Self-explaining deep models with logic rule reasoning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present a framework for integrating self-explaining capabilities into a given deep model to achieve high prediction performance and human precision. Human precision means that most model decisions are coherent with human decision logic, allowing users to quickly understand and identify a small fraction of problematic model behavior. We propose two desirable properties for ensuring high human precision and demonstrate that logic rule explanations naturally satisfy them, while also possessing the expressive power required for good predictive performance. Then, using logic rules, we propose a framework for a deep model to predict and explain. Our method does not require predefined logic rule sets, and can be learned in a differentiable way with widely-used deep learning modules. Extensive experiments show that our method achieves high prediction performance and human precision, as well as being naturally robust to noisy labels.

1 Introduction

Deep learning has shown high predictive accuracy in a wide range of tasks, but its inner working mechanisms are obscured by complex model designs. This raises important questions about whether a deep model is ethical, trustworthy, or capable of performing as intended under various conditions [1].

Many approaches have been proposed to help humans assess and comprehend model decisions. Recent work on explainability has primarily focused on providing *post-hoc* explanations for a black-box model that has already been trained [2, 3, 4, 5, 6, 7, 8, 9, 10]. Post-hoc methods do not change the model and hence preserve the predictive performance while providing the additional benefit of explainability. These methods have achieved considerable success in providing valuable insights for model understanding, but there are also known challenges such as computational cost [11] and trust issues [12]. For example, many popular post-hoc methods test the complex black-box model thousands of times to obtain a complete and faithful understanding of the model around a single instance [1, 13, 14]. Subroutines such as full optimization or reverse propagation are generally required, introducing approximations or heuristic assumptions that may lead to misinterpretation [14, 15]. Because there is no guarantee that explanations are always faithful to the model [12], there is a "general uneasiness" among practitioners about using and trusting post-hoc explanations [16],

Self-explaining models naturally solve these issues, making them an ideal choice in situations where interpretability can be considered from the model design phase [17, 18, 19, 20]. Because the explanation mechanism is integrated inherently, these models can predict and explain simultaneously with a single forward propagation without any approximations or heuristic assumptions that decrease the faithfulness of explanations. Self-explaining methods may also improve robustness [11] and provide actionable insights for directly refining model parameters without having to calibrate the dataset [21, 22].

Based on these observations, we regard self-explaining models as providing a stronger link between humans and machine learning models, reducing misunderstanding and allowing direct control of the model based on human insights. The main challenge in achieving this goal is ensuring the self-

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

(a) Rule example (b) Human precision (c) Our generative process

Figure 1: Reasoning with logic rules: (a) examples of logic rule explanations; (b) human precision for logic rule and linear regression explanations [11]; (c) generative process of our logic rule reasoning.

explaining model's precision, both in terms of predictive performance and *human precision*. Human precision means most model decisions are coherent with human decision logic such that humans can quickly understand the explanations and identify problematic model behavior. Existing approaches focus on generating explanations that are *easy to read*, such as requiring explanations to be simple and smooth in a local area [11]. However, there is little guarantee that a given explanation is a *correct* rationale for prediction according to human perception. For example, the explanation "*awesome*$\geq$2" (the word *awesome* appears twice in reviews) is a good rationale for positive sentiment, while "*is*$\geq$1 $\Rightarrow$ positive sentiment" is easy to read but unreasonable to humans. Without an insurance for human precision, users may constantly find such unreasonable explanations, which can significantly hamper user trust and prevent them from identifying actionable insights. An interesting research question then is: how can self-explaining models generate explanations that are consistent with human decision logic?

To answer this question, we need to carefully decide what information models obtain from humans. Collecting ground-truth labels of human decision processes are quite expensive, if not impossible. Moreover, forcing the model to make decisions exactly like humans may be unwise, since it could limit its capability of learning from data, and potential bias from humans may also lead to significant decrease in model prediction. To address this issue, it is important that the humans provide guidance at a higher level that allows the models to learn freely based on the data. Accordingly, we propose two desirable properties for human precision. The first property, *global coherency*, requires that the form of the explanations be consistent with human reasoning logic, thereby minimizing the probability of misinterpretation. The second property, *local coherency*, requires that each explanation naturally lead to the prediction according to human perception, thereby making explanations a correct rationale for the model output. As humans can hardly provide guidance for each explanation, it is more desirable that the models can automatically guarantee local coherency, based on human guidance on the desirable forms of explanations (global coherency).

A key to satisfying these two properties is logic rules. As shown in Fig. 1a, a logic rule consists of 1) an *antecedent*, which is the explanation or condition for applying the rule and 2) a *consequent*, which is the model prediction for the given antecedent. The antecedent form, including both the basic explanation unit (called *atom*) and logical connectives, is flexibly defined based on human logic and preferences, making it easy to satisfy global coherency. For example, the logical connectives can be traditional ones (e.g., AND, OR, and NOT) or self-defined ones (e.g., BEFORE). Moreover, the logic rules explicitly model whether an explanation (i.e., antecedent) can lead to a prediction (i.e., consequent) by testing the hypothesis across the entire dataset. This ensures a meaningful relationship between explanations and predictions that naturally leads to local coherency. As shown in Fig. 1b, logic rule explanations achieve higher human precision than linear-regression-based explanations with local stability, while providing a confidence score that correlates with human precision (see supplement for more details). Lastly, logic rules with different types of atoms and logical connectives correspond to a diverse set of feature interactions, providing the expressive power for good predictive performance.

