A Comprehensive Survey on Self-Interpretable Neural Networks

Yang Ji, Ying Sun*, *Member, IEEE*, Yuting Zhang, Zhigaoyuan Wang, Yuanxin Zhuang, Zheng Gong, Dazhong Shen, *Member, IEEE*, Chuan Qin, *Member, IEEE*, Hengshu Zhu, *Senior Member, IEEE*, and Hui Xiong*, *Fellow, IEEE*

***Abstract*—Neural networks have achieved remarkable success across various fields. However, the lack of interpretability limits their practical use, particularly in critical decision-making scenarios. Post-hoc interpretability, which provides explanations for pre-trained models, is often at risk of robustness and fidelity. This has inspired a rising interest in self-interpretable neural networks, which inherently reveal the prediction rationale through the model structures. Although there exist surveys on post-hoc interpretability, a comprehensive and systematic survey of self-interpretable neural networks is still missing. To address this gap, we first collect and review existing works on self-interpretable neural networks and provide a structured summary of their methodologies from five key perspectives: attribution-based, function-based, concept-based, prototype-based, and rule-based self-interpretation. We also present concrete, visualized examples of model explanations and discuss their applicability across diverse scenarios, including image, text, graph data, and deep reinforcement learning. Additionally, we summarize existing evaluation metrics for self-interpretability and identify open challenges in this field, offering insights for future research. To support ongoing developments, we present a publicly accessible resource to track advancements in this domain: https://github.com/yangji721/Awesome-Self-Interpretable-Neural-Network.**

***Index Terms*—Interpretability; Self-Interpretable Neural Networks; Explainable Artificial Intelligence; Model Explanation**

I. Introduction

OVER the past decade, neural networks have achieved remarkable success in solving complex problems across various fields. This success is largely due to their vast hypothesis space, facilitated by deep signal propagation through hidden units. Despite their impressive predictive power, the lack of interpretability poses significant challenges. Without a clear understanding of how the model arrives at its decisions, it becomes difficult to build user trust in the model's predictions, particularly in critical decision-making scenarios.

Ying Sun and Hui Xiong are corresponding authors.

Yang Ji, Ying Sun, Yuting Zhang, Zhigaoyuan Wang, Yuanxin Zhuang, and Zheng Gong are with the Artificial Intelligence Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China (e-mail: {yji655, yzhang755, zwang901, yzhuang436, zgong768}@connect.hkust-gz.edu.cn, yings@hkust-gz.edu.cn).

Dazhong Shen is with the College of Computer Science and Technology Nanjing University of Aeronautics and Astronautics, Nanjing 211100, China (e-mail: dazh.shen@gmail.com).

Chuan Qin and Hengshu Zhu are with the Computer Network Information Center, Chinese Academy of Sciences, Beijing 100083, China (e-mail: {chuanqin0426, zhuhengshu}@gmail.com).

Hui Xiong is with the Thrust of Artificial Intelligence, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511458, China (e-mail: xionghui@ust.hk).

Explainable AI (XAI) has emerged as a popular research area to address the interpretability challenges of neural networks. Post-hoc interpretability methods, which provide explanations for pre-trained models, have been widely adopted to enhance model transparency. These methods aim to generate human-understandable explanations for pre-trained model predictions. Despite the flexibility, post-hoc methods often fall short of offering a transparent view of the model's internal workings [1], [2]. Moreover, they are often computationally expensive and struggle to faithfully capture the true behavior of pre-trained models [3]–[6]. As highlighted in recent studies [7], [8], the inherent opaqueness of these models hinders the understanding of the decision-making process. This growing awareness of the limitations of post-hoc explanations has driven the demand for neural network architectures that can intrinsically reveal the reasoning behind their predictions.

Traditional machine learning models, such as decision trees and linear models, are naturally interpretable. However, their optimization and learning paradigms differ fundamentally from those of neural networks, limiting their integration in the current AI landscape, where neural networks dominate. For instance, while decision trees are effective for classification tasks when the features are pre-defined, they cannot operate within the gradient-based framework of neural networks, which prevents end-to-end learning. This incompatibility highlights the need to explore self-interpretable neural network approaches that retain the powerful learning capabilities of deep models while offering intrinsic interpretability.

Although numerous reviews have discussed interpretability techniques in neural networks, they either focus on specific domains or post-hoc explanations [9]–[13]. Model interpretation itself is a subjective and complex concept, which often varies significantly across domains and thus makes systematic summarization challenging. To the best of our knowledge, there still lacks a comprehensive and systematic review to summarize the current advances of self-interpretable neural networks (SINNs). Furthermore, the field lacks a consolidated summary of evaluation metrics for assessing self-interpretability, and future research directions in this area remain under-explored.

In this paper, we provide a comprehensive review of state-of-the-art SINNs. We propose a taxonomy that categorizes existing work into five key dimensions. This taxonomy design is grounded in the various explanation forms embedded within the architectures of these networks.

- **Attribution-based** methods identify which input elements most strongly influence the model's prediction.

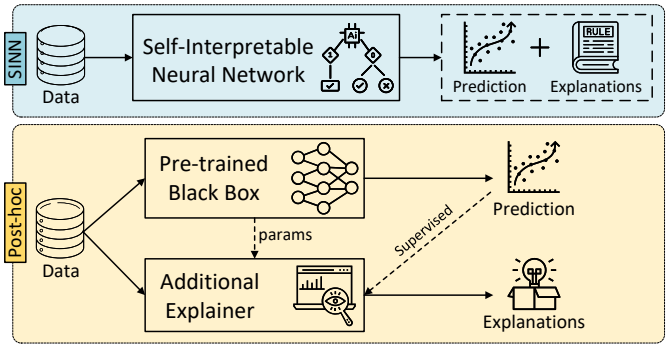


Fig. 1: Comparison between self-interpretation and post-hoc.

- **Function-based** methods approximate the model's function with simpler, transparent functions to understand the relationship between internal variables.
- **Concept-based** methods interpret models from the perspective of human-understandable concepts, which bridge the gap between the model's internal representations and domain-relevant ideas.
- **Prototype-based** methods deliver a transparent view of the model prediction by comparing the input with representative cases learned from the training data.
- **Rule-based** methods incorporate logical rules into the neural network architecture to achieve symbolic reasoning.

To further contextualize these approaches, we discuss widely used interpretability techniques and highlight emerging research directions across various domains, including image, text, graph data, and deep reinforcement learning (DRL). By providing a comprehensive review, we aim to offer insights into the practical applications, benefits, and potential impact of SINNs on various domains.

To summarize, our contributions are as follows:

- **New taxonomy and systematic review.** We present a unified perspective and conduct an up-to-date, systematic review of SINNs. Existing methods are categorized into five categories. For each category, we abstract its foundational frameworks, interpretable modeling techniques, and provide detailed comparisons highlighting its strengths and limitations. To the best of our knowledge, this is the first systematic and comprehensive review of SINNs.
- **Application Perspectives.** To better understand the practical implications of SINNs, we explore their applications across four popular domains. We provide concrete examples of model explanations with customized interpretation techniques, and discuss some trending research directions, such as large language models (LLMs) in these domains.
- **Quantitative Evaluation Metrics.** We offer an extensive overview of the state-of-the-art quantitative evaluation metrics for SINNs, providing researchers with concrete guidelines and references for comparing both existing and emerging methods.
- **Potential Research Directions.** We provide comprehensive analysis and discussions on the relationship between post-hoc interpretability and self-interpretability, identifying potential future research directions for future model designs and new application scenarios.

The rest of this survey will be organized as follows: Section II gives some brief background on terminology and related research areas. Section III introduces our taxonomy of SINNs and discusses the common structures used in these networks. Section IV describes the commonly used interpretability techniques from the mainstream application perspectives. The quantitative performance evaluation metrics are provided in Section V. Section VI discusses the promising research directions of self-interpretable neural networks, and Section VII gives a conclusion of this article.

II. Preliminaries

In this section, we begin by describing SINNs and distinguishing them from post-hoc interpretability methods to clarify the scope of our survey. We then provide an overview of foundational surveys in interpretable machine learning, emphasizing the unique focus of our survey.

A. Survey Scope

Building on the definition of interpretable machine learning by [28], [29], SINNs are neural architectures where interpretability is built-in architecturally and enforced through domain-specific constraints. The neural components (neurons, layers, or modules) could represent human-understandable units, allowing the network to explain its decisions based directly on its internal representations, without relying on external parsers or explainers. [1] These constraints can vary significantly depending on the application domain. SINNs have two key properties: self-interpretability [2], where the model provides explanations simultaneously with its predictions, and end-to-end differentiability, allowing for efficient neural network training. As shown in Figure 1, SINNs differ from post-hoc methods in how model explanations are generated. Post-hoc methods work on pre-trained models, interpreting their predictions through a separate explainer. In contrast, SINNs produce both predictions and explanations simultaneously during the inference process. This paper specifically focuses on the interpretability arised directly from the neural structures, excluding non-neural network-based methods, such as algorithmic interpretability [30], dataflow analysis [31], and parts of neuro-symbolic methods which reply on external program parsers for structured module organization [32].

We focus on collecting and reviewing SINN-related papers published in leading AI-related conferences and journals, including but not limited to NeurIPS, ICML, ICLR, AAAI, CVPR, ACL, KDD, WWW, IJCAI, TAPMI, JMLR, TKDE, TOIS, and Nature-related journals, as well as influential works from the broader research community. Our goal is to provide a systematic and accessible overview of the state-of-the-art design principles, applications, and evaluation metrics of SINNs. Additionally, we aim to identify commonalities and

[1] In this paper, the terms "interpretability" and "explainability" are used interchangeably to refer to the ability to explain the networks' decisions.

[2] Throughout the literature, the term "self-interpretable" is also known as "ante-hoc", "inherently interpretable", or "explainable-by-design", or "built-in explanations". They all refer to models with built-in interpretability rather than post-hoc explanations.

TABLE I: An overview of representative XAI Surveys on different interpretability aspects. ○ represents "Not Covered", ◒ represents "Partially Covered", and ● represents "Fully Covered".

Paper	Post-hoc	Self-Interpretation					Application Domains			
		Attribution	Function	Concept	Prototype	Rule	Image	Text	Graph	DRL
[10]	●	◒	○	○	○	◒				
[11]	●	◒	○	◒	○	◒	✔	✔		
[12]	◒	◒	○	○	○	○				
[13]	●	◒	○	○	○	◒				
[14]	●	◒	○	○	○	○	✔	✔		
[15]	●	○	○	○	○	○		✔		
[16]–[20]	◒	◒	○	◒	◒	◒		✔		
[21]–[23]	◒	◒	○	○	◒	○			✔	
[24]–[27]	◒	◒	○	◒	◒	◒				✔
Ours	○	●	●	●	●	●	✔	✔	✔	✔

differences across various SINN architectures and applications, thereby facilitating the development of novel SINN models and applications.

B. Related Surveys

Recent advancements in XAI have been extensively documented through various surveys. This subsection reviews foundational XAI surveys and elucidates the distinct focus of our survey on self-interpretable neural networks.

Several important surveys [33]–[37] take an interdisciplinary approach, using ideas from philosophy, psychology, and cognitive science. For example, Miller [33] combines philosophical and social science views to build a theory of explanations in XAI. Langer et al. [34] discuss what different users, like developers and regulators, expect from model explanations. Vilone and Longo [37] review definitions and methods for interpretability, connecting theory with practice. These surveys have shaped both technical and interdisciplinary research in XAI.

As summarized in Table I, numerous studies [9]–[14], [38], [39] delve into specific aspects of self-interpretability. However, many of these primarily focus on post-hoc interpretability methods, only briefly addressing self-interpretable approaches without thoroughly exploring their unique designs and contributions. Some surveys [12], [13], [38], [39] do review self-interpretable techniques but embed these discussions within the broader XAI context, lacking a systematic and dedicated exploration of SINNs as a standalone topic. This gap underscores the need for a focused survey on SINNs, given their diverse implementations across various domains and different innovative approaches.

Furthermore, surveys targeting specific data types or application domains, such as NLP [15], [18]–[20], [40], [41], graph data [21]–[23], and RL [24]–[27]. They provide unique taxonomies of interpretability methods tailored to those fields. However, many SINN architectures across diverse domains share common foundational ideas but lack unified understanding and categorization. Existing surveys do not fully capture the breadth and depth of SINNs, which span multiple fields and employ various innovative approaches. By integrating these shared concepts and examining domain-specific adaptations, our survey provides a cohesive framework for understanding self-interpretable network designs, thereby addressing a significant gap in previous surveys.

III. Self-Interpretation Neural Networks

In this section, we introduce the general principles of self-interpretation, focusing on ideas that are broadly applicable across various fields while omitting specific application details. For example, in computer vision, we will not discuss image feature extraction backbones and leave such topics in the next section. Notably, we highlight the common structures used in SINNs and classify them based on their interpretability techniques. We categorize SINNs into groups including attribution, function, concept, prototype, and rule-based approaches.

A. Attribution-based Self-Interpretation

Feature attribution is a key class of interpretability techniques that aims to identify the features most responsible for a model's predictions [42], [43]. As highlighted in previous studies [1], [44], [45], many interpretation algorithms can be unified under the framework of *additive attribution*, which has provided a cohesive lens for understanding post-hoc methods [1], [46]. Extending this viewpoint, self-attributing models pursue the same interpretive goals by providing decomposed and meaningful explanations. However, they differ in that post-hoc methods generate interpretations after training, whereas self-attributing models integrate interpretability directly into their network design.

Although different tasks may involve varying forms of input, the key idea is that feature attribution ultimately focuses on lower-dimensional representations. For instance, while many studies evaluate interpretability techniques on image datasets, the actual attributions often occur on feature maps extracted from raw pixels rather than solely on the pixel values themselves. In this work, we concentrate on the attribution component and leave the details of feature extraction (e.g., convolutional layers for images or embedding layers for textual data) abstracted away. Nevertheless, certain domain-specific constraints or properties (discussed further in Section IV) may shape how attributions are computed or interpreted in practice.

To formalize this, let r denote the original raw sample, and let $x = h(r)$ represent the features extracted by the backbone function $h(\cdot)$. In a general setting, let each feature x_i be in $\mathbb{R}^d$,

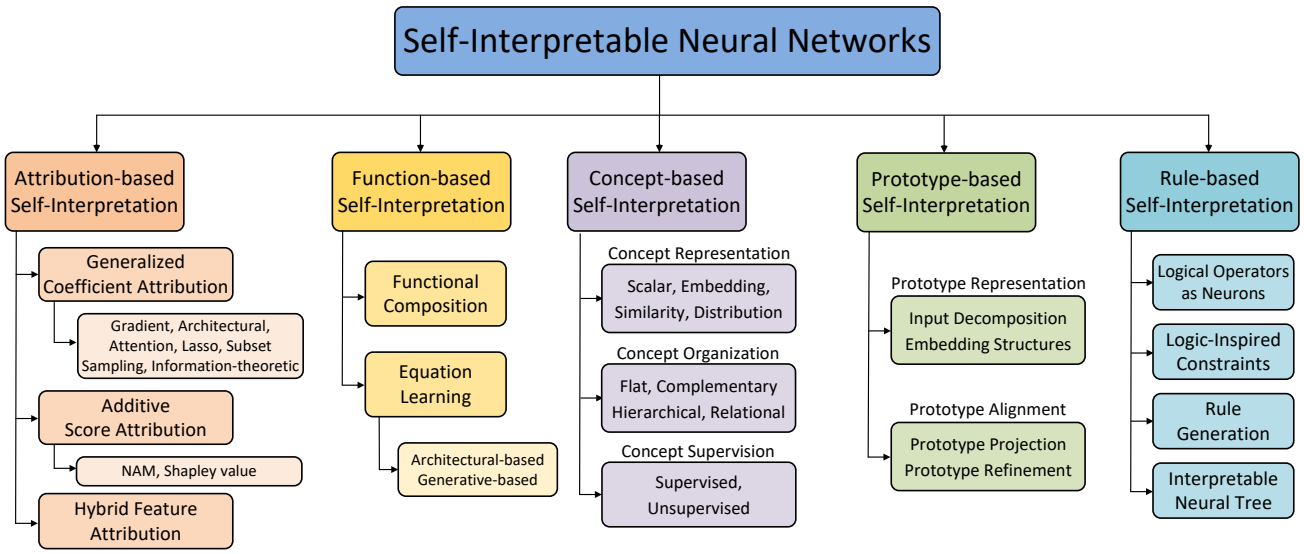


Fig. 2: Taxonomy of self-interpretable neural networks.

where d is the dimension of a given "feature field." Common examples include: (1) *Image data:* r is the raw image, N is the number of extracted feature maps, and d is the spatial size of each map; (2) *Tabular data:* $r = x$ is directly the feature vector, often with $d = 1$ if each feature is scalar; and (3) *Recommender systems:* one-hot vectors are extracted so that d corresponds to the embedding dimension or size of the data field. Without loss of generality, one can assume a consistent dimensionality d across all features, since any discrepancies in shape can be managed within a neural network via projection or reshaping. We thus write $\{x_1, \ldots, x_N\}$ to denote the set of N components subject to attribution in the model.

