pm 5.59}$	$31.82_{\pm 15.71}$	$20.67_{\pm 3.93}$	$17.29_{\pm 2.23}$	$12.97_{\pm 2.60}$
ABL-conf	$91.02_{\pm 0.95}$	$64.66_{\pm 43.45}$	$65.12_{\pm 43.76}$	$65.00_{\pm 43.97}$	$68.42_{\pm 1.79}$	$62.94_{\pm 25.74}$	$58.29_{\pm 30.19}$	$62.52_{\pm 26.15}$
A^{3}BL (Ours)	$\mathbf{91.86}_{\pm \mathbf{0.86}}$	$\mathbf{91.81}_{\pm \mathbf{0.87}}$	$\mathbf{92.49}_{\pm \mathbf{0.37}}$	$\mathbf{91.99}_{\pm \mathbf{0.46}}$	$\mathbf{72.82}_{\pm \mathbf{0.95}}$	$\mathbf{76.09}_{\pm \mathbf{0.18}}$	$\mathbf{74.45}_{\pm \mathbf{0.37}}$	$\mathbf{72.57}_{\pm \mathbf{1.22}}$

(b) Accuracies of Abduction Figure 3 illustrates the abduction processes employed by A^{3}BL, ABL-hamming, and ABL-conf on digit addition tasks. The process is conducted over 300 steps, corresponding to 10 epochs. In this context, *abduction accuracy* is defined as follows: for ABL-conf and ABL-hamming, accuracy depends on whether the result of their abduction process identifies the correct candidate. For A^{3}BL, accuracy is contingent upon whether the candidate with the highest weight is indeed the correct one. ABL-hamming failed to find the correct candidate, and ABL-conf also struggled with this. However, A^{3}BL effectively distinguishes the correct candidate among abduction candidates through dynamic weight assignment. Within 10 epochs, the abduction accuracy of A^{3}BL achieves nearly 1, which brings a clearly supervision for the machine learning model.

(c) Greater Stability and Faster Convergence Experimental results demonstrate that A^{3}BL exceeds the performance of ABL variants, DeepProbLog, DeepStochLog, NGS variants, and NeurASP, exhibiting faster convergence and superior performance. In the task of digit addition, both NeurASP and DeepProbLog underperformed, failing to surpass a random classifier. Particularly, in certain settings, both methods failed to complete even a single training epoch within two days; this high computational complexity limits their applicability in realistic scenarios. In the task of handwritten formula recognition, all NGS variants failed, possibly due to the increased difficulty from limiting the length of equations to five or more. Similarly, DeepProbLog also failed to complete a training epoch within two days in this task. As previously discussed, A^{3}BL outperforms ABL variants in all tasks. To summarize, A^{3}BL beats other methods in terms of *stability* and *convergence speed*.

4.4. Efficiency Discussion

As shown in Theorem 3.2, the E-step in our algorithm can be computed efficiently, with only one step calculation. In other words, we use a fixed-form approach to derive this step. For comparison, the time complexity of the E-step in A^{3}BL is $\mathcal{O}(n)$, where n is the size of the candidate set. However, the time complexity of ABL to select a candidate is still $\mathcal{O}(n)$, since they have to select the minimal inconsistent candidate from the set. After that, both ABL and A^{3}BL will optimize their risk, which is with the same complexity see Equation (3) and Equation (9). In conclusion, the total time complexity of A^{3}BL is the same as that of ABL.

Furthermore, we report the execution time for a sample training session (encompassing 5 epochs) for the Addition-CIFAR task with n ranging from 1 to 4, see Table 2. The results were obtained using a computer setup consisting of an Intel Xeon Platinum 8538 CPU and an NVIDIA A100-PCIE-40GB GPU on an Ubuntu 20.04 Focal platform.

Table 2. Execution time comparison on task Addition-CIFAR.

Method	$n = 1$	$n = 2$	$n = 3$	$n = 4$
ABL	14m 4s	17m 15s	23m 12s	1h 3m 58s
A^3BL	14m 5s	19m 42s	27m 20s	1h 8m 49s

5. Conclusions and Future Directions

This study is the first to identify and articulate the issue of ambiguity within the Abductive Learning (ABL) framework, a challenge that arises due to the inherent ambiguity in the abduction process. To mitigate this challenge, we propose Ambiguity-Aware Abductive Learning (A^3BL) as a novel solution. A^3BL diverges from the conventional approach of selecting a single candidate at once; it considers all potential candidates, aggregating them into an instance-level class distribution, which is then optimized using the EM algorithm. This modeling approach enables A^3BL to efficiently utilize ambiguous abduction results, thereby enhancing the learning system. Furthermore, we establish an error bound, which guarantees the promising performance of A^3BL. Experimental results demonstrate that A^3BL utilizes abduction results more effectively, achieving a high abduction accuracy in the training set within a few iterations. Compared to other baseline methods, A^3BL demonstrates faster convergence, superior stability and better performance.

Although both experimental and theoretical analysis support the superior performance of A^3BL, it may still encounter failures in some scenarios, e.g., the size of the abduction candidate set is tremendous. For instance, if the knowledge base is helpless, the abduction candidate set becomes the universal set, leading to an overwhelmingly large number of candidates, which can overburden the learning system. The challenge may be tackled from two perspectives: The first is Inductive Logic Programming (ILP), which derives knowledge from data, thus improving the knowledge base and subsequently aiding the abduction process. The other involves developing algorithms to streamline the abduction process, e.g., parallel abductive reasoning. Also, applying this framework across a broader spectrum of realistic scenarios presents a promising avenue.

Impact Statement

This paper introduces Ambiguity-Aware Abductive Learning (A^3BL), a framework enhancing the integration of *logical reasoning* with *machine learning*. The advancement in A^3BL primarily contributes to the efficiency and effectiveness of neuro-symbolic learning, with broad applicability in sectors such as healthcare and autonomous systems. We anticipate that this work will not introduce any negative ethical or social impacts.

Acknowledgments

This paper is supported by NSFC (62076121, 61921006) and Major Program (JD) of Hubei Province (2023BAA024). The authors would like to thank Prof. Wang-Zhou Dai for his helpful feedback on drafts of the paper.

References

Artzi, Y. and Zettlemoyer, L. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013.

Cai, L.-W., Dai, W.-Z., Huang, Y.-X., Li, Y.-F., Muggleton, S., and Jiang, Y. Abductive Learning with Ground Knowledge Base. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pp. 1815–1821, Virtual, 2021.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep Learning for Classical Japanese Literature. *CoRR*, abs/1812.01718, 2018.

Cour, T., Sapp, B., and Taskar, B. Learning from Partial Labels. *Journal of Machine Learning Research*, 12(42): 1501–1536, 2011.

Dai, W.-Z. and Muggleton, S. Abductive Knowledge Induction from Raw Data. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pp. 1845–1851, Virtual, 2021.

Dai, W.-Z., Xu, Q., Yu, Y., and Zhou, Z.-H. Bridging Machine Learning and Logical Reasoning By Abductive Learning. In *Advances in Neural Information Processing Systems 32*, pp. 2815–2826, Vancouver, BC, Canada, 2019.

De Raedt, L., Kimmig, A., and Toivonen, H. ProbLog: a Probabilistic Prolog and Its Application in Link Discovery. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pp. 2468–2473, Hyderabad, India, 2007.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B*, 39: 1–38, 1977.

Deng, L. The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. Solving the Multiple Instance Problem with Axis-parallel Rectangles. *Artificial Intelligence*, 89(1–2):31–71, 1997.

Dimopoulos, Y., Nebel, B., and Koehler, J. Encoding Planning Problems in Nonmonotonic Logic Programs. In *Proceedings of the 4th European Conference on Planning*, pp. 169–181, Toulouse, France, 1997.

Feng, L., Lv, J., Han, B., Xu, M., Niu, G., Geng, X., An, B., and Sugiyama, M. Provably Consistent Partial-label Learning. In *Advances in Neural Information Processing Systems 33*, pp. 10948–10960, Virtual, 2020.

