Deep Neuro-Symbolic Weight Learning in Neural Probabilistic Soft Logic

Connor Pryor [1] **Charles Dickens** [1] **Lise Getoor** [1]

Abstract

In this work, we extend the expressive power of the neuro-symbolic framework Neural Probabilistic Soft Logic (NEUPSL) (Pryor et al., 2023). We introduce NEUPSL DEEP WEIGHTS, which uses deep neural network predictions to parameterize the weights of symbolic rules. To demonstrate NEUPSL DEEP WEIGHTS applicability, we introduce a unique synthetic dataset specifically designed to challenge learning methods that do not utilize both data-driven learning (System 1) and deliberate symbolic reasoning (System 2). Across variations of this synthetic dataset, we show how NEUPSL DEEP WEIGHTS outperforms traditional PSL rule weights and existing joint System 1 and System 2 neural methods, such as graph neural networks.

1. Introduction

The field of machine learning has witnessed remarkable progress in recent decades, largely due to the combination of data-driven learning (System 1) and the development of sophisticated deep neural network architectures. These developments have resulted in impressive breakthroughs across various domains, particularly in the realms of natural language processing and computer vision. Models like ChatGPT and DALL·E have emerged as notable examples, finding extensive applications in academia, industry, and among the general public. Despite their undeniable success, these models still face challenges, including hallucinations, inconsistent predictions, and the need for substantial training data. Researchers are diligently working to tackle these issues, and one promising avenue of research seeks to integrate two complementary systems: System 1's rapid and automatic processing and System 2's slower, deliberate symbolic reasoning. A promising active subfield, neuro-symbolic computation (NeSy) (d'Avila Garcez et al., 2002), aims to integrate symbolic reasoning (System 2) into the training of low-level neural perception (System 1).

One of the key challenges within the NeSy community is the effective integration of subsymbolic and symbolic methods. This integration is crucial to enable fast, expressive, and differentiable neuro-symbolic systems. Our approach extends the expressivity of Neural Probabilistic Soft Logic (NEUPSL) (Pryor et al., 2023), a recently published NeSy framework designed for scalable joint (structured) prediction. NeuPSL combines state-of-the-art symbolic reasoning with the perceptual capabilities of deep neural networks through predicates in a Probabilistic Soft Logic (PSL) program (Bach et al., 2017). By leveraging these *deep predicates* (predicates parameterized by neural networks) and traditional symbolic predicates, NeuPSL combines them with constraints and weighted arithmetic or logical rules. These weights represent relative importance for NEUPSL predictions to satisfy instances of the rules. Every instance of a rule with the same structure shares a single learnable weight parameter. Weight sharing allows NEUPSL to generalize quickly in low-data settings and avoid overfitting. However, in many domains, communities or even individual entities may follow different symbolic structures, i.e., rules should be weighted differently. Moreover, a modeler may only have limited domain knowledge and a deep neural network provided with enough data can be used to supplement a model with a small set of rules.

This work extends the expressive power of NEUPSL with NEUPSL DEEP WEIGHTS: deep neural network parameterized rule weights. NEUPSL DEEP WEIGHTS are neural networks that are provided with features associated with each instantiated predicate in a rule. The NEUPSL DEEP WEIGHTS are trained to predict rule weights giving good prediction performance on a training data set. We demonstrate how NEUPSL DEEP WEIGHTS can be applied to identify communities following different symbolic structures with a novel synthetic dataset.

Our key contributions are: 1) *Extension of NeuPSL*: We introduce NEUPSL DEEP WEIGHTS, which extends the theoretical and expressive power of NeuPSL by incorporating neural network outputs as weights within the symbolic rules. 2) *Novel Synthetic Dataset*: We introduce a unique synthetic

[*]Equal contribution [1]Department of Computer Science, University of California, Santa Cruz, United States. Correspondence to: Connor Pryor <cfpryor@ucsc.edu>, Charles Dickens <cadicken@ucsc.edu>, Lise Getoor <getoor@ucsc.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

dataset that is specifically designed to challenge learning methods that do not utilize both System 1 and System 2. 3) *Comprehensive Evaluation*: We evaluate NEUPSL DEEP WEIGHTS on three progressively challenging versions of the synthetic dataset. Our evaluations demonstrate that NEUPSL DEEP WEIGHTS consistently outperforms powerful joint System 1 and System 2 neural approaches, such as GNNs, and additionally shows improved performance over traditional PSL rule weights.

2. Related Work

A related field of work, Neuro-Symbolic computing (NeSy) (d'Avila Garcez et al., 2002; Bader & Hitzler, 2005; d'Avila Garcez et al., 2009; Serafini & d'Avila Garcez, 2016; Besold et al., 2017; Donadello et al., 2017; Yang et al., 2017; Evans & Grefenstette, 2018; Manhaeve et al., 2021; d'Avila Garcez et al., 2019; De Raedt et al., 2020; Lamb et al., 2020; Badreddine et al., 2022), is an active area of research that aims to integrate logic-based reasoning with neural computation. NeSy methods can be categorized into four major groups: differentiable frameworks for logic reasoning, constrained outputs, executable logic programs, and neural networks as predicates. Although our approach does not fall directly into any of these categories, it aligns closely with the concept of neural networks as predicates. However, our approach differs in a significant manner, as we connect the output of neural networks with the *weights* of symbolic rules. For readers interested in a more comprehensive introduction to NeSy, we refer them to the excellent recent surveys: Besold et al. (2017) and De Raedt et al. (2020).

3. Neural Probabilistic Soft Logic

Neural Probabilistic Soft Logic (NEUPSL) is a NeSy framework designed for scalable joint reasoning (Pryor et al., 2023). NEUPSL integrates the symbolic reasoning capabilities of probabilistic soft logic (PSL) (Bach et al., 2017) and the low-level perception of neural networks. Specifically, in PSL, relations and attributes of entities are represented by *atoms*, and dependencies between atoms are encoded with first-order logical clauses and linear arithmetic inequalities referred to as *rules*. The rules define a joint probability distribution, and MAP inference is performed to obtain predictions.