However, merely training a neural network with a decomposition structure does not guarantee that attributions will be meaningful or domain-grounded. In the absence of explicit constraints, the attribution terms ϕ_i need not correspond to the true contribution of each feature i. For example, ϕ_i might encode spurious correlations or depend on higher-order feature interactions, leading to interpretations that are neither stable nor consistent across samples. To address these concerns, a variety of self-attributing models have been proposed that impose constraints, including regularizers and structural inductive biases, on the attribution terms. The subsequent sections categorize these self-attribution neural networks according to the attribution forms and constraints they employ. We will detail the core methods they use to ensure that attributions are meaningful and trustworthy.

1) Generalized Coefficient Attribution: The concept of using coefficients to interpret models is a long-standing tradition in statistics, with its roots in linear regression (LR) and generalized linear model (GLM). LR, expressed as $f(x) = \sum_{i=1}^{N} a_i x_i + a_0$, is one of the most interpretable modeling techniques, where the coefficients a_i quantify the sensitivity of the output to each input feature. GLM extends LR by introducing link functions and probabilistic frameworks, enabling broader applications while retaining the interpretable structure. Moreover, coefficient interpretation can be generalized to neural networks by applying generalized coefficients to the feature layer. These coefficients act as proxies for feature reweighting or selection and are learned by the model to quantify the importance of each feature in the prediction, as expressed:

$$f(\boldsymbol{x}; \theta, \phi_0) = \sum_{i=1}^{N} \alpha_i(\boldsymbol{x}; \theta)\boldsymbol{x}_i + \phi_0. \\ \boldsymbol{x} = h(r), \text{ s.t. } \boldsymbol{\alpha} \in \mathcal{C}. \tag{1}$$

In this case, $\boldsymbol{x}_i \in \mathbb{R}^d$ operates as the input features and is directly aggregated in the high dimensional space. The coefficients $\alpha_i(\boldsymbol{x}; \theta)$ can be regarded as the attribution value of the i-th feature. Various constraints $\mathcal{C}$ can be imposed on the coefficients to ensure meaningful attributions. These constraints should be sparse and interpretable, reflecting the actual sensitivity of the output to the input features. It is worth noting that feature attribution can be flexibly augmented with feature extractors upstream, transformations downstream, or even stacked in multiple layers. We will not discuss the profusion of variations, but rather focus on their general principles and implications.

Gradient-based Constraints. SENN [45] is a pioneering study to formulate self-attribution neural networks with generalized coefficients. SENN proposes to ensure that the coefficients $\alpha(x)$ mirror the sensitivity of the output $f(x)$ to the input $\boldsymbol{x}$ by introducing the constraint set: $\mathcal{C}_{\text{SENN}} = \{\boldsymbol{\alpha}(x) : \boldsymbol{\alpha}(x) \approx \nabla_x f(x)\}$. Using the chain rule, SENN generalizes this gradient alignment to the Jacobian matrix of the feature transformation function $h(\cdot)$, denoted as $\mathcal{C}_{\text{SENN}} = \{\boldsymbol{\alpha}(r) : \nabla_r f(r) \approx \boldsymbol{\alpha}(r)^\top J_r^h(r)\}$. Essentially, SENN utilizes a composition of local linear models to represent a complex nonlinearship, with each coefficient indicating the importance of a feature by its sensitivity to local disturbances[3].

Architectural Constraints. The difficulty of feature attribution lies in the highly nonlinearity of neural networks.

[3]While subsequent works [98], [101], [102] have followed the SENN framework, they notably diverge from its original gradient-based constraints.

TABLE II: Comparison of Attribution Methods, Constraints, and Representative Studies

	Attribution Methods	Constraints	Representative Studies
Generalized Coefficient Attribution (Sec III-A1)	Gradient-based	$\mathcal{C} = \{\boldsymbol{\alpha}(x) : \boldsymbol{\alpha}(x) \approx \nabla_x f(x)\}$	SENN [45]
	Architectural	$\mathcal{C} = \{\boldsymbol{\alpha}(x) : \boldsymbol{\alpha}(x) = g(W_a W_b x + b)\}$ $\mathcal{C} = \{\boldsymbol{\alpha}(x) : \boldsymbol{\alpha}(x) = \lvert\cos(\angle(x,w))\rvert^B \times \operatorname{sgn}(\cos(\angle(x,w)))\}$	Coda-Nets [47] B-cos Networks [48], [49]
	Attention-based	$\mathcal{C} = \{\boldsymbol{\alpha}(x) \in \mathbb{R}^N : \alpha_i \geq 0\ \forall i,\ \text{and}\ \sum_{i=1}^N \alpha_i = 1\}$	Attention variants [50]–[59]
	Lasso-based	$\mathcal{C} = \{\boldsymbol{\alpha}(x) \in \mathbb{R}^N : \lVert\boldsymbol{\alpha}(x)\rVert_1 \leq \lambda\}$	Lasso variants [60]–[64]
	Subset Sampling	$\mathcal{C} = \left\{\boldsymbol{\alpha}(x) \in \{0,1\}^N \;\middle|\; \sum_{i=1}^N \alpha_i(x) = K\right\}$	Soft [65]–[71], Hard [72]–[75]
	Information-theoretic	$\mathcal{C} = \{\boldsymbol{\alpha}(x) \mid \text{and } I(\boldsymbol{\alpha}(x)x; y) - \beta I(\boldsymbol{\alpha}(x)x; x) \text{ is maximized}\}$	Substructure extraction [76]–[84]
Additive Score Attribution (Sec III-A2)	Neural Additive Model	Uniqueness of Feature Dependence, Additive Structure	NAM variants [85]–[97]
	Shapley-based	Efficiency, Linearity, Nullity, Symmetry	SASANet [98], SHAPNet [99]
Hybrid Feature Attribution (Sec III-A3)		Coefficient Constraints, Additive Score Constraints	SSCN [100]

To address this challenge, several studies propose dynamic linear models that make predictions through a series of input-dependent linear transformations, which can be expressed as: $f(x;\theta) = l_L \circ l_{L-1} \circ \cdots \circ l_2 \circ l_1(x)$. Each layer typically computes the output as $l(x) = \boldsymbol{\alpha}(x)x$, where $\boldsymbol{\alpha}(x)$ is input-varying and upper-bounded by a threshold. To implement this idea, Coda-Nets [47] imposes the constraint to each layer $\mathcal{C}_{\mathrm{coda-net}} = \{\boldsymbol{\alpha}(x) : \boldsymbol{\alpha}(x) = g(W_a W_b x + b)\}$, where $g(\cdot)$ is a non-linear function (e.g., unit norm and squashing function), W_a, W_b, b are learnable parameters. This structural constraint ensures $\boldsymbol{\alpha}(x)$ to encode the most frequent patterns in the inputs while making this attribution inherited across layers. B-cos Networks [48], [49] further extend this idea with an increased exponent B for flexible weight adjustment: $\mathcal{C}_{\mathrm{B-cos}} = \{\boldsymbol{\alpha}(x) : \boldsymbol{\alpha}(x) = \lvert\cos(\angle(x,w))\rvert^B \times \operatorname{sgn}(\cos(\angle(x,w)))\}$, where w is a learnable vector with unit norm. All these studies propose to capture network nonlinearity into the coefficients in the linear form and allows linear contribution mapping to the output.

Attention-based Constraints. Attention mechanisms have been widely used in neural networks to dynamically highlight the most informative features for the prediction. The inherent interpretability lies in its attention weights, which typically constrained to be nonnegative and to sum to one, ensuring that they form a interpretable distribution: $\mathcal{C}_{\text{att}} = \{\boldsymbol{\alpha}(x) \in \mathbb{R}^N : \alpha_i \geq 0\ \forall i,\ \text{and}\ \sum_{i=1}^N \alpha_i = 1\}$. In essence, $\boldsymbol{\alpha}$ is derived by mapping a query $\mathbf{q}$ and keys K to the attention weights through a softmax function. For better interpretability, Sparsemax [103] and entmax [104] are introduced as alternatives to softmax, producing sparse probability distributions by zeroing out less relevant features. There are mainly two ways of computing the attention weights $\boldsymbol{\alpha}$: (1) additive attention, proposed by [105]: $\alpha(K,\mathbf{q})_i = \mathbf{v}^\top \tanh(W_1 K_i + W_2 \mathbf{q})$ and (2) the scaled dot product alignment function, as in the Transformer network [106]: $\boldsymbol{\alpha}(K,\mathbf{q}) = K^\top \mathbf{q}/\sqrt{m}$ where $\mathbf{v}$, W_1, and W_2 are weight matrices. In practice, diverse attention mechanisms adapt to specific contexts, including cross-attention [50], [51] with separate queries and keys, multi-head attention [52]–[54] for multi-faceted information processing, and hierarchical attention [55]–[59] to capture layered information.

Recent studies have questioned the faithfulness of attention weights as explanatory mechanisms, noting their potential misalignment with actual feature importance [107]–[110]. This discrepancy may stem from biased and noisy training data [111]. To address these concerns, Brunner et al. [112] demonstrated that attention weights can be decomposed to isolate "effective attention" that better captures feature importance, which was later extended by several researchers [113], [114]. Other solutions include SEAT [115], which stabilizes attention weights against training and testing perturbations, and Mohankumar's [116] diversity-driven approach using orthogonalization penalties. Several efforts have focused on incorporating human oversight by supervising attention weights with annotated rationales [117]–[119]. For example, Rigotti et al. [120] align model attention with human concepts through cross-attention between input features and concept embeddings. These studies collectively aim to improve the faithfulness and interpretability of attention mechanisms.

Lasso-based Constraints. The lasso regularization, also known as ℓ_1 norm, plays a fundamental role in achieving feature sparsity through coefficient regularization. When applied to model parameters, it effectively drives some coefficients to zero, naturally performing feature selection. A common approach is to directly impose ℓ_1 norm on the input features [101], [121], [122], which can be formulated as: $\mathcal{C}_{lasso} = \{\boldsymbol{\alpha}(x) \in \mathbb{R}^N : \lVert\boldsymbol{\alpha}(x)\rVert_1 \leq \lambda\}$, where λ is a hyperparameter. However, in the context of neural networks, this straightforward approach suffers from several interpretability issues, such as (1) *hierarchy problem*: due to multi-layer nonlinearity, a feature with a minuscule weight might still impact the output significantly, and (2) *stability*: the selected features may vary across similar samples. To address these issues, recent studies propose to preserve feature relationship and stability across the whole networks. For example, LassoNet [60] adopts a residual structure and impose hierarchical sparsity constraints to ensure direct and consistent feature selection across layers. Several lasso variants are also proposed to adjust the penalization degree of features, such as weighted lasso [61], truncated lasso [62], contextual lasso [63], and group lasso [64].

Subset Sampling Constraints. These constraints, also known as ℓ_0 norm, enhance interpretability by restricting the model to a discrete subset of features. A common formulation is to enforce $\alpha_i(x) \in \{0,1\}$ and $\sum_{i=1}^N \alpha_i(x) \leq K$, allowing at most K features to be selected [65]. Alternatively, one can impose $\sum_{i=1}^N \alpha_i(x) = K$ to sample exactly K features [73]. Comparing to ℓ_1 norm, the subset sampling constraints naturally lead to discrete inference process and avoid the hierarchy problem. However, it is computationally intractable and not differentiable for neural networks, which makes the optimiza-

tion problem particularly challenging. One common approach is to activate each feature with a probability, which can be formulated as: $\mathcal{C}_{\text{subset}} = \{\boldsymbol{\alpha}(x) \in \mathbb{R}^N : \boldsymbol{\alpha}(x) \sim P(x)\}$. In practice, $P(x)$ in SINNs could be modelled an independent Bernoulli distribution [65]–[67] or a parameterized distribution for stochastic gates [68]–[71]. Another method is to use a selection limit K that sets the number of samples taken from a concrete distribution, assigning a hard bound on the number of features selected. Reparameterization gradients, such as Gumbel-softmax [123], [124], are often used to optimize the model, allowing end-to-end training. This method is widely used in text ranking [72], salient mapping [73], [74], and tabular data analysis [75].

Information-theoretic Constraints. The constraints leverage the Information Bottleneck (IB) principle [125] to guide the attributions $\alpha(x)$ toward a compressed yet predictive representation. Formally, one enforces $\mathcal{C}_{\text{info}} = \{\boldsymbol{\alpha}(x) \mid \text{and } I(\boldsymbol{\alpha}(x)x; y) - \beta I(\boldsymbol{\alpha}(x)x; x) \text{ is maximized}\}$, where $I(\cdot;\cdot)$ is the mutual information, and β balances prediction fidelity versus input compression. $\boldsymbol{\alpha}(x)$ could be continuous or discrete, depending on the application. To optimize this objective, it typically introduces a variational lower bound of the IB objective [126] for approximation. Comparing to subset sampling constraints that explicitly choose K features, information-theoretic methods primarily regulate how "much" information is retained. It ensures a concise yet informative representation that captures the essential decision-making information. This approach has been widely used in various domains, such as graph bottleneck for efficient subgraph recognition [76]–[82], word extraction [83], and geometric learning [84].

Discussions. Typically, the generalized coefficient attribution methods focus on constructing instance-wise feature importance [59], [60], [65], [73] by learning the coefficients $\boldsymbol{\alpha}(\boldsymbol{x})$, which varies with the input x. This local interpretability contrasts with global feature importance [42], [67], [127], which is learned across the dataset. The granular, instance-level insights are particularly valuable in high-stakes applications, such as medical diagnosis [71], [101], where understanding the model's decision-making process is crucial.

2) Additive Score Attribution: Comparing to learning interpretable coefficients that modulate feature importance, another line of work focuses on directly learning transformation functions that map features to their final contributions. This distinction is fundamental: instead of separating feature importance (coefficients) from the predictor, additive score attribution unifies them into a single step through feature-specific scoring functions. Formally, the additive score attribution model can be expressed as:

$$f(\boldsymbol{x}; \theta, \phi_0) = \sum_{i=1}^{N} g_i(x_i; \theta_i) + \phi_0, \quad \text{s.t. } g \in \mathcal{C}. \tag{2}$$

where $g_i(x_i; \theta_i) : \mathbb{R} \to \mathbb{R}$ maps the i-th feature to its contribution and satisfies the constraints $\mathcal{C}$ that enforces:

1) *Uniqueness of feature dependence*:

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^N, \quad x_i = x_i' \implies g_i(\boldsymbol{x}; \theta_i) = g_i(\boldsymbol{x}'; \theta_i).$$

This ensures that the contribution of each feature is uniquely determined by its value.

2) *Additional interpretability properties*: one might require monotonicity, or smoothness constraints. For instance,

$$\int g_i(x_i) h_v(x_v) w(x) dx = 0, \quad \forall v \subset i, \ \forall h \in L^2(\mathbb{R}^v)$$

This ensures the pureness of the decomposition [128], where each component function g_i is orthogonal to other sub-functions. $w(x)$ represents the probability density.

The term ϕ_0 is a global bias. Together, these constraints preserve an additive structure while allowing flexible, potentially nonlinear transformations g_i for each feature. As a result, when x_i changes, only the corresponding $g_i(\cdot)$ affects $f(\boldsymbol{x})$, thus achieving *additive self-attribution*.

Neural Additive Model. NAM [129] is a fundamental study that implements the additive score attribution model in neural networks. NAM follows the math form of Generalized Additive Models (GAMs) [130], which decomposes the prediction into a sum of univariate functions. NAM extends this idea to neural networks by introducing a set of scoring functions $g_i(\cdot; \theta_i)$, which uses separate neural networks to directly learns a shape function for each feature.

Rather than directly using traditional MLPs, NAM designs ExU (exp-centered) units to learn jumpy, non-smooth functions in logarithmic space: $h(x) = f(e^w * (x - b))$. This helps capture sharp changes in feature effects that standard neural nets struggle with. Following this line, various studies have proposed extensions to NAM to improve the expressiveness and efficiency of function designs. For example, NODE-GA^{2}M [85] employs the expressiveness of neural oblivious decision ensembles, while GRAND-SLAMIN [86] introduces soft trees based on hyperplane splits. Some approaches focus on tensor-based decompositions to decompose higher-order interactions into a set of low-rank tensors, such as SPAM [87], CAT [131], and NBM [88]. It also inspired interpretable applications in a range of domains: NAISR [89] for 3D shape representation, NA2Q [90] for coordinating collaborative behaviors, and [91] for approximating CNNs.