Gao, E.-H., Huang, Y.-X., Hu, W.-C., Zhu, X.-H., and Dai, W.-Z. Knowledge-enhanced Historical Document Segmentation and Recognition. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, pp. 8409–8416, Vancouver, BC, Canada, 2024.

Garcez, A. S. d., Gabbay, D. M., and Broda, K. B. *Neural-Symbolic Learning System: Foundations and Applications*. Springer-Verlag, Berlin, Heidelberg, 2002.

Hamming, R. W. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

He, H.-Y., Dai, W.-Z., and Li, M. Reduced Implication-bias Logic Loss for Neuro-symbolic Learning. *Machine Learning*, 113:3357–3377, 2024.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, 2016.

Huang, Y.-X., Dai, W.-Z., Yang, J., Cai, L.-W., Cheng, S., Huang, R., Li, Y.-F., and Zhou, Z.-H. Semi-supervised Abductive Learning and Its Application to Theft Judicial Sentencing. In *Proceedings of the 20th IEEE International Conference on Data Mining*, pp. 1070–1075, Sorrento, Italy, 2020.

Huang, Y.-X., Hu, W.-C., Gao, E.-H., and Jiang, Y. ABLkit: A Python Toolkit for Abductive Learning. *Frontiers of Computer Science*, in press.

Kahneman, D. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, USA, 2011.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, San Diego, CA, USA, 2015.

Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. *Technical Report, Department of Computer Science, University of Toronto*, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, Lake Tahoe, Nevada, USA, 2012.

LeCun, Y. and Bengio, Y. Convolutional Networks for Images, Speech, and Time Series. In *The Handbook of Brain Theory and Neural Networks*, pp. 255–258. MIT Press, Cambridge, MA, USA, 1998.

Li, Q., Huang, S., Hong, Y., Chen, Y., Wu, Y. N., and Zhu, S.-C. Closed Loop Neural-symbolic Learning via Integrating Neural Perception, Grammar Parsing, and Symbolic Reasoning. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5884–5894, Virtual, 2020.

Lu, N., Niu, G., Menon, A. K., and Sugiyama, M. On the Minimal Supervision for Training Any Binary Classifier from Only Unlabeled Data. In *International Conference on Learning Representations*, New Orleans, LA, USA, 2019.

Magnani, L. *Abductive Cognition - The Epistemological and Eco-Cognitive Dimensions of Hypothetical Reasoning*, volume 3 of *Cognitive Systems Monographs*. Springer, Berlin, Heidelberg, 2009.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. DeepProbLog: Neural Probabilistic Logic Programming. In *Advances in Neural Information Processing Systems 31*, pp. 3753–3763, Montréal, Canada, 2018.

Manhaeve, R., Dumančić, S., Kimmig, A., Demeester, T., and De Raedt, L. Neural Probabilistic Logic Programming in DeepProbLog. *Artificial Intelligence*, 298(C): 103504, 2021.

Maurer, A. A Vector-contraction Inequality for Rademacher Complexities. In *International Conference on Algorithmic Learning Theory*, pp. 3–17, Bari, Italy, 2016.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of Machine Learning*. MIT Press, 2012.

Mooney, R. J. Integrating Abduction and Induction in Machine Learning. In *Abduction and Induction: Essays on their Relation and Integration*, pp. 181–191. Springer Netherlands, Dordrecht, 2000.

Muggleton, S. H. and Bryant, C. H. Theory Completion Using Inverse Entailment. In *Proceedings of the 10th Inductive Logic Programming*, pp. 130–146, London, UK, 2000.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with Noisy Labels. In *Advances in Neural Information Processing Systems 26*, pp. 1196–1204, Lake Tahoe, Nevada, USA, 2013.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, Granada, Spain, 2011.

Roychowdhury, S., Diligenti, M., and Gori, M. Regularizing Deep Networks with Prior Knowledge: A Constraint-based Approach. *Knowledge-Based Systems*, 222:106989, 2021.

Sun, R. *Integrating Rules and Connectionism for Robust Commonsense Reasoning*, pp. 273. John Wiley & Sons, Inc., 1994.

Tao, L., Huang, Y.-X., Dai, W.-Z., and Jiang, Y. Deciphering Raw Data in Neuro-Symbolic Learning with Provable Guarantees. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, pp. 15310–15318, Vancouver, BC, Canada, 2024.

Towell, G. G. and Shavlik, J. W. Knowledge-based Artificial Neural Networks. *Artificial Intelligence*, 70(1):119–165, 1994.

van Krieken, E., Acar, E., and van Harmelen, F. Analyzing Differentiable Fuzzy Logic Operators. *Artificial Intelligence*, 302:103602, 2022.

Vapnik, V. N. An Overview of Statistical Learning Theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30*, pp. 6000–6010, Long Beach, CA, USA, 2017.

Wang, J., Deng, D., Xie, X., Shu, X., Huang, Y.-X., Cai, L.-W., Zhang, H., Zhang, M.-L., Zhou, Z.-H., and Wu, Y. Tac-Valuer: Knowledge-based Stroke Evaluation in Table Tennis. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3688–3696, Virtual Event, Singapore, 2021.

Wang, K., Tsamoura, E., and Roth, D. On Learning Latent Models with Multi-instance Weak Supervision. In *Advances in Neural Information Processing Systems 36*, New Orleans, LA, USA, 2023.

Wei, Z., Feng, L., Han, B., Liu, T., Niu, G., Zhu, X., and Shen, H. T. A Universal Unbiased Method for Classification from Aggregate Observations. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 36804–36820, Honolulu, Hawaii, USA, 2023.

Winters, T., Marra, G., Manhaeve, R., and Raedt, L. D. DeepStochLog: Neural Stochastic Logic Programming. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pp. 10090–10100, Virtual, 2022.

Wu, Z., Lv, J., and Sugiyama, M. Learning with Proper Partial Labels. *Neural Computation*, 35(1):58–81, 2023.

Xie, Z., Liu, Y., He, H.-Y., Li, M., and Zhou, Z.-H. Weakly Supervised AUC Optimization: A Unified Partial AUC Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–16, 2024.

Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Van den Broeck, G. A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5502–5511, Stockholm, Sweden, 2018.

Yang, X.-W., Shao, J.-J., Tu, W.-W., Li, Y.-F., Dai, W.-Z., and Zhou, Z.-H. Safe Abductive Learning in the Presence of Inaccurate Rules. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, volume 38, pp. 16361–16369, Vancouver, BC, Canada, 2024.

Yang, Z., Ishay, A., and Lee, J. NeurASP: Embracing Neural Networks into Answer Set Programming. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pp. 1755–1762, Yokohama, Japan, 2020.

Zhang, D., Han, J., Cheng, G., and Yang, M.-H. Weakly Supervised Object Localization and Detection: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5866–5885, 2022.

Zhang, M.-L. and Zhou, Z.-H. A Review on Multi-label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.

Zhou, Z.-H. A Brief Introduction to Weakly Supervised Learning. *National Science Review*, 5(1):44–53, 2017.

Zhou, Z.-H. Abductive Learning: Towards Bridging Machine Learning and Logical Reasoning. *Science China Information Sciences*, 62(7):76101, 2019.

Zhou, Z.-H. and Huang, Y.-X. Abductive Learning. In *Neuro-Symbolic Artificial Intelligence: The State of the Art*, pp. 353–369. IOS Press, Amsterdam, 2022.

Appendix

The structure of this appendix is as follows:

- Appendix A includes the proofs[3] omitted in the main text because of the space limit.
- Appendix B introduces the details of the implementation of our method and the comparison baselines.
- Appendix C provides more experiments about our method.

A. Proofs

A.1. Proving the Equality between Equation (10) and Equation (9)

Proof. The Equation (10) is

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{\boldsymbol{c}\in\boldsymbol{s}^{(i)}} p(\boldsymbol{c}|\boldsymbol{x}^{(i)},\boldsymbol{s}^{(i)})\cdot\mathcal{L}(f(\boldsymbol{x}^{(i)}),\boldsymbol{c})\,.$$

The Equation (9) is

$$\frac{1}{m\cdot N}\sum_{i=1}^{N}\sum_{j=1}^{m}\mathcal{L}_{\text{cls}}\left(f(x_j^{(i)}),p(z|x_j^{(i)},\boldsymbol{s}^{(i)})\right)\,.$$

To demonstrate the equality, we consider a single sample $(\boldsymbol{x},y)$.