More formally, *Atoms* are expressions of the form $q(t_1, \cdots, t_k)$ where q is a predicate and each t_i is a *term*. A term in an atom is either a *constant* or a *variable*. A *ground atom* is an atom with all constant terms and an associated real value. Moreover, ground atoms are partitioned into *observed* and *unobserved* sets where the value of observed atoms are known while the value of unobserved atoms are unknown. The observed atoms are mapped to a vector of $n_{\mathbf{x}}$ variables $\mathbf{x}_{sy}$ while unobserved atoms are mapped to a vector of $n_{\mathbf{y}}$ target variables $\mathbf{y}$. All atoms in PSL have a value in the range $[0, 1]$. NEUPSL extends PSL with *deep predicates*, which create atoms that take values from a neural network.

To illustrate, consider a citation network with nodes representing academic papers and edges representing a citation link. Further, suppose each paper and citation is associated with a set of features, perhaps the abstract, the location, and the frequency of the citation in the paper. The task is to classify the topic of each paper in the network. NEUPSL can represent a neural network's prediction of a paper P's topic T given the paper features with the atom $\texttt{Neural}(P, T)$. `Neural` is a deep predicate associated with a single neural model with *neural weights* $\mathbf{w}_{nn}$. The terms P and T are variables that are substituted with constants to create ground atoms. For instance, suppose the domain of constants associated with the variable P is $\{p_1, \cdots, p_{10}\}$ and with the variable T is $\{t_1, t_2, t_3\}$. The value of a ground instance of the atom, e.g., $\texttt{Neural}(p_1, t_1)$, is equal to the deep predicate's neural model's prediction. Moreover, whether two papers with the same domain of constants as P, $P1$ and $P2$, are related by a citation can be captured by the atom $\texttt{Cites}(P1, P2)$. Finally, NEUPSL's label classification can be represented by the atom $\texttt{Topic}(P, T)$.

A *rule* is a logical or arithmetic expression relating atoms, and a *ground rule* is a rule with all *ground atoms*. Rules can be parameterized with a *symbolic weight* representing the relative importance of satisfying it in the model's prediction. The higher the weight of a rule relative to all other rules, the more important it is to satisfy each instance of it. Rules without weights are constraints that must be satisfied. For citation network node classification, an example NEUPSL model, i.e., a collection of rules, is:

$$w_1 : \textsc{Neural}(P, T) = \textsc{Topic}(P, T) \quad (1)$$

$$w_2 : \textsc{Topic}(P1, T) \wedge \textsc{Cites}(P1, P2) \rightarrow \textsc{Topic}(P2, T) \quad (2)$$

$$\textsc{Topic}(P, T1) + \textsc{Topic}(P, T2) + \textsc{Topic}(P, T3) = 1. \quad (3)$$

The first rule above is weighted with w_1 and states that the neural model's topic prediction should be equal to NEUPSL's prediction. The second rule above is weighted with w_2 and represents the propagation of topic labels across citation links. In other words, if both $\textsc{Topic}(P1, T)$ and $\textsc{Cites}(P1, P2)$ are 1, then that implies $\textsc{Topic}(P2, T)$ should also be 1. Finally, the third rule is unweighted and, hence, is a constraint. The constraint states that the sum of the topic predictions for a single paper should equal 1 and is how NEUPSL makes categorical predictions.

Ground rules are created by realizing substitutions of variables with constants in their domain. A ground instance of the first rule above is $w_1 : \textsc{Neural}(p_1, t_1) = \textsc{Topic}(p_1, t_1)$. Every ground rule NEUPSL creates is translated into one or more *potential functions* measuring the satisfaction of the rule. Logical templates use a continuous

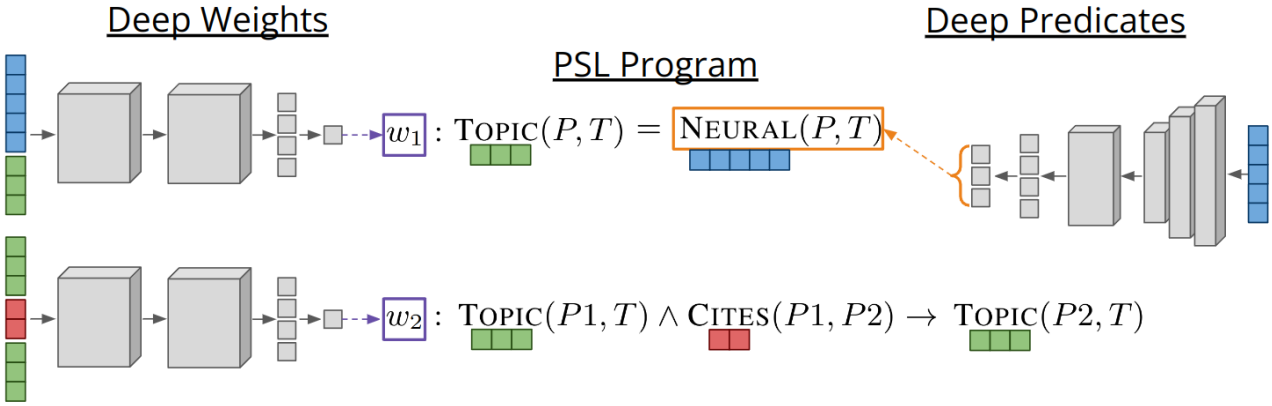


Figure 1. NEUPSL DEEP WEIGHTS and NEUPSL Deep Predicates overview.

relaxation known as *Łukasiewicz* logic to define hinge-loss potentials (Klir & Yuan, 1995), while arithmetic templates are defined to be the distance to satisfaction of a linear inequality. The set of all potentials created from the grounding process defines a probabilistic graphical model called a deep hinge-loss Markov random field (deep HL-MRF):

Definition 3.1 (Deep Hinge-Loss Markov Random Field)**.** Let $\mathbf{g} = [g_i]_{i=1}^{n_g}$ be functions with corresponding neural weights $\mathbf{w}_{nn} = [\mathbf{w}_{nn,i}]_{i=1}^{n_g}$ and inputs $\mathbf{x}_{nn}$ such that $g_i : (\mathbf{w}_{nn,i}, \mathbf{x}_{nn}) \mapsto [0,1]$. Let $\mathbf{y} \in [0,1]^{n_\mathbf{y}}$ and $\mathbf{x}_{sy} \in [0,1]^{n_\mathbf{x}}$. A **deep hinge-loss potential** is a function of the form:

$$\begin{aligned}&\phi_k(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \\ &:= (\max\{\mathbf{a}_{\phi_k,\mathbf{y}}^T \mathbf{y} + \mathbf{a}_{\phi_k,\mathbf{x}_{sy}}^T \mathbf{x}_{sy} + \mathbf{a}_{\phi_k,\mathbf{g}}^T \mathbf{g} + b_{\phi_k}, 0\})^{p_k},\end{aligned} \tag{4}$$

where $p_k \in \{1,2\}$. Let $\mathcal{T} = [t_i]_{i=1}^r$ denote an ordered partition of a set of m deep hinge-loss potentials. Further, define $\mathbf{\Phi}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) := [\sum_{k \in t_i} \phi_k(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))]_{i=1}^r$.

