

Neural Probabilistic Circuits: Enabling Compositional and Interpretable Predictions through Logical Reasoning

Weixin Chen^{*1}, Simon Yu^{*1}, Huajie Shao², Lui Sha¹, and Han Zhao^{†1}

¹University of Illinois Urbana-Champaign

²College of William and Mary

{weixinc2,jundayu2,lrs,hanzhao}@illinois.edu, {hshao}@wm.edu

Abstract

End-to-end deep neural networks have achieved remarkable success across various domains but are often criticized for their lack of interpretability. While post hoc explanation methods attempt to address this issue, they often fail to accurately represent these black-box models, resulting in misleading or incomplete explanations. To overcome these challenges, we propose an inherently transparent model architecture called Neural Probabilistic Circuits (NPCs), which enable compositional and interpretable predictions through logical reasoning. In particular, an NPC consists of two modules: an attribute recognition model, which predicts probabilities for various attributes, and a task predictor built on a probabilistic circuit, which enables logical reasoning over recognized attributes to make class predictions. To train NPCs, we introduce a three-stage training algorithm comprising attribute recognition, circuit construction, and joint optimization. Moreover, we theoretically demonstrate that an NPC’s error is upper-bounded by a linear combination of the errors from its modules. To further demonstrate the interpretability of NPC, we provide both the most probable explanations and the counterfactual explanations. Empirical results on four benchmark datasets show that NPCs strike a balance between interpretability and performance, achieving results competitive even with those of end-to-end black-box models while providing enhanced interpretability.

1 Introduction

End-to-end deep neural networks (DNNs) [Krizhevsky et al., 2012, He et al., 2016, Vaswani et al., 2017, Devlin et al., 2019] have demonstrated remarkable success across various domains [Hinton et al., 2012, Sutskever et al., 2014, Long et al., 2015]. However, many of them are black-box models containing complex operators, making it hard to interpret and understand how a decision was made. Although many efforts [Ribeiro et al., 2016, Lundberg and Lee, 2017, Selvaraju et al., 2017] have been made to explain a model’s decision in a post hoc manner, Alvarez-Melis and Jaakkola [2018], Laugel et al. [2019], Slack et al. [2020], Rudin [2019] show that these explanations are oftentimes not reliable as the explanation model might loosely approximate the underlying model. For example, the explanation model exhibits similar performance to the black-box model but yet relies on entirely different features. Such discrepancy between the explanation model and the black-box model could lead to misleading explanations, *e.g.*, attributing the decision to irrelevant features or missing out important features. Misleading

*Equal contributions.

†Corresponding author.

explanations are particularly concerning in high-stake applications such as medical analysis [Hou et al., 2024, Liu et al., 2023] and legal justice [Richmond et al., 2024, Deeks, 2019]. Rather than introducing post hoc explanations to explain a black-box model, Rudin [2019] argues that one should create an interpretable model in the first place where each component is designed with a distinct purpose, facilitating an interpretable prediction.

Concept bottleneck models (CBMs) [Koh et al., 2020] aim to enhance interpretability by introducing high-level, human-understandable concepts, such as “red color” and “round shape”, as an intermediate bottleneck, which decomposes a model into two modules: a concept recognition model and a task predictor. The neural-network-based concept recognition model maps the input image to probabilities associated with various concepts. Using these probabilities, the task predictor, typically a linear predictor, produces the probabilities for the various classes. Since the final prediction (*i.e.*, the class with the highest probability) can be interpreted in terms of these concepts, the model’s decision-making process becomes more intuitive for humans to understand. To improve performance on downstream tasks, methods like CEM [Zarlenga et al., 2022], ProbCBM [Kim et al., 2023], and others [Yeh et al., 2020, Kazhdan et al., 2020] change the outputs of the concept recognition model from concept probabilities to concept embeddings. While boosting task performance, such approaches significantly compromise interpretability since the dimensions within concept embeddings lack semantic meanings. On the other hand, to further improve interpretability, some approaches [Barbiero et al., 2023, Rodríguez et al., 2024, Ciravegna et al., 2023] propose architectures for the task predictor that incorporate logical rules, allowing task predictions to be explicitly explained through these rules. However, these logical rules are usually learned from data rather than being predefined by humans, limiting our ability to integrate prior domain knowledge into the model. Additionally, there is currently no theoretical guarantee regarding the performance of the overall model, obscuring the relationship between overall performance and that of the concept recognition model or the task predictor.

To address these challenges, we propose a novel model architecture called Neural Probabilistic Circuits (NPCs), which enable compositional and interpretable predictions through logical reasoning. An NPC comprises two modules: an attribute recognition model and a task predictor. Unlike existing approaches that primarily focus on numerous binary concepts (*e.g.*, “red color”, “yellow color”), we introduce a higher-level categorical characteristic called attributes, which describe the types of concepts (*e.g.*, “color”). This approach reduces the need for additional concept selection or pruning to improve model efficiency [Ciravegna et al., 2023, Barbiero et al., 2022, Zarlenga et al., 2023], while also achieving better performance in concept recognition. Given an input image, the neural-network-based attribute recognition model produces probability vectors for various attributes, with each vector representing the likelihood of various values for the corresponding attribute. These probability vectors serve as inputs to the task predictor, which is implemented using a probabilistic circuit. Probabilistic circuits [Poon and Domingos, 2011, Zhao et al., 2016b, 2015b, Choi et al., 2020], a type of graphical models [Koller and Friedman, 2009], aim to learn the joint distribution over input variables, in our case, attribute variables and the class variable. During learning, probabilistic circuits embed within their structures and parameters either implicit logical rules learned from data or explicit logical rules predefined by humans. The circuits enable tractable probabilistic reasoning tasks such as joint, marginal, and conditional inferences, thereby revealing relations among the attributes and classes. By leveraging these relations, NPCs can reason over outputs from the attribute recognition model to infer the most probable class. Specifically, the prediction score for a given class is the sum of the likelihood of each combination of attribute values weighted by their relevance to the class. As usual, the final prediction corresponds to the class with the highest score.

Given the compositional nature of NPCs, we propose a three-stage training algorithm. Specifically, the whole procedure involves the following stages: 1) Attribute recognition: We begin by training the attribute recognition model within a multi-task learning framework [Caruana, 1997, Ruder, 2017]. 2) Circuit construction: Next, we construct the circuit using two distinct approaches: i) Data-driven approach learns the circuit’s structure and optimizes its parameters based on data, allowing the underlying logical rules to be embedded within the circuit. ii) Knowledge-injected approach manually designs the circuit’s structure and assigns its parameters

to ensure that human-predefined logical rules are explicitly encoded within the circuit. 3) Joint optimization: Finally, the two modules are jointly optimized in an end-to-end manner to further enhance the overall model’s performance on downstream tasks.

To provide theoretical guarantees regarding the performance of the overall model, we demonstrate that, due to its compositional nature and the use of probabilistic circuits, NPCs exhibit a compositional error bound—the error of the overall model is upper-bounded by a linear combination of the errors from the various modules.

In addition, we provide various types of explanations to make it easier for humans to understand NPC’s predictions: 1) Most Probable Explanation (MPE) identifies the combination of attribute values that contributes most significantly to the predicted class. 2) Counterfactual Explanation (CE) answers the question: Would the model have made the correct prediction had the likelihoods of certain attribute values been adjusted?

Empirical results on four image classification datasets demonstrate NPC’s ability to strike an impressive balance between interpretability and performance on downstream tasks. In particular, NPC outperforms three representative concept-based models and delivers results competitive even with those of an end-to-end deep neural network. Additionally, we perform extensive ablation studies to investigate the advantages of integrating attributes over concepts, along with the impacts of attribute selections, predictor design approaches, and joint optimization.

Our main contributions are as follows,

1. We introduce Neural Probabilistic Circuits (NPCs), a novel model architecture that combines a neural-network-based attribute recognition model and a probabilistic-circuit-based task predictor, enabling compositional and interpretable predictions through logical reasoning.
2. We develop a three-stage training algorithm for NPCs, consisting of 1) attribute recognition through multi-task learning, 2) circuit construction via both data-driven and knowledge-injected approaches, and 3) end-to-end joint optimization.
3. To the best of our knowledge, we are the first to provide a theoretical guarantee on the performance of the compositional bottleneck models, which shows that NPC’s error can be upper-bounded by a linear combination of the errors from its modules.
4. We provide various types of explanations to facilitate human understanding of NPCs’ predictions, including most probable explanations and counterfactual explanations.
5. We empirically show that NPC demonstrates competitive performance in image classification tasks while providing enhanced interpretability.

2 Preliminaries

Probabilistic circuits are a class of graphical models that is used to express a joint distribution over a set of random variables $Z_{1:N}$. A probabilistic circuit f_S (henceforth simply referred to as a *circuit*) consists of a rooted directed acyclic graph where leaf nodes are univariate indicators of categorical variables¹ (*i.e.*, $\mathbb{I}(Z_i = z_i)$, $z_i \in \mathcal{Z}_i$, $i \in [N]$) and internal nodes consist of sum nodes and product nodes. Each sum node computes a weighted sum of its children, and each product node computes a product of its children. In an unnormalized circuit, the root node outputs the unnormalized joint probability over variables. Any unnormalized circuit can be transformed into an equivalent, normalized circuit via weight updating Peharz et al. [2015], Zhao et al. [2015a]. Hence, without loss of generality, we always assume that f_S is normalized; thus, $f_S(z_{1:N}) = \Pr(Z_{1:N} = z_{1:N})$. Readers are referred to Sánchez-Cauce et al. [2021] for more details on circuits.

In circuits, the *scope* of a node is defined as the set of variables that have indicators among the node’s descendants, which can be computed recursively—if v is a leaf node, say, an indicator over Z_i , then $\text{scope}(v) = \{Z_i\}$; otherwise, $\text{scope}(v) = \cup_{\tilde{v} \in \text{children}(v)} \text{scope}(\tilde{v})$. A circuit

¹We mainly focus on probabilistic circuits over categorical random variables. An extension to the continuous ones is standard.

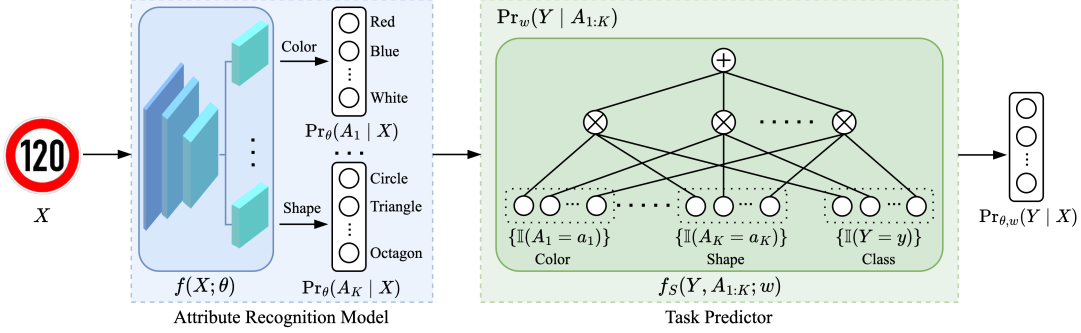


Figure 1: Model architecture of an NPC, consisting of an attribute recognition model and a task predictor. The attribute recognition model is a neural network $f(X; \theta)$ which takes an image X as input and outputs K probability vectors $\{\Pr_{\theta}(A_k | X)\}_{k=1}^K$. The task predictor is a probabilistic circuit $f_S(Y, A_{1:K}; w)$ taking an instance of attributes as input and providing the conditional probability $\Pr_w(Y | A_{1:K})$. By leveraging these relations between classes and attributes alongside the probability distributions of various attributes, NPC produces the probability vector $\Pr_{\theta, w}(Y | X)$.

is *smooth* iff each sum node has children with identical scope. A circuit is *decomposable* iff each product node has children with disjoint scopes. If a circuit is smooth and decomposable, then any marginal probability can be computed by setting the leaf nodes corresponding to the marginalized variables to 1. Consequently, inferences are efficient in a circuit as any joint, marginal, or conditional inference can be computed by at most two passes in a circuit. For instance, $\Pr(Z_1 = z_1 | Z_{2:N} = z_{2:N}) = \frac{\Pr(Z_{1:N} = z_{1:N})}{\Pr(Z_{2:N} = z_{2:N})} = \frac{f_S(Z_{1:N} = z_{1:N})}{f_S(Z_1 = \emptyset, Z_{2:N} = z_{2:N})}$ where $Z_1 = \emptyset$ implies $\mathbb{I}(Z_1 = \tilde{z}_1) = 1, \forall \tilde{z}_1 \in \mathcal{Z}_1$; thus, computing a conditional probability only requires two forward processes in a circuit, each in linear time *w.r.t.* its size. In this paper, we focus on smooth and decomposable circuits.

3 Neural Probabilistic Circuits

In this section, we introduce Neural Probabilistic Circuits (NPCs). We begin by describing the model architecture and the inference process, illustrating how NPCs enable compositional and interpretable predictions through logical reasoning (Section 3.1). Next, we elaborate on the three-stage training algorithm for training NPCs. In particular, we propose two distinct approaches for building circuits: a data-driven approach and a knowledge-injected approach (Section 3.2). Finally, we provide a theoretical analysis establishing the relationship between the error of the overall model and those of its individual modules (Section 3.3).

3.1 Model Architecture and Inference

Figure 1 presents an overview of an NPC, which consists of an attribute recognition model and a task predictor. The attribute recognition model is a neural network that processes an input image to identify its high-level visual attributes, such as color and shape. The task predictor is a (normalized) probabilistic circuit that models the joint distribution over attributes and classes, embedding either implicit or explicit logical rules within its structure and parameters during learning. The circuit enables efficient probabilistic reasoning, including joint, marginal, and conditional inferences. Specifically, given a particular assignment of attributes, the circuit can infer the probability of a specific class. By leveraging these conditional dependencies alongside the probability distributions of the various attributes (*i.e.*, outputs from the attribute recognition model), NPC produces the probabilities of the image belonging to various classes. The class with the highest probability is recognized as the predicted class.