On the one hand, we have:

$$\begin{aligned}\frac{1}{m}\sum_{j=1}^{m}\mathcal{L}_{\text{cls}}\left(f(x_j),p(z|x_j,\boldsymbol{s})\right) &= -\frac{1}{m}\sum_{j=1}^{m}\sum_{k=1}^{L}p(z=k|x_j,\boldsymbol{s})\log f(x_j)\\ &= -\frac{1}{m}\sum_{j=1}^{m}\sum_{k=1}^{L}\sum_{\boldsymbol{c}_i\in\boldsymbol{s}}\mathbb{I}[k=c_{ij}]p(\boldsymbol{z}=\boldsymbol{c}_i|\boldsymbol{x},\boldsymbol{s})\log f(x_j)\,.\end{aligned}\tag{14}$$

The second equality build upon:

$$p(z=k|x_j,\boldsymbol{s})=\sum_{\boldsymbol{c}_i\in\boldsymbol{s}}\mathbb{I}[k=c_{ij}]p(\boldsymbol{z}=\boldsymbol{c}_i|\boldsymbol{x},\boldsymbol{s})\,.$$

On the other hand, we also have:

$$\begin{aligned}\sum_{\boldsymbol{c}_i\in\boldsymbol{s}}p(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{s})\cdot\mathcal{L}(f(\boldsymbol{x}),\boldsymbol{c}_i) &= \sum_{\boldsymbol{c}_i\in\boldsymbol{s}}p(\boldsymbol{c}_i|\boldsymbol{x},\boldsymbol{s})\cdot\frac{1}{m}\sum_{j=1}^{m}\mathcal{L}_{\text{cls}}(f(x_j),c_{ij})\\ &= -\frac{1}{m}\sum_{\boldsymbol{c}_i\in\boldsymbol{s}}\sum_{j=1}^{m}\sum_{k=1}^{L}\mathbb{I}[k=c_{ij}]p(\boldsymbol{z}=\boldsymbol{c}_i|\boldsymbol{x},\boldsymbol{s})\log f(x_j)\,.\end{aligned}\tag{15}$$

Thus we prove the equality. □

Further, when the $p(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{s})$ is correctly estimated, we can prove that the Ambiguity-Aware Abductive Risk is actually an unbiased risk estimator.

[3]The similar techniques we used here are widely adopted in many literatures in the (weakly) supervised learning field. Specifically, the risk rewrite technology can be found in (Feng et al., 2020; Wu et al., 2023; Wei et al., 2023) and so on. The error bound technology can be found in (Vapnik, 1999; Mohri et al., 2012; Maurer, 2016; Lu et al., 2019; Feng et al., 2020; Wu et al., 2023) and so on.

Theorem A.1 (Unbiased Risk Estimator)**.** *The classification risk Definition 3.1 can be equivalently expressed as:*

$$R(f;\mathcal{L}) = \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{s})}\left[\sum_{\boldsymbol{c}\in\boldsymbol{s}} p(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{s})\mathcal{L}(f(\boldsymbol{x}),\boldsymbol{c})\right]. \tag{16}$$

We named this risk as Ambiguity-Aware Abductive Risk.

Proof.

$$\begin{aligned}
R(f;\mathcal{L}) &= \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{z})}\left[\mathcal{L}(f(\boldsymbol{x}),\boldsymbol{z})\right] && (17)\\
&= \int_{\mathcal{X}^m} \sum_{\boldsymbol{c}\in\mathcal{Z}^m} \mathcal{L}(f(\boldsymbol{x}),\boldsymbol{c})p(\boldsymbol{x},\boldsymbol{z}=\boldsymbol{c})\mathrm{d}\boldsymbol{x} && (18)\\
&= \int_{\mathcal{X}^m} \sum_{\boldsymbol{c}\in\mathcal{Z}^m} \sum_{\boldsymbol{s}\subseteq\mathcal{Z}^m} \mathcal{L}(f(\boldsymbol{x}),\boldsymbol{c})\cdot p(\boldsymbol{x},\boldsymbol{z}=\boldsymbol{c},\boldsymbol{s})\mathrm{d}x && (19)\\
&= \int_{\mathcal{X}^m} \sum_{\boldsymbol{c}\in\mathcal{Z}^m} \sum_{\boldsymbol{s}\subseteq\mathcal{Z}^m} \mathcal{L}(f(\boldsymbol{x}),\boldsymbol{c})\cdot p(\boldsymbol{z}=\boldsymbol{c}|\boldsymbol{x},\boldsymbol{s})\cdot p(\boldsymbol{x},\boldsymbol{s})\mathrm{d}x && (20)\\
&= \int_{\mathcal{X}^m} \sum_{\boldsymbol{c}\in\mathcal{Z}^m} \sum_{\boldsymbol{s}\subseteq\mathcal{Z}^m} \mathcal{L}(f(\boldsymbol{x}),\boldsymbol{c})\cdot p(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{s})\cdot p(\boldsymbol{x},\boldsymbol{s})\mathrm{d}x && (21)\\
&= \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{s})}\left[\sum_{\boldsymbol{c}\in\boldsymbol{s}} p(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{s})\cdot\mathcal{L}(f(\boldsymbol{x}),\boldsymbol{c})\right]. && (22)
\end{aligned}$$

□

A.2. Proof of Theorem 3.2

For reading convenience, we rewrite the theorem below:

Theorem A.2. *The optimization of Equation (10) is actually optimizing the log-likelihood* $\sum_{i=1}^{N}\log p_\theta(\boldsymbol{x}^{(i)},y^{(i)})$*, via the EM algorithm.*

Proof.

$$\begin{aligned}
\log p_\theta(\boldsymbol{x}^{(i)},y^{(i)}) &= \log\left(\sum_{\boldsymbol{c}\in\boldsymbol{s}^{(i)}} p_\theta(\boldsymbol{x}^{(i)},y^{(i)},\boldsymbol{c})\right) && (23)\\
&= \log\left(\sum_{\boldsymbol{c}\in\boldsymbol{s}^{(i)}} p_\theta(\boldsymbol{x}^{(i)},\boldsymbol{c})\cdot p_\theta(y^{(i)}|\boldsymbol{x}^{(i)},\boldsymbol{c})\right) && (24)\\
&= \log\left(\sum_{\boldsymbol{c}\in\boldsymbol{s}^{(i)}} p_\theta(\boldsymbol{x}^{(i)},\boldsymbol{c})\right) && (25)\\
&= \log\left(\sum_{\boldsymbol{c}\in\boldsymbol{s}^{(i)}} w_{\boldsymbol{c}}\frac{p_\theta(\boldsymbol{x}^{(i)},\boldsymbol{c})}{w_{\boldsymbol{c}}}\right) && (26)\\
&\geq \sum_{\boldsymbol{c}\in\boldsymbol{s}^{(i)}} w_{\boldsymbol{c}}\log(\frac{p_\theta(\boldsymbol{x}^{(i)},\boldsymbol{c})}{w_{\boldsymbol{c}}}) && (27)
\end{aligned}$$

The second equality holds because $p_\theta(y^{(i)}|\boldsymbol{x}^{(i)},\boldsymbol{c})$ should always be equal to 1, as we can derive $y^{(i)}$ through candidate $\boldsymbol{c}$ using background knowledge, e.g., $\sigma(\boldsymbol{c}) = y^{(i)}$.

The last inequality is derived from Jensen's inequality.