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:

$$E(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) := \mathbf{w}_{sy}^T \mathbf{\Phi}(\mathbf{y}, \mathbf{x_{sy}}, \mathbf{g}(\mathbf{x_{nn}}, \mathbf{w_{nn}})). \tag{5}$$

Let $\mathbf{a}_{c_k,\mathbf{y}} \in \mathbb{R}^{n_\mathbf{y}}$, $\mathbf{a}_{c_k,\mathbf{x}} \in \mathbb{R}^{n_\mathbf{x}}$, $\mathbf{a}_{c_k,\mathbf{g}} \in \mathbb{R}^{n_g}$, 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:

$$\begin{aligned}&\mathbf{\Omega}(\mathbf{x}_{sy}, \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) := \Big\{\mathbf{y} \in [0,1]^{n_\mathbf{y}} : \\ &\mathbf{a}_{c_k,\mathbf{y}}^T \mathbf{y} + \mathbf{a}_{c_k,\mathbf{x}}^T \mathbf{x}_{sy} + \mathbf{a}_{c_k,\mathbf{g}}^T \mathbf{g} + b_{c_k} \leq 0, \forall\, k = 1, \ldots, q\Big\}.\end{aligned} \tag{6}$$

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

$$P(\mathbf{y}|\mathbf{x}_{sy}, \mathbf{x}_{nn}) := \begin{cases} \frac{\exp(-E(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}))}{\int_{\mathbf{y} \in \mathbf{\Omega}(\cdot)} \exp(-E(\cdot)) d\mathbf{y}} & \mathbf{y} \in \mathbf{\Omega}(\cdot) \\ 0 & o.w. \end{cases} \tag{7}$$

Each hinge-loss potential is convex. Furthermore, NEUPSL typically enforces an additional constraint that the rule weights are all non-negative. With this, MAP inference is a convex problem and scalable algorithms are used to find globally optimal solutions. More formally, *neural inference* is computing the output of the neural networks given the input $\mathbf{x}_{nn}$, i.e., computing $g_{nn,i}(\mathbf{x}_{nn}, \mathbf{w}_{nn,i})$ for all i. NEUPSL *symbolic inference* minimizes the energy function over $\mathbf{y}$:

$$\mathbf{y}^* = \underset{\mathbf{y}|(\mathbf{y},\mathbf{x}_{sy}) \in \mathbf{\Omega}}{\arg\min}\ E(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{nn}, \mathbf{w}_{sy}) \tag{8}$$

NEUPSL learning is the task of fitting both neural and symbolic weights. Learning objectives map neural weights $\mathbf{w}_{nn}$, symbolic weights $\mathbf{w}_{sy}$, and a set of training examples $\mathcal{S} = \{(\mathbf{y}_i, \mathbf{x}_{sy,i}, \mathbf{x}_{nn,i}) : i = 1, \cdots, P\}$ to a real-valued loss: $\mathcal{L}(\mathcal{S}, \mathbf{w}_{nn}, \mathbf{w}_{sy})$. Learning objectives follow the standard empirical risk minimization framework and are sums of per-sample loss functions $L_i(\mathbf{y}_i, \mathbf{x}_i, \mathbf{x}_{nn,i}, \mathbf{w}_{nn}, \mathbf{w}_{sy})$. Concisely, NEUPSL learning is:

$$\begin{aligned}&\underset{\mathbf{w}_{nn}, \mathbf{w}_{sy}}{\arg\min}\ \mathcal{L}(\mathbf{w}_{nn}, \mathbf{w}_{sy}, \mathcal{S}) \\ &= \underset{\mathbf{w}_{nn}, \mathbf{w}_{sy}}{\arg\min} \sum_{i=1}^{P} L_i(\mathbf{y}_i, \mathbf{x}_{sy,i}, \mathbf{x}_{nn,i}, \mathbf{w}_{nn}, \mathbf{w}_{sy})\end{aligned} \tag{9}$$

Gradient-based learning algorithms use the following partial derivatives:

$$\frac{\partial E(\cdot)}{\partial \mathbf{w}_{sy}[i]} = \Phi_i(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{nn}) \tag{10}$$

$$\frac{\partial E(\cdot)}{\partial \mathbf{w}_{nn}[i]} = \mathbf{w}_{sy}^T \frac{\partial \Phi(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{nn})}{\partial \mathbf{w}_{nn}[i]}, \tag{11}$$

where $\frac{\partial \Phi(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{nn})}{\partial \mathbf{w}_{nn}[i]}$ is the vector of partial derivatives of the aggregated potentials with respect to $\mathbf{w}_{nn}[i]$. Standard backpropagation-based algorithms for computing gradients are applied for neural and symbolic parameter learning.

4. Neural Probabilistic Soft Logic Deep Weights

NEUPSL defines its energy function such that every deep hinge-loss potential instantiated by a rule shares a symbolic weight. Although there are positive effects of weight sharing, this parameterization limits the model expressivity. We propose Neural Probabilistic Soft Logic Deep Weights (NEUPSL DEEP WEIGHTS), a natural generalization of NEUPSL's neural and symbolic connection that improves the expressivity of the framework. The neural and symbolic connection of NEUPSL deep predicates and NEUPSL DEEP WEIGHTS is illustrated in Figure 1.