While eq. (2) focuses on unary (first-order) contributions, many real-world tasks require higher-order feature interactions, which can be interpreted as new, interaction-based features. In this research line, NAM can be expanded to the N-th order, which is expressed as:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \underbrace{g_i(x_i)}_{\text{order 1 (unary)}} + \sum_{j>i}^{N} \underbrace{g_{ij}(x_i, x_j)}_{\text{order 2 (pairwise)}} + \cdots + \underbrace{g_{1\ldots N}(x)}_{\text{order } N}, \tag{3}$$

where $g_i(\cdot)$ captures the contribution of the i-th feature, $g_{ij}(\cdot)$ captures the pairwise interaction between features i and j, and so forth, culminating in a potential N-way interaction term $g_{1\ldots N}(\cdot)$. Models restricted to the up-to-k-th order interactions are often called NA^{k}M, which balances the trade-off between model complexity and interpretability.

While higher-order interactions can improve predictive accuracy, they also lead to exponential growth in model complexity and reduced interpretability. A common strategy to address this challenge is to impose sparsity, which is aligned with feature selection methods. One line of work

enforces sparsity through regularization constraints, for instance, using disentangled group regularizers [92], group LASSO [93], data-dependent kernel expansions [94] or adaptive log-marginal likelihood regularization [95]. Another research line focuses on structural constraints to achieve sparsity. NODE-GA^{2}M [85] utilizes sparse Entmax transformation to produce a one-hot vector, forcing each tree to select only one specific feature. GRAND-SLAMIN [86] proposes a sparse back-propagation approach by progressively suppressing uninformative components through the Smooth-Step function. Finally, some approaches perform post-training pruning: SIAN [96] employs a specialized selection algorithm, while GAMI-Net [97] prunes according to the weak heredity principle. In particular, such post-training pruning methods can be computationally more efficient for higher-order expansions and can readily adapt to existing models without retraining.

Shapley-Based Attribution. Shapley values, rooted in coalition game theory [132], have emerged as a popular way to measure the contribution of individual features (or players) in a predictive model (or coalition). They satisfy four key axioms: (1) *Efficiency*: the sum of individual attributions equals the total model output; (2) *Linearity*: Shapley values respect linear combinations of models; (3) *Nullity*: a feature with no effect on the output receives zero attribution; and (4) *Symmetry*: two features with identical effects on the model receive identical attributions. While most Shapley-based explanations are post-hoc interpretation [1], [133], recent research has sought to embed Shapley values directly into model architectures to encourage unified self-attributing behavior [98], [99].

SHAPNet [99] incorporates Shapley values as latent representations in deep neural networks. It starts from a simplified notion of "presence" and "absence" of features. Let $z \in \{0,1\}^N$ be a binary indicator for the features used in a model input. A baseline or reference vector r represents the "absence" of a feature, giving the following mapping: $\Psi_{x,r}(z) = z \odot x + (1-z) \odot r$. It approximates the Shapley value $g_i(x, f)$ for feature i by summing over all z subsets that do not include i:

$$g_i(x,f)_i \triangleq \frac{1}{N} \sum_{z \in Z_i} w(z)[f(\Psi_{x,r}(z \cup \{i\})) - f(\Psi_{x,r}(z))], \quad (4)$$

where $w(z) = \binom{N-1}{\|z\|_1}^{-1}$. Unlike post-hoc methods, SHAPNet encodes this Shapley mechanism within each network layer as a *Shapley transform* module, providing *layer-wise* attributions. However, it does not directly yield a unified, *model-level* attribution without aggregating across layers, and it inherently relies on the reference vector r. Moreover, while SHAPNet enforces three of the four Shapley axioms (Efficiency, Linearity, and Nullity), it does not fully satisfy Symmetry.

SASANet [98] achieves model-wise Shapley attribution by introducing a *set-based* modeling strategy. Instead of toggling features on or off against a baseline, SASANet formulates its model $f(x) = \sum_{i=1}^{N} g_i(x;\theta) + g_0$, so that each $g_i(x;\theta)$ corresponds to the exact Shapley marginal contribution of feature i across *all* possible subsets and orders:

$$g_i(x;\theta) = \frac{1}{N!} \sum_{O \in \pi(N)} \sum_{k=1}^{N} \mathbb{I}\{O_k = i\} \triangle(x_i, \boldsymbol{x}_{O_{1:k-1}}; \theta), \quad (5)$$

where $\pi(N)$ is the set of all permutations of N, and $\triangle(x_i, x_{O_{1:k-1}}) = f(\boldsymbol{x}_{O_{1:k-1} \cup \{i\}}; \theta) - f(\boldsymbol{x}_{O_{1:k-1}}; \theta)$. A crucial aspect is that SASANet avoids approximation by training an internal *positional distillation* mechanism that converges to the true Shapley value of its own output. It thus satisfies all four Shapley axioms, including Symmetry. Crucially, this eliminates the need for a handpicked baseline r and provides a *model-wide* Shapley attribution rather than layer-specific ones.

3) Hybrid Feature Attribution: To capture more nuanced feature information, some works propose a unified framework that jointly models both coefficient attribution and additive score attribution. This hybrid model leverages both the feature importance weighting mechanism of coefficient attribution and the non-linear feature-specific transformations. The general form can be expressed as:

$$f(\boldsymbol{x}; \theta, g_0) = \sum_{i=1}^{N} a_i(\boldsymbol{x}; \theta) g_i(x_i; \theta_i) + g_0, \quad (6)$$

where $a_i(x;\theta)$ represents the context-dependent coefficient for the i-th feature, and $g_i(x_i;\theta_i)$ is the feature-specific scoring function. This formulation allows for a more expressive attribution mechanism that captures both the global importance of features through coefficients and their individual non-linear contributions through scoring functions.

SSCN [100] is a representative model that adopts this joint feature attribution framework. SSCN models job salary as the weighted average of required skills' values, where $a_i(x;\theta)$ denotes the skill's dominance and $g_i(x_i;\theta_i)$ denotes the skill's value. The model is designed to capture the dynamic influence of skills on salary under various contexts, providing a more nuanced understanding of the labor market. Regarding each skill as a feature for salary prediction, SSCN intrinsically aims at Eq. 6 with a constraint that $\mathcal{C}(a,g,x,r) = \{\forall x, x' \quad x_i = x'_i \rightarrow g(x;\theta_v)_i = g(x';\theta_v)_i\}$ for $i \in \mathcal{N}$. To realize such constraints, SSCN models v with a single-skill network and models d with self-attention mechanism. This structure has high interpretability because the extracted feature x only depends on one input in r. This assures the global physical meaning of it. Abundant analysis are done with the modeled global value and the local weight and make meaningful analysis in the field of labor market analysis. The global value is regarded as skill value in their work and supports the analysis of skill's dynamic influence on salary under various contexts.

B. Function-based Self-Interpretation

Transparent function-based SINNs aim to render neural computations interpretable through structured mathematical forms. By mapping the transformations between inputs, hidden representations, and outputs, these models provide a clear mapping from raw data to final predictions. In this subsection, we discuss two key paradigms for interpretable neural networks: (1) structured compositions of univariate functions (functional decomposition) and (2) multivariate symbolic equations (equation learning), which aim to discover explicit mathematical formulas.

C. Functional Decomposition.

Functional decomposition provides a principled way to break down complex neural networks into interpretable sub-functions, each focusing on specific relationships among input features and hidden concepts. Formally, a neural network can be expressed as

$$f(\boldsymbol{x}) := \big(\Phi_L \circ \Phi_{L-1} \circ \cdots \circ \Phi_2 \circ \Phi_1\big)\boldsymbol{x}, \tag{7}$$

where Φ_i represents the transformation (e.g., layer) at stage i. Unlike standard MLPs with dense, fully connected layers, functional-decomposition approaches enforce structural constraints that reveal how features evolve into increasingly abstract representations.

VOTEN [134] implements functional decomposition via a hierarchical "voting" strategy. Each layer constructs higher-level concepts by applying univariate nonlinear transformations to the current variables, then aggregating them with weights that sum to one. Formally, the j-th concept in layer $l+1$ is computed as:

$$x_j^{(l+1)} = \sum_{k=1}^{n_l} a_{k,j}^{(l)} f_{k,j}^{(l)}(x_k^{(l)}) \quad \text{s.t.} \quad \sum_{k=1}^{n_l} a_{k,j}^{(l)} = 1, \tag{8}$$

where $x_j^{(l+1)}$ is the j-th concept in layer $l+1$, $a_{k,j}^{(l)}$ are the voting weights, and n_l denotes the number of concepts in the l-th layer. By enforcing normalized weights and single-valued transformations, VOTEN's design reveals how each feature evolves into more abstract concepts at successive layers.

Inspired by the Kolmogorov–Arnold representation theorem, KAN [135] generalizes functional decomposition by replacing standard neural connections with learnable univariate spline functions. Formally, the activation value of the j-th neuron in layer $l+1$ is simply summed over all neurons in layer l:

$$x_j^{(l+1)} = \sum_{k=1}^{n_l} \phi_{k,j}^{(l)}(x_k^{(l)}), \tag{9}$$

where $\phi_{k,j}^{(l)}$ represents the univariate spline function connecting the k-th neuron in layer l to the j-th neuron in layer $l+1$. Each $\phi_{k,j}^{(l)}$ is parameterized as:

$$\phi_{k,j}^{(l)}(x) = w_b b(x) + w_s \sum_i c_i B_i(x), \tag{10}$$

where $b(x)$ is a base function (e.g., silu activation, $\text{silu}(x) = x/(1+e^{-x})$), $B_i(x)$ are B-spline basis functions, and w_b, w_s, c_i are learnable parameters. This formulation generalizes the Kolmogorov-Arnold representation theorem while maintaining interpretability through explicit univariate transformations at each edge. The spline parameterization allows for flexible function learning while preserving the ability to visualize and understand each transformation's behavior.

Despite different implementations, both VOTEN and KAN share the core objective of disentangling a high-dimensional function into interpretable sub-functions. Their architectures, however, can be complex—choosing an optimal network structure remains challenging. To tackle this, both works propose pruning strategies to remove redundant components and reveal the most salient features. Looking ahead, automating network design and leveraging progressive growing techniques represent promising directions for simplifying functional decomposition models while preserving their interpretability.

TABLE III: Function-based Self-Interpretation Methods

Methods	Principles	Representative Works
Functional Decomposition	Break down NNs into interpretable sub-functions	VOTEN [134], KAN [135]
Equation Learning	Directly Learning Equations	Architectural: [136], [136], [137] Generative [138]–[143]

D. Equation Learning.

Compared to the functional decomposition paradigm, equation learning focuses on directly learning concise, generalizable, and interpretable mathematical equations from data. It provides a more straightforward and global view on model interpretability, as the learned equations can be directly inspected and analyzed by domain experts. Equation learning has shown particular success in discovering physical laws and scientific relationships. Typically, equation-learning optimization target is formulated as:

$$f^*(\boldsymbol{x}) = \arg\min_{f \in \mathcal{F}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \text{loss}(f(\boldsymbol{x}_i), y_i), \tag{11}$$

where $\mathcal{F}$ denotes the space of interpretable functions (such as polynomials, trigonometric functions, and their combinations), $\mathcal{D}$ is the training dataset consisting of input-output pairs, and $\text{loss}(\cdot)$ is typically mean squared error or similar regression losses. In essence, this is a constrained optimization task aimed at identifying the most interpretable function $f^*(\cdot)$ that fits the data well while maintaining mathematical simplicity and physical plausibility.

Architectural-based methods [136], [137], [144] typically rely on multi-layer feed-forward networks that incorporate specialized units reflecting basic algebraic operators. These units often include unary operations (e.g., identity mapping $f(x) = x$, trigonometric functions sin(x) and cos(x), exponential functions e^x), binary operations (e.g., multiplication $x_1 \cdot x_2$, division x_1/x_2), and linear transformations implemented through weight matrices. For example, EQL$^\div$ [144] successfully learns physics equations like Coulomb's law $F = \frac{1}{4\pi\epsilon_0}\frac{q_1 q_2}{r^2}$ and GINN-LP [137] can discover complex multivariate Laurent polynomials of the form $P = \sum_{i=1}^{n} c_i \prod_{j=1}^{k} V_j^{p_i(j)}$, where $c_i \in \mathbb{R}$ and $p_i(j) \in \mathbb{Z}$ represent coefficients and powers respectively. A key limitation of such designs is their dependence on pre-defined network structures, which can lead to unnecessarily complex equations and reduced flexibility in modeling intricate data relationships. To mitigate these issues, architectural-based models often employ sparsity-inducing constraints [136] (like ℓ_0, ℓ_1 regularization term), systematic architecture selection through validation performance, or adaptive network growth strategies to achieve both conciseness and predictive effectiveness while avoiding overfitting. For instance, GINN-LP uses an iterative growth strategy where the optimization objective combines prediction error and model complexity: $L = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2 + \lambda_1 \sum_{i=1}^{n} |W_i| + \lambda_2 \sum_{i=1}^{n} W_i^2$,

where the last two terms promote sparsity and prevent overfitting respectively.

In contrast, generative-based methods [138]–[140] cast equation learning as a *set-to-sequence* modeling task, wherein a set of numerical data points is transformed into a sequence of symbolic tokens. Two predominant workflows are employed in this domain: (1) *skeleton prediction followed by constant fitting* [140]–[142] and (2) *simultaneous prediction of model structure and constants* [138], [139], [143]. Furthermore, recent studies have shown that these generative models benefit from pretraining on large corpora of symbolic equations, thereby improving their generalization capabilities [140]–[142].

E. Concept-based Self-Interpretation

In this section, we review concept-based methods, which enhance model interpretability by structuring learned representations around human-understandable concepts. These models not only predict outcomes but also articulate how specific concept information within the input contributes to decisions, providing intuitive understanding of the prediction.

Despite different architectural designs, concept-based models could be broadly categorized into two stages: (1) mapping raw inputs to interpretable concepts, and (2) leveraging these concepts for final predictions. The core idea is to design a specialized "bottleneck" layer that encodes human-interpretable concepts, such as specific image features, into an intermediate representation. These concepts could be predefined, learned from data, or a combination of both. This strategy enforces explicit concept learning before prediction, enabling practitioners to analyze both the presence of specific concepts and their influence on the model's prediction.

Formally, concept models predict a target $y \in \mathbb{R}$ from input $x \in \mathbb{R}^d$ through intermediate concepts $c \in \mathbb{R}^k$ using a composite function $f(g(x))$ with two components:

1) $g(\cdot) : \mathbb{R}^d \to \mathbb{R}^k$, mapping raw inputs x into the representation restricted to a concept space c.
2) $f(\cdot) : \mathbb{R}^k \to \mathbb{R}$, mapping the concept-based representation c to the target y, typically through a linear or logic layer.

The model minimizes loss functions L_C for concepts and L_Y for the target, supporting three training schemes: *independent*, where $f(\cdot)$ and $g(\cdot)$ are learned separately; *sequential*, where $g(\cdot)$ is trained first followed by $f(\cdot)$; and *joint* training, optimizing both functions simultaneously with a balancing hyperparameter λ. Most concept models [145], [148] also allows users to directly manipulate the predicted concepts at test time and observe how these changes impact the final prediction. This facilitates counterfactual reasoning ("What if the model didn't think this feature was present?"), and allows users to correct potential model errors.

Recent theoretical analysis by Luyten et al. [165] reveals that concept model performance depends on two critical factors: *concept expressiveness* (the capacity to capture salient information) and *model-aware inductive bias* (coherence between concept set and model architecture). This is also evidenced by the model's reliance on concept annotations, which can be costly and incomplete. Therefore, recent works have focused on enhancing the performance and scalability by designing a more expressive and model-aware concept layer. We categorize these into two key directions: *concept Representation*, *concept organization* and *concept supervision*. Table IV summarizes representative concept-based self-interpretation methods, highlighting their concept layer designs and human intervention strategies.

1) Concept Representation: In concept-based models, the design of concept representation is crucial for regularizing the model's latent space, thereby shaping both interpretability and performance. The Concept Bottleneck Model (CBM) [145] employs direct scalar activation, mapping inputs to scalar concept scores via an activation function $s(\cdot) : \mathbb{R} \to [0, 1]$. This mapping can be hard, applying thresholding $(s(x) = \mathbb{1}_{x>0.5})$, or soft, using a sigmoid activation function $(s(x) = 1/(1 + e^{-x}))$. ENN [147] refines concept mapping with *differentia* neurons that distinguish sub-concept pairs and sub-concept neurons that aggregate these distinctions. Although these methods offer intuitive interpretability, they struggle to capture nuanced concept semantics.