Suppose the size of candidate set $\boldsymbol{s}^{(i)} = l_i$, when the equality holds, which means:

$$\frac{p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c}_1)}{w_{\boldsymbol{c}_1}} = \cdots = \frac{p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c}_{l_i})}{w_{\boldsymbol{c}_{l_i}}} = C. \tag{28}$$

Thus we can get:

$$\sum_{j=1}^{l_i} \frac{p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c}_j)}{C} = \sum_{j=1}^{l_i} w_{\boldsymbol{c}_j} = 1. \tag{29}$$

So we have:

$$\sum_{j=1}^{l_i} p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c}_j) = C. \tag{30}$$

It is easy to see that, $p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c}_j, \boldsymbol{s}^{(i)}) = p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c}_j)$ because we can derive the whole candidate set from one candidate by using the knowledge base, and further we get:

$$\sum_{j=1}^{l_i} p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c}_j, \boldsymbol{s}^{(i)}) = C = p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{s}^{(i)}). \tag{31}$$

Finally, we get:

$$\frac{p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c})}{p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{s}^{(i)})} = w_{\boldsymbol{c}} = \frac{p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{c}, \boldsymbol{s}^{(i)})}{p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{s}^{(i)})} = p(\boldsymbol{c}|\boldsymbol{x}^{(i)}, \boldsymbol{s}^{(i)}). \tag{32}$$

The formulation presented above indicates that by setting $w_{\boldsymbol{c}}$ as the candidate confidence introduced in Equation (7), we are performing the E-step in the Expectation-Maximization (EM) algorithm. After the E-step, the values of $w_{\boldsymbol{c}}$ are fixed, and we proceed to optimize Equation (10) in the M-step in order to maximize the likelihood. This supports our earlier statement. Consider that EM algorithm is well established that has convergence promise, this implies that the optimization of A^{3}BL can be guaranteed.

□

A.3. Proof of Theorem 3.3

Most of the prove tricks can be found in Mohri et al. (2012), especially the proof here were inspired by Wu et al. (2023) and Feng et al. (2020). For reading convenience, we rewrite the theorem below:

Theorem A.3 (Error bound)**.** *Suppose that $\mathcal{L}(f(\boldsymbol{x}), \boldsymbol{z})$ is ρ-Lipschitz with respect to $f(\boldsymbol{x})$ for all $\boldsymbol{z} \in \mathcal{Z}^m$ and upper-bounded by M, i.e., $M = \sup_{f\in\mathcal{F}, \boldsymbol{x}\in\mathcal{X}^m, \boldsymbol{z}\in\mathcal{Z}^m} \mathcal{L}(f(\boldsymbol{x}), \boldsymbol{z})$. Let $\mathfrak{R}_n(\mathcal{F}_i)$ be the Rademacher complexity of $\mathcal{F}_i$ with sample size n. Then for any $\delta > 0$, with probability at least $1 - \delta$,*

$$R(\hat{f}_{A^3}) - R(f^\star) \leq 4\sqrt{2}\rho \sum_{i=1}^{K} \mathfrak{R}_n(\mathcal{F}_i) + 2M\sqrt{\frac{\log(2/\delta)}{2n}}. \tag{33}$$

To prove the Theorem 3.3, we first introduce the following lemma:

Lemma A.4. *The following inequality holds:*

$$0 \leq R(\hat{f}_{A^3}) - R(f^\star) \leq 2\sup_f |R(f) - R_{A^3}(\hat{f})|. \tag{34}$$

Proof. By definition, $R(\hat{f}_{\mathrm{A}^3}) - R(f^\star) \geq 0$, thus the first inequality is proved.

Since,

$$R(\hat{f}_{\mathrm{A}^3}) - R(f^\star) = (R(\hat{f}_{\mathrm{A}^3}) - R_{\mathrm{A}^3}(\hat{f}_{\mathrm{A}^3})) + (R_{\mathrm{A}^3}(\hat{f}_{\mathrm{A}^3}) - R_{\mathrm{A}^3}(f^\star)) + (R_{\mathrm{A}^3}(f^\star) - R(f^\star)), \tag{35}$$

$$\leq (R(\hat{f}_{\mathrm{A}^3}) - R_{\mathrm{A}^3}(\hat{f}_{\mathrm{A}^3})) + (R_{\mathrm{A}^3}(f^\star) - R(f^\star)), \tag{36}$$

$$\leq 2 \sup_f |R(f) - R_{\mathrm{A}^3}(\hat{f})|, \tag{37}$$

thus proving the second inequality. □

Definition A.5 (Empirical Rademacher Complexity)**.** Let $\mathcal{G}$ be a class of functions mapping $\mathcal{Z} \mapsto \mathbb{R}$ and $S = (z_1, \cdots, z_n) \in \mathcal{Z}^n$ a fixed sample of size n. Then the empirical Rademacher complexity of $\mathcal{G}$ with respect to the sample S is defined as:

$$\hat{\mathfrak{R}}_S(\mathcal{G}) = \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i g(z_i) \right], \tag{38}$$

where $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_n)$, with σ_i is independent uniform random variables taking from $\{-1, +1\}$.

Definition A.6 (Rademacher Complexity)**.** Suppose the sample S with size n is drawn from distribution p i.i.d. The Rademacher complexity of $\mathcal{G}$ with respect to p is defined as:

$$\mathfrak{R}_n(\mathcal{G}) = \mathbb{E}_{z_i \sim p} \left[\hat{\mathfrak{R}}_S(\mathcal{G}) \right]. \tag{39}$$

We introduce a class of functions defined on $\mathcal{X}^m \times \mathcal{Z}^m$ according to Equation (7):

$$\mathcal{G} = \{(\boldsymbol{x}, \boldsymbol{z}) \mapsto \sum_{\boldsymbol{c} \in \boldsymbol{s}} p(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{s}) \mathcal{L}(f(\boldsymbol{x}), \boldsymbol{c}) : f \in \mathcal{F}\}. \tag{40}$$

Then the Rademacher complexity of $\mathcal{G}$ with respect to $p(\boldsymbol{x}, \boldsymbol{c})$ is given as:

$$\mathfrak{R}_n(\mathcal{G}) = \mathbb{E}_{(\boldsymbol{x}_i, \boldsymbol{c}_i) \sim p} \left[\mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i g(\boldsymbol{x}_i, \boldsymbol{c}_i) \right] \right]. \tag{41}$$

Lemma A.7. *Suppose $M = \sup_{\boldsymbol{x} \in \mathcal{X}^m, \boldsymbol{z} \in \mathcal{Z}^m, f \in \mathcal{F}} \mathcal{L}(f(\boldsymbol{x}), \boldsymbol{z}) < \infty$, then for any $\delta > 0$, the following inequality holds with the probability at least $1 - \delta$.*

$$\sup_{f \in \mathcal{F}} |R(f) - \hat{R}_{A^3}(f)| \leq 2\mathfrak{R}_n(\mathcal{G}) + M \sqrt{\frac{\log(2/\delta)}{2n}}. \tag{42}$$

Proof. For a sample S, we define $\phi(S) = \sup_{f \in \mathcal{F}} (R(f) - \hat{R}_{\mathrm{A}^3}(f))$. Suppose we replace an example $(\boldsymbol{x}_i, \boldsymbol{c}_i)$ in the sample S with another example $(\boldsymbol{x}'_i, \boldsymbol{c}'_i)$, the change of $\phi(S)$ is not greater than:

$$\sup_{g \in \mathcal{G}} \frac{g(\boldsymbol{x}_i, \boldsymbol{c}_i) - g(\boldsymbol{x}'_i, \boldsymbol{c}'_i)}{n} \leq \frac{M}{n}, \tag{43}$$

since $\mathcal{L}$ is bounded by M. Then by *McDiarmid's inequality*, for any $\delta > 0$, with a probability at least $1 - \frac{\delta}{2}$, the following inequality holds:

$$\phi(S) \leq \mathbb{E}_{(\boldsymbol{x}_i, \boldsymbol{c}_i) \sim p}[\phi(S)] + M \sqrt{\frac{\log(2/\delta)}{2n}}. \tag{44}$$

By Chapter 3 in Mohri et al. (2012), it is easy to show that $\mathbb{E}_{(\boldsymbol{x}_i, \boldsymbol{c}_i) \sim p} [\phi(S)] \leq 2\mathfrak{R}_n(\mathcal{G})$. Hence the following holds with probability at least $1 - \delta/2$:

$$\sup_{f \in \mathcal{F}} (R(f) - \hat{R}_{\mathrm{A}^3}(f)) \leq 2\mathfrak{R}_n(\mathcal{G}) + M \sqrt{\frac{\log(2/\delta)}{2n}}, \tag{45}$$

thus complete the proof. □

Lemma A.8. *Suppose that the loss $\mathcal{L}(f(\boldsymbol{x}), \boldsymbol{z})$ is ρ-Lipschitz with respect to $f(\boldsymbol{x})$ for all $\boldsymbol{z} \in \mathcal{Z}^m$. Then the following inequality holds:*

$$\mathfrak{R}_n(\mathcal{G}) \leq \sqrt{2}\rho \sum_{i=1}^{K} \mathfrak{R}_n(\mathcal{F}_i). \tag{46}$$