Based on the theoretical grounding, we propose a logic rule reasoning framework for high-precision self-explaining deep models. Unlike existing symbolic reasoning methods that handle the large solution space by leveraging reinforcement learning or external dataset [23, 24, 25, 26, 27, 28], we generate the possible combinations of atoms by training a separate deep model that acts as a consequent estimator and combining it with recursive Gumbel-Softmax [29] to search the solution space effectively. Our main contributions are as follows.

- Our work suggests that human precision is key for self-explaining models to bridge human and model decision logic seamlessly. Logic rule-based explanations enable high human precision while allowing the expressive power to achieve high model precision.

- We propose a novel framework for reasoning based on deep logic rules. Our framework does not depend on external data (such as predefined rule sets) and allows self-explaining models to be trained using widely used deep learning modules.
- Numerical experiments and user studies confirm several strengths of our framework in terms of prediction performance, human precision, and robustness against noisy labels.

2 Deep Logic Rule Reasoning

2.1 Formulation of Logic Rules

A logic rule $\boldsymbol{\alpha} \Rightarrow y$, as shown in Fig. 1a, consists of an antecedent $\boldsymbol{\alpha}$ and a consequent y.

- An **antecedent** $\boldsymbol{\alpha}$ is the condition to apply the rule and corresponds to an explanation in a logic form. It is represented as a sequence $\boldsymbol{\alpha} = (o_1 ..., o_L)$, where o_i is either an atom or a logical connective.
  - An **atom** is the smallest unit of explanation that corresponds to a single *interpretable feature* of a given input (e.g., "awesome$\geq$2"). The interpretable features may be different from the deep learning model's features. They could, for example, have a different granularity (e.g., words or phrases) than the model's features (e.g., partial words), be a statistical feature (e.g., word frequency), or be derived using external tools (e.g., grammatical tagging of a word). Mathematically, each atom o_i is a Boolean-value function, with $o_i(\mathbf{x})$ returning true if the i-th interpretable feature is present in the input $\mathbf{x}$ and false otherwise.
  - A **logical connective** combines the atoms to form an explanation. Logical connectives can be traditional ones like AND, OR, NOT, or self-defined ones, as long as they take one or multiple Boolean values as the input and output a single Boolean value.

  We say that an input sample $\mathbf{x}$ **satisfies** an antecedent $\boldsymbol{\alpha}$, if $\boldsymbol{\alpha}(\mathbf{x})$ is true.
- The **consequent** y is the model's prediction output given the antecedent. For example, y is the predicted class in a classification task, whereas y is an explicit number in a regression task. We mainly consider classification in the paper and extend the cases to regression in the supplement.

2.2 Framework for Deep Logic Rule Reasoning

Let us denote f as a deep learning model that estimates probability $p(y|\mathbf{x})$, where $\mathbf{x}$ is the input data sample and y is the class label. We upgrade model f to a self-explaining version by adding a latent variable $\boldsymbol{\alpha}$, which is an explanation in the logic form. We learn $\boldsymbol{\alpha}$ and y together by maximizing

$$p(y|\mathbf{x}, b) = \sum_{\boldsymbol{\alpha}} p(y|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}|\mathbf{x}, b) \propto \sum_{\boldsymbol{\alpha}} \underbrace{p(\boldsymbol{\alpha}|b)}_{\substack{\text{Human}\\\text{prior}}} \cdot \underbrace{p(y|\boldsymbol{\alpha})}_{\substack{\text{Consequent}\\\text{estimation}}} \cdot \underbrace{p(\boldsymbol{\alpha}|\mathbf{x})}_{\substack{\text{Deep antecedent}\\\text{generation}}} \tag{1}$$

where b represents a human's prior belief about the rules, e.g., the form of logical connectives. The proportional relationship $\propto$ is derived based on the generative process in Fig. 1c (see proof in the supplement), and the three terms on the right correspond to three major modules in our framework:

- **Human prior** $p(\boldsymbol{\alpha}|b)$ specifies possible forms of atoms and logical connectives and whether humans prefer shorter rules or those that cover more data points.
- **Consequent estimation** $p(y|\boldsymbol{\alpha})$ computes the probability of output y given an explanation $\boldsymbol{\alpha}$ by testing the logic rule $\boldsymbol{\alpha} \Rightarrow y$ over the entire dataset. This module ensures a meaningful and consistent relationship between the explanation and prediction, leading to high human precision.
- **Deep antecedent generation** $p(\boldsymbol{\alpha}|\mathbf{x})$ finds the best combination of atoms and composes them into an antecedent $\boldsymbol{\alpha}$ based on the input $\mathbf{x}$. The deep model f is used to encode the sample $\mathbf{x}$ so that the input is accurately and discriminately represented. An additional recursive generation layer is added to f so that the deep model generates explanations in a logic form instead of predicting y.

Explanations that lead to the correct prediction and fit human's prior beliefs will have a higher probability of being generated if Eq. (1) is maximized. Our framework for deep logic rule reasoning bears an interesting resemblance to the human mind, in which a powerful subconscious part lies underneath the critical-thought layers of the conscious mind [30]. The deep learning layers for encoding the input $\mathbf{x}$ in our framework are analogous to the subconscious mind, which is difficult to comprehend but often thought to be more powerful. The upper explainable layers, on the other hand, are analogous to the conscious mind responsible for critical thinking and logical rationality.

2.3 Human Prior $p(\boldsymbol{\alpha}|b)$

Human prior $p(\boldsymbol{\alpha}|b) = p_h(\boldsymbol{\alpha}|b)p_s(\boldsymbol{\alpha}|b)$ consists of hard priors $p_h(\boldsymbol{\alpha}|b)$ and soft ones $p_s(\boldsymbol{\alpha}|b)$.