To enhance expressiveness, the Concept Embedding Model (CEM) [148] and IntCEM [159] extend representations into high-dimensional embeddings, where each concept c_i comprises active and inactive states $(\hat{c}_i^+, \hat{c}_i^- \in \mathbb{R}^m)$, later mapped by a sigmoid to determine concept presence. Concept Whitening [146] aligns predefined concepts with latent axes via whitening and orthogonal transformations. Meanwhile, approaches such as PCBM [149], SITE [102], and Concept Transformer [120] adopt a geometric perspective, using similarity scores to project input embeddings onto concept subspaces. Recognizing the inherent ambiguity of real-world concepts, recent studies [151], [152], [160], [161] propose a distributional lens, modeling concepts as prior distributions, for example, Gaussian distributions $p(z_c \mid x) \sim \mathcal{N}(\mu_c, \text{diag}(\Sigma_c))$. This probabilistic framework naturally captures uncertainty and partial presence.

2) Concept Organization: Concept organization encompasses various approaches to modeling relationships between concepts, each offering distinct perspectives on capturing concept dependencies. While vanilla CBM use a flat structure with independent concepts, this simplification fails to capture the rich interdependencies in human conceptual understanding. As shown by Raman et al. [166], such independence assumptions can lead to instability and robustness issues in concept models.

One perspective on concept organization focuses on complementary information channels [152], [153], [158]. Typically, these methods introduce additional channels to encode complementary information to enhance the model's understanding of complex concepts. For example, Side-channel CBM [152] implements this through an auto-regressive architecture, where concept predictions incorporate both input features and previously predicted concepts.

Hierarchical organization offers a valuable perspective in structuring concepts. For example, CF-CBM [157], MICA [164], and ENN [147], arrange concepts across multiple levels, where high-level concepts guide the interpretation of lower-level features. This hierarchical approach is well-suited to domains with natural hierarchical relationships, including medical diagnosis and scene understanding.

TABLE IV: Summary of representative concept-based self-interpretation methods. "S/U" denotes supervised/unsupervised concept learning. "PTM" refers to pre-trained models, while "LLM" and "VLM" denote large language and vision language models, respectively. The papers are sorted by publication year.

Methods	Concept			Human Intervention	Required Architectures
	Organization	Representation	Supervision		
CBM [145]	Flat	Direct Scalar Activation	S	Test-time	-
Concept Whitening [146]	Flat	Whitening Orthogonality	S	-	-
ENN [147]	Hierarchical	Relative Distinction	S	-	-
ConceptTransformer [120]	Flat	Cross-attention	S	Test-time	Transformer
CEM [148]	Flat	Dual Embedding	S	Test-time	-
PCBM [149]	Flat	Similarity	S	Test-time, Global-aware	PTM
Glancenets [150]	Flat	Embedding Alignment	S, U	Test-time	VAE
ProbCBM [151]	Flat	Probalistic Embedding	S	Test-time	-
Side-channel CBM [152]	Flat + Side-channel	Probalistic Embedding	S, U	Test-time	-
CB2M [153]	Flat + 2-fold Memory	Direct Scalar Activation	S	Test-time, Memorized	-
Interactive CBM [154]	Flat	Direct Scalar Activation	S	Test-time, Interactive	-
Label-free CBM [155]	Flat	Similarity	U	Test-time	LLM, VLM
Labo [156]	Flat	Similarity	U	Test-time	LLM, VLM
CF-CBM [157]	Coarse-to-Fine	Similarity	S, U	Test-time	VLM
Res-CBM [158]	Flat + Residual Bank	Similarity	S, U	Test-time	VLM
IntCEM [159]	Flat	Dual Embedding	S	Test-time, Policy-guided	-
Energy-based CBM [160]	Flat	Probalistic Embedding	S	Test-time, Dependency-aware	PTM
Stochastic CBM [161]	Flat	Probalistic Embedding	S	Test-time, Dependency-aware Condidence Region-based	-
Relational CBM [162]	Relational Language	Atom Encoding	S	Test-time, Rule	-
SparseCBM [163]	Flat	Subnetwork Activation	S	Test-time	LLM
MICA [164]	Hierarchical	Multi-level Alignment	S	Test-time	-

Additionally, mathematical frameworks provide a formal method for modeling concept relationships. For instance, Relational CBM (R-CBM) [162] represents concept dependencies through a directed hypergraph, with hyperedges defining relational concept bottlenecks. This structure allows R-CBM to capture dependencies between concepts and tasks that involve multiple entities and relationships.

These diverse perspectives on concept organization offer a rich landscape for structuring concept relationships. The choice of organization method typically depends on specific domain requirements and the nature of the concept relationships that need to be captured.

3) Concept Supervision: Concept supervision includes various methods to define and learn concepts. Each method tackles the challenge of balancing human interpretability with scalable concept learning. We analyze them based on their supervision strategies and concept learning mechanisms. Vanilla CBMs employ human-annotated concept supervision, offering direct interpretability but requiring substantial annotation effort and potentially missing task-relevant information. To address these limitations, several unsupervised approaches have been developed for automatic concept discovery. Post-hoc CBM [149] discovers concepts by applying linear SVMs to pre-trained feature spaces, while other methods [167] identify concepts through data-driven analysis like GradCAM visualization and spectral clustering. Foundation models provide another avenue for concept discovery, with large language models generating concept candidates [149] and CLIP-based models enabling concept-image alignment [155], [156], [168].

Hybrid approaches combine supervised and unsupervised learning to leverage their complementary strengths. Side-channel CBM [152] employs a generative framework that jointly models predefined concepts c and latent concepts z through $p_\theta(c, z|x)$, capturing information beyond known concepts. Res-CBM [158] enhances this approach by developing methods to translate learned residual concepts into human-understandable terms, effectively bridging automated discovery with interpretability. Interactive approaches could optimize human involvement by employing memory-based methods [153], which enable the reuse of previous annotations, and strategic intervention policies [154], which selectively request human input based on uncertainty. Additionally, numerous studies explore inter-concept dependencies to enhance human intervention. Intervention propagation techniques [160], [161], [166] amplify individual annotations through concept relationships, allowing users to prioritize interventions on concepts with high uncertainty. These strategies collectively enhance the efficiency and effectiveness of human-in-the-loop systems in concept-based models.

F. Prototype-based Self-Interpretation

Prototype-based models are inspired by case-based reasoning, a human-like problem-solving paradigm that leverages past experiences to address new challenges. This approach treats previously encountered situations as "representative examples" [169], guiding decision-making for novel cases. As end-to-end trainable neural networks, prototype-based models aim to learn these representative and global examples from data. They automatically identify prototypes within the input feature space and utilize them to make predictions. The

prediction process involves computing the similarity between the input data and the learned prototypes and aggregating these similarities through a transparent function. This methodology not only facilitates accurate and robust predictions but also clearly indicates the decision-making pathway. Formally, this process can be abstracted as:

$$\begin{aligned} z &= f_{enc}(G(x)), \\ \text{sim}(z, p_j) &= \log\left(\frac{(z-p_j)^2+1}{(z-p_j)^2+\epsilon}\right), \\ \hat{y} &= \sum_{j=1}^{N_p} W_j \cdot \text{sim}(z, p_j), \end{aligned} \tag{12}$$

Here, $G(\cdot)$ represents an input processing function (e.g., substructure extraction), and $f_{\text{enc}}(\cdot)$ is an encoder network that maps the processed input to a latent representation z. Each prototype embedding p_j acts as an anchor reference point in the latent space shared with z. N_p denotes the total number of prototypes. Typically, the similarity function $\text{sim}(z, p_j)$ uses a log-activation form, which *decreases monotonically* with the squared L_2 distance between z and p_j. A small constant ϵ is included to avoid numerical instability. It could be replaced by other similarity metrics, such as cosine similarity [170] or Mahalanobis distance [171]. The prediction $\hat{y}$ is obtained by aggregating similarities through a linear weight matrix W. For classification tasks, the output layer often applies a Softmax function to produce a probability distribution over the classes.

A early work PrototypeDNN [172] introduced an L2 distance-based model for image classification, learning prototypes directly in the input feature space through dual clustering regularization. ProtoPNet [173] refined the framework by introducing patch-based processing, class-aware separation costs, and a more direct prototype alignment with training samples. Typically, prototype-based models follow an alternating optimization method [173], which includes three stages: (1) the initial training phase, which aims to learn diverse and representative prototype embeddings in the hidden state, (2) the prototype refinement phase, which maps learned embeddings into the nearest training data samples, and then prunes or merges duplicated prototypes for better human interpretability, and (3) the model fine-tuning phase, which fixed all the parameters of the encoder and prototype embeddings, and learns a sparse weight matrix for higher accuracy. All these stages aim to enhance the model's interpretability by learning meaningful prototype embeddings or aligning them with human-understandable case examples. Following this fundamental framework, recent research has focused on enhancing the interpretability and performance of prototype-based models by enhancing prototype representations and alignment strategies.

1) Prototype Representations.: To enhance the prototype interpretability and performance, recent studies have pursued three distinct but complementary approaches. The first direction optimizes input representation through decomposition strategies. With the inclusion of clustering regularization, the interpretability of prototypes largely depends on the quality of the original input data. ProtoPNet proposes to partition the input images into patches for capturing local fine-grained features. This patch-based methodology has gained widespread adoption in computer vision [174]–[177]. Domain-specific adaptations can be merged for different scenarios. For instance, ProtoryNet [178] implements syntactic decomposition for text. PGIB [78] develops information-theoretic subgraph extraction for graphs. The second direction focuses on prototype embedding methodologies through diverse mathematical frameworks. TesNet [179] establishes Grassmann manifold-based construction to ensure prototype orthogonality and representativeness. Several VAE-based approaches [171], [180], [181] conceptualize prototypes as the center of a Gaussian distribution in the latent space, which forms a probabilistic anchor point for representations. ProtoConcepts [182] extends each single prototype to a set of prototypes by introducing a ball-based prototype geometry. The third direction focuses on building organizational frameworks. ProtoTree [183] implements hierarchical tree-based decision paths. ProtoPool [184] establishes dynamic prototype allocation. ST-ProtoPNet [185] incorporates SVM theory into the prototype learning process, which learns two distinct sets of prototypes by differentiating boundary-distant and boundary-proximate prototypes.

Algorithm 1 Generalized Prototype Learning Algorithm

Input: Training dataset $\mathcal{D}$, number of prototypes m
Output: Encoder f_{enc}, prototypes $\{p_j\}$, classifier weights W

procedure TRAINMODEL
  for epoch $= 1$ to T **do**
    for each batch $\{x_i, y_i\}$ **do**
      $z_i \leftarrow f_{enc}(G(x_i))$ ▷ Encode input
      $\hat{y}_i \leftarrow \sum_j W_j \cdot \text{sim}(z_i, p_j)$ ▷ Comparison
      $L_{total} \leftarrow L_{ce}(\hat{y}_i, y_i) + L_{interp}(z_i, \{p_j\})$
      Update parameters using ∇L_{total}
    end for
    if epoch $\bmod \tau = 0$ **then**
      $p_j \leftarrow$ Search_Examples$(p_j, \mathcal{D})$ for each p_j
    end if
  end for
end procedure
procedure INFERENCE(x)
  return $\sum_j W_j \cdot \text{sim}(f_{enc}(x), p_j)$
end procedure

2) Prototype Alignment.: Prototype alignment with human-interpretable concepts has emerged as a crucial aspect of these models' development. While traditional approaches rely on iterative search to project prototypes onto nearest training samples [173], newer methods have introduced more efficient alternatives, especially for those data types with inherent combinatorial nature. ProtGNN [186] and PGIB [78] employ Monte Carlo Tree Search (MCTS) to navigate this complex search space efficiently. For sequence data, ProSeNet [187] employs greedy beam search algorithm. For set-structured data, SeSRDQN [170] implements a hybrid methodology combining neural decoders with progressive MCTS to balance computational efficiency with alignment accuracy. Human integration has also become increasingly important, with systems like PIP-Net [176] and ProtoPDebug [188] incorporating direct human feedback for prototype refinement. PW-Net [189] takes

a different approach by integrating human-designed prototype layers into pre-trained models. Additionally, optimization strategies have evolved to include cross-class prototype sharing, as demonstrated by ProtoPShare [190], which employs data-dependent similarity metrics to reduce redundancy while maintaining prediction performance.

3) Evaluation and Discussions.: Recent work in prototype interpretability has identified several key challenges that need to be addressed [191]. One major concern is *inconsistency*, where prototypes may represent different features across different images. Another challenge is *instability*, where small input perturbations can lead to significant shifts in prototype activations. To address these issues, researchers have developed various evaluation metrics and frameworks. The SDFA approach [191] aligns similarity structures between deep and shallow layers, while PIXPNET [192] introduces architectural constraints to ensure consistent part localization. Quantitative measures such as PLC, PAC, and PRC are developed to assess how prototypes respond to adversarial modifications [193].

These advancements have ignited extensive research across a variety of domains. Beyond computer vision, as demonstrated in works like [194], [195], this research extends to sequence classification [187], [196], graph-based models [186], [197], fairness issues [198], imitation learning [199], medical diagnosis [200], continual learning [201], and reinforcement learning [189], [202]. This demonstrates the framework's methodological versatility. Meanwhile, there is growing interest in systematically evaluating its interpretability with unified benchmarking and datasets.

G. Rule-based Self-Interpretation

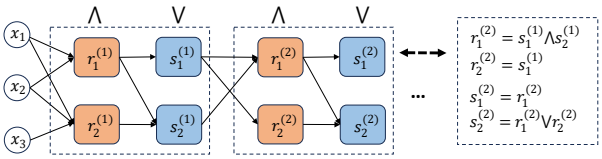
Rule-based methods aim to enhance the interpretability of neural networks by incorporating explicit rules into their architectures. These rules can be either predefined or learned during training, providing clear, human-understandable decision paths. In this section, we categorize rule-based methods into the following four subcategories based on their design principles. We then summarize the key works in each subcategory and discuss their strengths and limitations.

1) Logical Operators as Neurons: The opaqueness of standard neural networks stems from the non-linear functions modeled by neurons, making it challenging to disentangle the inner decision logics. One straightforward approach to improving interpretability is to replace standard neurons with differentiable logical operators. In practice, the logical activation neuron $f_{\text{logic}}(\cdot)$ needs to be differentiable to enable gradient-based optimization. Traditional Boolean logic operators (AND, OR, NOT) have discrete outputs $\{0,1\}$ and thus cannot be directly optimized. To address this issue, the logical neurons are typically realized as a continuous, fuzzy approximation (e.g., t-norm or t-conorm) [203], [204]. For instance, Van et al. [203] defines continuous fuzzy logic operations using product, as shown in Table V. These arithmetic functions naturally approximate Boolean logic (e.g., multiplication is 1 only when both inputs are 1) while remaining differentiable for inputs in [0,1]. ENRL [205] and FINRule [206] further propose relational neural logical operators (such as $\leq$, $\geq$, and $\in$) as a tree structure, where each path represents a rule. A key design challenge lies in specifying or discovering optimal network architectures that integrate logical operators, given the vast space of possible network combinations [207]. Inspired by Disjunctive Normal Form (DNF)'s ability to represent any Boolean formula, MLLP [208] and DR-Net [209] build a hierarchical structure through alternating conjunction and disjunction layers. For conjunction layers, each neuron computes $r_i^{(l)} = \bigwedge_{W_{i,j}^{(l)}=1} s_j^{(l-1)}$, while disjunction layers compute $s_i^{(l+1)} = \bigvee_{W_{i,j}^{(l+1)}=1} r_j^{(l)}$, where $W_{i,j} \in \{0,1\}$ are binary weight matrices. However, this framework faces vanishing gradients in high dimensions since its logical activation functions multiply many terms between 0 and 1, causing gradients to approach zero exponentially. To address this, RRL [210], [211] proposes improved activation functions that convert multiplications into additions via logarithm, with a projection mechanism $\text{Proj}(v) = -1/(-1 + \log(v))$ to maintain the properties of logical operations while preventing gradient vanishing. HyperLogic [212] instead introduces a hyper-network to learn a distribution of network weights, which can be sampled to generate weights for different modules, enabling more flexibility and accuracy. Moreover, neural architecture search can be introduced to automatically discover the optimal rule structure [205]. To achieve minimal memory (4 bits/neuron) and maximal inference speed through bitwise operations, Petersen et al. [204] designs networks where each neuron takes 2 inputs and learns a single logic gate via softmax during training: $f_{\text{logic}}(\boldsymbol{x}) = \sum_{g \in \mathcal{G}} p(g) g(\boldsymbol{x})$, where $\mathcal{G}$ is the set of 16 possible logic gates, and $p(g)$ is the learned probability for each gate. This design has been extended to support CNNs [213] and continuous input feature values [214], enabling broader applications beyond discrete binary inputs.