Proof. Let $\Pi = \{(\boldsymbol{x}, \boldsymbol{c}) \mapsto \mathcal{L}(f(\boldsymbol{x}), \boldsymbol{c}) : f \in \mathcal{F}\}$. Notice that the candidate confidence $p(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{s})$ is between 0 and 1, and that $\sum_{\boldsymbol{c} \in \boldsymbol{s}} p(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{s}) = 1$. In this way, we can obtain:

$$\mathfrak{R}_n(\mathcal{G}) = \mathbb{E}_{(\boldsymbol{x}_i, \boldsymbol{c}_i) \sim p} \left[\mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i g(\boldsymbol{x}_i, \boldsymbol{c}_i) \right] \right], \tag{47}$$

$$\leq \mathbb{E}_{(\boldsymbol{x}_i, \boldsymbol{c}_i) \sim p} \left[\mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \mathcal{L}(f(\boldsymbol{x}_i), \boldsymbol{c}_i) \right] \right], \tag{48}$$

$$= \mathfrak{R}_n(\Pi). \tag{49}$$

Since $\mathcal{L}$ is ρ-Lipschitz, following the Rademacher vector contraction inequality (Maurer, 2016), we have:

$$\mathfrak{R}_n(\Pi) \leq \sqrt{2}\rho \sum_{i=1}^{K} \mathfrak{R}_n(\mathcal{F}_i), \tag{50}$$

which concludes this proof. □

Finally, the proof of Theorem 3.3 can be completed by combining the above lemmas.

B. Implementation Details

All experiments were conducted on a system equipped with an NVIDIA GeForce RTX 3090 GPU, Intel Xeon Silver 4210 CPU, 64GB of RAM, and Ubuntu 20.04 Focal. All experiments use Adam (Kingma & Ba, 2015) as the optimizer. For convenience, a summary table detailing the configurations of various methods is presented in Table 3.

Table 3. For the experimental setup, the configurations for the methods compared were chosen according to their official implementations, and A^{3}BL aligns with the standard configurations of ABL.

	Digit Addition				Handwritten Formula Recognition			
Method	Optimizer	Learning rate	Batch size	Epoch	Optimizer	Learning rate	Batch size	Epoch
A^{3}BL	Adam	0.001	256	50	Adam	0.001	1024	50
ABL-hamming	Adam	0.001	256	50	Adam	0.001	1024	50
ABL-conf	Adam	0.001	256	50	Adam	0.001	1024	50
DeepProbLog	Adam	0.001	128	50	Adam	0.001	32	50
DeepStochLog	Adam	0.001	256	50	Adam	0.003	32	50
NeurASP	Adam	0.001	1000	50	-	-	-	-
NGS	-	-	-	-	Adam	0.0005	64	50

B.1. Details of Datasets

For convenience, the information of datasets can be seen in Table 4.

Construction of Digit Addition The construction of the digit addition task is based on a ten-class dataset, such as MNIST. To construct a digit addition input $\boldsymbol{x}$, we randomly select $2 * n$ images from the dataset, where n represents the digit-size. The summation y of this input can be computed using digit addition rules. The size of the constructed equations remains constant across all digit-sizes. To ensure this consistency, each image is sampled n times. Consequently, the size of the constructed equations is invariably fixed to $N/2$, where N denotes the total number of images in the original dataset.

Construction of Handwritten Formula Recognition The development of the handwritten formula recognition task leverages the foundational implementation outlined by Li et al. (2020). In our experiments, we replace the corresponding digit with a randomly sampled image from a specific class within either the CIFAR or SVHN dataset. To ensure consistent training, for both CIFAR and SVHN datasets, we transform the original operator images from a dimension of 45×45 to $3 \times 32 \times 32$.

Table 4. Dataset details for Handwritten Formula Recognition and Digit Addition tasks.

Task	#instances of a sequence	#sequence of training set	#sequence of test set
Handwritten Formula Recognition	5 or 7	8000	1600
Digit Addition, n=1	2	30000	10000
Digit Addition, n=2	4	30000	10000
Digit Addition, n=3	6	30000	10000
Digit Addition, n=4	8	30000	10000

B.2. Implementation of A^{3}BL

The pseudo-code of A^{3}BL can be referred in Algorithm 1.

Abduction Process The official implementation of the ABL package implements the abduction process through zeroth-order optimization, abduction search, or by pre-building the knowledge base, that is, storing all possible candidates concerning the potential target label y. A^{3}BL adopts the same abduction process, which is built upon their implementations.

Algorithm 1 A^{3}BL Algorithm

Require: Knowledge base KB, Perception model f, dataset $(\hat{\mathcal{X}}^m, \hat{\mathcal{Y}})$.

for each pair $(\boldsymbol{x}^{(i)}, y^i)$ in $(\hat{\mathcal{X}}^m, \hat{\mathcal{Y}})$ **do**
  $\hat{\boldsymbol{z}}^{(i)} \leftarrow f(\boldsymbol{x}^{(i)})$
  $\hat{y}^{(i)} \leftarrow$ KB.logical_forward$(\hat{\boldsymbol{z}}^{(i)})$
  if $y^{(i)} \neq \hat{y}^{(i)}$ **then**
    $\boldsymbol{s}^{(i)} \leftarrow$ KB.abduce$(y^{(i)}, \hat{\boldsymbol{z}}^i)$ # Abduce all candidates that valid.
  end if
  loss $\leftarrow \frac{1}{N}\sum_{i=1}^{N}\sum_{\boldsymbol{c}\in\boldsymbol{s}^{(i)}} p_\theta(\boldsymbol{c}|\boldsymbol{x}^{(i)}, \boldsymbol{s}^{(i)}) \cdot \mathcal{L}(f(\boldsymbol{x}^{(i)}), \boldsymbol{c})$. # Calculate the ambiguity-aware abductive risk.
  Update the perception model f
end for

Candidate Confidence For the computation of Equation (7), the equation can be reformulated as follows:

$$p(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{s}) = \frac{p(\boldsymbol{c}, \boldsymbol{x}, \boldsymbol{s})}{p(\boldsymbol{x}, \boldsymbol{s})} = \frac{p(\boldsymbol{c}, \boldsymbol{s})}{p(\boldsymbol{s}, \boldsymbol{x})} = \frac{p(\boldsymbol{c}|\boldsymbol{x})}{p(\boldsymbol{s}|\boldsymbol{x})}. \tag{51}$$

The validity of the second equation stems from the capability to generate the entire candidate set from a singular candidate by leveraging the knowledge base. Consider the case SUM(1, 2), where a candidate is given as 1 $= 2$, 2 $= 1$. Although this candidate may not be correct, it demonstrates that the SUM(1, 2) $= 3$, thus can derive the entire candidate set.

Consequently, a parameterized model, exemplified by f, is utilized to estimate the final term $p(\boldsymbol{c}|\boldsymbol{x})/p(\boldsymbol{s}|\boldsymbol{x}) = p_\theta(\boldsymbol{c}|\boldsymbol{x})/p_\theta(\boldsymbol{s}|\boldsymbol{x})$. The denominator term $p_\theta(\boldsymbol{s}|\boldsymbol{x})$ is calculated using the equation $p_\theta(\boldsymbol{s}|\boldsymbol{x}) = \sum_{\boldsymbol{c}\in\boldsymbol{s}} p_\theta(\boldsymbol{c}|\boldsymbol{x})$. The term $p_\theta(\boldsymbol{c}|\boldsymbol{x})$ is computed utilizing a neural network. Subsequently, all candidate confidences are normalized using a softmax function with temperature adjustment. In practice, the temperature is set to 0.3 without careful tuning .