Consider the citation network node classification setting and NEUPSL model introduced in the previous section. Traditionally, every instance of the label propagation rule, (2), has the same weight, allowing NEUPSL to avoid overfitting and perform in low-data settings. However, in many problems, there are groups of variables with different symbolic structures. In the citation network example, there may exist a community of papers that commonly cite papers from a diverse range of topics, while another community is more focused on a single topic. In this case, the weight of (2) should be different depending on the community membership of the papers in a rule instance. NEUPSL can capture this dependency by explicitly creating a new atom representing the community membership of papers and creating multiple variants of (2). However, this puts a burden and expectation on the modeler to define a finite number of communities that are going to be represented as well as creating variants of the symbolic rules to capture the dependency. Moreover, community structure was only used for motivation and the weight of a rule instance may additionally depend on more complex relationships between features associated with each atom in a rule.

Rather than relying solely on the modeler to fully specify the dependencies of the weights, we train a deep neural network to use the low-level features associated with all the atoms to specialize the weights of each rule instance. This specialization allows NEUPSL to seamlessly model domains with communities following a different symbolic structure. Formally, the Deep HL-MRF definition is generalized to support a deep parameterization of a subset of rule weights. Suppose the NeuPSL model is made up of r weighted rules defining $\mathcal{T} = [t_i]_{i=1}^{r}$, an ordered partition of a set of m deep hinge-loss potentials $\{\phi_1, \cdots, \phi_m\}$. Without loss of generality, suppose the first $r_g \leq r$ weighted rules are parameterized by a neural network. For each partition t_i such that $i \leq r_g$ let $\mathbf{g}_{i,r}(\mathbf{x}_{nn}, \mathbf{w}_{nn})$ denote the vector of neural weight predictions for each potential instantiated by the rule and let $\mathbf{g}_r(\mathbf{x}_{nn}, \mathbf{w}_{nn}) := \|_{i=1}^{r} \mathbf{g}_{i,r}(\mathbf{x}_{nn}, \mathbf{w}_{nn})$ be the concatenation of the vectors of predictions. Furthermore, define I_{t_i} to be an ordering of the indexes of potentials created by t_i. Then, define:

$$\Phi_i(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) \tag{12}$$

$$:= \begin{cases} [\phi_{I_{t_i}[j]}(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn}))]_{j=1}^{|t_i|} & i \leq r_g \\ \sum_{j \in t_i} \phi_j(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{g}(\mathbf{x}_{nn}, \mathbf{w}_{nn})) & \text{o.w.} \end{cases}. \tag{13}$$

Further, let $\mathbf{w}_{sy}$ be the vector of non-negative symbolic weights corresponding to the rules not parameterized by neural networks. Then the deep HL-MRF energy function with neural parameterized weights is:

$$E(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{sy}, \mathbf{w}_{nn}) \tag{14}$$

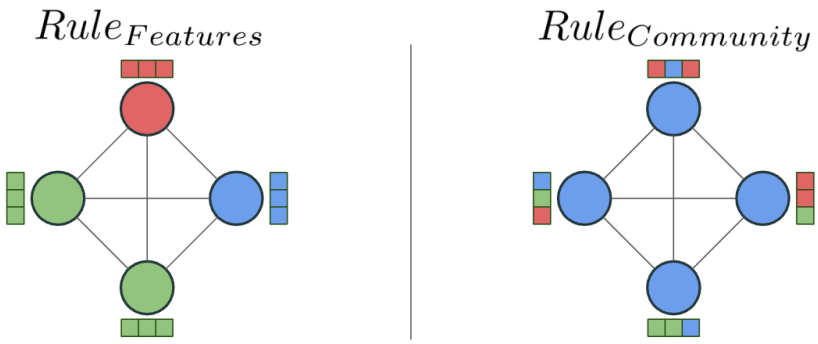
$$:= \begin{bmatrix} \mathbf{g}_r(\mathbf{x}_{nn}, \mathbf{w}_{nn}) \\ \mathbf{w}_{sy} \end{bmatrix}^T \mathbf{\Phi}(\mathbf{y}, \mathbf{x_{sy}}, \mathbf{g}(\mathbf{x_{nn}}, \mathbf{w_{nn}})). \tag{15}$$

With this energy function, symbolic inference is still convex and the same inference algorithms used to solve (8) are applicable. Learning, on the other hand, is affected by the new parameterization. Specifically, the gradient with respect to the neural parameters needs to be generalized. Using the product rule of differentiation, we have:

$$\frac{\partial E(\cdot)}{\partial \mathbf{w}_{nn}[i]} = \begin{bmatrix} \frac{\partial \mathbf{g}_r(\mathbf{x}_{nn}, \mathbf{w}_{nn})}{\partial \mathbf{w}_{nn}[i]} \\ \mathbf{0} \end{bmatrix}^T \Phi(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{nn}) + \begin{bmatrix} \mathbf{g}_r(\mathbf{x}_{nn}, \mathbf{w}_{nn}) \\ \mathbf{w}_{sy} \end{bmatrix}^T \frac{\partial \Phi(\mathbf{y}, \mathbf{x}_{sy}, \mathbf{x}_{nn}, \mathbf{w}_{nn})}{\partial \mathbf{w}_{nn}[i]}, \tag{16}$$

where $\frac{\partial \mathbf{g}_r(\mathbf{x}_{nn}, \mathbf{w}_{nn})}{\partial \mathbf{w}_{nn}[i]}$ is the vector of partial derivatives of the neural weight predictions with respect to $\mathbf{w}_{nn}[i]$. With (16) gradient-based learning algorithms for NEUPSL are applicable.

The NEUPSL DEEP WEIGHTS framework is general and there are many ways a modeler may choose to design the neural architecture and rules. However, a powerful modeling pattern we introduce is demonstrated in Figure 1. Here, the neural network parameterizing the rule weight is given access to features associated with each atom present in the rule, the two papers and the citation link in the label propagation example. This differs from the typical modeling pattern for deep predicates, which only gives the neural model access to features associated with a single atom, referring to a single paper in (1). Giving a neural network access to features for each atom in the rule allows it to learn and model complex interactions between the atoms. For instance, the features may provide a signal identifying the community membership of a paper where some communities adhere to the label propagation rule while others do not. This information can be used to increase or decrease the weights of the rules. In the following section, we introduce a synthetic dataset that exemplifies communities following different symbolic structures.



Name	Symbolic Meaning
$Rule_{Features}$	FEATURES(Node, Class) → LABEL(Node, Class)
$Rule_{Community}$	COMMUNITY(Node, ID) ∧ COMMUNITYLABEL(ID, Class) → LABEL(Node, Class)

Figure 2. Rules, symbolic meaning, and graphical representation used to generate features and labels for the synthetic datasets.

