HyperLogic: Enhancing Diversity and Accuracy in Rule Learning with HyperNets

Yang Yang[1], Wendi Ren[1], Shuang Li[1]*
[1]School of Data Science, The Chinese University of Hong Kong (Shenzhen)
{yangyang8, wendiren}@link.cuhk.edu.cn, lishuang@cuhk.edu.cn

Abstract

Exploring the integration of if-then logic rules within neural network architectures presents an intriguing area. This integration seamlessly transforms the rule learning task into neural network training using backpropagation and stochastic gradient descent. From a well-trained sparse and shallow neural network, one can interpret each layer and neuron through the language of logic rules, and a global explanatory rule set can be directly extracted. However, ensuring interpretability may impose constraints on the flexibility, depth, and width of neural networks. In this paper, we propose HyperLogic: a flexible approach leveraging hypernetworks to generate weights of the main network. HyperLogic can be combined with existing differentiable rule learning methods to generate diverse rule sets, each capable of capturing heterogeneous patterns in data. This provides a simple yet effective method to increase model flexibility and preserve interpretability. We theoretically analyze the benefits of the HyperLogic by examining the approximation error and generalization capabilities under two types of regularization terms: sparsity and diversity regularization. Experiments on real data demonstrate that our method can learn more diverse, accurate, and concise rules. Our code is publicly available at https://github.com/YangYang-624/HyperLogic.

1 Introduction

Despite the significant impact of deep learning on society, its lack of interpretability limits its use in critical areas that demand high transparency. For instance, in high-risk domains such as healthcare, finance, and law, the decision-making process of models needs to be fully open and explainable to users and relevant regulatory authorities to gain necessary trust and legitimacy [1, 2, 3]. Compared to the "black box" models, which may perform well but are uninterpretable, people prefer intrinsically interpretable model for decision support [4, 5], such as a set of concise IF-THEN rules.

Traditional rule learning methods, which are based on statistical or heuristic approaches, have been extensively explored but often struggle to simultaneously achieve two key objectives: 1) simplicity and accuracy in rules, and 2) noise tolerance and scalability in data handling [6, 7, 8, 9, 10]. These methods typically rely on search-based techniques, which can be limited when dealing with complex and noisy data. In contrast, deep learning, which focuses on representation learning through embedding, is robust to noise and effectively manages large datasets. This has led to the exploration of differentiable rule learning [11], which leverages the high performance of deep learning while ensuring interpretability through explicit rule formulation

Differentiable rule learning primarily includes two types of methods. The first approach outputs rules directly through the network, leveraging powerful and diverse neural models to enhance handling of complex data patterns [12, 13, 14]. The second approach is to extract rules from network weights [15, 16, 17], which promotes rapid training and efficient data management while enabling the use of predefined structures to integrate expert priors [18].

*Corresponding author

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

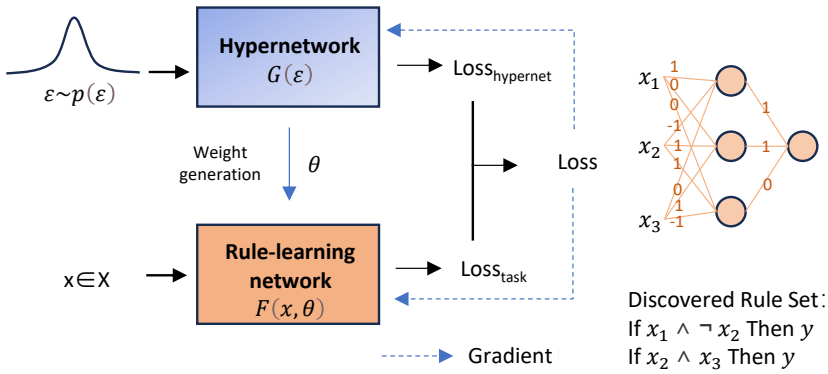

Figure 1: The framework of HyperLogic: Hypernetwork generates θ for the main network, which is a rule-learning network. An example of a main rule-learning network is shown on the right, from which rules can be extracted based on the learned network weights.

We focus on the second approach, aiming to extract rules like "IF $a \wedge b \wedge \neg c$ THEN y is True" from network weights, where a, b, c are predefined predicates. Although integrating if-then logic rules within neural network architectures offers clear advantages, their performance is often hindered by the restrictive network structures required for interpretability. These structures tend to be overly simplistic and not scalable, limiting their ability to capture complex data patterns and making them prone to only capturing partial or local patterns in the data. This raises a critical question: How can we modify these models to better harness the full potential of neural networks without compromising their interpretability?

In this work, we introduce **HyperLogic**, a novel framework that integrates hypernetworks with a main rule-learning network (as illustrated in Fig. 1, where an example of a main rule-learning network is shown on the right). Hypernetworks are a type of neural network that generate weights for a main rule-learning network. In our framework, the hypernetwork takes random samples from a high-dimensional Gaussian distribution as input and outputs weights for different parts of the main network. These weights can be utilized in two ways: either directly as the weights for the main network throughout the training process (meaning the main network itself has no parameters), or by retaining trainable weights in the main network and combining them with the hypernetwork-generated weights through weighted fusion. In both scenarios, this approach yields an interpretable set of rules derived from the comprehensive weights.

HyperLogic can be seen as a mixture of expert model with an infinite number of experts, providing a simple yet effective way to enhance model flexibility and adaptability. We analyze the benefits of HyperLogic by examining its approximation error and generalization capabilities under two types of regularization: sparsity and diversity regularization. Our findings demonstrate that HyperLogic acts as a universal approximation estimator in the Barron space [19] and proves its generalization ability across various regularization methods. These theoretical benefits are further supported by our empirical experiments.

Additionally, hypernetworks enable the generation of multiple candidate rule sets in a single training session without significantly increasing computational overhead. This is in stark contrast to traditional methods, which typically learn only one set of rules at a time. Consequently, HyperLogic greatly enhances the efficiency and flexibility of the rule-learning process.

We summarize our contributions as follows:

1. We introduce HyperLogic, a pioneering framework that integrates hypernetworks into differentiable rule learning, significantly enhancing the rule-learning landscape.
2. We theoretically justify the performance of HyperLogic and provide insights into its effectiveness. Specifically, we examine the approximation error and generalization capabilities under two types of regularization: sparsity and diversity.
3. We validate HyperLogic through extensive experiments on multiple datasets, demonstrating that it enables the learning of multiple diverse rule sets and yields more concise and accurate rules.

2 Related Work

Learning Rules from Network Weights Many methods for extracting rules from model weights are not based on neural networks; instead, they primarily focus on interpreting the weights of smaller,

simpler models to discover rules [20, 7]. Recently, extensive methods have emerged that attempt to extract rules using neural network models. These approaches often involve data preprocessing techniques to prepare the data before training [21] or utilize post-hoc methods for rule extraction after model training [22, 23, 24], which can often lead to a loss of accuracy.

A more promising alternative allows for the direct interpretation of neural network weights as precise rules by integrating rule extraction directly during training. This leads to a more seamless and effective learning process. For example, methods such as [25, 26, 27] utilize low-dimensional embeddings to represent atomic conditions and rules, treating rule learning as a differentiable discrete combinatorial optimization problem encoded by a feedforward neural network. Specifically, FinRule [27] explores higher-order interactions between atomic conditions, which aligns somewhat with our approach of employing hypernetworks. However, it lacks a detailed analysis of the specific roles of these higher-order interactions and cannot learn a rich set of candidate rules as our method does. Additionally, these approaches often impose constraints on rule templates, such as fixing the rule length and the number of rules.

Another common class of methods employs modified simple feedforward neural networks, such as DR-net [17] and others [28, 29, 30]. These methods simulate logical operations by altering activation functions and implementing mechanisms that ensure differentiability and support backpropagation. While they overcome the limitations of rigid rule templates, their model structures tend to be relatively simple. Furthermore, the training process often involves freezing the weights of one component while adjusting another, which restricts the model's ability to fully optimize performance.

