Journal Title
XX(X):2–85

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

# A Mathematical Framework and a Suite of Learning Techniques for Neural-Symbolic Systems

**Charles Dickens[1], Connor Pryor[1], Changyu Gao[2], Alon Albalak[3], Eriq Augustine[1], William Wang[3], Stephen Wright[2], and Lise Getoor[1]**

**Abstract**

The field of Neural-Symbolic (NeSy) systems is growing rapidly. Proposed approaches show great promise in achieving symbiotic unions of neural and symbolic methods. However, a unifying framework is needed to organize common NeSy modeling patterns and develop general learning approaches. In this paper, we introduce Neural-Symbolic Energy-Based Models (NeSy-EBMs), a unifying mathematical framework for discriminative and generative NeSy modeling. Importantly, NeSy-EBMs allow the derivation of general expressions for gradients of prominent learning losses, and we introduce a suite of four learning approaches that leverage methods from multiple domains, including bilevel and stochastic policy optimization. Finally, we ground the NeSy-EBM framework with Neural Probabilistic Soft Logic (NeuPSL), an open-source NeSy-EBM library designed for scalability and expressivity, facilitating the real-world application of NeSy systems. Through extensive empirical analysis across multiple datasets, we demonstrate the practical advantages of NeSy-EBMs in various tasks, including image classification, graph node labeling, autonomous vehicle situation awareness, and question answering.

**Keywords**

Neural-Symbolic AI, Energy-based Models, Deep Learning

*Prepared using* sagej.cls *[Version: 2017/01/17 v1.20]*

## 1 Introduction

The promise of mutually beneficial neural and symbolic integrations has motivated significant advancements in machine learning research. Much of the recent progress has been achieved in the neural-symbolic (NeSy) computing literature (d'Avila Garcez et al. 2002, 2009, 2019). NeSy is a large and rapidly growing community that has been hosting regular workshops since 2005 (NeSy2005) and began holding conferences in 2024 (NeSy2024). At a high level, NeSy research aims to build algorithms and architectures that combine neural and symbolic components (Xu et al. 2018; Yang et al. 2020; Cohen et al. 2020; Manhaeve et al. 2021a; Wang et al. 2019; Badreddine et al. 2022; Ahmed et al. 2022a; Pryor et al. 2023a). With the continued growth of the field, NeSy requires a solid theoretical foundation built on a unifying framework. Such a framework should support understanding and organizing the strengths and limitations of existing NeSy approaches, while guiding design decisions to better match the requirements of specific applications. Furthermore, it should enable the development of general-purpose and widely applicable NeSy inference and learning algorithms.

In this paper, we introduce *Neural-Symbolic Energy-Based Models* (NeSy-EBMs), a mathematical framework for NeSy. NeSy-EBMs are a family of Energy-Based Models (EBMs) (LeCun et al. 2006) defined by energy functions that are compositions of neural and symbolic components. The neural component consists of a collection of deep models, and its output is provided to the symbolic component, which measures the compatibility of variables using domain knowledge and constraints. This formulation serves as a foundation for characterizing NeSy modeling paradigms and for developing general-purpose inference and learning algorithms. Moreover, by grounding NeSy in the well-established EBM perspective, NeSy-EBMs connects NeSy to the broader machine learning literature.

The insights derived from the NeSy-EBM framework motivate the development of a comprehensive system that supports key modeling paradigms. To this end, we introduce Neural Probabilistic Soft Logic (NeuPSL), an open-source, expressive, and efficient library for constructing NeSy-EBMs. NeuPSL uses the principled and comprehensive semantics of Probabilistic Soft Logic (PSL) (Bach et al. 2017) to create a NeSy-EBM symbolic component. Then, the neural component can be built using any deep modeling library and seamlessly integrated with the PSL symbolic component. Further, to ensure differentiability properties and provide principled forms of gradients for learning, we

[1]University of California Santa Cruz
[2]University of Wisconsin Madison
[3]University of California Santa Barbara

**Corresponding author:**
Charles Dickens, Department of Computer Science and Engineering University of California, Santa Cruz, CA 95060, USA
Email: cadicken@ucsc.edu

*Prepared using* sagej.cls

present a new formulation and regularization of PSL inference as a constrained quadratic program.

Further, we develop a suite of principled neural and symbolic parameter learning techniques for NeSy. NeSy-EBM predictions are typically obtained by finding a state of variables with high compatibility (i.e., low energy). The high compatibility state is found by minimizing the energy function via an optimization algorithm, for instance, an interior point method for continuous variables (Nocedal and Wright 2006) or a branch-and-bound strategy for discrete problems (H. Papadimitriou and Steiglitz 1998). The complex prediction process makes finding a gradient or descent direction of a standard machine learning loss with respect to the parameters difficult. To formalize these challenges and propose solutions, we introduce a categorization of learning losses based on the complexity of the relation to the NeSy-EBM energy function. We derive general expressions for gradients of the categorized learning losses with respect to the neural and symbolic parameters when the loss is differentiable. Additionally, we introduce four NeSy-EBM learning algorithms: one for learning the neural and symbolic weights separately and three for end-to-end learning. Our end-to-end learning algorithms make use of ideas from the bilevel optimization and reinforcement learning literature. Moreover, we discuss the strengths and limitations of each algorithm and describe its applicability to various modeling paradigms.

We empirically investigate the utility of NeSy-EBM's for four use-cases: 1) constraint satisfaction and joint reasoning, 2) fine-tuning and adaptation, 3) few-shot and zero-shot reasoning, and 4) semi-supervised learning. We simultaneously analyze multiple NeSy-EBM modeling paradigms and learning algorithms in an extensive empirical analysis across numerous variations of seven datasets. We show compelling results for real-world applications, including graph node classification, computer vision object detection, and natural language question answering. Notably, NeSy-EBMs are shown to enhance neural network prediction accuracy, enforce constraints, and improve label and data efficiency in semi-supervised and low-data settings, respectively.

This paper integrates and expands on our prior work on NeSy integrations and applications via the NeSy-EBM framework (Pryor et al. 2023a; Dickens et al. 2024a,b). The strengths of NeSy-EBM models have been demonstrated on a variety of tasks, including dialog structure induction (Pryor et al. 2023b), natural language (Pan et al. 2023; Dickens et al. 2024b) and visual question answering (Yi et al. 2019), autonomous vehicle situation awareness (Giunchiglia et al. 2023), human activity recognition (Arrotta et al. 2024), recommendation (Carraro et al. 2022), and autonomous agent navigation and exploration (Zhou et al. 2023). Additionally, the NeSy-EBM framework has enabled a deeper understanding of the connections and capabilities of NeSy systems (Dickens et al. 2024b). Moreover, general NeSy inference and learning algorithms have been developed (Dickens et al. 2024a) along with new open-source NeSy implementations (Pryor et al. 2023a). The NeSy-EBM framework has become a powerful tool for formalizing connections and capabilities of NeSy models and for developing new NeSy architectures and learning algorithms. In this work, we organize and advance our prior work to 1) present NeSy-EBMs, a mathematical framework for NeSy, 2) introduce NeuPSL, an open-source tool for building NeSy-EBMs, 3) develop a suite of general NeSy learning

techniques, and 4) simultaneously demonstrate the value of NeSy and analyze our learning techniques with an extensive empirical analysis.

This paper is organized as follows. In Section 2 we discuss related work on NeSy frameworks and NeSy applications. In Section 3, we formally define NeSy-EBMs and introduce three practical NeSy modeling paradigms. Next, in Section 4, we introduce NeuPSL, a scalable and expressive NeSy-EBM implementation. In Section 5, we present a suite of NeSy learning techniques. Then, in Section 6, we use NeuPSL to build NeSy-EBMs for our empirical analysis of NeSy use cases, modeling paradigms, and learning algorithms. Finally, we discuss limitations, takeaways, and future work in Section 7 and Section 8.

## 2 Related Work

There is a long, rich history of research on the integration of symbolic knowledge and reasoning with neural networks, which has rapidly evolved in the past decade. In this work, we establish a unifying framework for achieving such integration by connecting two foundational areas of machine learning research: Neural-Symbolic (NeSy) AI and energy-based modeling (EBMs). The remainder of this section provides an overview of NeSy frameworks and applications. Additionally, we provide an extended related work in Appendix B, covering EBMs, and bilevel optimization.

### *Neural-Symbolic Frameworks*

NeSy empowers neural models with domain knowledge and reasoning through integrations with symbolic systems (d'Avila Garcez et al. 2002, 2009, 2019; De Raedt et al. 2020; Besold et al. 2022). Various taxonomies have been proposed to categorize NeSy literature. Bader and Hitzler (2005), d'Avila Garcez et al. (2019), and most recently Besold et al. (2022) provide extensive surveys using characteristics such as knowledge representation, neural-symbolic connection, and applications to compare and describe methods. Similarly, the works of De Raedt et al. (2020) and Lamb et al. (2020) propose taxonomies to connect NeSy to statistical relational learning and graph neural networks, respectively. Focused taxonomies are described by Giunchiglia et al. (2022) and van Krieken et al. (2022) for deep learning with constraints and symbolic knowledge representations and Dash et al. (2022) for integrating domain knowledge into deep neural networks. Marconato et al. (2023) characterizes the common reasoning mistakes made by NeSy models, and Marconato et al. (2024) presents an ensembling technique that calibrates the model's concept-level confidence to attempt to identify these mistakes. Recently, Wan et al. (2024) explored various NeSy AI approaches primarily focusing on workloads on hardware platforms, examining runtime characteristics and underlying compute operators. Finally, van Krieken et al. (2024) propose a language for NeSy called ULLER that aims to unify the representation of major NeSy systems, with the long-term goal of developing a shared Python library. Each of these surveys and taxonomies contributes to the comparison, understanding, and organization of the diverse collection of NeSy methodologies. We contribute to these efforts by introducing a common mathematical framework (Section 3) and describe a collection NeSy modeling paradigms (Section 3).

We organize our exposition of related NeSy AI frameworks into three research areas: learning from constraints, differentiable reasoning layers, and reasoner agnostic systems. The first subsection discusses NeSy learning losses. Whereas, the second and third subsections cover NeSy approaches to both learning and inference. In the following subsections, we define each of the research areas and describe prominent examples of NeSy models belonging to the area.

*Learning from Constraints* Learning from constraints is using domain knowledge and common sense to construct a learning loss function (Giunchiglia et al. 2022; van Krieken et al. 2022). This approach encodes the knowledge captured by the loss into the weights of the network. A key motivation is to ensure the compatibility of predictions with domain knowledge and common sense. Moreover, learning with constraints avoids potentially expensive post-prediction interventions that would be necessary with a model that is not aligned with domain knowledge. However, consistency with domain knowledge and sound reasoning are not guaranteed during inference for NeSy models in this class. This is because there is no symbolic reasoning performed to obtain predictions from the system.

Demeester et al. (2016), Rocktäschel and Riedel (2017), Diligenti et al. (2017b), Bošnjak et al. (2017), and Xu et al. (2018) are prominent examples of the learning-with-constraints NeSy paradigm. Demeester et al. (2016) incorporates domain knowledge and common sense into natural language and knowledge base representations by encouraging partial orderings over embeddings via a regularization of the learning loss. Similarly, Rocktäschel and Riedel (2017) leverage knowledge represented as a differentiable loss derived from logical rules to train a matrix factorization model for relation extraction. Diligenti et al. (2017b) use fuzzy logic to measure how much a model's output violates constraints, which is minimized during learning. Xu et al. (2018) introduces a loss function that represents domain knowledge and common sense by using probabilistic logic semantics. More recently, Giunchiglia et al. (2023) introduced an autonomous event detection dataset with logical requirements, and Stoian et al. (2023) shows that incorporating these logical requirements during the learning improves generalization.

*Differentiable Reasoning Layers* Another successful area of NeSy is in differentiable reasoning layers. The primary difference between this family of NeSy approaches and learning from constraints is that an explicit representation of knowledge and reasoning is maintained in the model architecture during both learning and inference. Moreover, a defining aspect of differentiable reasoning layers is the instantiation of knowledge and reasoning components as differentiable computation graphs. Differentiable reasoning layers support automatic differentiation during learning and symbolic reasoning during inference.

Pioneering works in differentiable reasoning include those of Wang et al. (2019), Cohen et al. (2020), Yang et al. (2020), Manhaeve et al. (2021a), Derkinderen et al. (2024), Badreddine et al. (2022), Ahmed et al. (2022a) and Ahmed et al. (2023a). Wang et al. (2019) integrates logical reasoning and deep models by introducing a differentiable smoothed approximation to a maximum satisfiability (MAXSAT) solver as a layer. Cohen et al. (2020) introduces a probabilistic first-order logic called TensorLog. This framework compiles tractable probabilistic logic programs into differentiable layers.

A TensorLog system is end-to-end differentiable and supports efficient parallelizable inference. Similarly, Yang et al. (2020) and Manhaeve et al. (2021a) compile tractable probabilistic logic programs into differentiable functions with their frameworks NeurASP and DeepProblog, respectively. NeurASP and DeepProblog use answer set programming (Brewka et al. 2011) and ProbLog (De Raedt et al. 2007) semantics, respectively. Winters et al. (2022) proposes DeepStochLog, a NeSy framework based on stochastic definite clause grammars that define a probability distribution over possible derivations. Recently, Maene and Raedt (2024) proposes DeepSoftLog, a superset of ProbLog, adding embedded terms that result in probabilistic rather than fuzzy semantics. The logic tensor network (LTN) framework proposed by Badreddine et al. (2022) uses neural network predictions to parameterize functions representing symbolic relations with real-valued or fuzzy logic semantics. The fuzzy logic functions are aggregated to define a satisfaction level. Predictions can be obtained by evaluating the truth value of all possible outputs and returning the highest-valued configuration. Badreddine et al. (2023) has expanded upon LTNs and presents a configuration of fuzzy operators for grounding formulas end-to-end in the logarithm space that is more effective than previous proposals. Recently, Ahmed et al. (2022a) introduced a method for compiling differentiable functions representing knowledge and logic using the semantics of probabilistic circuits (PCs) (Choi et al. 2020). Their approach, called semantic probabilistic layers (SPLs), performs exact inference over tractable probabilistic models to enforce constraints over the predictions and uses the PC framework to ensure that the NeSy model is end-to-end trainable.

As pointed out by Cohen et al. (2020), answering queries in many (probabilistic) logics is equivalent to the weighted model counting problem, which is #P-complete or worse. Similarly, the MAXSAT problem studied by Wang et al. (2019) is NP-hard. Thus, since deep neural networks can be evaluated in time polynomial in their size, no polysize network can implement general logic queries unless #P=P, or MAXSAT solving, unless NP=P. For this reason, researchers have made progress towards building more efficient differentiable reasoning systems by, for example, restricting the probabilistic logic to tractable families (Cohen et al. 2020; Ahmed et al. 2022a; Maene et al. 2024), or performing approximate inference (Wang et al. 2019; Manhaeve et al. 2021b; van Krieken et al. 2023).

*Reasoner Agnostic Systems* More recently, researchers have sought to build NeSy frameworks with more general reasoning and knowledge representation capacities with expressive mathematical program blocks for reasoning. Mathematical programs are capable of representing cyclic dependencies across variables and ensuring the satisfaction of prediction constraints during learning and inference. Moreover, the system's high-level inference and training algorithms are agnostic to the solver used for the mathematical program.

Prominent reasoner-agnostic systems include the works of Amos and Kolter (2017), Agrawal et al. (2019a), Vlastelica et al. (2020), and Cornelio et al. (2023). Amos and Kolter (2017) integrate linearly constrained quadratic programming problems (LCQP) as layers in deep neural networks with their OptNet framework, and show that the solutions to the LCQP problems are differentiable with respect to the program parameters.

The progress of OptNet was continued by the work of Agrawal et al. (2019a) with the application of domain-specific languages (DSLs) for instantiating the LCQP program layers. DSLs provide a syntax for specifying LCQPs representing knowledge and constraints, making optimization layers more accessible. Vlastelica et al. (2020) propose a method for computing gradients of solutions to mixed integer linear programs based on a continuous interpolation of the program's objective. In contrast to the works of Amos and Kolter (2017) and Agrawal et al. (2019a), the approach introduced by Vlastelica et al. (2020) supports integer constraints and achieves this by approximating the true gradient of the program output. Cornelio et al. (2023) takes a different approach from these three methods by employing reinforcement learning techniques to support more general mathematical programs. Specifically, the neural model's predictions are interpreted as a state in a Markov decision process. Actions from a policy are taken to identify components that violate constraints to obtain a new state. The new state is provided to a solver, which corrects the violations, and a reward is computed. The solver is not assumed to be differentiable, and the REINFORCE algorithm (Williams 1992) with a standard policy loss is used to train the system end-to-end without the need to backpropagate through the solver.

## *Applications*

We highlight five proven applications NeSy: 1) constraint satisfaction and joint reasoning, 2) post-training, 3) few-shot and zero-shot reasoning, 4) semi-supervised learning, and 5) reasoning with noisy data. This list of use cases is not exhaustive. However, the efficacy of the NeSy approach in these applications is well established, and we will illustrate four of these use cases in our empirical evaluation. The following subsections define the problem and the high-level motivation for utilizing NeSy techniques in such settings. Additionally, we discuss collections of existing NeSy systems for each application.

*Constraint Satisfaction and Joint Reasoning* In real-world settings, a deployed model's predictions must meet well-defined requirements. Additionally, leveraging known patterns or dependencies in the output can significantly improve a model's accuracy and trustworthiness. *Constraint satisfaction* is finding a prediction that satisfies all requirements. NeSy systems perform constraint satisfaction by reasoning across their output to provide a structured prediction, typically using some form of joint reasoning. In other words, NeSy systems integrate constraints and knowledge into the prediction process.

A commonly used example of constraint satisfaction and joint reasoning with NeSy techniques is puzzle-solving. Many NeSy frameworks are introduced with an evaluation on visual Sudoku and its variants (Wang et al. 2019; Augustine et al. 2022). In the visual Sudoku problem, puzzles are constructed with handwritten digits, and a model must classify the digits and infer numbers to fill in the empty cells using the rules of Sudoku. Empirical evaluations of NeSy systems that perform constraint satisfaction and joint reasoning on visual Sudoku problems can be found in Wang et al. (2019), Augustine et al. (2022), Pryor et al. (2023a), and Morra et al. (2023). Similarly, Vlastelica et al. (2020) introduces the shortest path finding problem as a NeSy task. Images of terrain maps are

partitioned into a grid, and the model must find a continuous lowest-cost path between two points. The works of Vlastelica et al. (2020) and Ahmed et al. (2022a) perform constraint satisfaction and joint reasoning with NeSy models for shortest path finding.

Constraint satisfaction and joint reasoning with NeSy models are also effective for real-world natural language tasks. For instance, Sachan et al. (2018) introduces the Nuts&Bolts NeSy system to build a pipeline for parsing physics problems. The NeSy system jointly infers a parsing from multiple components that incorporates domain knowledge and prevents the accumulation of errors that would occur from a naive composition. In another work, Zhang et al. (2023) propose GeLaTo (generating language with tractable constraints) for imposing constraints on text generated from language models. GeLaTo generates text tokens by autoregressively sampling from a distribution constructed from a pre-trained language model and a tractable probabilistic model encoding the constraints. More recently, Pan et al. (2023) introduced the Logic-LM framework for integrating LLMs with symbolic solvers to improve complex problem-solving. Logic-LM formulates a symbolic model using an LLM that uses prompts of the syntax and semantics of the symbolic language. Finally, Abraham et al. (2024) introduced CLEVR-POC, which requires leveraging logical constraints to generate plausible answers to questions about a hidden object in a given partial scene. They then demonstrated remarkable performance improvements over neural methods by integrating an LLM with a visual perception network and a formal logical reasoner.

Computer vision systems also benefit from the constraint satisfaction and joint reasoning capabilities of NeSy models. For instance, semantic image interpretation (SII) is the task of extracting structured descriptions from images. Donadello et al. (2017) implemented a NeSy model for SII using the Logic Tensor Network (LTN) (Badreddine et al. 2022) framework for reasoning about "part-of" relations between objects with logical formulas. Similarly, Yi et al. (2019) propose a NeSy visual question-answering framework (NS-VQA). The authors employ deep representation learning for visual recognition to recover a structured representation of a scene and then language understanding to formulate a program from a question. A symbolic solver executes the formulated program to obtain an answer. Sikka et al. (2020) introduced Deep Adaptive Semantic Logic (DASL) for predicting relationships between pairs of objects in an image given the bounding boxes and object category labels, i.e., visual relationship detection. The DASL system allows a modeler to express knowledge using first-order logic and to combine domain-specific neural components into a single deep network. A DASL model is trained to maximize a measured truth value of the knowledge.

*Post-training* We are in the era of foundation models in AI (Bommasani et al. 2022). It is now commonplace to adjust a model that is pre-trained on large amounts of general data (typically using self-supervision) for downstream tasks. Post-training is the process of updating the parameters of a pre-trained model to perform in a new domain (Devlin et al. 2019; J. Hu et al. 2022). Fine-tuning and alignment are two example post-training techniques that adjust the pre-trained model parameters by minimizing a learning objective over a dataset, both of which are specialized for the downstream tasks. These are necessary steps in the modern AI development process.

NeSy frameworks are used in post-training to design principled learning objectives that integrate knowledge and constraints relevant to the downstream task and the application domain. Giunchiglia et al. (2022) provides a recent survey of the use of logically specified background knowledge to train neural models. NeSy learning losses are applied in the work of Giunchiglia et al. (2023) to post-train a neural system for autonomous vehicle situation awareness (Singh et al. 2021). In another computer vision task, Arrotta et al. (2024) develop a NeSy loss for training a neural model to perform context-aware human activity recognition. NeSy post-training has also been explored in the natural language processing literature. Recently, Ahmed et al. (2023b) proposed the pseudo-semantic loss for detoxifying large language models. The authors disallow a list of toxic words and show this intuitive approach steers a language model's generation away from harmful language and achieves state-of-the-art detoxification scores. Feng et al. (2024) has explored directly learning the reasoning process of logical solvers within the LLM to avoid parsing errors. Finally, Cunnington et al. (2024) introduced NeSyGPT, which post-trains a vision-language foundation model to extract symbolic features from raw data before learning some answer set program.

*Few-Shot and Zero-Shot Reasoning* Training data for a downstream task may be limited or even nonexistent. In *few-shot* settings, only a few examples are available, while in *zero-shot* settings, no explicit training data is provided for the task. In these settings, few-shot and zero-shot reasoning techniques are used to enable a model to generalize beyond the limited available training data. Leveraging pre-trained models and domain knowledge are key ideas for succeeding in few-shot and zero-shot contexts.

NeSy techniques have been successfully applied for various few-shot and zero-shot settings. Integrating symbolic knowledge and reasoning enables better generalization from a small number of examples. NeSy systems can utilize symbolic knowledge to make deductions about unseen classes or tasks. For instance, providing recommendations for new items or users can be viewed as a few-shot or zero-shot problem. Kouki et al. (2015) introduce the HyPER (hybrid probabilistic extensible recommender) framework for incorporating and reasoning over a wide range of information sources. By combining multiple information sources via logical relations, the authors outperformed the state-of-the-art approaches of the time. More recently, Carraro et al. (2022) developed an LTN-based recommender system to overcome data sparsity. This model uses background knowledge to generalize predictions for new items and users quickly. Few-shot and zero-shot reasoning tasks are also prevalent in object navigation. The ability to navigate to novel objects and unfamiliar environments is vital for the practical use of embodied agents in the real world. In this context, Zhou et al. (2023) presents a method for "exploration with soft commonsense constraints" (ESC). ESC first employs a pre-trained vision and language model for semantic scene understanding, then a language model to reason from the spatial relations, and finally PSL to leverage symbolic knowledge and reasoning to guide exploration. In natural language processing, Pryor et al. (2023b) infers the latent dialog structure of a goal-oriented conversation using domain knowledge to overcome the challenges of limited data and out-of-domain generalization. Sikka et al. (2020) (mentioned above) also finds that the few-shot and zero-shot capabilities of NeSy models

help in visual relationship detection. Specifically, the addition of commonsense reasoning and knowledge improves performance by over $10\%$ in data-scarce settings.

*Semi-Supervised Learning* Semi-supervised approaches facilitate learning from labeled as well as unlabeled data by combining the goals of supervised and unsupervised machine learning. We refer the reader to the excellent recent survey on semi-supervised approaches by E. van Engelen and H. Hoos (2020). In short, supervised methods fit a model to predict an output label given a corresponding input, while unsupervised methods infer the underlying structure in the data. The ability to leverage both labeled and unlabeled data leads to performance improvements, better generalization, and reduced labeling costs.

NeSy is a functional approach to semi-supervised learning that leverages knowledge and domain constraints to train a model. This is achieved with loss functions that encode domain knowledge and structure and depend only on the input and output; that is, they do not require a label. Early work on semi-supervision with knowledge was carried out by Chang et al. (2007), who unify and leverage task-specific constraints to encode structure in the input and output data and possible labels. They evaluate their semi-supervised learning method on the task of named entity recognition in citations as well as advertisements. More recently, Ahmed et al. (2022b) introduced the neuro-symbolic entropy regularization loss to encourage model confidence in predictions satisfying a set of constraints on the output. They demonstrate that the regularization improves model performances in the task of entity relation extraction in text. Additionally, Stoian et al. (2023) studied the effect of various t-norms used to soften the logical constraints for the symbolic component and demonstrated on a challenging road event detection dataset with logical requirements (Giunchiglia et al. 2023) that the incorporation of a symbolic loss drastically improves performance.

## 3 A Mathematical Framework for NeSy

In this section, we introduce Neural-symbolic energy-based models (NeSy-EBMs): a unifying mathematical framework for NeSy. Intuitively, NeSy-EBMs formalize the neural-symbolic interface as a composition of functions. In other words, NeSy-EBMs organize modules by roles, specifically perception and reasoning. The theory and notation introduced in this section are used throughout the rest of this paper.

### *Neural Symbolic Energy-Based Models*

NeSy-EBMs are a family of EBMs (LeCun et al. 2006) that integrate deep architectures with explicit encodings of symbolic relations via an energy function. EBM energy functions measure the compatibility of variables, where low energy states correspond to high compatibility. For NeSy-EBMs, high compatibility indicates that the variables are consistent with domain knowledge and common sense. In the following section, the formal NeSy-EBM definition is grounded with intuitive examples of NeSy modeling paradigms.

As diagrammed in Fig. 1, a NeSy-EBM energy function composes a neural component with a symbolic component, represented by the functions $\mathbf{g}_{nn}$ and $\mathbf{g}_{sy}$, respectively. The

neural component is a deep model (or collection of deep models) parameterized by weights from a domain $\mathcal{W}_{nn}$, that takes a neural input from a domain $\mathcal{X}_{nn}$ and outputs a real-valued vector of dimension $d_{nn}$. The symbolic component encodes domain knowledge and is parameterized by weights from a domain $\mathcal{W}_{sy}$. It maps the inputs of a domain $\mathcal{X}_{sy}$, target (or output) variables from $\mathcal{Y}$, and neural outputs from $\text{Range}(\mathbf{g}_{nn})$ to a scalar value. In other words, the symbolic component measures the compatibility of targets, inputs, and neural outputs with domain knowledge. Intuitively, the neural component has the capacity and responsibility to perform low-level perception or generation, while the symbolic component has the role of performing high-level symbolic reasoning.

**Definition 1**
*A **NeSy-EBM energy function** is a mapping parameterized by neural and symbolic weights from domains $\mathcal{W}_{nn}$ and $\mathcal{W}_{sy}$, respectively, and quantifies the compatibility of a target variable from a domain $\mathcal{Y}$ and neural and symbolic inputs from the domains $\mathcal{X}_{nn}$ and $\mathcal{X}_{sy}$, respectively, with a scalar value:*

$$E : \mathcal{Y} \times \mathcal{X}_{sy} \times \mathcal{X}_{nn} \times \mathcal{W}_{sy} \times \mathcal{W}_{nn} \to \mathbb{R}. \tag{1}$$

*A NeSy-EBM energy function is a composition of a **neural** and **symbolic component**. Neural weights parameterize the neural component, which outputs a real-valued vector of dimension $d_{nn}$:*

$$\mathbf{g}_{nn} : \mathcal{X}_{nn} \times \mathcal{W}_{nn} \to \mathbb{R}^{d_{nn}}. \tag{2}$$

*The symbolic component maps the symbolic variables, symbolic parameters, and a real-valued vector of dimension $d_{nn}$ to a scalar value:*

$$g_{sy} : \mathcal{Y} \times \mathcal{X}_{sy} \times \mathcal{W}_{sy} \times \mathbb{R}^{d_{nn}} \to \mathbb{R}. \tag{3}$$

*The NeSy-EBM energy function is*

$$E : (\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \mapsto g_{sy}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})). \quad \square$$

Given inputs and parameters $(\mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \in \mathcal{X}_{sy} \times \mathcal{X}_{nn} \times \mathcal{W}_{sy} \times \mathcal{W}_{nn}$, NeSy-EBM energy functions can be used to define several inference tasks, for instance:

- *Prediction, classification, and decision making*: Find targets minimizing the energy function.

$$\arg\min_{\hat{\mathbf{y}} \in \mathcal{Y}} E(\hat{\mathbf{y}}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}). \tag{4}$$

- *Ranking*: Sort a set of targets in order of increasing energy.

$$E(\mathbf{y}^{\mathbf{r}_1}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \leq \cdots \leq E(\mathbf{y}^{\mathbf{r}_p}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \tag{5}$$

- *Detection*: Determine if a target, $\mathbf{y}$, is below a threshold $\tau$.

$$D(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \tau) := \begin{cases} 1 & E(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \leq \tau \\ 0 & o.w. \end{cases} \tag{6}$$

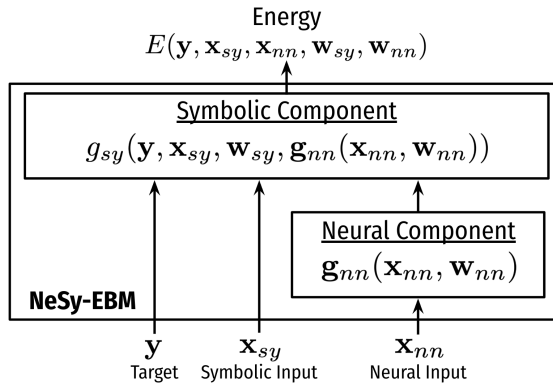



**Figure 1.** A neural-symbolic energy-based model.

- *Density estimation*: Estimate the conditional probability of a target, $\mathbf{y}$. The energy function is used to define a probability density, such as a Gibbs distribution.

$$P(\mathbf{y}|\mathbf{x}_{sy}\mathbf{x}_{nn}; \mathbf{w}_{sy}, \mathbf{w}_{nn}) := \frac{e^{-\beta E(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})}}{\int_{\hat{\mathbf{y}} \in \mathcal{Y}} e^{-\beta E(\hat{\mathbf{y}}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})}}, \tag{7}$$

  where $\beta$ is the positive inverse temperature parameter.

- *Generation*: Sample a target variable state using a distribution defined by the energy function.

$$\mathbf{y} \sim P(\mathbf{y}|\mathbf{x}_{sy}\mathbf{x}_{nn}; \mathbf{w}_{sy}, \mathbf{w}_{nn}). \tag{8}$$

In this paper, we focus on the first and most common task in this list: prediction, classification, and decision-making (4). Prediction with NeSy-EBMs captures various forms of reasoning, including probabilistic, logical, arithmetic, and their combinations. It can represent standard applications of prominent NeSy systems, including, DeepProbLog (Manhaeve et al. 2021a), LTNs (Badreddine et al. 2022), Semantic Probabilistic Layers (Ahmed et al. 2022a), and NeuPSL (Pryor et al. 2023a), to name a few.

## *Modeling Paradigms for NeSy*

Using the NeSy-EBM framework, this subsection introduces three typical NeSy modeling paradigms determined by the nature of the neural-symbolic interface. The paradigms are characterized by the integration of the neural component within the symbolic component to define the prediction program in Equation 4.

To formalize the modeling paradigms, we introduce an additional layer of abstraction we refer to as *symbolic potentials*, denoted by $\psi$. Further, we collect symbolic potentials into *symbolic potential sets*, denoted by $\mathbf{\Psi}$. Symbolic potentials organize the arguments of the symbolic component by the role they play in formulating the prediction program in (4).

**Definition 2**
*A **symbolic potential** $\psi$ is a function of variables from a domain $V_\psi$ and parameters from a domain $Params_\psi$, outputting a scalar value:*

$$\psi : V_\psi \times Params_\psi \to \mathbb{R}. \tag{9}$$

*A **symbolic potential set**, denoted by $\mathbf{\Psi}$, is a set of potential functions indexed by $\mathbf{J}_\mathbf{\Psi}$.* □

With this formalization, a modeling paradigm is defined by specifying a set of symbolic potentials along with their respective domains. We introduce three key modeling paradigms in the following subsections: deep symbolic variables (DSVar), deep symbolic parameters (DSPar), and deep symbolic potentials (DSPot). In the following section, we present a novel NeSy framework that supports all of the outlined modeling paradigms. Additionally, Appendix C formalizes three widely used NeSy approaches—DeepProbLog (Manhaeve et al. 2021a), Logic Tensor Networks (Badreddine et al. 2022), and Semantic Loss (Xu et al. 2018)—within these modeling paradigms. While these modeling paradigms capture the fundamental characteristics of many NeSy systems, some approaches may not fit neatly into these categories.