Hard priors categorize the feasible solution space for the rules: $p_h(\boldsymbol{\alpha}|b) = 0$ if $\boldsymbol{\alpha}$ is not a feasible solution. Humans can easily define hard priors by choosing the atom types, such as whether the interpretable features are words, phrases, or statistics like word frequency. The logical connectives to be considered (e.g., AND or NOT) can also be chosen, as well as the antecedent's maximum length L. Our framework does not require a predefined rule set. Nonetheless, we allow users to enter one if it is more desirable in some application scenarios. A large solution space increases the time cost for deep logic rule reasoning (Sec. 2.6) but also decreases the probability of introducing undesirable bias.

Soft priors can be used to model human preferences for logic rules. For example, people may prefer shorter rules to improve readability or high-coverage rules that are satisfied by many input samples. The energy function can parameterize such soft priors: $p_s(\boldsymbol{\alpha}|b) \propto \exp(-\mathcal{L}_b(\boldsymbol{\alpha}))$, where $\mathcal{L}_b$ is the loss function for punishing undesirable logic rules. Our experiment results show that reasonably good explainability and predictive performance can be achieved without any soft priors (Sec. 3).

2.4 Consequent Estimation $p(y|\boldsymbol{\alpha})$

Consequent estimation computes the probability $p(y|\boldsymbol{\alpha})$ of a consequent y give explanation $\boldsymbol{\alpha}$. Unlike other self-explanatory methods that predict based on a local neighborhood around sample $\mathbf{x}$, consequent estimation predicts at a global level by testing the logic rule $\boldsymbol{\alpha} \Rightarrow y$ across the entire training dataset. This ensures a meaningful and consistent relation between explanations and predictions. For example, in our framework, explanation "awesome≥2 AND tasty≥1" always leads to the prediction "positive sentiment" with a large probability, since training samples that mention "awesome" more than twice and "tasty" more than once are usually labeled as positive.

Empirical estimation. A straightforward way to compute $p(y|\boldsymbol{\alpha})$ is to first obtain all samples that satisfy antecedent $\boldsymbol{\alpha}$, and then calculate the percentage of them that have label y [8]. Mathematically, this is equivalent to approximating $p(y|\boldsymbol{\alpha})$ with the empirical probability $\hat{p}(y|\boldsymbol{\alpha})$:

$$\hat{p}(y|\boldsymbol{\alpha}) = \frac{n_{\boldsymbol{\alpha},y}}{n_{\boldsymbol{\alpha}}} = \frac{\sum_{(\mathbf{x}',y')} \mathbb{I}(\boldsymbol{\alpha}(\mathbf{x}'))\mathbb{I}(y'=y)}{\sum_{\mathbf{x}'} \mathbb{I}(\boldsymbol{\alpha}(\mathbf{x}'))} \tag{2}$$

where $n_{\boldsymbol{\alpha}}$ is the number of training samples that satisfy the antecedent $\boldsymbol{\alpha}$, $n_{\boldsymbol{\alpha},y}$ is the number of training samples that satisfy the antecedent $\boldsymbol{\alpha}$ and has the consequent y, and $\mathbb{I}$ is an indicator function that converts false to 0 and true to 1.

Directly setting $p(y|\boldsymbol{\alpha})$ to $\hat{p}(y|\boldsymbol{\alpha})$ can cause two problems. First, when $n_{\boldsymbol{\alpha}}$ is not large enough, the empirical probability $\hat{p}(y|\boldsymbol{\alpha})$ may be inaccurate, and the modeling of such uncertainty is inherently missing in this formulation. Second, computing $\hat{p}(y|\boldsymbol{\alpha})$ for every antecedent $\boldsymbol{\alpha}$ is intractable, since the number of feasible antecedents A increases exponentially with antecedent length L.

Neural estimation of categorical distribution. To address the aforementioned problems, we jointly model $\hat{p}(y|\boldsymbol{\alpha})$ and the uncertainty caused by low-coverage antecedents with the categorical distribution, and use a neural network to generalize to similar rules and better handle noise.

Assume that given the antecedent, y follows a categorical distribution, with each category corresponding to a class. Then, according to the posterior predictive distribution, y takes one of K potential classes, and we may compute the probability of a new observation y given existing observations:

$$p(y|\boldsymbol{\alpha}) = p(y|\mathcal{Y}_{\boldsymbol{\alpha}}, \beta) = \frac{\hat{p}(y|\boldsymbol{\alpha})n_{\boldsymbol{\alpha}} + \beta}{n_{\boldsymbol{\alpha}} + K\beta} \tag{3}$$

Here, $\mathcal{Y}_{\boldsymbol{\alpha}}$ denotes the $n_{\boldsymbol{\alpha}}$ observations of class label y obtained by checking the training data, and β is the concentration hyperparameter of the categorical distribution that we automatically learn with backpropagation. Eq. (3) becomes Eq. (2) when $n_{\boldsymbol{\alpha}}$ increases to ∞, and becomes a uniform distribution when $n_{\boldsymbol{\alpha}}$ goes to 0. Thus, a low-coverage antecedent with a small $n_{\boldsymbol{\alpha}}$ is automatically considered uncertain (close to uniform distribution). By optimizing Eq. (3), our method chooses the antecedent by balancing the empirical probability $\hat{p}(y|\boldsymbol{\alpha})$ and the number of observations $n_{\boldsymbol{\alpha}}$. Probability $p(y|\alpha)$ also serves as a confidence score for the logic rule $\alpha \Rightarrow y$.