Our HyperLogic is a unified framework that can be integrated with various neural network weight generators, enabling easy adaptation to different main rule learning networks. Currently, our main network draws inspiration from DR-net [17], but it can be replaced with other architectures based on task requirements. Unlike recent Bayesian approaches, which often require multiple hyperparameters to model uncertainty in network weights, HyperLogic simplifies the process by needing fewer hyperparameters while maintaining scalability. This allows for efficient training and adaptability across various tasks without the added complexity of managing uncertainty. Additionally, our approach can generate multiple sets of rule sets in a single training session, a capability not available in other methods. For further comparisons, including classic rule-based algorithms, please refer to Appendix A.

Hypernetwork Hypernetworks, or hypernets, are neural networks that generate weights for main networks [31]. Benefiting from over-parameterization and strategic design, these networks enhance training flexibility and adaptability, improve information sharing, and accelerate training processes. While widely used in many areas like continual learning [32, 33], transfer learning [34], uncertainty quantification [35, 36], natural language processing [37] and computer vision [38], their potential in rule learning remains largely untapped. Our HyperLogic framework can bridge this gap. Specifically, our hypernetwork takes inspirations from HyperGAN [36] and similarly we add a loss term to encourage the diversity of the generated network weights.

Unlike typical applications in other domains that focus on enhancing parameter efficiency and training speed, our approach addresses the unique challenges of adding model flexibility while preserving interpretability. Our main goal is to expand the parameter space to produce more diverse, concise and accurate rule sets.

3 HyperLogic

HyperLogic is a versatile framework designed to enhance various existing neural rule-learning methods. The concept of hypernetworks can be applied to different types of main networks, provided they focus on learning a set of interpretable weights in a differentiable manner. Importantly, HyperLogic imposes no additional restrictions on data formats or rule languages; it simply builds upon the capabilities of the chosen main network.

3.1 Main Network: Rule-Learning Network

Our main network is currently based on DR-Net [17], which features a simple architecture and is designed for binary classification tasks. We chose this architecture for its simplicity, which aids in our proofs and explanations. However, we recognize its limitations, so we can also explore other more flexible neural approaches as the primary network to overcome restrictions related to data formats and rule languages. This is further discussed in Appendix B.

The main network is a two-layer neural network as shown in Fig. 1 (right part). Due to its interpretability, one can directly extract the if-then rules from the trained neural networks. The input layer of the main network is fed with a D-dimensional binary data $x = [x_d]$, each element $x_d \in \{1, -1\}$. Here x_d indicates the grounded predicate, where 1 indicates True and -1 means false. Note that common input types may include binary, categorical, or numerical features, all of which are discretized and binarized to form our binary input features. The hidden layer has K neurons, where K determines the total number of rules and serves as a hyperparameter. The first layer is referred to as the Rule Layer, and the output layer as the OR layer.

Rule Layer: Each neuron in the hidden layer is denoted as $o_k \in \{0, 1\}$. The output represents whether a rule is satisfied. Each neuron is calculated as:

$$o_k = \mathbb{1}\left\{\sum_{d=1}^{D} w_{d,k} x_d - \sum_{d=1}^{D} |w_{d,k}| = 0\right\}, \quad \text{for } k = 1, \ldots, K \tag{1}$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function. Note that the indicator function is one if and only if all inputs match the sign of the corresponding weights: all positive weights should have the inputs of 1, and all negative weights should have the inputs of -1, and zero weights mean that the corresponding inputs are excluded from the rule.

Note that the indicator function is non-differentiable, which will make the gradient hard to compute. Here, we will instead use a differentiable smooth function to approximate the indicator function (this will also ease the theoretical analyses). For example, we can use $h(u) = \exp\left(-\frac{u^2}{\tau}\right)$, $\tau > 0$, to approximate $\mathbb{1}\{u = 0\}$, where τ is the tunable temperature and controls the approximation error.

OR Layer: The second layer operation is defined as

$$f(x) = \sum_{k=1}^{K} u_k o_k \tag{2}$$

where we assume that the rules contribute to the final prediction in a weighted additive form to reflect the OR composition, where u_k denotes the weight assigned to the k-th rule.

3.2 Hypernetwork: Generate Network Weights

Define the network weights for the previous model as $\theta = (w, u)$, where $w \in R^{D \times K}$, and $u \in R^K$. Instead of learning only one set of θ, we will learn the distribution of θ, denoted as μ. Specifically, we introduce a generative model as the hypernetwork to produce samples of θ, i.e.,

$$\theta = G(\epsilon), \quad \text{where } \epsilon \sim p(\epsilon), \quad \theta \in R^{(D+1)K}. \tag{3}$$

Here, ϵ is drawn from some simple base distribution such as Gaussian distribution. More details about the hypernetwork structure can be seen in Appendix C. Note that, in this way, our model parameters have been changed to μ, which is defined as the distribution of θ, and the proposed Hyperlogic model is

$$f_\mu(x) = \mathbb{E}_\mu \left[\sum_{k=1}^{K} u_k h\left(\sum_{d=1}^{D} w_{d,k} x_d - \sum_{d=1}^{D} |w_{d,k}|\right)\right]. \tag{4}$$

In practice, we can use the Monte Carlo method to estimate the above expectation by randomly drawing M samples from μ, denoted as $\theta^1, \ldots, \theta^M$, and we define

$$f_M(x) = \frac{1}{M} \sum_{m=1}^{M} \left[\sum_{k=1}^{K} u_k^m h\left(\sum_{d=1}^{D} w_{d,k}^m x_d - \sum_{d=1}^{D} |w_{d,k}^m|\right)\right]. \tag{5}$$

This model can be regarded as a mixture of expert models with finite M experts. Each expert is a shallow two-layer neural network that encodes at most K if-then rules.

3.3 Expand to High-Dimensional Data

When working with high-dimensional data (e.g., 5000 dimensions), the number of parameters in the main network increases proportionally with the input dimensions, even if the network structure remains unchanged. This makes training a hypernetwork to generate weights for each module of the main network more challenging. To tackle this issue, we considered two strategies:

Combining Original Weights with Hypernetwork-Generated Weights Instead of allowing the hypernetwork-generated weights (W_{hyper}) to fully dictate the main network's weights, we combine them with the original weights (W_{main}). The final weights are calculated as follows:

$$W_{\text{main-final}} = \alpha \cdot W_{\text{hyper}} + (1 - \alpha) \cdot W_{\text{main}},$$

where α is a learnable parameter constrained between 0 and 1. This strategy enhances stability without sacrificing the hypernetwork's ability to produce diverse weights. If the hypernetwork is poorly trained and generates inappropriate weights, the model adjusts α toward 0, effectively reverting to the standard main network without hypernetwork influence. We employed this strategy in our subsequent experiments.

Generating Weights for Only Some Modules of the Main Network We test this strategy when the dataset dimension and size are very large in the supplementary experiment. It shows that generating only part of the main network's weights using the hypernetwork still significantly improves results. For more details, please see the supplementary experiments in Appendix B.

3.4 Loss Function

One can learn μ by minimizing the loss function $\ell(y, f_\mu(x))$, where $f_\mu(x)$ can be approximated by the finite-sample estimator in Eq. (5). Our loss contains two parts. One is the task-related loss function, denoted as $\ell_{task}(y, f_\mu(x))$. Suppose the problem is a binary classification problem, one can use the binary cross-entropy loss (BCE), where we first map $f_\mu(x)$ to a probability value between 0 and 1 using a link function such as sigmoid and the loss is defined as the negative log-likelihood of a sample belong to a class. Here, the task-related loss measures the prediction accuracy.

The second part of the loss is the regularization loss, denoted as ℓ_{reg}, which is introduced to encourage the diversity of the generated θ and model sparsity (rule simplicity for each expert), i.e.,

$$\ell_{reg} = \lambda_1 D_{\text{KL}}(\mu \| \mu_0) + \lambda_2 \mathbb{E}_\mu[|u|]. \tag{6}$$

For the first regularization term, we introduce a prior distribution μ_0 with a high entropy, and we minimize the relative entropy or KL divergence of the μ and μ_0. Minimizing the relative entropy encourages the diversity of the generated θ. For the second regularization term, we are considering the ℓ_1 sparsity norm for the second OR layer's weights, where we hope that for each expert, the discovered number of rules is as compact as possible. In our paper, ℓ_{reg} represents the loss incurred by the hypernetwork, as illustrated in Fig. 1.