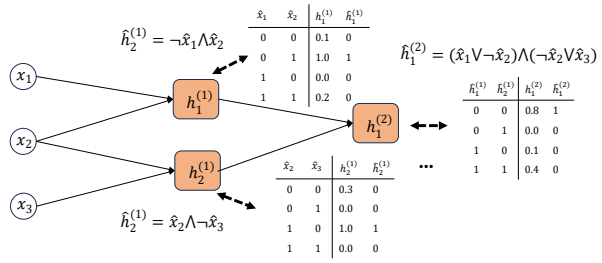
Operators	t-norm		
	Product	**Gödel**	**Łukasiewicz**
$x \wedge y$	$x \cdot y$	$\min(x, y)$	$\max(0, x + y - 1)$
$x \vee y$	$x + y - x \cdot y$	$\max(x, y)$	$\min(1, x + y)$
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \Rightarrow y$	$x \leq y ? 1 : \frac{y}{x}$	$x \leq y ? 1 : y$	$\min(1, 1 - x + y)$

TABLE V: The algebraic operations for fundamental t-norm fuzzy logics [215].

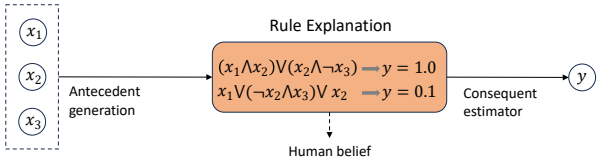
2) Logic-Inspired Constraints: Unlike traditional approaches that directly use logical operators as neurons, some studies focus on extracting logic rules by analyzing the behavior of neural networks through empirical truth tables. These truth tables are constructed from observed data instances and capture the underlying logic of each neuron. However, unlike classical truth tables, they are not necessarily complete and may contain repeated rows due to the nature of empirical data. To enforce sparsity and ensure meaningful logic extraction, the networks are regularized using sparsity constraints or specialized modules that identify feature relevance and logical relationships. Ciravegna et al. [216]–[218] propose a Generalized Logic Explanation Network (LEN) that operates in a k-dimensional concept space, $C = [0,1]^k$, and maps the



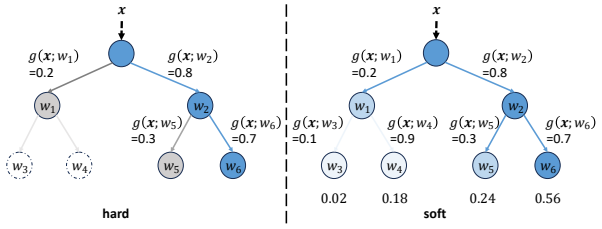
(a) Logical Operators as Neurons [208]



(b) Logic-Inspired Constraints [216]



(c) Rule Generation Network [70]



(d) Interpretable Neural Tree [222]

Fig. 3: Graphical illustrations of rule-based self-interpretation.

data to an r-dimensional rule space, $E = [0,1]^r$, through a function $f(\cdot) : C \to E$. Each coordinate in the rule space corresponds to the activation of a rule. The network generates empirical truth tables from booleanized input-output pairs for each neuron, and these truth tables are combined according to the network structure. The framework also supports different logical relationships through specialized loss functions, such as IF rules, ONLY IF rules, and IFF rules. Information-based constraint approaches [219], [220] introduces an entropy layer that learns to identify relevant concepts through weight-based relevance scores and competition between concepts. Furthermore, DCR [221] extends this framework by employing operand embeddings to capture the fuzzy logic relevance of inputs to outputs. These modules operate on concept embeddings to generate fuzzy logic rules that are then executed on concept truth degrees, maintaining interpretability while handling complex concept relationships.

3) Rule Generation Network: Compared to approaches that rely on logical neurons or binarized weight constraints, rule generation networks utilize standard neural network components while explicitly encoding logical rules as hidden state. These networks typically follow a two-step process: (1) establishing a rule space through either pre-mining or dynamic generation, and (2) learning to select or compose rules for interpretation. For example, Rule-Constrained Network (RCN) [223] works with pre-mined rule sets (e.g., from Eclat or random forests), where each rule contains antecedents and consequents with associated class probabilities. RCN then uses attention mechanisms over these rules to make interpretable predictions. While this provides a controlled rule space, it is limited by the coverage of pre-defined rules.

More recent approaches aim to generate rules dynamically. SELOR [70] implements this by combining a deep antecedent generator and a neural consequent estimator. The antecedent generator learns to predict logical combinations of input features using logical connectives (AND, OR, NOT), while the consequent estimator evaluates the confidence of generated rules by testing them across the training dataset. This eliminates the need for pre-defined rules while ensuring the generated rules remain semantically meaningful. EPR [224] takes a different approach by first using a transformer encoder to detect meaningful phrases from input sentences, then employing a phrase alignment module to identify corresponding phrases between sentences. For each aligned phrase pair, it predicts their logical relationship (entailment, contradiction, or neutral) and combines these local relationships through fuzzy logic formulas to derive the final logical explanation.

4) Interpretable Neural Tree: An interpretable neural tree is a differentiable neural architecture designed to mimic the hierarchical structure of traditional decision trees. This approach leverages neural networks' gradient-based optimization and flexible function approximation while preserving the interpretability of decision trees through rule-based decision paths.

The neural tree simulates a decision tree by representing each internal node with a soft routing function and each leaf node with a decision function. Unlike traditional trees that make discrete decisions, the neural tree employs soft decisions, allowing inputs to be routed through multiple paths in a differentiable manner. This enables end-to-end training using backpropagation, integrating tree-like interpretability with neural network learning capabilities.

Formally, the neural tree is defined as:

$$T = (\mathcal{N}, E),$$

where $\mathcal{N}$ is the set of nodes and E is the set of directed edges. Nodes are categorized as:

- *Internal node* $u \in \mathcal{U}$ with a soft routing function:

$$g_u(\boldsymbol{x}; \theta_u) : \mathbb{R}^d \to [0,1],$$

  determining the probability of routing input $\boldsymbol{x}$ to each child.
- *Leaf node* $w \in \mathcal{W}$ with a decision function:

$$f_w(\boldsymbol{x}; \theta_w) : \mathbb{R}^d \to \mathbb{Y},$$

  providing the final prediction results.

To unify these nodes, let $h_n(\boldsymbol{x})$ denote the output of any node $n \in \mathcal{N}$. The neural tree output is recursively defined as:

$$h_n(\boldsymbol{x}) = \begin{cases} \sum_{c \in \text{child}(n)} g_c(\boldsymbol{x}; \theta_c)\, h_c(\boldsymbol{x}), & \text{if } n \in \mathcal{U}, \\ f_n(\boldsymbol{x}; \theta_n), & \text{if } n \in \mathcal{W}. \end{cases} \tag{13}$$

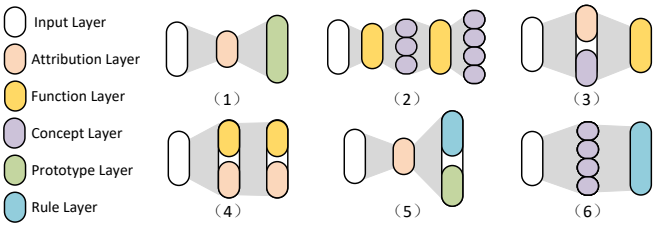


Fig. 4: A graphical illustration of hybrid self-interpretation methods that combine different self-interpretation components.

Finally, the overall model output is $h_{\text{root}}(\boldsymbol{x})$, where root is the root node. This formulation maintains differentiability for end-to-end training while enforcing a decision-tree-style structure for interpretability.

A foundational model in this domain is the Deep Neural Decision Tree (DNDT) [225]. It introduces a soft-binning function for path routing: $g_u(\boldsymbol{x};\theta_u) = \text{softmax}\Big(\big(\boldsymbol{w}_u^\top \boldsymbol{x} + b_u\big)/\tau\Big)$, where $\theta_u = \{\boldsymbol{w}_u, b_u\}$ contains learned cut points that enables differentiable routing while preserving explicit decision boundaries. In DNDT, leaf nodes assign class labels directly: $f_w(\boldsymbol{x};\theta_w) = k$. In contrast, NBDT [222] introduces a more flexible routing mechanism based on the similarity between the input features and learnable node representations: $g_u(\boldsymbol{x};\theta_u) = \text{softmax}\left(\langle \boldsymbol{n}_u, \boldsymbol{x}\rangle\right)$, allowing smoother, softer decision boundaries. At the leaf nodes, NBDT uses learnable distributions: $f_w(\boldsymbol{x};\theta_w) = \sigma(c_w)$, which results in probabilistic class assignments, offering greater flexibility compared to DNDT's direct class labels. These differences make NBDT more robust and adaptable, particularly for complex tasks like image classification. The field took a significant leap forward with ProtoTree [183], which introduced prototypical parts as the basis for decisions. Its routing function is: $g_u(\boldsymbol{x};\theta_u) = \min_{\boldsymbol{z}\,\in\,\text{patches}(\boldsymbol{x})} \|\boldsymbol{z} - \boldsymbol{p}_u\|^2$, where $\theta_u = \{\boldsymbol{p}_u\}$ represents learned prototypes, while the leaf distributions remain trainable as in NBDT. This innovation grounds decisions in concrete visual patterns, making the model's reasoning process more intuitive and human-interpretable. ViT-NeT [226] advances ProtoTree by incorporating transformer architectures and a log-similarity routing mechanism: $g_u(\boldsymbol{x};\theta_u) = \max_{\boldsymbol{z}\in\text{patches}(\boldsymbol{x})} \log(\|\boldsymbol{z} - \boldsymbol{p}_u\|^2 + 1)/(\|\boldsymbol{z} - \boldsymbol{p}_u\|^2 + \epsilon)$. Like ProtoTree, ViT-NeT routes input through nodes based on the similarity between the input and prototypes. However, ViT-NeT's use of transformers enables self-attention, allowing the model to capture both local features and global context. This is in contrast to ProtoTree, which relies on a simpler prototype matching mechanism. ViT-NeT's leaf prediction function combines local and global features: $f_w(\boldsymbol{x};\theta_w) = \sigma\big(c_w^l + c_w^g\big)$, where c_w^l and c_w^g represent local features and global context, respectively. This integration of self-attention and global context enables ViT-NeT to achieve state-of-the-art performance while maintaining an interpretable tree structure.

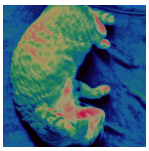
H. Hybrid Self-Interpretation with Combinatorial Methods

While previous sections focused on the design of individual self-interpretation methods, these modules can be integrated together instead of existing in isolation. Recent research demonstrates the feasibility and benefits of this hybrid approach, which provides richer and more comprehensive explanations. As illustrated in Figure 4, various representative hybrid SINNs exemplify this integration:

1) PGIB [78] utilizes an information bottleneck to extract informative local graph substructures, which are then fed into a prototype-based module to capture global examples.
2) VOTEN [134] bridges concept layers with transparent functions, effectively modeling the relationships between low-level and high-level concepts.
3) CAT [131] attributes features into groups and employs a Taylor expansion to approximate the concept-based decision function, thereby modeling complex relationships between different feature groups.
4) Several studies [47]–[49] design explicit attribution functions of the form $l(x) = \boldsymbol{\alpha}(x)x$ between layers to clearly map contributions of each layer.
5) Prototype-based Neural Trees [183], [226] uses prototypical parts as routing functions in neural trees, enabling decisions to be traced to specific image patches and paths.
6) Rule-based studies [219]–[221] ensure transparent predictions by applying logical rules to extracted concepts.

These examples illustrate the potential of hybrid self-interpretation methods to provide richer and more comprehensive explanations. By combining different self-interpretation components, hybrid SINNs can leverage the strengths of individual techniques while mitigating their limitations, resulting in more robust and multifaceted interpretability.

TABLE VI: Explanation Visualizations of Images

Understandable Terms	Visualizations	Explanation Types
Pixels		Spatial Regions, Patches
Patterns		Colors, Materials, Textures, Scenes
Objects		High-Level Object Concepts

IV. Domain-Specific Self-Interpretation

Due to the heterogeneous nature of data types and tasks, SINNs have been tailored to meet specific requirements across different fields. In this section, we review their applications for image, text, graph, and DRL. For each domain, we briefly summarize their popular explanation forms, customized methodologies, and discuss emerging domain-specific topics.

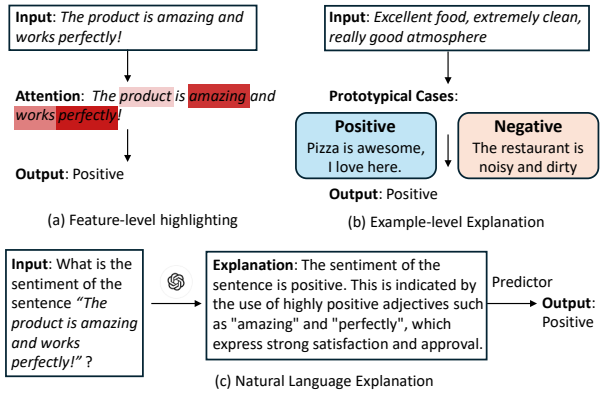


Fig. 5: Examples for text explanations.

A. Image Data

In the field of SINNs for image data, pixel-level, pattern-level, and object-level explanations represent different granularities of interpretability. As shown in Table VI, each offers unique visual insights and serves distinct applications.

Visual explanation granularity vary in terms of the level of detail they provide. *Pixel explanations* group pixels into cohesive regions that capture basic visual units while maintaining spatial coherence. These regions can be pre-defined by fixed grids, such as patches in prototype-based methods [176], [183]–[185], or they can be learned dynamically using masking techniques like attention [227] or information bottleneck [43]. Dynamic regions offer greater adaptability to complex visual patterns and often incorporate sparsity and semantic regularization for better interpretability. *Pattern explanations* identify recurring visual motifs like textures or colors instead of focusing on individual pixels. Concept-based methods [145], [148], [151] represent images using pattern-level attributes derived from human annotations or learned directly from data through VLMs [155], [156]. *Object-level explanations* emphasize the relationships between distinct objects within an image. This approach is particularly useful for tasks such as object detection and visual question answering. Methods [228], [229] use pre-trained models to extract object boundaries and learn symbolic relationships between objects, providing insights into the structural composition and interactions within the visual content.

Interpretable CNN filter is a popular approach for built-in explanations for images. It is a small matrix of weights that are convolved with the input image to produce a feature map. These filters are designed to capture specific object parts or visual patterns, aligning the learning process with human-understandable concepts. For example, ICNN [230] groups filters into object parts within ball-like areas, while ICCNN [231] extends it to represent both object parts with specific shapes and image regions without clear structures.. The class-specific filters in [232] further disentangle the relationship between filters and classes, allowing filters to be shared by similar classes and enabling a class to have a variable number of filters. PICNN [233] further considers the probabilistic filter-class relationships for more nuanced interpretability.

TABLE VII: Explanation forms of text data

Understandable Terms	Explanation Type	Representative Works
Feature	words, n-grams, phrases, sub-sentence,	[55], [73], [122], [234]–[238]
Example	Prototypical cases	[178], [187], [239]
Natural Language	explain-then-predict, predict-and-explain	[240]–[250]

B. Text Data

In this subsection, we review SINN researches for text data. Figure 5 and Table VII present its explanation examples and types, including feature, example, and natural language explanations, each at different abstraction levels.

Feature-level explanations identify critical input text units (e.g., words, subsequences) that influence model decisions. Techniques like sampling [122], [234], attention mechanisms [55], [235], [236] and sparsity-inducing constraints [73], [237] can highlight influential elements in the input. These methods are effective at identifying local textual input that drive predictions. To provide higher-level coherence, they can combine external knowledge bases [238] for annotation.

Example-level explanations provide a more abstract perspective by grounding model decisions in representative examples. For instance, ProSeNet [187] determines sentiment by comparing a new review with prototypes of both positive and negative reviews, then aggregates their contributions into the final score. Beyond document-level examples [187], [239], ProtoryNet [178] refines predictions by leveraging a trajectory of sentence-level prototypes that capture nuanced sentiments.

Natural language explanation have emerged as an accessible, self-interpretable way to clarify text model predictions, especially with generative methods. They provide textual justifications that mimic the reasoning style of human experts and end users [16]. One strategy is *explain-then-predict* [243], [251], where an intermediate explanation is generated first and then guides the prediction [240]–[242], [244]–[246]. Another strategy is *predict-and-explain* [247]–[250], which generates predictions with explanations in an integrated manner. With the rise of LLMs, natural language explanations have gained more popularity through techniques like Chain-of-Thought [252] and Tree-of-Thought [253]. Despite their promise, these self-interpretations often face reliability and faithfulness issues [16], [254], [255]. The key challenges lie in generating explanations that accurately align with human reasoning and transparently using them for inference.

C. Graph Data

In recent years, GNNs have been widely used in many critical domains, such as molecular structures and financial systems. Unlike image and text data, graphs have irregular structures and complex relationships, which pose unique challenges in inherent interpretation. In this context, we review the existing self-interpretable GNNs from three perspectives: local subgraph extraction, global graph patterns, and applications.