We then employ a neural model to predict $\hat{p}(y|\boldsymbol{\alpha})$ and $n_{\boldsymbol{\alpha}}$ to better manage noise, generalize to similar rules, and improve efficiency. In particular, we obtain A' samples of $\boldsymbol{\alpha}$ and compute $\hat{p}(y|\boldsymbol{\alpha})$

and $n_{\boldsymbol{\alpha}}$ by checking the training data. Here A' is significantly smaller than the total number of feasible antecedents A (Sec. 2.6). Then, we use the multi-task learning framework in [31] to train the neural network with these samples. In particular, we minimize $\mathcal{L}_c = \frac{1}{2\sigma_p^2}||\hat{p}(y|\boldsymbol{\alpha}) - \tilde{p}(y|\boldsymbol{\alpha})||^2 + \frac{1}{2\sigma_n^2}||n_{\boldsymbol{\alpha}} - \tilde{n}_{\boldsymbol{\alpha}}||^2 + \log \sigma_p \sigma_n$, where σ_p and σ_n for balancing the two tasks are automatically learned, and $\tilde{p}(y|\boldsymbol{\alpha})$, $\tilde{n}_{\boldsymbol{\alpha}}$ are the predicted empirical probability and coverage given by the neural model.

The input of the neural model is the antecedent embedding, which is obtained by $[\mathbf{o}_1 ...; \mathbf{o}_L]$, where $\mathbf{o}_i$ is the embedding of o_i in $\boldsymbol{\alpha} = (o_1 ..., o_L)$. For each atom, $\mathbf{o}_i$ is initialized as the average embedding of all training samples that satisfy the atom, where the sample embedding can be derived using a pretrained model or f. The embeddings of logical connectives are initialized at random, and can be omitted when there is only one logical connective (e.g., AND). We use the Transformer encoder [32] as the backbone neural network to emphasize the contextual interaction between atoms and logical connectives. More details for training the neural network are described in the supplement.

2.5 Deep Antecedent Generation $p(\boldsymbol{\alpha}|\mathbf{x})$

$p(\boldsymbol{\alpha}|\mathbf{x})$ identifies the most important interpretable features in $\mathbf{x}$ and organize them into a logic form $\boldsymbol{\alpha}$. Computing $p(\boldsymbol{\alpha}|\mathbf{x})$ for every $\boldsymbol{\alpha} = (o_1 ..., o_L)$ is usually intractable since the number of feasible antecedents grows exponentially with L. Thus, we generate each element o_i in $\boldsymbol{\alpha}$ with a recursive formulation, which reduces the complexity to be linear with L (Sec. 2.6). Formally,

$$p(\boldsymbol{\alpha}|\mathbf{x}) = \prod_{i \in [1,L]} p(o_i|\mathbf{x}, o_1 ..., o_{i-1}), \quad p(o_i|\mathbf{x}, o_1 ..., o_{i-1}) = \frac{\mathbb{I}(o_i \in \mathcal{C}_i)\exp(\mathbf{h}_i^T \mathbf{o}_i)}{\sum_{\alpha_i'} \mathbb{I}(\alpha_i' \in \mathcal{C}_i)\exp(\mathbf{h}_i^T \boldsymbol{\alpha}_i')}, \quad (4)$$

Here, $\mathcal{C}_i$ is the set of candidates for o_i. Every candidate should satisfy both global and local constraints. The hard priors provide the global constraint in Sec. 2.3. It ensures that $\boldsymbol{\alpha}$ has a human-defined valid logic form. The local constraint requires that $\mathbf{x}$ satisfies the sampled antecedent $\boldsymbol{\alpha}$. An atom "awesome$\geq$2", for example, will be sampled only if $\mathbf{x}$ mentions awesome more than twice.

$\mathbf{o}_i$ is the embedding of o_i, which is the same as in the neural consequent estimator, and $\mathbf{h}_i$ is a hidden state that may be modeled with any neural sequence encoder such as GRU [33] or Transformer:

$$\mathbf{h}_i = Encoder([\mathbf{f}; \mathbf{o}_1 ...; \mathbf{o}_{i-1}]), \qquad o_i = Gumbel(p(o_i' \in \mathcal{O}|\mathbf{x}, o_1 ..., o_{i-1})) \quad (5)$$

where $\mathbf{f}$ is the latent representation of $\mathbf{x}$ learned by deep model f, and $Gumbel$ is the Straight-Through Gumbel-Softmax [29], a differentiable function for sampling discrete values $o_i' \sim p(o_i'|\mathbf{x}, o_1 ..., o_{i-1})$. $Gumbel$ produces a sampled o_i that is represented as a one-hot vector with a dimension of $|\mathcal{O}|$, which is then multiplied with the embedding matrix of atoms and logical connectives to derive an embedding $\mathbf{o}_i$. We add a special atom NULL, which will be chosen when the model prefers to generate an explanation shorter than L.

2.6 Optimization and Complexity Analysis

Optimization. A deep logic rule reasoning model is learned in two steps. The first step optimizes the neural consequent estimator by minimizing the loss $\mathcal{L}_c$ (Sec. 2.4). The neural consequent estimator only needs to be trained once for each dataset, and then it can be used for a variety of deep models and hyperparameters. The second step converts the deep model f to an explainable version by maximizing $p(y|\mathbf{x}, b)$ in Eq. (1). Because the number of $\boldsymbol{\alpha}$ is exponentially large, directly maximizing Eq. (1) is expensive. Thus, we approximate Eq. (1) by sampling $\boldsymbol{\alpha}$:

$$p(y|\mathbf{x}, b) \propto \sum_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|b)\, p(y|\boldsymbol{\alpha})\, p(\boldsymbol{\alpha}|\mathbf{x}) \approx \frac{1}{S} \sum_{\substack{s \in [1,S] \\ \boldsymbol{\alpha}^{(s)} \sim p(\boldsymbol{\alpha}|\mathbf{x})}} p(\boldsymbol{\alpha}^{(s)}|b)\, p(y|\boldsymbol{\alpha}^{(s)}) \quad (6)$$