5. Synthetic Dataset

To study the effectiveness of NEUPSL DEEP WEIGHTS, we introduce a synthetic dataset that will be examined alongside other System 1 and System 2 methods in Section 6. This synthetic dataset has been carefully designed to challenge a system's ability to reason effectively with both local features and the overall structure. It requires the utilization of both System 1 and System 2 during the learning process.

5.1. Definition and Generation

At a high level, the synthetic dataset is defined as a set of disjoint vertex and edge communities that are fully connected. Each community adheres to an underlying rule governing its node's labels and features. Figure 2 summarizes the underlying rules, their symbolic interpretations, and a graphical example of each. Communities generated according to $Rule_{Features}$ will have random node labels but features directly correlating to these labels. On the other hand, communities generated based on $Rule_{Community}$ will have a single common label, but node features are not correlated with the community label.

Dataset Definition: Define dataset $\mathcal{D}$ to be a finite graph $G = (V, E)$ with node features X_V, labels Y_V, and k disjoint and fully connected communities $\{S_1 = (V_1, E_1), \cdots, S_k = (V_k, E_k)\}$ such that $\bigcup S_i = G$, $\bigcup V_i = V$, $\bigcup E_i = E$, $\bigcap V_i = \emptyset$, and $\bigcap E_i = \emptyset$.

Dataset Generation: Let $k \in \mathbb{Z}^+$ be the number of communities for dataset $\mathcal{D}$, $a_{min}, a_{max} \in \mathbb{Z}^+$ be the minimum and maximum number of nodes a community can have, and $L = \{l_1, \cdots, l_c\}$ be the label space.

For each community, S_i, generation begins by randomly sampling a rule $r_i \in_R \{Rule_{Features}, Rule_{Community}\}$, community label $s_i \in_R L$ and the number of nodes within the community $n_i \in_R \{a_{min}, \cdots, a_{max}\}$.

Then, n_i, nodes are iteratively created by sampling a label:

$$y = \begin{cases} \in_R L & r_i = Rule_{Features} \\ s_i & r_i = Rule_{Community} \end{cases}$$

and features:

$$\mathbf{x} = \begin{cases} \ell(y) & r_i = Rule_{Features} \\ \mathcal{U}(-m, m) & r_i = Rule_{Community} \end{cases}$$

where $\ell(y) \in \mathbb{R}^d$ is a function that will depend on the type of features (Gaussian or One-Hot) and $\mathcal{U}(-m, m)$ is a uniform distribution. Finally, a cross-product of all nodes V_i, within the community S_i, creates the set of edges E_i, i.e., each community is a fully connected subgraph.

5.2. Inductive vs. Transductive

The synthetic dataset can be utilized to generate settings that fall into two categories: *inductive* and *transductive* (Figure 3) (Hamilton et al., 2017). In the inductive setting, the nodes and edges present in the graph during training may be a subset or entirely different from the test graph. In other words, a model must generalize to unseen data. On the other hand, in transductive settings, the training and test graph are the same and only the test node labels are unavailable during training.

In our empirical evaluations, we focus on the inductive setting. Specifically, we introduce new nodes to existing communities. This setting is illustrated in the left-hand pane of Figure 3. In this inductive data setting, models have the opportunity to identify an unseen node's community membership and, thus, the symbolic structure it follows.

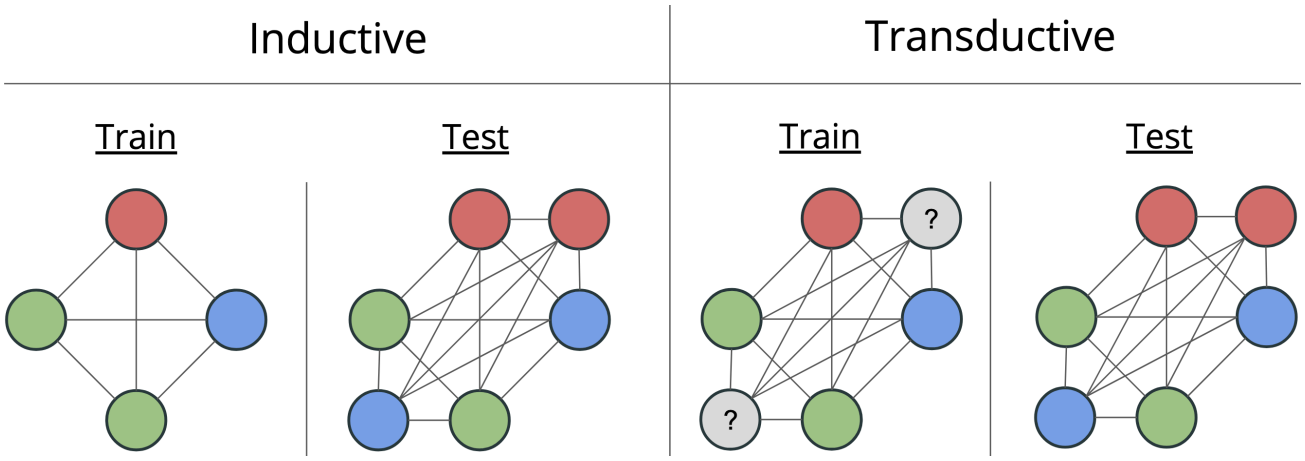


Figure 3. Example of inductive and transductive settings for the synthetic dataset.

5.3. Features

In Section 6, we evaluate this synthetic dataset with three sets of features: One-Hot Rule Features + One-Hot Community Features ($Feats_{OH+OH}$), Gaussian Rule Features + One-Hot Community Features ($Feats_{G+OH}$), and Gaussian Rule Features + Gaussian Community Features ($Feats_{G+G}$).

$Feats_{OH+OH}$: Features $\ell(y)$ are a concatenation of a one-hot encoding of the label y and a one-hot encoding of the community id i.

$Feats_{G+OH}$: Features $\ell(y)$ are a concatenation of a sample from a multivariate Gaussian with a mean and covariance defined for each class label in L and a one-hot encoding of the community id i.