Challenge: The Expanding Size of the Candidate Set A significant challenge inherent in our method is the expansion of the candidate set size with increasing background knowledge complexity, as illustrated in Figure 5. For instance, in the digit addition task, increasing the number of digits from one to two causes the average size of the candidate set to grow from approximately five to nearly fifty, potentially overburdening the learning system.

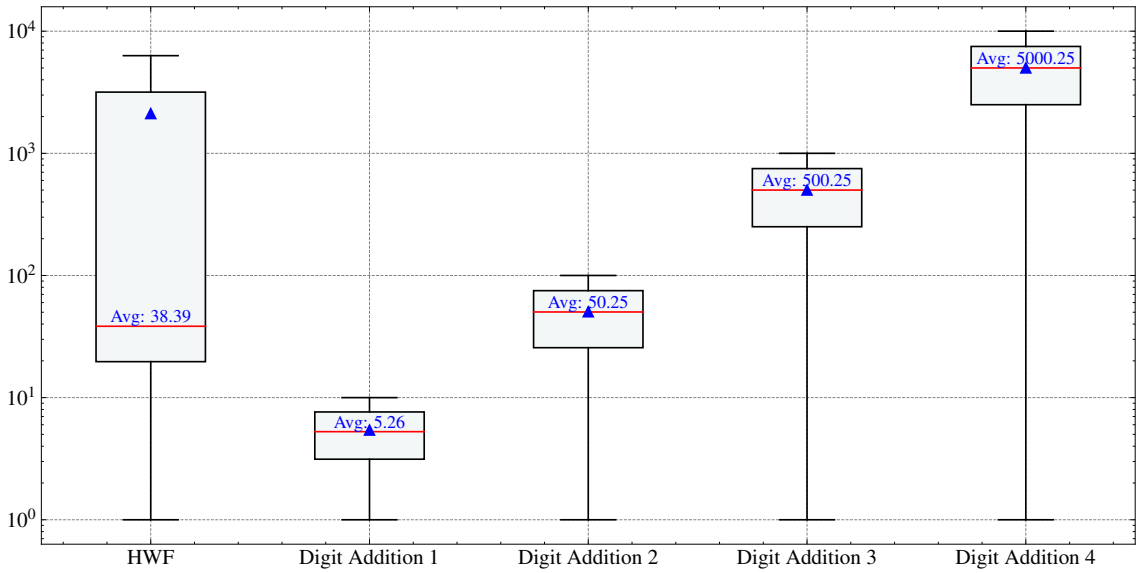


Figure 5. Variation in the size of abduction candidates across different tasks. This reveals two key observations: (a) As the size of the digits increases, the size of the abduction candidate set grows exponentially; (b) The size of the abduction candidate set for the HWF task varies from 1 to approximately 10^4, yet the average size remains relatively small.

Speed Up In Equation (10), different weights are assigned to various candidates based on their confidence levels. However, a significant number of candidates exhibit low confidence, resulting in minimal contribution to the optimization objective. To speed up, our objective is to balance between the consistency of risk and the efficiency of the training process. Consequently, a top-k selection, based on confidence, is introduced within the candidate set to facilitate algorithm implementation. While further tuning of this hyper-parameter could be beneficial to balance efficiency and performance, in this study, we chose $k = 32$ for all experiments.

Sensitivity Analysis about k To balance risk consistency and computational efficiency, a top-k selection, based on confidence, is introduced within the candidate set to facilitate algorithm implementation. Intuitively, as k increases, the derived risk tends to be unbiased, potentially leading to faster or more stable algorithm convergence.

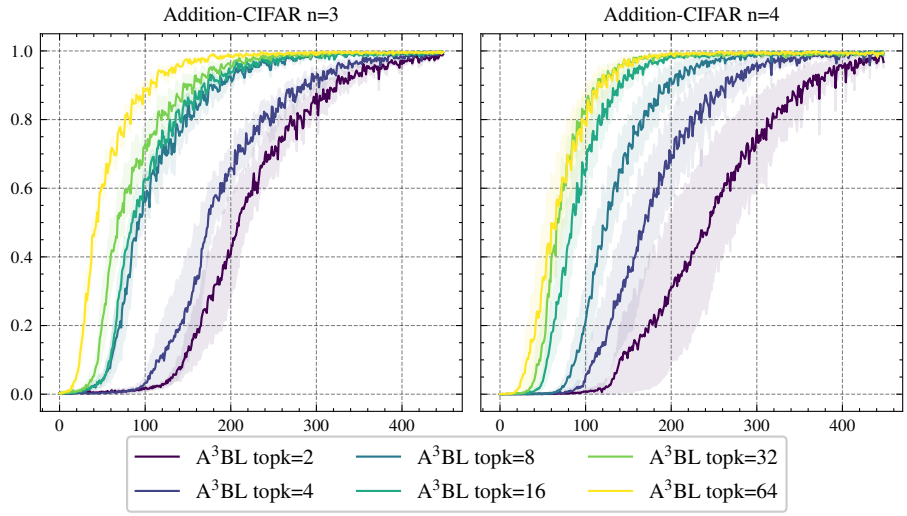


Figure 6. Abduction process: x-axis refers the steps, y-axis refers the abduction accuraies. The abduction accuracy means whether the candidate with the highest weight is indeed the correct one.

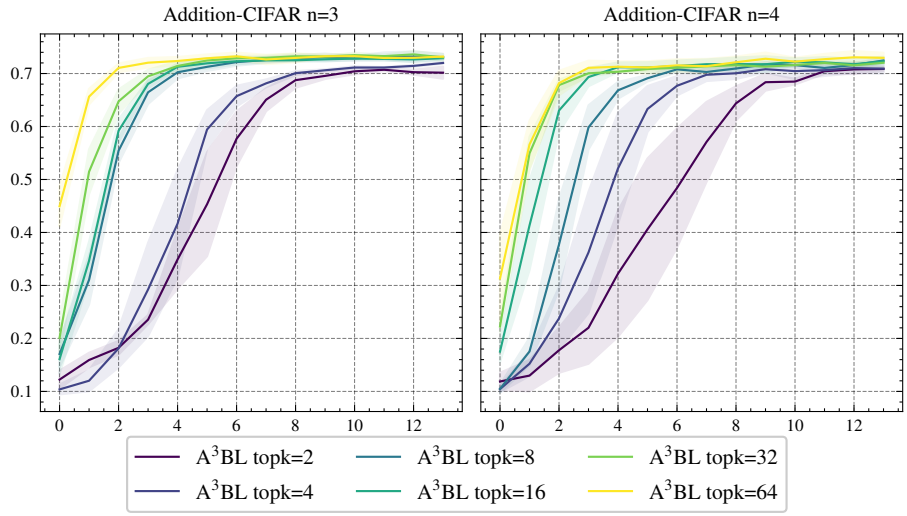


Figure 7. Test accuracies: x-axis refers the epoches, y-axis refers the (concept) test accuraies.

As shown in the results in Figure 6 and Figure 7. For a small k, the model behaves similarly to a naive ABL approach, exhibiting high variance. Increasing k to a moderate value, such as 8, stabilizes performance and speeds up convergence. Further increasing k beyond a certain threshold does not significantly improve performance, indicating a saturation point.

B.3. Implementation of Compared Methods

The implementations of the compared methods are all based on their official implementations. To ensure a fair comparison, we may modify certain backbones of the machine learning models to maintain consistency across all methods.