4 Theoretical Analysis

We will employ theoretical analysis to demonstrate that while the main network (i.e., a two-layer shallow neural network) has low capacity, incorporating the hypernet idea significantly enhances the model's expressive power. Specifically, we will establish that the proposed HyperLogic model serves as a universal approximator in the Barron space [19], which will be defined later. For simplicity, but without loss of generality, we consider the case where the hypernetwork fully generates the weights and the main network itself has no parameters. Additionally, we will present the generalization error under the above two types of regularization.

Reparametrization: To ease the proof, let's first rewrite Eq. (5) as

$$f_M(x) = \frac{1}{M} \sum_{m=1}^{M} \left[\sum_{k=1}^{K} u_k^m h\left(w_k^{m\top} x \right) \right] \tag{7}$$

by reparametrization. To achieve this, we first use the split variable trick by defining $w^+ = \max\{w, 0\}$ and $w^- = -\min\{w, 0\}$, and reparametrize $w = w^+ - w^-$and $|w| = w^+ + w^-$. In this way, we can get rid of the absolute value and have

$$\sum_{d=1}^{D} w_{d,k} x_d - \sum_{d=1}^{D} |w_{d,k}| = \sum_{d=1}^{D} w_{d,k}^+ (x_d - 1) + \sum_{d=1}^{D} w_{d,k}^- (-x_d - 1). \tag{8}$$

Therefore, we can simply reparametrize w_k, where $w_k \geq 0$, as a concatenation of the vector $[w_{d,k}^+]_{d=1,\ldots,D}$ and $[w_{d,k}^-]_{d=1,\ldots,D}$. We construct the new input data x by making a copy of the original data and modifying each copy given Eq. (8). That is, for one copy, we subtract it by 1, and for another copy, we flip the sign and subtract it by 1. Then, we concatenate the two copies of data.

Given such a reparametrization, we get a simple form as Eq. (7). In the following analysis, we will still assume that $w_k \in R^D$ and $x \in R^D$, which doesn't affect the generality.

Preparation for the Theoretical Analysis: Our analysis below is based on the following observations. Given the generated random variable $((w_1, u_1), \ldots, (w_K, u_K)) \in \mathbb{R}^{(D+1)K}$, we have denoted their joint distribution as μ. Let us further denote the marginal distribution of (w_k, u_k) as μ_k, $k = 1, \ldots, K$, respectively. Define a random variable $(\tilde{w}, \tilde{u}) \in \mathbb{R}^{D+1}$, which draws a sample from μ_k with (equal) probability $1/K$, $k = 1, \ldots, K$, and then scale the u-component by K. Denote the resulting mixture distribution as $\tilde{\mu}$. Then it follows from the linearity of expectation that

$$\begin{aligned} f_\mu(x) &= \mathbb{E}_{((w_1,u_1),\ldots,(w_K,u_K))\sim\mu}\left[\sum_{k=1}^{K} u_k h\left(w_k^\top x\right)\right] \\ &= \mathbb{E}_{((w_1,u_1),\ldots,(w_K,u_K))\sim\mu}\left[\frac{1}{K}\sum_{k=1}^{K} \tilde{u}_k h\left(w_k^\top x\right)\right] = \mathbb{E}_{(\tilde{w},\tilde{u})\sim\tilde{\mu}}\left[\tilde{u} h\left(\tilde{w}^\top x\right)\right]. \end{aligned} \tag{9}$$

We define $\tilde{f}_{\tilde{\mu}}(x) := \mathbb{E}_{(\tilde{w},\tilde{u})\sim\tilde{\mu}}\left[\tilde{u} h\left(\tilde{w}^\top x\right)\right]$ and we have shown that $f_\mu(x) = \tilde{f}_{\tilde{\mu}}(x)$ as stated above.

The motivation for introducing $\tilde{f}_{\tilde{\mu}}(x)$ is to facilitate the analysis of approximation error and generalization error. There exist theoretical results for a two-layer neural network of the form $g_M(x) = \sum_{m=1}^{M} u^m h\left(w^{m\top} x\right)$, where $h(\cdot)$ belongs to certain classes of smooth activation functions, and (w^m, u^m), $m = 1, \ldots, M$, are M independent samples drawn from a fixed distribution. In contrast, our proposed HyperLogic model involves randomly drawing samples $((w_1, u_1), \ldots, (w_K, u_K))$ from μ, each with K components. To align this with the existing theoretical framework, we perform a transformation as shown in Eq. (9). This involves converting the process of drawing M samples, each with K components, into drawing MK samples, each with a single component. This reformulation maintains the same expectation results, allowing us to leverage existing theoretical results to analyze HyperLogic. Note that the conversion mentioned is purely for theoretical proof purposes and is not implemented in the actual algorithm.

4.1 Approximation Error of Finite Experts

Using the connection as shown in Eq. (9), we can directly leverage the approximation error results for a single hidden layer neural network. Let's first define the Barron space, which provides a set of functions for which neural networks can achieve good approximation properties.

Definition 1 (Barron Space [19])**.** A function $f : \mathbb{R}^D \to \mathbb{R}$ belongs to the Barron space $\mathcal{B}$ if it can be represented as:

$$f(x) = \int_{\mathbb{R}\times\mathbb{R}^D\times\mathbb{R}} uh\left(w^\top x + b\right) d\mu(u, w, b)$$

where h is an activation function, μ is a probability measure on $\mathbb{R} \times \mathbb{R}^D \times \mathbb{R}$, and the following Barron norm is finite:

$$\|f\|_{\mathcal{B}} = \inf\left\{\int_{\mathbb{R}\times\mathbb{R}^D\times\mathbb{R}} |u|\|(w, b)\| d\mu(u, w, b) : f(x) = \int_{\mathbb{R}\times\mathbb{R}^D\times\mathbb{R}} uh\left(w^\top x + b\right) d\mu(u, w, b)\right\}.$$

Note that the representation of f in the form $f(x) = \int uh\left(w^\top x + b\right) d\mu(u, w, b)$ may not be unique. The Barron norm seeks the representation that minimizes the integral of $|u|\|(w, b)\|$. The introduction of the Barron norm provides a quantitative measure of the "complexity" of a function in the context of neural networks. Functions with a smaller Barron norm are considered simpler and are easier to approximate with neural networks. Next, let's provide the approximation error analysis for our HyperLogic model (the proof can be found in Appendix D):

Theorem 1. *For any function f in the Barron space, there exist M experts $((w_1^m, u_1^m), \ldots, (w_K^m, u_K^m)) \in \mathbb{R}^{(D+1)\times K}$, $m = 1, \ldots, M$, forming a predictor f_M as shown in Eq. (7), such that*

$$\|f - f_M\|_{L^2} \leq \frac{\|f\|_{\mathcal{B}}}{\sqrt{MK}},$$

where the L^2 norm for a function $f : \mathbb{R}^D \to \mathbb{R}$ is defined as $\|f\|_{L^2} = \left(\int_{\mathbb{R}^D} |f(x)|^2 d\mathcal{D}(x)\right)^{\frac{1}{2}}$ where $\mathcal{D}(x)$ is the probability distribution over the input data.

The above theorem asserts that a HyperLogic model (essentially a mixture of expert models) can approximate any continuous function in the Barron space to arbitrary precision given enough M, K and appropriate weights.

4.2 Generalization Error: Entropic Regularization and Sparse Regularization

Let's derive the generalization error bounds for the HyperLogic model under entropic regularization (the first term of Eq. (6)) and sparse regularization (the second term of Eq. (6)). The key steps involve calculating the Rademacher complexity of the model class and then using this to bound the generalization error. The proof can be found in Appendix E. Let's give the results here.