*Deep Symbolic Variables* The deep symbolic variables (DSVar) paradigm trains neural components efficiently with a loss that captures domain knowledge. Concisely, the neural component directly predicts the values of target or latent variables in a symbolic potential.* In other words, there is a one-to-one mapping from the neural output to the targets. However, note that the mapping is not necessarily *onto*, that is, there may be target or latent variables without a corresponding neural output. Prominent NeSy approaches exemplifying this paradigm include logic tensor networks Badreddine et al. (2022), learning with logical constraints Giunchiglia et al. (2022), semantic-based regularization Diligenti et al. (2017a), and deep logic models Marra et al. (2019).

**Definition 3**
*In the **deep symbolic variables** (DSVar) modeling paradigm the symbolic potential set is a singleton $\mathbf{\Psi} = \{\psi\}$ with a trivial index set $\mathbf{J}_\mathbf{\Psi} = \{1\}$ such that $\mathbf{\Psi}_1 = \psi$. Further, the neural prediction is treated as a variable by the symbolic potential; thus $V_\psi = \mathcal{Y} \times \mathcal{X}_{sy} \times \mathbb{R}^{d_{nn}}$. Then, the symbolic parameters are the symbolic weights, $Params_\psi = \mathcal{W}_{sy}$. The neural component controls the NeSy-EBM prediction via this function:*

$$I_\mathcal{Y}(\mathbf{y}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) := \begin{cases} 0 & \mathbf{y}_i = [\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})]_i \,, \forall i \in \{1, \cdots, d_{nn}\} \\ \infty & o.w. \end{cases}, \tag{10}$$

*where $\mathbf{y}_i$ and $\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})_i$ denote the $i$'th entry of the variable and neural output vectors, respectively. Then, the symbolic component expressed via the symbolic potential*

*This section focuses on deep symbolic variables in the context of target variables. Extending to latent variables is straightforward.

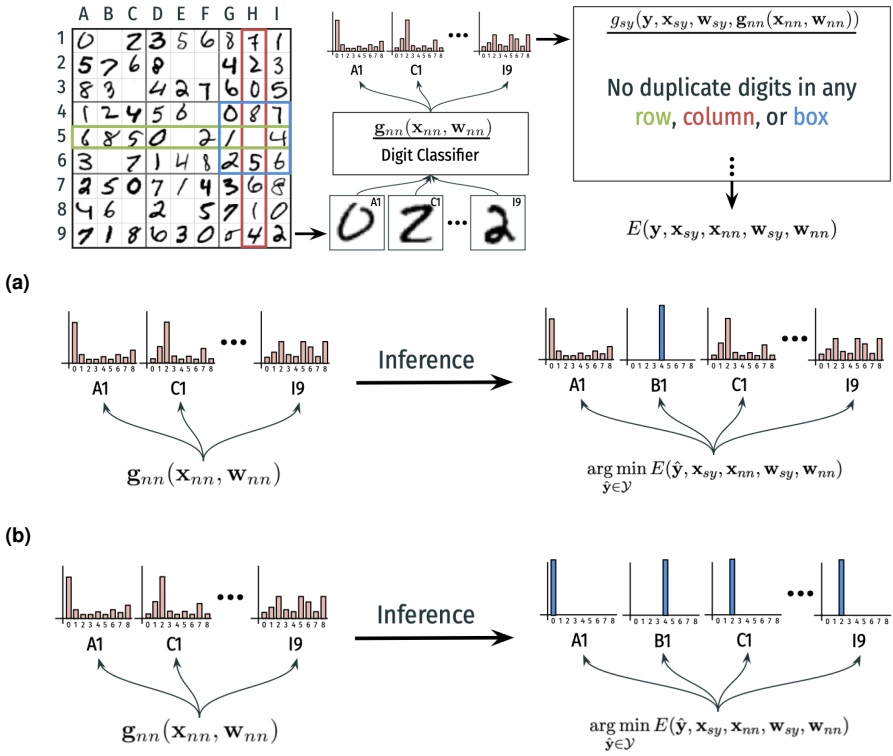



**Figure 2.** (a) A NeSy-EBM for solving a Sudoku board constructed from handwritten digits. The neural component classifies handwritten digits. Then, the symbolic component uses the digit classifications and Sudoku rules to quantify the compatibility of the inputs, neural predictions, and targets. (b) In the DSVar modeling paradigm inference process, the neural component predicts squares with digits, while the symbolic component measures incompatibility and predicts the latent (blank) squares. (c) In the DSPar modeling paradigm inference process, the neural component predicts squares with digits, and the symbolic component can alter these predictions to adhere to symbolic constraints.

*is:*

$$
\begin{aligned}
&g_{sy}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \\
&:= \psi([\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})], \mathbf{w}_{sy}) + I_{\mathcal{Y}}(\mathbf{y}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})),
\end{aligned} \tag{11}
$$

*where* $[\cdot]$ *denotes concatenation.* □

The DSVar modeling paradigm typically yields the most straightforward prediction program compared to the other modeling paradigms. This is because the neural model fixes a subset of the decision variables, making the prediction program smaller. This is achieved

by adding the function (Equation 10) in the definition above to the symbolic potential, so that infinite energy is assigned to variable values that do not match the predictions of the neural model. However, for the same reason that this modeling paradigm typically has a simpler prediction program, the symbolic component cannot be used to resolve constraint violations made by the neural component. Rather, DSVar models rely on learning to train a neural component to adhere to constraints. The DSVar paradigm is demonstrated in the following example.

**Example 1**
*Visual Sudoku (Wang et al. 2019) puzzle solving is the problem of recognizing handwritten digits in non-empty puzzle cells and reasoning with the rules of Sudoku (no repeated digits in any row, column, or box) to fill in empty cells. Fig. 2 shows a partially complete Sudoku puzzle created with MNIST images (LeCun et al. 1998) and a NeSy-EBM designed for visual Sudoku solving. The neural component is a digit classifier predicting the label of MNIST images, and the symbolic component quantifies rule violations.*

*Formally, the target variables, $\mathbf{y}$, are the categorical labels of both the handwritten digits and the empty entries in the puzzle, i.e., the latent variables. The symbolic inputs, $\mathbf{x}_{sy}$, indicate whether two puzzle positions are in the same row, column, or box. The neural model, $\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})$, is the categorical label of the handwritten digits predicted by the neural component. Then, the symbolic parameters, $\mathbf{w}_{sy}$, are used to shape the single symbolic potential function, $\psi$, that quantifies the amount of Sudoku rule violations.*

The DSVar modeling paradigm is specifically designed to allow the neural component to directly influence the random variables within the symbolic model. Although this paradigm allows direct influence on the predictions of a symbolic model, its scope is strictly confined to random variables. In scenarios where the neural model must exert indirect influence on variables or interact with other elements of the symbolic model, such as entire symbolic potentials or parameters associated with individual constraints, a different modeling paradigm becomes necessary. The following subsection introduces a paradigm that extends the neural component's influence, enabling connections to other parameters or constants within the symbolic model, beyond just the random variables.

*Deep Symbolic Parameters* The deep symbolic parameters (DSPar) modeling paradigm allows targets and neural predictions to be unequal or represent different concepts. Prominent NeSy frameworks supporting this technique include DeepProbLog (Manhaeve et al. 2021a), and semantic probabilistic layers (Ahmed et al. 2022a). Succinctly, the neural component is applied as a parameter in the symbolic potential. This paradigm allows the symbolic component to correct constraints violated by the neural component during prediction.

**Definition 4**
*In the **deep symbolic parameters** (DSPar) modeling paradigm, the symbolic potential set is a singleton $\mathbf{\Psi} = \{\psi\}$ with a trivial index set $\mathbf{J}_{\mathbf{\Psi}} = \{1\}$ such that $\mathbf{\Psi}_1 = \psi$. Further, the neural prediction is treated as a parameter by the symbolic potential, thus $Params_\psi = \mathcal{W}_{sy} \times \mathbb{R}^{d_{nn}}$. Then the symbolic variables are the targets and the symbolic inputs: $V_\psi = \mathcal{Y} \times \mathcal{X}_{sy}$. The symbolic component expressed via the single symbolic*

*potential is:*

$$g_{sy}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) := \psi([\mathbf{y}, \mathbf{x}_{sy}], [\mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})]). \quad \square$$

This paradigm is demonstrated in the following example.

**Example 2**
*Again, consider the Visual Sudoku puzzle-solving problem illustrated in Fig. 2. As in the DSVar model, the neural component of the DSPar model is a digit classifier predicting the label of MNIST images. However, the digit classifications of the neural component are used as initial predictions in the symbolic component, as a prior for a probabilistic model. Then, the symbolic component is used to quantify rule violations as well as the difference between neural outputs and target variables.*

*The target variables, $\mathbf{y}$, are the categorical labels of both the handwritten digits and the puzzle's empty entries. The symbolic inputs, $\mathbf{x}_{sy}$, indicate whether two puzzle positions are in the same row, column, or box. The neural model, $\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})$ consists of the categorical labels of the handwritten digits predicted by the neural component. The symbolic parameters $\mathbf{w}_{sy}$ are used to shape the single symbolic potential function $\psi$ that quantifies the amount of Sudoku rule violations.*

The DSPar modeling paradigm is widely applicable. For instance, the DSPar modeling paradigm is applied for constraint satisfaction, fine-tuning, few-shot, and semi-supervised settings in our empirical analysis. However, note that the DSVar and DSPar models have only a single fixed symbolic potential. This property makes these paradigms well-suited for dedicated tasks but less applicable to open-ended settings, where the relevant domain knowledge depends on context. To address this challenge, the following modeling paradigm leverages generative modeling to perform in open-ended tasks.

*Deep Symbolic Potentials* Deep-symbolic potentials (DSPot), the most advanced paradigm we propose, enhances deep models with symbolic reasoning tools. The Logic-LM pipeline proposed by Pan et al. (2023) is an excellent example of this modeling paradigm. At a high level, the neural component is a generative model that samples symbolic potentials from a set to define the symbolic component. Specifically, input data is used as context to retrieve relevant domain knowledge and formulate a program to perform inference in open-ended problems.

**Definition 5**
*In the **deep symbolic potentials** modeling paradigm, the symbolic potential set $\mathbf{\Psi}$ is the set of all potential functions that can be created by a NeSy framework. $\mathbf{\Psi}$ is indexed by the output of the neural component, i.e., $\mathbf{J}_\Psi = Range(\mathbf{g}_{nn})$ and $\mathbf{\Psi}_{\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})}$ is the potential function indexed by the neural prediction. The variable and parameter domains of the sampled symbolic potential are $V_\psi = \mathcal{Y} \times \mathcal{X}_{sy}$, and $Params_\psi = \mathcal{W}_{sy}$, respectively. The symbolic component expressed via the symbolic potential is:*

$$g_{sy}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) := \mathbf{\Psi}_{\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})}([\mathbf{y}, \mathbf{x}_{sy}], \mathbf{w}_{sy}). \quad \square$$

This paradigm is demonstrated in the following example.

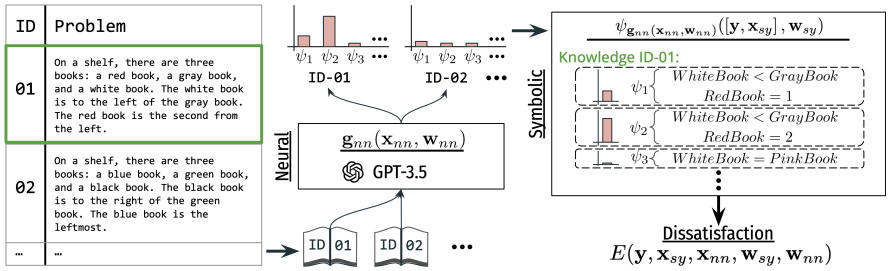



**Figure 3.** A deep symbolic potential model for answering questions about a set of objects' order described in natural language. The neural component is an LLM that generates syntax to create a symbolic potential. The symbolic potential is used to perform deductive reasoning and answer the question. See Example 3 for details.

**Example 3**

*Question answering is the problem of giving a response to a question posed in natural language. Fig. 3 shows a set of word problems asking for the order of a set of objects given information expressed in natural language and a NeSy-EBM designed for question answering. The neural component is a large language model (LLM) that is prompted with a word problem and tasked with generating a program within the syntax of a symbolic framework. The symbolic framework uses the generated program to instantiate a symbolic component used to perform deductive reasoning.*

*Formally, the target variables, $\mathbf{y}$, represent object positions, and there is no symbolic input, $\mathbf{x}_{sy}$, in this example. The neural input, $\mathbf{x}_{nn}$, is a natural language prompt that includes the word problem. The neural model, $\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})$, is an LLM that generates syntax for a declarative symbolic modeling framework that creates the symbolic potential. For instance, the symbolic potential generated by the neural model $\boldsymbol{\Psi}_{\mathbf{g}_{nn}(\mathbf{x}_{nn},\mathbf{w}_{nn})}([\mathbf{y}, \mathbf{x}_{sy}], \mathbf{w}_{sy})$ could be the total amount of violation of arithmetic constraints representing ordering. Finally, the symbolic parameters, $\mathbf{w}_{sy}$, shape the symbolic potential function.*

In our view, DSPot is the only applicable paradigm for truly open-ended tasks. Moreover, DSPot enhances generative models, such as LLMs, with consistent symbolic reasoning capabilities. This feature is demonstrated in constraint satisfaction and joint reasoning experiments in our empirical analysis. DSPot's limitation is that the neural component must learn to sample from a large potential set. For instance, in the example, an LLM must reliably generate syntax to define a symbolic potential for solving the word problem. LLMs require a substantial amount of computational resources to train and then fine-tune for a specific NeSy framework. Furthermore, the inference time is dependent on the sampled symbolic potential. If the neural component samples a complex symbolic potential, inference may be slow.

## 4 Neural Probabilistic Soft Logic and Deep Hinge-Loss Markov Random Fields

We introduce *Neural Probabilistic Soft Logic* (NeuPSL), an expressive framework for constructing a broad class of NeSy-EBMs by extending the probabilistic soft logic (PSL) probabilistic programming language (Bach et al. 2017). NeuPSL is designed to be expressive and efficient to support every modeling paradigm and easily be used for a range of applications. We begin by presenting the essential syntax and semantics of NeuPSL, encompassing Deep Hinge-Loss Markov Random Fields (deep HL-MRF), the underlying probabilistic graphical model (see Bach et al. (2017) for an in-depth introduction to PSL syntax and semantics). Then, we present a new formulation and regularization of (Neu)PSL inference as a constrained quadratic program. Our formulation is utilized to guarantee differentiability properties and provide principled gradients to support end-to-end neural and symbolic parameter learning, instantiating the learning algorithms introduced in Section 5.

### *Neural Probabilistic Soft Logic*

NeuPSL is a declarative language used to construct NeSy-EBMs. Fundamentally, NeuPSL provides a syntax for encoding dependencies between relations and attributes of entities and for integrating neural components in a symbolic model. Specifically, dependencies and neural component compositions are expressed as first-order logical or arithmetic statements referred to as *rules*. Each rule is a template for instantiating, i.e., *grounding*, potentials or constraints to define the NeuPSL energy function. Every rule is grounded over a set of domains, $\mathbf{D} = \{D_1, D_2, \cdots\}$, where each of the domains $D_i$ is a finite set of elements referred to as *constants*. For instance, referring to the visual Sudoku problem described in Example 2, the constant "A1" can denote the cell at position $A1$ in a Sudoku puzzle, and the constant "1" can denote the digit 1. Constants are grouped and aligned with a corresponding domain from $\mathbf{D}$ using placeholders or *variables*. Relations between constants are *predicates*. In NeuPSL, a predicate is referenced using its unique identifier. For instance, CELLDIGIT is a predicate that can represent whether a cell contains a specified digit. Another example is the predicate SUDOKUVIOLATION representing whether a Sudoku rule is violated given the digits in two specified cells. Finally, the predicate NEURALCLASSIFIER is a predicate that represents the predicted digit in a cell made by a neural network classifier. Predicates with specified constant domains are *atoms*. NeuPSL extends PSL with *deep atoms*: atoms backed by a deep model.

**Definition 6**
*Atom. An* ***atom****, $A$, is a predicate associated with a list of $k > 0$ domains $D'_1, \cdots, D'_k$ from $\mathbf{D}$:*

$$A : (D'_1 \times \cdots \times D'_k) \rightarrow [0, 1]$$

*where $k$ is the corresponding predicate's arity.*

*A **deep atom**, $DA$, with domains $D'_1, \cdots, D'_k$ from $\mathbf{D}$ is an atom parameterized by a set of weights $\mathbf{w}_{nn}$ from a domain $\mathcal{W}_{nn}$*

$$DA : (D'_1 \times \cdots \times D'_k; \mathbf{w}_{nn}) \to [0, 1].$$

*A **ground atom** is an atom with constant arguments.* □

With the above definition, we can now formally define a NeuPSL rule.

**Definition 7**
*Rule. A **rule**, $R$, is a function of $s \geq 1$ variables $v_1, \cdots, v_s$ from the domains $D'_1, \cdots, D'_s \in \mathbf{D}$, respectively:*

$$R : (D'_1 \times \cdots \times D'_s) \to [0, 1]$$
$$v_1, \cdots, v_s \mapsto R(v_1, \cdots, v_s)$$

*Moreover, a rule is a composition of $l \geq 1$ atoms, $A_1, \cdots A_l$.*

*All rules are either associated with a non-negative weight and a value $q \in \{1, 2\}$, or are unweighted. The weight (or absence of) and value $q$ of a rule determine the structure of the potentials the rule instantiates. A weighted rule is known as a **soft rule**, and an unweighted rule is known as a **hard rule**.*

*A **logical rule** is expressed as a logical implication of atoms.*

*An **arithmetic rule** is expressed as a linear inequality of atoms.*

*A **ground rule** is a rule with constant arguments, i.e., a rule with only ground atoms.* □

For instance, the following is an example of two rules for solving visual Sudoku with NeuPSL.

$$\begin{aligned} 1.0: \quad & \textsc{NeuralClassifier}(\texttt{Pos}, \texttt{Digit}) = \textsc{CellDigit}(\texttt{Pos}, \texttt{Digit}) \\ & \textsc{CellDigit}(\texttt{Pos1}, \texttt{Digit1}) \wedge \textsc{SudokuViolation}(\texttt{Pos1}, \texttt{Pos2}, \texttt{Digit1}, \texttt{Digit2}) \\ & \quad \to \neg \textsc{CellDigit}(\texttt{Pos2}, \texttt{Digit2}) \quad . \end{aligned}$$

The first rule in the example above is soft as it is weighted with weight $1.0$. Moreover, the first rule is arithmetic and encodes a dependency between the digit label predicted by a neural classifier and the atom $\textsc{CellDigit}(\texttt{Pos}, \texttt{Digit})$, i.e., if the neural classifier predicts the digit `Digit` is in position `Pos`, then the `Digit` is in position `Pos`. The second rule is a hard, logical rule that encodes the rules of Sudoku. Moreover, the second rule can be read as follows: if the digit `Digit1` is in position `Pos1` and the `Digit2` in `Pos2` causes a Sudoku rule violation, then `Digit2` is not in `Pos2`.

Rules are grounded by performing every distinct substitution of the variables in the atoms for constants in their respective domain. For example, every substitution for the `Pos` and `Digit` variable arguments from the domains of non-empty Sudoku puzzle cell positions, $A1, \cdots, I9$, and digits $1, \cdots, 9$ is realized to ground the first rule:

$$\begin{aligned} 1.0: \quad & \textsc{NeuralClassifier}(\text{“}A1\text{”}, \text{“}1\text{”}) = \textsc{CellDigit}(\text{“}A1\text{”}, \text{“}1\text{”}) \\ & \vdots \\ 1.0: \quad & \textsc{NeuralClassifier}(\text{“}19\text{”}, \text{“}9\text{”}) = \textsc{CellDigit}(\text{“}I9\text{”}, \text{“}9\text{”}) \end{aligned}$$

Similarly, every substitution for the `Pos1`, `Pos2`, `Digi1`, and `Digit2` variable arguments from the domains of all Sudoku puzzle cell positions, $A1, \cdots, I9$, and digits $1, \cdots, 9$ is realized to ground the second rule:

$$\begin{aligned}
&\textsc{CellDigit}(\text{“}A1\text{”}, \text{“}1\text{”}) \wedge \textsc{SudokuViolation}(\text{“}A1\text{”}, \text{“}A2\text{”}, \text{“}1\text{”}, \text{“}1\text{”}) \\
&\quad \rightarrow \neg\textsc{CellDigit}(\text{“}A2\text{”}, \text{“}1\text{”}) \quad . \\
&\quad \vdots \\
&\textsc{CellDigit}(\text{“}I9\text{”}, \text{“}9\text{”}) \wedge \textsc{SudokuViolation}(\text{“}I9\text{”}, \text{“}I8\text{”}, \text{“}9\text{”}, \text{“}9\text{”}) \\
&\quad \rightarrow \neg\textsc{CellDigit}(\text{“}I8\text{”}, \text{“}9\text{”}) \quad .
\end{aligned}$$

### *Deep-Hinge Loss Markov Random Fields*

The rule instantiation process described in the previous subsection results in a set of ground atoms. Each ground atom is mapped to either an observed variable, $x_{sy,i}$, target variable, $y_i$, or a neural function with inputs $\mathbf{x}_{nn}$ and parameters $\mathbf{w}_{nn,i}$: $g_{nn,i}(\mathbf{x}_{nn}, \mathbf{w}_{nn,i})$. Specifically, all atoms instantiated from a deep atom are mapped to a neural function, and the observed and target atom partitions are pre-specified. Further, variables are aggregated into the vectors $\mathbf{x}_{sy} = [x_{sy_i}]_{i=1}^{n_x}$ and $\mathbf{y} = [y_i]_{i=1}^{n_y}$ and neural outputs are aggregated into the vector $\mathbf{g}_{nn} = [g_{nn,i}]_{i=1}^{n_g}$.

The ground rules and variables are used to define linear inequalities in a standard form: $\ell(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \leq 0$, where $\ell$ is a linear function of its arguments. To achieve this, logical rules are first converted into disjunctive normal form. Then, the rules are translated into linear inequalities using an extended interpretation of the logical operators, namely Łukasiewicz logic (Klir and Yuan 1995). Similarly, arithmetic rules define one or more standard form inequalities that preserve the rules' dependencies via algebraic operations.

Linear inequalities instantiated from hard ground rules are constraints in NeuPSL. Further, linear inequalities instantiated from soft ground rules define *potential functions* of the form:

$$\phi(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) := (\max\{\ell(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})), 0\})^q. \tag{12}$$

Intuitively, the value of potential is the, possibly squared, level of dissatisfaction of the linear inequality created by the ground rule. Further, each potential is associated with the weight of its instantiating rule. Weight sharing among the potentials is formalized by defining a partitioning using the instantiating rules, i.e., every potential instantiated by the same rule belongs to the same partition and shares a weight. The potentials and weights from the instantiation process are used to define a tractable class of graphical models, which we refer to as *deep hinge-loss Markov random fields* (Deep HL-MRF):

**Definition 8** Deep Hinge-Loss Markov Random Field.
*Let* $\mathbf{g}_{nn} = [g_{nn,i}]_{i=1}^{n_g}$ *be functions with corresponding weights* $\mathbf{w}_{nn} = [\mathbf{w}_{nn,i}]_{i=1}^{n_g}$ *and inputs* $\mathbf{x}_{nn}$ *such that* $g_{nn,i} : (\mathbf{w}_{nn,i}, \mathbf{x}_{nn}) \mapsto [0, 1]$. *Let* $\mathbf{y} \in [0, 1]^{n_y}$ *and* $\mathbf{x}_{sy} \in [0, 1]^{n_x}$.

*A **deep hinge-loss potential** is a function of the form:*

$$\phi(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) := (\max\{\mathbf{a}_{\phi,\mathbf{y}}^T \mathbf{y} + \mathbf{a}_{\phi,\mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{\phi,\mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_\phi, 0\})^q \tag{13}$$

*where $\mathbf{a}_{\phi,\mathbf{y}} \in \mathbb{R}^{n_y}$, $\mathbf{a}_{\phi,\mathbf{x}_{sy}} \in \mathbb{R}^{n_x}$, and $\mathbf{a}_{\phi,\mathbf{g}_{nn}} \in \mathbb{R}^{n_g}$ are variable coefficient vectors, $b_\phi \in \mathbb{R}$ is a vector of constants, and $q \in \{1, 2\}$. Let $\mathcal{T} = [\tau_i]_{i=1}^r$ denote an ordered partition of a set of $m$ deep hinge-loss potentials. Further, define*

$$\mathbf{\Phi}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) := \left[\sum_{k \in \tau_i} \phi_k(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))\right]_{i=1}^r . \tag{14}$$

*Let $\mathbf{w}_{sy}$ be a vector of $r$ non-negative symbolic weights corresponding to the partition $\mathcal{T}$. Then, a **deep hinge-loss energy function** is:*

$$E(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) := \mathbf{w}_{sy}^T \mathbf{\Phi}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})). \tag{15}$$

*Let $\mathbf{a}_{c_k,\mathbf{y}} \in \mathbb{R}^{n_y}$, $\mathbf{a}_{c_k,\mathbf{x}_{sy}} \in \mathbb{R}^{n_x}$, $\mathbf{a}_{c_k,\mathbf{g}_{nn}} \in \mathbb{R}^{n_g}$, and $b_{c_k} \in \mathbb{R}$ for each $k \in 1, \ldots, q$ and $q \geq 0$ be vectors defining linear inequality constraints and a feasible set:*

$$\mathbf{\Omega}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) := \left\{\mathbf{y} \in [0,1]^{n_y} \mid \mathbf{a}_{c_k,\mathbf{y}}^T \mathbf{y} + \mathbf{a}_{c_k,\mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{c_k,\mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_{c_k} \leq 0 \,, \forall\, k = 1, \ldots, q\right\}.$$

*Then a **deep hinge-loss Markov random field** defines the conditional probability density:*

$$P(\mathbf{y}|\mathbf{x}_{sy}, \mathbf{x}_{nn}) := \begin{cases} \frac{\exp(-E(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}))}{\int_{\hat{\mathbf{y}}} \exp(-E(\hat{\mathbf{y}}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}))d\hat{\mathbf{y}}} & \mathbf{y} \in \mathbf{\Omega}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \\ 0 & o.w. \end{cases} \tag{16}$$

□

NeuPSL models are NeSy-EBMs with an extended-value deep HL-MRF energy function capturing the constraints that define the feasible set. In other words, the symbolic component of NeuPSL is infinity if the targets are outside of the deep HL-MRF feasible set, else it is equal to the deep HL-MRF energy function:

$$g_{sy}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) = \begin{cases} \mathbf{w}_{sy}^T \mathbf{\Phi}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) & \mathbf{y} \in \mathbf{\Omega}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \\ \infty & o.w. \end{cases} \tag{17}$$

Further, NeuPSL prediction is finding the MAP state of the deep HL-MRF conditional distribution. Note that in deep HL-MRFs, the partition function is constant over the target variables. Moreover, as the exponential function is monotonically increasing, prediction is equivalent to finding the minimizer of the negative log probability of the deep HL-MRF joint distribution. This reduces to minimizing the deep HL-MRF energy function

constrained to the feasible set. Therefore, deep HL-MRF MAP inference is equivalent to minimizing the NeuPSL symbolic component in (18):

$$\begin{aligned} \arg\max_{\mathbf{y}\in\mathbb{R}^{n_\mathbf{y}}} P(\mathbf{y}|\mathbf{x}_{sy},\mathbf{x}_{nn}) &\equiv \arg\min_{\mathbf{y}\in\mathbb{R}^{n_\mathbf{y}}} g_{sy}(\mathbf{y},\mathbf{x}_{sy},\mathbf{w}_{sy},\mathbf{g}_{nn}(\mathbf{x}_{nn},\mathbf{w}_{nn})) \qquad (18)\\ &\equiv \arg\min_{\mathbf{y}\in\mathbb{R}^{n_\mathbf{y}}} \mathbf{w}_{sy}^T\mathbf{\Phi}(\mathbf{y},\mathbf{x}_{sy},\mathbf{g}_{nn}(\mathbf{x}_{nn},\mathbf{w}_{nn}))\\ &\quad \text{s.t.} \quad \mathbf{y}\in\mathbf{\Omega}(\mathbf{x}_{sy},\mathbf{g}_{nn}(\mathbf{x}_{nn},\mathbf{w}_{nn})) \qquad (19)\end{aligned}$$

Deep HL-MRF potentials are non-smooth and convex. Thus, as Deep HL-MRF energy functions are non-negative weighted sums of the potentials, they are also non-smooth and convex. Moreover, Deep HL-MRFs feasible sets are, by definition, convex polyhedrons. Therefore, Deep HL-MRF inference, as defined above in (19), is a non-smooth convex linearly constrained program. A natural extension of the definition above that is often used in practice adds support for integer constraints on the target variables. This change is useful in discrete problems and for leveraging hard logic semantics. However, adding integer constraints breaks the convexity property of MAP inference. Nevertheless, for many problems of practical scale, global minimizers or high-quality approximations of the MAP inference problem with integer constraints can be quickly found with modern solvers.

## *A Smooth Formulation of Deep HL-MRF Inference*

This subsection introduces a primal and dual formulation of Deep HL-MRF MAP inference as a linearly constrained convex quadratic program (LCQP) (see Appendix D for details). The primal and dual LCQP formulation has theoretical and practical advantages. Theoretically, the new formulation will be utilized to prove the continuity and curvature properties of Deep HL-MRFs that are valuable for learning. Practically, LCQP solvers (e.g. Gurobi (Gurobi Optimization 2024)) can be employed to achieve highly efficient MAP inference. Moreover, features of modern solvers, including support for integer constraints, can be leveraged to improve prediction.

In summary, $m$ slack variables with lower bounds and $2\cdot n_\mathbf{y}+m$ linear constraints are defined to represent the target variable bounds and deep hinge-loss potentials. All $2\cdot n_\mathbf{y}+m$ variable bounds, $m$ potentials, and $q\geq 0$ constraints are collected into a $(2\cdot n_\mathbf{y}+q+2\cdot m)\times(n_\mathbf{y}+m)$ dimensional matrix $\mathbf{A}$ and a vector of $(2\cdot n_\mathbf{y}+q+2\cdot m)$ elements that is an affine function of the neural predictions and symbolic inputs $\mathbf{b}(\mathbf{x}_{sy},\mathbf{g}_{nn}(\mathbf{x}_{nn},\mathbf{w}_{nn}))$. Moreover, the slack variables and a $(n_\mathbf{y}+m)\times(n_\mathbf{y}+m)$ positive semi-definite diagonal matrix, $\mathbf{D}(\mathbf{w}_{sy})$, and a $(n_\mathbf{y}+m)$ dimensional vector, $\mathbf{c}(\mathbf{w}_{sy})$, are created using the symbolic weights to define a quadratic objective. Further, we gather the original target variables and the slack variables into a vector $\nu\in\mathbb{R}^{n_\mathbf{y}+m}$. Altogether, the regularized convex LCQP reformulation of Deep HL-MRF MAP inference is:

$$\begin{aligned} &V(\mathbf{w}_{sy},\mathbf{b}(\mathbf{x}_{sy},\mathbf{g}_{nn}(\mathbf{x}_{nn},\mathbf{w}_{nn}))) := \qquad (20)\\ &\quad \min_{\nu\in\mathbb{R}^{n_\mathbf{y}+m}} \nu^T(\mathbf{D}(\mathbf{w}_{sy})+\epsilon\mathbf{I})\nu+\mathbf{c}(\mathbf{w}_{sy})^T\nu \quad \text{s.t. } \mathbf{A}\nu+\mathbf{b}(\mathbf{x}_{sy},\mathbf{g}_{nn}(\mathbf{x}_{nn},\mathbf{w}_{nn}))\leq 0,\end{aligned}$$

where $\epsilon \geq 0$ is a scalar regularization parameter added to the diagonal of $\mathbf{D}$ to ensure strong convexity. The function $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ in (20) is the *optimal value-function* of the LCQP formulation of NeuPSL inference.

By Slater's constraint qualification, we have strong duality when there is a feasible solution to (20) Boyd and Vandenberghe (2004). In this case, an optimal solution to the dual problem yields an optimal solution to the primal problem. The Lagrange dual problem of (20) is:

$$\begin{aligned} \min_{\mu \in \mathbb{R}^{2\cdot(n_{\mathbf{y}}+m)+q}_{\geq 0}} \quad & h(\mu; \mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) \\ := \quad & \frac{1}{4}\mu^T \mathbf{A}(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon \mathbf{I})^{-1}\mathbf{A}^T\mu \\ & + \frac{1}{2}(\mathbf{A}(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon \mathbf{I})^{-1}\mathbf{c}(\mathbf{w}_{sy}) \\ & - 2\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))^T\mu, \end{aligned} \tag{21}$$

where $\mu$ is the vector of dual variables and $h(\mu; \mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ is the LCQP dual objective function. As $(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon \mathbf{I})$ is diagonal, it is easy to invert, and thus it is practical to work in the dual space and map dual to primal variables. The dual-to-primal variable mapping is:

$$\nu \leftarrow -\frac{1}{2}(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon \mathbf{I})^{-1}(\mathbf{A}^T\mu + \mathbf{c}(\mathbf{w}_{sy})). \tag{22}$$

On the other hand, the primal-to-dual mapping is more computationally expensive and requires calculating a pseudo-inverse of the constraint matrix $\mathbf{A}$.