Specifically:

- The implementation of ABL-hamming and ABL-conf is based on their official package. [4]
- The implementation of NGS is based on their official package.[5]
- The implementation of DeepStochLog is based on their official package.[6]
- The implementation of DeepProbLog is based on their official package.[7]
- The implementation of NeurASP is based on their official package.[8]

Some methods, such as DeepStochLog, DeepProbLog, and NeurASP, require the writing of Prolog-like knowledge bases, which are listed below.

```
digit(Y) :- member(Y,[0,1,2,3,4,5,6,7,8,9]).
nn(number, [X], Y, digit) :: is_number(Y) --> [X].
addition(N) --> is_number(N1), is_number(N2), {N is N1 + N2}.
multi_addition(N, 1) --> addition(N).
multi_addition(N, L) --> {L > 1, L2 is L - 1}, addition(N1), multi_addition(N2, L2), {N is
     N1*(10**L2) + N2}.
```

Listing 1. Digit Addition knowledge base for DeepStochLog

```
dom_number(X) :- member(X, [0,1,2,3,4,5,6,7,8,9]).
nn(number, [X], Y, dom_number) :: is_number(Y) --> [X].

dom_operator(X) :- member(X, [plus, minus, times, div]).
nn(operator, [X], Y, dom_operator) :: operator(Y) --> [X].
factor(N) --> is_number(N).

0.34 :: term(N) -->  factor(N).
0.33 :: term(N) -->  term(N1), operator(times), factor(N2), {N is N1 * N2}.
0.33 :: term(N) -->  term(N1), operator(div), factor(N2), {N2>0, N is N1 / N2}.

0.34 :: expression(N) -->  term(N).
0.33 :: expression(N) -->  expression(N1), operator(plus), term(N2), {N is N1 + N2}.
0.33 :: expression(N) -->  expression(N1), operator(minus), term(N2), {N is N1 - N2}.
```

Listing 2. Handwritten Formula Recognition knowledge base for DeepStochLog

[4] https://github.com/AbductiveLearning/ABLkit/tree/Dev
[5] https://github.com/liqing-ustc/NGS
[6] https://github.com/ML-KULeuven/deepstochlog/tree/main/examples
[7] https://github.com/ML-KULeuven/deepproblog/tree/master/src/deepproblog/examples
[8] https://github.com/azreasoners/NeurASP/tree/master/examples

```
nn(mnist_net,[X],Y,[0,1,2,3,4,5,6,7,8,9]) :: digit(X,Y).

number([],Result,Result).
number([H|T],Acc,Result) :- digit(H,Nr), Acc2 is Nr+10*Acc,number(T,Acc2,Result).
number(X,Y) :- number(X,0,Y).

multi_addition(X,Y,Z) :- number(X,X2),number(Y,Y2), Z is X2+Y2.
addition(X,Y,Z) :- digit(X,X2), digit(Y,Y2), Z is X2+Y2.
```

Listing 3. Digit Addition knowledge base for DeepProbLog

```
nn(net1,[X],Y,[0,1,2,3,4,5,6,7,8,9]) :: detect_number(X,Y).
nn(net2,[X],Y,[+,-,*,/]) :: detect_operator(X,Y).

detect_all([N],[N2]) :- detect_number(N,N2).
detect_all([N,O|T],[N2,O2|T2]) :- detect_number(N,N2),
detect_operator(O,O2), detect_all(T,T2).

almost_equal(X,Y) :- ground(Y),abs(X-Y) < 0.0001.
almost_equal(X,Y) :- var(Y), Y is float(X).

expression(Images,Result) :-  detect_all(Images,Symbols),parse(Symbols,Result).

parse([N],R) :-almost_equal(N,R).

parse([N1,+|T], R) :- parse(T,R2), almost_equal(N1+R2,R).

parse([N1,-|T], R) :- parse([-1,*|T],R2), almost_equal(N1+R2,R).

parse([N1,*,N2|T], R) :- N3 is N1*N2, parse([N3|T],R).

parse([N1,/,N2|T], R) :- N2 \== 0, N3 is N1/N2, parse([N3|T],R).
```

Listing 4. Handwritten Formula Recognition knowledge base for DeepProbLog

```
img(i1). img(i2).
addition(A,B,N) :- digit(0,A,N1), digit(0,B,N2), N=N1+N2.
nn(digit(1,X), [0,1,2,3,4,5,6,7,8,9]) :- img(X).
```

Listing 5. Digit Addition knowledge base for NeurASP

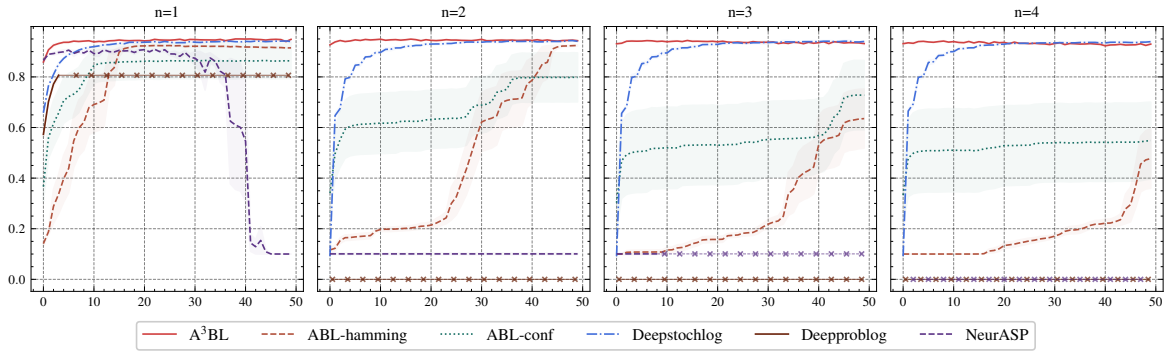
C. More Experiments

This section includes additional experiments not covered in the main text due to space limitations, as outlined below:

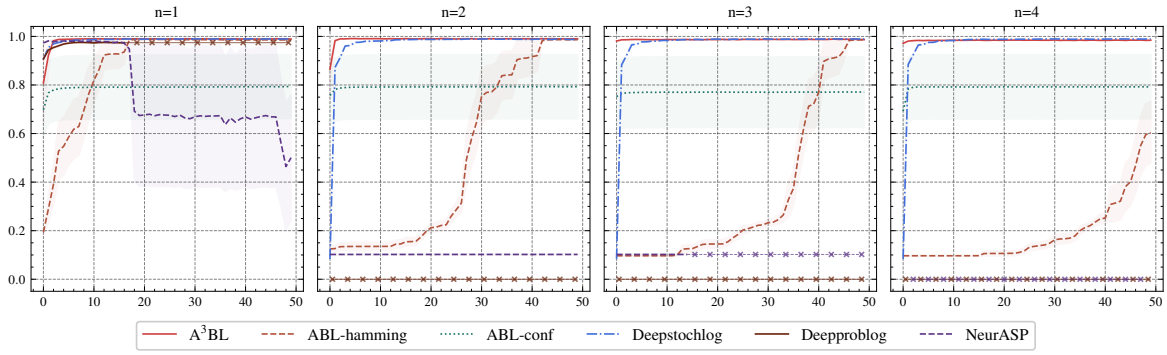
(a) Digit Addition The additional experimental results concerning Addition-MNIST and Addition-KMNIST are presented in Figure 8 and Table 5, respectively. The conclusion remains the same as for Addition-CIFAR and Addition-SVHN: ABL suffers from ambiguity in the abduction candidates, while A^{3}BL can effectively utilize the abduction candidates. It is also worth noting that DeepStochLog performs well in these settings. However, A^{3}BL converges faster and exhibits a smoother learning curve.

Table 5. Test accuracies (mean $\pm$ standard deviation) on digit addition task with different perception dataset. Each experiment was conducted five times. The best performance is denoted in boldface. 'N/A' indicates that the method failed to complete a single epoch within 48 hours.