Theorem 2. *For the function class* $\mathcal{F}_{\mathrm{KL}} := \{f_\mu(\cdot) : D_{\mathrm{KL}}(\mu\|\mu_0) \leq B_{\mathrm{KL}}\}$*, we have*

$$\mathbb{E}_{\mathcal{D}}[\ell(f)] - \hat{\ell}(f) \leq 2\sqrt{\frac{2B_{\mathrm{KL}}}{n}} + C\sqrt{\frac{\log(1/\delta)}{2n}}. \tag{10}$$

Similarly, for the function class $\mathcal{F}_1 := \left\{f_\mu(\cdot) : \mathbb{E}_\mu\left[\frac{1}{K}\sum_{k=1}^{K}|u_k|\right] \leq B_1\right\}$*, we have*

$$\mathbb{E}_{\mathcal{D}}[\ell(f)] - \hat{\ell}(f) \leq 64B_1\sqrt{(D+1)/n} + C\sqrt{\frac{\log(1/\delta)}{n}}.$$

Here, $\mathbb{E}_{\mathcal{D}}[\ell(f)]$ *is the expected loss over data distribution,* $\hat{\ell}(f)$ *is the empirical loss,* n *is the number of samples,* C *is a constant dependent on the loss function, and* δ *is the confidence level.*

From the results, we see that the relative entropy regularization effectively balances fitting the training data and adhering to the prior distribution with high entropy. Increasing the sample size or decreasing the KL bound enhances the model's generalization ability, ensuring good performance on unseen data. Similar conclusions can be arrived for the sparse regularization.

5 Experiment

In this section, we report experimental results to answer the following questions:

- **RQ1:** How does the performance of the optimal rule set selected by HyperLogic compare to the rule sets obtained by other methods?
- **RQ2:** How rich are the rule sets generated by HyperLogic, and how are their accuracy and diversity affected by parameters?
- **RQ3:** Can we further leverage the advantages of HyperLogic through ensemble learning to enhance performance?

5.1 Experiment Setup

Implementation Details: During training, for each data batch, we randomly generate M_1 samples of network weights to approximate the expectation (as shown in Eq. (5), here we compute $f_{M_1}(x)$ as an approximation in the training stage). Increasing M_1 enhances the stability of hypernetwork training but raises computational costs. In our experiments, M_1 is set to 5 or 10 . After training, we generate M_2 sets of weights from the hypernetwork, resulting in M_2 rule sets. In other words, in the inference stage, we use $f_{M_2}(x)$ instead. While M_1 is relatively small, M_2 can be large (e.g., 5000). We then select the rule set with the highest training accuracy as the optimal set, though other criteria, such as minimal loss or custom evaluation metrics balancing accuracy and complexity, can also be used.

For HyperLogic, We use Adam as the optimizer, and the learning rate is 1×10^{-4}, with weight decay is 1×10^{-4}. The number of training epochs is 10000. In the experiments for selecting the optimal rules for comparison, we set hyperparameter $M_1 = 5$, $M_2 = 5000$, $\lambda_1 = 0.01$, and $\lambda_2 = 0.1$ (λ_1 and λ_2 are related to the hypernetwork loss or regularization loss, as defined in Eq. (6)). In the following experiments that analyze the influence of hyperparameter, we adjusted only the corresponding hyperparameter while keeping others unchanged.

Datasets: We selected four publicly available binary classification datasets: MAGIC gamma telescope (magic), adult census (adult), FICO HELOC (heloc), and home price prediction (house). In all datasets, preprocessing was performed to encode categorical and numerical attributes as binary variables, which can be found in [17]. We compared our model HyperLogic with other state-of-the-art rule learning methods: DR-Net [17], CG [7], BRS [20], and RIPPER [6]. Since CG, BRS, and RIPPER cannot learn negative conditions, we additionally appended negative conditions for these models.

5.2 Performance Comparison

In the performance comparison section, we sampled $M_2 = 5000$ times from HyperLogic, selecting the rule set that performed best on the training set as the optimal rule set, and compared it with other methods. Table 1 presents the comparison of the accuracy of the optimal rule sets selected by

our method and those generated by other methods. Our method further improves upon DR-Net and outperforms other methods across all four datasets.

It is important to note that the rule set that performs best on the training set does not strictly guarantee the best performance on the test set, as shown in Fig. 2. However, this selection method is sufficiently simple, and experiments have shown that it can achieve good performance.

Table 1: Test accuracy based on a nested 5-fold cross-validation (%, mean ± standard error). Results corresponding to methods marked with * are directly sourced from [17].

Method	**Dataset**			
	magic	adult	house	heloc
HyperLogic	**84.90** ± 0.73	**83.11** ± 0.55	**85.22** ± 0.62	**71.03** ± 1.07
DR-Net	83.69 ± 0.55	82.95 ± 0.45	85.05 ± 0.51	70.07 ± 0.83
CG*	83.68 ± 0.87	82.67 ± 0.48	83.90 ± 0.18	68.65 ± 3.48
BRS*	81.44 ± 0.61	79.35 ± 1.78	83.04 ± 0.11	69.42 ± 3.72
RIPPER*	82.22 ± 0.51	81.67 ± 1.05	82.47 ± 1.84	69.67 ± 2.09

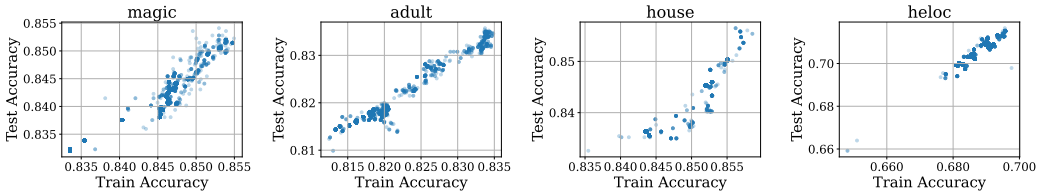

Figure 2: Train and test accuracies for sampled rule sets across four datasets

In addition to rule accuracy, we also considered two metrics to measure the compactness of the rules: model complexity and rule complexity. Model complexity is defined as the sum of the number of rules and the total number of conditions in the rule set; rule complexity is the average number of conditions in each rule of the model. Fig. 3 shows that our method reduces both model complexity and rule complexity compared to DR-Net, indicating that we have not only improved the accuracy of the rules but also made them more concise, demonstrating the effectiveness of our framework.

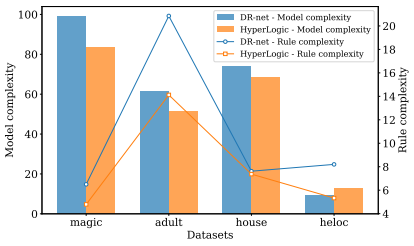

Figure 3: Model complexity and rule complexity comparison

5.3 Rule Analysis

In this section, we primarily considered the effects of three parameters, M_1, λ_1, and M_2, on the learned rules. Among these, M_2 primarily affects the selection of the final optimal rule set by controlling the number of candidate rule sets sampled from the hypernetwork after training phase, while M_1 and λ_1 directly influence the training of the hypernetwork.

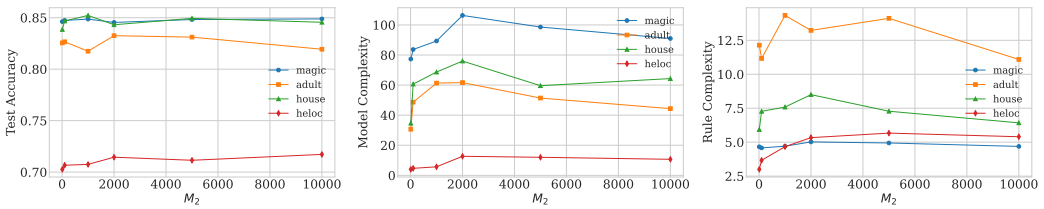

Figure 4: Analysis of the impact of M_2 on all datasets

For M_2, we considered the values 10, 100, 1000, 2000, 5000, and 10000 to examine its impact on the accuracy, model complexity, and rule complexity of the generated optimal rules, as shown in Fig.