We use the LCQP formulation in (20) to establish continuity and curvature properties of the NeuPSL energy minimizer and the optimal value-function provided in the following theorem:

**Theorem 9**

*Suppose for any setting of $\mathbf{w}_{nn} \in \mathbb{R}^{n_g}$ there is a feasible solution to NeuPSL inference (20). Further, suppose $\epsilon > 0$, $\mathbf{w}_{sy} \in \mathbb{R}^r_+$, and $\mathbf{w}_{nn} \in \mathbb{R}^{n_g}$. Then:*

- *The minimizer of (20), $\mathbf{y}^*(\mathbf{w}_{sy}, \mathbf{w}_{nn})$, is a $O(1/\epsilon)$ Lipschitz continuous function of $\mathbf{w}_{sy}$.*
- *$V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$, is concave over $\mathbf{w}_{sy}$ and convex over $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$.*
- *$V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ is differentiable with respect to $\mathbf{w}_{sy}$. Moreover,*

$$\nabla_{\mathbf{w}_{sy}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) = \mathbf{\Phi}(\mathbf{y}^*(\mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})).$$

  *Furthermore, $\nabla_{\mathbf{w}_{sy}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ is Lipschitz continuous over $\mathbf{w}_{sy}$.*
- *If there is a feasible point $\nu$ strictly satisfying the $i'th$ inequality constraint of (20), i.e., $\mathbf{A}[i]\nu + \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))[i] < 0$, then $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ is subdifferentiable with respect to the $i'th$ constraint constant $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))[i]$.*

*Moreover,*

$$\partial_{\mathbf{b}[i]} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) = \{\mu^*[i] \,|\, \mu^* \in \underset{\mu \in \mathbb{R}^{2\cdot(n_{\mathbf{y}}+m)+q}_{\geq 0}}{\arg\min} \; h(\mu; \mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))\}.$$

*Furthermore, if* $\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})$ *is a smooth function of* $\mathbf{w}_{nn}$*, then so is* $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$*, and the set of regular subgradients of* $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ *is:*

$$\hat{\partial}_{\mathbf{w}_{nn}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) \supset \nabla_{\mathbf{w}_{nn}} \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))^T \partial_{\mathbf{b}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))). \tag{23}$$

**Proof.** See Appendix D.

Theorem 9 establishes the continuity properties of the NeuPSL optimal value-function, complementing the results in the following section, specifically in Theorem 10. Further, it provides a simple explicit form of the value-function gradient with respect to the symbolic weights and a regular subgradient with respect to the neural weights. Thus, Theorem 9 supports the principled application of the end-to-end learning algorithms presented in the following sections for training both the symbolic and neural weights of a NeuPSL model.

## 5 A Suite of Learning Techniques for NeSy

Having identified a variety of modeling and inference paradigms, we turn to learning. This section formalizes the NeSy-EBM learning problem, identifies challenges, and proposes effective solutions. At a high level, NeSy-EBM learning is finding weights of an energy function that associates higher compatibility scores (lower energy) to targets and neural outputs near their true labels provided in training data. Further, predictions with NeSy-EBMs are obtained by minimizing a complex mathematical program, raising several obstacles to learning. For instance, NeSy-EBM predictions may not be differentiable with respect to the model parameters, and a direct application of automatic differentiation may not be possible or may fail to produce principled descent directions for the learning objective. Moreover, we will show that even when predictions are differentiable, their gradients are functions of properties of the energy function at its minimizer that are prohibitively expensive to compute. We create general and principled learning frameworks for NeSy-EBMs that address these challenges.

This section is organized into four subsections. We begin with preliminary notation and a general definition of NeSy-EBM learning. Then, we present a classification of learning losses and establish theoretical differentiability results for NeSy-EBMs. The learning losses motivate and organize the exposition of four NeSy-EBM learning frameworks, one for learning the neural and symbolic weights separately and three for end-to-end learning.

### *NeSy-EBM Learning*

We use the following notation and general definition of NeSy-EBM learning throughout this section. The training dataset, denoted by $\mathcal{S}$, is comprised of $P$ samples and indexed by $\{1, \cdots, P\}$. Each sample, $\mathcal{S}_i$ where $i \in \{1, \cdots, P\}$, is a tuple of *inputs*, *labels*, and *latent variable domains*. Sample *inputs* consist of neural inputs, $\mathbf{x}^i_{nn}$ from $\mathcal{X}_{nn}$, and symbolic inputs, $\mathbf{x}^i_{sy}$ from $\mathcal{X}_{sy}$. Similarly, sample *labels* consist of neural and symbolic labels, which are truth values corresponding to a subset of the neural predictions and target variables, respectively. Neural labels, denoted by $\mathbf{t}^i_{nn}$, are $d^i_{nn} \leq d_{nn}$ dimensional real vectors from a domain $\mathcal{T}^i_{nn}$, i.e., $\mathbf{t}^i_{nn} \in \mathcal{T}^i_{nn} \subseteq \mathbb{R}^{d^i_{nn}}$. Target labels, denoted by $\mathbf{t}^i_{\mathcal{Y}}$, are from a domain $\mathcal{T}^i_{\mathcal{Y}}$ that is a $d_{\mathcal{T}^i_{\mathcal{Y}}} \leq d_{\mathcal{Y}}$ subspace of the target domain $\mathcal{Y}$, i.e., $\mathbf{t}^i_{\mathcal{Y}} \in \mathcal{T}^i_{\mathcal{Y}}$. Lastly, the neural and symbolic *latent variable domains* are subspaces of the range of the neural component and the target domain, respectively, corresponding to the set of unlabeled variables. The range of the neural component, $\mathbb{R}^{d^{nn}}$, is a superset of the Cartesian product of the neural latent variable domain, denoted by $\mathcal{Z}^i_{nn}$, and $\mathcal{T}^i_{nn}$, i.e., $\mathbb{R}^{d^{nn}} \supseteq \mathcal{T}^i_{nn} \times \mathcal{Z}^i_{nn}$. Similarly, the target domain $\mathcal{Y}$ is a superset of the Cartesian product of the latent variable domain, denoted by $\mathcal{Z}^i_{\mathcal{Y}}$, and $\mathcal{T}^i_{\mathcal{Y}}$, i.e., $\mathcal{Y} \supseteq \mathcal{T}^i_{\mathcal{Y}} \times \mathcal{Z}^i_{\mathcal{Y}}$. With this notation, the training dataset is expressed as follows:

$$\mathcal{S} := \{(\mathbf{t}^1_{\mathcal{Y}}, \mathbf{t}^1_{nn}, \mathcal{Z}^1_{nn}, \mathcal{Z}^1_{\mathcal{Y}}, \mathbf{x}^1_{sy}, \mathbf{x}^1_{nn}), \cdots, (\mathbf{t}^P_{\mathcal{Y}}, \mathbf{t}^P_{nn}, \mathcal{Z}^P_{nn}, \mathcal{Z}^P_{\mathcal{Y}}, \mathbf{x}^P_{sy}, \mathbf{x}^P_{nn})\}. \tag{24}$$

A learning objective, denoted by $\mathcal{L}$, is a functional that maps an energy function and a training dataset to a scalar value. Formally, let $\mathcal{E}$ be a family of energy functions indexed by weights from $\mathcal{W}_{sy} \times \mathcal{W}_{nn}$:

$$\mathcal{E} := \{E(\cdot, \cdot, \cdot, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \mid (\mathbf{w}_{sy}, \mathbf{w}_{nn}) \in \mathcal{W}_{sy} \times \mathcal{W}_{nn}\}. \tag{25}$$

Then, a learning objective is the function:

$$\mathcal{L} : \mathcal{E} \times \{\mathcal{S}\} \to \mathbb{R}. \tag{26}$$

Learning objectives follow the standard empirical risk minimization framework and are separable over elements of $\mathcal{S}$ as a sum of *per-sample loss functionals* denoted by $L^i$ for each $i \in \{1, \cdots, P\}$. A loss functional for the sample $\mathcal{S}_i \in \mathcal{S}$ is the function:

$$L^i : \mathcal{E} \times \{\mathcal{S}_i\} \to \mathbb{R}. \tag{27}$$

A regularizer, denoted by $\mathcal{R} : \mathcal{W}_{sy} \times \mathcal{W}_{nn} \to \mathbb{R}$, is added to the learning objective and NeSy-EBM learning is the following minimization problem:

$$\begin{aligned} &\underset{(\mathbf{w}_{sy}, \mathbf{w}_{nn}) \in \mathcal{W}_{sy} \times \mathcal{W}_{nn}}{\arg\min} \mathcal{L}(E(\cdot, \cdot, \cdot, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathcal{S}) + \mathcal{R}(\mathbf{w}_{sy}, \mathbf{w}_{nn}) \\ &= \underset{(\mathbf{w}_{sy}, \mathbf{w}_{nn}) \in \mathcal{W}_{sy} \times \mathcal{W}_{nn}}{\arg\min} \frac{1}{P} \sum_{i=1}^{P} L^i(E(\cdot, \cdot, \cdot, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathcal{S}_i) + \mathcal{R}(\mathbf{w}_{sy}, \mathbf{w}_{nn}). \end{aligned} \tag{28}$$

### Learning Losses

A NeSy-EBM learning loss functional, $L^i$, is separable into three parts: *neural*, *value-based*, and *minimizer-based* losses. In this subsection, we formally define each of the three loss types. At a high level, the neural loss measures the quality of the neural component independent from the symbolic component. Then, the value-based and minimizer-based losses measure the quality of the NeSy-EBM as a whole. Moreover, value-based and minimizer-based losses are functionals mapping a parameterized energy function and a training sample to a real value and are denoted by $L_{Val} : \mathcal{E} \times \mathcal{S} \to \mathbb{R}$ and $L_{Min} : \mathcal{E} \times \mathcal{S} \to \mathbb{R}$, respectively. The learning loss components are aggregated via summation:

$$
\begin{aligned}
& L^i(E(\cdot,\cdot,\cdot,\mathbf{w}_{sy},\mathbf{w}_{nn}),\mathcal{S}_i) && && (29)\\
&= \quad L_{NN}(\mathbf{g}_{nn}(\mathbf{x}^i_{nn},\mathbf{w}_{nn}),\mathbf{t}^i_{nn}) && \text{Neural}\\
&\quad + L_{Val}(E(\cdot,\cdot,\cdot,\mathbf{w}_{sy},\mathbf{w}_{nn}),\mathcal{S}_i) && \text{Value-Based}\\
&\quad + L_{Min}(E(\cdot,\cdot,\cdot,\mathbf{w}_{sy},\mathbf{w}_{nn}),\mathcal{S}_i) && \text{Minimizer-Based}
\end{aligned}
$$

*Neural Learning Losses* *Neural* learning losses are scalar functions of the neural network output and the neural labels and are denoted by $L_{NN} : \text{Range}(\mathbf{g}_{nn}) \times \mathcal{T}^i_{nn} \to \mathbb{R}$. For example, a neural learning loss may be the familiar binary cross-entropy loss applied in many categorical prediction settings. Minimizing a neural learning loss with respect to neural component parameters is achievable via backpropagation and standard gradient-based algorithms.

*Value-Based Learning Losses* *Value-based* learning losses depend on the model weights strictly via minimizing values of an objective defined with the energy. More formally, denote an objective function by $f$, which maps a compatibility score, target variables, and the training sample to a scalar value:

$$
f : \mathbb{R} \times \mathcal{Y} \times \{\mathcal{S}_i\} \to \mathbb{R}. \tag{30}
$$

An *optimal value-function*, denoted by $V$, is the value of $f$ composed with the energy function and minimized over the target variables:

$$
\begin{aligned}
V(\mathbf{w}_{sy},\mathbf{w}_{nn},\mathcal{S}_i) &:= \min_{\hat{\mathbf{y}}\in\mathcal{Y}} f\left(E(\hat{\mathbf{y}},\mathbf{x}^i_{sy},\mathbf{x}^i_{nn},\mathbf{w}_{sy},\mathbf{w}_{nn}),\hat{\mathbf{y}},\mathcal{S}_i\right)\\
&:= \min_{\hat{\mathbf{y}}\in\mathcal{Y}} f(g_{sy}(\hat{\mathbf{y}},\mathbf{x}^i_{sy},\mathbf{w}_{sy},\mathbf{g}_{nn}(\mathbf{x}^i_{nn},\mathbf{w}_{nn})),\hat{\mathbf{y}},S_i) \qquad (31)
\end{aligned}
$$

Value-based learning losses are functions of one or more optimal value functions. In this work, we consider three instances of optimal value functions: 1) *latent*, $V_{\mathcal{Z}}$, 2) *full*, $V_{\mathcal{Y}}$, 3) and *convolutional*, $V_{conv}$. The latent optimal value function is the minimizing value of the energy over the latent targets. Further, the labeled targets are fixed to their true values using the following indicator function:

$$
I_{\mathcal{T}^i_{\mathcal{Y}}}(\mathbf{y},\mathbf{t}^i_{\mathcal{Y}}) := \begin{cases} 0 & \mathbf{y} = \mathbf{t}^i_{\mathcal{Y}} \\ \infty & \text{o.w.} \end{cases}. \tag{32}
$$

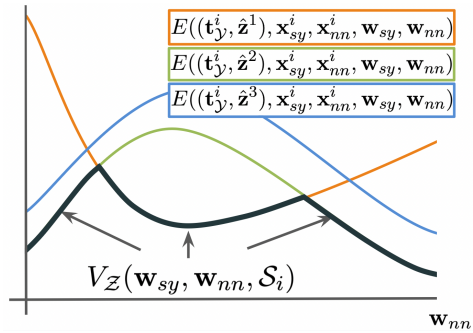



**Figure 4.** An illustrated example of a latent optimal value-function with a scalar neural component output and a discrete latent variable domain $\mathcal{Z} := \{\hat{\mathbf{z}}^1, \hat{\mathbf{z}}^2, \hat{\mathbf{z}}^3\}$.

The full optimal value function is the minimizing value of the energy over all of the targets. Lastly, the convolutional optimal value function is the infimal convolution of the energy function and a function $d : \mathcal{Y} \times \mathcal{Y} \to \mathcal{R}$ scaled by a positive real value $\lambda \in \mathcal{R}$. Formally:

$$\begin{aligned} V_{\mathcal{Z}}(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i) &:= \min_{\hat{\mathbf{y}} \in \mathcal{Y}} E(\hat{\mathbf{y}}, \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) + I_{\mathcal{T}^i_{\mathcal{Y}}}(\hat{\mathbf{y}}, \mathbf{t}^i_{\mathcal{Y}}), \\ &= \min_{\hat{\mathbf{z}} \in \mathcal{Z}^i_{\mathcal{Y}}} E((\mathbf{t}^i_{\mathcal{Y}}, \hat{\mathbf{z}}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \qquad \text{latent} \end{aligned} \tag{33}$$

$$V_{\mathcal{Y}}(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i) := \min_{\hat{\mathbf{y}} \in \mathcal{Y}} E(\hat{\mathbf{y}}, \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \qquad \text{full} \tag{34}$$

$$V_{conv}(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i; \mathbf{y}, \lambda) := \min_{\hat{\mathbf{y}} \in \mathcal{Y}} E(\hat{\mathbf{y}}, \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) + \lambda \cdot d(\hat{\mathbf{y}}, \mathbf{y}). \qquad \text{convolutional} \tag{35}$$

An illustration of an example latent optimal value-function is provided in Fig. 4. Intuitively, the latent optimal value-function is the greatest lower bound of the set of symbolic components defined for each latent variable.

The simplest value-based learning loss is the *energy loss*, denoted by $L_{Energy}$. The energy loss is the latent optimal value function,

$$L_{Energy}(E(\cdot, \cdot, \cdot, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathcal{S}_i) := V_{\mathcal{Z}}(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i). \tag{36}$$

Minimizing the energy loss encourages the parameters of the energy function to produce low energies given the observed true values of the input and target variables. This loss is motivated by the intuition that the energy should be low for the desired values of the targets. Notice, however, that the loss does not consider the energy of incorrect target variable values. An extreme illustration of the issue this causes involves two energy functions. In the first function, the minimizing point corresponds to the desired true values of the targets, while in the second function, the maximizing point corresponds to the desired true values of the targets. Despite these differences, both functions could technically have the

same energy loss; however, the first energy function is clearly preferred. Thus, the energy loss does not universally lead to energy functions with better predictions.

The *Structured Perceptron* loss, denoted by $L_{SP}$, pushes the energy of the current energy minimizer up and the energy of the true values of the targets down (LeCun et al. 1998; Collins 2002). Specifically, the structured perceptron loss is the difference between the latent and full optimal value functions,

$$L_{SP}(E(\cdot,\cdot,\cdot,\mathbf{w}_{sy},\mathbf{w}_{nn}),\mathcal{S}_i) := V_{\mathcal{Z}}(\mathbf{w}_{sy},\mathbf{w}_{nn},\mathcal{S}_i) - V_{\mathcal{Y}}(\mathbf{w}_{sy},\mathbf{w}_{nn},\mathcal{S}_i). \tag{37}$$

Although the structured perceptron loss will technically encourage the target's desired values to be an energy minimizer, i.e., a valid prediction, it still has degenerate solutions for some energy function architectures. For instance, one could minimize the energy for all target values, leading to a collapsed energy function (equal energy for all targets) with no predictive power.

The energy and structured perceptron losses require regularization and specific energy architectures to work well in practice. For instance, energy architectures that naturally push up on other target values when pushing down on the desired targets. Energies with limited total energy mass are examples of functions with this property.

The gradient of a value-based loss with respect to neural and symbolic weights is non-trivial since both the energy function and the point the energy function is evaluated at are dependent on the neural output and symbolic weights, as exemplified by the definition of an optimal value function in (31). Nonetheless, Milgrom and Segal (2002) delivers a general theorem providing the gradient of optimal value-functions with respect to problem parameters, if they exist. We specialize their result in the following theorem for optimal value-functions of NeSy-EBMs.

**Theorem 10** Milgrom and Segal (2002) Theorem 1 for NeSy-EBMs.
*Consider the weights* $\mathbf{w}_{sy} \in \mathcal{W}_{sy}$ *and* $\mathbf{w}_{nn} \in \mathcal{W}_{nn}$ *and the sample* $\mathcal{S}_i = (\mathbf{t}^i_{\mathbf{y}}, \mathbf{t}^i_{nn}, \mathcal{Z}^i_{nn}, \mathcal{Z}^i_{\mathcal{Y}}, \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}) \in \mathcal{S}$. *Suppose there exists a minimizer of the objective function* $f$,

$$\mathbf{y}^* \in \arg\min_{\hat{\mathbf{y}}\in\mathcal{Y}} f(E(\hat{\mathbf{y}},\mathbf{x}^i_{sy},\mathbf{x}^i_{nn},\mathbf{w}_{sy},\mathbf{w}_{nn}),\hat{\mathbf{y}},\mathcal{S}_i),$$

*such that* $f(E(\mathbf{y}^*,\mathbf{x}^i_{sy},\mathbf{x}^i_{nn},\mathbf{w}_{sy},\mathbf{w}_{nn}),\mathbf{y}^*,\mathcal{S}_i)$ *is finite.*
*If the optimal value-function:*

$$\begin{aligned} V(\mathbf{w}_{sy},\mathbf{w}_{nn},\mathcal{S}_i) &:= \min_{\hat{\mathbf{y}}\in\mathcal{Y}} f(E(\hat{\mathbf{y}},\mathbf{x}^i_{sy},\mathbf{x}^i_{nn},\mathbf{w}_{sy},\mathbf{w}_{nn}),\hat{\mathbf{y}},\mathcal{S}_i), \\ &:= \min_{\hat{\mathbf{y}}\in\mathcal{Y}} f(g_{sy}(\hat{\mathbf{y}},\mathbf{x}^i_{sy},\mathbf{w}_{sy},\mathbf{g}_{nn}(\mathbf{x}^i_{nn},\mathbf{w}_{nn})),\hat{\mathbf{y}},S_i), \end{aligned}$$

*is differentiable with respect to the neural weights,* $\mathbf{w}_{nn}$, *then the gradient of* $V$ *with respect to* $\mathbf{w}_{nn}$ *is:*

$$\begin{aligned} &\nabla_{\mathbf{w}_{nn}} V(\mathbf{w}_{sy},\mathbf{w}_{nn},\mathcal{S}_i) \\ &= \frac{\partial}{\partial 1} f(E(\mathbf{y}^*,\mathbf{x}^i_{sy},\mathbf{x}^i_{nn},\mathbf{w}_{sy},\mathbf{w}_{nn}),\mathbf{y}^*,\mathcal{S}_i) \cdot \nabla_5 E(\mathbf{y}^*,\mathbf{x}^i_{sy},\mathbf{x}^i_{nn},\mathbf{w}_{sy},\mathbf{w}_{nn}), \end{aligned} \tag{38}$$

*where $\frac{\partial}{\partial 1} f$ is the partial derivative of $f$ with respect to its $1$st argument, and $\nabla_5 E$ is the gradient of the energy with respect to its $5$th argument with all other arguments fixed.*

*Similarly, if $V$ is differentiable with respect to the symbolic weights, $\mathbf{w}_{sy}$, then the gradient of $V$ with respect to $\mathbf{w}_{sy}$ is:*

$$\begin{aligned}&\nabla_{\mathbf{w}_{sy}} V(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i) \\ &\quad = \frac{\partial}{\partial 1} f(E(\mathbf{y}^*, \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{y}^*, \mathcal{S}_i) \cdot \nabla_4 E(\mathbf{y}^*, \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}).\end{aligned} \tag{39}$$

**Proof.** We first establish the partial derivative of the optimal value-function with respect to each component of the neural output, $\mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})$. Then, we use the chain rule to derive the expression for the gradient of the optimal value-function with respect to the neural weights, $\mathbf{w}_{nn}$.

For an arbitrary index $j \in \{1, \cdots, d_{nn}\}$, let $\mathbf{e}^j$ be the $j'th$ standard basis vector of $\mathbb{R}^{d_{nn}}$, i.e., $\mathbf{e}^j \in \mathbb{R}^{d_{nn}}$ such that $\mathbf{e}^j_j = 1$ and $\mathbf{e}^j_k = 0$ for $k \neq j$. Further, to clarify the relationship between the optimal value-function and the neural component output, define the following function:

$$\begin{aligned}\overline{V} : \mathcal{W}_{sy} \times \mathbb{R}^{d_{nn}} \times \mathcal{S}_i &\to \mathbb{R} \\ (\mathbf{w}_{sy}, \mathbf{u}, S_i) &\mapsto \min_{\hat{\mathbf{y}} \in \mathcal{Y}} f(g_{sy}(\hat{\mathbf{y}}, \mathbf{x}^i_{sy}, \mathbf{w}_{sy}, \mathbf{u}), \hat{\mathbf{y}}, \mathcal{S}_i)\end{aligned}$$

In other words, the optimal value-function, $V$, is equal to $\overline{V}$ evaluated at the neural output:

$$V(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i) \triangleq \overline{V}(\mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn}), \mathcal{S}_i).$$

For any $\delta \in \mathbb{R}$, by definition we have:

$$\begin{aligned}&f(g_{sy}(\mathbf{y}^*, \mathbf{x}^i_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn}) + \delta \mathbf{e}^j), \mathbf{y}^*, S_i) \\ &\quad - f(g_{sy}(\mathbf{y}^*, \mathbf{x}^i_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})), \mathbf{y}^*, S_i) \\ &\geq \overline{V}(\mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn}) + \delta \mathbf{e}^j, S_i) - \overline{V}(\mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn}), S_i).\end{aligned}$$

For $\delta \neq 0$, dividing both sides by $\delta$ and taking the limit as $\delta \to 0+$ and as $\delta \to 0-$ yields upper and lower bounds relating partial derivatives of $f$ to $\overline{V}$ when $f$ and $\overline{V}$ are right and left hand differentiable, respectively.

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})_j +} f(g_{sy}(\mathbf{y}^*, \mathbf{x}^i_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})), \mathbf{y}^*, S_i) \\ &\quad \geq \frac{\partial}{\partial \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})_j +} \overline{V}(\mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn}), S_i), \\ &\frac{\partial}{\partial \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})_j -} f(g_{sy}(\mathbf{y}^*, \mathbf{x}^i_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})), \mathbf{y}^*, S_i) \\ &\quad \leq \frac{\partial}{\partial \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})_j -} \overline{V}(\mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn}), S_i),\end{aligned}$$

Then, by the squeeze theorem, we obtain the partial derivatives of $\overline{V}$ with respect to each component of the neural output when $\overline{V}$ is differentiable.

$$\frac{\partial}{\partial \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})_j} f(g_{sy}(\mathbf{y}^*, \mathbf{x}^i_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})), \mathbf{y}^*, S_i) = \frac{\partial}{\partial \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})_j} \overline{V}(\mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn}), S_i).$$

Then, the chain rule of differentiation and the partial derivatives of $\overline{V}$ with respect to each component of the neural output derived above yields the gradient in (38).

A similar approach is used to obtain gradients with respect to symbolic weights in (39).

Theorem 10 holds for arbitrary target variable domains and energy functions and is, therefore, widely applicable. However, it is important to emphasize that Theorem 10 states *if* the value-function is differentiable, then the gradients have the form provided in (38) and (39). Milgrom and Segal (2002) also provide sufficient conditions for guaranteeing the differentiability of optimal value-functions with arbitrary decision variable domains. Beyond Milgrom and Segal's (2002) work, there is extensive literature on analyzing the sensitivity of optimal value-functions and guaranteeing their differentiability, including the seminal papers of Danskin (1966) on parameterized objective functions and Rockafellar (1974) for parameterized constraints. We direct the reader to the cited articles for properties that guarantee differentiability of value-functions and, hence, NeSy-EBM value-based losses.

The conditions ensuring differentiability of the optimal value-functions as well as the tractability of computing the gradient of the symbolic component with respect to its arguments in (38) and (39) directly connect to the energy function architecture and modeling paradigms discussed in the previous section. Specifically, if principled gradient-based learning is desired, then practitioners must design the symbolic potential such that it is 1) differentiable with respect to the neural output and symbolic potentials, 2) the gradient of the symbolic potential with respect to its arguments is tractable, and 3) it satisfies sufficient conditions for ensuring differentiability of its minimizing value over the targets.

Performance metrics are not always aligned with value-based losses. Moreover, they are known to have degenerate solutions (LeCun et al. 2006; Pryor et al. 2023a). For example, without a carefully designed inductive bias, the energy loss in (36) may only learn to reduce the energy of all target variables without improving the predictive performance of the NeSy-EBM. One fundamental cause of this issue is that value-based losses are not directly functions of the NeSy-EBM prediction as defined in (4), i.e., value-based losses are not functions of an energy minimizer, which is what we turn to next.

*Minimizer-Based Learning Losses* A *minimizer-based* loss is a composition of a differentiable loss, such as cross-entropy or mean squared error, with the energy minimizer. Intuitively, minimizer-based losses penalize parameters yielding predictions distant from the labeled training data. In the remainder of this subsection, we formally define minimizer-based learning losses. Further, for completeness, we derive general expressions for

gradients of minimizer-based losses with respect to symbolic and neural weights. However, as will be shown, direct computation of minimizer-based loss gradients requires prohibitive assumptions on the energy function and can be impractical to compute. Moreover, the derivation of the gradients motivates learning algorithms that do not perform direct gradient descent on minimizer-based losses. For this reason, in the following subsection we propose algorithms that do not require minimizer gradients.

To ensure a minimizer-based loss is well-defined, we assume a unique energy minimizer exists, denoted by $\mathbf{y}^*$, for every training sample. This assumption is formalized below.

**Assumption 1**
*The energy function is minimized over the targets at a single point for every input and weight and is, therefore, a function:*

$$\begin{aligned}\mathbf{y}^* : \mathcal{X}_{sy} \times \mathcal{X}_{nn} \times \mathcal{W}_{sy} \times \mathcal{W}_{nn} &\to \mathcal{Y} \\ (\mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) &\mapsto \arg\min_{\hat{\mathbf{y}} \in \mathcal{Y}} E(\hat{\mathbf{y}}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})\end{aligned} \tag{40}$$

Under Assumption 1, $d$ is a mapping of targets and labels to a scalar value:

$$d : \mathcal{Y} \times \mathcal{T}_{\mathcal{Y}}^i \to \mathbb{R}, \tag{41}$$

and a minimizer-based loss is a composition of $d$ and $\mathbf{y}^*$:

$$\begin{aligned}L_{Min}(E(\cdot,\cdot,\cdot,\mathbf{w}_{sy},\mathbf{w}_{nn}),\mathcal{S}_i) &:= d(\arg\min_{\hat{\mathbf{y}} \in \mathcal{Y}} E(\hat{\mathbf{y}}, \mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{t}_{\mathcal{Y}}^i) \\ &:= d(\mathbf{y}^*(\mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{t}_{\mathcal{Y}}^i)\end{aligned} \tag{42}$$

To ensure principled gradient-based learning, we must further assume that the minimizer is differentiable.

**Assumption 2**
*The minimizer, $\mathbf{y}^*$, is differentiable with respect to the weights at every point in $\mathcal{X}_{sy} \times \mathcal{X}_{nn} \times \mathcal{W}_{sy} \times \mathcal{W}_{nn}$.*

Under Assumption 2, the chain rule of differentiation yields the gradient of a minimizer-based loss with respect to the neural and symbolic weights:

$$\begin{aligned}&\nabla_{\mathbf{w}_{sy}} L_{Min}(\mathbf{y}^*(\mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn})), \mathbf{t}_{\mathcal{Y}}^i) \\ &\quad = \nabla_3 \mathbf{y}^*(\mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn})^T \nabla_1 d(\mathbf{y}^*(\mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{t}_{\mathcal{Y}}^i),\end{aligned} \tag{43}$$

$$\begin{aligned}&\nabla_{\mathbf{w}_{nn}} L_{Min}(\mathbf{y}^*(\mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn})), \mathbf{t}_{\mathcal{Y}}^i) \\ &\quad = \nabla_4 \mathbf{y}^*(\mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn})^T \nabla_1 d(\mathbf{y}^*(\mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{t}_{\mathcal{Y}}^i),\end{aligned} \tag{44}$$

where $\nabla_3 \mathbf{y}^*(\mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn})$ and $\nabla_4 \mathbf{y}^*(\mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn})$ are the Jacobian matrices of the unique energy minimizer with respect to the third and fourth arguments of $\mathbf{y}^*$, the symbolic and neural weights, respectively, and $\nabla_1 d(\mathbf{y}^*(\mathbf{x}_{sy}^i, \mathbf{x}_{nn}^i, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{t}_{\mathcal{Y}}^i)$ is the gradient of the supervised loss with respect to its first argument.

A primary challenge of minimizer-based learning is computing the Jacobian matrices of partial derivatives, $\nabla_3 \mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})$ and $\nabla_4 \mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})$. To derive explicit expressions for them typically demands the following additional assumption on the continuity properties of the energy function.

**Assumption 3**
*The energy, $E$, is twice differentiable with respect to the targets at the minimizer, $\mathbf{y}^*$, and the Hessian matrix of second-order partial derivatives with respect to the targets, $\nabla_{1,1} E(\mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})$, is invertible. Further, the minimizer is the unique target satisfying first-order conditions of optimality, i.e.,*

$$\forall \mathbf{y} \in \mathcal{Y}, \quad \nabla_1 E(\mathbf{y}, \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) = 0 \iff \mathbf{y} = \mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \tag{45}$$

Assumption 3 is satisfied by energy functions that are, for instance, smooth and strongly convex in the targets. Under Assumption 3, the first-order optimality condition establishes the minimizer as an implicit function of the weights, and implicit differentiation yields the following equalities:

$$\begin{aligned} &\nabla_{1,1} E(\mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \nabla_3 \mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \\ &\quad = -\nabla_{1,4} E(\mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \end{aligned} \tag{46}$$

$$\begin{aligned} &\nabla_{1,1} E(\mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \nabla_4 \mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \\ &\quad = -\nabla_{1,5} E(\mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \end{aligned} \tag{47}$$

Solving for the Jacobians of the minimizer:

$$\begin{aligned} \nabla_3 \mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) = -\big(&\nabla_{1,1} E(\mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})^{-1} \\ &\nabla_{1,4} E(\mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})\big), \end{aligned} \tag{48}$$

$$\begin{aligned} \nabla_4 \mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) = -\big(&\nabla_{1,1} E(\mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})^{-1} \\ &\nabla_{1,5} E(\mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})\big). \end{aligned} \tag{49}$$

The Jacobians in (48) and (49) applied to (43) and (44), respectively, are referred to as hypergradients in the machine learning literature and are utilized in hyperparameter optimization and meta-learning (Do et al. 2007; Pedregosa 2016; Rajeswaran et al. 2019). Oftentimes, approximations of the (inverse) Hessian matrices are made to estimate the hypergradient.

## *Learning Algorithms*

Next, we present four principled techniques for learning the neural and symbolic weights of a NeSy-EBM to minimize the losses introduced in the previous subsection: 1) Modular,

2) Gradient Descent, 3) Bilevel Value-Function Optimization, and 4) Stochastic Policy Optimization. The four techniques are defined, and we discuss their strengths and limitations in relation to the modeling paradigms in Section 3.

*Modular Learning* The first and most straightforward NeSy-EBM learning technique is to train and connect the neural and symbolic components as independent modules. For instance, the neural component can be trained via backpropagation and Adam to optimize a neural loss given neural labels. Then, the symbolic component can be trained using an appropriate method to optimize a value or minimizer-based loss. The neural component weights are frozen during the symbolic weight learning process.

By definition, modular learning algorithms are not trained end-to-end, i.e., the neural and symbolic parameters are not jointly optimized to minimize the learning loss. For this reason, modular approaches may struggle to find a weight setting with a learning loss as low as end-to-end techniques. Moreover, modular approaches are not suitable for fine-tuning and adaptation. Additionally, they require labels to train the neural component. Thus, modular learning is not used to learn neural parameters in unsupervised or semi-supervised settings.