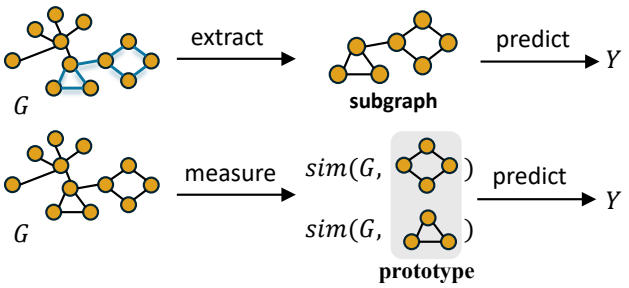


Fig. 6: Framework of self-interpretable GNNs.

Local subgraph extraction identifies the most informative subgraph for each instance to explain predictions. One approach uses the graph information bottleneck [256], [257] to extract compressed subgraphs that retain key information. Various strategies for mutual information estimation and edge sampling [82], [84], [258] enhance stability and efficiency. Another approach imposes structural constraints to derive subgraphs with meaningful roles. These methods focus on similarity to labeled nodes [259], alignment with pre-defined patterns [260], [261], or invariant causal structures [262].

Global graph patterns provide a holistic view of GNN behavior by identifying key patterns across instances. Several works [78], [186], [197] rely on prototype learning to capture class-specific patterns. Especially, PGIB [78] discovers subgraph-level patterns through the information theory, while GIP [197] clusters nodes to match the global interactive instances. Furthermore, GDM [263] captures evolving patterns during training via distribution matching, capturing more more nuanced patterns.

Applications. Self-interpretable GNNs have been applied across various areas, including graph condensation [79], temporal graphs [81], [264], [265], knowledge graphs [264], [266], anomaly detection [80], heterogeneous graphs [267], pre-trained [268] and self-supervised setting [269]. Beyond graph-specific tasks, its applications also include drug discovery [270], brain network analysis [271], chemical property prediction [272], and fake news detection [52]. These applications demonstrate that self-interpretation mechanisms could reveal key domain insights, such as molecular substructures, anomalous temporal patterns, and entity relationships, which prompts further exploration in scientific and industrial domains.

D. Deep Reinforcement Learning

Self-interpretation is crucial in DRL to ensure that agents' transparent and accountable decisions. In this subsection, we review recent advancements in SINNs for DRL from three aspects: influencing environment perception, and value function decomposition, interpretable policy representation.

Influencing environment perception explains an agent's decisions by attributing importance to specific features or regions in the input. It is similar to feature attribution methods discussed in Section III-A, but tailored to different data types and applications. Some approaches focus on visualizing low-level feature importance using superpixels, attention mechanisms, or saliency maps [273]–[275], while others emphasize self-supervised feature discovery to dynamically identify relevant features [276], [277].

Value function decomposition explains decisions by breaking down value functions into interpretable components, clarifying how actions lead to specific outcomes over time. Several methods [90], [278], [279] redistribute rewards to reveal causal links between actions and their effects. Scenario-based approaches [189], [202], [273] use prototypes or key-value memories to illustrate typical agent behaviors. Some works [127], [280] identify critical decision points or generate contrastive explanations through belief maps and time-step analysis, providing insights into long-term strategies.

Interpretable policy representation aims at replacing opaque neural policies with human-readable symbolic or logical structures to enhance the transparency and verifiability. For instance, NUDGE [281] involves a two-step process: training a neural agent and using it to guide the selection and refinement of weighted logical rules, resulting in a policy that is both effective and interpretable. ESPL [282] and DILP [283] differ this by using symbolic operators within networks to directly learn symbolic expressions.

V. Quantitative Evaluation Metrics

In this section, we summarize the existing quantitative evaluation metrics for SINNs, and discuss their differences with post-hoc interpretability evaluation metrics.

1) **Model Performance.** SINNs, as neural networks with built-in explanations, are evaluated using traditional metrics like accuracy, efficiency, and generalizability. Unlike post-hoc methods, SINNs aim to maintain performance comparable to black-box models. Model performance can also be assessed at various stages to gauge the effectiveness of interpretable components, such as human intervention [145], [159]. Additionally, some studies consider the extra efficiency costs for explanation generation, including substructure extraction [186] and prototype alignment [170].
2) **Explanation Evaluation.** Explanation evaluation metrics assess different aspects of model interpretability: (1) *Stability* refers to the consistency of explanations across similar inputs or geometric transformations [45], [102]. Recent work on prototype interpretability has introduced metrics like SDFA [191], PIXPNET [192], and adversarial robustness measures like PLC, PAC, and PRC [193]. (2) *Faithfulness* evaluates how well the model's explanations reflect true feature importance, often by comparing them to ground truth [148], [157]. (3) *Effectiveness* measures the contribution of relevant explanations to decision-making across samples, with metrics such as Normalized Effective Concept Number [168] and ConceptSHAP [284]. (4) *Completeness* assesses a model's ability to capture all relevant concepts, using metrics like completeness scores [284] and error rates for concept conformity [285]. (5) *Interrelationship* evaluates how explanation units relate to each other, considering diversity [197] and dependency [286].
3) **Human-centric Feedback.** Human feedback is crucial for evaluating the interpretability of SINNs. User studies, questionnaires, and interviews [287] can assess the quality

of explanations and how well users understand the model's decision-making. For instance, Groundability and Factuality metrics [156] assess how closely the concepts align with human understanding, based on annotators' selection of images that best represent the concepts.

The distinction between SINNs and post-hoc methods lies in the inherent nature of their explanations. SINNs provide explanations directly from the model, with evaluation metrics focusing on consistency, human interpretability, and accuracy. In contrast, post-hoc methods generate separate explanations, with metrics centered on fidelity and relevance to the pre-trained model's predictions.

VI. Discussions

This section explores the relationship between post-hoc and self-interpretation methods and suggests potential directions.

A. Relationship between post-hoc and self-interpretation

Share the forms of interpretation. Both self-interpretation and post-hoc methods use similar attribution techniques, such as linear aggregation and Shapley value attribution. Self-explaining models, like SENNs [45], resemble post-hoc methods like LIME [46] and gradient-based approaches [44]. As shown in SHAP [1], these methods can be unified under the additive attribution framework, which applies to both categories. Despite differences in how explanations are generated, post-hoc and self-interpretation methods can inspire each other and share interpretability insights.

Self-interpretation can be used for post-hoc interpretation. Recent works also utilize model distillation to transform black-box models into interpretable surrogates [288]–[290]. The distillation process often involves mapping the original model's internal representations to hierarchical decision structures, effectively decomposing complex behaviors into more transparent decision paths.

Self-interpretation can be used to diagnose post-hoc interpretation. Despite their limited applicability compared to post-hoc methods, self-interpretation models can serve as valuable diagnostic tools. By training models with provably correct self-interpretation (such as Neural Additive Models [129]), we can establish reliable ground truth interpretations. These can then be used to evaluate post-hoc interpretation methods by comparing their outputs against the known correct interpretations [98], provided both methods produce comparable forms of interpretation.

Post-hoc interpretation can be also used for self-interpretation models. Although self-interpretation models aim for inherent transparency, they often include approximated or opaque components. Post-hoc methods can serve as diagnostic tools to assess these models' limitations and identify potential deficiencies. Additionally, aligning post-hoc and self-interpretation results provides empirical validation, especially given the imperfections of current self-interpretation approaches. For example, regularization methods may not fully ensure interpretability, making post-hoc verification beneficial.

B. Potential Future Directions

Hybrid explanation forms. As discussed in Section III-H, hierarchical explanations can provide a more comprehensive understanding by moving beyond single-level interpretations. For instance, in image classification, a hierarchical approach might first identify relevant objects and then explain how they interact through rules or global concepts to lead to the final prediction [183], [226]. Combining self-interpretation and post-hoc methods can further enhance explanations, leveraging the intrinsic interpretability of self-interpretation models and the flexibility of post-hoc techniques.

Systematic evaluation metrics and benchmarks. A major challenge for SINNs is the lack of enough well-established evaluation metrics [286] and benchmarks [193]. Future research should focus on developing systematic metrics tailored to SINNs, considering the unique characteristics of each category and providing a comprehensive assessment of interpretability. Standardized benchmarks would also facilitate model comparison and promote research reproducibility.

Leveraging LLMs for enhanced SINNs. The expressiveness of LLMs presents new opportunities for enhancing SINNs. By integrating LLMs, we can leverage their pre-trained knowledge to improve SINN interpretability. For example, LLMs could generate concepts or rules for SINNs [168], provide additional context for model explanations, or reduce the need for manual labeling and data cleaning [156]. Furthermore, combining LLMs with SINNs could strengthen the generalization of model explanations, as LLMs excel at capturing complex patterns and relationships within the data.

Multi-modal Self-interpretation. While current SINNs primarily focus on single modalities, there is significant potential for progress in multi-modal contexts. Future research could explore inherently interpretable architectures that model cross-modal interactions and provide coherent, unified explanations across modalities [291]. This could lead to more comprehensive e explanations in multi-modal applications, such as autonomous driving and medical imaging.

VII. Conclusion

In this paper, we presented a comprehensive survey of self-interpretable neural networks, focusing on attribution-based, function-based, concept-based, prototype-based, and rule-based approaches. We provided concrete, visualized examples of model explanations and discussed their applicability across various domains, including image, text, graph, and deep reinforcement learning. Additionally, we summarized the quantitative evaluation metrics for SINNs, highlighting their differences from post-hoc metrics. We also explored the relationship between post-hoc and self-interpretation methods, emphasizing the potential for mutual benefit. We hope this survey will inspire future research in the field of self-interpretable neural networks and contribute to the development of more interpretable machine learning models.

Acknowledgment

This work is partly supported by the National Key Research and Development Program of China (No. 2023YFF0725001),

the National Natural Science Foundation of China (No. 62306255, 92370204, 62176014), the Natural Science Foundation of Guangdong Province (No. 2024A1515011839), the Fundamental Research Project of Guangzhou (No. 2024A04J4233), the Guangzhou-HKUST(GZ) Joint Funding Program (No.2023A03J0008), and the Education Bureau of Guangzhou Municipality.

REFERENCES

[1] M. Scott, L. Su-In *et al.*, "A unified approach to interpreting model predictions," *NeurIPS*, vol. 30, pp. 4765–4774, 2017.
[2] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *ICML*, 2017, pp. 3145–3153.
[3] A. Ghorbani, A. Abid, and J. Zou, "Interpretation of neural networks is fragile," in *AAAI*, 2019, pp. 3681–3688.
[4] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," in *AIES*, 2020, pp. 180–186.
[5] D. Slack, A. Hilgard, S. Singh, and H. Lakkaraju, "Reliable post hoc explanations: Modeling uncertainty in explainability," *NeurIPS*, pp. 9391–9404, 2021.
[6] J. Li, M. Pang, Y. Dong, J. Jia, and B. Wang, "Graph neural network explanations are fragile," in *ICML*, 2024.
[7] T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki, "The dangers of post-hoc interpretability: Unjustified counterfactual explanations," in *IJCAI*, 2019, pp. 2801–2807.
[8] C. Frye, C. Rowat, and I. Feige, "Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability," *NeurIPS*, vol. 33, pp. 1229–1239, 2020.
[9] G. Yang, Q. Ye, and J. Xia, "Unbox the black-box for the medical explainable ai via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond," *Inf. Fusion*, pp. 29–52, 2022.
[10] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *J. Artif. Intell. Res.*, pp. 245–317, 2021.
[11] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Inf. Fusion*, pp. 82–115, 2020.
[12] B. Chander, C. John, L. Warrier, and K. Gopalakrishnan, "Toward trustworthy artificial intelligence (tai) in the context of explainability and robustness," *ACM Comput. Surv.*, 2024.
[13] R. Dwivedi, D. Dave, H. Naik, S. Singhal, R. Omer, P. Patel, B. Qian, Z. Wen, T. Shah, G. Morgan *et al.*, "Explainable ai (xai): Core ideas, techniques, and solutions," *ACM Comput. Surv.*, pp. 1–33, 2023.
[14] Y. Gao, S. Gu, J. Jiang, S. R. Hong, D. Yu, and L. Zhao, "Going beyond xai: A systematic survey for explanation-guided learning," *ACM Comput. Surv.*, pp. 1–39, 2024.
[15] A. Madsen, S. Reddy, and S. Chandar, "Post-hoc interpretability for neural nlp: A survey," *ACM Comput. Surv.*, pp. 1–42, 2022.
[16] N. Calderon and R. Reichart, "On behalf of the stakeholders: Trends in nlp model interpretability in the era of llms," *arXiv preprint arXiv:2407.19200*, 2024.
[17] Q. Lyu, M. Apidianaki, and C. Callison-Burch, "Towards faithful model explanation in nlp: A survey," *Comput. Linguist.*, pp. 1–67, 2024.
[18] S. Luo, H. Ivison, S. C. Han, and J. Poon, "Local interpretations for explainable natural language processing: A survey," *ACM Comput. Surv.*, pp. 1–36, 2024.
[19] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, and P. Sen, "A survey of the state of explainable ai for natural language processing," in *AACL*, 2020.
[20] J. E. Zini and M. Awad, "On the explainability of natural language processing deep models," *ACM Comput. Surv.*, pp. 1–31, 2022.
[21] C. Agarwal, M. Zitnik, and H. Lakkaraju, "Towards a rigorous theoretical analysis and evaluation of gnn explanations," *arXiv preprint arXiv:2106.09078*, 2021.
[22] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *TPAMI*, pp. 5782–5799, 2022.
[23] J. Kakkad, J. Jannu, K. Sharma, C. Aggarwal, and S. Medya, "A survey on explainability of graph neural networks," *Bull. IEEE Comp. Soc. Tech. Comm. Data Eng.*, pp. 35–63, 2023.
[24] Y. Qing, S. Liu, J. Song, H. Wang, and M. Song, "A survey on explainable reinforcement learning: Concepts, algorithms, challenges," *arXiv preprint arXiv:2211.06665*, 2022.
[25] G. A. Vouros, "Explainable deep reinforcement learning: state of the art and challenges," *ACM Comput. Surv.*, pp. 1–39, 2022.
[26] T. Hickling, A. Zenati, N. Aouf, and P. Spencer, "Explainability in deep reinforcement learning: A review into current methods and applications," *ACM Comput. Surv.*, pp. 1–35, 2023.
[27] S. Milani, N. Topin, M. Veloso, and F. Fang, "Explainable reinforcement learning: A survey and comparative review," *ACM Comput. Surv.*, pp. 1–36, 2024.
[28] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nat. Mach. Intell.*, pp. 206–215, 2019.
[29] M. Al-Shedivat, A. Dubey, and E. Xing, "Contextual explanation networks," *JMLR*, pp. 1–44, 2020.
[30] M. Moshkovitz, S. Dasgupta, C. Rashtchian, and N. Frost, "Explainable k-means and k-medians clustering," in *ICML*, 2020, pp. 7055–7065.
[31] Z. Qin, L. Yang, Q. Wang, Y. Han, and Q. Hu, "Reliable and interpretable personalized federated learning," in *CVPR*, 2023, pp. 20 422–20 431.
[32] W. Norcliffe-Brown, S. Vafeias, and S. Parisot, "Learning conditioned graph structures for interpretable visual question answering," *NeurIPS*, vol. 31, 2018.
[33] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, pp. 1–38, 2019.
[34] M. Langer, D. Oster, T. Speith, H. Hermanns, L. Kästner, E. Schmidt, A. Sesing, and K. Baum, "What do we want from explainable artificial intelligence (xai)?–a stakeholder perspective on xai and a conceptual model guiding interdisciplinary xai research," *Artif. Intell.*, 2021.
[35] S. T. Mueller, R. R. Hoffman, W. Clancey, A. Emrey, and G. Klein, "Explanation in human-ai systems: A literature meta-review, synopsis of key ideas and publications, and bibliography for explainable ai," *arXiv preprint arXiv:1902.01876*, 2019.
[36] W. Saeed and C. Omlin, "Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities," *Knowl. Based Syst.*, 2023.
[37] G. Vilone and L. Longo, "Notions of explainability and evaluation approaches for explainable artificial intelligence," *Inf. Fusion*, pp. 89–106, 2021.
[38] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, "Interpretable machine learning: Fundamental principles and 10 grand challenges," *Stat. Surv.*, pp. 1–85, 2022.
[39] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang, "A survey on neural network interpretability," *IEEE Trans. Emerg. Topics Comput. Intell.*, pp. 726–742, 2021.
[40] E. Cambria, L. Malandri, F. Mercorio, M. Mezzanzanica, and N. Nobani, "A survey on xai and natural language explanations," *Inf. Process. Manag.*, 2023.
[41] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du, "Explainability for large language models: A survey," *ACM Trans. Intell. Syst. Technol.*, pp. 1–38, 2024.
[42] D. Afchar, V. Guigue, and R. Hennequin, "Towards rigorous interpretations: a formalisation of feature attribution," in *ICML*, 2021, pp. 76–86.
[43] K. Schulz, L. Sixt, F. Tombari, and T. Landgraf, "Restricting the flow: Information bottlenecks for attribution," in *ICLR*, 2020.
[44] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *ICML*, 2017, pp. 3319–3328.
[45] D. Alvarez Melis and T. Jaakkola, "Towards robust interpretability with self-explaining neural networks," *NeurIPS*, vol. 31, 2018.
[46] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *SIGKDD*, 2016, pp. 1135–1144.
[47] M. Bohle, M. Fritz, and B. Schiele, "Convolutional dynamic alignment networks for interpretable classifications," in *CVPR*, 2021, pp. 10 029–10 038.
[48] M. Böhle, N. Singh, M. Fritz, and B. Schiele, "B-cos alignment for inherently interpretable cnns and vision transformers," *TPAMI*, 2024.
[49] M. Böhle, M. Fritz, and B. Schiele, "B-cos networks: Alignment is all we need for interpretability," in *CVPR*, 2022, pp. 10 329–10 338.
[50] Y. Zhang, M. Jiang, and Q. Zhao, "Query and attention augmentation for knowledge-based explainable reasoning," in *CVPR*, 2022, pp. 15 576–15 585.
[51] P. Bai, F. Miljković, B. John, and H. Lu, "Interpretable bilinear attention network with domain adaptation improves drug–target prediction," *Nat. Mach. Intell.*, vol. 5, no. 2, pp. 126–136, 2023.