4. It can be observed that the accuracy, model complexity, and rule complexity of the optimal rules generally increase with the increase of M_2, and then level off or slightly decrease. This is because, when M_2 is small, we may not be able to sample sufficiently suitable rule sets to fit the training set. When M_2 is too large, the sampled optimal rule set is more likely to overfit the training set, leading to a slight decrease in test accuracy.

So far, we have focused on the optimal rule set extracted from the hypernetwork. Table 2 further shows the different optimal results obtained in three training sessions for the heloc dataset, indicating that we can find different high-quality rule sets in different training sessions. However, considering only one optimal rule set is not enough to reflect the advantage of the hypernetwork in generating multiple high-quality rule sets in one training session, which provides more options and a deeper understanding of the data, something that methods learning only one set of rules cannot offer.

Table 2: Examples of optimal rule sets learned from different training runs on the heloc dataset

Version	Rule	Train Acc	Test Acc
1	①$\neg$AvgFile $\leq 40.0 \wedge \neg$ExtRisk $\leq 69.0 \wedge \neg$PctNeverDelq $\leq 78.0 \wedge$ NetFracBurden ≤ 77.0	69.28	71.51
2	①$\neg$AvgFile $\leq 40.0 \wedge \neg$ExtRisk $\leq 66.0 \wedge \neg$OldTradeOpen $\leq 87.0 \wedge$ $\neg$PctNeverDelq $\leq 78.0 \wedge$ NetFracBurden ≤ 60.0 ②$\neg$MaxDelq2Rec12M $\leq 5.0 \wedge$ BankTradesHighUtil $\leq 2.0 \wedge$ $\neg$ExtRisk ≤ 74.0	70.59	71.41
3	①$\neg$ExtRisk $\leq 74.0 \wedge \neg$PctNeverDelq $\leq 92.0 \wedge$ NetFracBurden ≤ 47.0	69.95	69.60

The ability of the hypernetwork to generate diverse rules is mainly related to M_1 and λ_1. λ_1, as the coefficient of diversity regularization term, directly affects the diversity of the hypernetwork output, while M_1 indirectly affects the process by influencing the stability of hypernetwork updates. For the impacts of M_1 and λ_1, we are no longer concerned with the optimal rule set but focus on all the rule sets generated by the hypernetwork, and describe the diversity and accuracy of the rule sets as a whole.

The diversity of the rules is measured in two aspects: 1. How many different rule sets can we generate by sampling a certain number of weights from the hypernetwork? 2. What is the degree of similarity between the generated different rule sets? We use Jaccard similarity to measure this.

We take the magic dataset as an example to analyze M_1 and λ_1; the results for the remaining datasets are in Appendix F. The impact of M_1 on rule diversity and test accuracy is shown in Fig. 5. It can be seen that when $M_1 = 5$, there is the best performance in all aspects. This may be because, when M_1 is small, the hypernetwork generates weights fewer times, making the training less stable. When M_1 is too large, the training and updates of the hypernetwork become too stable and conservative, slowing down the convergence speed and resulting in a performance decline of the hypernetwork.

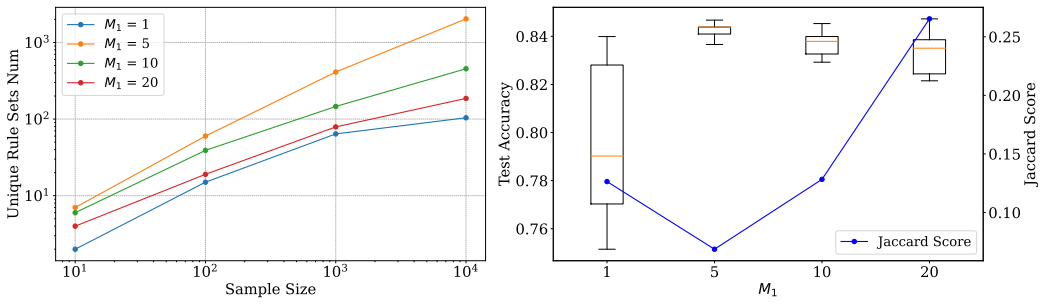

Figure 5: Analysis of the impact of M_1 on magic dataset

The impact of λ_1 on rule diversity and test accuracy is shown in Fig. 6. It can be seen that increasing λ_1 significantly enhances the diversity of the rule sets, but excessive emphasis on rule diversity may affect the accuracy of the rules.

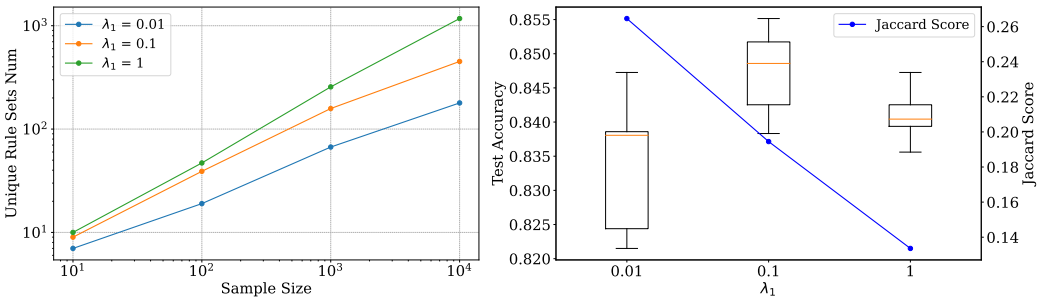

Figure 6: Analysis of the impact of λ_1 on magic dataset

5.4 Ensemble Learning

Since the hypernetwork can generate a diverse and rich set of rule sets, a natural idea is to use ensemble learning to achieve better classification performance. We use the simple averaging voting method for ensemble learning. We select the top L rule sets based on their accuracy on the training set and report their accuracy on the test set. The values of L are 1, 5, 10, 30, 50, 100, and 200. As shown in Fig. 7, the test accuracy initially increases with the increase of L, then slightly decreases.

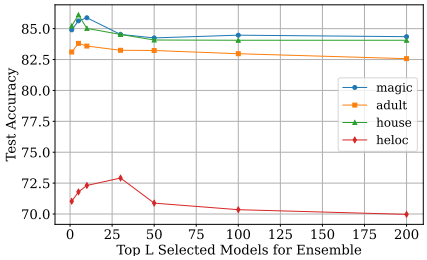

Figure 7: Test accuracy using top L rule sets in ensemble learning.

This may because the top single rule set is already highly effective, leaving little room for improvement. When L is too large, less effective rule sets decrease the ensemble's overall performance. Additionally, our current strategy focuses on balancing diversity and accuracy of single rule sets. Adjusting the strategy specifically for ensemble learning could yield better results.

6 Conclusion and Limitations

In this paper, we proposed HyperLogic, a novel framework that enhances the field of differentiable rule learning through the integration of hypernetworks. We also provided a theoretical foundation for HyperLogic's performance, explaining its effectiveness. This includes an analysis of approximation error and generalization capabilities under sparsity and diversity regularization. What's more, we conducted extensive experiments on multiple datasets, demonstrating that HyperLogic accelerates the training process while producing more concise and accurate rules.

Despite these advancements, HyperLogic introduces additional hyperparameters and requires further exploration of ensemble learning within this framework. Future research will focus on alternative training strategies, more stable weight combination methods, and applying HyperLogic to a broader range of datasets and tasks to enhance its generality and practicality.

7 Broader Impacts

The development and utilization of interpretable logic rules through HyperLogic have significant positive societal impacts, particularly in high-risk domains such as healthcare, finance, and legal systems. By generating multiple candidate rule sets, HyperLogic provides more flexible and comprehensive insights, enhancing transparency and trust in decision-making processes. This is crucial for ensuring safety and regulatory compliance. However, it is important to note that while our method provides richer rule sets and aids experts in understanding data patterns, these rules should complement rather than replace expert judgment. Relying solely on automated rules without expert oversight in critical areas could pose significant risks. Thus, our approach emphasizes the collaborative role of machine learning models and human experts to mitigate potential negative impacts.