	Addition-MNIST				Addition-KMNIST			
Method	1	2	3	4	1	2	3	4
DeepStochLog (Winters et al., 2022)	$\mathbf{99.00_{\pm 0.13}}$	$98.88_{\pm 0.02}$	$\mathbf{98.94_{\pm 0.15}}$	$\mathbf{98.98_{\pm 0.13}}$	$94.00_{\pm 0.73}$	$94.06_{\pm 0.30}$	$\mathbf{94.00_{\pm 0.29}}$	$\mathbf{93.93_{\pm 0.49}}$
NeurASP (Yang et al., 2020)	$50.01_{\pm 0.44}$	$10.21_{\pm 0.39}$	$10.21_{\pm 0.39}$	N/A	$10.01_{\pm 0.58}$	$10.05_{\pm 0.51}$	N/A	N/A
Deepproblog (Manhaeve et al., 2021)	$97.42_{\pm 0.29}$	N/A	N/A	N/A	$80.54_{\pm 0.21}$	N/A	N/A	N/A
ABL-hamming (Dai et al., 2019)	$98.49_{\pm 0.22}$	$98.65_{\pm 0.08}$	$98.63_{\pm 0.12}$	$60.41_{\pm 40.27}$	$91.48_{\pm 0.54}$	$92.43_{\pm 0.65}$	$63.51_{\pm 36.91}$	$47.74_{\pm 37.13}$
ABL-conf	$79.31_{\pm 41.35}$	$79.32_{\pm 41.61}$	$77.10_{\pm 43.53}$	$79.22_{\pm 41.65}$	$86.32_{\pm 25.40}$	$79.80_{\pm 34.91}$	$72.76_{\pm 38.75}$	$54.84_{\pm 42.97}$
A^{3}BL (Ours)	$98.92_{\pm 0.15}$	$\mathbf{99.03_{\pm 0.09}}$	$98.68_{\pm 0.17}$	$98.42_{\pm 0.08}$	$\mathbf{94.84_{\pm 0.25}}$	$\mathbf{94.31_{\pm 0.21}}$	$93.16_{\pm 1.00}$	$92.97_{\pm 0.44}$



(a) Addition-KMNIST



(b) Addition-MNIST

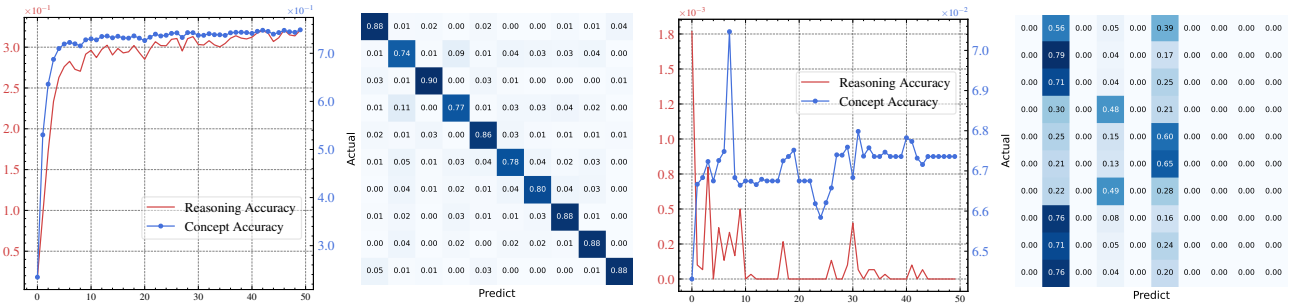
Figure 8. Performance curves (accuracy vs. epochs) of compared methods on digit addition tasks. The shaded area represents the standard error of the methods over 5 repetitions. Cross marks denote experiments that were not finished within 48 hours.

(b) Handwritten Formula Recognition The test accuracy results are presented in Table 6. Both DeepProbLog and variants of NGS underperform in this setting, while DeepStochLog performs well in HWF, it falls behind ABL-conf and A^{3}BL in the HWF-CIFAR and HWF-SVHN settings. It is important to note that the average size of the candidate set is relatively small, approximately 39, as shown in Figure 5. Therefore, ABL-conf demonstrates strong performance in this task, but it still converge slower than A^{3}BL.

Table 6. Test accuracies (mean $\pm$ standard deviation) on handwritten formula recognition tasks with different perception dataset. Each experiment was conducted five times. The best performance is denoted in boldface. 'N/A' indicates that the method failed to complete a single epoch within 48 hours.

Method	HWF-CIFAR	HWF-SVHN	HWF
NGS-BS (Li et al., 2020)	$28.25_{\pm 0.90}$	$26.91_{\pm 0.81}$	$31.65_{\pm 4.40}$
NGS-RL (Li et al., 2020)	$16.45_{\pm 1.62}$	$23.38_{\pm 3.31}$	$16.92_{\pm 0.30}$
NGS-MAPO (Li et al., 2020)	$20.99_{\pm 6.83}$	$24.23_{\pm 2.03}$	$16.89_{\pm 0.26}$
DeepStochLog (Winters et al., 2022)	$75.41_{\pm 2.21}$	$75.29_{\pm 23.81}$	$\mathbf{99.65_{\pm 0.06}}$
Deepproblog (Manhaeve et al., 2021)	N/A	N/A	N/A
ABL-hamming (Dai et al., 2019)	$41.73_{\pm 2.47}$	$37.18_{\pm 6.57}$	$99.40_{\pm 0.10}$
ABL-conf	$78.47_{\pm 0.69}$	$91.21_{\pm 0.34}$	$99.47_{\pm 0.17}$
A^{3}BL (Ours)	$\mathbf{78.92_{\pm 0.50}}$	$\mathbf{91.83_{\pm 0.55}}$	$99.41_{\pm 0.12}$

(c) Worst Case Analysis A worst-case analysis of ABL-hamming is depicted in Figure 9, focusing on the Addition-CIFAR task with a digit size of two. The learning curve of A^{3}BL demonstrates consistent improvement in both reasoning and conceptual accuracy over time. However, ABL-hamming does not exhibit this trend. ABL-hamming struggles, displaying significant instability in its training process, particularly in the reasoning accuracy curve. Furthermore, the confusion matrix of ABL-hamming suggests that the model's predictions are nearly equivalent to random guessing. This suggests that the ambiguity inherent in the abduction results poses challenges for the machine learning model.



(a) A^{3}BL **(b)** ABL-hamming

Figure 9. Worst case performance tendency comparison between A^{3}BL and ABL-hamming.

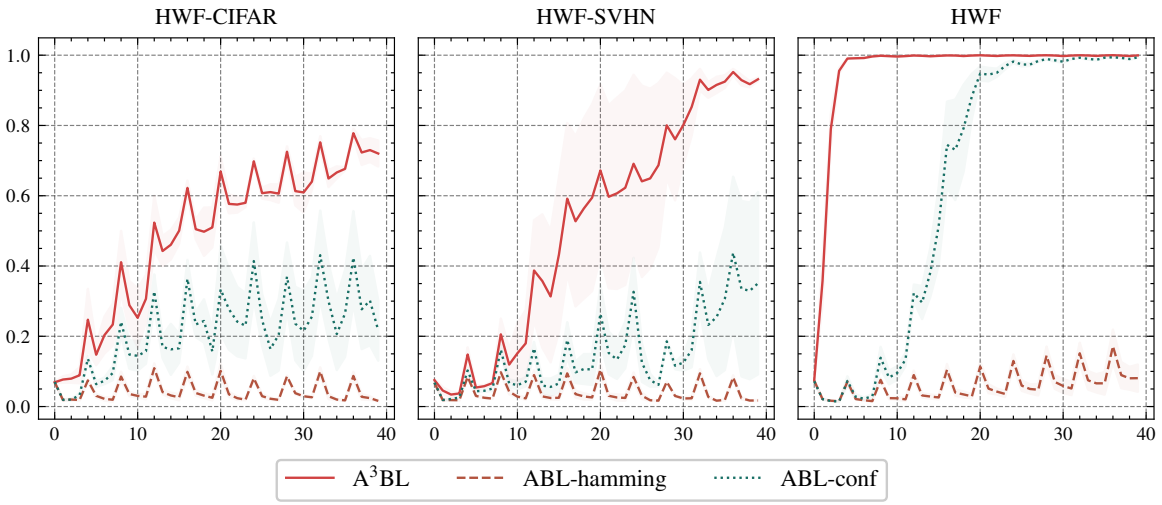


Figure 10. Abduction process (abduction accuracies vs. steps) of A^3BL and ABL on handwritten formula recognition tasks. The shaded area represents the standard error of the methods over 5 repetitions.

(d) Abduction Accuracies Figure 10 illustrates the abduction processes employed by A^3BL, ABL-hamming, and ABL-conf on handwritten formula recognition tasks. The process is conducted over 40 steps, corresponding to 10 epochs. ABL-hamming was unable to identify the correct candidate, similarly, ABL-conf faced challenges in this regard. However, A^3BL effectively distinguishes the correct candidate among abduction candidates through dynamic weight assignment to all potential candidates. Within 10 epochs, A^3BL attains high abduction accuracy, providing clear supervision to the machine learning model.