[52] J. Zhu, C. Gao, Z. Yin, X. Li, and J. Kurths, "Propagation structure-aware graph transformer for robust and interpretable fake news detection," in *KDD*, 2024, pp. 4652–4663.

[53] C.-H. Yeh, Y.-C. Fan, and W.-C. Peng, "Interpretable multi-task learning for product quality prediction with attention mechanism," in *ICDE*, 2019, pp. 1910–1921.

[54] L. Ren, G. Yu, J. Wang, L. Liu, C. Domeniconi, and X. Zhang, "A diversified attention model for interpretable multiple clusterings," *TKDE*, vol. 35, no. 9, pp. 8852–8864, 2022.

[55] L. Luo, X. Ao, F. Pan, J. Wang, T. Zhao, N. Yu, and Q. He, "Beyond polarity: Interpretable financial sentiment analysis with hierarchical query-driven attention," in *IJCAI*, 2018, pp. 4244–4250.

[56] L. Wu, Y. Rao, X. Yang, W. Wang, and A. Nazir, "Evidence-aware hierarchical interactive attention networks for explainable claim verification," in *IJCAI*, 2021, pp. 1388–1394.

[57] X. Xu, Z. Wang, C. Deng, H. Yuan, and S. Ji, "Towards improved and interpretable deep metric learning via attentive grouping," *TPAMI*, vol. 45, no. 1, pp. 1189–1200, 2022.

[58] Y. Zhu, D. Xi, B. Song, F. Zhuang, S. Chen, X. Gu, and Q. He, "Modeling users' behavior sequences with hierarchical explainable network for cross-domain fraud detection," in *WWW*, 2020, pp. 928–938.

[59] S. O. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *AAAI*, vol. 35, no. 8, 2021, pp. 6679–6687.

[60] I. Lemhadri, F. Ruan, L. Abraham, and R. Tibshirani, "Lassonet: A neural network with feature sparsity," *JMLR*, vol. 22, no. 127, pp. 1–29, 2021.

[61] S. He, X. Li, V. Sivakumar, and A. Banerjee, "Interpretable predictive modeling for climate variables with weighted lasso," in *AAAI*, 2019, pp. 1385–1392.

[62] X. Zhang, D. Lee, and S. Wang, "Comprehensive attribution: Inherently explainable vision model with feature detector," in *ECCV*, 2025, pp. 196–213.

[63] R. Thompson, A. Dezfouli, and R. Kohn, "The contextual lasso: Sparse linear models via deep neural networks," *NeurIPS*, pp. 19 940–19 961, 2023.

[64] V. C. Dinh and L. S. Ho, "Consistent feature selection for analytic deep neural networks," *NeurIPS*, pp. 2420–2431, 2020.

[65] J. Yoon, J. Jordon, and M. Van der Schaar, "Invase: Instance-wise variable selection using neural networks," in *ICLR*, 2018.

[66] N. Jethani, M. Sudarshan, Y. Aphinyanaphongs, and R. Ranganath, "Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations," in *AISTATS*, 2021, pp. 1459–1467.

[67] X. Jiang, A. Margeloiu, N. Simidjievski, and M. Jamnik, "Protogate: Prototype-based neural networks with global-to-local feature selection for tabular biomedical data," in *ICML*, 2024.

[68] J. Yang, O. Lindenbaum, and Y. Kluger, "Locally sparse neural networks for tabular biomedical data," in *ICML*, 2022, pp. 25 123–25 153.

[69] Y. Yamada, O. Lindenbaum, S. Negahban, and Y. Kluger, "Feature selection using stochastic gates," in *ICML*, 2020, pp. 10 648–10 659.

[70] S. Lee, X. Wang, S. Han, X. Yi, X. Xie, and M. Cha, "Self-explaining deep models with logic rule reasoning," *NeurIPS*, pp. 3203–3216, 2022.

[71] W. Li, X. Feng, H. An, X. Y. Ng, and Y.-J. Zhang, "Mri reconstruction with interpretable pixel-wise operations using reinforcement learning," in *AAAI*, 2020, pp. 792–799.

[72] J. Leonhardt, K. Rudra, and A. Anand, "Extractive explanations for interpretable text ranking," *TOIS*, pp. 1–31, 2023.

[73] J. Chen, L. Song, M. Wainwright, and M. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *ICML*, 2018, pp. 883–892.

[74] M. F. Balın, A. Abid, and J. Zou, "Concrete autoencoders: Differentiable feature selection and reconstruction," in *ICML*, 2019, pp. 444–453.

[75] J. Si, W. Y. Cheng, M. Cooper, and R. G. Krishnan, "Interpretabnet: Distilling predictive signals from tabular data by salient feature interpretation," in *ICML*, 2024.

[76] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, and R. He, "Graph information bottleneck for subgraph recognition," in *ICLR*, 2020.

[77] ——, "Recognizing predictive substructures with subgraph information bottleneck," *TPAMI*, vol. 46, no. 3, pp. 1650–1663, 2021.

[78] S. Seo, S. Kim, and C. Park, "Interpretable prototype-based graph information bottleneck," *NeurIPS*, vol. 36, 2024.

[79] J. Fang, X. Li, Y. Sui, Y. Gao, G. Zhang, K. Wang, X. Wang, and X. He, "Exgc: Bridging efficiency and explainability in graph condensation," in *WWW*, 2024, pp. 721–732.

[80] Y. Liu, K. Ding, Q. Lu, F. Li, L. Y. Zhang, and S. Pan, "Towards self-interpretable graph-level anomaly detection," *NeurIPS*, 2024.

[81] J. Chen and R. Ying, "Tempme: Towards the explainability of temporal graph neural networks via motif discovery," *NeurIPS*, 2024.

[82] S. Miao, M. Liu, and P. Li, "Interpretable and generalizable graph learning via stochastic attention mechanism," in *ICML*, 2022, pp. 15 524–15 543.

[83] A. Sekhon, H. Chen, A. Shrivastava, Z. Wang, Y. Ji, and Y. Qi, "Improving interpretability via explicit word interaction graph layer," in *AAAI*, 2023, pp. 13 528–13 537.

[84] S. Miao, Y. Luo, M. Liu, and P. Li, "Interpretable geometric deep learning via learnable randomness injection," in *ICLR*, 2023.

[85] C.-H. Chang, R. Caruana, and A. Goldenberg, "Node-gam: Neural generalized additive model for interpretable deep learning," in *ICLR*, 2024.

[86] S. Ibrahim, G. Afriat, K. Behdin, and R. Mazumder, "Grand-slamin'interpretable additive modeling with structural constraints," *NeurIPS*, vol. 36, 2024.

[87] A. Dubey, F. Radenovic, and D. Mahajan, "Scalable interpretability via polynomials," *NeurIPS*, vol. 35, pp. 36 748–36 761, 2022.

[88] F. Radenovic, A. Dubey, and D. Mahajan, "Neural basis models for interpretability," *NeurIPS*, vol. 35, pp. 8414–8426, 2022.

[89] Y. Jiao, C. Zdanski, J. Kimbell, A. Prince, C. Worden, S. Kirse, C. Rutter, B. Shields, W. Dunn, J. Mahmud *et al.*, "Naisr: A 3d neural additive model for interpretable shape representation," in *ICLR*, 2024.

[90] Z. Liu, Y. Zhu, and C. Chen, "NA^2q: Neural attention additive model for interpretable multi-agent q-learning," in *ICML*, 2023, pp. 22 539–22 558.

[91] W. Brendel and M. Bethge, "Approximating cnns with bag-of-local-features models works surprisingly well on imagenet," in *ICLR*, 2018.

[92] M. Tsang, H. Liu, S. Purushotham, P. Murali, and Y. Liu, "Neural interaction transparency (nit): Disentangling learned interactions for improved interpretability," *NeurIPS*, 2018.

[93] S. Xu, Z. Bu, P. Chaudhari, and I. J. Barnett, "Sparse neural additive model: Interpretable deep learning with feature selection via group sparsity," in *ICLR*, 2023.

[94] G. Liu, H. Chen, and H. Huang, "Sparse shrunk additive models," in *ICML*, 2020, pp. 6194–6204.

[95] K. Bouchiat, A. Immer, H. Yèche, G. Ratsch, and V. Fortuin, "Improving neural additive models with bayesian principles," in *ICML*, 2023.

[96] J. Enouen and Y. Liu, "Sparse interaction additive networks via feature interaction detection and sparse selection," *NeurIPS*, vol. 35, pp. 13 908–13 920, 2022.

[97] Z. Yang, A. Zhang, and A. Sudjianto, "Gami-net: An explainable neural network based on generalized additive models with structured interactions," *Pattern Recognit.*, vol. 120, p. 108192, 2021.

[98] Y. Sun, H. Zhu, and H. Xiong, "Towards faithful neural network intrinsic interpretation with shapley additive self-attribution," *arXiv preprint arXiv:2309.15559*, 2023.

[99] R. Wang, X. Wang, and D. Inouye, "Shapley explanation networks," in *ICLR*, 2021.

[100] Y. Sun, F. Zhuang, H. Zhu, Q. Zhang, Q. He, and H. Xiong, "Market-oriented job skill valuation with cooperative composition neural network," *Nat. Commun.*, vol. 12, no. 1, pp. 1–12, 2021.

[101] T. Quinn, D. Nguyen, S. Rana, S. Gupta, and S. Venkatesh, "Deepcoda: personalized interpretability for compositional health data," in *ICML*, 2020, pp. 7877–7886.

[102] Y. Wang and X. Wang, "Self-interpretable model with transformation equivariant interpretation," *NeurIPS*, vol. 34, pp. 2359–2372, 2021.

[103] A. Martins and R. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," in *ICML*, 2016, pp. 1614–1623.

[104] G. M. Correia, V. Niculae, and A. F. Martins, "Adaptively sparse transformers," in *EMNLP*, 2019, pp. 2174–2184.

[105] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[106] A. Vaswani, "Attention is all you need," *NeurIPS*, 2017.

[107] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *EMNLP*, 2019, pp. 11–20.

[108] S. Serrano and N. A. Smith, "Is attention interpretable?" in *ACL*, 2019.

[109] C. Meister, S. Lazov, I. Augenstein, and R. Cotterell, "Is sparse attention more interpretable?" in *ACL*, 2021, pp. 122–129.

[110] A. Bibal, R. Cardon, D. Alfter, R. Wilkens, X. Wang, T. François, and P. Watrin, "Is attention explanation? an introduction to the debate," in *ACL*, 2022, pp. 3889–3900.

[111] B. Bai, J. Liang, G. Zhang, H. Li, K. Bai, and F. Wang, "Why attentions may not be interpretable?" in *SIGKDD*, 2021, pp. 25–34.

[112] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer, "On identifiability in transformers," in *ICLR*, 2020.

[113] K. Sun and A. Marasovi'c, "Effective attention sheds light on interpretability," in *ACL*, 2021, pp. 4126–4135.

[114] G. Kobayashi, T. Kuribayashi, S. Yokoi, and K. Inui, "Attention is not only a weight: Analyzing transformers with vector norms," in *EMNLP*, 2020.

[115] L. Hu, Y. Liu, N. Liu, M. Huai, L. Sun, and D. Wang, "Seat: stable and explainable attention," in *AAAI*, 2023, pp. 12 907–12 915.

[116] A. K. Mohankumar, P. Nema, S. Narasimhan, M. M. Khapra, B. V. Srinivasan, and B. Ravindran, "Towards transparent and explainable attention models," in *ACL*, 2020, pp. 4206–4216.

[117] T. H. Nguyen and K. Rudra, "Learning faithful attention for interpretable classification of crisis-related microblogs under constrained human budget," in *WWW*, 2023, pp. 3959–3967.

[118] D. Pruthi, M. Gupta, B. Dhingra, G. Neubig, and Z. C. Lipton, "Learning to deceive with attention-based explanations," in *ACL*, 2020.

[119] J. Stacey, Y. Belinkov, and M. Rei, "Supervising model attention with human explanations for robust natural language inference," in *AAAI*, 2022, pp. 11 349–11 357.

[120] M. Rigotti, C. Miksovic, I. Giurgiu, T. Gschwind, and P. Scotton, "Attention-based interpretability with concept transformers," in *ICLR*, 2021.

[121] Y. Li, S. Lin, B. Zhang, J. Liu, D. Doermann, Y. Wu, F. Huang, and R. Ji, "Exploiting kernel sparsity and entropy for interpretable cnn compression," in *CVPR*, 2019, pp. 2800–2809.

[122] T. Lei, R. Barzilay, and T. Jaakkola, "Rationalizing neural predictions," in *EMNLP*, 2016, pp. 107–117.

[123] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," in *ICLR*, 2017.

[124] S. M. Xie and S. Ermon, "Reparameterizable subset sampling via continuous relaxations," in *IJCAI*, 2019, pp. 3919–3925.

[125] R. A. Amjad and B. C. Geiger, "Learning representations for neural network-based classification using the information bottleneck principle," *TPAMI*, vol. 42, no. 9, pp. 2225–2239, 2019.

[126] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *ICLR*, 2022.

[127] W. Guo, X. Wu, U. Khan, and X. Xing, "Edge: Explaining deep reinforcement learning policies," *NeurIPS*, vol. 34, pp. 12 222–12 236, 2021.

[128] X. Sun, Z. Wang, R. Ding, S. Han, and D. Zhang, "Puregam: Learning an inherently pure additive model," in *KDD*, 2022, pp. 1728–1738.

[129] R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton, "Neural additive models: Interpretable machine learning with neural nets," *NeurIPS*, vol. 34, pp. 4699–4711, 2021.

[130] T. J. Hastie, "Generalized additive models," in *Statistical models in S*. Routledge, 2017, pp. 249–307.

[131] V. Duong, Q. Wu, Z. Zhou, H. Zhao, C. Luo, E. Zavesky, H. Yao, and H. Shao, "Cat: Interpretable concept-based taylor additive models," in *KDD*, 2024, pp. 723–734.

[132] L. S. Shapley, "A value for n-person games," *Contrib. Theory Games*, vol. 2, 1953.

[133] J. Bento, P. Saleiro, A. F. Cruz, M. A. Figueiredo, and P. Bizarro, "Timeshap: Explaining recurrent models through sequence perturbations," in *SIGKDD*, 2021, pp. 2565–2573.

[134] Y. Sun, H. Zhu, C. Qin, F. Zhuang, Q. He, and H. Xiong, "Discerning decision-making process of deep neural networks with hierarchical voting transformation," *NeurIPS*, vol. 34, pp. 17 221–17 234, 2021.

[135] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "Kan: Kolmogorov-arnold networks," *arXiv preprint arXiv:2404.19756*, 2024.

[136] S. Kim, P. Y. Lu, S. Mukherjee, M. Gilbert, L. Jing, V. Čeperić, and M. Soljačić, "Integration of neural network-based symbolic regression in deep learning for scientific discovery," *TNNLS*, pp. 4166–4177, 2020.

[137] N. Ranasinghe, D. Senanayake, S. Seneviratne, M. Premaratne, and S. Halgamuge, "Ginn-lp: A growing interpretable neural network for discovering multivariate laurent polynomial equations," in *AAAI*, 2024, pp. 14 776–14 784.

[138] P.-A. Kamienny, S. d'Ascoli, G. Lample, and F. Charton, "End-to-end symbolic regression with transformers," *NeurIPS*, pp. 10 269–10 281, 2022.