Acknowledgments

Shuang Li's research was in part supported by the National Science and Technology Major Project under grant No. 2022ZD0116004, the NSFC under grant No. 62206236, Shenzhen Stability Science Program 2023, Shenzhen Key Lab of Cross-Modal Cognitive Computing under grant No. ZDSYS20230626091302006, Longgang District Key Laboratory of Intelligent Digital Economy Security, and SRIBD Innovation Fund SIF20240010.

References

[1] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

[2] Michelle Goddard. The eu general data protection regulation (gdpr): European regulation that has a global impact. *International Journal of Market Research*, 59(6):703–705, 2017.

[3] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. Xai—explainable artificial intelligence. *Science robotics*, 4(37):eaay7120, 2019.

[4] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[5] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys*, 16:1–85, 2022.

[6] William W Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier, 1995.

[7] Sanjeeb Dash, Oktay Gunluk, and Dennis Wei. Boolean decision rules via column generation. *Advances in neural information processing systems*, 31, 2018.

[8] Mark Law, Alessandra Russo, Elisa Bertino, Krysia Broda, and Jorge Lobo. Fastlas: Scalable inductive logic programming incorporating domain-specific optimisation criteria. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2877–2885, 2020.

[9] Andrew Cropper and Rolf Morel. Learning programs by learning from failures. *Machine Learning*, 110(4):801–856, 2021.

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

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

[12] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems*, 32, 2019.

[13] Yuan Yang and Le Song. Learn to explain efficiently via neural logic inductive learning. *arXiv preprint arXiv:1910.02481*, 2019.

[14] Kewei Cheng, Nesreen K Ahmed, and Yizhou Sun. Neural compositional rule learning for knowledge graph reasoning. *arXiv preprint arXiv:2303.03581*, 2023.

[15] Hikaru Shindo, Masaaki Nishino, and Akihiro Yamamoto. Differentiable inductive logic programming for structured examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5034–5041, 2021.

[16] Claire Glanois, Zhaohui Jiang, Xuening Feng, Paul Weng, Matthieu Zimmer, Dong Li, Wulong Liu, and Jianye Hao. Neuro-symbolic hierarchical rule induction. In *International Conference on Machine Learning*, pages 7583–7615. PMLR, 2022.

[17] Litao Qiao, Weijia Wang, and Bill Lin. Learning accurate and interpretable decision rule sets from neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4303–4311, 2021.

[18] Ruixuan Yan, Yunshi Wen, Debarun Bhattacharjya, Ronny Luss, Tengfei Ma, Achille Fokoue, and Anak Agung Julius. Weighted clock logic point process. In *International Conference on Learning Research (ICLR) 2023*, 2023.

[19] Tong Zhang. *Mathematical analysis of machine learning algorithms*. Cambridge University Press, 2023.

[20] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A bayesian framework for learning rule sets for interpretable classification. *Journal of Machine Learning Research*, 18(70):1–37, 2017.

[21] Wei Zhang, Yongxiang Liu, Zhuo Wang, and Jianyong Wang. Learning to binarize continuous features for neuro-rule networks. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 4584–4592, 2023.

[22] Nuri Cingillioglu and Alessandra Russo. pix2rule: End-to-end neuro-symbolic rule learning. *arXiv preprint arXiv:2106.07487*, 2021.

[23] Zohreh Shams, Botty Dimanov, Sumaiyah Kola, Nikola Simidjievski, Helena Andres Terre, Paul Scherer, Urška Matjašec, Jean Abraham, Pietro Liò, and Mateja Jamnik. Rem: an integrative rule extraction methodology for explainable data analysis in healthcare. *medRxiv*, pages 2021–01, 2021.

[24] Mateo Espinosa Zarlenga, Zohreh Shams, and Mateja Jamnik. Efficient decompositional rule extraction for deep neural networks. *arXiv preprint arXiv:2111.12628*, 2021.

[25] Shaoyun Shi, Yuexiang Xie, Zhen Wang, Bolin Ding, Yaliang Li, and Min Zhang. Explainable neural rule learning. In *Proceedings of the ACM Web Conference 2022*, pages 3031–3041, 2022.

[26] Lu Yu, Meng Li, Xiaoguang Huang, Wei Zhu, Yanming Fang, Jun Zhou, and Longfei Li. Metarule: A meta-path guided ensemble rule set learning for explainable fraud detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4650–4654, 2022.

[27] Lu Yu, Meng Li, Ya-Lin Zhang, Longfei Li, and Jun Zhou. Finrule: Feature interactive neural rule learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 3020–3029, 2023.

[28] Zhuo Wang, Wei Zhang, Ning Liu, and Jianyong Wang. Scalable rule-based representation learning for interpretable classification. *Advances in Neural Information Processing Systems*, 34:30479–30491, 2021.

[29] Remy Kusters, Yusik Kim, Marine Collery, Christian de Sainte Marie, and Shubham Gupta. Differentiable rule induction with learned relational features. *arXiv preprint arXiv:2201.06515*, 2022.

[30] Nils Philipp Walter, Jonas Fischer, and Jilles Vreeken. Finding interpretable class-specific patterns through efficient neural search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9062–9070, 2024.

[31] Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. A brief review of hypernetworks in deep learning. *arXiv preprint arXiv:2306.06955*, 2023.

[32] Johannes Von Oswald, Christian Henning, Benjamin F Grewe, and João Sacramento. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

[33] Yizhou Huang, Kevin Xie, Homanga Bharadhwaj, and Florian Shkurti. Continual model-based reinforcement learning with hypernetworks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 799–805. IEEE, 2021.

[34] Tomer Volk, Eyal Ben-David, Ohad Amosy, Gal Chechik, and Roi Reichart. Example-based hypernetworks for out-of-distribution generalization. *arXiv preprint arXiv:2203.14276*, 2022.

[35] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

[36] Neale Ratzlaff and Li Fuxin. Hypergan: A generative model for diverse, performant neural networks. In *International Conference on Machine Learning*, pages 5361–5369. PMLR, 2019.

[37] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*, 2021.

[38] Lorenz K Muller. Overparametrization of hypernetworks at fixed flop-count enables fast neural image enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 284–293, 2021.

[39] Fulton Wang and Cynthia Rudin. Falling rule lists. In *Artificial intelligence and statistics*, pages 1013–1022. PMLR, 2015.

[40] Chaofan Chen and Cynthia Rudin. An optimization approach to learning falling rule lists. In *International conference on artificial intelligence and statistics*, pages 604–612. PMLR, 2018.

[41] Hugo M Proença and Matthijs van Leeuwen. Interpretable multiclass classification by mdl-based rule lists. *Information Sciences*, 512:1372–1393, 2020.

[42] Lucile Dierckx, Rosana Veroneze, and Siegfried Nijssen. Rl-net: Interpretable rule learning with neural networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 95–107. Springer, 2023.

[43] Hongyu Yang, Cynthia Rudin, and Margo Seltzer. Scalable bayesian rule lists. In *International conference on machine learning*, pages 3921–3930. PMLR, 2017.

[44] Fernando Jiménez, Gracia Sánchez, and José M Juárez. Multi-objective evolutionary algorithms for fuzzy classification in survival prediction. *Artificial intelligence in medicine*, 60(3):197–219, 2014.

A Compare with classical rule learning methods

We first compare HyperLogic with falling rule lists(FRL) methods like [39, 40, 41], which are often used in practice. FRL methods explicitly construct a falling list of rules with probabilities; Differentiable HyperLogic focuses on directly mining patterns and generating rules from data in a scalable manner using neural networks. Our differentiable approach demonstrates **better scalability for pattern mining tasks on large-scale data** (see Section 2). The main differences between FRL and the HyperLogic are:

1. **Rule Generation:**
   - FRL methods rely on other rule mining techniques to generate initial candidate rules.
   - HyperLogic uses a differentiable neural network approach to directly mine patterns and generate rules from data in an end-to-end manner.
2. **Scalability:**
   - FRL methods may face scalability issues when dealing with large candidate rule sets or complex data.
   - HyperLogic, a neural network-based approach, is more scalable for pattern mining in large-scale data.
3. **Rule Ordering:**
   - FRL methods explicitly order the rules into a descending list based on rule probabilities or other criteria.
   - HyperLogic generates an unordered set of rules.
4. **Probabilistic Interpretation:**
   - FRL methods provide probabilistic interpretations for the rules by construction.
   - HyperLogic does not directly output rule probabilities, although probabilities could potentially be derived from the learned patterns.
5. **Integration Potential:** A recent study [42] proposed a neural method to learn ordered rule lists, which could potentially be integrated with HyperLogic to enable joint rule generation and probabilistic ordering, combining the strengths of both approaches.