where $\boldsymbol{\alpha}^{(s)}$ is the s-th sample of $\boldsymbol{\alpha}$, and S is the number of samples. In practice, we sample one antecedent for an input at each epoch ($S = 1$), and maximizing $\log\ p(y|\mathbf{x}, b)$ by minimizing loss $\mathcal{L}_d = -\log\ p(y|\boldsymbol{\alpha}^{(s)}) - \mathcal{L}_b(\boldsymbol{\alpha}^{(s)})$. Accordingly, the model learns to find $\boldsymbol{\alpha}^{(s)}$ that results in the maximum prediction performance $\log p(y|\boldsymbol{\alpha}^{(s)})$ and fits human prior $\mathcal{L}_b(\boldsymbol{\alpha}^{(s)})$. We sample the discrete value $\boldsymbol{\alpha}^{(s)}$ in a differentiable way by applying Gumbel-Softmax recursively as introduced in Sec. 2.5.

Complexity analysis. Time complexity is compared in Table 1. The complexity for antecedent generation corresponds to the time added for generating the antecedents during model training and the time required for training the base deep model f. Here, N is the number of training samples, and C is the time complexity for computing the consequent of each antecedent. As shown in the table, removing the recursive antecedent generator (RE) or the neural consequent estimator (NE) brings an additional linear complexity with the number of feasible antecedents A, which is much larger than A'. For example, in our experiment, setting A' to 10^4 is good enough to train an accurate neural consequent estimator, while the number of all possible antecedents is $A = 6.25 \times 10^{12}$.

Table 1: Time complexity analysis. -RG and -NE denote our method without recursive antecedent generation and neural consequent estimator.

	Consequent estimation	Antecedent generation
Ours	$O(A'C + A'L^2)$	$O(N\lvert\mathcal{O}\rvert L + NL^2)$
-RG	$O(A'C + A'L^2)$	$O(NA + NL^2)$
-RG-NE	$O(AC)$	$O(NA)$

3 Experiment

3.1 Experimental settings.

Datasets. We conduct experiments on three datasets. The first two are textual, and the third is tabular. **Yelp** classifies reviews of local businesses into positive or negative sentiment [34], and **Clickbait News Detection** from Kaggle labels whether a news article is a clickbait [35]. **Adult** from the UCI machine learning repository [36], is an imbalanced tabular dataset that provides labels about whether the annual income of an adult is more than $50K/yr or not. For Yelp, we use a down-sampled subset (10%) for training, as per existing work [23]. More details are in the supplement.

Baselines. We compare our model to four baselines. Two self-explainable models, **SENN** [11] and **RCN** [23], are compared in accuracy, robustness, explainability, and efficiency. Two post-hoc explainable methods, **LIME** [1]and **ANCHOR** [14], are compared in explainability and efficiency.

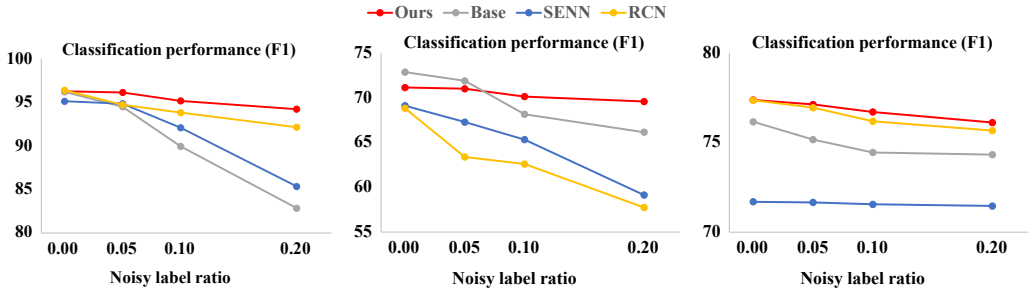
Implementation details. To match with baselines, we use the AND operation by default in explanations. The impact of using other logical connectives are presented in the supplement. The atoms, or interpretable features, are the same as in the majority of baselines, i.e., words for the textual dataset (e.g., "amazing") and categorical features for the tabular data (e.g., "age<28"). Only the top 5,000 words are kept in the textual data, and the experiments are run on NVIDIA Tesla V100.

3.2 Classification Performance and Robustness

Classification performance. Table 2 shows the classification performance of our method and baselines. BERT and RoBERTa are used as the backbone networks in the NLP datasets, while 3-Layer DNN is used for the tabular dataset. The base method is the vanilla backbone network, which does not have any explainability. Note that the prediction performance of post-hoc methods, LIME and Anchor, is the same to the base model as they utilize trained model without any extra optimization. Our method achieves comparable average performance with the unexplainable base model and outperforms other self-explaining models by 1.9%. Moreover, our method consistently achieves the best or comparable results on various datasets or backbone models, demonstrating the expressive power of logic rules for high prediction performance. RCN, on the other hand, cannot perform as well

Table 2: Comparison of classification performance (F1). The average results from five runs are shown. The best results among self-explaining models are marked in **bold**, and the highlighted cells indicate a similar or better result compared with the unexplainable backbone (Base). The numbers in subscript indicates the difference compared to the base model.