Formally, let $X \in \mathcal{X}, A_k \in \mathcal{A}_k, Y \in \mathcal{Y}$ denote the input variable, the k -th attribute variable, and the class variable. The variables’ instantiations are represented by x, a_k, y , respectively. In particular, we consider K attributes, *i.e.*, A_1, \dots, A_K (or $A_{1:K}$ in short). Each attribute A_k has q_k possible values, *i.e.*, $|\mathcal{A}_k| = q_k$. The attribute recognition model $f(X; \theta)$ is parameterized by θ . Given an input instance x , the model outputs K probability vectors. The k -th probability vector, denoted as $f_k(x; \theta) \in \mathbb{R}^{q_k}$, shows the probabilities of x ’s k -th attribute taking different values a_k , *i.e.*, $[f_k(x; \theta)]_{a_k} = \Pr_\theta(A_k = a_k | X = x)$. The task predictor $f_S(Y, A_{1:K}; w)$ is a probabilistic circuit with structure S and parameters w , which models the joint distribution over $Y, A_{1:K}$. Specifically, when taking an instance of attributes $a_{1:K}$ and a class label y as input, the circuit outputs the joint probability $\Pr_w(Y = y, A_{1:K} = a_{1:K})$. By the virtue of its efficient inferences, the circuit also supports efficient conditional queries, *e.g.*, $\Pr_w(Y = y | A_{1:K} = a_{1:K}) = f_S(y, a_{1:K}; w) / f_S(\emptyset, a_{1:K}; w)$.

Prior to describing how an NPC predicts a class, we make the following mild assumptions on the selected attributes.

Assumption 1 (Sufficient Attributes). *Given the attributes, the class label is conditionally independent of the input, *i.e.*, $Y \perp X | A_1, \dots, A_K$.*

Assumption 2 (Complete Information). *Given any input, all attributes are conditionally mutually independent, *i.e.*, $A_1 \perp A_2 \perp \dots \perp A_K | X$.*

Remarks. The first assumption essentially assumes that the attributes are sufficient to infer the class label of interest. The second assumption assumes the input contains complete information regarding the attributes such that they are conditionally mutually independent. These assumptions are mild and often hold in practice. For instance, in the context of traffic signs, if the attributes include the shape (*e.g.*, circle), color (*e.g.*, red), and symbol (*e.g.*, slash) of a sign, they collectively provide enough information to infer the class label (*e.g.*, no entry) without requiring additional details from the raw image. On the other hand, the raw image fully encodes the attributes (*e.g.*, shape, color, and symbol). Once the input is observed, these attributes can be independently determined.

Under Assumption 1 and 2, an NPC outputs the probability of an input x being a class y as follows,

$$\begin{aligned} \Pr_{\theta, w}(Y = y | X = x) &= \sum_{a_{1:K}} \Pr_w(Y = y | A_{1:K} = a_{1:K}, X = x) \cdot \Pr_\theta(A_{1:K} = a_{1:K} | X = x) \\ &= \sum_{a_{1:K}} \underbrace{\Pr_w(Y = y | A_{1:K} = a_{1:K})}_{\text{task predictor}} \cdot \underbrace{\prod_{k=1}^K \Pr_\theta(A_k = a_k | X = x)}_{\text{attribute recognition model}}. \end{aligned} \quad (1)$$

Equation (1) is derived using the assumptions, and the two interior terms are given by the circuit-based task predictor and the attribute recognition model, respectively. Subsequently, the predicted class is the one with the largest probability, *i.e.*, $\hat{y} = \arg \max_{y \in \mathcal{Y}} \Pr_{\theta, w}(Y = y | X = x)$.

In summary, we propose a novel model architecture for image recognition tasks. The architecture is interpretable by design, thanks to the integration of an attribute bottleneck and the probabilistic semantics of probabilistic circuits. Together, these modules enable predictions which can be interpreted using the likelihood of different attributes and the conditional dependencies between attributes and classes.

3.2 Three-Stage Training Algorithm

In this section, we will propose a three-stage training algorithm for NPCs comprising the following stages: 1) attribute recognition through multi-task learning (Section 3.2.1), 2) circuit construction via both data-driven and knowledge-injected approaches (Section 3.2.2), and 3) joint optimization (Section 3.2.3).

3.2.1 Attribute Recognition

We aim to train the attribute recognition model $f(X; \theta)$ such that each attribute is recognized well. To this end, we adopt a multi-task learning framework [Zhang and Yang, 2021], where each task is to recognize a particular attribute. Specifically, we use the cross-entropy loss for each task and assign weights to the task losses based on the size of the corresponding attribute space. These weights normalize the task losses, preventing certain tasks from dominating the training process [Kendall et al., 2018, Grégoire et al., 2024, Wang and Chen, 2020]. The overall training loss for attribute recognition is defined as follows,

$$\mathcal{L}_{\text{Attribute}}(\theta; D) = -\frac{1}{K} \sum_k \frac{1}{\log q_k} \left(\frac{1}{|D|} \sum_{x \in D} g_k^T(x) f_k(x; \theta) \right). \quad (2)$$

The term inside the parentheses represents the mean cross-entropy loss over the training dataset D , where $f_k(x; \theta) \in \mathbb{R}^{q_k}$ and $g_k(x) \in \mathbb{R}^{q_k}$ are, respectively, the output vector and the label vector corresponding to the k -th attribute. Specifically, $g_k(x)$ is a probability vector that sums to one, with each entry representing the ground-truth likelihood of x having a particular attribute value. For instance, for the color attribute, an image of a polar bear would have a one-hot label vector with 1 representing “white”. In contrast, an image of a zebra might have a probabilistic label vector, with 0.5 representing “black” and 0.5 representing “white”.

3.2.2 Circuit Construction

We aim to construct a probabilistic circuit $f_S(Y, A_{1:K}; w)$ that models the joint distribution over $Y, A_{1:K}$. To achieve this, we propose two distinct approaches for building the circuit’s structure and parameters: a data-driven approach and a knowledge-injected approach.

Data-Driven Approach This approach *learns a circuit’s structure and optimizes its parameters* using data in the form of $\{(y, a_{1:K})\}$. As described above, the training dataset is defined as $D = \{(x, g_{1:K}(x), y)\}$, where each attribute label is represented as a probability vector rather than a single value. For each data, we sample an a_k from the categorical distribution defined by $g_k(x)$, *i.e.*, $a_k \sim \text{Categorical}(g_k(x))$, $k \in [K]$, which results in a processed dataset $\bar{D} = \{(x, y, a_{1:K})\}$. **1) Structure Learning:** LearnSPN Gens and Domingos [2013] is a mainstream algorithm for learning a circuit’s structure from data. LearnSPN recursively identifies independent groups to create product nodes, clusters data to form sum nodes, and assigns single variables as leaf nodes. In our approach, we apply LearnSPN on \bar{D} to derive a structure tailored to the observed data. **2) Parameter Learning:** Given the learned structure, optimizing the circuit’s weights (*i.e.*, the weights of edges emanating from sum nodes) is framed as a maximum likelihood estimation (MLE) problem, with the following loss function:

$$\mathcal{L}_{\text{MLE}}(w; D) = - \sum_{(y, a_{1:K}) \in \bar{D}} \log f_S(y, a_{1:K}; w).$$

We adopt the widely used CCCP algorithm Zhao et al. [2016b] which iteratively applies multiplicative weight updates on w to minimize $\mathcal{L}_{\text{MLE}}(w; \bar{D})$. CCCP is guaranteed to converge monotonously. Overall, with the learned structure and optimized parameters, the circuit captures the underlying logical rules present in the observed data, thus effectively modeling the joint distribution over attributes and classes.

Knowledge-Injected Approach Incorporating domain knowledge into a model helps ensure that its behavior aligns with the human’s understanding of the domain. In practice, domain knowledge can be represented as a set of weighted logical rules. These rules are usually derived by observing patterns in existing samples, with the weight of each rule reflecting the frequency with which the rule holds true among the observed data. For instance, in the context of traffic signs, a rule could be: $\mathbb{I}(\text{shape} = \text{circle}) \wedge \mathbb{I}(\text{color} = \text{red}) \wedge \mathbb{I}(\text{symbol} = \text{slash}) \wedge \mathbb{I}(\text{class} = \text{no entry})$,

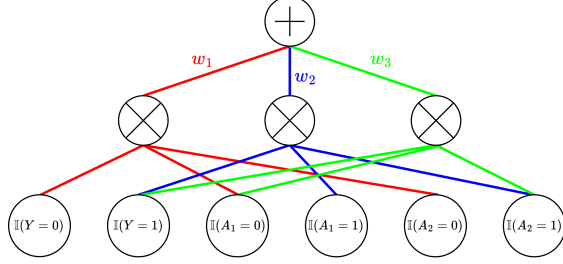


Figure 2: Illustration of a probabilistic circuit constructed by the knowledge-injected approach. The circuit encodes the set of weighted logical rules: $\{w_1 \cdot (\mathbb{I}(A_1 = 0) \wedge \mathbb{I}(A_2 = 0) \wedge \mathbb{I}(Y = 0)), w_2 \cdot (\mathbb{I}(A_1 = 1) \wedge \mathbb{I}(A_2 = 1) \wedge \mathbb{I}(Y = 1)), w_3 \cdot (\mathbb{I}(A_1 = 0) \wedge \mathbb{I}(A_2 = 1) \wedge \mathbb{I}(Y = 1))\}$.

as this holds true for an image of a “no entry” sign. The rules summarized for various tasks, along with the process of establishing them, are detailed in Appendix E. By leveraging these rules, the knowledge-injected approach *manually constructs a circuit’s structure and assigns its parameters* to model the joint distribution over $Y, A_{1:K}$. **1) Structure Construction:** We construct the structure of the circuit to be a depth-2 weighted sum-of-products formula. More specifically, consider L rules of the form $\{r_l := \mathbb{I}(A_1 = a_1^l) \wedge \dots \wedge \mathbb{I}(A_K = a_K^l) \wedge \mathbb{I}(Y = y^l)\}_{l=1}^L$. The structure is constructed as follows: i) A set of leaf nodes is created to represent the indicator variables of $Y, A_{1:K}$. ii) Based on these leaf nodes, a layer of product nodes is built, where each product node is associated with a rule. Specifically, the l -th product node connects to the leaf nodes that represent the conditions in the rule r_l , *i.e.*, $\mathbb{I}(A_1 = a_1^l), \dots, \mathbb{I}(A_K = a_K^l), \mathbb{I}(Y = y^l)$. iii) A single sum node, which serves as the root node of the circuit, is placed above the product node layer. This sum node aggregates the outputs of all product nodes. **2) Parameter Assignment:** The parameters in the circuit refer to the weights of edges connecting the product nodes to the sum node. The weight for the l -th edge is assigned as the frequency of the rule r_l . An example of a circuit constructed using this approach is illustrated in Figure 2. Through these two steps, the human-predefined logical rules are manually encoded into the circuit’s structure and parameters. Proposition 1 ensures that the output of the circuit’s root node represents the empirical joint probability over attributes and classes.

Proposition 1. *The circuit constructed using the knowledge-injected approach models the empirical joint distribution over attributes and classes. Specifically, the output of the root node represents the empirical joint probability of attributes and classes.*

Proof. Let R denote the rule variable. For a given instance $(y, a_{1:K})$, the output of the root node is computed as,

$$\begin{aligned} f_S(y, a_{1:K}; w) &= \sum_l \Pr_w(R = r_l) \cdot \mathbb{I}(a_1 = a_1^l) \dots \mathbb{I}(a_K = a_K^l) \cdot \mathbb{I}(y = y^l) \\ &= \sum_l \Pr_w(R = r_l) \cdot \Pr_w(A_{1:K} = a_{1:K}, Y = y \mid R = r_l) \\ &= \Pr_w(A_{1:K} = a_{1:K}, Y = y), \end{aligned}$$

where $\Pr_w(R = r_l)$ is the weight for the l -th edge and $\mathbb{I}(a_1 = a_1^l) \dots \mathbb{I}(a_K = a_K^l) \cdot \mathbb{I}(y = y^l)$ is the output of the l -th product node. The condition $R = r_l$ indicates that the rule $\mathbb{I}(A_1 = a_1^l) \wedge \dots \wedge \mathbb{I}(A_K = a_K^l) \wedge \mathbb{I}(Y = y^l)$ is satisfied. \square

3.2.3 Joint Optimization

Thanks to the differentiability of circuits, NPCs can be fine-tuned in an end-to-end manner to further improve the performance of the overall model on downstream tasks. Specifically, the

loss function is defined as follows,

$$\mathcal{L}_{\text{Joint}}(\theta, w; (x, y)) = - \sum_{(x,y) \in D} \log \Pr_{\theta,w}(Y = y | X = x). \quad (3)$$

To optimize this loss, we simply employ the stochastic gradient descent algorithm for updating θ , while the projected gradient descent algorithm is used to update w to ensure the positivity of the circuit weights. The detailed optimization process is provided in Appendix A.

3.3 Theoretical Analysis

In this section, we present an error analysis for NPCs to understand how the performance of individual modules affects that of the overall model. Given the overall model and the attribute recognition model are discriminative models, while the probabilistic circuit is a generative model, we define the following errors: **1) Error of the overall model:** $\epsilon_{\theta,w} := \mathbb{E}_X [d_{\text{TV}}(\Pr_{\theta,w}(Y | X), \Pr(Y | X))]$, which represents the expected total variance distance between the learned and true conditional distributions of Y given X . **2) Error of the attribute recognition model:** $\epsilon_{\theta} := \mathbb{E}_X [d_{\text{TV}}(\Pr_{\theta}(A_{1:K} | X), \Pr(A_{1:K} | X))]$, which quantifies the expected total variation distance between the learned and true conditional distributions of the attributes $A_{1:K}$ given X . Additionally, we define $\epsilon_{\theta}^k := \mathbb{E}_X [d_{\text{TV}}(\Pr_{\theta}(A_k | X), \Pr(A_k | X))]$ as the error for each individual attribute A_k . **3) Error of the probabilistic circuit:** $\epsilon_w := d_{\text{TV}}(\Pr_w(Y, A_{1:K}), \Pr(Y, A_{1:K}))$, which measures the total variation distance between the learned and true joint distributions of Y and $A_{1:K}$. The above errors capture how closely the learned models approximate the underlying true distributions.