Nevertheless, modular learning approaches are appealing and widely used for their simplicity and general applicability. Importantly, no assumptions are made about the neural-symbolic interface; hence, modular learning is effective for every modeling paradigm presented in Section 3. Notably, minimizers and value-functions of DSPot models are typically non-differentiable with respect to the neural weights due to the complex neural-symbolic interface. However, because modular techniques are not end-to-end, this is not an issue. Moreover, modular learning can be used to train a NeSy-EBM for constraint satisfaction and joint reasoning, zero-shot reasoning, and reasoning with noisy data. There are many established and effective modular neural and symbolic learning algorithms (see Srinivasan et al. (2021) for a recent taxonomy of symbolic weight learning algorithms).

*Gradient Descent* A conceptually simple but oftentimes difficult in-practice technique for end-to-end NeSy-EBM training is direct gradient descent. Specifically, the gradients derived in the previous subsection are directly used with a gradient-based algorithm to optimize a NeSy-EBM loss with respect to both the neural and symbolic weights. Backpropagation and Theorem 10 produce relatively inexpensive gradients for neural and value-based losses for a general class of NeSy-EBMs. Moreover, for a smaller family of NeSy-EBMs, gradients of energy minimizers exist and may be cheap to compute. For instance, if the energy minimizer is determined via a simple closed-form expression (e.g., if inference is an unconstrained strongly convex quadratic program or a finite computation graph).

As shown in Section 5, learning loss gradients for fully expressive NeSy-EBMs only exist under certain conditions. Further, computing the gradients generally requires expensive second-order information about the energy function at the minimizer. For this reason, direct gradient descent only applies to a relatively small class of NeSy-EBMs with specialized architectures that ensure principled and efficient gradient computation. Such

specialized architectures are less likely to support more complex modeling paradigms such as DSPar and DSPot.

*Bilevel Value-Function Optimization* As shown in the earlier subsection, minimizer gradients are relatively more computationally expensive to compute and require more assumptions than value-function gradients. In this subsection, we devise a technique for optimizing a minimizer-based loss with only first-order gradients. This technique is built on the fact that the general definition of NeSy-EBM learning (28) is naturally formulated as bilevel optimization. In other words, the NeSy learning objective is a function of variable values obtained by solving a lower-level inference problem that is symbolic reasoning:

$$\begin{aligned}\underset{\substack{(\mathbf{w}_{sy},\mathbf{w}_{nn})\in\mathcal{W}_{sy}\times\mathcal{W}_{nn}\\(\hat{\mathbf{y}}^1,\cdots,\hat{\mathbf{y}}^P)\in\mathcal{Y}_1\times\cdots\times\mathcal{Y}_P}}{\arg\min}\ &\frac{1}{P}\sum_{i=1}^{P}\Bigg(L_{NN}(\mathbf{g}_{nn}(\mathbf{x}^i_{nn},\mathbf{w}_{nn}),\mathbf{t}^i_{nn})+L_{Val}(E(\cdot,\cdot,\cdot,\mathbf{w}_{sy},\mathbf{w}_{nn}),\mathcal{S}_i)\\&+d(\hat{\mathbf{y}}^i,\mathbf{t}^i_{\mathcal{Y}})\Bigg)+\mathcal{R}(\mathbf{w}_{sy},\mathbf{w}_{nn})\qquad(50)\\\text{s.t.}\quad&\hat{\mathbf{y}}^i\in\underset{\tilde{\mathbf{y}}\in\mathcal{Y}}{\arg\min}\,E(\tilde{\mathbf{y}},\mathbf{x}^i_{sy},\mathbf{x}^i_{nn},\mathbf{w}_{sy},\mathbf{w}_{nn}),\quad\forall i\in\{1,\cdots,P\}.\end{aligned}$$

Regardless of the continuity and curvature properties of the upper and lower level objectives, (50) is equivalent to the following:

$$\begin{aligned}\underset{\substack{(\mathbf{w}_{sy},\mathbf{w}_{nn})\in\mathcal{W}_{sy}\times\mathcal{W}_{nn}\\(\hat{\mathbf{y}}^1,\cdots,\hat{\mathbf{y}}^P)\in\mathcal{Y}_1\times\cdots\times\mathcal{Y}_P}}{\arg\min}\ &\frac{1}{P}\sum_{i=1}^{P}\Bigg(L_{NN}(\mathbf{g}_{nn}(\mathbf{x}^i_{nn},\mathbf{w}_{nn}),\mathbf{t}^i_{nn})+L_{Val}(E(\cdot,\cdot,\cdot,\mathbf{w}_{sy},\mathbf{w}_{nn}),\mathcal{S}_i)\\&+d(\hat{\mathbf{y}}^i,\mathbf{t}^i_{\mathcal{Y}})\Bigg)+\mathcal{R}(\mathbf{w}_{sy},\mathbf{w}_{nn})\qquad(51)\\\text{s.t.}\quad&E(\hat{\mathbf{y}}^i,\mathbf{x}^i_{sy},\mathbf{x}^i_{nn},\mathbf{w}_{sy},\mathbf{w}_{nn})-V_{\mathcal{Y}}(\mathbf{w}_{sy},\mathbf{w}_{nn},\mathcal{S}_i)\leq 0,\quad\forall i\in\{1,\cdots,P\}.\end{aligned}$$

The formulation in (51) is referred to as a *value-function* approach in bilevel optimization literature (V. Outrata 1990; Liu et al. 2021, 2022; Sow et al. 2022; Kwon et al. 2023). Value-function approaches view the bilevel program as a single-level constrained optimization problem by leveraging the value-function as a tight lower bound on the lower-level objective.

The inequality constraints in (51) do not satisfy any of the standard *constraint qualifications* that ensure the feasible set near the optimal point is similar to its linearized approximation (Nocedal and Wright 2006). This raises a challenge for providing theoretical convergence guarantees for constrained optimization techniques. Following a recent line of value-function approaches to bilevel programming (Liu et al. 2021; Sow et al. 2022; Liu et al. 2023), we overcome this challenge by allowing at most an $\iota > 0$ violation in each constraint in (51). With this relaxation, strictly feasible points exist and, for instance, the linear independence constraint qualification (LICQ) can hold.

Another challenge that arises from (51) is that the energy function of NeSy-EBMs is typically non-smooth with respect to the targets and even infinite-valued to represent constraints implicitly. As a result, penalty or augmented Lagrangian functions derived

from (51) are intractable. Therefore, we substitute each instance of the energy function evaluated at the training sample $\mathcal{S}_i$, where $i \in \{1, \cdots, P\}$, and with weights $\mathbf{w}_{sy}$ and $\mathbf{w}_{nn}$ in the constraints of (51) with the following function:

$$M(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \rho) := \inf_{\tilde{\mathbf{y}} \in \mathcal{Y}} \left( E(\tilde{\mathbf{y}}, \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) + \frac{1}{2\rho} \|\tilde{\mathbf{y}} - \hat{\mathbf{y}}^i\|_2^2 \right), \quad (52)$$
$$= V_{conv}(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i; \hat{\mathbf{y}}^i, \frac{1}{2\rho})$$

where $\rho$ is a positive scalar. For convex $E$, (52) is the Moreau envelope of the energy function (Rockafellar 1970; Parikh and Boyd 2013). In general, even for non-convex energy functions, $M$ is finite for all $\mathbf{y} \in \mathcal{Y}$ and it preserves global minimizers and minimum values, i.e.,

$$\mathbf{y}^*(\mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) = \arg\min_{\hat{\mathbf{y}}^i \in \mathcal{Y}} M(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \rho), \quad (53)$$

$$V_{\mathcal{Y}}(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i) = \min_{\hat{\mathbf{y}}^i \in \mathcal{Y}} M(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \rho). \quad (54)$$

When the energy function is a lower semi-continuous convex function, its Moreau envelope is convex, finite, and continuously differentiable, and its gradient with respect to $\hat{\mathbf{y}}^i$ is:

$$\nabla_{\hat{\mathbf{y}}^i} M(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \rho) \quad (55)$$
$$= \frac{1}{\rho} \left( \hat{\mathbf{y}}^i - \arg\min_{\tilde{\mathbf{y}} \in \mathcal{Y}} \left( \rho E(\tilde{\mathbf{y}}, \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) + \frac{1}{2} \|\tilde{\mathbf{y}} - \hat{\mathbf{y}}^i\|_2^2 \right) \right).$$

Convexity is a sufficient but not necessary condition to ensure $M$ is differentiable with respect to $\hat{\mathbf{y}}^i$. See Bonnans and Shapiro (2000) for results regarding the sensitivity of optimal value-functions to perturbations. Further, as $M$ is a value-function, gradients of $M$ with respect to weights are derived using Theorem 10.

We propose the following relaxed and smoothed value-function approach to finding an approximate solution of (50):

$$\arg\min_{\substack{(\mathbf{w}_{sy}, \mathbf{w}_{nn}) \in \mathcal{W}_{sy} \times \mathcal{W}_{nn} \\ (\hat{\mathbf{y}}^1, \cdots, \hat{\mathbf{y}}^P) \in \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_P}} \frac{1}{P} \sum_{i=1}^{P} \Bigg( L_{NN}(\mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn}), \mathbf{t}^i_{nn}) + L_{Val}(E(\cdot, \cdot, \cdot, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathcal{S}_i)$$
$$+ d(\hat{\mathbf{y}}^i, \mathbf{t}^i_{\mathcal{Y}}) \Bigg) + \mathcal{R}(\mathbf{w}_{sy}, \mathbf{w}_{nn}) \quad (56)$$
$$\text{s.t.} \quad M(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \rho) - V_{\mathcal{Y}}(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i) \leq \iota, \quad \forall i \in \{1, \cdots, P\},$$

The formulation (56) is the core of our proposed NeSy-EBM learning framework outlined in Algorithm 1 below. The algorithm proceeds by approximately solving instances of (56) in a sequence defined by a decreasing $\iota$. This is a graduated approach to solving (51) with instances of (56) that are increasingly tighter approximations.

**Algorithm 1** Bilevel Value-Function Optimization for NeSy-EBM Learning

**Require:** Moreau Param.: $\rho$, Starting weights: $(\mathbf{w}_{sy}, \mathbf{w}_{nn}) \in \mathcal{W}_{sy} \times \mathcal{W}_{nn}$

$\hat{\mathbf{y}}^i \leftarrow (\mathbf{t}^i_{\mathcal{Y}}, \arg\min_{\hat{\mathbf{z}} \in \mathcal{Z}^i_{\mathcal{Y}}} E((\mathbf{t}^i_{\mathcal{Y}}, \hat{\mathbf{z}}), \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn})), \quad \forall i = 1, \cdots, P$

$\iota \leftarrow \max_{i \in \{1,\cdots,P\}} M(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \rho) - V_{\mathcal{Y}}(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i)$

**for** $t = 0, 1, 2, \cdots$ **do**

  Find $\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathbf{y}^1, \cdots, \mathbf{y}^P$ minimizing (56) with $\iota$

  **if** Stopping criterion satisified **then**

    Stop with: $\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathbf{y}^1, \cdots, \mathbf{y}^P$

  $\iota \leftarrow \frac{1}{2} \cdot \iota$

We suggest starting points for each $\hat{\mathbf{y}}^i$ to be the corresponding latent inference minimizer and $\iota$ to be the maximum difference in the value-function and the smooth energy function. At this suggested starting point, the supervised loss is initially 0, and the subproblem reduces to minimizing the learning objective without increasing the most violated constraint. Then, the value for $\iota$ is halved every time an approximate solution to the subproblem, (56), is reached. The outer loop of the NeSy-EBM learning framework may be stopped by either watching the progress of a training or validation evaluation metric or by specifying a final value for $\iota$.

Each instance of (56) in Algorithm 1 can be optimized using only first-order gradient-based methods. Specifically, we employ the bound-constrained augmented Lagrangian algorithm, Algorithm 17.4 from Nocedal and Wright (2006), which finds approximate minimizers of the problem's augmented Lagrangian for a fixed setting of the penalty parameters using gradient descent. To simplify notation, let the constraints in (56) be denoted by:

$$c(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \iota) := M(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \rho) - V_{\mathcal{Y}}(\mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}_i) - \iota, \tag{57}$$

for each constraint indexed $i \in \{1, \cdots, P\}$. Moreover, let

$$\mathbf{c}(\mathbf{y}^1, \cdots, \mathbf{y}^P, \mathcal{S}, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \iota) := [c(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \iota)]_{i=1}^P. \tag{58}$$

The augmented Lagrangian function corresponding to (56) introduces a quadratic penalty parameter $\mu$ and $P$ linear penalty parameters $\lambda := [\lambda_i]_{i=1}^P$, as follows:

$$\begin{aligned}
&\mathcal{L}_A(\hat{\mathbf{y}}^1, \cdots, \hat{\mathbf{y}}^P, \mathbf{w}_{sy}, \mathbf{w}_{nn}, \mathcal{S}, \mathbf{s}; \lambda, \mu, \iota) \qquad (59)\\
&:= \frac{1}{P} \sum_{i=1}^{P} \left( L_{NN}(\mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn}), \mathbf{t}^i_{nn}) + L_{Val}(E(\cdot, \cdot, \cdot, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathcal{S}_i) + d(\hat{\mathbf{y}}^i, \mathbf{t}^i_{\mathcal{Y}}) \right)\\
&\quad + \frac{\mu}{2} \sum_{i=1}^{P} \left( c(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \iota) + s_i \right)^2\\
&\quad + \sum_{i=1}^{P} \lambda_i \left( c(\hat{\mathbf{y}}^i, \mathcal{S}_i, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \iota) + s_i \right) + \mathcal{R}(\mathbf{w}_{sy}, \mathbf{w}_{nn}),
\end{aligned}$$

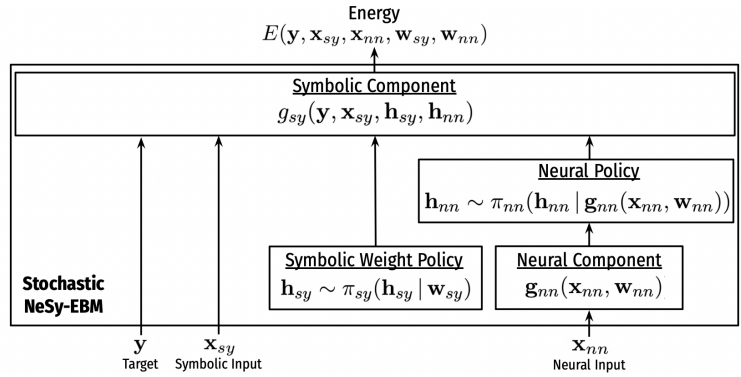



**Figure 5.** A stochastic NeSy-EBM. The symbolic weights and the neural component parameterize stochastic policies. A sample from the policies is drawn to produce arguments of the symbolic component.

where we introduced $P$ slack variables, $\mathbf{s} = [s_i]_{i=1}^{P}$, for each inequality constraint. The bound-constrained augmented Lagrangian algorithm provides a principled method for updating the penalty parameters and ensures fundamental convergence properties of our learning framework. Notably, we have that limit points of the iterate sequence are stationary points of $\|\mathbf{c}(\mathbf{y}^1, \cdots, \mathbf{y}^P, \mathcal{S}, \mathbf{w}_{sy}, \mathbf{w}_{nn}; \iota) + \mathbf{s}\|^2$ when the problem has no feasible points. When the problem is feasible, and LICQ holds at the limits, they are KKT points of (56) (Theorem 17.2 in Nocedal and Wright (2006)). Convergence rates and stronger guarantees are possible by analyzing the structure of the energy function for specific NeSy-EBMs.

The bilevel value-function optimization technique in Algorithm 1 is an end-to-end algorithm for minimizing a general NeSy-EBM learning loss with only first-order value-function gradients. Thus, Algorithm 1 is a more practical and widely applicable technique for NeSy-EBM learning than modular and direct gradient descent methods. The bilevel approach can be employed for a broader class of NeSy-EBMs than direct gradient descent methods and for every motivating application. Moreover, we demonstrate that it can be used to train DSVar and DSPot NeSy-EBMs in our empirical evaluation.

*Stochastic Policy Optimization* Finally, another approach to NeSy-EBM learning that avoids directly computing the energy minimizer's gradients with respect to the weights is to re-formulate NeSy learning as stochastic policy optimization. Fig. 5 shows the modifications to the standard NeSy-EBM framework to create a stochastic NeSy-EBM. The symbolic and neural weights are used to condition a symbolic weight and neural policy, denoted by $\pi_{sy}$ and $\pi_{nn}$, respectively. Samples from the policies replace the symbolic weights and neural output as arguments of the symbolic component. Specifically, given symbolic and neural weights $\mathbf{w}_{sy}$ and $\mathbf{w}_{nn}$ and input features $\mathbf{x}_{nn}^i$ from a training sample

$\mathcal{S}_i \in \mathcal{S}$, $\mathbf{h}_{sy}$ and $\mathbf{h}^i_{nn}$ are random variables with the following conditional distributions:

$$\mathbf{h}_{sy} \sim \pi_{sy}(\mathbf{h}_{sy} \,|\, \mathbf{w}_{sy}), \tag{60}$$
$$\mathbf{h}^i_{nn} \sim \pi_{nn}(\mathbf{h}^i_{nn} \,|\, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})). \tag{61}$$

Moreover, the random variables $\mathbf{h}_{sy}$ and $\mathbf{h}^i_{nn}$ are modeled independently, thus the conditional joint distribution, denoted by $\pi$, is:

$$\pi(\mathbf{h}_{sy}, \mathbf{h}^i_{nn} \,|\, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})) := \pi_{sy}(\mathbf{h}_{sy} \,|\, \mathbf{w}_{sy}) \cdot \pi_{nn}(\mathbf{h}^i_{nn} \,|\, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})) \tag{62}$$

The stochastic NeSy-EBM energy is the symbolic component evaluated at a sample from the joint distribution above:

$$E(\mathbf{y}, \mathbf{x}^i_{sy}, \mathbf{x}^i_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) := g_{sy}(\mathbf{y}, \mathbf{x}^i_{sy}, \mathbf{h}_{sy}, \mathbf{h}^i_{nn}) \tag{63}$$

The NeSy-EBM energy and all of the NeSy-EBM per-sample loss functionals discussed in Section 5 are, therefore, random variables with distributions that are defined by $\pi$. Under the stochastic policy optimization framework, loss functionals are generally denoted by the function $J^i$ for each $i \in \{1, \cdots, P\}$ such that:

$$J^i(g_{sy}(\cdot, \mathbf{x}^i_{sy}, \mathbf{h}_{sy}, \mathbf{h}^i_{nn}), \mathcal{S}_i) := L^i(E(\cdot, \cdot, \cdot, \mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathcal{S}_i) \tag{64}$$

Learning is minimizing the expected value of the stochastic loss functional and is formulated as:

$$\underset{(\mathbf{w}_{sy}, \mathbf{w}_{nn}) \in \mathcal{W}_{sy} \times \mathcal{W}_{nn}}{\arg\min} \frac{1}{P} \sum_{i=1}^{P} \mathbb{E}_\pi \left[ J^i(g_{sy}(\cdot, \mathbf{x}^i_{sy}, \mathbf{h}_{sy}, \mathbf{h}^i_{nn}), \mathcal{S}_i) \right] + \mathcal{R}(\mathbf{w}_{sy}, \mathbf{w}_{nn}), \tag{65}$$

where $\mathbb{E}_\pi$ is the expectation over the joint distribution $\pi$.

We apply gradient-based learning algorithms to find an approximate solution to (65). The policy gradient theorem (Williams 1992; Sutton et al. 1999; Sutton and Barto 2018) yields the following expression for the gradients of the expected value of a loss functional:

$$\nabla_{\mathbf{w}_{nn}} \mathbb{E}_\pi \left[ J^i(g_{sy}(\cdot, \mathbf{x}^i_{sy}, \mathbf{h}_{sy}, \mathbf{h}^i_{nn}), \mathcal{S}_i) \right] \tag{66}$$
$$= \mathbb{E}_\pi \left[ J^i(g_{sy}(\cdot, \mathbf{x}^i_{sy}, \mathbf{h}_{sy}, \mathbf{h}^i_{nn}), \mathcal{S}_i) \cdot \nabla_{\mathbf{w}_{nn}} \log \pi(\mathbf{h}_{sy}, \mathbf{h}^i_{nn} \,|\, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})) \right].$$
$$\nabla_{\mathbf{w}_{sy}} \mathbb{E}_\pi \left[ J^i(g_{sy}(\cdot, \mathbf{x}^i_{sy}, \mathbf{h}_{sy}, \mathbf{h}^i_{nn}), \mathcal{S}_i) \right] \tag{67}$$
$$= \mathbb{E}_\pi \left[ J^i(g_{sy}(\cdot, \mathbf{x}^i_{sy}, \mathbf{h}_{sy}, \mathbf{h}^i_{nn}), \mathcal{S}_i) \cdot \nabla_{\mathbf{w}_{sy}} \log \pi(\mathbf{h}_{sy}, \mathbf{h}^i_{nn} \,|\, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})) \right].$$

The expression for the gradient of the expected loss functional above motivates a family of gradient estimators. Notably, the REINFORCE gradient estimator for NeSy-EBM learning

is:

$$\nabla_{\mathbf{w}_{nn}} \mathbb{E}_{\pi} \left[ J^i(g_{sy}(\cdot, \mathbf{x}^i_{sy}, \mathbf{h}_{sy}, \mathbf{h}^i_{nn}), \mathcal{S}_i) \right] \tag{68}$$
$$\approx \frac{1}{N} \sum_{k=1}^{N} \left( J^i(g_{sy}(\cdot, \mathbf{x}^i_{sy}, \mathbf{h}^{(k)}_{sy}, \mathbf{h}^{i(k)}_{nn}), \mathcal{S}_i) \nabla_{\mathbf{w}_{nn}} \log \pi(\mathbf{h}^{(k)}_{sy}, \mathbf{h}^{i(k)}_{nn} \mid \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})) \right),$$

$$\nabla_{\mathbf{w}_{sy}} \mathbb{E}_{\pi} \left[ J^i(g_{sy}(\cdot, \mathbf{x}^i_{sy}, \mathbf{h}_{sy}, \mathbf{h}^i_{nn}), \mathcal{S}_i) \right] \tag{69}$$
$$\approx \frac{1}{N} \sum_{k=1}^{N} \left( J^i(g_{sy}(\cdot, \mathbf{x}^i_{sy}, \mathbf{h}^{(k)}_{sy}, \mathbf{h}^{i(k)}_{nn}), \mathcal{S}_i) \nabla_{\mathbf{w}_{sy}} \log \pi(\mathbf{h}^{(k)}_{sy}, \mathbf{h}^{i(k)}_{nn} \mid \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}^i_{nn}, \mathbf{w}_{nn})) \right),$$

where each $\mathbf{h}^{(k)}_{sy}$ and $\mathbf{h}^{i(k)}_{nn}$ for $k \in \{1, \cdots, N\}$ is an independent sample of the random variables.

Stochastic policy optimization techniques are broadly applicable for end-to-end training of NeSy-EBMs because they do require differentiation through the neural-symbolic interface and the symbolic inference process. Moreover, they can be used for every modeling paradigm. The tradeoff with the stochastic policy approach, however, is the high variance in the sample estimates for the policy gradient. This is a common challenge in policy optimization that becomes more prominent with increasing dimensionality of the policy output space (Sutton and Barto 2018). Thus, learning with a stochastic policy optimization approach may take significantly more iterations to converge compared to the other presented techniques.

## 6 Empirical Analysis

In this section, we perform an empirical analysis of the NeSy-EBM modeling paradigms and learning algorithms presented in this work using the NeuPSL system introduced in Section 4. Our experiments are designed to investigate the following four research questions:

- RQ1: Can the NeSy-EBM framework enhance the accuracy and reasoning capabilities of deep learning models?
- RQ2: Can the value-function gradients provided in Theorem 10 be used as a reliable descent direction for value-based learning losses?
- RQ3: Can symbolic constraints be used to train a deep learning model with partially labeled data?
- RQ4: What are the prediction performance and runtime tradeoffs among the presented modular, value-based, and minimizer-based learning approaches?

Our empirical analysis is organized into four subsections. First, in Section 6, we introduce the neural-symbolic datasets and models used in the experiments. In Section 6, we study the application of NeSy-EBMs for constraint satisfaction and joint reasoning. In Section 6, we evaluate the performance of modular learning and the performance

and empirical convergence properties of the value-based, bilevel, and stochastic policy optimization learning algorithms presented in Section 5 for fine-tuning and few-shot learning. Finally, in Section 6, we analyze the effectiveness of the NeSy-EBM framework for training a neural component in a semi-supervised setting. All code and data for reproducing our empirical analysis are available at https://github.com/linqs/dickens-arxiv24.

### Datasets and Models

This subsection introduces the NeSy datasets and models, which will be utilized throughout the empirical analysis. Moreover, any modifications made to answer specific research questions will be described in the following subsections. Additional details on the architectures of both the neural and symbolic components are available at https://github.com/linqs/dickens-arxiv24.

- **MNIST-Add-k Dataset:** MNIST-Add-$k$ is a canonical NeSy dataset introduced by Manhaeve et al. (2021a) where models must determine the sum of each pair of digits from two lists of MNIST images. An MNIST-Add$k$ equation consists of two lists of $k > 0$ MNIST images. For instance, $[\,\image_ref{1}\,] + [\,\image_ref{2}\,] = \mathbf{8}$ is an MNIST-Add1 equation, and $[\,\image_ref{3}, \image_ref{4}\,] + [\,\image_ref{5}, \image_ref{6}\,] = \mathbf{41}$ is an MNIST-Add2 equation.

  **Evaluation:** For all experiments, we evaluate models over 5 splits of the low-data setting proposed by Manhaeve et al. (2021a) with 600 total images for training and $1,000$ images each for validation and test. Prediction performance in this setting is measured by the accuracy of the image classifications and the inferred sums. Constraint satisfaction consistency in this setting is the proportion of predictions that satisfy the semantics of addition.

  **Baseline Architecture:** The baseline neural architecture for all MNIST-Add$k$ datasets is a ResNet18 convolutional neural network backbone (He et al. 2016) with a 2-layer multi-layer perceptron (MLP) prediction head. The baseline is trained and applied as a digit classifier. Further, to allow the baseline to leverage the unlabeled training data in the semi-supervised settings, the digit classifier backbone is pre-trained using the SimCLR self-supervised learning framework (Chen et al. 2020). Augmentations are used to obtain positive pairs for the contrastive pre-training process.

  **NeSy-EBM Architecture:** The NeSy-EBM architecture is a composition of the baseline digit classifier and a symbolic component created with NeuPSL that encodes the semantics of addition. The target variables of the symbolic component are the labels of the MNIST digits and their sum. The neural classification is used as a prior for the digit labels.

- **Visual-Sudoku Dataset:** Visual-Sudoku, first introduced by Wang et al. (2019), is a dataset containing a collection of $9 \times 9$ Sudoku puzzles constructed from MNIST images. In each puzzle, 30 cells are filled with MNIST images and are referred to as *clues*. The remaining cells are empty. The task is to correctly classify all clues

and fill in the empty cells with digits that satisfy the rules of Sudoku: no repeated digits in any row, column, or box.

**Evaluation:** For all experiments, results are reported across 5 splits with 20 puzzles for training and 100 puzzles each for validation and test. There is an equal number of MNIST images (600) in the training datasets for Visual-Sudoku and MNIST-Add-k. Prediction performance in this setting is measured by the accuracy of the image classifications. Constraint satisfaction consistency in this setting is the proportion of predictions that satisfy the rules of Sudoku.

**Baseline Architecture:** The baseline neural architecture for Visual-Sudoku is the same as that of the MNIST-Add$k$.

**NeSy-EBM Architecture:** The NeSy-EBM architecture is a composition of the baseline digit classifier and a symbolic component created with NeuPSL that encodes the rules of Sudoku. The target variables of the symbolic component are the labels of the clues and the empty cells. The neural classification is used as a prior for the clues.

- **Pathfinding Dataset:** Pathfinding is a NeSy dataset introduced by Vlastelica et al. (2020) consisting of 12000 randomly generated images of terrain maps from the Warcraft II tileset. The images are partitioned into $12 \times 12$ grids where each vertex represents a terrain with a cost. The task is to find the lowest cost path from the top left to the bottom right corner of each image.

  **Evaluation:** For all experiments, results are reported over 5 splits generated by partitioning the images into sets of $10,000$ for training, $1,000$ for validation, and $1,000$ for testing. Prediction performance in this setting is measured by the proportion of valid predicted paths, i.e., continuous, and that have a minimum cost. Constraint satisfaction continuity in this setting is measured by the proportion of predictions with a continuous predicted path.

  **Baseline Architecture:** The baseline neural architecture for the Pathfinding dataset is a ResNet18 convolutional neural network. The input of the ResNet18 path-finder baseline is the full Warcraft II map, and the output is the predicted shortest path. The model is trained using the labeled paths from the training data set.

  **NeSy-EBM Architecture:** The NeSy-EBM architecture is a composition of the baseline path-finder and a symbolic component created with NeuPSL that encodes end-points and continuity constraints, i.e., the path from the top left corner of the map to the bottom right corner must be continuous. The target variables of the symbolic component are variables indicating whether a vertex of the map grid is on the path. The neural classification is used as a prior for the path, and the symbolic component finds a valid path near the neural prediction.

- **Citeseer and Cora Dataset:** Citeseer and Cora are two widely studied citation network node classification datasets first introduced by Sen et al. (2008). Citeseer consists of $3,327$ scientific publications classified into one of 6 topics, while Cora contains $2,708$ scientific publications classified into one of 7 topics.

**Evaluation:** For all experiments, we evaluate models over 5 randomly sampled splits using 20 examples of each topic for training, 200 of the nodes for validation, and 1000 nodes for testing. Prediction performance in this setting is measured by the categorical accuracy of a paper label.

**Baseline Architecture:** The baseline neural architecture for the Citation network settings is a Simple Graph Convolutional Network (SGC) (Wu et al. 2019). SGCs are graph convolutional networks with linear activations in the hidden layers to reduce computational complexity. The SGC neural baseline uses bag-of-words feature vectors associated with each paper as node features and citations as bi-directional edges. Then, a MLP is trained to predict the topic label given the SGC-transformed features.

**NeSy-EBM Architecture:** The NeSy-EBM architecture is a composition of the baseline SGC and a symbolic component created with NeuPSL that encodes the homophilic structure of the citation network, i.e., two papers connected in the network are more likely to have the same label. Target variables indicate the degree to which a paper has a particular topic. The neural classification is used as a prior for the labels of the nodes, and the symbolic component propagates this knowledge to its neighbors.

- **RoadR Dataset:** RoadR is an extension of the ROAD (Road event Awareness Dataset) dataset, initially introduced by Singh et al. (2021). The ROAD dataset was developed to evaluate the situational awareness of autonomous vehicles in various road environments, weather conditions, and times of day. It contains 22 videos, $122k$ labeled frames, $560k$ bounding boxes, and a total of $1.7M$ labels, which include $560k$ agents, $640k$ actions, and $499k$ locations. RoadR builds upon this by adding 243 logical requirements that must be satisfied, further enhancing its utility for testing autonomous vehicles. For instance, a traffic light should never be simultaneously predicted as red and green.

  **Evaluation:** For all experiments, we evaluate models with 15 videos for training and 3 videos for testing. Prediction performance in this setting is measured by the matching boxes using Intersection over Union (IoU) and then multi-class f1. Constraint satisfaction consistency in this setting is the proportion of frame predictions with no constraint violations.

  **Baseline Architecture:** The baseline neural architecture for the RoadR dataset is a DEtection TRansformer (DETR) model with a ResNet50 backbone (Carion et al. 2020). The baseline is trained and applied to detect objects in a frame, along with a multi-label classification for its class labels (e.g., car, red, traffic light, etc.).

  **NeSy-EBM Architecture:** The NeSy-EBM architecture is a composition of the baseline object detector and classifier and a symbolic component created with NeuPSL that encodes the logical requirements. The target variables are the classification labels of a bounding box. The neural classification is used as both the bounding box creation and a prior on the labels that the symbolic component uses as a starting point to find a valid solution to the constraints.

- **Logical-Deduction** is a multiple-choice question-answering dataset introduced by Srivastava et al. (2022). These questions require deducing the order of a sequence of objects given a natural language description and then answering a multiple-choice question about that ordering.

  **Evaluation:** We report results for a single test set of 300 deduction problems, with a prompt containing two examples. Prediction performance in this setting is measured by the accuracy of the predicted multiple-choice answer.

  **Baseline Architecture:** The baseline neural architecture for the Logical-Deduction dataset is the models presented in Pan et al. (2023) on GPT-3.5-turbo and GPT-4 OpenAI (2024). Each model is run using *Standard* and *Chain-of-Thought (CoT)* (Wei et al. 2022) prompting.

  **NeSy-EBM Architecture:** The NeSy-EBM architecture is a composition of the baseline LLM that is being prompted to create the constraints within the symbolic program. Symbolic inference is then performed, and the output is returned to the LLM for final evaluation. In this sense, the NeSy-EBM writes a program to perform reasoning rather than depending on the language model to reason independently.