Other statistical rule mining methods like Bayesian rule lists [43] and Bayesian decision sets [20] provide probabilistic interpretations but are limited to binary classification and small rule sets. Fuzzy rule-based models [44] incorporate human-like reasoning but lack probabilistic predictions and scalability. In the supplementary experiments B, we include the current advanced traditional rule learning method CLASSY in the comparison to further demonstrate the advantages of our method over traditional methods.

B Results for HyperLogic with DIFFNAPS

To expand our experiments to larger and more complicated cases, we considered the latest Neuro-Symbolic algorithm **DIFFNAPS** [30], which is capable of pattern mining under large-scale data conditions. For **HyperLogic**, we selected only the classifier part of DIFFNAPS as the main network to compare with vanilla DIFFNAPS.

B.1 Large Synthetic Datasets

Following the original experiments, we tested the model's pattern mining performance under a **fixed input dimension of 5000** and varying total number of **categories K (ranging from 2 to 50)**, measured by the F1 score, with each category containing **1000 samples**.

Compared with the current data set, in our **new data set**, the feature dimension has been raised from a maximum of 154 dimensions to a maximum of **5k dimensions**, the amount of data has been raised from a maximum of 24,000 to a maximum of **50,000**, and the task has been raised from a maximum of 2 categories to a maximum of **50 categories**, reflecting the characteristics of the **task diversity and complexity**.

The experimental results are shown in the table below. It can be seen that in datasets with fewer categories, due to the smaller total number of samples, **HyperLogic** has not yet received sufficient training and does not perform ideally. However, in more challenging classification datasets with an increased number of samples, the model's performance has significantly improved, **with an average F1 score increase of 6%**. This fully demonstrates that our framework can empower diverse neural rule learning networks, capable of handling large-scale data and possessing a good range of applications.

Dataset (K=)	DIFFNAPS	HyperLogic
2	**0.788 ± 0.080**	0.726 ± 0.084
5	**0.703 ± 0.030**	0.675 ± 0.036
10	0.726 ± 0.013	**0.751 ± 0.020**
15	0.622 ± 0.017	**0.800 ± 0.028**
20	0.712 ± 0.015	**0.760 ± 0.011**
25	0.688 ± 0.020	**0.770 ± 0.020**
30	0.602 ± 0.014	**0.766 ± 0.021**
35	0.557 ± 0.014	**0.668 ± 0.018**
40	0.635 ± 0.022	**0.712 ± 0.011**
45	0.611 ± 0.009	**0.694 ± 0.009**
50	0.603 ± 0.010	**0.702 ± 0.012**

Table 3: The F1 score (± std) of two methods among 11 synthetic datasets

B.2 Large Real Datasets

We evaluated our method on four large biological datasets following the settings of DIFFNAPS: Cardio, Disease, BRCA-N, and BRCA-S, using the area under the curve (AUC) as the metric. We continued to combine our approach (HyperLogic) with DIFFNAPS as the main network and compared it to vanilla DIFFNAPS. Additionally, FRL [39] cannot scale to non-trivial data, while CLASSY [41] was already compared in the original paper.

The table shows the dataset details (i.e. samples (n), features (D), and classes (K)), number of discovered patterns (#P), average pattern length (|P|), and AUC scores (results for DIFFNAPS and CLASSY are taken directly from [30]).

Dataset	n	D	K	HyperLogic			DIFFNAPS			CLASSY		
				#P	$\|P\|$	AUC	#P	$\|P\|$	AUC	#P	$\|P\|$	AUC
Cardio	68k	45	2	15	2	**0.57**	14	2	0.56	10	2	0.36
Disease	5k	131	41	866	2	**0.86**	838	2	0.84	25	2	0.11
BRCA-N	222	20k	2	187	6	**0.95**	146	9	0.91	3	1	0.45
BRCA-S	187	20k	4	1k	2	**0.89**	1k	2	0.86	2	1	0.23

Table 4: Comparison of HyperLogic, DIFFNAPS, and CLASSY across 4 real datasets.

CLASSY lacks the finesse to effectively mine patterns for large-scale real-world tasks. Moreover, despite competing with the strong DIFFNAPS baseline, HyperLogic achieved further improvements, demonstrating its potential for handling large, real-world datasets.

C Hypernetwork Details

We adopted HyperGAN as our hypernetwork. Assuming the main network comprises N distinct weight partitions, HyperGAN includes a mixer Q and N generators $G_1, G_2, \ldots, G_N$.

Mixer Q: Receives high-dimensional Gaussian distributed random samples $s \sim \mathcal{N}(0, I)$, and transforms it into a $N \times h$ dimensional vector z, further split into N samples of h-dimensional vectors $z_1, z_2, \ldots, z_N$. The mixer's design reflects the necessity for correlations between layer weights, as each layer's output becomes the subsequent layer's input.

Generators G_i: Each generator receives a vector z_i from the mixer, producing an output vector of dimension m_i, where m_i represents the parameter count for the i-th part of the network weights. These vectors are then reshaped to meet the specifications of the corresponding layers in the main network.

In our specific task, the number of generators N is set to 2, and each generator's input dimension h is 64. The input dimension of the Mixer, which is the sampled noise, is 256, and it produces an output of $N \times h = 128$. Each of the above models has two hidden layers with dimension 512, and the activation function is ReLu.

Our main network is designed to generate $K = 50$ rules. For a dataset with dimension D:

- **Generator 1:** Produces an output of dimension $(D \times K, 1)$, representing the Rule layer.
- **Generator 2:** Produces an output of dimension $(K \times 1, 1)$, representing the OR layer.

These outputs are then reshaped to meet the specifications of the corresponding layers in the main network.

D Proof for Theorem 1

Proof. Using Barron's theorem (e.g., [19, Theorem 11.3]), there exist weight coefficients $(w_k^m, \tilde{u}_k^m)$, $m = 1, \ldots, M$, $k = 1, \ldots, K$, such that the function $\tilde{f}$ defined by

$$\tilde{f}(x) = \frac{1}{MK} \sum_{m=1}^{M} \sum_{k=1}^{K} \tilde{u}_k^m h\left((w_k^m)^\top x\right)$$

satisfies

$$\|f - \tilde{f}\|_{L^2} \leq \frac{\|f\|_{\mathcal{B}}}{\sqrt{MK}}.$$

Setting $u_k^m = \tilde{u}_k^m / K$, $w^m = (w_1^m, \ldots, w_K^m)$, $u^m = (u_1^m, \ldots, u_K^m)$ yields the desired result. □

E Proof for Theorem 2

E.1 Generalization Error: Diverse Regularization

Let us compute the Rademacher complexity of the model class

$$\mathcal{F}_{\mathrm{KL}} := \left\{ f_\mu(\cdot) : D_{\mathrm{KL}}(\mu \| \mu_0) \leq B_{\mathrm{KL}} \right\},$$

where μ_0 is a product distribution $\tilde{\mu}_0^{\otimes K}$. Using the property of relative entropy, for each pair of marginal distributions μ_k and $\tilde{\mu}_0$, $k = 1, \ldots, K$, their relative entropy satisfies $D_{\mathrm{KL}}(\mu_k \| \tilde{\mu}_0) \leq B_{\mathrm{KL}}$. Using the convexity of the relative entropy, the mixture distribution $\tilde{\mu}$ satisfies $D_{\mathrm{KL}}(\tilde{\mu} \| \tilde{\mu}_0) \leq B_{\mathrm{KL}}$. Hence, the function class $\mathcal{F}_{\mathrm{KL}}$ belongs to the function class

$$\tilde{\mathcal{F}}_{\mathrm{KL}} := \left\{ \tilde{f}_{\tilde{\mu}}(\cdot) : D_{\mathrm{KL}}(\tilde{\mu} \| \tilde{\mu}_0) \leq B_{\mathrm{KL}} \right\}.$$

Using [19, Corollary 10.17], the Rademacher complexity of $\tilde{\mathcal{F}}_{\mathrm{KL}}$ is bounded by $\sqrt{2B_{\mathrm{KL}}/n}$. Given our bound on the Rademacher complexity, for $f \in \tilde{\mathcal{F}}_{\mathrm{KL}}$, we get:

$$\mathbb{E}_{\mathcal{D}}[\ell(f)] - \hat{\ell}(f) \leq 2\sqrt{\frac{2B_{\mathrm{KL}}}{n}} + C\sqrt{\frac{\log(1/\delta)}{2n}}. \tag{11}$$

where $\mathbb{E}_{\mathcal{D}}[\ell(f)]$ is the expected loss, $\hat{\ell}(f)$ is the empirical loss, n is the number of samples, C is a constant dependent on the loss function, and δ is the confidence level.