Theorem 2 (Compositional Error). *Under Assumptions 1 and 2, the error of an NPC is bounded by a linear combination of the errors of the attribute recognition model and the circuit-based task predictor. In particular, the error of the attribute recognition model across all attributes is bounded by the sum of the errors for each attribute, i.e.,*

$$\epsilon_{\theta,w} \leq \epsilon_{\theta} + 2\epsilon_w \leq \sum_{k=1}^K \epsilon_{\theta}^k + 2\epsilon_w.$$

Proof Sketch. By leveraging Equation (1), the upper bound of $\epsilon_{\theta,w}$ is decomposed into two terms. The first term depends only on θ and represents the error of the attribute recognition model, while the second term depends only on w and captures the error of the circuit. In particular, the overall error of the attribute recognition model is further expanded into the errors across individual attributes. \square

The complete proof is deferred to Appendix B. Theorem 2 demonstrates that reducing the error for any single attribute helps reduce the overall error of the attribute recognition model. More importantly, the error bound of an NPC is decomposable into contributions from its individual modules, which accredits to the compositional nature of NPCs and the incorporation of probabilistic circuits. Consequently, reducing the error of any individual module helps improve the performance of the NPC.

4 Model Explanations

As discussed in Section 3.1, model predictions can be interpreted using the attribute recognition results and the conditional dependencies between classes and attributes. To further enhance the human’s understanding of the model predictions, we provide various explanations addressing the following questions: 1) *Which assignment of attributes contributes most to the model prediction?* 2) *In cases where the model prediction was incorrect, could an adjustment to the attribute recognition results lead to a correct prediction?* With a slight abuse of notation, let θ denote the parameters of the trained attribute recognition model and let S, w denote the structure and parameters of the constructed probabilistic circuit.

4.1 Most Probable Explanations

To address the first question, we define Most Probable Explanations (MPEs) for NPCs for identifying the highest contributing attribute assignments.

Definition 1 (Most Probable Explanations). *Given an input x with prediction \hat{y} , the most probable explanation is defined as the attribute assignment that contributes the most in $\Pr_{\theta,w}(Y = \hat{y} \mid X = x)$. Formally, $a_{1:K}^* := \arg \max_{a_{1:K}} \Pr_w(Y = \hat{y} \mid A_{1:K} = a_{1:K}) \cdot \prod_{k=1}^K \Pr_{\theta}(A_k = a_k \mid X = x)$.*

MPE inference is generally challenging for probabilistic circuits. Such inference is tractable for selective circuits [Sánchez-Cauce et al., 2021], but this type of circuit is relatively restrictive in expressiveness. As the number of attributes is small in our experimental settings, we simply use the brute-force algorithm to infer MPEs. Developing more efficient heuristics for MPE inference remains an open problem and is not the primary focus of this paper. Therefore, we leave it for future work.

MPEs provide a concrete explanation as to how the model arrives at a specific class prediction. Specifically, the predicted class is primarily due to the input image’s attributes being recognized as $a_{1:K}^*$. These explanations offer attribute-level insights into the model’s predictions, thereby enhancing interpretability and the human’s understanding of the predictions.

To gain deeper insights into how these explanations represent the model’s behavior, we define a property for MPEs, called *alignment*, and introduce a corresponding metric to characterize the behavior of the model.

Definition 2 (Alignment Rate). *An MPE $a_{1:K}^*$ is considered aligned with a sample x if $a_{1:K}^*$ matches the ground-truth attribute assignments of x , i.e., $a_k^* \in \{j \in [q_k] \mid g_k^j(x) > 0\}$, $k \in [K]$. The alignment rate is defined as the proportion of aligned MPEs among all correctly predicted samples.*

A high alignment rate reflects strong model reliability, as it suggests that the ground-truth attribute assignments contribute the most during prediction. In other words, the model closely adheres to the human’s understanding when making predictions.

4.2 Counterfactual Explanations

To address the second question, we define Counterfactual Explanations (CEs) [Wachter et al., 2017] for NPCs to explore admissible changes in attribute recognition results that can correct any incorrectly predicted classes.

Definition 3 (Counterfactual Explanations). *Given an input x which has an incorrect model prediction. Denote $\mathbf{b} := \{\mathbf{b}_k\}_{k \in [K]}$, $\mathbf{b}_k := (\dots b_{k,a_k} \dots)_{a_k \in \mathcal{A}_k}^\top$, and $\Pr_{\mathbf{b}}(Y = y \mid X = x) := \sum_{a_{1:K}} \Pr_w(Y = y \mid A_{1:K} = a_{1:K}) \cdot \prod_{k=1}^K b_{k,a_k}$. The counterfactual explanation for the ground-truth label y is a set of probability vectors \mathbf{b} that maximizes $\Pr_{\mathbf{b}}(Y = y \mid X = x)$, i.e., the solution to the following optimization problem,*

$$\max_{\mathbf{b}} \Pr_{\mathbf{b}}(Y = y \mid X = x), \text{ s.t. } \sum_{a_k \in \mathcal{A}_k} b_{k,a_k} = 1 \ (0 \leq b_{k,a_k} \leq 1), \ \forall k \in [K]. \quad (4)$$

We adopt the projected gradient ascent algorithm to generate the CEs, which is detailed in Algorithm 1.

Algorithm 1 Generation of Counterfactual Explanations

- 1: **Input:** Feasible region $\mathcal{C} := \{\mathbf{b} : \sum_{a_k \in \mathcal{A}_k} b_{k,a_k} = 1 \ (0 \leq b_{k,a_k} \leq 1), \ \forall k \in [K]\}$, initial values $b_{k,a_k}^{(0)} := \Pr_{\theta}(A_k = a_k \mid X = x)$, $a_k \in \mathcal{A}_k$, $k \in [K]$, learning rate γ
- 2: **for** $t = 0, 1, \dots, T - 1$ **do**
 - # Computing $b_{k,a_k}^{(t+1)} := \mathcal{P}_{\mathcal{C}} \left\{ b_{k,a_k}^{(t)} + \gamma \frac{\partial \log \Pr_{\mathbf{b}^{(t)}}(Y=y|X=x)}{\partial b_{k,a_k}} \right\}$ [Wang and Carreira-Perpinán, 2013]
- 3: **for** $k = 1, \dots, K$ **do**
- 4: Sort $\{b_{k,a_k}^{(t)} + \gamma \frac{\partial \log \Pr_{\mathbf{b}^{(t)}}(Y=y|X=x)}{\partial b_{k,a_k}}\}_{a_k \in \mathcal{A}_k}$ into $u_{k,1} \geq u_{k,2} \geq \dots \geq u_{k,q_k}$
- 5: Find $\rho_k := \sum_{j=1}^{q_k} \mathbb{I} \left(u_{k,j} > \frac{1}{j} \left(\sum_{l=1}^j u_{k,l} - 1 \right) \right)$
- 6: Define $\lambda_k := \frac{1}{\rho_k} \left(1 - \sum_{j=1}^{\rho_k} u_{k,j} \right)$
- 7: Let $b_{k,a_k}^{(t+1)} := \max \left\{ b_{k,a_k}^{(t)} + \gamma \frac{\partial \log \Pr_{\mathbf{b}^{(t)}}(Y=y|X=x)}{\partial b_{k,a_k}} + \lambda_k, 0 \right\}$, $a_k \in \mathcal{A}_k$, $k \in [K]$
- 8: **Output:** Counterfactual explanations $\mathbf{b}^{(T)}$

CEs reveal the model’s inner workings by identifying the attribute recognition changes required to correct an incorrect prediction. Similar to MPES, these explanations deliver attribute-level insights into the model’s decision-making process, thereby improving interpretability.

Next, we introduce a metric to evaluate the effectiveness of CEs in correcting the model predictions.

Definition 4 (Correction Rate). *The correction rate is defined as the proportion of initially incorrectly predicted samples that are corrected by CEs among all initially incorrectly predicted samples.*

A high correction rate signifies that the generated CEs effectively correct model predictions by adjusting the attribute recognition results.

5 Experiments

5.1 Experimental Setup

Datasets We evaluate the model performance on a variety of benchmark datasets. **1) MNIST-Addition:** We derive this dataset from the original MNIST dataset [LeCun et al., 1998] by following the general preprocessing steps and procedures detailed in [Manhaeve et al., 2018]. Each MNIST-Addition sample consists of two images randomly selected from the original MNIST. The digits in these images, ranging from 0 to 9, represent two attributes, with their sum serving as the class label. A total of 35,000 samples are created for MNIST-Addition. **2) GTSRB:** GTSRB [Stallkamp et al., 2012] is a dataset comprising 39,209 images of German traffic signs, with class labels indicating the type of signs. Additionally, we annotate each sample with four attributes: “color”, “shape”, “symbol”, and “text”. The values for these attributes are detailed in Appendix D. **3) CelebA:** CelebA [Liu et al., 2015] consists of 202,599 celebrity face images annotated with 40 binary concepts. Here, we select the 8 most balanced binary concepts² and group them into 5 attributes: “mouth”, “face”, “cosmetic”, “hair”, and “appearance”. Following Zarlenga et al. [2022], each unique combination of concept values is treated as a group. To balance the dataset and increase its complexity, we rank these groups by the number of images they contain and pair them strategically: the group with the most images is merged with the one with the fewest, the second most with the second fewest, and so on. The above strategy results in 127 total classes. **4) AwA2:** AwA2 [Xian et al., 2018] contains 37,322 images of 50 types of animals, each annotated with 85 binary concepts. Certain concepts,

²Certain concepts are excluded due to political or ethical concerns.

Table 1: Model properties. “Interpretability” indicates whether the outputs produced by a concept/attribute recognition model are interpretable and whether humans can interpret the final decisions using these outputs. “Data-Driven Rules” denotes whether a model can incorporate logical rules learned from data. “Human-Predefined Rules” specifies whether a model can integrate logical rules predefined by humans. “Theoretical Guarantee” indicates whether a model provides a theoretical guarantee on the relationship between the performance of the overall model and that of its individual components.

Model	Interpretability	Data-Driven Rules	Human-Predefined Rules	Theoretical Guarantee
CBM [Koh et al., 2020]	✓	✗	✗	✗
CEM [Zarlenga et al., 2022]	✗	✗	✗	✗
DCR [Barbiero et al., 2023]	✓	✓	✗	✗
NPC (ours)	✓	✓	✓	✓

such as those describing non-visual properties (*e.g.*, “fast”, “domestic”) or indistinctive features (*e.g.*, “chewteeth”), or those representing background information (*e.g.*, “desert” and “forest”), are excluded. After the exclusion, 29 concepts remain, which are then grouped into 4 attributes: “color”, “surface”, “body”, and “limb”. The specific values for these attributes are provided in Appendix D. For all datasets, we split the samples into training, validation, and testing sets by a ratio of 8:1:1.

Baselines We select CBM [Koh et al., 2020] and several representative variants as baselines. Specifically, we choose CEM [Zarlenga et al., 2022], a method that uses high-dimensional concept embeddings as the bottleneck instead of concept probabilities, and DCR [Barbiero et al., 2023], which introduces a deep concept reasoner as the task predictor rather than relying on a simple linear layer. Additionally, we train an end-to-end DNN [He et al., 2016] as an additional baseline. It is important to note that CEM and the end-to-end DNN are not interpretable, as their components are not explicitly understandable by humans, even though they may achieve competitive performance on downstream tasks. A comparison of model properties is summarized in Table 1. Detailed descriptions of model architectures and training details are deferred to Appendix C.

Evaluation Metrics Considering the compositional nature of NPCs, we introduce distinct evaluation metrics for the various modules as well as the overall model. **1) Attribute Recognition Model:** We employ two metrics for evaluating the attribute recognition model. First, we propose the *mean total variation (TV) distance* between the output probability vectors from the attribute recognition model and the corresponding ground-truth probability vectors. The metric is defined as: $\frac{1}{K} \sum_k \frac{1}{|D_{\text{test}}|} \sum_{x \in D_{\text{test}}} d_{\text{TV}}(f_k(x; \theta), g_k(x))$, where a smaller distance indicates better performance. For baseline models producing concept probabilities, we group the outputs into attributes and process them as probability vectors. Second, we adapt the *mean concept accuracy* metric used in [Zarlenga et al., 2022] for NPCs. For an input x with n_k ground-truth values for the attribute A_k , we identify the top- n_k values in $f_k(x; \theta)$ and set them to one while the rest are set to zero. We then calculate the accuracy by comparing these processed outputs with the ground-truth concept labels. **2) Circuit-Based Task Predictor:** We use the *mean likelihood* to measure the performance of circuit-based task predictors, which is given by: $\frac{1}{|D_{\text{test}}|} \sum_{(y, a_{1:K}) \in \bar{D}_{\text{test}}} \Pr_w(Y = y, A_{1:K} = a_{1:K})$. A larger mean likelihood suggests that the circuit models the joint distribution more accurately. **3) Overall Model:** We adopt the *standard classification accuracy* as the evaluation metric for the overall model.

Table 2: Classification accuracy of NPCs and four baseline models on four benchmark datasets over five random seeds. “*” denotes uninterpretable models. The best results for each dataset are highlighted in bold, while the second-best results are underlined.

Model	MNIST-Add (%)	GTSRB (%)	CelebA (%)	AwA2 (%)
DNN*	99.057 ± 0.08	<u>99.939</u> ± 0.04	36.963 ± 0.72	93.351 ± 0.17
CBM	98.606 ± 0.03	99.810 ± 0.04	16.552 ± 0.87	82.286 ± 0.47
CEM*	98.740 ± 0.10	99.736 ± 0.06	25.218 ± 0.30	85.102 ± 0.27
DCR	94.597 ± 2.05	87.071 ± 6.93	7.055 ± 3.04	44.117 ± 10.05
NPC(Data)	<u>99.171</u> ± 0.11	99.888 ± 0.08	<u>33.739</u> ± 0.90	<u>87.281</u> ± 0.39
NPC(Knowledge)	99.189 ± 0.08	99.944 ± 0.04	31.727 ± 0.51	68.519 ± 3.54

5.2 NPCs vs. Baselines

We compare NPCs against baseline models across the four benchmark datasets, with the results summarized in Table 2. Specifically, we refer to the NPC whose circuit was learned using the data-driven approach as “NPC(Data)” and the NPC whose circuit was constructed using the knowledge-injected approach as “NPC(Knowledge)”.

The results in Table 2 demonstrate that NPCs outperform all other concept-based baseline models. Specifically, NPC(Knowledge) achieves the best performance on the MNIST-Addition and GTSRB datasets, while NPC(Data) leads on the CelebA and AwA2 datasets. Notably, NPCs achieve superior performance even compared to CEM, an uninterpretable model that relies on high-dimensional concept embeddings, highlighting NPC’s effectiveness in leveraging interpretable concept probabilities for downstream classification tasks.