$Feats_{G+G}$: Features $\ell(y)$ are a concatenation of a sample from a multivariate Gaussian with a mean and covariance defined for each class label in L and a concatenation of a sample from a multivariate Gaussian with a mean and covariance defined for each community id.

6. Experimental Evaluation

We investigate the following questions: Q1) How does NeuPSL Deep Weights perform against traditional PSL rule weights? Q2) Can NeuPSL Deep Weights provide a boost over conventional purely data-driven neural models (System 1)? Q3) How does NeuPSL Deep Weights perform as features become less representative of the underlying label?[1].

[1]All code and data will be made publicly available upon acceptance.

6.1. Models

We compare NeuPSL Deep Weights with one System 1 method and two joint System 1 and System 2 methods: MLP (System 1), GNN (System 1 and System 2), and PSL (System 1 and System 2).[2]

Multi-Layer Perception (MLP): The MLP baseline consists of an input layer, a single hidden layer, and an output layer. Since this method relies solely on low-level perceptual features as input, we classify it as a System 1 approach.

Graph Neural Network (GNN): The GNN model follows the GraphSAGE framework proposed by (Hamilton et al., 2017). Although the aggregation of node features and the concatenation of this aggregated information may initially appear as System 2 knowledge, in the synthetic data scenario outlined in Section 5, such simplistic aggregation and concatenation approaches represent considerably weak symbolic knowledge.

Probabilistic Soft Logic (PSL): The baseline PSL model uses traditional single-parameter rule weights. Figure 4 summarizes the rules used within the PSL baseline. A local MLP model trained on only rule features (no community features) is used for the neural predicate within the local information rule. This local MLP consists of an input layer, a single hidden layer, and an output layer. While PSL is a joint System 1 and System 2 approach, the weight-sharing mechanism employed in the symbolic rules allows for reduced data requirements, making it more aligned with System 2. As we will see in the following section, System 1 is leveraged in a limited capacity to enhance performance.

[2]Hyperparameter ranges and final values can be found in the Appendix.

 # Local Information

$w_1 : \text{NEURAL}(\texttt{Node}, \texttt{Class}) = \text{LABEL}(\texttt{Node}, \texttt{Class})$

 # Label Propagation

$w_2 : \text{EDGE}(\texttt{Node1}, \texttt{Node2}) \wedge \text{LABEL}(\texttt{Node1}, \texttt{Class}) \rightarrow \text{LABEL}(\texttt{Node2}, \texttt{Class})$

 # Simplex Constraints

$\text{LABEL}(\texttt{Node}, +\texttt{Class}) = 1.$

Figure 4. Rules used for PSL and NeuPSL in all synthetic settings.

Method	$Feats_{OH+OH}$	$Feats_{G+OH}$	$Feats_{G+G}$
MLP	99.09 ± 1.82	88.10 ± 5.69	89.04 ± 2.72
GNN	98.43 ± 1.94	94.56 ± 1.98	81.75 ± 2.65
PSL	81.07 ± 5.14	87.06 ± 5.61	86.71 ± 5.46
NEUPSL DEEP WEIGHTS	**100.00 ± 0.00**	**100.00 ± 0.00**	**93.35 ± 2.23**

Table 1. Average categorical accuracy on the highest correlation between features and labels for $Feats_{OH+OH}$, $Feats_{G+OH}$, and $Feats_{G+G}$ data settings. Best-performing methods are in bold.

NEUPSL DEEP WEIGHTS**:** Figure 4 summarizes the rules used within all synthetic settings for the NEUPSL DEEP WEIGHTS model. For each of these symbolic rules, a neural model is created to predict the weight of the rule. Two MLPs are used for the deep weights models, one for each rule. A local MLP model trained on only rule features (no community features) is used for the neural predicate within the local information rule. All MLP models have an input layer, a single hidden layer, and an output layer.

6.2. Experimental Results

We conducted two experiments using the four models described above. The first assesses the ability of each model to reason about the problem when the features for communities generated from $Rule_{Features}$ directly represent the underlying labels. This direct representation is evaluated using the $Feats_{OH+OH}$ setting and the $Feats_{G+OH}$ and $Feats_{G+G}$ settings when the covariance identity matrix is multiplied by a small scalar value of 0.1. In this setting, most node labels are identifiable from the features and graph structure. The second experiment examines each model's performance as the set of features for the community generated from $Rule_{Features}$ becomes less correlated with the node label. This situation is simulated by gradually increasing covariance for $Feats_{G+OH}$ and $Feats_{G+G}$. The covariance values in this experiment are from the range: $\{0.1, 1.0, 10.0, 50.0, 100.0\}$.

In all experiments the number of communities is $k = 25$, and the node label space is $L = \{0, 1, 2, 3\}$. The minimum and maximum community sizes were set to $a_{min} = 10$ and $a_{max} = 15$, respectively. It was ensured that an equal number of communities generated from $Rule_{Features}$ and $Rule_{Community}$ were present. Each experiment was performed on 5 splits using $60/30/10$ train-test-valid partitions of the inductive setting. Every community was generated to contain at least two nodes within the train set.

Table 1 reports the average and standard deviation of the categorical accuracy of each model in the first experimental setting. In all cases, NEUPSL DEEP WEIGHTS performs the best (Q1 and Q2) and only lose a few percentage points with the most challenging features ($Feats_{G+G}$). Notably, PSL seems unable to simultaneously model the symbolic structure of the two node community types, motivating the application of NEUPSL DEEP WEIGHTS (Q1). Additionally, while GNNs generalize, without explicit encodings of rules (System 2), it still makes mistakes that NEUPSL DEEP WEIGHTS can overcome (Q2).