	Yelp		Clickbait		Adult	Average
	BERT	RoBERTa	BERT	RoBERTa	DNN	
Base	96.20	97.16	72.84	74.25	76.15	83.32
SENN	$95.12_{-1.12\%}$	$96.07_{-1.12\%}$	$69.09_{-5.15\%}$	$70.99_{-4.39\%}$	$71.69_{-5.86\%}$	$80.59_{-3.28\%}$
RCN	$\mathbf{96.38}_{+0.19\%}$	$\mathbf{97.36}_{+0.21\%}$	$68.80_{-5.55\%}$	$68.64_{-7.56\%}$	$\mathbf{77.35}_{+1.58\%}$	$81.77_{-1.86\%}$
Ours	$\mathbf{96.26}_{-0.06\%}$	$\mathbf{97.13}_{-0.03\%}$	$\mathbf{71.12}_{-2.36\%}$	$\mathbf{74.20}_{-0.07\%}$	$\mathbf{77.37}_{+1.60\%}$	$\mathbf{83.34}_{+0.02\%}$



(a) Yelp (b) Clickbait (c) Adult

Figure 2: Experimental results on robustness under different ratios of noisy labels.

on challenging textual datasets like Clickbait because it computes soft attention over a predefined rule set. This indicates that (potentially biased) predefined rule sets will limit the modeling capability.

Robustness to noisy labels. Following the literature [37, 38], we assess the robustness of the proposed model against randomly corrupted labels. We hypothesize that the effect of the noisy label can be alleviated by our consequent estimation term $p(y|\boldsymbol{\alpha})$, where model decisions are verified by testing the logic rule over the entire dataset. For experiments, symmetric noise is introduced by randomly flipping the labels for a subset of the training data. Fig. 2 shows the performance results over Yelp, Clickbait, and Adult datasets with multiple levels of noise ratio from 5% to 20%. Our model outperforms other models across every dataset and every noise ratio. The improvement is substantial even with a high noise ratio (i.e., 20%). For a noise ratio above 10%, our method consistently performs better than the unexplainable base models (2.4% to 13.7%).

3.3 Explainability

User study on human precision. To evaluate human precision, we recruited three native English speakers through a vendor company [39]. Each participant was provided with 50 randomly selected Yelp reviews and 50 randomly selected Adult samples. Five explanations obtained from different methods were provided for each sample, and the participants reviewed whether explanations were reasonable rationales. Participants provided two labels to each explanation, indicating whether it was good and if it was the best one. In general, a **good** explanation should naturally lead to the prediction, but it can contain noisy features. For example, "amazing, are" is a good explanation for positive sentiment. The **best** explanation is the one that contains the most important features and least noisy features. The participants are allowed to choose multiple best explanations only if the chosen ones are the same. To conduct a fair comparison, we show all explanations in the same form: a list of features that each method considers important for prediction. Example explanations generated by our method and the baselines are given in Fig. 3. Note that ˜ in RCN means negation (NOT). More details about explanation generation and labeling guidelines are given in the supplement.

Table 3 shows that our method marks the highest percentage of good explanations, with an average ratio of 90.0% on Yelp and 87.3% on Adult. Ours is also frequently chosen as the best. Although logic rules are promising, choosing from a small set of predefined rules with attention may not be good enough (RCN) due to the potential bias in the rule set. For example, RCN uses rules extracted with traditional machine learning methods that meet the global data distribution but frequently fails to adequately represent each sample, particularly on datasets with many features (Yelp). As a result, RCN is rarely chosen as the best explanation, especially for Yelp text data. Post-hoc methods also tend to offer good human precision. The best ratio of LIME and Anchor, however, is substantially

...The staff was amazing with him and made him feel so comfortable that he actually sat through an entire cleaning without being upset once...The dentist and the assistants were made to work with children and that made it all very relaxing and enjoyable for him.

LIME	amazing, comfortable, enjoyable, more
Anchor	amazing, so
SENN	about
RCN	amazing, ~went, ~seeing, ~did
Ours	amazing, relaxing, comfortable, enjoyable

Adult		LIME	Anchor	SENN	RCN	Ours
age	*27*				<32.5	<28
education	*HS-grad*	✓				
educational-num	*9*				<12.5	<10
marital-status	*Separated*		✓	✓	✓	
occupation	*Craft-repair*			✓		✓
relationship	*Own-child*	✓			✓	
race	*White*					
capital-gain	*0*	✓		✓		
capital-loss	*0*	✓		✓		

Figure 3: Example explanations produced by five methods on Yelp (left) and Adult (right).

Table 3: User study results on human precision. Participants P1-P3 are asked to annotate whether an explanation is a reasonable rationale for the prediction. For each compared method, we show the percentage of explanations that are considered good (a, b) or best (c, d). Avg. and Agr. denote the average and inter-participant agreement, respectively. Best results are highlighted in **bold**.

(a) Percentage of good (Yelp)

	Lime	Anchor	SENN	RCN	Ours
P1	88.0	86.0	26.0	70.0	**90.0**
P2	82.0	74.0	22.0	32.0	**84.0**
P3	**96.0**	92.0	18.0	6.0	**96.0**
Avg.	88.7	84.0	22.0	36.0	**90.0**
Agr.	86.7	85.3	90.7	54.7	89.3

(b) Percentage of good (Adult)

	Lime	Anchor	SENN	RCN	Ours
P1	**88.0**	30.0	**88.0**	78.0	84.0
P2	24.0	38.0	16.0	70.0	**88.0**
P3	88.0	32.0	**90.0**	86.0	**90.0**
Avg.	66.7	33.3	64.7	78.0	**87.3**
Agr.	50.7	84.0	50.7	78.7	89.3

(c) Percentage of best (Yelp)

	Lime	Anchor	SENN	RCN	Ours
P1	30.0	24.0	2.0	2.0	**44.0**
P2	36.0	20.0	4.0	2.0	**40.0**
P3	24.0	**36.0**	4.0	2.0	**36.0**
Avg.	30.0	26.7	3.3	2.0	**40.0**
Agr.	84.0	84.0	98.7	100.0	86.7

(d) Percentage of best (Adult)

	Lime	Anchor	SENN	RCN	Ours
P1	0.0	22.0	6.0	10.0	**62.0**
P2	2.0	18.0	10.0	10.0	**60.0**
P3	0.0	20.0	8.0	8.0	**64.0**
Avg.	0.7	20.0	8.0	9.3	**62.0**
Agr.	98.7	92.0	96.0	85.3	76.0

lower than ours, indicating that the base model may rely on more noisy features for prediction. In contrast, our model can verify its decision by testing the logic rule across the entire dataset.