Remarkably, NPCs achieve superior performance even compared to that of the end-to-end DNN, surpassing its classification accuracy on the MNIST-Addition and GTSRB datasets. Nevertheless, small gaps remain between the end-to-end DNN and NPC on more complex datasets like CelebA and AwA2. The above findings demonstrate that, while there is still small room for improvement compared to black-box models, NPCs strike a compelling balance between interpretability and task performance. Overall, the results underscore the remarkable potential of interpretable models, demonstrating their ability to achieve competitive performance in downstream tasks compared to baselines and even black-box end-to-end DNN models.

5.3 Ablation Studies

In this section, we delve into NPCs from additional perspectives. Specifically, we will analyze the integration of attributes, the influence of attribute selections, the effects of various approaches to constructing the task predictor, and, lastly, the impact of joint optimization.

5.3.1 Attributes vs. Concepts

Unlike existing concept-based models that utilize individual binary concepts (*e.g.*, “red color”, “yellow color”), NPCs focus on concept groups, *i.e.*, attributes (*e.g.*, “color”). Here, we aim to explore *the benefits of using attributes compared to individual concepts*. To this end, we replace the concept recognition model in CBM [Koh et al., 2020] with an attribute recognition model, resulting in a new model called the Attribute Bottleneck Model (ABM). ABM comprises an attribute recognition model and a linear layer serving as the task predictor. We employ the training loss from CBM to train ABM, replacing the concept loss with the attribute loss as defined in Equation (2). The performance comparison between CBM and ABM is presented in Table 3.

The results in Table 3 show that, in terms of the mean TV distance, ABM outperforms CBM on the MNIST-Addition and GTSRB datasets, albeit exhibiting slightly worse performance on the CelebA and AwA2 datasets. On the other hand, ABM consistently surpasses CBM in

Table 3: Comparison between CBM and ABM across four benchmark datasets in terms of mean TV distance and mean concept accuracy. For TV distance, lower values are better. For accuracy, higher values are better. The better values are in bold.

Metric	Model	MNIST-Add	GTSRB	CelebA	AwA2
Mean TV Distance	CBM	0.0080	0.0023	0.1744	0.0514
	ABM	0.0058	0.0007	0.1759	0.0775
Mean Concept Accuracy	CBM	98.64%	99.80%	32.52%	82.21%
	ABM	98.99%	99.84%	45.00%	89.69%

mean concept accuracy across all datasets. These findings underscore the effectiveness of the attribute recognition model, indicating that attributes capture more nuanced information. In particular, we hypothesize that inherent relationships may exist both within the values of any single attribute and across different attributes. Therefore, treating all values as independent concepts neglects these interdependencies, potentially leading to inferior performance. Overall, these results suggest the benefits of utilizing attributes in preserving relational constraints within predictions and thus improving model performance.

5.3.2 Impact of Attribute Selection

During inference, an NPC utilizes sufficient attributes to produce final predictions. Here, we aim to investigate the following questions: *How does the exclusion of one particular attribute during inference impact the performance of NPC on downstream tasks? How does the exclusion of different attributes vary the impact?*

Assuming the excluded attribute is A_1 , the predicted score for the class y would become $\sum_{a_{2:K}} \Pr_w(Y = y | A_{2:K} = a_{2:K}) \cdot \prod_{k=2}^K \Pr_\theta(A_k = a_k | X)$. We apply this inference process on the GTSRB dataset and the resulting task performance is presented in Figure 3 (Left).

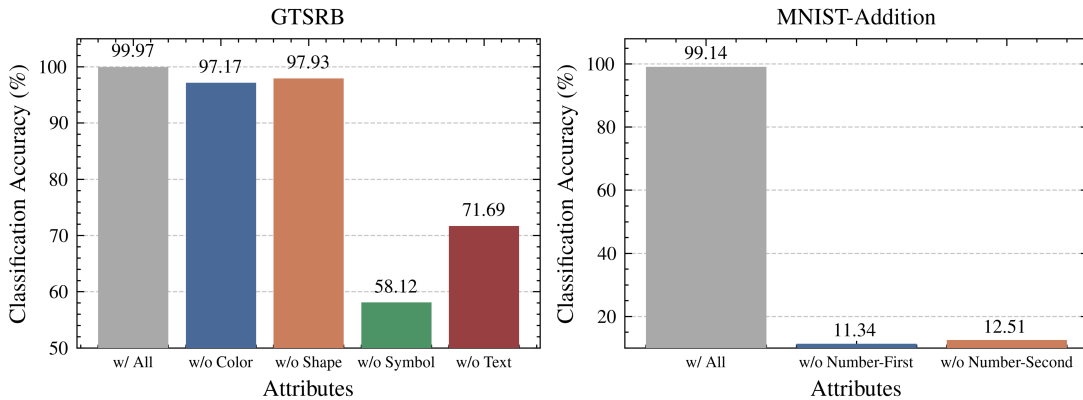


Figure 3: NPC classification accuracy on the GTSRB and MNIST-Addition datasets with attribute exclusions. The gray bars indicate inference with all attributes, while non-gray bars indicate inference with one particular attribute excluded.

We observe that, when NPC has one fewer attribute at its disposal during inference, the classification accuracy decreases, due to the fact that the assumption of sufficient attributes (Assumption 1) no longer holds in this scenario, and that the equality in Equation (1) was violated. Therefore, the formula above does not correctly represent $\Pr_{\theta,w}(Y = y | X)$. As a result, relying on the above formula for inference can adversely affect the predictions.

On the other hand, we observe that excluding different attributes can lead to varying impacts on the task performance. Specifically, excluding the “color” or “shape” attributes results in

only a slight drop in accuracy, whereas excluding the “symbol” or “text” attributes leads to a significant decline. We accredit such discrepancy to the distinct properties of these attributes. More specifically, attributes such as “color” and “shape” are generally not decisive, meaning that they do not decisively determine the final class, and that their absence can be compensated by leveraging information from the other attributes. For instance, even without having any indication of “red” or “octagon”, it is still possible to infer an input representing a stop sign so long as the “text” attribute indicates “stop”. Thus, excluding the non-decisive attributes has minimal impact on the performance. In contrast, “symbol” and “text” attributes are decisive for many instances and thus are crucial for distinguishing between certain classes. For example, without the “text” attribute, it then becomes impossible to differentiate speed limit signs indicating different speeds. Similarly, without the “symbol” attribute, distinguishing between a left-curve and a right-curve sign is no longer feasible. Hence, excluding decisive attributes can severely impair predictions.

For the MNIST-Addition dataset, Figure 3 (Right) demonstrates similar findings. In particular, as both attributes are essential for determining the final class, *i.e.*, the sum, the removal of either attribute results in a substantial performance decline.

In summary, utilizing insufficient attributes compromises NPC’s performance on downstream tasks, and the impact of excluding different attributes varies depending on the properties of the attributes themselves.

5.3.3 Impact of Task Predictor Construction Approaches

In Section 3.2.2, we introduce two distinct approaches for constructing probabilistic circuits: the data-driven approach and the knowledge-injected approach. Here, our objective is to investigate the impact of these construction methods. Specifically, we aim to address the following questions: *Which approach produces a circuit that better captures the data distribution? Which approach produces a circuit that performs more effectively as a task predictor?* We start by comparing the mean likelihood of the two circuits. Then, we examine the classification accuracy of the overall models consisting of a well-trained attribute recognition model together with either circuit (data driven or knowledge injected). The results are summarized in Table 4.

Table 4: Performance comparison between circuits constructed using data-driven and knowledge-injected approaches, along with the classification accuracy of overall models combining either circuit with a well-trained attribute recognition model. The better results are in bold.

Metric	Architecture	MNIST-Add	GTSRB	CelebA	AwA2
Mean Likelihood	Circuit (Data)	1.010e-2	3.663e-02	2.082e-02	1.263e-04
	Circuit (Knowledge)	1.007e-2	3.663e-02	2.091e-02	1.255e-04
Accuracy	Model (Data)	99.17%	99.85%	32.74%	68.52%
	Model (Knowledge)	99.17%	99.85%	31.56%	23.47%

In terms of classification accuracy, the models combining the different circuits exhibit similar performance on MNIST-Addition, GTSRB, and CelebA datasets. Such similarity suggests that the two-layered circuit constructed using the knowledge-injected approach is sufficient to provide accurate relational information among attributes and classes for simpler datasets.

In contrast, for the more complex AwA2 dataset, Model (Data) outperforms Model (Knowledge) by a wide margin due to the presence of multi-valued attributes in AwA2, which results in a large number of valid combinations of attribute values. Consequently, each combination may correspond to a very small joint probability. Under these circumstances, even a slight difference in mean likelihood may induce a significant change in the circuit’s ability to capture the data distribution. For instance, even a marginally lower mean likelihood might indicate the failure of the circuit to capture the joint probabilities of a large quantity of combinations. As shown in

Table 4, the mean likelihood of Circuit (Knowledge) is slightly lower than that of Circuit (Data) for the AWA2 dataset, suggesting that, in this case, the knowledge-injected circuit might have failed to capture sufficient nuanced relationships among attributes and classes, while the data-driven circuit is better suited for representing the joint distribution of the data, thus leading to a better performance on the downstream task.

5.3.4 Impact of Joint Optimization

When training the NPCs, we adopt a three-stage training algorithm, where we first independently train the attribute recognition model and the task predictor, followed by a joint optimization for the overall model. Here, we aim to investigate *how the third stage, i.e., joint optimization, affects the performance of NPCs*. To this end, we compare the performance of NPCs before and after applying the joint optimization. The comparison is illustrated in Figure 4.

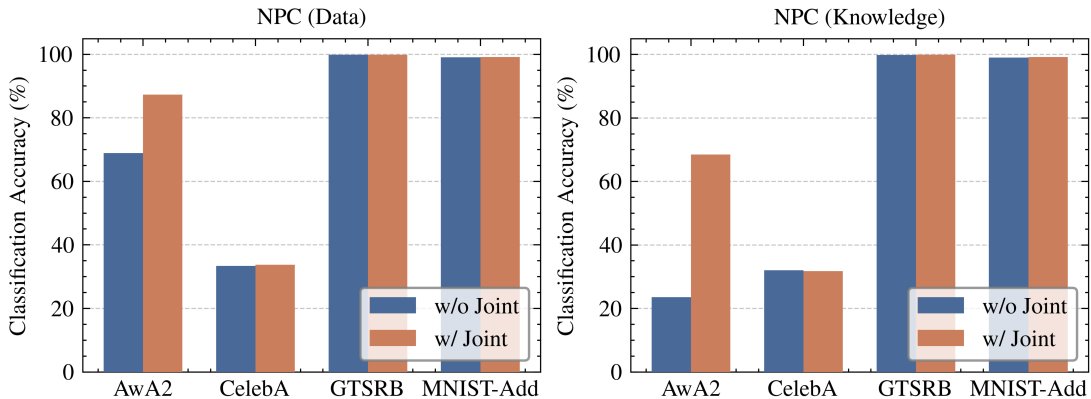


Figure 4: Classification accuracy of NPC(Data) and NPC(Knowledge) on the four benchmark datasets. The blue bars indicate performance prior to joint optimization, while the orange bars illustrate performance after applying joint optimization.

In general, joint optimization enhances model performance across the various datasets. Specifically, for the AWA2 dataset, joint optimization leads to substantial improvements for both NPC(Data) and NPC(Knowledge), demonstrating its effectiveness. In contrast, for the CelebA dataset, the performance stays roughly the same after applying joint optimization, with a marginal increase in performance for NPC(Data) and a marginal decrease in performance only for NPC(Knowledge). Nevertheless, slight performance improvements are generally observed on the GTSRB and MNIST-Addition datasets for both NPCs. Overall, these results suggest that joint optimization provides additional benefits for NPCs in terms of downstream tasks. In particular, for datasets with superior performance after the initial training, joint optimization provides additional but modest gains. In contrast, for datasets with moderate performance after the initial training, joint optimization plays a pivotal role in delivering significant performance improvements.

5.4 Model Explanations

In this section, we explore two types of explanations and provide examples to illustrate how the explanations facilitate the human’s understanding of NPC’s inner workings and interpret the model’s behavior.

5.4.1 Most Probable Explanations

Figure 5 presents some examples for NPC(Data) from the four benchmark datasets. Specifically, each example comprises an image, the ground-truth labels for the class and attributes, the class predicted by NPC(Data), and, lastly, the corresponding MPE which accounts for the attribute assignment that contributes most significantly to the prediction. In these instances, NPC(Data) provides correct class predictions, and the MPEs are aligned with the ground-truth attribute labels. For instance, the MPE for the example from GTSRB is {Color: Red; Shape: Circle; Symbol: Text; Text: 30}, which precisely matches the ground-truth attribute labels. Such alignment between MPEs and attribute labels indicates that the model employs human-like reasoning and makes reliable decisions. Examples for NPC(Knowledge) are deferred to Appendix F.





MNIST-Add		Class Label: 6 Attribute Labels: { <u>Number-First:</u> 6; <u>Number-Second:</u> 0} Class Prediction: 6 MPE: { <u>Number-First:</u> 6; <u>Number-Second:</u> 0}
GTSRB		Class Label: Regulatory-Maximum-Speed-Limit-30 Attribute Labels: { <u>Color:</u> Red; <u>Shape:</u> Circle; <u>Symbol:</u> Text; <u>Text:</u> 30} Class Prediction: Regulatory-Maximum-Speed-Limit-30 MPE: { <u>Color:</u> Red; <u>Shape:</u> Circle; <u>Symbol:</u> Text; <u>Text:</u> 30}
CelebA		Class Label: Group 5 Attribute Labels: { <u>Mouth:</u> Mouth-Slightly-Open, Smiling; <u>Face:</u> High-Cheekbones; <u>Cosmetic:</u> Heavy-Makeup, Wearing-Lipstick; <u>Hair:</u> Wavy-Hair; <u>Appearance:</u> Attractive} Class Prediction: Group 5 MPE: { <u>Mouth:</u> Mouth-Slightly-Open; <u>Face:</u> High-Cheekbones; <u>Cosmetic:</u> Heavy-Makeup; <u>Hair:</u> Wavy-Hair; <u>Appearance:</u> Attractive}
Awa2		Class Label: Zebra Attribute Labels: { <u>Color:</u> Black, White; <u>Surface:</u> Furry, Stripes, Toughskin; <u>Body:</u> Lean, Longleg, Longneck, Quadrapedal, Tail; <u>Limb:</u> Hooves} Class Prediction: Zebra MPE: { <u>Color:</u> White; <u>Surface:</u> Stripes; <u>Body:</u> Longleg; <u>Limb:</u> Hooves}

Figure 5: Examples from the four benchmark datasets. Each example includes an image, the ground-truth class label, the ground-truth attribute labels, the NPC(Data) class prediction, and, lastly, the corresponding MPE. These examples illustrate MPEs that align with the ground-truth attribute labels. The CelebA image is redacted in compliance with its terms of use.