Table 2 reports the average and standard deviation of the categorical accuracy obtained by each model in the second experimental setting. It is evident that as the features become less representative of the underlying label (i.e., increasing covariance), the performance across all models is negatively affected. However, it is crucial to note that the underlying graph structure, specifically the edges and observed node labels, remains unchanged. Consequently, methods that employ symbolic label propagation, such as PSL and NEUPSL DEEP WEIGHTS, can still achieve reasonably accurate predictions for approximately half of the nodes,

Features	Covariance	MLP	GNN	PSL	NeuPSL Deep Weights
$Feats_{G+OH}$	0.1	88.10 ± 5.69	94.56 ± 1.98	87.06 ± 5.61	**100.00 ± 0.00**
	1.0	88.59 ± 5.87	95.91 ± 2.35	87.06 ± 5.61	**100.00 ± 0.00**
	10.0	81.71 ± 7.31	90.91 ± 4.38	85.23 ± 5.25	**96.35 ± 2.23**
	50.0	76.92 ± 5.63	71.30 ± 7.96	76.11 ± 3.21	**78.85 ± 6.27**
	100.0	**69.70 ± 3.22**	60.53 ± 8.58	69.47 ± 5.06	62.44 ± 5.70
$Feats_{G+G}$	0.1	89.04 ± 2.72	81.75 ± 2.65	86.71 ± 5.46	**93.36 ± 5.06**
	1.0	89.27 ± 2.44	82.42 ± 6.40	86.71 ± 5.46	**92.48 ± 3.99**
	10.0	78.84 ± 1.71	80.15 ± 3.17	84.44 ± 4.69	**91.36 ± 2.61**
	50.0	53.00 ± 4.08	58.30 ± 3.37	**74.18 ± 4.91**	72.57 ± 6.58
	100.0	42.10 ± 4.63	54.14 ± 6.19	**69.39 ± 5.09**	51.66 ± 6.89

Table 2. Average categorical accuracy on varying covariance matrices used for synthetic data generation in the $Feats_{G+OH}$ and $Feats_{G+G}$ data settings. Higher covariance results in a lower correlation between features and labels. Best-performing methods are in bold.

considering that half of the communities are generated from $Rule_{Community}$. One particularly exciting observation is that NeuPSL Deep Weights consistently outperforms other models or performs comparably within the standard deviation of the best models across most settings (Q3). This finding highlights a notable strength of our approach. Regardless of the presence of noise in the low-level features, if a strong signal exists within the problem's structure, NeuPSL Deep Weights will leverage the standard PSL symbolic performance to overcome the limitations posed by the noise.

7. Conclusion

This paper presents NeuPSL Deep Weights, an extended version of the scalable joint neuro-symbolic framework NeuPSL. NeuPSL Deep Weights incorporates the outputs of deep neural networks into the symbolic potential weights of a NeuPSL program. We conduct an investigation on a newly created synthetic dataset designed to evaluate joint learning between System 1 and System 2. We demonstrate that the increased expressiveness introduced by NeuPSL Deep Weights surpasses traditional PSL weight learning methods. Moreover, it outperforms established joint System 1 and System 2 neural techniques such as GNNs.

Although NeuPSL Deep Weights exhibits remarkable success in this synthetic scenario, it is important to consider two limitations. Firstly, NeuPSL Deep Weights was trained exclusively on supervised data, which may lead to challenges in generalization when introducing unsupervised latent variables. To address this, future research can draw inspiration from the success of attention mechanisms in neural networks or explore alternative learning algorithms that handle gradients for latent variables differently. Secondly, the results presented in this paper are based solely on synthetic datasets. Investigating the applicability of NeuPSL Deep Weights to real-world problems is a compelling avenue for future exploration.

Acknowledgments

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

References

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

Bader, S. and Hitzler, P. Dimensions of neural-symbolic integration - A structured survey. *arXiv*, 2005.

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

Besold, T. R., d'Avila Garcez, A. S., Bader, S., Bowman, H., Domingos, P. M., Hitzler, P., Kühnberger, K., Lamb, L. C., Lowd, D., Lima, P. M. V., de Penning, L., Pinkas, G., Poon, H., and Zaverucha, G. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv*, 2017.

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

d'Avila Garcez, A. S., Broda, K., and Gabbay, D. M. *Neural-Symbolic Learning Systems: Foundations and Applications*. Springer, 2002.

d'Avila Garcez, A. S., Lamb, L. C., and Gabbay, D. M. *Neural-Symbolic Cognitive Reasoning*. Springer, 2009.

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

Donadello, I., Serafini, L., and d'Avila Garcez, A. S. Logic tensor networks for semantic image interpretation. In *IJCAI*, 2017.

Evans, R. and Grefenstette, E. Learning explanatory rules from noisy data. *JAIR*, 61:1–64, 2018.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Klir, G. J. and Yuan, B. *Fuzzy Sets and Fuzzy Logic - Theory and Applications*. Prentice Hall, 1995.

Lamb, L. C., d'Avila Garcez, A., Gori, M., Prates, M. O. R., Avelar, P. H. C., and Vardi, M. Y. Graph neural networks meet neural-symbolic computing: A survey and perspective. In *IJCAI*, 2020.

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

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

Serafini, L. and d'Avila Garcez, A. S. Learning and reasoning with logic tensor networks. In *AI*IA*, 2016.

Yang, F., Yang, Z., and Cohen, W. W. Differentiable learning of logical rules for knowledge base reasoning. In *NeurIPS*, 2017.

A. Appendix

The appendix includes the following sections: Computational Hardware Details and Hyperparameters.

B. Computational Hardware Details

All experiments were performed on an Ubuntu 22.04.1 Linux machine with Intel Xeon Processor E5-2630 v4 at 3.10GHz.

C. Hyperparameters

Table 3 and Table 4 summarize the hyperparameters, tuning ranges, and final values for the settings using features $Feats_{OH+OH}$, and the features $Feats_{G+OH}$ and $Feats_{G+G}$, respectively.

Model	Hyperparameter	Tuning Range	Final Value
MLP	Units	{32, 64, 128}	32
	Learning Rate	{1e-2, 1e-3}	1e-2
	Epochs	{1000, 2000}	1000
	Dropout	{0.0, 0.2}	0.0
	Weight Reg.	{1e-3, 1e-6}	1e-6
GNN	Units	{32, 64, 128}	64
	Layers	{1}	1
	Learning Rate	{5e-2, 1e-2, 1e-3}	1e-2
	Epochs	{2000, 2500, 3000}	2000
	Dropout	{0.0, 0.2}	0.0
	Weight Reg.	{1e-3, 1e-6}	1e-6
	Aggregation	{sum}	sum
	Combination	{concat}	concat
Local Rule NEURAL MLP	Units	{32, 64, 128}	64
	Learning Rate	{1e-3, 1e-4}	1e-3
	Epochs	{250, 350, 450, 550}	350
	Dropout	{0.0, 0.2}	0.0
	Weight Reg.	{1e-3, 1e-6}	1e-6
Local Rule Deep Weight MLP	Units	{32, 64, 128}	64
	Learning Rate	{1e-3, 1e-4}	1e-3
	Epochs	{500, 1500, 2500}	500
	Dropout	{0.0, 0.2}	0.0
	Weight Reg.	{1e-3, 1e-6}	1e-6
Label Prop. Rule Deep Weight MLP	Units	{32, 64, 128}	64
	Learning Rate	{1e-3, 1e-4}	1e-3
	Epochs	{500, 1500, 2500}	500
	Dropout	{0.0, 0.2}	0.0
	Weight Reg.	{1e-3, 1e-6}	1e-6

Table 3. Hyperparameters for $Feats_{OH+OH}$.