E.2 Generalization Error: Sparse Regularization

Next, we will derive the generalization error bounds for the HyperLogic model under sparse regularization (as shown in the second term of Eq. (6)).

Let us compute the Rademacher complexity of the model class

$$\mathcal{F}_1 := \left\{ \tilde{f}_\mu(\cdot) : \mathbb{E}_\mu\Big[\frac{1}{K} \sum_{k=1}^{K} |u_k|\Big] \leq B_1 \right\}.$$

Observe that the constraint $\mathbb{E}_\mu\Big[\frac{1}{K}\sum_{k=1}^{K} |u_k|\Big] \leq B_1$ implies that

$$\mathbb{E}_{\tilde{\mu}}\left[|\tilde{u}|\right] = \mathbb{E}_\mu\left[\left|\frac{1}{K}\sum_{k=1}^{K} u_k\right|\right] \leq \mathbb{E}_\mu\left[\frac{1}{K}\sum_{k=1}^{K} |u_k|\right] \leq B_1.$$

Hence, the function class $\mathcal{F}$ belongs to the function class

$$\tilde{\mathcal{F}}_1 := \left\{ \tilde{f}_{\tilde{\mu}}(\cdot) : \mathbb{E}_{\tilde{\mu}}\left[|\tilde{u}|\right] \leq B_1 \right\}.$$

Using [19, Proposition 11.23], the Rademacher complexity of $\tilde{\mathcal{F}}_1$ is bounded by $32B_1\sqrt{(D+1)/n}$. The generalization error bound for the HyperLogic model under sparse regularization, derived from the Rademacher complexity, can be expressed as follows:

$$\mathbb{E}_{\mathcal{D}}[\ell(f)] - \hat{\ell}(f) \leq 64B_1\sqrt{(D+1)/n} + C\sqrt{\frac{\log(1/\delta)}{n}}$$

The explaination of this error bound is similar with the previous one.

F Supplementary Result

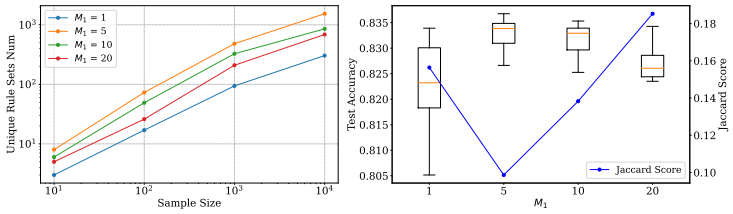

Figure 8: Analysis of the impact of M_1 on adult dataset

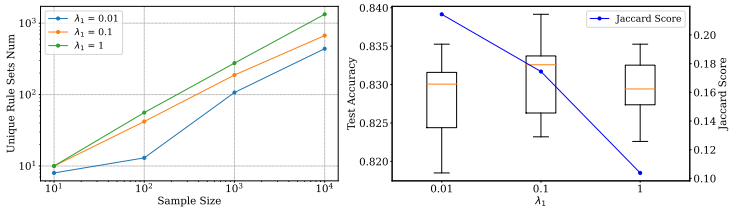

Figure 9: Analysis of the impact of λ_1 on adult dataset

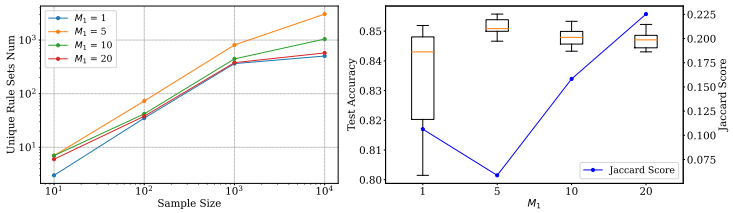

Figure 10: Analysis of the impact of M_1 on house dataset

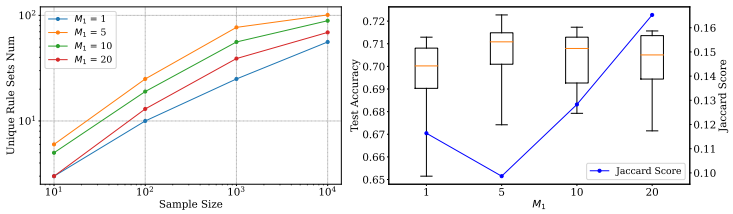

Figure 12: Analysis of the impact of M_1 on heloc dataset

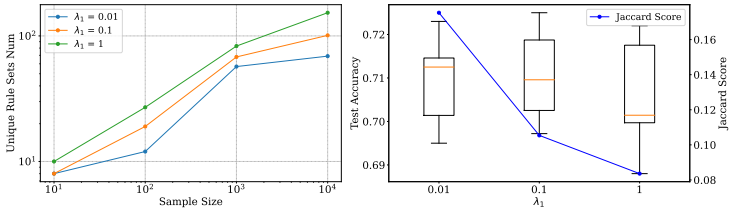

Figure 13: Analysis of the impact of λ_1 on heloc dataset

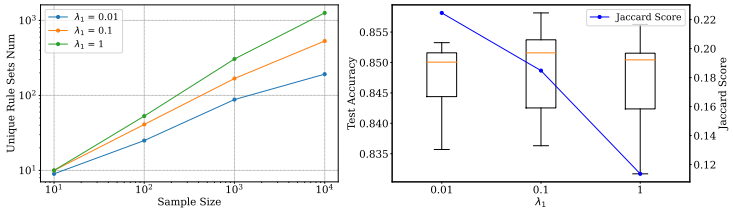

Figure 11: Analysis of the impact of λ_1 on house dataset

G Computing Infrastructure

For our method, all experiments were conducted on a Linux server with an Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz and 30Gi of memory, running Ubuntu 20.04.5 LTS, using one of the NVIDIA GeForce RTX 3090 GPUs available on the server. Each experimental run took approximately 10-20 minutes to complete. This setup ensures that our experiments are reproducible and efficient.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We demonstrate our innovations and contributions in the abstract and introduction. We also mention the important assumptions and limitations of our methods.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of our proposed model in the conclusion part.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

   Justification: We provide theoretical proofs about our proposed model in the theoretical analysis section.

   Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We provide the details of hyper parameters and neural network structures for reproducibility.

   Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The code will be made publicly available upon the paper's acceptance.

   Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: We provide all the training and test details including hyperparameters, type of optimizer, learning rates, hidden layers of networks in the appendix.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: Our rule performance results mean accuracy with the standard error across a nested 5-fold cross-validation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).

- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We demonstrate the computer resource requirements in the appendix.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The research presented in our paper fully complies with the NeurIPS Code of Ethics in all aspects.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: We discuss potential positiveand negative societal impacts of the work in the appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: Our paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the original paper that produced the dataset.

    Guidelines:

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

13. **New Assets**

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

    Answer: [NA]

    Justification: Our paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

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

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: Our paper does not involve study participants.

    Guidelines:

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