The MPEs alignment rates are shown in Figure 6 (Left). We observe that NPC(Knowledge) exhibits a relatively low alignment rate on the Awa2 dataset, suggesting that the ground-truth attribute assignment does not contribute the most to the correct predictions for certain instances. Such misalignment could be caused by the knowledge-injected approach failing to capture the relatively more complex conditional dependencies between classes and attributes for this particular dataset, thereby affecting the model’s prediction process. On the other hand, the MPEs alignment rates in the other scenarios are close to 100%, indicating that when it makes correct predictions, the model predominantly relies on attribute assignments matching the ground truths. Therefore, the model is considered reliable as its prediction process properly aligns with the human’s decision-making process.

5.4.2 Counterfactual Explanations

Figure 7 illustrates some examples for NPC(Data) from the four benchmark datasets. Each sample consists of an image, the attribute and class predictions, the generated CE, and, lastly,

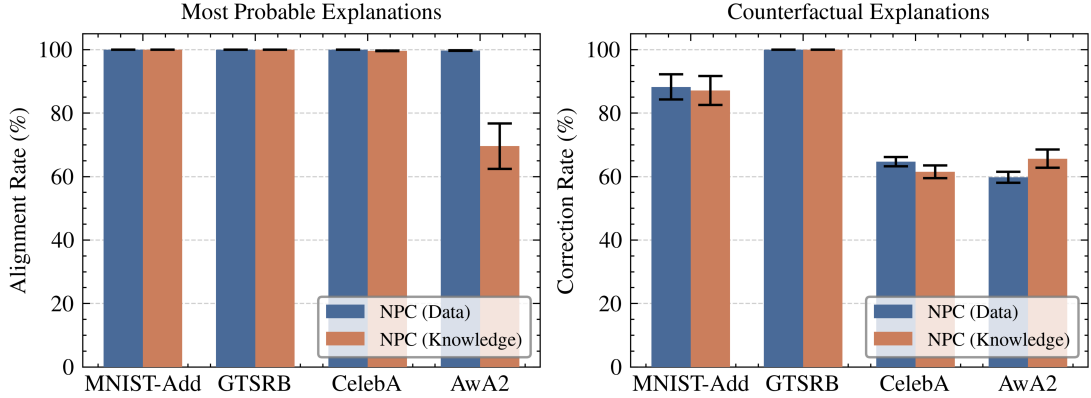


Figure 6: MPE alignment rate and CE correction rate for NPCs across the four benchmark datasets. The blue bars represent NPC(Data), while the orange bars represent NPC(Knowledge). Error bars indicate the standard deviation over five random seeds.

the class prediction corrected by the CE. In these examples, NPC(Data) incorrectly predicts the class, while CEs effectively correct them by making minimal adjustments to the attribute predictions. For instance, for the MNIST-Addition dataset, changing the “Number-Second” attribute from “4” to “9” corrects the class prediction from “11” to “16”. Similarly, for the GTSRB dataset, the predicted class is corrected by adjusting the “Text” attribute prediction. For the CelebA dataset, the CE yields the correct class prediction primarily by increasing the probability of “Attractive” from the “Appearance” attribute. Finally, for the AwA2 dataset, the CE corrects the class prediction from “Horse” to “Deer” primarily by increasing the probability of “Brown” for the “Color” attribute. Examples for NPC(Knowledge) are provided in Appendix F.

MNIST-Add		Attribute Predictions: { <u>Number-First:</u> 7 (100.0%); <u>Number-Second:</u> 4 (82.4%)} Class Prediction: 11 CE: { <u>Number-First:</u> 7 (100.0%); <u>Number-Second:</u> 9 (50.3%)} Corrected Class Prediction: 16
GTSRB		Attribute Predictions: { <u>Color:</u> Red (100%); <u>Shape:</u> Circle (100.0%); <u>Symbol:</u> Text (100.0%); <u>Text:</u> 120 (53.5%)} Class Prediction: Regulatory-Maximum-Speed-Limit-120 CE: { <u>Color:</u> Red (100%); <u>Shape:</u> Circle (100.0%); <u>Symbol:</u> Text (100.0%); <u>Text:</u> 80 (50.4%)} Corrected Class Prediction: Regulatory-Maximum-Speed-Limit-80
CelebA		Attribute Predictions: { <u>Mouth:</u> Mouth-Slightly-Open (100.0%); <u>Face:</u> High-Cheekbones (100.0%); <u>Cosmetic:</u> Wearing-Lipstick (68.0%); <u>Hair:</u> None (100.0%); <u>Appearance:</u> None (88.9%)} Class Prediction: Group 23 CE: { <u>Mouth:</u> Mouth-Slightly-Open (100.0%); <u>Face:</u> High-Cheekbones (100.0%); <u>Cosmetic:</u> Wearing-Lipstick (66.0%); <u>Hair:</u> None (100.0%); <u>Appearance:</u> Attractive (54.8%)} Corrected Class Prediction: Group 7
AwA2		Attribute Predictions: { <u>Color:</u> Gray (55.8%); <u>Surface:</u> Furry (99.1%); <u>Body:</u> Longleg (95.9%); <u>Limb:</u> Hooves (90.4%)} Class Prediction: Horse CE: { <u>Color:</u> Brown (63.5%); <u>Surface:</u> Furry (100.0%); <u>Body:</u> Longleg (96.9%); <u>Limb:</u> Hooves (100.0%)} Corrected Class Prediction: Deer

Figure 7: Examples from the four benchmark datasets. Each example includes an image, the NPC(Data) attribute and class predictions, the corresponding CE, and the class prediction corrected by CE. These examples focus on CEs that successfully correct the class predictions. For simplicity, both attribute predictions and CEs display only the most confident value for each attribute. The CelebA image is redacted in compliance with its terms of use.

The CE correction rates are presented in Figure 6 (Right). For simpler datasets like MNIST-Addition and GTSRB, the generated CEs exhibit high correction rates, indicating their effec-

tiveness in correcting predictions. However, for more complex datasets such as CelebA and AwA2, the correction rates are lower, highlighting the limit of the efficacy of the generated CEs on complex datasets. Such a limit underscores the future need for more advanced algorithms for generating CEs that can correct misclassifications with excellent correction rates, even for complex datasets.

6 Limitations and Discussions

In this section, we discuss the limitations of NPCs from multiple perspectives, highlighting potential future directions for improvement.

Model Architecture Compared to end-to-end DNNs, NPCs offer superior interpretability by decomposing a model into semantically meaningful modules, enabling humans to combine module outputs to understand the final decisions. Nevertheless, the attribute recognition model itself remains a black box, and its opaque inner workings make it difficult to ensure that its outputs truly represent the probabilities for the various attributes. For instance, the model might learn spurious correlations and incorrectly map background features, instead of actual attributes, to outputs. Future work may focus on increasing the transparency within the attribute recognition model, thereby enhancing its interpretability.

Structure of Probabilistic Circuits In NPCs, the task predictor, implemented using a probabilistic circuit, is either learned using LearnSPN [Gens and Domingos, 2013] or manually constructed based on human-predefined rules. The circuit generated by LearnSPN, however, may contain an excessive number of nodes and edges, resulting in a slower inference. Alternative methods [Vergari et al., 2015, Mauro et al., 2017] may be explored to create more compact circuits for added inference efficiency. On the other hand, manually constructed circuits employ simpler structures with only two layers. While it may improve efficiency, such simplicity may limit the circuit’s expressiveness, potentially degrading its performance on complex datasets like AwA2. Future work may focus on improved balancing between circuit expressiveness and structural complexity.

Assumption of Complete Information The assumption of complete information in this paper assumes that attributes are conditionally independent given the input. However, recent studies suggest that this assumption limits model expressiveness [van Krieken et al., 2024] and potentially introduces spurious correlations, also known as reasoning shortcuts [Marconato et al., 2023, Bortolotti et al., 2024]. Investigating whether this issue arises in NPCs and identifying ways to address it would be a valuable direction for future research.

Reducing Trade-Offs between Interpretability and Task Performance In this paper, we show that, with the integration of attribute recognition and probabilistic circuit, NPC produces interpretable predictions for downstream tasks while achieving superior performance. Looking ahead, we believe that, by incorporating more fine-grained and diverse attributes that are semantically meaningful, along with a structure that reasons over these attributes using logical rules with increased complexity, we shall devise compositional model designs that further reduce the trade-offs between interpretability and performance of downstream tasks.

7 Related Work

In this section, we discuss several areas of research relevant to our proposed method.

Concept Bottleneck Models and Variants Concept bottleneck models (CBMs) and their variants are a class of machine learning models that organize their decision-making process around high-level, human-understandable concepts, offering enhanced transparency. First introduced by Koh et al. [2020], CBMs decompose a black-box DNN into two modules: a concept recognition model, responsible for predicting various human-specified concepts, and a task predictor, which performs classifications on the predicted concepts.

Subsequent research has focused on improving these two modules. Zarlenga et al. [2022], Yeh et al. [2020], Kazhdan et al. [2020] extend the concept recognition model by representing concepts as high-dimensional embeddings rather than simple probabilities. Additionally, Mahinpei et al. [2021], Sawada and Nakamura [2022], Sarkar et al. [2022], Marconato et al. [2022] introduce unsupervised neurons into the bottleneck to enhance the model’s learning capacity. While improving the performance on downstream tasks, these extensions compromise interpretability, as the dimensions within concept embeddings and the unsupervised neurons lack explicit semantic meanings. In contrast, utilizing predicted concept probabilities gives better interpretability. On the other hand, there have been recent efforts to improve the interpretability of the task predictor. Instead of using a linear layer as the task predictor, several approaches [Barbiero et al., 2023, Ciravegna et al., 2023, Rodríguez et al., 2024] design new architectures to embed logical rules and enable classifications via reasoning. For instance, Barbiero et al. [2023] propose a deep concept reasoner, while Rodríguez et al. [2024] introduce a soft decision tree as the task predictor. These approaches optimize their parameters using observed data, thus extracting the underlying logical rules inherent within the data. In comparison, architectures that directly encode human-predefined logical rules through their structure and parameters offer means to explicitly represent domain knowledge.

Probabilistic Circuits Probabilistic circuits [Sánchez-Cauce et al., 2022] are rooted directed acyclic graphs designed to represent the joint distribution of a set of variables. The circuits comprise three types of nodes: 1) leaf nodes, which correspond to input variables; 2) sum nodes, which compute weighted sums of their child nodes; and 3) product nodes, which compute products of their child nodes. When satisfying the properties of decomposability and smoothness, a probabilistic circuit becomes a tractable probabilistic model, ensuring efficient inferences over various distributions [Poon and Domingos, 2011]. Specifically, joint, marginal, and conditional probabilities of input variables can be computed in at most two passes (from leaf nodes to the root node), with computational complexity linear in the size of the circuit. Consequently, probabilistic circuits combine the expressiveness of traditional graphical models with the scalability of modern deep learning frameworks.

Structure learning for probabilistic circuits aims to design structures that effectively balance expressiveness and computational efficiency. Xia et al. [2023] categorize existing structure learning methods into four types: 1) handcrafted structure learning, where structures are manually designed for specific datasets [Gens and Domingos, 2012, Poon and Domingos, 2011]; 2) data-based structure learning, which uses heuristic [Adel et al., 2015, Dennis and Ventura, 2012, Gens and Domingos, 2013, Krakovna and Looks, 2016, Molina et al., 2018, Rahman and Gogate, 2016, Rooshenas and Lowd, 2014, Vergari et al., 2015] or non-heuristic algorithms [Peharz et al., 2014, Lee et al., 2014, Trapp et al., 2016, Peharz et al., 2019] to learn structures from data; 3) random structure learning, where structures are randomly generated as a flexible starting point [Peharz et al., 2019, Rashwan et al., 2016, Trapp et al., 2019]; and 4) ensemble structure learning, which combines multiple structures to improve generalization to high-dimensional data [Ventola et al., 2020]. In this paper, we utilize the first and second types of structure learning approaches to embed explicit and implicit logical rules, respectively.

Parameter learning for probabilistic circuits involves finding optimal parameters for a given structure, enabling the circuits to accurately capture the underlying probability distributions within the observed data. Parameter learning can be broadly categorized into two types: generative and discriminative. Generative parameter learning [Poon and Domingos, 2011, Peharz, 2015, Rashwan et al., 2016, Zhao et al., 2016a,b], the most common paradigm, aims to maximize the joint probabilities of all variables. The generative approach is particularly suited for

tasks such as density estimation, generative modeling, and probabilistic reasoning. In contrast, discriminative parameter learning [Gens and Domingos, 2012, Adel et al., 2015, Rashwan et al., 2018] focuses on maximizing the conditional probabilities of a class variable given other variables, making it ideal for classification and regression tasks. In this paper, we adopt CCCP [Zhao et al., 2016b], a generative parameter learning approach, as it admits multiplicative parameter updates that provide a monotonic increase in the log-likelihood and lead to faster and more stable convergence.

On the other hand, some works develop probabilistic circuits parameterized by neural networks. For instance, Ahmed et al. [2022] use an amortized neural network to output the weights in circuits. To address the overfitting problem of large probabilistic circuits, Shih et al. [2021] exploit the generalization ability of a small-scale neural network and employ it to generate the parameters of a large circuit. Shao et al. [2022] aims to learn a probabilistic circuit to model the conditional distribution of target variables given input variables. In their approach, a neural network, conditioned on the input variables, is utilized to generate the circuit’s parameters. This integration of neural networks enhances the expressiveness of probabilistic circuits and has been used in neuro-symbolic integration [Ahmed et al., 2022, Manhaeve et al., 2018]. In this paper, the circuit’s integration with the attribute recognition model can be seen as parameterizing the input distributions of the circuit, instead of the weights of the circuit. The final predictions are a result of utilizing the outputs of a circuit through the law of total probability.