Dataset	Model	Hyperparameter	Tuning Range	Covariance 0.1	1.0	10.0	50.0	100.0
$Feats_{G+OH}$	MLP	Units	{32, 64, 128}	32	32	32	32	32
		Learning Rate	{1e-2, 1e-3}	1e-2	1e-2	1e-2	1e-2	1e-2
		Epochs	{1000, 2000}	1000	1000	1000	1000	1000
		Dropout	{0.0, 0.2}	0.2	0.2	0.2	0.2	0.2
		Weight Reg.	{1e-3, 1e-6}	1e-6	1e-6	1e-6	1e-6	1e-6
	GNN	Units	{32, 64, 128}	64	64	64	64	64
		Layers	{1}	1	1	1	1	1
		Learning Rate	{5e-2, 1e-2, 1e-3}	5e-2	5e-2	5e-2	5e-2	5e-2
		Epochs	{2000, 2500, 3000}	3000	3000	3000	3000	3000
		Dropout	{0.0, 0.2}	0.2	0.2	0.2	0.2	0.2
		Weight Reg.	{1e-3, 1e-6}	1e-6	1e-6	1e-6	1e-6	1e-6
		Aggregation	{sum}	sum	sum	sum	sum	sum
		Combination	{concat}	concat	concat	concat	concat	concat
	Local Rule NEURAL MLP	Units	{32, 64, 128}	64	64	64	64	64
		Learning Rate	{1e-3, 1e-4}	1e-3	1e-3	1e-3	1e-3	1e-3
		Epochs	{250, 350, 450, 550}	350	350	350	350	350
		Dropout	{0.0, 0.2}	0.0	0.0	0.0	0.0	0.0
		Weight Reg.	{1e-3, 1e-6}	1e-6	1e-6	1e-6	1e-6	1e-6
	Local Rule Deep Weight MLP	Units	{32, 64, 128}	64	64	64	64	64
		Learning Rate	{1e-3, 1e-4}	1e-4	1e-4	1e-4	1e-4	1e-4
		Epochs	{500, 1500, 2500}	2500	2500	2500	2500	2500
		Dropout	{0.0, 0.2}	0.0	0.0	0.0	0.0	0.0
		Weight Reg.	{1e-3, 1e-6}	1e-6	1e-6	1e-6	1e-6	1e-6
	Label Prop. Rule Deep Weight MLP	Units	{32, 64, 128}	64	64	64	64	64
		Learning Rate	{1e-3, 1e-4}	1e-4	1e-4	1e-4	1e-4	1e-4
		Epochs	{500, 1500, 2500}	2500	2500	2500	2500	2500
		Dropout	{0.0, 0.2}	0.0	0.0	0.0	0.0	0.0
		Weight Reg.	{1e-3, 1e-6}	1e-6	1e-6	1e-6	1e-6	1e-6
$Feats_{G+G}$	MLP	Units	{32, 64, 128}	32	32	64	128	32
		Learning Rate	{1e-2, 1e-3}	1e-2	1e-2	1e-2	1e-2	1e-2
		Epochs	{1000, 2000}	1000	1000	1000	1000	1000
		Dropout	{0.0, 0.2}	0.0	0.0	0.2	0.0	0.2
		Weight Reg.	{1e-3, 1e-6}	1e-6	1e-6	1e-6	1e-6	1e-6
	GNN	Units	{32, 64, 128}	64	64	64	64	64
		Layers	{1}	1	1	1	1	1
		Learning Rate	{5e-2, 1e-2, 1e-3}	5e-2	5e-2	5e-2	5e-2	5e-2
		Epochs	{2000, 2500, 3000}	2500	2500	2500	2500	2500
		Dropout	{0.0, 0.2}	0.0	0.0	0.0	0.0	0.0
		Weight Reg.	{1e-3, 1e-6}	1e-6	1e-6	1e-6	1e-6	1e-6
		Aggregation	{sum}	sum	sum	sum	sum	sum
		Combination	{concat}	concat	concat	concat	concat	concat
	Local Rule NEURAL MLP	Units	{32, 64, 128}	64	64	64	64	64
		Learning Rate	{1e-3, 1e-4}	1e-3	1e-3	1e-3	1e-3	1e-3
		Epochs	{250, 350, 450, 550}	350	350	350	350	350
		Dropout	{0.0, 0.2}	0.0	0.0	0.0	0.0	0.0
		Weight Reg.	{1e-3, 1e-6}	1e-6	1e-6	1e-6	1e-6	1e-6
	Local Rule Deep Weight MLP	Units	{32, 64, 128}	64	64	64	64	64
		Learning Rate	{1e-3, 1e-4}	1e-4	1e-4	1e-4	1e-4	1e-4
		Epochs	{500, 1500, 2500}	2500	2500	2500	2500	2500
		Dropout	{0.0, 0.2}	0.0	0.0	0.0	0.0	0.0
		Weight Reg.	{1e-3, 1e-6}	1e-6	1e-6	1e-6	1e-6	1e-6
	Label Prop. Rule Deep Weight MLP	Units	{32, 64, 128}	64	64	64	64	64
		Learning Rate	{1e-3, 1e-4}	1e-4	1e-4	1e-4	1e-4	1e-4
		Epochs	{500, 1500, 2500}	2500	2500	2500	2500	2500
		Dropout	{0.0, 0.2}	0.0	0.0	0.0	0.0	0.0
		Weight Reg.	{1e-3, 1e-6}	1e-6	1e-6	1e-6	1e-6	1e-6

Table 4. Hyperparameters for $Feats_{G+OH}$ and $Feats_{G+G}$.