Case study on model debugging and refinement. What useful insights can our method provide for improving model performance? In a study of 20,000 sampled Yelp reviews, we clustered the generated explanations into 10 clusters by applying K-Means on the antecedent embeddings. Table 4 shows five clusters with the lowest training accuracy with the potential reasons for the bad performance highlighted. Here, NULL represents an empty atom when the model generated explanations that were shorter than the predefined length L.

We make the following observations. First, **cluster 5** with the lowest training accuracy is caused by non-English reviews in the dataset. This accounts for a small fraction (around 0.31%), which increases the possibility of underfitting. Second, **cluster 4** corresponds to short samples (average length: 119), which contain an overwhelming number of NULL atoms (on average 2 NULL atoms per explanation). This means that the model cannot find enough evidence for prediction. Third, cluster 3 contains 13.4% samples that have positive sentiment, and its training accuracy (98.6%) is much larger than cluster 4. However, some samples in this cluster tend to include "vegas" in the explanation, which does not seem directly related to sentiment classification. Fourth, clusters 1 and 2 have reasonable generated atoms, which is consistent with their good training accuracy (99.2%).

Our self-explaining framework allows us to directly steer the model. For example, after identifying the potentially noisy feature "vegas", we can prevent the model from including the term by removing the feature from the candidate atom list $\mathcal{C}_i$. This type of refinement can be easily achieved during testing, unlike the efforts-taking dataset calibration or model retraining. Fig. 4 shows the performance change of the 169 samples that previously included "vegas" in their explanations. The left bottom histogram shows the atoms that are more frequently generated after removing "vegas." The model sometimes relies on similar atoms such as "las" or does not find a good candidate (choosing NULL), which may

Table 4: Case study on Yelp. We cluster the explanations for the training samples and show the 5 clusters with the lowest training accuracy. Num is the number of explanations in the cluster, and Len is the average text length of the reviews. Potential reasons for bad performance are marked in brown.

	Acc	Label	Num	Len	Atoms in the explanations (ordered by frequency)
1	99.2	99.1% Neg	1,763 8.82%	643	not(290) bad(233) no(185) mediocre(153) bland(149) never(123) again(122) worst(119) ok(115) disappointing(115) terrible(107)
2	99.2	99.2% Pos	2,730 13.7%	584	great(667) delicious(508) best(330) love(294) fresh(285) tasty(255) definitely(254) friendly(232) perfect(199) amazing(198) favorite(194)
3	98.6	98.6% Pos	2,686 13.4%	548	great(793) friendly(329) always(315) best(304) love(295) fun(223) definitely(218) helpful(163) awesome(159) amazing(136) vegas(117)
4	93.2	57.8% Pos	848 4.24%	119	NULL(1738) great(133) not(67) best(39) service(39) love(34) friendly(29) good(26) awesome(22) fast(22) overpriced(20)
5	83.9	58.1% Neg	62 0.31%	682	NULL(24) nicht(13) un(11) eine(9) service(8) pas(8) der(8) die(7) und(7) um(6) den(5) pour(5) de(5) das(4) prix(4) je(4) zu(3) im(3)

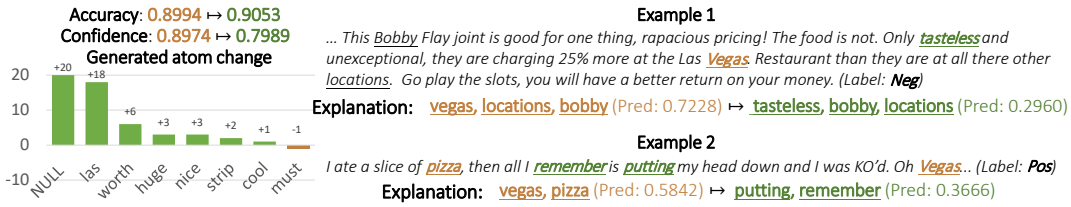


Figure 4: Steering the model without re-training. The self-explaining framework allows users to exclude noisy features from explanations during testing, which may simultaneously improve explanation quality and prediction accuracy. This figure shows the performance before and after removing "vegas."

Table 5: Time costs in seconds on Yelp (BERT) and Adult.

	Consequent estimator training	Deep model training (1 epoch)				Explanation generation (1 sample)				
		Base	SENN	RCN	Ours	LIME	Anchor	SENN	RCN	Ours
Yelp	2041.4	571.9	224.5	503.7	665.6	55.0	2854.2	0.037	0.071	0.055
Adult	1502.8	12.8	9.1	953.9	98.2	2.5	1.18	0.02	0.17	0.015

lead to decreased confidence. However, the probability of including more meaningful atoms also increases (e.g., "worth" in the histogram, and "tasteless" in Example 1). One may also verify assumptions by checking the samples whose prediction score changes. For example, after removing "vegas," the model can no longer predict Example 2 correctly. In Example 2, there is no strong indication of sentiment, and "vegas" may be the most helpful feature. This contradicts our previous assumption ("vegas" is useless for sentiment classification), which provides new insights and guidance for further improvement (e.g., punishing "vegas" with a soft punishment instead of directly removing it).