Integration of Probabilistic Graphical Models Probabilistic graphical models (PGMs) are frameworks that use graphs to express conditional dependencies between variables and represent their joint probability distributions. With their expressiveness, PGMs can be integrated into the decision-making process to enhance models from various perspectives. Our work demonstrates one approach to integrating PGMs, *i.e.*, probabilistic circuits, to improve the transparency and interpretability of a model’s predictive process. In contrast, Yang et al. [2022], Gürel et al. [2021], Zhang et al. [2023], Kang et al. [2024] have focused on leveraging PGMs integration to enhance the adversarial robustness of deep classification models.

Neuro-Symbolic Learning Neuro-symbolic learning integrates neural networks with symbolic representations, combining data-driven learning with symbolic reasoning to leverage the strengths of both. Variants of CBMs that embed inherent rules within the task predictor serve as one exemplifying application of neuro-symbolic learning. Beyond CBMs, this neuro-symbolic paradigm can be implemented in various other forms. One line of research focuses on designing symbolic-based objective functions. For instance, Badreddine et al. [2022] propose objectives that maximize the satisfiability of predefined symbolic rules over a neural network’s outputs. Similarly, Xu et al. [2018], Ahmed et al. [2023] define objectives that maximize the probabilities of generating outputs aligned with symbolic rules. These objectives can also function as regularization terms alongside standard classification losses, encouraging a neural network to adhere to specific rules by optimizing its parameters accordingly. Another line of research emphasizes the design of model architectures. For example, Ahmed et al. [2022] introduce a semantic probabilistic layer, a predictive layer designed for structured-output predictions, which can be seamlessly integrated into neural networks to ensure predictions align with certain symbolic constraints. Overall, these works ensure that learned models follow specific symbolic rules, either through the design of objective functions or the modification of model architectures. However, while these approaches enforce rule compliance, the explicit meaning of individual model components often remains unclear, raising concerns about their transparency and interoperability.

Knowledge Compilation In the field of knowledge compilation [Darwiche and Marquis, 2002], previous studies have made efforts to compile PGMs [Choi et al., 2013, Chavira and Darwiche, 2005] or logical formulas [Chavira and Darwiche, 2005, Darwiche, 2002, Pipatsrisawat and Darwiche, 2008] into computationally efficient structures that support various inference tasks. In this paper, the proposed knowledge-injected parameter learning approach adopts a simple

compilation strategy that compiles a set of weighted AND rules into a two-layer probabilistic circuit. Specifically, each product node corresponds to an individual AND rule, while the sum node encodes the weights associated with these rules.

8 Conclusions

In this paper, we propose Neural Probabilistic Circuits (NPCs), a novel architecture that decomposes the decision-making process into attribute recognition and logical reasoning, enabling compositional and interpretable predictions. Experimental results across four image classification datasets demonstrate that NPC delivers competitive performance when compared with four baseline models. In addition, we conduct a series of ablation studies and observe the following findings: 1) Compared with individual binary concepts, utilizing attributes preserves relational constraints, thus enhancing model performance. 2) Insufficient attributes compromise NPC’s performance on downstream tasks, and the impact of excluding specific attributes varies depending on their inherent properties. 3) For simple datasets, manually constructed circuits using the knowledge-injected approach are sufficient to result in competitive performance, whereas data-driven circuits are more effective on complex datasets. 4) Joint optimization leads to further improvements in the model performance. Furthermore, we demonstrate how the provided most probable explanations and counterfactual explanations offer attribute-level insights into the model’s decision-making process, thereby enhancing the human’s understanding. Finally, we discuss the limitations and potential future research directions for NPCs from various perspectives, providing insights for future improvements. Our work demonstrates the potential of NPCs to enhance model interpretability and performance by integrating semantically meaningful attributes with probabilistic circuits, offering actionable insights for future advancements in interpretable machine learning through logical reasoning.

Acknowledgments We would like to extend our gratitude to Antonio Vergari for pointing out relevant literature on probabilistic circuits and knowledge compilation, and for the discussion on the relationship between NPCs and semantic probabilistic layers.

References

- Tameem Adel, David Balduzzi, and Ali Ghodsi. Learning the structure of sum-product networks via an svd-based algorithm. In *UAI*, 2015.
- Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, and Antonio Vergari. Semantic probabilistic layers for neuro-symbolic learning. In *NeurIPS*, 2022.
- Kareem Ahmed, Kai-Wei Chang, and Guy Van den Broeck. A pseudo-semantic loss for autoregressive models with logical constraints. In *NeurIPS*, 2023.
- David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.
- Samy Badreddine, Artur S. d’Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artif. Intell.*, 2022.
- Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Pietro Lió, Marco Gori, and Stefano Melacci. Entropy-based logic explanations of neural networks. In *AAAI*, 2022.
- Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Mateo Espinosa Zarlenga, Lucie Charlotte Magister, Alberto Tonda, Pietro Lio, Frédéric Precioso, Mateja Jamnik, and Giuseppe Marra. Interpretable neural-symbolic concept reasoning. In *ICML*, 2023.

- Samuele Bortolotti, Emanuele Marconato, Tommaso Carraro, Paolo Morettin, Emile van Krieken, Antonio Vergari, Stefano Teso, and Andrea Passerini. A neuro-symbolic benchmark suite for concept quality and reasoning shortcuts. In *NeurIPS Track on Datasets and Benchmarks*, 2024.
- Rich Caruana. Multitask learning. *Machine learning*, 1997.
- Mark Chavira and Adnan Darwiche. Compiling bayesian networks with local structure. In *IJCAI*, 2005.
- Arthur Choi, Doga Kisa, and Adnan Darwiche. Compiling probabilistic graphical models using sentential decision diagrams. In *ECSQARU*, 2013.
- Y Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. *UCLA*. URL: <http://starai.cs.ucla.edu/papers/Prob-Circ20.pdf>, page 6, 2020.
- Gabriele Ciravegna, Pietro Barbiero, Francesco Giannini, Marco Gori, Pietro Liò, Marco Maggini, and Stefano Melacci. Logic explained networks. *Artif. Intell.*, 2023.
- Adnan Darwiche. A compiler for deterministic, decomposable negation normal form. In *AAAI*, 2002.
- Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.
- Ashley Deeks. The judicial demand for explainable artificial intelligence. *Columbia Law Review*, 119(7):1829–1850, 2019.
- Aaron Dennis and Dan Ventura. Learning the architecture of sum-product networks using clustering on variables. In *NeurIPS*, 2012.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.
- Robert Gens and Pedro M. Domingos. Discriminative learning of sum-product networks. In *NeurIPS*, 2012.
- Robert Gens and Pedro M. Domingos. Learning the structure of sum-product networks. In *ICML*, 2013.
- Emilie Grégoire, Muhammad Hafeez Chaudhary, and Sam Verboven. Sample-level weighting for multi-task learning with auxiliary tasks. *Applied Intelligence*, 2024.
- Nezihe Merve Gürel, Xiangyu Qi, Luka Rimanic, Ce Zhang, and Bo Li. Knowledge enhanced machine learning pipeline against diverse adversarial attacks. In *ICML*, 2021.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 2012.
- Junlin Hou, Sicen Liu, Yequan Bie, Hongmei Wang, Andong Tan, Luyang Luo, and Hao Chen. Self-explainable ai for medical image analysis: A survey and new outlooks. *arXiv preprint arXiv:2410.02331*, 2024.

- Mintong Kang, Nezihe Merve Gürel, Linyi Li, and Bo Li. Colep: Certifiably robust learning-reasoning conformal prediction via probabilistic circuits. In *ICLR*, 2024.
- Dmitry Kazhdan, Botty Dimanov, Mateja Jamnik, Pietro Liò, and Adrian Weller. Now you see me (CME): concept-based model extraction. In *CIKM (Workshops)*, 2020.
- Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.
- Eunji Kim, Dahuin Jung, Sangha Park, Siwon Kim, and Sungroh Yoon. Probabilistic concept bottleneck models. In *ICML*, 2023.
- Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *ICML*, 2020.
- Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- Viktoriya Krakovna and Moshe Looks. A minimalistic approach to sum-product network learning for real applications. *arXiv preprint arXiv:1602.04259*, 2016.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.
- Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, and Marcin Detyniecki. Issues with post-hoc counterfactual explanations: a discussion. *arXiv preprint arXiv:1906.04774*, 2019.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.
- Sang-Woo Lee, Christopher Watkins, and Byoung-Tak Zhang. Non-parametric bayesian sum-product networks. In *ICML Workshop on Learning Tractable Probabilistic Models*, 2014.
- Wei Liu, Feng Zhao, Achyut Shankar, Carsten Maple, J Dinesh Peter, Byung-Gyu Kim, Adam Slowik, BD Parameshachari, and Jianhui Lv. Explainable ai for medical image analysis in medical cyber-physical systems: Enhancing transparency and trustworthiness of iomt. *IEEE Journal of Biomedical and Health Informatics*, 2023.
- Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *CVPR*, 2015.
- Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
- Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*, 2017.
- Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models. *arXiv preprint arXiv:2106.13314*, 2021.
- Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *NeurIPS*, 2018.
- Emanuele Marconato, Andrea Passerini, and Stefano Teso. Glancenets: Interpretable, leak-proof concept-based models. In *NeurIPS*, 2022.
- Emanuele Marconato, Stefano Teso, Antonio Vergari, and Andrea Passerini. Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts. In *NeurIPS*, 2023.

- Nicola Di Mauro, Floriana Esposito, Fabrizio Giuseppe Ventola, and Antonio Vergari. Alternative variable splitting methods to learn sum-product networks. In *AI*IA*, 2017.
- Alejandro Molina, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting. Mixed sum-product networks: A deep architecture for hybrid domains. In *AAAI*, 2018.
- Robert Peharz. *Foundations of sum-product networks for probabilistic modeling*. PhD thesis, PhD thesis, Medical University of Graz, 2015.
- Robert Peharz, Robert Gens, and Pedro Domingos. Learning selective sum-product networks. In *ICML*, 2014.
- Robert Peharz, Sebastian Tschiatschek, Franz Pernkopf, and Pedro M. Domingos. On theoretical properties of sum-product networks. In *AISTATS*, 2015.
- Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Xiaoting Shao, Kristian Kersting, and Zoubin Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *UAI*, 2019.
- Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In *AAAI*, 2008.
- Hoifung Poon and Pedro M. Domingos. Sum-product networks: A new deep architecture. In *UAI*, 2011.
- Tahrira Rahman and Vibhav Gogate. Merging strategies for sum-product networks: From trees to graphs. In *UAI*, 2016.
- Abdullah Rashwan, Han Zhao, and Pascal Poupart. Online and distributed bayesian moment matching for parameter learning in sum-product networks. In *AISTATS*, 2016.
- Abdullah Rashwan, Pascal Poupart, and Zhitang Chen. Discriminative training of sum-product networks by extended baum-welch. In *PGM*, 2018.
- Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *KDD*, 2016.
- Karen McGregor Richmond, Satya M Muddamsetty, Thomas Gammeltoft-Hansen, Henrik Palmer Olsen, and Thomas B Moeslund. Explainable ai and law: an evidential survey. *Digital Society*, 3(1):1, 2024.
- David M. Rodríguez, Manuel P. Cuéllar, and Diego Pedro Morales. Concept logic trees: enabling user interaction for transparent image classification and human-in-the-loop learning. *Appl. Intell.*, 2024.
- Amirmohammad Rooshenas and Daniel Lowd. Learning sum-product networks with direct and indirect variable interactions. In *ICML*, 2014.
- S Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.
- Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 2019.
- Raquel Sánchez-Cauce, Iago París, and Francisco Javier Díez. Sum-product networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3821–3839, 2021.

- Raquel Sánchez-Cauce, Iago París, and Francisco Javier Díez. Sum-product networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.
- Anirban Sarkar, Deepak Vijaykeerthy, Anindya Sarkar, and Vineeth N. Balasubramanian. A framework for learning ante-hoc explainable models via concepts. In *CVPR*, 2022.
- Yoshihide Sawada and Keigo Nakamura. Concept bottleneck model with additional unsupervised concepts. *IEEE Access*, 2022.
- Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.
- Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. Conditional sum-product networks: Modular probabilistic circuits via gate functions. *Int. J. Approx. Reason.*, 2022.
- Andy Shih, Dorsa Sadigh, and Stefano Ermon. Hyperspns: Compact and expressive probabilistic circuits. In *NeurIPS*, 2021.
- Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling LIME and SHAP: adversarial attacks on post hoc explanation methods. In *AIES*, 2020.
- Johannes Stalkamp, Marc Schlipf, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 2012.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NeurIPS*, 2014.
- Martin Trapp, Robert Peharz, Marcin Skowron, Tamas Madl, Franz Pernkopf, and Robert Trapp. Structure inference in sum-product networks using infinite sum-product trees. In *NeurIPS Workshop on Practical Bayesian Nonparametrics*, 2016.
- Martin Trapp, Robert Peharz, Hong Ge, Franz Pernkopf, and Zoubin Ghahramani. Bayesian learning of sum-product networks. In *NeurIPS*, 2019.
- Emile van Krieken, Pasquale Minervini, Edoardo M. Ponti, and Antonio Vergari. On the independence assumption in neurosymbolic learning. In *ICML*, 2024.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- Fabrizio Ventola, Karl Stelzner, Alejandro Molina, and Kristian Kersting. Residual sum-product networks. In *International Conference on Probabilistic Graphical Models*. PMLR, 2020.
- Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Simplifying, regularizing and strengthening sum-product network structure learning. In *ECML PKDD*, 2015.
- Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 2017.
- Tianyi Wang and Shu-Ching Chen. Multi-label multi-task learning with dynamic task weight balancing. In *IEEE International Conference on Information Reuse and Integration for Data Science (IRI)*, 2020.
- Weiran Wang and Miguel A Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013.