### *Constraint Satisfaction and Joint Reasoning*

We begin our experimental evaluation by exploring the advantages of employing NeSy-EBMs for performing constraint satisfaction and joint reasoning, which is relevant to answering research question RQ1. We employ a modular training approach to set up these experiments to obtain weights for our models' neural and symbolic components. Specifically, neural components undergo training using the complete training dataset for supervision, and symbolic weights are trained using a simple random grid search. After this modular training phase, NeSy-EBM inference is carried out to predict binary, $0, 1$, valued target variables that align with established domain knowledge and logical reasoning. For this reason, *DSPar* NeSy-EBMs are used for MNIST-Add-k, Visual-Sudoku, Pathfinding, RoadR, Citeseer, and Cora, and a *DSPot* is used for Logical Deduction.

To investigate constraint satisfaction and joint reasoning, we use the dataset settings outlined in Section 6 for Visual-Sudoku, Pathfinding, RoadR, Citeseer, Cora, and Logic Deduction. Additionally, we introduce the following variant of the MNIST-Add-$k$ dataset.

- **MNIST-Add**$k$: The $k = 1, 2, 4$ MNIST-Add$k$ datasets with the sums of the MNIST-Add-$k$ equations available as observations during inference. Prediction performance is measured by the accuracy of the image classifications.

The MNIST-Add-$k$ modification allows the NeSy-EBM to use the semantics of addition and the sum observation to form constraints to correct the neural component predictions. For instance, consider the MNIST-Add-1 equation $[\,\mathbf{3}\,] + [\,\mathbf{5}\,] = \mathbf{8}$. If the neural component incorrectly classifies the first MNIST image, $\mathbf{3}$, as an $8$ with low confidence but correctly classifies the second MNIST image, $\mathbf{5}$, as a $5$ with high confidence, then it can use the sum label, $8$, to correct the first digit label.

Tables 1 to 3 report the prediction performance and constraint satisfaction consistency of a neural baseline and NeuPSL model on the MNIST-Add$k$, Visual-Sudoku, Pathfinding,

**Table 1.** Digit accuracy and constraint satisfaction consistency of the ResNet18 and NeuPSL models on the MNIST-Add-$k$ and Visual-Sudoku datasets.

| | ResNet18 | | NeuPSL | |
|---|---|---|---|---|
| | **Digit Acc.** | **Consistency** | **Digit Acc.** | **Consistency** |
| *MNIST-Add1* | $97.60 \pm 0.55$ | $93.04 \pm 1.33$ | $\mathbf{99.80 \pm 0.14}$ | $\mathbf{100.0 \pm 0.00}$ |
| *MNIST-Add2* | | $86.56 \pm 2.72$ | $\mathbf{99.68 \pm 0.22}$ | $\mathbf{100.0 \pm 0.00}$ |
| *MNIST-Add4* | | $75.04 \pm 4.81$ | $\mathbf{99.72 \pm 0.29}$ | $\mathbf{100.0 \pm 0.00}$ |
| *Visual-Sudoku* | | $70.20 \pm 2.17$ | $\mathbf{99.37 \pm 0.11}$ | $\mathbf{100.0 \pm 0.00}$ |

**Table 2.** Accuracy of finding a minimum cost path (Min. Cost Acc.) and consistency in satisfying continuity constraints (Continuity) of the ResNet18 and NeuPSL models on the Pathfinding dataset.

| | ResNet18 | | NeuPSL | |
|---|---|---|---|---|
| | **Min. Cost Acc.** | **Continuity** | **Min. Cost Acc.** | **Continuity** |
| *Pathfinding* | $80.12 \pm 22.44$ | $84.80 \pm 17.11$ | $\mathbf{90.02 \pm 11.70}$ | $\mathbf{100.0 \pm 0.00}$ |

**Table 3.** Object detection F1 and constraint satisfaction consistency of the DETR and NeuPSL models on the RoadR dataset.

| | DETR | | NeuPSL | |
|---|---|---|---|---|
| | **F1** | **Consistency** | **F1** | **Consistency** |
| *RoadR* | 0.457 | 27.5 | **0.461** | **100.0** |

and RoadR datasets, respectively. Across all settings, the baseline neural models frequently violate constraints within the test dataset. Further, the frequency of these violations increases with the complexity of the constraints. This behavior is best illustrated in the MNIST-Add$k$ datasets, where consistency decreases as the number of digits, $k$, increases. This decline can be attributed to the baseline ResNet18 model treating each digit prediction independently and thus failing to account for the dependencies from the sum relation. Moreover, in the RoadR experiment, the DETR baseline adheres to road event constraints only $27.5\%$ of the time. On the other hand, NeuPSL always satisfies the problem constraints in the MNIST-Add$k$, Visual-Sudoku, Pathfinding, and RoadR datasets, achieving $100\%$ consistency. This is because the DSPar NeSy-EBM models used in these experiments can enforce constraints on all target variables. This allows the NeSy-EBM models to leverage the structural relations inherent in the constraints to infer target variables and jointly improve prediction accuracy. Prediction performance gains from constraint satisfaction and joint reasoning are possible when the neural component accurately quantifies its confidence. The symbolic component uses the confidence of the neural component and the constraints together to correct the neural model's erroneous predictions. This observation motivates an exciting avenue of future research: exploring

whether calibrating the confidence of the neural component can further improve the structured prediction and joint reasoning capabilities of NeSy-EBMs.

**Table 4.** Node classification accuracy of the SGC and NeuPSL models on the Citeseer and Cora datasets.

| | **SGC** | **NeuPSL** |
|---|---|---|
| *Citeseer* | $65.14 \pm 2.96$ | $\mathbf{66.52 \pm 3.26}$ |
| *Cora* | $80.90 \pm 1.54$ | $\mathbf{81.82 \pm 1.73}$ |

**Table 5.** Comparison of accuracy in answering logical deduction questions using two large language models, GPT-3.5-turbo and GPT-4 OpenAI (2024), across three methods: Standard, Chain of Thought (CoT), and NeuPSL.

| | **LLM** | **Standard** | **CoT** | **NeuPSL** |
|---|---|---|---|---|
| *Logical Deduction* | *GPT-3.5-turbo* | 40.00 | 42.33 | **70.33** |
| | *GPT-4* | 71.33 | 75.25 | **90.67** |

Unlike MNIST-Add$k$, Visual-Sudoku, Pathfinding, and RoadR, which have hard constraints on the target variables, the citation network datasets showcase the capacity of NeSy-EBMs to perform joint reasoning with constraints and dependencies that are not strictly adhered to. For Citeseer and Cora, NeuPSL enhances prediction accuracy by leveraging the homophilic structure of the citation networks, i.e., papers that are linked tend to share topic labels. Similarly, in the question-answering logical deduction problem, NeuPSL uses an LLM to generate rules representing the dependencies described in natural language. Although the LLM may sometimes fail to generate accurate rules, NeuPSL will consistently use the rules for logical reasoning.

Tables 4 and 5 report the baseline and NeuPSL NeSy-EBM prediction performance on the citation network node classification and logical deduction datasets, respectively. In all instances, NeuPSL outperforms the baseline. The performance gain from NeuPSL in the citation network experiments is verified to be statistically significant with a paired t-test and p-value less than $0.05$. Further, in the Logical Deduction setting, NeuPSL obtains a $15\%$ improvement over the LLM. This performance gain is achieved despite the fact that the LLM neural component in NeuPSL could produce invalid syntax or an infeasible set of logical constraints. The LLM was able to produce valid programs $89.0\%$ and $98.7\%$ of the time with gpt-3.5-turbo and gpt-4, respectively. This observation motivates a promising avenue of future research in employing self-refinement approaches similar to that of Pan et al. (2023) to attempt to correct the infeasible programs and further improve LLM reasoning capabilities.

The results in this section are evidence for the affirmative answer to the research question RQ1. Across diverse datasets—MNIST-Add$k$, Visual-Sudoku, Pathfinding, and RoadR—the baseline neural models frequently violate constraints, with violations increasing in complexity. For instance, the baseline model's performance on MNIST-Add$k$

deteriorates as the number of digits increases, and in RoadR, it adheres to constraints only 27.5% of the time. In stark contrast, the NeSy-EBM framework, incorporating symbolic components to enforce constraints, consistently achieves 100% constraint satisfaction and improves prediction accuracy. Additionally, in citation network and logical deduction scenarios, NeSy-EBM facilitates joint reasoning, leveraging the inherent structure and logical coherence, leading to significant performance gains.

## *NeSy-EBM Learning*

Next, we investigate NeSy-EBM learning, focusing on answering research questions RQ2, RQ3, and RQ4. Our analysis is divided into two parts. First, we study the performance of modular learning NeSy-EBM methods in Section 6. Second, we examine the performance and empirical convergence properties of end-to-end gradient-based NeSy-EBM learning algorithms in Section 6. All models within this subsection use the *DSPar* modeling paradigm.

*Modular NeSy-EBM Learning* In our modular learning experiments, the neural components are first trained using supervised neural losses and are then frozen. The symbolic component is trained using either a minimizer-based or value-based loss. Specifically, we compare the prediction performance of two value-based losses, Energy and Structure Perceptron (SP), and two minimizer-based losses, Mean Square Error (MSE), and Binary Cross Entropy (BCE). The modular learning experiments are conducted on the seven datasets listed in Table 6 below. The table overviews each dataset's inference task and the corresponding prediction performance metric. Additional details on these datasets are provided in Appendix E.

**Table 6.** Datasets used for modular experimental evaluations.

| Dataset | Task | Perf. Metric |
|---|---|---|
| Debate (Hasan and Ng 2013) | Stance Class. | AUROC |
| 4Forums (Walker et al. 2012) | Stance Class. | AUROC |
| Epinions (Richardson et al. 2003) | Link Pred. | AUROC |
| DDI (Wishart et al. 2006) | Link Pred. | AUROC |
| Yelp (Kouki et al. 2015) | Regression | MAE |
| Citeseer (Sen et al. 2008) | Node Class. | Accuracy |
| Cora (Sen et al. 2008) | Node Class. | Accuracy |

Table 7 reports the prediction performance achieved by each of the four learning techniques across the seven modular datasets. Models trained with bilevel-based losses consistently achieve better average predictive performance than those trained with value-based losses. Notably, on the Cora dataset, the NeuPSL model trained with the BCE loss achieved a remarkable improvement of over six percentage points compared to the SP loss, which was the better-performing value-based loss. The models trained with the Energy and SP loss suffered from a collapsed solution, i.e., symbolic parameters giving nearly equal energy to all settings of the target variables.

**Table 7.** Prediction performance of HL-MRF models trained on value and minimizer-based losses.

| | Value-Based | | Bilevel | |
|---|---|---|---|---|
| | **Energy** | **SP** | **MSE** | **BCE** |
| **Debate** | $64.76 \pm 9.54$ | $64.68 \pm 11.05$ | $\mathbf{65.33 \pm 11.98}$ | $64.83 \pm 9.70$ |
| **4Forums** | $62.96 \pm 6.11$ | $63.15 \pm 6.40$ | $64.22 \pm 6.41$ | $\mathbf{64.85 \pm 6.01}$ |
| **Epinions** | $78.96 \pm 2.29$ | $79.85 \pm 1.62$ | $\mathbf{81.18 \pm 2.21}$ | $80.89 \pm 2.32$ |
| **Citeseer** | $70.29 \pm 1.54$ | $70.92 \pm 1.33$ | $71.22 \pm 1.56$ | $\mathbf{71.94 \pm 1.17}$ |
| **Cora** | $54.30 \pm 1.74$ | $74.16 \pm 2.32$ | $81.05 \pm 1.41$ | $\mathbf{81.07 \pm 1.31}$ |
| **DDI** | $94.54 \pm 0.00$ | $94.61 \pm 0.00$ | $94.70 \pm 0.00$ | $\mathbf{95.08 \pm 0.00}$ |
| **Yelp** | $18.11 \pm 0.34$ | $18.57 \pm 0.66$ | $18.14 \pm 0.36$ | $\mathbf{17.93 \pm 0.50}$ |

*End-to-End NeSy-EBM Learning* This subsection analyzes the performance and empirical convergence properties of the three following end-to-end gradient-based NeSy-EBM learning algorithms.

- **Energy**: Gradient descent on the value-based energy loss.
- **Bilevel**: The bilevel value-function optimization for NeSy-EBM learning algorithm. For all datasets, binary cross-entropy is the minimizer-based loss, and the energy loss is the value-based loss.
- **IndeCateR**: The stochastic policy optimization algorithmic framework with the Independent Categorical REINFORCE gradient estimator (De Smet et al. 2023). The evaluation metric of the dataset is directly applied as the learning loss.

Theorem 10 in Section 5 is used to compute the learning gradients with respect to the neural output and symbolic weights for the Energy and Bilevel algorithms. Similarly, the IndeCateR estimate is used to compute the learning gradients with respect to the neural output and symbolic weights for stochastic policy optimization. Then, gradients with respect to the neural parameters are found via backpropagation for all methods. The neural parameters are updated via AdamW (Loshchilov and Hutter 2019), and the symbolic parameters are updated using gradient descent with a fixed step size. Additional details on the hardware and hyperparameters settings of the learning algorithms are provided in Appendix E.

To investigate the performance of our NeSy-EBM learning algorithms, we use the dataset settings outlined in Section 6 for Citeseer and Cora. Additionally, we introduce the following variant of the MNIST-Add$k$, Visual-Sudoku, and Pathfinding datasets:

- **MNIST-Add$k$**: The $k = 1, 2$ MNIST-Add$k$ datasets with no digit supervision, i.e., parameters are learned only from the addition relations.
- **Visual-Sudoku**: A few-shot setting with 5 labeled examples of each of the 9 possible classes available for training. The remaining images in the training data are unlabeled, and the model must primarily rely on the Sudoku rules for learning.

- **Pathfinding**: A limited supervision setting where only $10\%$ of the training data is labeled, and the remaining training data is unlabeled. Specifically, only $5\%$ of the map vertices are observed to be on or off the labeled minimum cost path. In other words, supervision is distributed across maps, and the minimum cost paths for a map are only partially observed.

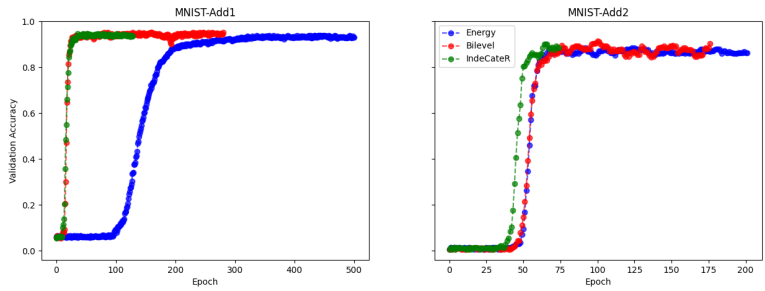
**Table 8.** The average and standard deviation of the prediction performance of NeuPSL NeSy-EBMs trained using gradient-based learning algorithms on 7 datasets.

| | NeuPSL | | |
|---|---|---|---|
| | **Energy** | **IndeCateR** | **Bilevel** |
| *MNIST-Add1* | $93.80 \pm 1.12$ | $94.52 \pm 0.99$ | $\mathbf{94.92 \pm 1.40}$ |
| *MNIST-Add2* | $87.92 \pm 1.63$ | $86.88 \pm 1.82$ | $\mathbf{89.36 \pm 1.54}$ |
| *Visual-Sudoku* | $\mathbf{98.12 \pm 0.37}$ | TIMEOUT | $98.10 \pm 0.19$ |
| *Path-Finding* | $22.53 \pm 0.75$ | TIMEOUT | $\mathbf{22.85 \pm 1.33}$ |
| *Citeseer* | $67.04 \pm 1.82$ | TIMEOUT | $\mathbf{67.96 \pm 1.11}$ |
| *Cora* | $80.40 \pm 0.74$ | TIMEOUT | $\mathbf{81.88 \pm 0.65}$ |

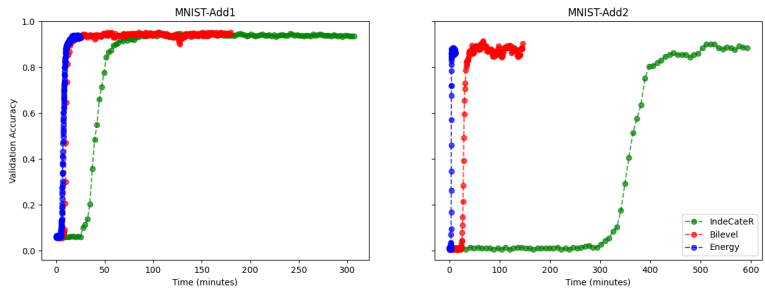
Table 8 presents the average and standard deviation of the prediction performance for the symbolic component of the NeuPSL NeSy-EBM model across the six datasets examined in this subsection. In five of the six datasets, the Bilevel learning algorithm achieves the best results. Notably, in MNIST-Add1, IndeCateR's performance was comparable to Bilevel's. However, as the complexity of the target variable constraints increased, IndeCateR's performance deteriorated, exemplified by poor results in MNIST-Add2 and failures to find viable solutions within the allotted time in the other datasets.

While Energy generally underperformed compared to Bilevel across most settings, it was the fastest in execution time. For instance, Fig. 6 plots the validation image classification accuracy of the MNIST-Add1 and MNIST-Add2 NeuPSL NeSy-EBMs trained with the Energy, IndeCater, and Bilevel learning algorithms versus the training epoch and wall-clock time for a single fold. The Bilevel and IndeCateR algorithms reach higher validation performance levels than the Energy algorithm on both MNIST-Add$k$ datasets for the reported fold. This pattern is consistent with the average prediction performance results reported in Table 8 for MNIST-Add1. For the MNIST-Add2 dataset, on the other hand, the IndeCateR algorithm was timed out after 10 hours of training rather than allowing it to fully converge, which explains the drop in the relatively lower average test performance results in Table 8. Surprisingly, the IndeCateR algorithm has the best empirical rate of improvement with respect to training epochs on both datasets; the next best is Bilevel, and finally, Energy. However, the IndeCateR algorithm's per-iteration cost counteracts its advantage, and it has a significantly slower rate of improvement with respect to wall-clock time. On the other end of the spectrum, Energy has the slowest rate of prediction performance improvement, but its per iteration cost is low enough that it converges the fastest with respect to wall-clock time. The Bilevel algorithm balances the strengths of the two algorithms. It has a lower per-iteration cost because it only uses

**Figure 6.** Validation image classification accuracy versus training (a) epoch and (b) time in minutes for NeuPSL models trained with the Energy, IndeCateR, and Bilevel NeSy-EBM learning algorithms.




**(a)**




**(b)**

value-function gradients and optimizes a minimizer-based loss. The convergence results in Fig. 6 motivate future work on training pipelines that pre-train with a value-based loss and fine-tune with a more expensive minimizer-based loss to achieve the fastest training time and best final prediction performance.

## *Semi-Supervision*

In this set of experiments, we investigate the effectiveness of the NeSy-EBM framework in training a deep learning model in a semi-supervised setting. This experiment aims to further investigate research questions RQ3 and RQ4. Specifically, we compare the prediction performance of a neural baseline trained solely with a supervised neural loss to that of a NeuPSL model's neural component (with the same architecture) trained using an end-to-end NeSy-EBM loss. In both cases, only a subset of the training set labels is available to the neural component. To enhance neural performance with a structured loss, the MNIST-Addk and Visual-Sudoku models in this subsection employ the *DSVar* modeling paradigm due to its simplicity and speed, while Pathfinding, Citeseer, and Cora

models use the *DSPar* modeling paradigm. We use the following variants of four datasets for our experiments.

- **MNIST-Add$k$**: The $k = 1, 2$ MNIST-Add$k$ datasets with the proportion of image class labels available in the training data varying over $\{1.0, 0.5, 0.1, 0.05\}$. Prediction performance in this subsection is measured by the accuracy of the image classifications.
- **Visual-Sudoku**: The proportion of image class labels available in the training data varies over $\{1.0, 0.5, 0.1, 0.05\}$.
- **Pathfinding**: Supervision is distributed across all training maps, so the shortest paths in the training data are only partially observed. The proportion of vertex labels available in the training data varies over $\{1.0, 0.5, 0.1\}$.
- **Citeseer and Cora**: The proportion of paper topic labels available in the training data varies over $\{1.0, 0.5, 0.1\}$.

The Bilevel learning algorithm is applied to train the NeSy-EBM neural components for the MNIST-Add$k$, Citeseer, and Cora datasets. The Energy learning algorithm is applied to train the NeSy-EBM neural components for the Visual-Sudoku and Pathfinding datasets. Additional details on the hardware and hyperparameter settings of the learning algorithms are provided in Appendix E.

**Table 9.** Digit accuracy of the ResNet18 models trained with varying levels of supervision.

| | | ResNet18 | |
|---|---|---|---|
| | **Labeled** | **Supervised** | **NeuPSL Semi-Supervised** |
| *MNIST-Add1* | 1.00 | $\mathbf{97.84 \pm 0.23}$ | $97.40 \pm 0.51$ |
| | 0.50 | $\mathbf{97.42 \pm 0.30}$ | $97.02 \pm 0.65$ |
| | 0.10 | $93.05 \pm 0.69$ | $\mathbf{96.78 \pm 0.80}$ |
| | 0.05 | $75.35 \pm 0.33$ | $\mathbf{96.82 \pm 0.72}$ |
| *MNIST-Add2* | 1.00 | $\mathbf{97.84 \pm 0.23}$ | $97.22 \pm 0.19$ |
| | 0.50 | $\mathbf{97.42 \pm 0.30}$ | $96.84 \pm 0.42$ |
| | 0.10 | $93.05 \pm 0.69$ | $\mathbf{95.14 \pm 1.21}$ |
| | 0.05 | $75.35 \pm 0.33$ | $\mathbf{95.90 \pm 0.43}$ |
| *Visual-Sudoku* | 1.00 | $97.84 \pm 0.23$ | $\mathbf{97.89 \pm 0.15}$ |
| | 0.50 | $\mathbf{97.42 \pm 0.30}$ | $97.26 \pm 0.70$ |
| | 0.10 | $93.05 \pm 0.69$ | $\mathbf{96.82 \pm 0.32}$ |
| | 0.05 | $75.35 \pm 0.33$ | $\mathbf{96.49 \pm 0.67}$ |

Tables 9 to 11 report the average and standard deviation of the prediction performance of the supervised neural baseline and the semi-supervised neural component on the MNIST-Add$k$, Visual-Sudoku, Citeseer, Cora, and Pathfinding datasets. Across all datasets, as

**Table 10.** Topic accuracy of the trained SGC models with varying levels of supervision.

| | | SGC | |
|---|---|---|---|
| | **Labeled** | **Supervised** | **NeuPSL Semi-Supervised** |
| *Citeseer* | 1.00 | $\mathbf{76.12 \pm 1.71}$ | $75.92 \pm 2.23$ |
| | 0.50 | $\mathbf{74.70 \pm 1.68}$ | $74.38 \pm 1.82$ |
| | 0.10 | $68.64 \pm 1.06$ | $\mathbf{69.66 \pm 0.16}$ |
| | 0.05 | $64.56 \pm 1.68$ | $\mathbf{66.12 \pm 1.22}$ |
| *Cora* | 1.00 | $\mathbf{87.62 \pm 0.97}$ | $87.18 \pm 1.08$ |
| | 0.50 | $85.82 \pm 0.50$ | $\mathbf{86.74 \pm 0.54}$ |
| | 0.10 | $80.88 \pm 2.00$ | $\mathbf{81.96 \pm 2.62}$ |
| | 0.05 | $74.98 \pm 3.32$ | $\mathbf{78.88 \pm 2.85}$ |

**Table 11.** Accuracy of finding a minimum cost path (Min. Cost Acc.) and consistency in satisfying continuity constraints (Continuity) of the ResNet18 models with varying levels of supervision.

| | | ResNet18 | | | |
|---|---|---|---|---|---|
| | | Supervised | | NeuPSL Semi-Supervised | |
| | **Labeled** | **Min. Cost Acc.** | **Continuity** | **Min. Cost Acc.** | **Continuity** |
| *Pathfinding* | 1.00 | $80.12 \pm 22.44$ | $\mathbf{84.80 \pm 17.11}$ | $\mathbf{80.90 \pm 21.93}$ | $83.02 \pm 20.09$ |
| | 0.50 | $52.06 \pm 14.77$ | $61.86 \pm 14.28$ | $\mathbf{59.84 \pm 16.51}$ | $\mathbf{67.94 \pm 14.25}$ |
| | 0.10 | $2.60 \pm 1.04$ | $9.02 \pm 1.90$ | $\mathbf{4.26 \pm 1.40}$ | $\mathbf{35.18 \pm 3.40}$ |

the proportion of unlabeled data increases, the semi-supervised neural component begins to outperform the supervised baseline. This behavior indicates that NeSy-EBMs are able to leverage the unlabeled training data by using the knowledge encoded in the NeuPSL rules. The benefit of utilizing symbolic knowledge is most evident in the lowest supervision settings, with the NeuPSL semi-supervised ResNet18 model achieving over 20 percentage points of improvement when there is only $5\%$ percent of the training labels in the MNIST-Add$k$ and Visual-Sudoku datasets. Surprisingly, this outcome is repeated in the Citeseer and Cora datasets, where the NeuPSL rules are not always adhered to. In other words, leveraging domain knowledge becomes more valuable for improving prediction performance as the amount of supervision decreases, even if the domain knowledge is not strictly accurate.

The Pathfinding results in Table 11 show there is not only a prediction performance gain achievable by making use of the symbolic component but also a reliability improvement. The reported Continuity metric measuring the consistency of the ResNet18 model in satisfying path continuity constraints is significantly improved when there is limited supervision and the model is trained with a NeSy-EBM loss. The NeuPSL semi-supervised ResNet18 model attains an over 25 percentage point improvement in path continuity

consistency when only $10\%$ of training labels are available. These results show NeSy-EBMs are valuable for aligning neural networks with desirable properties beyond accuracy.

## 7 Limitations

In this section, we discuss the limitations of the NeSy-EBM framework, NeuPSL, and our empirical analysis. The NeSy-EBM framework is a high-level and general paradigm for NeSy. The value of the framework is that it provides a unifying theory for NeSy and a foundation for creating widely applicable modeling paradigms and learning algorithms. Progress on developing highly efficient NeSy inference algorithms, on the other hand, may benefit from a perspective that considers the specific structure of the energy function and inference task. For instance, the inference task of density estimation for NeSy systems such as semantic probabilistic layers (Ahmed et al. 2022a) is made highly efficient by levering constraints on the design of the energy function. Similarly, we show prediction in NeuPSL is a quadratic program, a property that is leveraged by Dickens et al. (2024a) to create an inference algorithm tailored for leveraging warm starts to realize learning runtime improvements. In this work, we make limited assumptions on the form of the energy function to develop modeling paradigms and learning algorithms and do not focus on building or analyzing specific inference techniques.

The collection of NeSy modeling paradigms introduced in Section 3 is not exhaustive. For instance, it omits NeSy systems that integrate symbolic knowledge extraction from deep neural networks (Tran and d'Avila Garcez 2018). Moreover, we do not discuss DSVar, DSPar, and DSPot model combinations. We leave the exploration of utilizing multiple NeSy modeling paradigms to fuse neural components operating over multiple modalities for future work.

The four learning techniques proposed in this manuscript are presented with necessary assumptions on the energy function. For instance, direct gradient descent can only be principally applied to minimize a NeSy-EBM loss at points where the energy function is twice differentiable with respect to the neural output and symbolic weights. Similarly, the bilevel technique is principled at points where the optimal value-function is differentiable with respect to the neural output and symbolic weights. We do not explore methods for extending the gradient descent and bilevel learning techniques to support NeSy-EBMs that do not satisfy all assumptions. One approach is to substitute the inference program with an approximation. The modular and stochastic policy optimization learning techniques require significantly fewer assumptions on the form of the energy function. However, these two techniques have their own limitations, which we discuss in their respective subsections.

The NeuPSL system, while expressive, does not support every NeSy-EBM energy function and inference task. Specifically, NeuPSL can create energy functions defined as a weighted sum of potentials derived via arithmetic, logic, and Lukasiewicz real-logic semantics, as described in Section 4. NeuPSL does not support potentials constructed from other real-logic semantics. Further, NeuPSL is currently only designed to perform non-probabilistic inference tasks (e.g., prediction, ranking, and detection). This is due to

the complexities of computing marginal distributions with the Gibbs partition function defined from the energy.

Our empirical evaluations do not encompass every NeSy application, for instance, reasoning with noisy data. Furthermore, although our research advances the incorporation of commonsense reasoning and domain knowledge into LLMs for question answering, we have not extended our investigation to more complex reasoning tasks like summarization or explanation.

## 8 Conclusion and Future Work

This paper establishes a mathematical framework for neural-symbolic (NeSy) reasoning with Neural-Symbolic Energy-Based Models (NeSy-EBMs). The NeSy-EBM framework is a unifying foundation and a bridge for adapting techniques from the broader machine learning literature to solve challenges in NeSy. Additionally, we introduce Neural Probabilistic Soft Logic (NeuPSL), an open-source and highly expressive implementation of NeSy-EBMs. NeuPSL supports the primary modeling paradigms and continuity properties required for efficient end-to-end neural and symbolic parameter learning.

We show that NeSy-EBMs provide a unifying view of NeSy by categorizing fundamental NeSy modeling paradigms. Our modeling paradigms organize the strengths and limitations of NeSy systems and clarify architecture requirements for applications. NeSy-EBMs and the paradigms are valuable mechanisms for practitioners and researchers to understand the growing NeSy literature and design effective systems. Further, NeSy-EBMs illuminate connections between NeSy and the broader machine learning community. Specifically, we formalize a general NeSy learning loss and the necessary assumptions for supporting direct gradient descent on the loss. Moreover, we leverage methods from reinforcement learning and bilevel optimization literature to work around the assumptions and design more practical and general algorithms.

The insights we gained from creating the mathematical framework, the collection of modeling paradigms, and the suite of learning techniques shaped the development of the NeuPSL NeSy modeling library. NeuPSL is built to support every modeling paradigm and learning technique we cover. We demonstrate the effectiveness of NeuPSL in our empirical analysis. Specifically, we explore four practical use cases of NeSy. We show compelling results in real-world applications and see NeSy-EBMs enhance neural network predictions, enforce constraints, improve label and data efficiency, and empower LLMs with consistent reasoning.

Looking ahead, several promising avenues for future research have emerged. For instance, a more extensive exploration into techniques for leveraging symbolic knowledge to fine-tune and adapt foundation models is a promising direction. The NeSy-EBM framework and our proposed learning techniques are a solid basis for building pipelines to fine-tune foundation models. Moreover, stochastic policy optimization for end-to-end NeSy learning has great potential due to its general applicability to every modeling paradigm and most NeSy-EBMs. Finally, contributing to the active area of research on overcoming the challenge of high-variance gradient estimates would be highly beneficial for improving NeSy learning.

## Acknowledgements

This work was partially supported by the National Science Foundation grants CCF-2023495 and a Google Faculty Research Award.

# A Introduction

The appendix includes the following sections: Extended Related work (Appendix B), NeSy Apporaches as NeSy-EBMs (Appendix C), Extended Neural Probabilistic Soft Logic (Appendix D), and Extended Empirical Analysis (Appendix E).

# B Extended Related Work

## *Bilevel Optimization*

In this work, we use bilevel optimization as a natural formulation of learning for a general class of NeSy systems (Bracken and McGill 1973; Colson et al. 2007; F. Bard 2013; Dempe and Zemkoho 2020). The NeSy learning objective is a function of predictions obtained by solving a lower-level program that encapsulates symbolic reasoning. In the broader deep learning community, bilevel optimization also arises in hyperparameter optimization (Pedregosa 2016), meta-learning (Franceschi et al. 2018; Rajeswaran et al. 2019), generative adversarial networks (Goodfellow et al. 2014), and reinforcement learning (Sutton and Barto 2018). Researchers typically take one of the following three approaches to bilevel optimization.

**Implicit Differentiation.** There is a long history of research on analyzing the stability of solutions to optimization problems using implicit differentiation (Fiacco and McCormick 1968; Robinson 1980; Bonnans and Shapiro 2000). These methods compute or approximate the Hessian matrix at the lower-level problem solution to derive an analytic expression for the gradient of the upper-level objective, sometimes called a hypergradient. Bilevel algorithms of this type make varying assumptions about the problem structure (Do et al. 2007; Pedregosa 2016; Ghadimi and Wang 2018; Rajeswaran et al. 2019; Giovannelli et al. 2022; Khanduri et al. 2023). Building on these foundational techniques, the deep learning community has proposed architectures that contain layers that are functions of convex programs with analytic expressions for gradients derived from implicit differentiation (Amos and Kolter 2017; Agrawal et al. 2019a,b; Wang et al. 2019).

**Automatic Differentiation.** This approach unrolls inference into a differentiable computational graph (Stoyanov et al. 2011; Domke 2012; Belanger et al. 2017; Ji et al. 2021), and then leverages automatic differentiation techniques (Griewank and Walther 2008). However, unrolling the inference computation creates a large, complex computational graph that can accumulate numerical errors dependent on the solver.

**Bilevel Value-Function Approach.** An increasingly popular approach is to reformulate the bilevel problem as a single-level constrained program using the optimal value of the lower-level objective (the value-function) to develop principled gradient-based algorithms that do not require the calculation of Hessian matrices for the lower-level problem (V. Outrata 1990; J. Ye and L. Zhu 1995; Liu et al. 2021; Sow et al. 2022; Liu et al. 2022, 2023; Kwon et al. 2023). Existing bilevel value-function approaches are not directly applicable to NeSy systems as they typically assume the lower-level problem to be unconstrained and the objective to be smooth. Bilevel optimization with constraints in the lower level problem, is an open area of research. Until now, implicit differentiation methods are applied with strong assumptions about the structure of the lower-level problem (Giovannelli et al.