3.4 Efficiency

Table 5 shows that post-hoc explanation methods like LIME and Anchor require longer time to generate an explanation. RCN has the largest complexity among the self-explaining methods since it enumerates all possible rules and combines them with soft attention. To alleviate this problem, RCN uses a predefined rule set and hence its efficiency becomes dependent on the size and quality of the rule set. In contrast, our model is trained within acceptable time even for large solution space, and humans only need to define the types of atoms and logical connectives.

Moreover, during testing, we generate each explanation with a linear complexity with the explanation length L, while RCN goes over all possible rules, whose complexity typically increases exponentially with L. Despite the advantages in training and testing, our method requires a separate neural consequent estimator, which requires logic rule sampling and additional neural training that takes 35 minutes on Yelp and 25 minutes on Adult. We consider this acceptable, as this step is only required once on each dataset. The consequent estimator can be reused when training the self-explaining models with different hyperparameters or backbone networks.

4 Conclusion and Broader Impact

This work presented a new method to incorporate self-explanatory capabilities into a deep model to maintain high prediction performance while also providing high human precision. This was accomplished by a novel deep reasoning framework that enables a model to predict and explain its decisions using logic rules. Our method does not require predefined rule sets and can be learned using widely used deep learning modules in a differentiable way. Extensive tests involving human evaluation show that our method achieves high prediction performance and human precision while being resistant to noisy labels.

Although this research aimed to seamlessly connect the decision-making processes of humans and machines, it is unclear whether the explanations will lead to overconfidence in the model's performance or whether malicious users will manipulate steerable self-explaining models. Answering these questions may require closer collaboration among machine learning, human-machine interaction, and psychology researchers, and we are interested in actively pursuing such collaboration in the future.

References

[1] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?" explaining the predictions of any classifier. In *KDD*, 2016.

[2] Sebastian Thrun. Extracting rules from artificial neural networks with distributed representations. 1994.

[3] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Nothing else matters: Model-agnostic explanations by identifying prediction invariance. *stat*, 2016.

[4] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *EMNLP*, 2016.

[5] Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. Automatic text scoring using neural networks. In *ACL*, 2016.

[6] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE TVCG*, 2017.

[7] W James Murdoch, Peter J Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. In *ICLR*, 2018.

[8] Georgina Peake and Jun Wang. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *KDD*, 2018.

[9] Jian Liang, Bing Bai, Yuren Cao, Kun Bai, and Fei Wang. Adversarial infidelity learning for model interpretation. In *KDD*, 2020.

[10] Jingyue Gao, Xiting Wang, Yasha Wang, Yulan Yan, and Xing Xie. Learning groupwise explanations for black-box models. In *IJCAI*, 2021.

[11] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *NeurIPS*, 2018.

[12] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 2019.

[13] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *NeurIPS*, 2017.

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, 2018.

[15] Chaoyu Guan, Xiting Wang, Quanshi Zhang, Runjin Chen, Di He, and Xing Xie. Towards a deep and unified understanding of deep neural models in nlp. In *ICML*, 2019.

[16] Sungsoo Ray Hong, Jessica Hullman, and Enrico Bertini. Human factors in model interpretability: Industry practices, challenges, and needs. *PACM HCI*, 2020.

[17] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *AOAS*, 2015.

[18] Hongyu Yang, Cynthia Rudin, and Margo Seltzer. Scalable bayesian rule lists. In *ICML*, 2017.

[19] Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 2018.

[20] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists. In *KDD*, 2017.

[21] Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. Interpretable and steerable sequence learning via prototypes. In *KDD*, 2019.

[22] Zhongxia Chen, Xiting Wang, Xing Xie, Mehul Parsana, Akshay Soni, Xiang Ao, and Enhong Chen. Towards explainable conversational recommendation. In *IJCAI*, 2020.

[23] Yuzuru Okajima and Kunihiko Sadamasa. Deep neural networks constrained by decision rules. In *AAAI*, 2019.

[24] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. 2017.

[25] Kangzhi Zhao, Xiting Wang, Yuren Zhang, Li Zhao, Zheng Liu, Chunxiao Xing, and Xing Xie. Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs. In *SIGIR*, 2020.

[26] Danyang Liu, Jianxun Lian, Zheng Liu, Xiting Wang, Guangzhong Sun, and Xing Xie. Reinforced anchor knowledge graph generation for news recommendation reasoning. In *KDD*, 2021.

[27] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. Neural collaborative reasoning. In *WebConf*, 2021.

[28] Xiting Wang, Kunpeng Liu, Dongjie Wang, Le Wu, Yanjie Fu, and Xing Xie. Multi-level recommendation reasoning over knowledge graphs with reinforcement learning. In *WebConf*, 2022.

[29] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *stat*, 2017.

[30] Henri F Ellenberger. *The Discovery of the Unconscious: The History and Evolution of Dynamic Psychiatry*. 1970.

[31] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[33] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *SSST*, 2014.

[34] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. 2015.

[35] Open Data Science (`ODS.ai`). Kaggle clickbait news detection. `https://www.kaggle.com/c/clickbait-news-detection`, 2020.

[36] Dheeru Dua and Casey Graf. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2017.

[37] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *ICLR*, 2019.

[38] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[39] Speechocean. `https://en.speechocean.com/`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 4.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2.1 and Section 2.3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 2.2, Section 2.4 and the Appendix - Theoretical proof in the supplement.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] They would be included in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix - Implementation details in the supplement.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Appendix - Implementation details in the supplement.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 3.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 3.1.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] The code of our proposed model is in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] See Section 3.3 for text of instructions, and see Appendix - User study in the supplement for more details and screenshots.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes] See Appendix - User study in the supplement.