- Riting Xia, Yan Zhang, Xueyan Liu, and Bo Yang. A survey of sum-product networks structural learning. *Neural Networks*, 2023.
- Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *TPAMI*, 2018.
- Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, 2018.
- Zhuolin Yang, Zhikuan Zhao, Boxin Wang, Jiawei Zhang, Linyi Li, Hengzhi Pei, Bojan Karlas, Ji Liu, Heng Guo, Ce Zhang, and Bo Li. Improving certified robustness via statistical learning with logical reasoning. In *NeurIPS*, 2022.
- Chih-Kuan Yeh, Been Kim, Sercan Ömer Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. In *NeurIPS*, 2020.
- Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frédéric Precioso, Stefano Melacci, Adrian Weller, Pietro Lió, and Mateja Jamnik. Concept embedding models: Beyond the accuracy-explainability trade-off. In *NeurIPS*, 2022.
- Mateo Espinosa Zarlenga, Katie Collins, Krishnamurthy Dvijotham, Adrian Weller, Zohreh Shams, and Mateja Jamnik. Learning to receive help: Intervention-aware concept embedding models. In *NeurIPS*, 2023.
- Jiawei Zhang, Linyi Li, Ce Zhang, and Bo Li. Care: Certifiably robust learning with reasoning via variational inference. In *SaTML*, 2023.
- Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.
- Han Zhao, Mazen Melibari, and Pascal Poupart. On the relationship between sum-product networks and bayesian networks. In *ICML*, 2015a.
- Han Zhao, Mazen Melibari, and Pascal Poupart. On the relationship between sum-product networks and bayesian networks. In *International Conference on Machine Learning*, pages 116–124. PMLR, 2015b.
- Han Zhao, Tameem Adel, Geoffrey J. Gordon, and Brandon Amos. Collapsed variational inference for sum-product networks. In *ICML*, 2016a.
- Han Zhao, Pascal Poupart, and Geoffrey J. Gordon. A unified approach for learning the parameters of sum-product networks. In *NeurIPS*, 2016b.

Appendix A Joint Optimization

In this section, we provide additional algorithmic details for the proposed joint optimization approach. The approach aims to jointly optimize the attribute recognition model and the circuit-based task predictor in an NPC using the loss function defined in Equation (3). Specifically, we employ the stochastic gradient descent algorithm to optimize the parameters of the attribute recognition model, *i.e.*, θ , and use the projected gradient descent algorithm to optimize the parameters of the circuit, *i.e.*, w , to ensure their positivity.

Optimizing θ Let η_A denote the learning rate for the attribute recognition model. The parameters of the model at the $(t+1)$ -th optimization step are as follows,

$$\theta^{(t+1)} = \theta^{(t)} + \eta_A \cdot \sum_{(x,y) \in D} \frac{\partial \log \Pr_{\theta^{(t)}, w^{(t)}}(Y = y | X = x)}{\partial \theta}.$$

Here, $\Pr_{\theta^{(t)}, w^{(t)}}(Y = y | X = x)$ is computed using Equation (1).

Optimizing w Let η_C denote the learning rate for the circuit. The d -th weight at the $(t+1)$ -th optimization step is as follows,

$$\begin{aligned} w_d^{(t+1)} &= \mathcal{P}_{\mathbf{R}_{++}} \left\{ w_d^{(t)} + \eta_C \cdot \sum_{(x,y) \in D} \frac{\partial \log \Pr_{\theta^{(t)}, w^{(t)}}(Y = y | X = x)}{\partial w_d} \right\} \\ &= \mathcal{P}_{\mathbf{R}_{++}} \left\{ w_d^{(t)} + \eta_C \cdot \sum_{(x,y) \in D} \frac{1}{\Pr_{\theta^{(t)}, w^{(t)}}(Y = y | X = x)} \frac{\partial \Pr_{\theta^{(t)}, w^{(t)}}(Y = y | X = x)}{\partial w_d} \right\}. \end{aligned}$$

Here, $\Pr_{\theta^{(t)}, w^{(t)}}(Y = y | X = x)$ is computed using Equation (1), and the gradient term is given by,

$$\begin{aligned} \frac{\partial \Pr_{\theta^{(t)}, w^{(t)}}(Y = y | X = x)}{\partial w_d} &= \sum_{a_{1:K}} \prod_{k=1}^K \Pr_{\theta^{(t)}}(A_k = a_k | X = x) \cdot \frac{\partial \Pr_{w^{(t)}}(Y = y | A_{1:K} = a_{1:K})}{\partial w_d} \\ &= \sum_{a_{1:K}} \prod_{k=1}^K \Pr_{\theta^{(t)}}(A_k = a_k | X = x) \cdot \frac{f_S(y, a_{1:K}; w^{(t)})}{f_S(\emptyset, a_{1:K}; w^{(t)})} \cdot \left[\frac{\partial \log f_S(y, a_{1:K}; w^{(t)})}{\partial w_d} - \frac{\partial \log f_S(\emptyset, a_{1:K}; w^{(t)})}{\partial w_d} \right]. \end{aligned}$$

In particular, $\prod_{k=1}^K \Pr_{\theta^{(t)}}(A_k = a_k | X = x)$ and $\frac{f_S(y, a_{1:K}; w^{(t)})}{f_S(\emptyset, a_{1:K}; w^{(t)})}$ are provided by the attribute recognition model and the circuit, respectively. $\frac{\partial \log f_S(y, a_{1:K}; w^{(t)})}{\partial w_d}$ and $\frac{\partial \log f_S(\emptyset, a_{1:K}; w^{(t)})}{\partial w_d}$ can be computed recursively within a circuit Poon and Domingos [2011].

Appendix B Proof for Compositional Error

In this section, we present a detailed proof for Theorem 2. Throughout the proof, expressions with parameters such as \Pr_θ , \Pr_w , and $\Pr_{\theta, w}$ refer to probabilities learned by the models, while those without parameters represent ground-truth probabilities.

$$\begin{aligned} \epsilon_{\theta, w} &= \mathbb{E}_X \left[\frac{1}{2} \sum_y \left| \Pr_{\theta, w}(Y = y | X) - \Pr(Y = y | X) \right| \right] \\ &\leq \mathbb{E}_X \left[\frac{1}{2} \sum_y \sum_{a_{1:K}} \left| \prod_k \Pr_\theta(A_k = a_k | X) \cdot \Pr_w(Y = y | A_{1:K} = a_{1:K}) - \prod_k \Pr(A_k = a_k | X) \cdot \Pr_w(Y = y | A_{1:K} = a_{1:K}) \right. \right. \\ &\quad \left. \left. + \prod_k \Pr(A_k = a_k | X) \cdot \Pr_w(Y = y | A_{1:K} = a_{1:K}) - \prod_k \Pr(A_k = a_k | X) \cdot \Pr(Y = y | A_{1:K} = a_{1:K}) \right| \right] \\ &\leq \mathbb{E}_X \left[\frac{1}{2} \sum_y \sum_{a_{1:K}} \left| \prod_k \Pr_\theta(A_k = a_k | X) - \prod_k \Pr(A_k = a_k | X) \right| \cdot \Pr_w(Y = y | A_{1:K} = a_{1:K}) \right] \quad (5) \\ &\quad + \mathbb{E}_X \left[\frac{1}{2} \sum_y \sum_{a_{1:K}} \prod_k \Pr(A_k = a_k | X) \cdot \left| \Pr_w(Y = y | A_{1:K} = a_{1:K}) - \Pr(Y = y | A_{1:K} = a_{1:K}) \right| \right]. \quad (6) \end{aligned}$$

Thus, the upper bound of $\epsilon_{\theta, w}$ is decomposed into two terms.

We derive the upper bound for the first term, *i.e.*, Eq(5), as follows,

$$\begin{aligned} \text{Eq(5)} &= \mathbb{E}_X \left[\frac{1}{2} \sum_{a_{1:K}} \left| \prod_k \Pr_{\theta}(A_k = a_k | X) - \prod_k \Pr(A_k = a_k | X) \right| \right] = \mathbb{E}_X \left[d_{\text{TV}} \left(\Pr_{\theta}(A_{1:K} | X), \Pr(A_{1:K} | X) \right) \right] = \epsilon_{\theta} \\ &\leq \mathbb{E}_X \left[\frac{1}{2} \sum_k \sum_{a_k} \left| \Pr_{\theta}(A_k = a_k | X) - \Pr(A_k = a_k | X) \right| \right] = \sum_k \mathbb{E}_X \left[d_{\text{TV}} \left(\Pr_{\theta}(A_k = a_k | X), \Pr(A_k = a_k | X) \right) \right] = \sum_k \epsilon_{\theta}^k. \end{aligned}$$

We derive the upper bound for the second term, *i.e.*, Eq(6), as follows,

$$\begin{aligned} \text{Eq(6)} &= \frac{1}{2} \sum_x \sum_y \sum_{a_{1:K}} \Pr(X = x, A_{1:K} = a_{1:K}) \cdot \left| \Pr(Y = y | A_{1:K} = a_{1:K}) - \Pr(Y = y | A_{1:K} = a_{1:K}) \right| \\ &= \frac{1}{2} \sum_y \sum_{a_{1:K}} \Pr(A_{1:K} = a_{1:K}) \cdot \left| \Pr_w(Y = y | A_{1:K} = a_{1:K}) - \Pr(Y = y | A_{1:K} = a_{1:K}) \right| \\ &= \frac{1}{2} \sum_y \sum_{a_{1:K}} \left| \Pr(A_{1:K} = a_{1:K}) \cdot \Pr_w(Y = y | A_{1:K} = a_{1:K}) - \Pr_w(A_{1:K} = a_{1:K}) \cdot \Pr_w(Y = y | A_{1:K} = a_{1:K}) \right. \\ &\quad \left. + \Pr_w(A_{1:K} = a_{1:K}) \cdot \Pr_w(Y = y | A_{1:K} = a_{1:K}) - \Pr(A_{1:K} = a_{1:K}) \cdot \Pr(Y = y | A_{1:K} = a_{1:K}) \right| \\ &\leq \frac{1}{2} \sum_y \sum_{a_{1:K}} \Pr_w(Y = y | A_{1:K} = a_{1:K}) \cdot \left| \Pr(A_{1:K} = a_{1:K}) - \Pr_w(A_{1:K} = a_{1:K}) \right| \\ &\quad + \frac{1}{2} \sum_y \sum_{a_{1:K}} \left| \Pr_w(Y = y, A_{1:K} = a_{1:K}) - \Pr(Y = y, A_{1:K} = a_{1:K}) \right| \\ &= \frac{1}{2} \sum_{a_{1:K}} \left| \Pr(A_{1:K} = a_{1:K}) - \Pr_w(A_{1:K} = a_{1:K}) \right| + d_{\text{TV}} \left(\Pr_w(Y, A_{1:K}), \Pr(Y, A_{1:K}) \right) \\ &= \frac{1}{2} \sum_{a_{1:K}} \left| \sum_y \Pr(Y = y, A_{1:K} = a_{1:K}) - \sum_y \Pr_w(Y = y, A_{1:K} = a_{1:K}) \right| + d_{\text{TV}} \left(\Pr_w(Y, A_{1:K}), \Pr(Y, A_{1:K}) \right) \\ &\leq \frac{1}{2} \sum_{a_{1:K}} \sum_y \left| \Pr(Y = y, A_{1:K} = a_{1:K}) - \Pr_w(Y = y, A_{1:K} = a_{1:K}) \right| + d_{\text{TV}} \left(\Pr_w(Y, A_{1:K}), \Pr(Y, A_{1:K}) \right) \\ &= 2d_{\text{TV}} \left(\Pr_w(Y, A_{1:K}), \Pr(Y, A_{1:K}) \right) = 2\epsilon_w. \end{aligned}$$

Combining results from Eq(5) and Eq(6), we have $\epsilon_{\theta, w} \leq \epsilon_{\theta} + 2\epsilon_w \leq \sum_k \epsilon_{\theta}^k + 2\epsilon_w$.

Appendix C Experimental Setup

In this section, we provide additional details regarding the experimental setup.

Model Architectures To ensure a fair comparison with the various baselines, we consistently adopt ResNet-34 [He et al., 2016] as the backbone for all methods. Specifically, in CBM [Koh et al., 2020], the concept recognition model is based on ResNet-34, where the final layer outputs concept probabilities. The task predictor is implemented as a linear layer that takes these concept probabilities as input and outputs predicted class scores. For CEM [Zarlenga et al., 2022], the first module is implemented using ResNet-34, with its final layer being the embedding layer defined in [Zarlenga et al., 2022]. This embedding layer produces both concept embeddings and concept probabilities. The subsequent task predictor uses only the concept embeddings as input to produce the predicted class scores. In DCR [Barbiero et al., 2023], the first module is identical to that of CEM, while the task predictor is the proposed deep concept reasoner (DCR). This reasoner takes concept probabilities as input and outputs predicted class scores, during which it leverages concept embeddings to formulate logical rules. For NPC, the attribute recognition model is implemented using a multi-task learning framework. Specifically, ResNet-34, without its original final layer, serves as the feature extractor to capture common features across multiple attributes. For each attribute, a series of two linear layers acts as a dedicated task head, outputting a probability vector corresponding to the attribute. The task predictor in NPC is implemented as a probabilistic circuit, with detailed construction provided

in Section 3.2.2.

Training Details For the baseline methods, training is performed with a batch size of 256 over 150 epochs. We use the SGD optimizer, configured with a learning rate of 0.01, a momentum of 0.9, and a weight decay of 0.0004. During training, we reduce the learning rate by a factor of 0.1 if the validation loss plateaus for 10 epochs. Additionally, the concept loss weight is set to 1 for CBM, CEM, and DCR, while the concept embedding size is configured to be 16 for both CEM and DCR [Zarlenga et al., 2022]. For NPC, the training process consists of three stages. In the first stage, the attribute recognition model is trained with a batch size of 256 over 150 epochs using the SGD optimizer. During training, we reduce the learning rate by a factor of 0.1 if the validation loss plateaus for 10 epochs. In the second stage, the circuit is learned and optimized following the procedures described by Gens and Domingos [2013], Zhao et al. [2016b], respectively. In the final stage, a learning rate of 0.01 is applied to the attribute recognition model while the circuit remains frozen, which is empirically shown to yield faster training and improved task performance. We run NPCs and all baseline models over five random seeds: 42, 52, 62, 72, and 82.

Counterfactual Explanations We adopt Algorithm 1 to generate CEs over 100 iterations. The learning rate γ is set to 0.005 for MNIST-Addition, 0.01 for GTSRB, 0.1 for CelebA, and 0.05 for AwA2.