2022; Khanduri et al. 2023). Our framework is, to the best of our knowledge, the first value-function approach to work with lower-level problem constraints.

### *Energy-Based Models*

Our NeSy framework makes use of Energy-Based Models (EBMs) (LeCun et al. 2006). EBMs measure the compatibility of a collection of observed (input) variables $\mathbf{x} \in \mathcal{X}$ and target (output) variables $\mathbf{y} \in \mathcal{Y}$ via a scalar-valued energy function: $E : \mathcal{Y} \times \mathcal{X} \to \mathbb{R}$. Low energy states represent high compatibility. EBMs are appealing due to their generality in both modeling and application. For instance, EBMs can be used to perform density estimation by defining conditional, joint, and marginal Gibbs distributions with the energy function:

$$P(\mathbf{y}|\mathbf{x}) := \frac{e^{-\beta E(\mathbf{y},\mathbf{x})}}{\int_{\hat{\mathbf{y}}\in\mathcal{Y}} e^{-\beta E(\hat{\mathbf{y}},\mathbf{x})}}, \tag{70}$$

$$P(\mathbf{y},\mathbf{x}) := \frac{e^{-\beta E(\mathbf{y},\mathbf{x})}}{\int_{\hat{\mathbf{y}}\in\mathcal{Y},\hat{\mathbf{x}}\in\mathcal{X}} e^{-\beta E(\hat{\mathbf{y}},\hat{\mathbf{x}})}}, \tag{71}$$

$$P(\mathbf{x}) := \frac{\int_{\hat{\mathbf{y}}\in\mathcal{Y}} e^{-\beta E(\mathbf{y},\mathbf{x})}}{\int_{\hat{\mathbf{y}}\in\mathcal{Y},\hat{\mathbf{x}}\in\mathcal{X}} e^{-\beta E(\hat{\mathbf{y}},\hat{\mathbf{x}})}}. \tag{72}$$

A fundamental motivation for the use of the Gibbs distribution is that any density function can be represented by the distribution shown above with a (potentially un-normalized) energy function $E$. For this reason, EBMs are a unified framework for probabilistic and non-probabilistic approaches and are applicable for generative and discriminative modeling.

EBMs are applied throughout machine learning to model data and provide predictions. The Boltzmann machine (Ackley et al. 1985; Salakhutdinov and Larochelle 2010) and Helmholtz machine (Dayan et al. 1995) are some of the earliest EBMs to appear in the machine learning literature. Hinton (2002) is another seminal work that shows EBMs to be useful for building mixture-of-expert models. Specifically, a single complex distribution is produced by multiplying many simple distributions together and then renormalizing.

More recently, the EBM framework has been utilized for generative modeling (Zhao et al. 2017; Du and Mordatch 2019; Du et al. 2023). Zhao et al. (2017) introduce energy-based generative adversarial networks (EBGANs), which view the GAN discriminator as an energy function that attributes low energies (high compatibility) to points near the data manifold. The EBGAN approach is a principled framework for using GAN discriminators with a variety of architectures and learning loss functionals to achieve more stable training than traditional GANs. Du and Mordatch (2019) advocate for using EBMs *directly* for generative modeling, citing as motivation their simplicity, stability, parameter efficiency, flexibility of generation, and compositionality. They show generative results that achieve performance close to modern GANs, achieving state-of-the-art results in out-of-distribution classification, adversarially robust classification, and other tasks. In more recent work, Du et al. (2023) propose an energy-based parameterization of diffusion models to support compositional generation.

The EBM framework was shown recently to improve discriminative modeling (Grathwohl et al. 2020; Liu et al. 2020). Grathwohl et al. (2020) reinterpret discriminative classifiers as EBMs to propose the joint energy-based model (JEM). A JEM allows the parameters of the model to be fit on unlabeled data with a likelihood-based loss, leading to improved accuracy, robustness, calibration, and out-of-distribution detection. Similarly, Liu et al. (2020) developed an EBM for out-of-distribution detection to achieve state-of-the-art performance. Liu et al. (2020) creates a purely discriminative training objective (in contrast with the probabilistic approach of JEM) and shows that unnormalized energy scores can be used directly for out-of-distribution detection.

A primary challenge of the EBM framework is learning with a potentially in-tractable partition function induced by the Gibbs distributions in (70), (71), and (72). Some of the earliest EBMs worked around the partition function using the contrastive divergence algorithm (Hinton 2002) to estimate derivatives of the negative log-likelihood loss of an EBM with Markov chain Monte Carlo (MCMC) sampling from the Gibbs distribution. Later work on EBMs has improved the traditional biased MCMC sampling-based approximation methods with a sampler based on stochastic gradient Langevin dynamics (SGLD) (Welling and Teh 2011). For instance, Du and Mordatch (2019) use SGLD for training generative EBMs and Grathwohl et al. (2020) for discriminative models with a negative log-likelihood loss.

Score matching is an alternative probabilistic approach to training an EBM that fits the slope (or score) of the model density to the score of the data distribution, avoiding the need to estimate the Gibbs distribution partition function (Hyvarinen 2005; P. Kingma and LeCun 2010; Song and Ermon 2019). Hyvarinen (2005) initially proposed score matching for estimating non-normalized statistical models. Later, P. Kingma and LeCun (2010) used score matching to train an EBM for image denoising and super-resolution. Song and Ermon (2019) suggested training an EBM to approximate the score of the data distribution that is then used with Langevin dynamics for generation.

EBMs may also be trained via non-probabilistic losses that do not require estimating the Gibbs distribution partition function (LeCun et al. 1998; Collins 2002; Scellier and Bengio 2017). For instance, the perceptron loss, which is the difference between the energy of the observed training data and the minimum value of the energy function (see Section 5 for a formal definition), has been used for recognizing handwritten digits (LeCun et al. 1998) and part-of-speech tagging (Collins 2002). More recently, Scellier and Bengio (2017) proposed *equilibrium propagation*, a two-phase learning algorithm for training EBMs with a twice differentiable energy function. The equilibrium propagation algorithm can be used to train EBMs with an arbitrary differentiable loss. A step of the learning algorithm proceeds by minimizing the energy given some input (the free phase) and then minimizing the energy augmented with a cost function (the nudged phase). The gradient of the learning objective is a function of the results of these two minimizations.

The EBM framework has proven effective for a wide range of tasks in both generative and discriminative modeling. The versatility of EBMs supports modeling complex dependencies, the composition and fusion of models, and leveraging both labeled and unlabeled data. Moreover, EBMs provide a common theoretical framework spanning probabilistic and non-probabilistic methods.

## C Expressing NeSy Approaches via NeSy-EBMs

Chapter 3 introduced NeSy-EBMs as a unifying framework for understanding neural-symbolic integration. In this section, we demonstrate how to formulate three widely recognized NeSy approaches within this framework, highlighting their distinct strategies for combining neural learning with symbolic reasoning. These approaches differ in how they incorporate symbolic knowledge, enforce constraints, and structure their overall modeling paradigms:

- **Semantic Loss (SL)** (Xu et al. 2018): A probabilistic NeSy approach in which the neural network outputs define a probability distribution over variables or facts within probabilistic propositional logical constraints. These constraints are integrated as a regularization term within the loss function of the neural network. This approach is typically expressed as a deep symbolic parameter model.

- **DeepProbLog (DPL)** (Manhaeve et al. 2021a): A probabilistic NeSy approach in which the neural network outputs define a probability distribution over variables or facts within a probabilistic logic setting. Probabilistic constraints can be applied either as a loss regularization term or as an additional reasoning layer on top of the neural network outputs. This approach is typically expressed as a deep symbolic parameter model.

- **Logic Tensor Networks (LTNs)** (Badreddine et al. 2022): A fuzzy NeSy approach in which the neural network outputs define variable values within a fuzzy logic system. Fuzzy logic constraints are incorporated through an additional computation graph layer on the neural network outputs. This approach is typically expressed as a deep symbolic variable model.

### *Semantic Loss (SL)*

Semantic loss is a probabilistic NeSy approach that interprets neural network outputs as probability distributions over propositional variables constrained by logical rules (Xu et al. 2018). It measures the extent to which the neural network's predictions satisfy the given constraints, serving as a regularization term to encourage logical consistency by penalizing violations.

*Defintion and Background* Formally, let $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ be a set of $n$ propositional variables and a *world* $\omega$ is an instantiation of all variables $\mathbf{y}$, i.e., $\omega = \{y_1 = v_1, \cdots, y_n = v_n\}$ for $v_i \in \{0, 1\}$. A world $\omega$ satisfies a sentence $\alpha$, denoted $\omega \models \alpha$, if the sentence evaluates to true in that world. A sentence $\alpha$ *entails* another sentence $\beta$, denoted $\alpha \models \beta$, if all worlds that satisfy $\alpha$ also satisfy $\beta$.

In semantic loss, each propositional variable $y_i \in \mathbf{y}$ is assigned a probability $p_i$ that it is true. These probabilities $\mathbf{p} = \{p_1, p_2, \ldots, p_n\}$ are defined from a neural network. The *semantic loss*, denoted as $L_{SL}(\alpha, \mathbf{p})$, measures how closely the neural network's predictions satisfy the sentence $\alpha$ with:

$$L_{SL}(\alpha, \mathbf{p}) \propto -\log\left(\sum_{\omega \models \alpha} \prod_{i:\omega \models y_i} p_i \prod_{i:\omega \models \neg y_i} (1-p_i)\right)$$

The summation is taken over all worlds $\omega$ that satisfy $\alpha$. The product $\prod_{i:\omega \models y_i} p_i$ represents the probability of all variables assigned true in $\omega$, while $\prod_{i:\omega \models \neg y_i}(1-p_i)$ accounts for the probability of all variables assigned false in $\omega$. The right-hand side of the equation corresponds to the well-known reasoning task of weighted model counting (WMC) (Chavira and Darwiche 2008), which computes the total probability of satisfying $\alpha$ under a given probability distribution.

In practice, semantic loss—weighted by a hyperparameter $\gamma$—serves as a regularization term that encourages the neural model to produce predictions consistent with logical constraints. Formally, semantic loss is incorporated as follows:

$$L_{\text{total}} = L_{\text{existing}} + \gamma \cdot L_{SL}(\alpha, \mathbf{p}),$$

where $L_{\text{existing}}$ represents the original loss function, and $L_{SL}(\alpha, \mathbf{p})$ enforces logical consistency within the learned predictions.

With the semantic loss formalized, we now illustrate its application through an example: enforcing mutual exclusivity constraints in multi-class classification.

**Example 4**

*Consider a multi-class classification problem where exactly one of $n$ possible outcomes should be true. The sentence $\alpha_{\textit{exactly\_one}}$ enforces that exactly one of the variables $y_1, \ldots, y_n$ is true:*

$$\alpha_{\textit{exactly\_one}} = (y_1 \wedge \neg y_2 \wedge \cdots \wedge \neg y_n) \vee (\neg y_1 \wedge y_2 \wedge \cdots \wedge \neg y_n) \vee \cdots \vee (\neg y_1 \wedge \cdots \wedge y_n).$$

*The semantic loss for this multi-class setting is computed as:*

$$L_{SL}(\alpha_{\textit{exactly\_one}}, \mathbf{p}) \propto -\log\left(\sum_{i=1}^{n} p_i \prod_{\substack{j=1 \\ j \neq i}}^{n} (1-p_j)\right),$$

*where $\mathbf{p} = \{p_1, \ldots, p_n\}$ are the probabilities predicted by the neural network for each class.*

While this demonstrates how more complex logical sentences can be incorporated into the problem, it is crucial to recognize that this requires computing the weighted model count (Chavira and Darwiche 2008), a task that is #P-hard. To mitigate this computational challenge, a common approach in the literature is to compile logical formulas into circuits, enabling efficient inference across multiple queries (Choi et al. 2020; Kisa et al. 2014; Manhaeve et al. 2021a). However, while circuit compilation can significantly accelerate inference, the compilation process itself can be exponential in both time and memory, with no guarantees that the resulting circuit size remains tractable (van Krieken et al. 2023).

*NeSy-EBM Formulation* Semantic loss can be formulated as a DSPar NeSy-EBM where the propositional variable probabilities are neural network outputs. Since semantic loss does not take as input any observed random variables, then there are no external parameters, i.e., let $\mathbf{x}_{sy} \in \emptyset$ and $\mathbf{w}_{sy} \in \emptyset$. Let $g_{nn}$ be a function with parameters $\mathbf{w}_{nn} \in W_{nn}$ and inputs $\mathbf{x}_{nn} \in X_{nn}$ such that:

$$g_{nn} : W_{nn} \times X_{nn} \mapsto [0,1]^n,$$

where $n$ is the number of propositional variables involved in the constraints, this function outputs the predicted probabilities:

$$g_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) = [p_i]_{i=1}^n,$$

where $p_i$ represents the predicted probability for the $i$-th variable.

Define the logical sentence $\alpha$ as a hard constraint $C_H(\omega)$ using an indicator function that represents whether a world $\omega$ (an assignment of the propositional variables) satisfies the sentence $\alpha$:

$$C_H(\omega) = \mathbb{I}(\omega \models \alpha),$$

where $\mathbb{I}(\omega \models \alpha)$ is 1 if the world satisfies the constraint $\alpha$, and 0 otherwise.

Assuming the weighted model count formulation, the symbolic component of the NeSy-EBM is a DSPar potential function:

$$\begin{aligned} g_{sy}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) &= \psi_{SL}([\mathbf{y}, \mathbf{x}_{sy}], [\mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})]) \\ &= -\log\left(\sum_{\omega \in \{0,1\}^{|y|}} C_H(\omega) \prod_{i=1}^{n} p_i^{\omega_i}(1-p_i)^{1-\omega_i}\right) \end{aligned}$$

where $\omega_i$ is the $i^{th}$ propositional variable value.

Given the symbolic potential and variables defined above, the semantic energy function is defined as:

$$\begin{aligned} E_{SL}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) &= g_{sy}(y, \mathbf{x}_{sy}, \mathbf{w}_{sy}, g_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \\ &= -\log\left(\sum_{\omega \in \{0,1\}^{|y|}} C_H(\omega) \prod_{i=1}^{n} p_i^{\omega_i}(1-p_i)^{1-\omega_i}\right) \end{aligned} \tag{73}$$

## *DeepProbLog (DPL)*

DeepProbLog (DPL) (Manhaeve et al. 2021a), like semantic loss, is a probabilistic NeSy approach that integrates neural network predictions with symbolic reasoning. However, while semantic loss is confined to propositional logic, DeepProbLog operates within a first-order logic framework, allowing for more expressive reasoning capabilities. Specifically, DeepProbLog assigns neural network outputs as probabilities within the probabilistic programming language ProbLog (De Raedt et al. 2007) through the use of neural predicates. This integration allows DeepProbLog to leverage both probabilistic logic programming and program induction, making it well-suited for reasoning over more complex symbolic structures and relational dependencies.

*Definition and Background* To begin, we review the basics of probabilistic logic programming in ProbLog, following the presentation from (Manhaeve et al. 2021a) (see (De Raedt et al. 2007) for further details).

*ProbLog:* A ProbLog program consists of two main components:

- A set of *probabilistic facts* $\mathcal{F}$ of the form $p :: y$, where $p$ is the probability that the binary target random variable $y$ is true (i.e., $y \in \{0, 1\}$). Note: this can be extended to include a set of observed facts or evidence $p :: x_{sy}$.
- A set $\mathcal{R}$ of *symbolic rules*, which describe how different facts relate to each other.

A subset of the probabilistic facts $F \subseteq \mathcal{F}$ defines a possible instantiation, or *world* $\omega$. This world includes all facts in $F$ and all facts derivable from $F$ using the rules in $\mathcal{R}$:

$$\omega = F \cup \{y \mid \mathcal{R} \cup F \models y\},$$

where $\mathcal{R} \cup F \models y$ means that the fact $y$ can be derived from the combination of rules $\mathcal{R}$ and facts $F$. The probability of a world $\omega$ is given by the product of the probabilities of the facts in that world:

$$P(\omega) := \prod_{y_i \in F} p_i \prod_{y_i \in \mathcal{F} \setminus F} (1 - p_i),$$

where $p_i$ is the probability assigned to fact $y_i$.

**Example 5**
*Consider the following ProbLog program, which models the likelihood of a burglary or earthquake causing an alarm:*

***Probabilistic Facts:***

$0.1 :: \mathit{burglary}, \quad 0.2 :: \mathit{earthquake},$

$0.5 :: \mathit{hearsAlarm(mary)}, \quad 0.4 :: \mathit{hearsAlarm(john)}.$

***Rules:***

$\mathit{alarm} :- \mathit{earthquake}.$

$\mathit{alarm} :- \mathit{burglary}.$

$\mathit{calls(X)} :- \mathit{alarm}, \mathit{hearsAlarm(X)}.$

*Now, consider the subset* $F = \{\mathit{burglary}, \mathit{hearsAlarm(mary)}\}$ *of probabilistic facts. The corresponding world* $\omega$ *includes the derived facts:*

$$\omega = \{\mathit{burglary}, \mathit{hearsAlarm(mary)}, \mathit{alarm}, \mathit{calls(mary)}\}.$$

*The probability of this world is:*

$$P(\omega) = 0.1 \cdot 0.5 \cdot (1 - 0.2) \cdot (1 - 0.4) = 0.024.$$

*DeepProbLog:* A DeepProbLog program extends the syntax and semantics of ProbLog by introducing neural predicates, allowing the specification of probabilistic facts based on neural network outputs (Manhaeve et al. 2021a). Specifically, DeepProbLog introduces neural annotated disjunctions (nADs), which integrate neural network predictions directly into the logic. A neural annotated disjunction is specified as:

$$nn(m_{g_{nn}}, \mathbf{x}_{nn}, u_1) \;::\; h(\mathbf{x}_{nn}, u_1)\,; \cdots ; \; nn(m_{g_{nn}}, \mathbf{x}_{nn}, u_n) \;::\; h(\mathbf{x}_{nn}, u_n) \;\vDash\; b_1, \cdots, b_m, \tag{74}$$

where $\mathbf{x}_{nn}$ is a vector of features accessible to the neural component identified by $m_{g_{nn}}$. The terms $u_1, \ldots, u_n$ represent the possible outputs of the neural network, and the atoms $b_1, \ldots, b_m$ are logical conditions. The output of the neural network, $nn(m_{g_{nn}}, \mathbf{x}_{nn}, u_i)$, is interpreted as the probability that the atom $h(\mathbf{x}_{nn}, u_i)$ is true, and the sum of the neural model's outputs must satisfy:

$$\sum_{i=1}^{n} nn(m_{g_{nn}}, \mathbf{x}_{nn}, u_i) = 1.$$

The meaning of the nAD is that whenever all the atoms $b_1, \ldots, b_m$ hold true, each $h(\mathbf{x}_{nn}, u_i)$ becomes true with probability $nn(m_{g_{nn}}, \mathbf{x}_{nn}, u_i)$.

*NeSy-EBM Formulation* DeepProbLog can be formulated as a DSPar NeSy-EBM where the fact probabilities are defined with both the symbolic parameters and the neural network outputs. Let $\mathbf{x}_{sy}$ be the observed random variables (evidence), $\mathbf{y}$ be the target random variables (facts), and $\mathbf{w}_{sy}$ be symbolic parameters over facts not parameterized by a neural network (probabilities). Let $g_{nn}$ be a function with parameters $\mathbf{w}_{nn} \in W_{nn}$ and inputs $\mathbf{x}_{nn} \in X_{nn}$ such that:

$$g_{nn} : W_{nn} \times X_{nn} \mapsto [0,1]^n,$$

where $n$ is the number of propositional variables involved in the constraints. Without loss of generality the fact probabilities, $\mathbf{p}$, are paritioned into symbolic parameters and these neural network outputs:

$$\mathbf{p} = \begin{bmatrix} \mathbf{w}_{sy} \\ \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) \end{bmatrix}$$

where $p_i$ represents the predicted probability for the $i$-th variable.

The probability of a world $\omega$, defined by a subset of probabilistic facts $F \subseteq \mathcal{F}$, is a function of the fact probabilities $\mathbf{p}$, and therefore a function of $\mathbf{w}_{sy}$ and the neural network outputs $\mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})$:

$$P_\omega(\mathbf{w}_{sy}, \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) = \prod_{\mathbf{w}^j_{sy} \in F} \mathbf{w}^j_{sy} \prod_{\mathbf{w}^j_{sy} \in \mathcal{F} \setminus F} (1 - \mathbf{w}^j_{sy}) \prod_{\mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})^j \in F} \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})^j \prod_{\mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})^j \in \mathcal{F} \setminus F} (1 - \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})^j)$$

Finally, unlike semantic loss, DeepProbLog typically evaluates probabilities with respect to *queries*. A *query* is a symbolic atom $q$ whose probability we want to compute based on the probabilistic facts and the neural network outputs. For example, the marginal probability of a query atom $q$ is computed by summing over the probabilities of all worlds $\omega$ in which $q$ is true.

Let $C_H(\omega, q)$ be the constraint function, which acts as an indicator function returning 1 if the world $\omega$ satisfies the condition that the query atom $q$ is true:

$$C_H(\omega, q) = \begin{cases} 1 & \text{if } q \in \omega \\ 0 & \text{otherwise.} \end{cases}$$

Assuming the weighted model count formulation, the symbolic component of the NeSy-EBM is a DSPar potential function for a single query q is defined as:

$$\begin{aligned} g_{sy}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) &= \psi^q_{DPL}([\mathbf{y}, \mathbf{x}_{sy}], [\mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})]) \\ &= d\left(q, \sum_{\omega \in \{0,1\}^{|\mathbf{y}|}} C_H(\omega, q) P_\omega(\mathbf{w}_{sy}, \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))\right) \end{aligned}$$

Where $d(\cdot, \cdot)$ is the distance between the predicted probability of the query and its true probability. Now, the energy function is defined over a sum of all queries $\mathbf{q}$.

$$E_{DPL}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) = \sum_{i=1}^{|\mathbf{q}|} \psi^{q_i}_{DPL}([\mathbf{y}, \mathbf{x}_{sy}], [\mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})]).$$

### *Logic Tensor Networks (LTNs)*

Logic Tensor Networks (LTNs) (Badreddine et al. 2022) are a fuzzy Neural-Symbolic Energy-Based Model (NeSy-EBM) approach, integrating neural network predictions with logic-based reasoning. In LTNs, neural networks provide real-valued truth values for predicates, which are then manipulated using fuzzy logic operations to evaluate logical formulae. The satisfaction levels of the logical formulae are aggregated through generalized mean semantics, which form the basis of the energy function.

LTNs use product real logic operators to define fuzzy truth values for logical connectives:

$$\neg(a) := 1 - a, \quad \wedge(a, b) := a \cdot b, \quad \vee(a, b) := a + b - a \cdot b, \quad \implies (a, b) := a + b - a \cdot b.$$

Additionally, LTNs use formula aggregators, such as generalized mean semantics, to handle existential and universal quantifiers over collections of truth values, denoted by $\mathbf{a} = [a_i]_{i=1}^n$:

$$\exists(\mathbf{a}) := \left(\frac{1}{n}\sum_{i=1}^{n} a_i^p\right)^{\frac{1}{p}}, \quad \forall(\mathbf{a}) := 1 - \left(\frac{1}{n}\sum_{i=1}^{n}(1 - a_i)^p\right)^{\frac{1}{p}},$$

where $p \in \mathbb{R}_+$ is a hyperparameter controlling the smoothness of the quantifiers.

In LTNs, neural networks instantiate predicates with values from $[0, 1]$, representing the degree to which a predicate is satisfied. For example, given two entities $u$ and $v$, the predicate $P(u, v)$ can be defined as the output of a neural network $g_{nn}(\mathbf{X}[u], \mathbf{X}[v]; \mathbf{w}_{nn})$, which maps the feature vectors $\mathbf{X}[u]$ and $\mathbf{X}[v]$ to a truth value in $[0, 1]$.

**Example 6**
*Consider the following logical formula, which expresses that for each entity $u$, there exists some entity $v$ such that both predicates $P(u, v)$ and $Q(v)$ hold true:*

$$\exists v \in \mathcal{V} \left( P(u, v) \wedge Q(v) \right).$$

*Let $\mathbf{X}_{\mathcal{U}}$ and $\mathbf{X}_{\mathcal{V}}$ represent the feature vectors for the sets of entities $\mathcal{U}$ and $\mathcal{V}$, respectively. The predicates $P(u, v)$ and $Q(v)$ can be instantiated with neural network outputs: - $P(u, v)$ is given by the neural network $g_{nn}(\mathbf{X}[u], \mathbf{X}[v]; \mathbf{w}_{nn})$, - $Q(v)$ could be a constant truth value or another neural network prediction.*

*Using the generalized mean semantics for the existential quantifier, we define the real-valued logic function for the above formula as:*

$$h_u(\mathbf{X}_{\mathcal{U}}, \mathbf{X}_{\mathcal{V}}, \mathbf{x}_Q; \mathbf{w}_{nn}) = \left( \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \left( g_{nn}(\mathbf{X}[u], \mathbf{X}[v]; \mathbf{w}_{nn}) \cdot \mathbf{x}_Q[v] \right)^p \right)^{\frac{1}{p}},$$

*where $\mathbf{x}_Q[v]$ is the truth value for the predicate $Q(v)$, and $g_{nn}(\mathbf{X}[u], \mathbf{X}[v]; \mathbf{w}_{nn})$ is the neural network output for the predicate $P(u, v)$.*

*The satisfaction level of the formula for all entities $u$ is then aggregated using the universal quantifier:*

$$G(\mathbf{w}_{nn}) = 1 - \left( \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \left( 1 - h_u(\mathbf{X}_{\mathcal{U}}, \mathbf{X}_{\mathcal{V}}, \mathbf{x}_Q; \mathbf{w}_{nn}) \right)^p \right)^{\frac{1}{p}}.$$

This example illustrates how LTNs leverage neural networks to assign fuzzy truth values to predicates and apply these values in evaluating logical formulas. The next section will explain how LTNs can be represented as NeSy-EBMs using symbolic constraints.

*NeSy-EBM Formulation* LTNs can be formulated as a DSVar NeSy-EBM, where the satisfaction of symbolic constraints is driven by neural network outputs. Let $\mathbf{x}_{sy}$ be the observed random variables (constants), and since the symbolic component does not have trainable parameters, define $\mathbf{w}_{sy} \in \emptyset$. Let $g_{nn}$ be a function with parameters $\mathbf{w}_{nn} \in W_{nn}$ and inputs $\mathbf{x}_{nn} \in X_{nn}$ such that:

$$g_{nn} : W_{nn} \times X_{nn} \mapsto [0, 1]^n,$$

where $n$ is the number of variables involved in the constraints.

Define $C_S^{Agg}(\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}), \mathbf{x}_{sy}, \mathbf{w}_{sy})$ as the soft constraint function that takes as input the output of a set of neural networks $\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})$ and applies the collection of aggregation functions $Agg$ in some way that maintains differentability (e.g., the generalized mean or quantifiers).

The symbolic component of the NeSy-EBM is a DSVar potential function:

$$\begin{aligned} g_{sy}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) &= \psi_{LTN}([\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})], [\mathbf{w}_{sy}]) \\ &= C_S^{Agg}(\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}), \mathbf{x}_{sy}, \mathbf{w}_{sy}) \end{aligned}$$

Given the symbolic potential and variables defined above, the energy function for LTNs is defined as:

$$\begin{aligned} E_{LTN}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) &= g_{sy}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{w}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \\ &= C_S^{Agg}(\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}), \mathbf{x}_{sy}, \mathbf{w}_{sy}). \end{aligned}$$

## D Extended Neural Probabilistic Soft Logic

In this section, we expand on the smooth formulation of NeuPSL inference and provide proofs for the continuity results presented in Section 4.

### *Extended Smooth Formulation of Inference*

Recall the primal formulation of NeuPSL inference restated below:

$$\arg\min_{\mathbf{y} \in \mathbb{R}^{n_{\mathbf{y}}}} \mathbf{w}_{sy}^T \mathbf{\Phi}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \quad \text{s.t. } \mathbf{y} \in \mathbf{\Omega}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})). \tag{75}$$

Importantly, note the structure of the deep hinge-loss potentials defining $\mathbf{\Phi}$:

$$\phi_k(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \tag{76}$$

$$:= (\max\{\mathbf{a}_{\phi_k,y}^T \mathbf{y} + \mathbf{a}_{\phi_k,\mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{\phi_k,\mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_{\phi_k}, 0\})^{p_k}. \tag{77}$$

The LCQP NeuPSL inference formulation is defined using ordered index sets: $\mathbf{I}_S$ for the partitions of squared hinge potentials (indices $k$ which for all $j \in t_k$ the exponent term $p_j = 2$) and $\mathbf{I}_L$ for the partitions of linear hinge potentials (indices $k$ which for all $j \in t_k$ the exponent term $p_j = 1$). With the index sets, we define

$$\mathbf{W}_S := \begin{bmatrix} w_{\mathbf{I}_S[1]}\mathbf{I} & 0 & \cdots & 0 \\ 0 & w_{\mathbf{I}_S[2]}\mathbf{I} & & \\ \vdots & & \ddots & \end{bmatrix} \quad \text{and} \quad \mathbf{w}_L := \begin{bmatrix} w_{\mathbf{I}_L[1]}\mathbf{1} \\ w_{\mathbf{I}_L[2]}\mathbf{1} \\ \vdots \end{bmatrix} \tag{78}$$

Let $m_S := |\cup_{\mathbf{I}_S} t_k|$ and $m_L := |\cup_{\mathbf{I}_L} t_k|$, be the total number of squared and linear hinge potentials, respectively, and define slack variables $\mathbf{s}_S := [s_j]_{j=1}^{m_S}$ and $\mathbf{s}_L := [s_j]_{j=1}^{m_L}$ for each of the squared and linear hinge potentials, respectively. NeuPSL inference is equivalent to the following LCQP:

$$\min_{\mathbf{y} \in [0,1]^{n_y}, \mathbf{s}_S \in \mathbb{R}^{m_S}, \mathbf{s}_{\mathbf{H}} \in \mathbb{R}_+^{m_L}} \mathbf{s}_S^T \mathbf{W}_S \mathbf{s}_S + \mathbf{w}_L^T \mathbf{s}_L \tag{79a}$$

$$\text{s.t.} \quad \mathbf{a}_{c_i,\mathbf{y}}^T \mathbf{y} + \mathbf{a}_{c_i,\mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{c_i,\mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_{c_i} \leq 0 \quad \forall i = 1, \ldots, q, \tag{79b}$$

$$\mathbf{a}_{\phi_j,\mathbf{y}}^T \mathbf{y} + \mathbf{a}_{\phi_j,\mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{\phi_j,\mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_{\phi_j} - s_j \leq 0 \quad \forall j \in I_S \cup I_L. \tag{79c}$$

We ensure strong convexity by adding a square regularization with parameter $\epsilon$ to the objective. Let the bound constraints on $\mathbf{y}$ and $\mathbf{s}_L$ and linear inequalities in the LCQP be captured by the $(2 \cdot n_y + q + m_S + 2 \cdot m_L) \times (n_y + m_S + m_L)$ matrix $\mathbf{A}$ and $(2 \cdot n_y + q + m_S + 2 \cdot m_L)$ dimensional vector $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$. More formally,

$\mathbf{A} := [a_{ij}]$ where $a_{ij}$ is the coefficient of a decision variable in the implicit and explicit constraints in the formulation above:

$$a_{i,j} := \begin{cases} 0 & (i \leq q) \wedge (j \leq m_S + m_L) \\ \mathbf{a}_{c_i,\mathbf{y}}[j - (m_S + m_L)] & (i \leq q) \wedge (j > m_S + m_L) \\ 0 & (q < i \leq q + m_S + m_L) \wedge (j \leq m_S + m_L) \wedge (j \neq i - q) \\ -1 & (q < i \leq q + m_S + m_L) \wedge (j \leq m_S + m_L) \wedge (j = i - q) \\ \mathbf{a}_{\phi_{i-q},\mathbf{y}}[j - (m_S + m_L)] & (q < i \leq q + m_S + m_L) \wedge (j > m_S + m_L) \\ 0 & (q + m_S + m_L < i \leq q + m_S + 2 \cdot m_L + n_y) \\ & \wedge (j \neq i - (q + m_L)) \\ -1 & (q + m_S + m_L < i \leq q + m_S + 2 \cdot m_L + n_y) \\ & \wedge (j = i - (q + m_L)) \\ 0 & (q + m_S + 2 \cdot m_L + n_y < i \leq q + m_S + 2 \cdot m_L + 2 \cdot n_y) \\ & \wedge (j \neq i - (q + m_S + m_L)) \\ 1 & (q + m_S + 2 \cdot m_L + n_y < i \leq q + m_S + 2 \cdot m_L + 2 \cdot n_y) \\ & \wedge (j = i - (q + m_S + m_L)) \end{cases} \tag{80}$$

Furthermore, $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) = [b_i(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))]$ is the vector of constants corresponding to each constraint in the formulation above:

$$b_i(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \tag{81}$$

$$:= \begin{cases} \mathbf{a}_{c_i,\mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{c_i,\mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_{c_i} & i \leq q \\ \mathbf{a}_{\phi_{i-q},\mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{\phi_{i-q},\mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_{\phi_{i-q}} & q < i \leq q + m_S + m_L \\ 0 & q + m_S + m_L < i \\ & \quad \leq q + m_S + 2 \cdot m_L + n_y \\ -1 & q + m_S + 2 \cdot m_L + n_y < i \\ & \quad \leq q + m_S + 2 \cdot m_L + 2 \cdot n_y \end{cases} \tag{82}$$

Note that $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$ is a linear function of the neural network outputs, hence, if $\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})$ is a smooth function of the neural parameters, then $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$ is also smooth.

With this notation, the regularized inference problem is:

$$\begin{aligned} & V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) \\ & \qquad := \min_{\mathbf{y},\mathbf{s_S},\mathbf{s_H}} \begin{bmatrix} \mathbf{s}_S \\ \mathbf{s}_L \\ \mathbf{y} \end{bmatrix}^T \begin{bmatrix} \mathbf{W}_S + \epsilon I & 0 & 0 \\ 0 & \epsilon I & 0 \\ 0 & 0 & \epsilon I \end{bmatrix} \begin{bmatrix} \mathbf{s}_S \\ \mathbf{s}_L \\ \mathbf{y} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{w}_L \\ 0 \end{bmatrix}^T \begin{bmatrix} \mathbf{s}_S \\ \mathbf{s}_L \\ \mathbf{y} \end{bmatrix} \\ & \qquad \text{s.t.} \quad \mathbf{A} \begin{bmatrix} \mathbf{s}_S \\ \mathbf{s}_L \\ \mathbf{y} \end{bmatrix} + \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \leq 0. \end{aligned} \tag{83}$$

For ease of notation, let

$$D(\mathbf{w}_{sy}) := \begin{bmatrix} \mathbf{W}_S & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{c}(\mathbf{w}_{sy}) := \begin{bmatrix} 0 \\ \mathbf{w}_L \\ 0 \end{bmatrix}, \nu := \begin{bmatrix} \mathbf{s}_S \\ \mathbf{s}_L \\ \mathbf{y} \end{bmatrix}. \tag{84}$$

Then the regularized primal LCQP MAP inference problem is concisely expressed as

$$\begin{aligned} \min_{\nu \in \mathbb{R}^{n_{\mathbf{y}}+m_S+m_L}} \quad & \nu^T(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon\mathbf{I})\nu + \mathbf{c}(\mathbf{w}_{sy})^T\nu \\ \text{s.t.} \quad & \mathbf{A}\nu + \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \leq 0. \end{aligned} \tag{85}$$

By Slater's constraint qualification, we have strong-duality when there is a feasible solution. In this case, an optimal solution to the dual problem yields an optimal solution to the primal problem. The Lagrange dual problem of (85) is

$$\begin{aligned} &\arg\max_{\mu \geq 0} \min_{\nu \in \mathbb{R}^{n_{\mathbf{y}}+m_S+m_L}} \nu^T(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon\mathbf{I})\nu + \mathbf{c}(\mathbf{w}_{sy})^T\nu + \mu^T(\mathbf{A}\nu + \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) \\ &= \arg\max_{\mu \geq 0} -\frac{1}{4}\mu^T\mathbf{A}(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon\mathbf{I})^{-1}\mathbf{A}^T\mu \\ &\qquad - \frac{1}{2}(\mathbf{A}(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon\mathbf{I})^{-1}\mathbf{c}(\mathbf{w}_{sy}) - 2\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))^T\mu \end{aligned} \tag{86}$$

where $\mu = [\mu_i]_{i=1}^{n_\mu}$ are the Lagrange dual variables. For later reference, denote the negative of the Lagrange dual function of MAP inference as:

$$h(\mu; \mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) \tag{87}$$

$$\begin{aligned} &:= \frac{1}{4}\mu^T\mathbf{A}(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon\mathbf{I})^{-1}\mathbf{A}^T\mu + \frac{1}{2}(\mathbf{A}(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon\mathbf{I})^{-1}\mathbf{c}(\mathbf{w}_{sy}) \\ &\quad - 2\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))^T\mu. \end{aligned} \tag{88}$$

The dual LCQP has more decision variables but is only over non-negativity constraints rather than the complex polyhedron feasible set. The dual-to-primal variable translation is:

$$\nu = -\frac{1}{2}(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon\mathbf{I})^{-1}(\mathbf{A}^T\mu + \mathbf{c}(\mathbf{w}_{sy})) \tag{89}$$

As $(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon\mathbf{I})$ is diagonal, it is easy to invert and hence it is practical to work in the dual space to obtain a solution to the primal problem.

## *Extended Continuity of Inference*

We now provide background on sensitivity analysis that we then apply in our proofs on the continuity properties of NeuPSL inference.

### *Background*

**Theorem 11** Boyd and Vandenberghe (2004) p. 81.
*If for each* $\mathbf{y} \in \mathcal{A}$, $f(\mathbf{x}, \mathbf{y})$ *is convex in* $\mathbf{x}$ *then the function*

$$g(\mathbf{x}) := \sup_{\mathbf{y} \in \mathcal{A}} f(\mathbf{x}, \mathbf{y}) \tag{90}$$

*is convex in* $\mathbf{x}$.

**Theorem 12** Boyd and Vandenberghe (2004) p. 81.
*If for each* $\mathbf{y} \in \mathcal{A}$, $f(\mathbf{x}, \mathbf{y})$ *is concave in* $\mathbf{x}$ *then the function*

$$g(\mathbf{x}) := \inf_{\mathbf{y} \in \mathcal{A}} f(\mathbf{x}, \mathbf{y}) \tag{91}$$

*is concave in* $\mathbf{x}$.

**Definition 13** Convex Subgradient: Boyd and Vandenberghe (2004) and Shalev-Shwartz (2012).
*Consider a convex function* $f : \mathbb{R}^n \to [-\infty, \infty]$ *and a point* $\overline{\mathbf{x}}$ *with* $f(\overline{\mathbf{x}})$ *finite. For a vector* $\mathbf{v} \in \mathbf{R}^n$, *one says that* $\mathbf{v}$ *is a (convex) subgradient of* $f$ *at* $\overline{\mathbf{x}}$, *written* $\mathbf{v} \in \partial f(\overline{\mathbf{x}})$, *iff*

$$f(\mathbf{x}) \geq f(\overline{\mathbf{x}}) + < \mathbf{v}, \mathbf{x} - \overline{\mathbf{x}} >, \quad \forall \mathbf{x} \in \mathbf{R}^n. \tag{92}$$

**Definition 14** Closedness: Bertsekas (2009).
*If the epigraph of a function* $f : \mathbb{R}^n \to [-\infty, \infty]$ *is a closed set, we say that* $f$ *is a closed function.*

**Definition 15** Lower Semicontinuity: Bertsekas (2009).
*The function* $f : \mathbb{R}^n \to [-\infty, \infty]$ *is lower semicontinuous (lsc) at a point* $\overline{\mathbf{x}} \in \mathbb{R}^n$ *if*

$$f(\overline{\mathbf{x}}) \leq \liminf_{k \to \infty} f(\mathbf{x}_k), \tag{93}$$

*for every sequence* $\{\mathbf{x}_k\} \subset \mathbb{R}^n$ *with* $\mathbf{x}_k \to \overline{\mathbf{x}}$. *We say* $f$ *is lsc if it is lsc at each* $\overline{\mathbf{x}}$ *in its domain.*

**Theorem 16** Closedness and Semicontinuity: Bertsekas (2009) Proposition 1.1.2..
*For a function* $f : \mathbb{R}^n \to [-\infty, \infty]$, *the following are equivalent:*

1. *The level set* $V_\gamma = \{\mathbf{x} \,|\, f(\mathbf{x}) \leq \gamma\}$ *is closed for every scalar* $\gamma$.
2. $f$ *is lsc.*
3. $f$ *is closed.*

The following definition and theorem are from Rockafellar and Wets (1997) and they generalize the notion of subgradients to non-convex functions and the chain rule of differentiation, respectively. For complete statements see Rockafellar and Wets (1997) Rockafellar and Wets (1997).

**Definition 17** Regular Subgradient: Rockafellar and Wets (1997) Definition 8.3.
*Consider a function $f : \mathbb{R}^n \to [-\infty, \infty]$ and a point $\overline{\mathbf{x}}$ with $f(\overline{\mathbf{x}})$ finite. For a vector $\mathbf{v} \in \mathbf{R}^n$, one says that $\mathbf{v}$ is a regular subgradient of $f$ at $\overline{\mathbf{x}}$, written $\mathbf{v} \in \hat{\partial} f(\overline{\mathbf{x}})$, iff*

$$f(\mathbf{x}) \geq f(\overline{\mathbf{x}}) + \langle \mathbf{v}, \mathbf{x} - \overline{\mathbf{x}} \rangle + o(\mathbf{x} - \overline{\mathbf{x}}), \quad \forall \mathbf{x} \in \mathbf{R}^n, \tag{94}$$

*where the $o(t)$ notation indicates a term with the property that*

$$\lim_{t \to 0} \frac{o(t)}{t} = 0. \tag{95}$$

The relation of the regular subgradient defined above and the more familiar convex subgradient is the addition of the $o(\mathbf{x} - \overline{\mathbf{x}})$ term. Evidently, a convex subgradient is a regular subgradient.

**Theorem 18** Chain Rule for Regular Subgradients: Rockafellar and Wets (1997) Theorem 10.6.
*Suppose $f(\mathbf{x}) = g(F(\mathbf{x}))$ for a proper, lsc function $g : \mathbb{R}^m \to [-\infty, \infty]$ and a smooth mapping $F : \mathbb{R}^n \to \mathbb{R}^m$. Then at any point $\overline{\mathbf{x}} \in dom\, f = F^{-1}(dom\, g)$ one has*

$$\hat{\partial} f(\overline{\mathbf{x}}) \supset \nabla F(\overline{\mathbf{x}})^T \hat{\partial} g(F(\overline{\mathbf{x}})), \tag{96}$$

*where $\nabla F(\overline{\mathbf{x}})^T$ is the Jacobian of $F$ at $\overline{\mathbf{x}}$.*

**Theorem 19** Danskin's Theorem: Danskin (1966) and Bertsekas (1971) Proposition A.22.
*Suppose $\mathcal{Z} \subseteq \mathbb{R}^m$ is a compact set and $g(\mathbf{x}, \mathbf{z}) : \mathbb{R}^n \times \mathcal{Z} \to (-\infty, \infty]$ is a function. Suppose $g(\cdot, \mathbf{z}) : \mathbb{R}^n \to \mathbb{R}$ is closed proper convex function for every $\mathbf{z} \in \mathcal{Z}$. Further, define the function $f : \mathbb{R}^n \to \mathbb{R}$ such that*

$$f(\mathbf{x}) := \max_{\mathbf{z} \in \mathcal{Z}} g(\mathbf{x}, \mathbf{z}).$$

*Suppose $f$ is finite somewhere. Moreover, let $\mathcal{X} := int(dom f)$, i.e., the interior of the set of points in $\mathbb{R}^n$ such that $f$ is finite. Suppose $g$ is continuous on $\mathcal{X} \times \mathcal{Z}$. Further, define the set of maximizing points of $g(\mathbf{x}, \cdot)$ for each $\mathbf{x}$*

$$Z(\mathbf{x}) = \arg\max_{\mathbf{z} \in \mathcal{Z}} g(\mathbf{x}, \mathbf{z}).$$

*Then the following properties of $f$ hold.*

1. *The function $f(\mathbf{x})$ is a closed proper convex function.*
2. *For every $\mathbf{x} \in \mathcal{X}$,*

$$\partial f(\mathbf{x}) = conv\left\{\partial_{\mathbf{x}} g(\mathbf{x}, \mathbf{z}) \,|\, \mathbf{z} \in Z(\mathbf{x})\right\}. \tag{97}$$

**Corollary 20**
*Assume the conditions for Danskin's Theorem above hold. For every $\mathbf{x} \in \mathcal{X}$, if $Z(\mathbf{x})$ consists of a unique point, call it $\mathbf{z}^*$, and $g(\cdot, \mathbf{z}^*)$ is differentiable at $\mathbf{x}$, then $f(\cdot)$ is differentiable at $\mathbf{x}$, and*

$$\nabla f(\mathbf{x}) := \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{z}^*). \tag{98}$$

**Theorem 21** Bonnans and Shapiro (1998) Theorem 4.2, Rockafellar (1974) p. 41.
*Let* $\mathbf{X}$ *and* $\mathbf{U}$ *be Banach spaces. Let* $\mathbf{K}$ *be a closed convex cone in the Banach space* $\mathbf{U}$. *Let* $G : \mathbf{X} \to \mathbf{U}$ *be a convex mapping with respect to the cone* $\mathbf{C} := -\mathbf{K}$ *and* $f : \mathbf{X} \to (-\infty, \infty]$ *be a (possibly infinite-valued) convex function. Consider the following convex program and its optimal value function:*

$$v_P(\mathbf{u}) := \min_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x}) \qquad (P)$$
$$s.t. \quad G(\mathbf{x}) + \mathbf{u} \in \mathbf{K}.$$

*Moreover, consider the (Lagrangian) dual of the program:*

$$v_D(\mathbf{u}) := \max_{\lambda\in\mathbf{K}^-} \min_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x}) + \lambda^T(G(\mathbf{x}) + \mathbf{u}) \qquad (D)$$

*Suppose* $v_P(\mathbf{0})$ *is finite. Further, suppose the feasible set of the program is nonempty for all* $\mathbf{u}$ *in a neighborhood of* $\mathbf{0}$, *i.e.,*

$$\mathbf{0} \in int\{G(\mathbf{X}) - \mathbf{K}\}. \tag{99}$$

*Then,*

1. *There is no primal dual gap at* $u = 0$, *i.e.,* $v_P(0) = v_D(0)$.
2. *The set,* $\Lambda_0$, *of optimal solutions to the dual problem with* $\mathbf{u} = 0$ *is non-empty and bounded.*
3. *The optimal value function* $v_P(\mathbf{u})$ *is continuous at* $\mathbf{u} = 0$ *and* $\partial v_P(\mathbf{0}) = \Lambda_0$.

**Theorem 22** Bonnans and Shapiro (2000) Proposition 4.3.2.
*Consider two optimization problems over a non-empty feasible set* $\mathbf{\Omega}$:

$$\min_{\mathbf{x}\in\mathbf{\Omega}} f_1(\mathbf{x}) \qquad and \qquad \min_{\mathbf{x}\in\mathbf{\Omega}} f_2(\mathbf{x}) \tag{100}$$

*where* $f_1, f_2 : \mathcal{X} \to \mathbb{R}$. *Suppose* $f_1$ *has a non-empty set* $\mathbf{S}$ *of optimal solutions over* $\mathbf{\Omega}$. *Suppose the second order growth condition holds for* $\mathbf{S}$, *i.e., there exists a neighborhood* $\mathcal{N}$ *of* $\mathbf{S}$ *and a constant* $\alpha > 0$ *such that*

$$f_1(\mathbf{x}) \geq f_1(\mathbf{S}) + \alpha(dist(\mathbf{x}, \mathbf{S}))^2, \qquad \forall \mathbf{x} \in \mathbf{\Omega} \cap \mathcal{N}, \tag{101}$$

*where* $f_1(\mathbf{S}) := inf_{\mathbf{x}\in\mathbf{\Omega}} f_1(\mathbf{x})$. *Define the difference function:*

$$\Delta(\mathbf{x}) := f_2(\mathbf{x}) - f_1(\mathbf{x}). \tag{102}$$

*Suppose* $\Delta(\mathbf{x})$ *is L-Lipschitz continuous on* $\mathbf{\Omega} \cap \mathcal{N}$. *Let* $\mathbf{x}^* \in \mathcal{N}$ *be an* $\delta$*-solution to the problem of minimizing* $f_2(\mathbf{x})$ *over* $\mathbf{\Omega}$. *Then*

$$dist(\mathbf{x}^*, \mathbf{S}) \leq \frac{L}{\alpha} + \sqrt{\frac{\delta}{\alpha}}. \tag{103}$$

*Proofs* We provide proofs of theorems presented in the main paper and restate them here for completeness.

**Theorem 11**
*Suppose for any setting of* $\mathbf{w}_{nn} \in \mathbb{R}^{n_g}$ *there is a feasible solution to NeuPSL inference* (20)*. Further, suppose* $\epsilon > 0$, $\mathbf{w}_{sy} \in \mathbb{R}^r_+$, *and* $\mathbf{w}_{nn} \in \mathbb{R}^{n_g}$. *Then:*

- *The minimizer of* (20)*,* $\mathbf{y}^*(\mathbf{w}_{sy}, \mathbf{w}_{nn})$*, is a* $O(1/\epsilon)$ *Lipschitz continuous function of* $\mathbf{w}_{sy}$.
- $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$*, is concave over* $\mathbf{w}_{sy}$ *and convex over* $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$.
- $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ *is differentiable with respect to* $\mathbf{w}_{sy}$. *Moreover,*

$$\nabla_{\mathbf{w}_{sy}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) = \mathbf{\Phi}(\mathbf{y}^*(\mathbf{w}_{sy}, \mathbf{w}_{nn}), \mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})).$$

  *Furthermore,* $\nabla_{\mathbf{w}_{sy}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ *is Lipschitz continuous over* $\mathbf{w}_{sy}$.
- *If there is a feasible point* $\nu$ *strictly satisfying the* $i'th$ *inequality constraint of* (20)*, i.e.,* $\mathbf{A}[i]\nu + \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))[i] < 0$*, then* $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ *is subdifferentiable with respect to the* $i'th$ *constraint constant* $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))[i]$. *Moreover,*

$$\partial_{\mathbf{b}[i]} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) = \{\mu^*[i] \,|\, \mu^* \in \underset{\mu \in \mathbb{R}^{2 \cdot n_{\mathbf{y}} + m + q}_{\geq 0}}{\arg\min} h(\mu; \mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))\}$$

  *Furthermore, if* $\mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})$ *is a smooth function of* $\mathbf{w}_{nn}$*, then so is* $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$*, and the set of regular subgradients of* $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ *is:*

$$\begin{aligned} &\hat{\partial}_{\mathbf{w}_{nn}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) \\ &\quad \supset \nabla_{\mathbf{w}_{nn}} \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))^T \partial_{\mathbf{b}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))). \end{aligned} \tag{104}$$

**Proof of Theorem 9.** We first show the minimizer of the LCQP formulation of NeuPSL inference, $\nu^*$, with $\epsilon > 0$, $\mathbf{w}_{sy} \in \mathbb{R}^r_+$, and $\mathbf{w}_{nn} \in \mathbb{R}^{n_g}$ is a Lipschitz continuous function of $\mathbf{w}_{sy}$. Suppose $\epsilon > 0$ and $\mathbf{w}_{nn} \in \mathbb{R}^{n_g}$ is given. To show continuity over $\mathbf{w}_{sy} \in \mathbb{R}^r_+$, first note the matrix $(\mathbf{D} + \epsilon\mathbf{I})$ is positive definite and the primal inference problem (21) is an $\epsilon$-strongly convex LCQP with a unique minimizer denoted by $\nu^*(\mathbf{w}_{sy}, \mathbf{w}_{nn})$. We leverage the Lipschitz stability result for optimal values of constrained problems from Bonnans and Shapiro (2000) and presented here in Theorem 22. Define the primal objective as an explicit function of the weights:

$$f(\nu, \mathbf{w}_{sy}, \mathbf{w}_{nn}) := \nu^T(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon\mathbf{I})\nu + \mathbf{c}^T(\mathbf{w}_{sy})\nu \tag{105}$$

Note that the solution $\nu^* = \begin{bmatrix} \mathbf{s}_S^* \\ \mathbf{s}_L^* \\ \mathbf{y}^* \end{bmatrix}$ will always be bounded, since from (79c) in LCQP we always have for all $j \in I_S \cup I_L$,

$$\begin{aligned} 0 \leq s_j^* &= \max(\mathbf{a}_{\phi_k, y}^T \mathbf{y}^* + \mathbf{a}_{\phi_k, \mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{\phi_k, \mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_{\phi_k}, 0) && (106) \\ &\leq \|\mathbf{a}_{\phi_k, y}\| + |\mathbf{a}_{\phi_k, \mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{\phi_k, \mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_{\phi_k}|. && (107) \end{aligned}$$

Thus, setting these trivial upper bounds for $s_j$ will not change the solution of the problem. We can henceforth consider the problem in a bounded domain $\|\nu\| \leq C$ where $C$ does not depend on $\mathbf{w}$'s.

Let $\mathbf{w}_{1,sy}, \mathbf{w}_{2,sy} \in \mathbb{R}^r_+$ and $\mathbf{w}_{nn} \in \mathcal{W}_{nn}$ be arbitrary. As $\epsilon > 0$, $f(\nu, \mathbf{w}_{1,sy}, \mathbf{w}_{nn})$ is strongly convex in $\nu$ and it therefore satisfies the second-order growth condition in $\nu$. Define the difference function:

$$\Delta_{\mathbf{w}_{sy}}(\nu) := f(\nu, \mathbf{w}_{2,sy}, \mathbf{w}_{nn}) - f(\nu, \mathbf{w}_{1,sy}, \mathbf{w}_{nn}) \tag{108}$$

$$= \nu^T(\mathbf{D}(\mathbf{w}_{2,sy}) + \epsilon\mathbf{I})\nu + \mathbf{c}^T(\mathbf{w}_{2,sy})\nu - \left(\nu^T(\mathbf{D}(\mathbf{w}_{1,sy}) + \epsilon\mathbf{I})\nu + \mathbf{c}^T(\mathbf{w}_{1,sy})\nu\right) \tag{109}$$

$$= \nu^T(\mathbf{D}(\mathbf{w}_{2,sy}) - \mathbf{D}(\mathbf{w}_{1,sy}))\nu + (\mathbf{c}(\mathbf{w}_{2,sy}) - \mathbf{c}(\mathbf{w}_{1,sy}))^T\nu. \tag{110}$$

The difference function $\Delta_{\mathbf{w}_{sy}}(\nu)$ over $\mathcal{N}$ has a finitely bounded gradient:

$$\|\nabla\Delta_{\mathbf{w}_{sy}}(\nu)\|_2 = \left\|2(\mathbf{D}(\mathbf{w}_{2,sy}) - \mathbf{D}(\mathbf{w}_{1,sy}))\nu + \mathbf{c}(\mathbf{w}_{2,sy}) - \mathbf{c}(\mathbf{w}_{1,sy})\right\|_2 \tag{111}$$

$$\leq \|\mathbf{c}(\mathbf{w}_{2,sy}) - \mathbf{c}(\mathbf{w}_{1,sy})\|_2 + 2\|(\mathbf{D}(\mathbf{w}_{2,sy}) - \mathbf{D}(\mathbf{w}_{1,sy}))\nu\|_2 \tag{112}$$

$$\leq \|\mathbf{w}_{2,sy} - \mathbf{w}_{1,sy}\|_2 + 2\|\mathbf{w}_{2,sy} - \mathbf{w}_{1,sy}\|_2 \, \|\nu\|_2 \tag{113}$$

$$\leq \|\mathbf{w}_{2,sy} - \mathbf{w}_{1,sy}\|_2(1 + 2C) =: L_{\mathcal{N}}(\mathbf{w}_{1,sy}, \mathbf{w}_{2,sy}). \tag{114}$$

Thus, the distance function, $\Delta_{\mathbf{w}_{sy}}(\nu)$ is $L_{\mathcal{N}}(\mathbf{w}_{1,sy}, \mathbf{w}_{2,sy})$-Lipschitz continuous over $\mathcal{N}$. Therefore, by Bonnans and Shapiro (2000) (Theorem 22), the distance between $\nu^*(\mathbf{w}_{1,sy}, \mathbf{w}_{nn})$ and $\nu^*(\mathbf{w}_{2,sy}, \mathbf{w}_{nn})$ is bounded above:

$$\|\nu^*(\mathbf{w}_{2,sy}, \mathbf{w}_{nn}) - \nu^*(\mathbf{w}_{1,sy}, \mathbf{w}_{nn})\|_2 \leq \frac{L_{\mathcal{N}}(\mathbf{w}_{1,sy}, \mathbf{w}_{2,sy})}{\epsilon} = \frac{(1 + 2C)}{\epsilon}\|\mathbf{w}_{2,sy} - \mathbf{w}_{1,sy}\|_2. \tag{115}$$

Therefore, the function $\nu^*(\mathbf{w}_{sy}, \mathbf{w}_{nn})$ is $O(1/\epsilon)$-Lipschitz continuous in $\mathbf{w}_{sy}$ for any $\mathbf{w}_{nn}$.

Next, we prove curvature properties of the value-function with respect to the weights. Observe NeuPSL inference is an infimum over a set of functions that are concave (affine) in $\mathbf{w}_{sy}$. Therefore, by Theorem 12, we have that $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ is concave in $\mathbf{w}_{sy}$.

We use a similar argument to show $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ is convex in the constraint constants, $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$. Assuming for any setting of the neural weights, $\mathbf{w}_{nn} \in \mathbb{R}^{n_g}$, there is a feasible solution to the NeuPSL inference problem, then (20) satisfies the conditions for Slater's constraint qualification. Therefore, strong duality holds, i.e., $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ is equal to the optimal value of the dual inference problem (86). Observe that the dual NeuPSL inference problem is a supremum over a set of functions convex (affine) in $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$. Therefore, by Theorem 11, we have that $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$ is convex in $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$.

We can additionally prove convexity in $\mathbf{b}$ from first principles. For simplicity, fix other parameters, and write the objective and the value function as $Q(\nu)$ and $V(\mathbf{b})$, respectively.

Let us first consider the domain where the optimization is bounded and the optimal solution exists. Given $\mathbf{b}_1$ and $\mathbf{b}_2$, let the corresponding optimal solutions of (85) parameterized by $\mathbf{b}_1$ and $\mathbf{b}_2$ be $\nu_1$ and $\nu_2$. Take any $\alpha \in [0,1]$, note that $\alpha\nu_1 + (1-\alpha)\nu_2$ is feasible for the optimization problem parameterized by $\mathbf{b} = \alpha\mathbf{b}_1 + (1-\alpha)\mathbf{b}_2$. Because we take the inf over all $\nu$s, the optimal $\nu$ for this $\mathbf{b}$ might be even smaller. Thus, we have (for convex quadratic objective $Q$) that

$$\begin{aligned} V(\alpha\mathbf{b}_1 + (1-\alpha)\mathbf{b}_2) &\leq Q(\alpha\nu_1 + (1-\alpha)\nu_2) \\ &\leq \alpha Q(\nu_1) + (1-\alpha)Q(\nu_2) \\ &= \alpha V(\mathbf{b}_1) + (1-\alpha)V(\mathbf{b}_2), \end{aligned} \tag{116}$$

which shows that $V$ is convex in $\mathbf{b}$. To establish the convexity when $V(\mathbf{b})$ takes extended real-values ($\mathbb{R} \cup \{-\infty\}$) to allow for unbounded optimization problems, it suffices to consider sequences $\{\nu_i^k\}_{k=1}^{\infty}$ for $\mathbf{b}_i$ ($i = 1, 2$, $\mathbf{b}_1 \neq \mathbf{b}_2$) as follows:

(1) If $V(\mathbf{b}_i)$ is finite, let $\nu_i^k = \nu_i$ for all $k$, where $\nu_i$ is the optimal solution.

(2) If $V(\mathbf{b}_i) = -\infty$, there exists sequence $\{\nu_i^k\}_{k=1}^{\infty}$ such that $Q(\nu_i^k) \to -\infty$ as $k \to \infty$.

Now, for any $0 < \alpha < 1$, observe:

Case 1: Both $V(\mathbf{b}_1)$ and $V(\mathbf{b}_2)$ are finite. We can reuse the argument above.

Case 2: At least one of $V(\mathbf{b}_1)$ and $V(\mathbf{b}_2)$ is $-\infty$. By convexity of $Q$, $Q(\alpha\nu_1^k + (1-\alpha)\nu_2^k) \leq \alpha Q(\nu_1^k) + (1-\alpha)Q(\nu_2^k)$. Therefore, we have $Q(\alpha\nu_1^k + (1-\alpha)\nu_2^k) \to -\infty$ as $k \to \infty$ when $0 < \alpha < 1$. Note that for all $k$, $\alpha\nu_1^k + (1-\alpha)\nu_2^k$ is feasible for the optimization problem parameterized by $\mathbf{b} = \alpha\mathbf{b}_1 + (1-\alpha)\mathbf{b}_2$. It follows that $V(\alpha\mathbf{b}_1 + (1-\alpha)\mathbf{b}_2) = -\infty$.

Therefore, convexity holds when $V(\mathbf{b})$ takes extended real-values ($\mathbb{R} \cup \{-\infty\}$).

Next, we prove (sub)differentiability properties of the value-function. Suppose $\epsilon > 0$. First, we show the optimal value function, $V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))$, is differentiable with respect to the symbolic weights. Then we show subdifferentiability properties of the optimal value function with respect to the constraint constants. Finally, we apply the Lipschitz continuity of the minimzer result to show the gradient of the optimal value function is Lipschitz continuous with respect to $\mathbf{w}_{sy}$.

Starting with differentiability with respect to the symbolic weights, $\mathbf{w}_{sy}$, note, the optimal value function of the regularized LCQP formulation of NeuPSL inference, (20), is equivalently expressed as the following maximization over a continuous function in the primal target variables, $\mathbf{y}$, the slack variables, $\mathbf{s}_S$ and $\mathbf{s}_L$, and the symbolic weights, $\mathbf{w}_{sy}$:

$$\begin{aligned} &V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) \\ &= -\left( \max_{\mathbf{y}, \mathbf{s_H}, \mathbf{s_L}} -\left( \begin{bmatrix} \mathbf{s}_S \\ \mathbf{s}_L \\ \mathbf{y} \end{bmatrix}^T \begin{bmatrix} \mathbf{W}_S + \epsilon I & 0 & 0 \\ 0 & \epsilon I & 0 \\ 0 & 0 & \epsilon I \end{bmatrix} \begin{bmatrix} \mathbf{s}_S \\ \mathbf{s}_L \\ \mathbf{y} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{w}_L \\ 0 \end{bmatrix}^T \begin{bmatrix} \mathbf{s}_S \\ \mathbf{s}_L \\ \mathbf{y} \end{bmatrix} \right) \right) \\ &\quad \text{s.t.} \quad \mathbf{A} \begin{bmatrix} \mathbf{s}_S \\ \mathbf{s}_L \\ \mathbf{y} \end{bmatrix} + \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) \leq 0, \end{aligned} \tag{117}$$

where the matrix $\mathbf{W}_s$ and vector $\mathbf{w}_L$ are functions of the symbolic parameters $\mathbf{w}_{sy}$ as defined in (78). Moreover, the objective above is and convex (affine) in $\mathbf{w}_{sy}$. Additionally, note that the decision variables can be constrained to a compact domain without breaking the equivalence of the formulation. Specifically, the target variables are constrained to the box $[0,1]^{\mathbf{n}_y}$, while the slack variables are nonnegative and have a trivial upper bound derived from (79c):,

$$\begin{aligned} 0 \leq s_j^* &= \max(\mathbf{a}_{\phi_k,y}^T \mathbf{y}^* + \mathbf{a}_{\phi_k,\mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{\phi_k,\mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_{\phi_k}, 0) \\ &\leq \|\mathbf{a}_{\phi_k,y}\| + |\mathbf{a}_{\phi_k,\mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{\phi_k,\mathbf{g}_{nn}}^T \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}) + b_{\phi_k}|, \end{aligned} \tag{118}$$

for all $j \in I_S \cup I_L$. Therefore, the negative optimal value function satisfies the conditions for Danskin's theorem Danskin (1966) (stated in Appendix D). Moreover, as there is a single unique solution to the inference problem when $\epsilon > 0$, and the quadratic objective in (20) is differentiable for all $\mathbf{w}_{sy} \in \mathbb{R}_+^r$, we can apply Corollary 20. The optimal value function is therefore concave and differentiable with respect to the symbolic weights with

$$\nabla_{\mathbf{w}_{sy}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) = \mathbf{\Phi}(\mathbf{y}^*, \mathbf{x_{sy}}, \mathbf{g_{nn}}(\mathbf{x_{nn}}, \mathbf{w_{nn}})). \tag{119}$$

Next, we show subdifferentiability of the optimal value-function with respect to the constraint constants, $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$. Suppose at a setting of the neural weights $\mathbf{w}_{nn} \in \mathbb{R}^{n_g}$ there is a feasible point $\nu$ for the NeuPSL inference problem. Moreover, suppose $\nu$ strictly satisfies the $i'th$ inequality constraint of (20), i.e., $\mathbf{A}[i]\nu + \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))[i] < 0$. Observe that the following strongly convex conic program is equivalent to the LCQP formulation of NeuPSL inference, (20):

$$\begin{aligned} \min_{\nu \in \mathbb{R}^{n_\mathbf{y}+m_S+m_L}} \quad & \nu^T(\mathbf{D}(\mathbf{w}_{sy}) + \epsilon\mathbf{I})\nu + \mathbf{c}(\mathbf{w}_{sy})^T\nu + P_{\Omega\setminus i}(\nu) \\ \text{s.t.} \quad & \mathbf{A}[i]\nu + \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))[i] \in \mathbb{R}_{\leq 0}, \end{aligned} \tag{120}$$

where $P_{\Omega\setminus i}(\nu) : \mathbb{R}^{n_\mathbf{y}+m_S+m_L} \to \{0, \infty\}$ is the indicator function identifying feasibility w.r.t. all the constraints of the LCQP formulation of NeuPSL inference in (20) except the $i'th$ constraint: $\mathbf{A}[i]\nu + \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))[i] \leq 0$. In other words, in the conic formulation above only the $i'th$ constraint is explicit. Note that $\mathbb{R}_{\leq 0}$ is a closed convex cone in $\mathbb{R}$. Moreover, both the objective in the program and the mapping $G(\nu) := \mathbf{A}[i]\nu + \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))[i]$ are convex. Lastly, note the constraint qualification (99) is similar to Slater's condition in the case of (120) which is satisfied by the supposition there exists a feasible $\nu$ that strictly satisfies the $i'th$ inequality constraint of (20). Therefore, (120) satisfies the conditions of Theorem 21. Thus, the value function is continuous in the constraint constant $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))[i]$ at $\mathbf{w}_{nn}$ and

$$\begin{aligned} &\partial_{\mathbf{b}[i]} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))) \\ &\quad = \{\mu^*[i] \,|\, \mu^* \in \operatorname*{arg\,min}_{\mu \in \mathbb{R}_{\geq 0}^{2 \cdot n_\mathbf{y}+m+q}} h(\mu; \mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})))\}. \end{aligned} \tag{121}$$

Moreover, when $\mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))$ is a smooth function of the neural weights $\mathbf{w}_{nn}$, then we can apply the chain rule for regular subgradients, Theorem 18, to get

$$\begin{aligned} &\hat{\partial}_{\mathbf{w}_{nn}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \\ &\quad \supset \nabla \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})^T \partial_{\mathbf{b}} V(\mathbf{w}_{sy}, \mathbf{b}(\mathbf{x}_{sy}, \mathbf{g}_{nn}(\mathbf{x}_{nn}, \mathbf{w}_{nn})). \end{aligned} \tag{122}$$

To prove the optimal value function is Lipschitz smooth over $\mathbf{w}_{sy}$, it is equivalent to show it is continuously differentiable and that all gradients have bounded magnitude. To show the value function is continuously differentiable, we first apply the result asserting the minimizer is unique and a continuous function of the symbolic parameters $\mathbf{w}_{sy}$. Therefore, the optimal value function gradient is a composition of continuous functions, hence continuous in $\mathbf{w}_{sy}$. The fact that the value function has a bounded gradient magnitude follows from the fact that the decision variables $\mathbf{y}$ have a compact domain over which the gradient is finite; hence a trivial and finite upper bound exists on the gradient magnitude.