Appendix D Attribute Configurations

Table 5 provides an overview of the attributes used, along with their corresponding values, for the four benchmark datasets. In MNIST-Addition and GTSRB, attributes are *single-valued*, meaning each sample is associated with only one value per attribute. In contrast, CelebA and AwA2 feature *multi-valued* attributes, allowing a sample to be associated with multiple values for a single attribute. For example, an image from the “zebra” class has both “black” and “white” as color attributes. Furthermore, attribute annotations in GTSRB, CelebA, and AwA2 are *class-wise*, where each class of samples all corresponds to the same specific set of attribute values. Conversely, in MNIST-Addition, samples from the same class may have different combinations of attribute values. For instance, a sample from the class “8” may correspond to attribute values such as “0, 8”, “1, 7”, or “8, 0”. Table 6 presents sample images from the four datasets.

Appendix E Human-Predefined Logical Rules

Human experience can be used to formulate specific logical rules for downstream tasks, representing valuable domain knowledge that can be integrated into models to enhance their reliability. Here, we demonstrate a two-step procedure for constructing logical rules based on a given set of observed samples $\bar{D} = \{(x, y, a_{1:K})\}$. 1) For each observed sample $(x, y, a_{1:K})$, we establish a corresponding logical rule of the form $\mathbb{I}(A_1 = a_1) \wedge \dots \wedge \mathbb{I}(A_K = a_K) \wedge \mathbb{I}(Y = y)$. 2) The occurrences of the logical rules derived from all observed samples are then normalized to form a weight for each rule, ensuring that the rules reflect the distribution of the observed data. Table 7 illustrates the rules constructed using training samples from the MNIST-Addition and GTSRB datasets. In addition to the standardized procedure introduced above, humans can also leverage their expertise to incorporate more diverse and task-specific rules beyond this form.

Table 5: Attributes and their corresponding values for the four benchmark datasets.

Dataset	Attribute	Values
MNIST-Addition	Number-First	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
	Number-Second	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
GTSRB	Color	Blue, Red, White
	Shape	Circle, Diamond, Octagon, Triangle
	Symbol	Arrow-Consecutive-Turns, Arrow-Down-Left, Arrow-Down-Right, Arrow-Left, Arrow-Right, Arrow-Roundabout, Arrow-Up, Arrow-Up-and-Left, Arrow-Up-and-Right, Bicycle, Bump, Car, Car-Truck, Car-Two, Deer, Exclamation-Mark, Ice-or-Snow, Person, Person-Two, Road-Narrows, Roadworks, Slash, Text, Traffic-Signal, Truck, Undefined
	Text	100, 120, 20, 30, 50, 60, 70, 80, Stop, Undefined
CelebA	Mouth	Mouth-Slightly-Open, None, Smiling
	Face	High-Cheekbones, None, Oval-Face
	Cosmetic	Heavy-Makeup, None, Wearing-Lipstick
	Hair	None, Wavy-Hair
	Appearance	Attractive, None
AwA2	Color	Black, Blue, Brown, Gray, None, Orange, Red, White, Yellow
	Surface	Furry, Hairless, Patches, Spots, Stripes, Toughskin
	Body	Bipedal, Bulbous, Horns, Lean, Longleg, Longneck, Quadrupedal, Tail, Tusks
	Limb	Claws, Flippers, Hands, Hooves, None, Pads, Paws

Table 6: Example images and their annotations from the four benchmark datasets. The CelebA image is redacted in compliance with its terms of use.





Image	Attribute	Values
	Number-First	3
	Number-Second	5
	Color	Red
	Shape	Octagon
	Symbol	Text
	Text	Stop
	Mouth	Mouth-Slightly-Open, Smiling
	Face	High-Cheekbones
	Cosmetic	Heavy-Makeup, Wearing-Lipstick
	Hair	Wavy-Hair
	Appearance	Attractive
	Color	Black, White
	Surface	Furry, Stripes, Toughskin
	Body	Lean, Longleg, Longneck, Quadrupedal, Tail
	Limb	Hooves

Table 7: Logical rules constructed using training samples from MNIST-Addition and GTSRB datasets (Part 1).

Dataset	Logical Rules
MNIST-Addition	$\mathbb{I}(\text{Number-First} = a_1) \wedge \mathbb{I}(\text{Number-Second} = a_2) \wedge \mathbb{I}(\text{Class} = a_1 + a_2), a_1, a_2 \in [0, 9]$
GTSRB	$\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Text}) \wedge \mathbb{I}(\text{Text} = 20) \wedge \mathbb{I}(\text{Class} = \text{regulatory-maximum-speed-limit-20})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Text}) \wedge \mathbb{I}(\text{Text} = 30) \wedge \mathbb{I}(\text{Class} = \text{regulatory-maximum-speed-limit-30})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Text}) \wedge \mathbb{I}(\text{Text} = 50) \wedge \mathbb{I}(\text{Class} = \text{regulatory-maximum-speed-limit-50})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Text}) \wedge \mathbb{I}(\text{Text} = 60) \wedge \mathbb{I}(\text{Class} = \text{regulatory-maximum-speed-limit-60})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Text}) \wedge \mathbb{I}(\text{Text} = 70) \wedge \mathbb{I}(\text{Class} = \text{regulatory-maximum-speed-limit-70})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Text}) \wedge \mathbb{I}(\text{Text} = 80) \wedge \mathbb{I}(\text{Class} = \text{regulatory-maximum-speed-limit-80})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Text}) \wedge \mathbb{I}(\text{Text} = 100) \wedge \mathbb{I}(\text{Class} = \text{regulatory-maximum-speed-limit-100})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Text}) \wedge \mathbb{I}(\text{Text} = 120) \wedge \mathbb{I}(\text{Class} = \text{regulatory-maximum-speed-limit-120})$ $\mathbb{I}(\text{Color} = \text{White}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Text}) \wedge \mathbb{I}(\text{Text} = 80) \wedge \mathbb{I}(\text{Class} = \text{regulatory-end-of-maximum-speed-limit-80})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Cat-Two}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-no-overtaking})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Car-Truck}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-no-overtaking-by-heavy-goods-vehicles})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Up}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-crossroads})$ $\mathbb{I}(\text{Color} = \text{White}) \wedge \mathbb{I}(\text{Shape} = \text{Diamond}) \wedge \mathbb{I}(\text{Symbol} = \text{Undefined}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-priority-road})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Undefined}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-yield})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Octagon}) \wedge \mathbb{I}(\text{Symbol} = \text{Text}) \wedge \mathbb{I}(\text{Text} = \text{Stop}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-stop})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Undefined}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-road-closed-to-vehicles})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Truck}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-no-heavy-goods-vehicles})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Slash}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-no-entry})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Exclamation-Mark}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-other-danger})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Left}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-curve-left})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Right}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-curve-right})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Consecutive-Turns}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-double-curve-first-left})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Bump}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-uneven-road})$ $\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Car}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-slippery-road-surface})$

Table 8: Logical rules constructed using training samples from MNIST-Addition and GTSRB datasets (Part 2).

Dataset	Logical Rules
GTSRB	$\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Road-Narrows}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-road-narrows-right})$
	$\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Roadworks}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-roadworks})$
	$\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Traffic-Signal}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-traffic-signals})$
	$\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Person}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-pedestrians-crossing})$
	$\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Person-Two}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-children})$
	$\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Bicycle}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-bicycles-crossing})$
	$\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Ice-or-Snow}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-ice-or-snow})$
	$\mathbb{I}(\text{Color} = \text{Red}) \wedge \mathbb{I}(\text{Shape} = \text{Triangle}) \wedge \mathbb{I}(\text{Symbol} = \text{Deer}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{warning-wild-animals})$
	$\mathbb{I}(\text{Color} = \text{White}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Undefined}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-end-of-prohibition})$
	$\mathbb{I}(\text{Color} = \text{Blue}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Right}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-turn-right-ahead})$
	$\mathbb{I}(\text{Color} = \text{Blue}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Left}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-turn-left-ahead})$
	$\mathbb{I}(\text{Color} = \text{Blue}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Up}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-go-straight})$
	$\mathbb{I}(\text{Color} = \text{Blue}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Up-and-Right}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-go-straight-or-turn-right})$
	$\mathbb{I}(\text{Color} = \text{Blue}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Up-and-Left}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-go-straight-or-turn-left})$
	$\mathbb{I}(\text{Color} = \text{Blue}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Down-Right}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-keep-right})$
	$\mathbb{I}(\text{Color} = \text{Blue}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Down-Left}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-keep-left})$
	$\mathbb{I}(\text{Color} = \text{Blue}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Arrow-Roundabout}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-roundabout})$
	$\mathbb{I}(\text{Color} = \text{White}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Car-Two}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-end-of-no-overtaking})$
	$\mathbb{I}(\text{Color} = \text{White}) \wedge \mathbb{I}(\text{Shape} = \text{Circle}) \wedge \mathbb{I}(\text{Symbol} = \text{Car-Truck}) \wedge \mathbb{I}(\text{Text} = \text{Undefined}) \wedge \mathbb{I}(\text{Class} = \text{regulatory-end-of-no-overtaking-by-heavy-goods-vehicles})$

Appendix F Model Explanations

Figure 8 illustrates examples for NPC(Knowledge) from the four benchmark datasets. Each example comprises an image, the ground-truth labels for the class and attributes, the class predicted by NPC(Knowledge), and, lastly, the corresponding MPE. In these instances, NPC(Knowledge) provides correct class predictions, and the MPEs are aligned with the ground-truth attribute labels.





MNIST-Add		<p>Class Label: 6</p> <p>Attribute Labels: {<u>Number-First</u>: 6; <u>Number-Second</u>: 0}</p> <p>Class Prediction: 6</p> <p>MPE: {<u>Number-First</u>: 6; <u>Number-Second</u>: 0}</p>
GTSRB		<p>Class Label: Regulatory-Maximum-Speed-Limit-30</p> <p>Attribute Labels: {<u>Color</u>: Red; <u>Shape</u>: Circle; <u>Symbol</u>: Text; <u>Text</u>: 30}</p> <p>Class Prediction: Regulatory-Maximum-Speed-Limit-30</p> <p>MPE: {<u>Color</u>: Red; <u>Shape</u>: Circle; <u>Symbol</u>: Text; <u>Text</u>: 30}</p>
CelebA		<p>Class Label: Group 5</p> <p>Attribute Labels: {<u>Mouth</u>: Mouth-Slightly-Open, Smiling; <u>Face</u>: High-Cheekbones; <u>Cosmetic</u>: Heavy-Makeup, Wearing-Lipstick; <u>Hair</u>: Wavy-Hair; <u>Appearance</u>: Attractive}</p> <p>Class Prediction: Group 5</p> <p>MPE: {<u>Mouth</u>: Mouth-Slightly-Open; <u>Face</u>: High-Cheekbones; <u>Cosmetic</u>: Heavy-Makeup; <u>Hair</u>: Wavy-Hair; <u>Appearance</u>: Attractive}</p>
AWA2		<p>Class Label: Sheep</p> <p>Attribute Labels: {<u>Color</u>: Black, White; <u>Surface</u>: Furry; <u>Body</u>: Bulbous, Quadrapedal; <u>Limb</u>: Hooves}</p> <p>Class Prediction: Sheep</p> <p>MPE: {<u>Color</u>: Black; <u>Surface</u>: Furry; <u>Body</u>: Quadrapedal; <u>Limb</u>: Hooves}</p>

Figure 8: Examples from the four benchmark datasets. Each example includes an image, the ground-truth class label, the ground-truth attribute labels, the NPC(Knowledge) class prediction, and, lastly, the corresponding MPE. These examples illustrate MPEs that align with the ground-truth attribute labels. The CelebA image is redacted in compliance with its terms of use.

Figure 9 illustrates some examples for NPC(Knowledge) from the four benchmark datasets. Each sample consists of an image, the attribute and class predictions, the generated CE, and, lastly, the class prediction corrected by the CE. In these examples, NPC(Knowledge) incorrectly predicts the class, while CEs effectively correct them by making minimal adjustments to the attribute predictions.

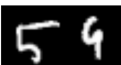



MINIST-Add		<p>Attribute Predictions: {<u>Number-First</u>: 5 (100.0%); <u>Number-Second</u>: 4 (99.6%)}</p> <p>Class Prediction: 9</p> <p>CE: {<u>Number-First</u>: 5 (100.0%); <u>Number-Second</u>: 9 (62.8%)}</p> <p>Corrected Class Prediction: 14</p>
GTSRB		<p>Attribute Predictions: {<u>Color</u>: Red (100%); <u>Shape</u>: Circle (100.0%); <u>Symbol</u>: Text (99.9%); <u>Text</u>: 30 (65.7%)}</p> <p>Class Prediction: Regulatory-Maximum-Speed-Limit-30</p> <p>CE: {<u>Color</u>: Red (100%); <u>Shape</u>: Circle (100.0%); <u>Symbol</u>: Text (100.0%); <u>Text</u>: 80 (46.3%)}</p> <p>Corrected Class Prediction: Regulatory-Maximum-Speed-Limit-80</p>
CelebA		<p>Attribute Predictions: {<u>Mouth</u>: Mouth-Slightly-Open (91.4%); <u>Face</u>: High-Cheekbones (99.0%); <u>Cosmetic</u>: Wearing-Lipstick (76.2%); <u>Hair</u>: None (99.6%); <u>Appearance</u>: None (53.5%)}</p> <p>Class Prediction: Group 40</p> <p>CE: {<u>Mouth</u>: Mouth-Slightly-Open (93.3%); <u>Face</u>: High-Cheekbones (100.0%); <u>Cosmetic</u>: None (56.0%); <u>Hair</u>: None (100.0%); <u>Appearance</u>: None (70.8%)}</p> <p>Corrected Class Prediction: Group 1</p>
Awa2		<p>Attribute Predictions: {<u>Color</u>: Black (98.8%); <u>Surface</u>: Hairless (95.0%); <u>Body</u>: Bulbous (99.6%); <u>Limb</u>: Flippers (100.0%)}</p> <p>Class Prediction: Humpback+Whale</p> <p>CE: {<u>Color</u>: Black (98.6%); <u>Surface</u>: Hairless (94.4%); <u>Body</u>: Tail (100.0%); <u>Limb</u>: Flippers (100.0%)}</p> <p>Corrected Class Prediction: Seal</p>

Figure 9: Examples from the four benchmark datasets. Each example includes an image, the NPC(Knowledge) attribute and class predictions, the corresponding CE, and the class prediction corrected by CE. These examples focus on CEs that successfully correct the class predictions. For simplicity, both attribute predictions and CEs display only the most confident value for each attribute. The CelebA image is redacted in compliance with its terms of use.