## E Extended Empirical Analysis

In this section, we provide additional information on the empirical analysis. The subsequent subsections will examine the modular datasets and the hyperparameters employed for each experiment. Additional model details, including neural model architectures and symbolic model constraints, can be found at https://github.com/linqs/dickens-arxiv24.

### *Modular Datasets*

- **4Forums and CreateDebate**: Stance-4Forums and Stance-CreateDebate are two datasets containing dialogues from online debate sites: 4forums.com and createdebate.com, respectively. In this paper, we study stance classification, i.e., the task of identifying the stance of a speaker in a debate as being for or against.

- **Epinions**: Epinions is a trust network with $2,000$ individuals connected by $8,675$ directed edges representing whether they know each other and whether they trust each other Richardson et al. (2003). We study link prediction, i.e., we predict if two individuals trust each other.

  In each of the $5$ data splits, the entire network is available, and the prediction performance is measured on $\frac{1}{8}$ of the trust labels. The remaining set of labels are available for training. We use The NeuPSL model from Bach et al. (2017). The data and NeuPSL model are available at https://github.com/linqs/psl-examples/tree/main/epinions.

- **Citeseer and Cora**: Citeseer and Cora are citation networks introduced by Sen et al. (2008). For Citeseer, $3,312$ documents are connected by $4,732$ edges representing citation links. For Cora, $2,708$ documents are connected by $5,429$ edges representing citation links. We study node classification, i.e., we classify the documents into one of $6$ topics for Citeseer and $7$ topics for Cora.

For each of the 10 folds, we randomly sample 5% of the node labels for training 5% of the node labels for validation and 1,000 for testing. The models for modular learning performance experiments are extended versions from Bach et al. (2017) Bach et al. (2017). Specifically, a copy of each rule is made that is specialized for the topic. Moreover, topic propagation across citation links is considered for papers with differing topics. For instance, the possibility of a citation from a paper with topic 'A' could imply a paper is more or less likely to be topic 'B'. The extended models are available at https://github.com/linqs/dickens-arxiv24/tree/main/modular_learning/psl-extended-examples. The models for learning prediction performance experiments are from Pryor et al. (2023a). The data and models are available at: https://github.com/linqs/dickens-arxiv24/tree/main/citation/models/symbolic.

- **DDI**: Drug-drug interaction (DDI) is a network of 315 drugs and 4,293 interactions derived from the DrugBank database (Wishart et al. 2006). The edges in the drug network represent interactions and seven different similarity metrics. In this paper, we perform link prediction, i.e., we infer unknown drug-drug interactions.

  The 5 data splits and the NeuPSL model we evaluate in this paper originated from Sridhar et al. (2016). The data and NeuPSL models are available at: https://github.com/linqs/psl-examples/tree/main/drug-drug-interaction.

- **Yelp**: Yelp is a network of 34,454 users and 3,605 items connected by 99,049 edges representing ratings. The task is to predict missing ratings, i.e., regression, which could be used in a recommendation system.

  In each of the 5 folds, 80% of the ratings are randomly sampled and available for training, and the remaining 20% is held out for testing. We use The NeuPSL model from Kouki et al. (2015). The data and NeuPSL model are available at: https://github.com/linqs/psl-examples/tree/main/yelp.

## *Hyperparameters*

The hyperparameter ranges were decided upon based on the results presented in Pryor et al. (2023a), Dickens et al. (2024a), and Dickens et al. (2024b). For the complete set of hyperparameter settings, please refer to the original papers or visit *https://github.com/linqs/dickens-arxiv24*.

## References

Abraham SS, Alirezaie M and Raedt LD (2024) Clevr-poc: Reasoning-intensive visual question answering in partially observable environments. *arXiv* .

Ackley D, Hinton G and Sejnowski T (1985) A learning algorithm for boltzmann machines. *Cognitive Science* 9(1): 147–169.

Agrawal A, Amos B, Barratt S, Boyd S, Diamond S and Kolter J (2019a) Differentiable convex optimization layers. In: *NeurIPS*.

**Table 12.** Hyperparameter ranges and final values for the NeSy-EBM learning experiments.

| | Algorithm | Parameter | Range | Final Value |
|---|---|---|---|---|
| *MNIST-Add1* | **Energy** | **Neural Learning Rate** | $\{10^{-3}, 10^{-4}, 10^{-5}\}$ | $10^{-4}$ |
| | **Bilevel** | **Energy Loss Coefficient** | $\{10^{-1}, 1, 10\}$ | $10$ |
| | | **Neural Learning Rate** | $\{10^{-3}, 10^{-4}, 10^{-5}\}$ | $10^{-4}$ |
| | **Policy** | **Energy Loss Coefficient** | $\{10^{-1}, 1, 10\}$ | $10$ |
| | | **Neural Learning Rate** | $\{10^{-3}, 10^{-4}, 10^{-5}\}$ | $10^{-4}$ |
| *MNIST-Add2* | **Energy** | **Neural Learning Rate** | $\{10^{-3}, 10^{-4}, 10^{-5}\}$ | $10^{-4}$ |
| | **Bilevel** | **Energy Loss Coefficient** | $\{10^{-1}, 1, 10\}$ | $10$ |
| | | **Neural Learning Rate** | $\{10^{-3}, 10^{-4}, 10^{-5}\}$ | $10^{-4}$ |
| | **Policy** | **Energy Loss Coefficient** | $\{10^{-1}, 1, 10\}$ | $10$ |
| | | **Neural Learning Rate** | $\{10^{-3}, 10^{-4}, 10^{-5}\}$ | $10^{-4}$ |
| *Visual-Sudoku* | **Energy** | **Neural Learning Rate** | $\{10^{-3}, 10^{-4}, 10^{-5}\}$ | $10^{-4}$ |
| | | **Alpha** | $\{0.1, 0.5, 0.9\}$ | $0.1$ |
| | **Bilevel** | **Energy Loss Coefficient** | $\{10^{-1}, 1, 10\}$ | $10$ |
| | | **Neural Learning Rate** | $\{10^{-3}, 10^{-4}, 10^{-5}\}$ | $10^{-3}$ |
| | | **Alpha** | $\{0.1, 0.5, 0.9\}$ | $0.1$ |
| | **Policy** | **Energy Loss Coefficient** | $\{10^{-1}, 1, 10\}$ | $10$ |
| | | **Neural Learning Rate** | $\{10^{-3}, 10^{-4}, 10^{-5}\}$ | $10^{-3}$ |
| | | **Alpha** | $\{0.1, 0.5, 0.9\}$ | $0.1$ |
| *Path-Finding* | **Energy** | **Neural Learning Rate** | $\{10^{-3}, 10^{-4}, 10^{-5}\}$ | $10^{-3}$ |
| | **Bilevel** | **Energy Loss Coefficient** | $\{10^{-1}, 1\}$ | $1$ |
| | | **Neural Learning Rate** | $\{5^{-4}, 10^{-4}, 10^{-5}\}$ | $5^{-4}$ |
| | **Policy** | **Energy Loss Coefficient** | $\{10^{-1}, 1\}$ | $1$ |
| | | **Neural Learning Rate** | $\{5^{-4}, 10^{-4}, 10^{-5}\}$ | $5^{-4}$ |
| | | **Alpha** | $\{0.1, 0.5, 0.9\}$ | $0.1$ |
| *Citeseer* | **Energy** | **Neural Learning Rate** | $\{10^{-1}, 10^{-2}, 10^{-3}\}$ | $10^{-3}$ |
| | | **Step Size** | $\{10^{-1}, 10^{-2}, 10^{-3}\}$ | $10^{-3}$ |
| | **Bilevel** | **Energy Loss Coefficient** | $\{0, 10^{-1}, 1, 10\}$ | $1$ |
| | | **Neural Learning Rate** | $\{10^{-1}, 10^{-2}, 10^{-3}\}$ | $10^{-3}$ |
| | | **Step Size** | $\{10^{-1}, 10^{-2}, 10^{-3}\}$ | $10^{-3}$ |
| | **Policy** | **Energy Loss Coefficient** | $\{0, 10^{-1}, 1, 10\}$ | $1$ |
| | | **Neural Learning Rate** | $\{10^{-1}, 10^{-2}, 10^{-3}\}$ | $10^{-3}$ |
| | | **Alpha** | $\{0.1, 0.5, 0.9\}$ | $0.1$ |
| *Citeseer* | **Energy** | **Neural Learning Rate** | $\{10^{-1}, 10^{-2}, 10^{-3}\}$ | $10^{-3}$ |
| | | **Step Size** | $\{10^{-1}, 10^{-2}, 10^{-3}\}$ | $10^{-3}$ |
| | **Bilevel** | **Energy Loss Coefficient** | $\{0, 10^{-1}, 1, 10\}$ | $1$ |
| | | **Neural Learning Rate** | $\{10^{-1}, 10^{-2}, 10^{-3}\}$ | $10^{-3}$ |
| | | **Step Size** | $\{10^{-1}, 10^{-2}, 10^{-3}\}$ | $10^{-3}$ |
| | **Policy** | **Energy Loss Coefficient** | $\{0, 10^{-1}, 1, 10\}$ | $1$ |
| | | **Neural Learning Rate** | $\{10^{-1}, 10^{-2}, 10^{-3}\}$ | $10^{-3}$ |
| | | **Alpha** | $\{0.1, 0.5, 0.9\}$ | $0.1$ |

Agrawal A, Barratt S, Boyd S, Busseti E and M Moursi W (2019b) Differentiating through a cone program. *Journal of Applied and Numerical Optimization* 1(2): 107–115.

Ahmed K, Chang KW and den Broeck GV (2023a) Semantic strengthening of neuro-symbolic learning. In: *AISTATS*.

Ahmed K, Chang KW and Van den Broeck G (2023b) A pseudo-semantic loss for autoregressive models with logical constraints. In: *NeurIPS*.

Ahmed K, Teso S, Chang KW, Van den Broeck G and Vergari A (2022a) Semantic probabilistic layers for neuro-symbolic learning. In: *NeurIPS*.

Ahmed K, Wang E, Chang KW and den Broeck GV (2022b) Neuro-symbolic entropy regularization. In: *UAI*.

Amos B and Kolter J (2017) Optnet: Differentiable optimization as a layer in neural networks. In: *ICML*.

Arrotta L, Civitarese G and Bettini C (2024) Semantic loss: A new neuro-symbolic approach for context-aware human activity recognition. *Proceeding of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7(144): 1–29.

Augustine E, Pryor C, Dickens C, Pujara J, Wang WY and Getoor L (2022) Visual sudoku puzzle classification: A suite of collective neuro-symbolic tasks. In: *International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*.

Bach S, Broecheler M, Huang B and Getoor L (2017) Hinge-loss Markov random fields and probabilistic soft logic. *Journal of Machine Learning Research (JMLR)* 18(1): 1–67.

Bader S and Hitzler P (2005) Dimensions of neural-symbolic integration - A structured survey. *ArXiv* .

Badreddine S, d'Avila Garcez A, Serafini L and Spranger M (2022) Logic tensor networks. *AI* 303(4): 103649.

Badreddine S, Serafini L and Spranger M (2023) logltn: Differentiable fuzzy logic in the logarithm space. *arXiv* .

Belanger D, Yang B and McCallum A (2017) End-to-end learning for structure prediction energy networks. In: *ICML*.

Bertsekas D (1971) *Control of Uncertain Systems with a Set-Membership Description of Uncertainty*. PhD Thesis, MIT.

Bertsekas D (2009) *Convex Optimization Theory*. Athena Scientific.

Besold TR, d'Avila Garcez AS, Bader S, Bowman H, Domingos PM, Hitzler P, Kühnberger K, Lamb LC, Lowd D, Lima PMV, de Penning L, Pinkas G, Poon H and Zaverucha G (2022) Neural-symbolic learning and reasoning: A survey and interpretation. *Neuro-Symbolic Artificial Intelligence: The State of the Art* .

Bommasani R, Hudson D, Adeli E, Altman R, Arora S, von Arx S, S Bernstein M, Bohg J, Bosselut A, Brunskill E and et al (2022) On the opportunities and risks of foundation models. *Arxiv* .

Bonnans J and Shapiro A (1998) Optimization problems with perturbations: A guided tour. *SIAM Review* 40(2): 228–264.

Bonnans J and Shapiro A (2000) *Perturbation Analysis of Optimization Problems*. Springer.

Bošnjak M, Rocktäschel T, Naradowsky J and Riedel S (2017) Programming with a differentiable forth interpreter. In: *ICML*.

Boyd S and Vandenberghe L (2004) *Convex Optimization*. Cambridge University Press.

Bracken J and McGill JT (1973) Mathematical programs with optimization problems in the constraints. *Operations Research* 21(1): 37–44.

Brewka G, Eiter T and Truszczynski M (2011) Answer set programming at a glance. *Communication of the ACM* 54(12): 92–103.

Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A and Zagoruyko S (2020) End-to-end object detection with transformers. In: *European conference on computer vision*.

Carraro T, Daniele A, Aiolli F and Serafini L (2022) Logic tensor networks for top-n recommendation. In: *International Conference of the Italian Association for Artificial Intelligence (AIxIA)*.

Chang MW, Ratinov L and Roth D (2007) Guiding semi-supervision with constraint-driven learning. In: *ACL*.

Chavira M and Darwiche A (2008) On probabilistic inference by weighted model counting. *Artificial Intelligence* 172(6-7): 772–799.

Chen T, Kornblith S, Norouzi M and Hinton G (2020) A simple framework for contrastive learning of visual representations. In: *ICML*.

Choi Y, Vergari A and Van den Broeck G (2020) Probabilistic circuits: A unifying framework for tractable probabilistic modeling. UCLA.

Cohen WW, Yang F and Mazaitis K (2020) Tensorlog: A probabilistic database implemented using deep-learning infrastructure. *JAIR* 67: 285–325.

Collins M (2002) Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In: *EMNLP*.

Colson B, Marcotte P and Savard G (2007) An overview of bilevel optimization. *Annals of Operations Research* 153(1): 235–256.

Cornelio C, Stuehmer J, Xu Hu S and Hospedales T (2023) Learning where and when to reason in neuro-symbolic inference. In: *ICLR*.

Cunnington D, Law M, Lobo J and Russo A (2024) The role of foundation models in neuro-symbolic learning and reasoning. *arXiv* .

Danskin J (1966) The theory of max-min, with applications. *SIAM Journal on Applied Mathematics* 14(4): 641–664.

Dash T, Chitlangia S, Ahuja A and Srinivasan A (2022) A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Scientific Reports* 12(1): 1040.

d'Avila Garcez A, Gori M, Lamb LC, Serafini L, Spranger M and Tran SN (2019) Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics* 6(4): 611–632.

d'Avila Garcez AS, Broda K and Gabbay DM (2002) *Neural-Symbolic Learning Systems: Foundations and Applications*. Springer.

d'Avila Garcez AS, Lamb LC and Gabbay DM (2009) *Neural-Symbolic Cognitive Reasoning*. Springer.

Dayan P, Hinton G, Neal R and Zemel R (1995) The helmholtz machine. *Neural Computation* 7(5): 889–904.

De Raedt L, Dumančić S, Manhaeve R and Marra G (2020) From statistical relational to neuro-symbolic artificial intelligence. In: *IJCAI*.

De Raedt L, Kimmig A and Toivonen H (2007) Problog: A probabilistic prolog and its application in link discovery. In: *IJCAI*.

De Smet L, Sansone E and Zuidberg Dos Martires P (2023) Differetiable sample of categorical distributions using the catlog-derivative trick. In: *NeurIPS*.

Demeester T, Rocktäschel T and Riedel S (2016) Lifted rule injection for relation embeddings. In: *EMNLP*.

Dempe S and Zemkoho A (2020) *Bilevel Optimization*. Springer.

Derkinderen V, Manhaeve R, Martires PZD and Raedt LD (2024) Semirings for probabilistic and neuro-symbolic logic programming. *International Journal of Approximate Reasoning* : 109130.

Devlin J, Chang M, Lee K and Toutanova K (2019) Bert: Pre-training of deep bidirectional transformers for language understanding. *Arxiv* .

Dickens C, Gao C, Pryor C, Wright S and Getoor L (2024a) Convex and bilevel optimization for neuro-symbolic inference and learning. In: *ICML*.

Dickens C, Pryor C and Getoor L (2024b) Modeling patterns for neural-symbolic reasoning using energy-based models. In: *AAAI Spring Symposium on Empowering Machine Learning and Large Language Models with Domain and Commonsense Knowledge*.

Diligenti M, Gori M and Saccà C (2017a) Semantic-based regularization for learning and inference. *Journal of Machine Learning Research* 18: 1–45.

Diligenti M, Roychowdhury S and Gori M (2017b) Integrating prior knowledge into deep learning. In: *ICMLA*.

Do C, Foo CS and Ng A (2007) Efficient multiple hyperparameter learning for log-linear models. In: *NeurIPS*.

Domke J (2012) Generic methods for optimization-based modeling. In: *AISTATS*.

Donadello I, Serafini L and d'Avila Garcez AS (2017) Logic tensor networks for semantic image interpretation. In: *IJCAI*.

Du Y, Durkan C, Strudel R, Tenenbaum J, Dieleman S, Fergus R, Sohl-Dickstein J, Doucet A and Grathwohl W (2023) Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In: *ICML*.

Du Y and Mordatch I (2019) Implicit generation and modeling with energy-based models. In: *NeurIPS*.

E van Engelen J and H Hoos H (2020) A survey on semi-supervised learning. *Machine Learning (ML)* 109: 373–440.

F Bard J (2013) *Practical Bilevel Optimization: Algorithms and Applications*. Springer Science & Business Media.

Feng J, Xu R, Hao J, Sharma H, Shen Y, Zhao D and Chen W (2024) Language models can be deductive solvers. In: *NAACL*.

Fiacco A and McCormick G (1968) *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley and Sons.

Franceschi L, Frasconi P, Salzo S, Grazzi R and Pontil M (2018) Bilevel programming for hyperparameter optimization and meta-learning. In: *ICML*.

Ghadimi S and Wang M (2018) Approximation methods for bilevel programming. *Arxiv* .

Giovannelli T, Kent G and Nune Vicente L (2022) Inexact bilevel stochastic gradient methods for constrained and unconstrained lower-level problems. *Arxiv* .

Giunchiglia E, Catalina Stoian M, Khan S and Lukasiewicz T (2023) Road-r: The autonomous driving dataset with logical requirements. *Machine Learning* 112(1): 3261–3291.

Giunchiglia E, Stoian MC and Lukasiewicz T (2022) Deep learning with logical constraints. In: *International Joint Conference on Artificial Intelligence (IJCAI)*.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y (2014) Generative adversarial nets. In: *NeurIPS*.

Grathwohl W, Wang K, Jacobsen J, Duvenaud D, Norouzi M and Swersky K (2020) Your classifier is secretly an energy-based model and you should treat it like one. In: *ICLR*.

Griewank A and Walther A (2008) *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM.

Gurobi Optimization L (2024) Gurobi optimizer reference manual. URL https://www.gurobi.com.

H Papadimitriou C and Steiglitz K (1998) *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation.

Hasan KS and Ng V (2013) Stance classification of ideological debates: Data, models, features, and constraints. In: *IJCNLP*.

He K, Zhang X, Ren S and Sun J (2016) Deep residual learning for image recognition. In: *CVPR*.

Hinton G (2002) Training products of experts by minimizing contrastive divergence. *Neural Computation* 14(8): 1771–1800.

Hyvarinen A (2005) Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research (JMLR)* 6: 695–709.

J Hu E, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, Wang L and Chen W (2022) Lora: Low-rank adaptation of large language models. In: *ICLR*.

J Ye J and L Zhu D (1995) Optimality conditions for bilevel programming problems. *Optimization* 33(1): 9–27.

Ji K, Yang J and Liang Y (2021) Bilevel optimization: Convergence analysis and enhanced design. In: *ICML*.

Khanduri P, Tsaknakis I, Zhang Y, Liu J, Liu S, Zhang J and Hong M (2023) Linearly constrained bilevel optimization: A smoothed implicit gradient approach. In: *ICML*.

Kisa D, den Broeck GV, Choi A and Darwiche A (2014) Probabilistic sentential decision diagrams. In: *KR*.

Klir GJ and Yuan B (1995) *Fuzzy Sets and Fuzzy Logic - Theory and Applications*. Prentice Hall.

Kouki P, Fakhraei S, Foulds J, Eirinaki M and Getoor L (2015) Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In: *ACM Conference on Recommender Systems (RecSys)*. Vienna, Austria.

Kwon J, Kwon D, Wright S and Nowak R (2023) A fully first-order method for stochastic bilevel optimization. In: *ICML*.

Lamb LC, d'Avila Garcez A, Gori M, Prates MOR, Avelar PHC and Vardi MY (2020) Graph neural networks meet neural-symbolic computing: A survey and perspective. In: *IJCAI*.

LeCun Y, Bottou L, Bengio Y and Haffner P (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.

LeCun Y, Chopra S, Hadsell R, Ranzato M and Huang FJ (2006) A tutorial on energy-based learning. *Predicting Structured Data* 1(0).

Liu B, Ye M, Wright S, Stone P and Liu Q (2022) Bome! bilevel optimization made easy: A simple first-order approach. In: *NeurIPS*.

Liu R, Liu X, Yuan X, Zeng S and Zhang J (2021) A value-function-based interior-point method for non-convex bi-level optimization. In: *ICML*.

Liu R, Liu X, Zeng S, Zhang J and Zhang Y (2023) Value-function-based sequential minimization for bi-level optimization. *Arxiv* .

Liu W, Wang X, Owens J and Li Y (2020) Energy-based out-of-distribution detection. In: *NeurIPS*.

Loshchilov I and Hutter F (2019) Decoupled weight decay regularization. In: *ICLR*.

Maene J, Derkinderen V and Raedt LD (2024) On the hardness of probabilistic neurosymbolic learning. *arXiv* .

Maene J and Raedt LD (2024) Soft-unification in deep probabilistic logic. In: *NeurIPS*.

Manhaeve R, Dumančić S, Kimmig A, Demeester T and De Raedt L (2021a) Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence (AI)* 298: 103504.

Manhaeve R, Marra G and De Raedt L (2021b) Approximate inference for neural probabilistic logic programming. In: *ICPKRR*.

Marconato E, Bortolotti S, van Krieken E, Vergari A, Passerini A and Teso S (2024) Bears make neuro-symbolic models aware of their reasoning shortcuts. *arXiv* .

Marconato E, Teso S, Vergari A and Passerini A (2023) Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts. In: *NeurIPS*.

Marra G, Giannini F, Diligenti M and Gori M (2019) Integrating learning and reasoning with deep logic models. In: *ECMLKDD*.

Milgrom P and Segal I (2002) Envelope theorems for arbitrary choice sets. *Econometrica* 70(2): 583–601.

Morra L, Azzari A, Bergamasco L, Braga M, Capogrosso L, Delrio F, Di Giacomo G, Eiraudo S, Ghione G, Giudice R, Koudounas A, Piano L, Rege Cambrin D, Risso M, Rondina M, Sebastien Russo A, Russo M, Taioli F, Vaiani L and Vercellino C (2023) Designing logic tensor networks for visual sudoku puzzle classification. In: *International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*.

NeSy2005 (2005) *Neural-Symbolic Learning and Reasoning Workshop at IJCAI*.

NeSy2024 (2024) *International Conference on Neural-Symbolic Learning and Reasoning*.

Nocedal J and Wright S (2006) *Numerical Optimization*. second edition. Springer.

OpenAI (2024) Gpt-4 technical report. Technical report, OpenAI.

P Kingma D and LeCun Y (2010) Regularized estimation of image statistics by score matching. In: *NeurIPS*.

Pan L, Albalak A, Wang X and Wang WY (2023) Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In: *EMNLP*.

Parikh N and Boyd S (2013) Proximal algorithms. *Foundations and Trends in Machine Learning (FTML)* 3(1): 123–231.

Pedregosa F (2016) Hyperparameter optimization with approximate gradient. In: *ICML*.

Pryor C, Dickens C, Augustine E, Albalak A, Wang WY and Getoor L (2023a) Neupsl: Neural probabilistic soft logic. In: *IJCAI*.

Pryor C, Yuan Q, Liu JZ, Kazemi SM, Ramachandran D, Bedrax-Weiss T and Getoor L (2023b) Using domain knowledge to guide dialog structure induction via neural probabilistic soft logic. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. Toronto, Canada.

Rajeswaran A, Finn C, M Kakade S and Levine S (2019) Meta-learning with implicit gradients. In: *NeurIPS*.

Richardson M, Agrawal R and Domingos P (2003) Trust management for the semantic web. In: *ISWC*.

Robinson S (1980) Strongly regular generalized equations. *Mathematics of Operations Research* 5(1): 43–62.

Rockafellar R (1970) *Convex Analysis*. Princeton University Press.

Rockafellar R (1974) Conjugate duality and optimization. In: *Regional Conference Series in Applied Mathematics*.

Rockafellar R and Wets R (1997) *Variational Analysis*. Springer.

Rocktäschel T and Riedel S (2017) End-to-end differentiable proving. In: *NeurIPS*.

Sachan M, Dubey KA, Mitchell TM, Roth D and Xing EP (2018) Learning pipelines with limited data and domain knowledge: A study in parsing physics problems. In: *NeurIPS*.

Salakhutdinov R and Larochelle H (2010) Efficient learning of deep boltzmann machines. In: *AISTATS*.

Scellier B and Bengio Y (2017) Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience* 11.

Sen P, Namata GM, Bilgic M, Getoor L, Gallagher B and Eliassi-Rad T (2008) Collective classification in network data. *AI Magazine* 29(3): 93–106.

Shalev-Shwartz S (2012) Online learning and online convex optimization. *Foundations and Trends in Machine Learning (FTML)* 4(2): 107–194.

Sikka K, Silberfarb A, Byrnes J, Sur I, Chow E, Divakaran A and Rohwer R (2020) Deep adaptive semantic logic (dasl): Compiling declarative knowledge into deep neural networks. Technical report, SRI International.

Singh G, Akrigg S, Di Maio M, Fontana V, Javanmard Alitappeh R, Saha S, Jeddi Saravi K, Yousefia F, Culley J, Nicholson T, Omokeowa J, Khan S, Grazioso S, Bradley A, Di Gironimo G and Cuzzolin F (2021) Road: The road event awareness dataset for autonomous driving. *IEEE TPAMI* 45: 1036–1054.

Song Y and Ermon S (2019) Generative modeling by estimating gradient of the data distribution. In: *NeurIPS*.

Sow D, Ji K, Guan Z and Liang Y (2022) A primal-dual approach to bilevel optimization with multiple inner minima. *Arxiv* .

Sridhar D, Fakhraei S and Getoor L (2016) A probabilistic approach for collective similarity-based drug-drug interaction prediction. *Bioinformatics* 32(20): 3175–3182.

Srinivasan S, Dickens C, Augustine E, Farnadi G and Getoor L (2021) A taxonomy of weight learning methods for statistical relational learning. *Machine Learning* .

Srivastava A, Rastogi A, Rao A, Shoeb AAM, Abid A, Fisch A, Brown AR, Santoro A, Gupta A, Garriga-Alonso A, Kluska A, Lewkowycz A, Agarwal A, Power A, Ray A, Warstadt A, Kocurek AW, Safaya A, Tazarv A, Xiang A, Parrish A, Nie A, Hussain A, Askell A, Dsouza A, Slone A, Rahane AA, Iyer AS, Andreassen A, Madotto A, Santilli A, Stuhlmuller A, Dai AM, La A, Lampinen AK, Zou A, Jiang A, Chen A, Vuong A, Gupta A, Gottardi A, Norelli A, Venkatesh A, Gholamidavoodi A, Tabassum A, Menezes A, Kirubarajan A, Mullokandov A, Sabharwal A, Herrick A, Efrat A, Erdem A, Karakacs A and et al (2022) Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *ArXiv* .

Stoian M, Giunchiglia E and Lukasiewicz T (2023) Exploiting t-norms for deep learning in autonomous driving. In: *NeSy*.

Stoyanov V, Ropson A and Eisner J (2011) Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In: *AISTATS*.

Sutton R and Barto A (2018) *Reinforcement Learning: An Introduction*. MIT Press.

Sutton R, McAllester D, Singh S and Mansour Y (1999) Policy gradient methods for reinforcement learning with function approximation. In: *NeurIPS*.

Tran S and d'Avila Garcez A (2018) Deep logic networks: Inserting and extracting knowledge from deep belief networks. *IEEE Transactions on Neural Networks and Learning Systems* 29(2): 246–258.

V Outrata J (1990) On the numerical solution of a class of stackelberg problems. *Methods and Models of Operations Research* 34(4): 255–277.

van Krieken E, Acar E and van Harmelen F (2022) Analyzing differentiable fuzzy logic operators. *Artificial Intelligence (AI)* 302: 103602.

van Krieken E, Badreddine S, Manhaeve R and Giunchiglia E (2024) Uller: A unified language for learning and reasoning. *arXiv* .

van Krieken E, Thanapalasingam T, Tomczak J, van Harmelen F and ten Teije A (2023) A-nesi: A scalable approximate method for probabilistic neurosymbolic inference. In: *NeurIPS*.

Vlastelica M, Paulus A, Musil V, Martius G and Rolínek M (2020) Differentiation of blackbox combinatorial solvers. In: *ICLR*.

Walker MA, Tree JEF, Anand P, Abbott R and King J (2012) A corpus for research on deliberation and debate. In: *LREC*.

Wan Z, Liu CK, Yang H, Li C, You H, Fu Y, Wan C, Krishna T, Lin Y and Raychowdhury A (2024) Towards cognitive ai systems: A survey and prospective on neuro-symbolic ai. *arXiv* .

Wang P, Donti P, Wilder B and Kolter Z (2019) Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In: *ICML*.

Wei J, Wang X, Schuurmans D, Bosma M, Xia F, Chi E, Le QV and Zhou D (2022) Chain-of-thought prompting elicits reasoning in large language models. In: *NeurIPS*.

Welling M and Teh Y (2011) Bayesian learning via stochastic gradient langevin dynamics. In: *ICML*.

Williams R (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8: 229–256.

Winters T, Marra G, Manhaeve R and Raedt LD (2022) Deepstochlog: Neural stochastic logic programming. In: *AAAI*.

Wishart DS, Knox C, Guo AC, Shrivastava S, Hassanali M, Stothard P, Chang Z and Woolsey J (2006) Drugbank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Research (NAR)* 34: D668–D672.

Wu F, Zhang T, Holanda de Souza Jr A, Fifty C, Yu T and Q Weinberger K (2019) Simplifying graph convolutional networks. In: *ICML*.

Xu J, Zhang Z, Friedman T, Liang Y and Van den Broeck G (2018) A semantic loss function for deep learning with symbolic knowledge. In: *ICML*.

Yang Z, Ishay A and Lee J (2020) Neurasp: Embracing neural networks into answer set programming. In: *IJCAI*.

Yi K, Wu J, Gan C, Torralba A, Kohli P and B Tenenbaum J (2019) Neural-symbolic vqa: Disentanlging reasoning from vision and language understanding. In: *NeurIPS*.

Zhang H, Dang M, Peng N and Van den Broeck G (2023) Tractable control for autoregressive language generation. In: *International Conference on Machine Learning (ICML)*.

Zhao J, Mathieu M and LeCun Y (2017) Energy-based generative adversarial networks. In: *ICLR*.

Zhou K, Zheng K, Pryor C, Shen Y, Jin H, Getoor L and Wang XE (2023) Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In: *International Conference on Machine Learning (ICML)*.