[139] T. Bendinelli, L. Biggio, and P.-A. Kamienny, "Controllable neural symbolic regression," in *ICML*, 2023, pp. 2063–2077.

[140] L. Biggio, T. Bendinelli, A. Neitz, A. Lucchi, and G. Parascandolo, "Neural symbolic regression that scales," in *ICML*, 2021, pp. 936–945.

[141] M. Valipour, B. You, M. Panju, and A. Ghodsi, "Symbolicgpt: A generative transformer model for symbolic regression," *arXiv preprint arXiv:2106.14131*, 2021.

[142] M. Landajuela, C. S. Lee, J. Yang, R. Glatt, C. P. Santiago, I. Aravena, T. Mundhenk, G. Mulcahy, and B. K. Petersen, "A unified framework for deep symbolic regression," *NeurIPS*, pp. 33 985–33 998, 2022.

[143] S. Holt, Z. Qian, and M. van der Schaar, "Deep generative symbolic regression," in *ICLR*, 2023.

[144] S. Sahoo, C. Lampert, and G. Martius, "Learning equations for extrapolation and control," in *ICML*, 2018, pp. 4442–4450.

[145] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang, "Concept bottleneck models," in *ICML*, 2020, pp. 5338–5348.

[146] Z. Chen, Y. Bei, and C. Rudin, "Concept whitening for interpretable image recognition," *Nat. Mach. Intell.*, vol. 2, no. 12, pp. 772–782, 2020.

[147] P. J. Blazek and M. M. Lin, "Explainable neural networks that simulate reasoning," *Nat. Comput. Sci.*, pp. 607–618, 2021.

[148] M. Espinosa Zarlenga, P. Barbiero, G. Ciravegna, G. Marra, F. Giannini, M. Diligenti, Z. Shams, F. Precioso, S. Melacci, A. Weller *et al.*, "Concept embedding models: Beyond the accuracy-explainability trade-off," *NeurIPS*, pp. 21 400–21 413, 2022.

[149] M. Yuksekgonul, M. Wang, and J. Zou, "Post-hoc concept bottleneck models," in *ICLR*, 2022.

[150] E. Marconato, A. Passerini, and S. Teso, "Glancenets: Interpretable, leak-proof concept-based models," *NeurIPS*, pp. 21 212–21 227, 2022.

[151] E. Kim, D. Jung, S. Park, S. Kim, and S. Yoon, "Probabilistic concept bottleneck models," in *ICML*, 2023, pp. 16 521–16 540.

[152] M. Havasi, S. Parbhoo, and F. Doshi-Velez, "Addressing leakage in concept bottleneck models," *NeurIPS*, pp. 23 386–23 397, 2022.

[153] D. Steinmann, W. Stammer, F. Friedrich, and K. Kersting, "Learning to intervene on concept bottlenecks," in *ICML*, 2023.

[154] K. Chauhan, R. Tiwari, J. Freyberg, P. Shenoy, and K. Dvijotham, "Interactive concept bottleneck models," in *AAAI*, 2023, pp. 5948–5955.

[155] T. Oikarinen, S. Das, L. Nguyen, and L. Weng, "Label-free concept bottleneck models," in *ICLR*, 2023.

[156] Y. Yang, A. Panagopoulou, S. Zhou, D. Jin, C. Callison-Burch, and M. Yatskar, "Language in a bottle: Language model guided concept bottlenecks for interpretable image classification," in *CVPR*, 2023, pp. 19 187–19 197.

[157] K. Panousis, D. Ienco, and D. Marcos, "Coarse-to-fine concept bottleneck models," in *NeurIPS*, 2024.

[158] C. Shang, S. Zhou, H. Zhang, X. Ni, Y. Yang, and Y. Wang, "Incremental residual concept bottleneck models," in *CVPR*, 2024, pp. 11 030–11 040.

[159] M. Espinosa Zarlenga, K. Collins, K. Dvijotham, A. Weller, Z. Shams, and M. Jamnik, "Learning to receive help: Intervention-aware concept embedding models," *NeurIPS*, 2024.

[160] X. Xu, Y. Qin, L. Mi, H. Wang, and X. Li, "Energy-based concept bottleneck models: unifying prediction, concept intervention, and conditional interpretations," *arXiv preprint arXiv:2401.14142*, 2024.

[161] M. Vandenhirtz, S. Laguna, R. Marcinkevičs, and J. E. Vogt, "Stochastic concept bottleneck models," *NeurIPS*, 2024.

[162] P. Barbiero, F. Giannini, G. Ciravegna, M. Diligenti, and G. Marra, "Relational concept bottleneck models," in *NeurIPS*, 2024.

[163] Z. Tan, T. Chen, Z. Zhang, and H. Liu, "Sparsity-guided holistic explanation for llms with interpretable inference-time intervention," in *AAAI*, 2024, pp. 21 619–21 627.

[164] Y. Bie, L. Luo, and H. Chen, "Mica: Towards explainable skin lesion diagnosis via multi-level image-concept alignment," in *AAAI*, 2024, pp. 837–845.

[165] M. R. Luyten and M. van der Schaar, "A theoretical design of concept sets: improving the predictability of concept bottleneck models," in *NeurIPS*, 2024.

[166] N. J. Raman, M. E. Zarlenga, and M. Jamnik, "Understanding inter-concept relationships in concept-based models," in *ICML*, 2024.

[167] S. Yan, Z. Yu, X. Zhang, D. Mahapatra, S. S. Chandra, M. Janda, P. Soyer, and Z. Ge, "Towards trustable skin cancer diagnosis via rewriting model's decision," in *CVPR*, 2023, pp. 11 568–11 577.

[168] D. Srivastava, G. Yan, and T.-W. Weng, "Vlg-cbm: Training concept bottleneck models with vision-language guidance," in *NeurIPS*, 2024.

[169] T. Van Gog and N. Rummel, "Example-based learning: Integrating cognitive and social-cognitive research perspectives," *Educ. Psychol. Rev.*, vol. 22, pp. 155–174, 2010.

[170] Y. Sun, Y. Ji, H. Zhu, F. Zhuang, Q. He, and H. Xiong, "Market-aware long-term job skill recommendation with explainable deep reinforcement learning," *TOIS*, 2024.

[171] A. Haselhoff, K. Trelenberg, F. Küppers, and J. Schneider, "The gaussian discriminant variational autoencoder (gdvae): A self-explainable model with counterfactual explanations," in *ECCV*, 2025, pp. 305–322.

[172] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *AAAI*, 2018.

[173] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su, "This looks like that: deep learning for interpretable image recognition," *NeurIPS*, vol. 32, 2019.

[174] E. Kim, S. Kim, M. Seo, and S. Yoon, "Xprotonet: diagnosis in chest radiography with global and local explanations," in *CVPR*, 2021, pp. 15 719–15 728.

[175] M. Keswani, S. Ramakrishnan, N. Reddy, and V. N. Balasubramanian, "Proto2proto: Can you recognize the car, the way i do?" in *CVPR*, 2022, pp. 10 233–10 243.

[176] M. Nauta, J. Schlötterer, M. van Keulen, and C. Seifert, "Pip-net: Patch-based intuitive prototypes for interpretable image classification," in *CVPR*, 2023, pp. 2744–2753.

[177] M. Sacha, D. Rymarczyk, Ł. Struski, J. Tabor, and B. Zieliński, "Protoseg: Interpretable semantic segmentation with prototypical parts," in *WACV*, 2023, pp. 1481–1492.

[178] D. Hong, T. Wang, and S. Baek, "Protorynet-interpretable text classification via prototype trajectories," *JMLR*, vol. 24, no. 264, pp. 1–39, 2023.

[179] J. Wang, H. Liu, X. Wang, and L. Jing, "Interpretable image recognition by constructing transparent embedding space," in *ICCV*, 2021, pp. 895–904.

[180] R. Kjærsgaard, A. Boubekki, and L. Clemmensen, "Pantypes: Diverse representatives for self-explainable models," in *AAAI*, 2024, pp. 13 230–13 237.

[181] S. Gautam, A. Boubekki, S. Hansen, S. Salahuddin, R. Jenssen, M. Höhne, and M. Kampffmeyer, "Protovae: A trustworthy self-explainable prototypical variational model," *NeurIPS*, vol. 35, pp. 17 940–17 952, 2022.

[182] C. Ma, B. Zhao, C. Chen, and C. Rudin, "This looks like those: Illuminating prototypical concepts using multiple visualizations," *NeurIPS*, vol. 36, 2024.

[183] M. Nauta, R. Van Bree, and C. Seifert, "Neural prototype trees for interpretable fine-grained image recognition," in *CVPR*, 2021, pp. 14 933–14 943.

[184] D. Rymarczyk, Ł. Struski, M. Górszczak, K. Lewandowska, J. Tabor, and B. Zieliński, "Interpretable image classification with differentiable prototypes assignment," in *ECCV*, 2022, pp. 351–368.

[185] C. Wang, Y. Liu, Y. Chen, F. Liu, Y. Tian, D. McCarthy, H. Frazer, and G. Carneiro, "Learning support and trivial prototypes for interpretable image classification," in *ICCV*, 2023, pp. 2062–2072.

[186] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C. Lee, "Protgnn: Towards self-explaining graph neural networks," in *AAAI*, 2022, pp. 9127–9135.

[187] Y. Ming, P. Xu, H. Qu, and L. Ren, "Interpretable and steerable sequence learning via prototypes," in *SIGKDD*, 2019, pp. 903–913.

[188] A. Bontempelli, S. Teso, K. Tentori, F. Giunchiglia, A. Passerini *et al.*, "Concept-level debugging of part-prototype networks," in *ICLR*, 2023.

[189] E. M. Kenny, M. Tucker, and J. Shah, "Towards interpretable deep reinforcement learning with human-friendly prototypes," in *ICLR*, 2023.

[190] D. Rymarczyk, Ł. Struski, J. Tabor, and B. Zieliński, "Protopshare: Prototypical parts sharing for similarity discovery in interpretable image classification," in *SIGKDD*, 2021, pp. 1420–1430.

[191] Q. Huang, M. Xue, W. Huang, H. Zhang, J. Song, Y. Jing, and M. Song, "Evaluation and improvement of interpretability for self-explainable part-prototype networks," in *ICCV*, 2023, pp. 2011–2020.

[192] Z. Carmichael, S. Lohit, A. Cherian, M. J. Jones, and W. J. Scheirer, "Pixel-grounded prototypical part networks," in *WACV*, 2024, pp. 4768–4779.

[193] M. Sacha, B. Jura, D. Rymarczyk, Ł. Struski, J. Tabor, and B. Zieliński, "Interpretability benchmark for evaluating spatial misalignment of prototypical parts explanations," in *AAAI*, 2024, pp. 21 563–21 573.

[194] T. Feng, R. Quan, X. Wang, W. Wang, and Y. Yang, "Interpretable3d: An ad-hoc interpretable classifier for 3d point clouds," in *AAAI*, 2024, pp. 1761–1769.

[195] M. Xue, Q. Huang, H. Zhang, J. Hu, J. Song, M. Song, and C. Jin, "Protopformer: Concentrating on prototypical parts in vision transformers for interpretable image recognition," in *IJCAI*, 2024, pp. 1516–1524.

[196] D. Rajagopal, V. Balachandran, E. H. Hovy, and Y. Tsvetkov, "Self-explain: A self-explaining architecture for neural text classifiers," in *EMNLP*, 2021, pp. 836–850.

[197] Y. Wang, S. Liu, T. Zheng, K. Chen, and M. Song, "Unveiling global interactive patterns across graphs: Towards interpretable graph neural networks," in *KDD*, 2024, pp. 3277–3288.

[198] M. Tucker and J. A. Shah, "Prototype based classification from hierarchy to fairness," in *ICML*, 2022, pp. 21 884–21 900.

[199] Y. Jiang, W. Yu, D. Song, L. Wang, W. Cheng, and H. Chen, "Fedskill: Privacy preserved interpretable skill learning via imitation," in *SIGKDD*, 2023, pp. 1010–1019.

[200] A. J. Barnett, F. R. Schwartz, C. Tao, C. Chen, Y. Ren, J. Y. Lo, and C. Rudin, "A case-based interpretable deep learning model for classification of mass lesions in digital mammography," *Nat. Mach. Intell.*, pp. 1061–1070, 2021.

[201] D. Rymarczyk, J. van de Weijer, B. Zieliński, and B. Twardowski, "Icicle: Interpretable class incremental continual learning," in *ICCV*, 2023, pp. 1887–1898.

[202] R. Ragodos, T. Wang, Q. Lin, and X. Zhou, "Protox: Explaining a reinforcement learning agent via prototyping," *NeurIPS*, vol. 35, pp. 27 239–27 252, 2022.

[203] E. van Krieken, E. Acar, and F. van Harmelen, "Analyzing differentiable fuzzy logic operators," *Artif. Intell.*, 2022.

[204] F. Petersen, C. Borgelt, H. Kuehne, and O. Deussen, "Deep differentiable logic gate networks," *NeurIPS*, pp. 2006–2018, 2022.

[205] S. Shi, Y. Xie, Z. Wang, B. Ding, Y. Li, and M. Zhang, "Explainable neural rule learning," in *WWW*, 2022, pp. 3031–3041.

[206] L. Yu, M. Li, Y.-L. Zhang, L. Li, and J. Zhou, "Finrule: Feature interactive neural rule learning," in *CIKM*, 2023, pp. 3020–3029.

[207] I. Donadello, L. Serafini, and d. G. Artur, "Logic tensor networks for semantic image interpretation," in *IJCAI*, 2017, pp. 1596–1602.

[208] Z. Wang, W. Zhang, L. Ning, and J. Wang, "Transparent classification with multilayer logical perceptrons and random binarization," in *AAAI*, 2020, pp. 6331–6339.

[209] L. Qiao, W. Wang, and B. Lin, "Learning accurate and interpretable decision rule sets from neural networks," in *AAAI*, 2021, pp. 4303–4311.

[210] Z. Wang, W. Zhang, N. Liu, and J. Wang, "Scalable rule-based representation learning for interpretable classification," *NeurIPS*, pp. 30 479–30 491, 2021.

[211] ——, "Learning interpretable rules for scalable data representation and classification," *TPAMI*, 2023.

[212] Y. Yang, W. Ren, and S. Li, "Hyperlogic: Enhancing diversity and accuracy in rule learning with hypernets," in *NeurIPS*, 2024.

[213] F. Petersen, H. Kuehne, C. Borgelt, J. Welzel, and S. Ermon, "Convolutional differentiable logic gate networks," in *NeurIPS*, 2024.

[214] W. Zhang, Y. Liu, Z. Wang, and J. Wang, "Learning to binarize continuous features for neuro-rule networks." in *IJCAI*, 2023, pp. 4584–4592.

[215] G. Marra, M. Diligenti, F. Giannini, M. Gori, and M. Maggini, "Relational neural machines," in *ECAI*, 2020, pp. 1340–1347.

[216] G. Ciravegna, F. Giannini, M. Gori, M. Maggini, S. Melacci *et al.*, "Human-driven fol explanations of deep learning," in *IJCAI*, 2020, pp. 2234–2240.

[217] G. Ciravegna, P. Barbiero, F. Giannini, M. Gori, P. Lió, M. Maggini, and S. Melacci, "Logic explained networks," *Artif. Intell.*, 2023.

[218] R. Jain, G. Ciravegna, P. Barbiero, F. Giannini, D. Buffelli, P. Lio *et al.*, "Extending logic explained networks to text classification," in *EMNLP*, 2022, pp. 8838–8857.

[219] G. Ciravegna, F. Giannini, S. Melacci, M. Maggini, and M. Gori, "A constraint-based approach to learning and explanation," in *AAAI*, 2020, pp. 3658–3665.

[220] P. Barbiero, G. Ciravegna, F. Giannini, P. Lió, M. Gori, and S. Melacci, "Entropy-based logic explanations of neural networks," in *AAAI*, 2022, pp. 6046–6054.

[221] P. Barbiero, G. Ciravegna, F. Giannini, M. E. Zarlenga, L. C. Magister, A. Tonda, P. Lió, F. Precioso, M. Jamnik, and G. Marra, "Interpretable neural-symbolic concept reasoning," in *ICML*, 2023, pp. 1801–1825.

[222] A. Wan, L. Dunlap, D. Ho, J. Yin, S. Lee, S. Petryk, S. A. Bargal, and J. E. Gonzalez, "Nbdt: Neural-backed decision tree," in *ICLR*, 2020.

[223] Y. Okajima and K. Sadamasa, "Deep neural networks constrained by decision rules," in *AAAI*, 2019, pp. 2496–2505.

[224] Z. Wu, Z. X. Zhang, A. Naik, Z. Mei, M. Firdaus, and L. Mou, "Weakly supervised explainable phrasal reasoning with neural fuzzy logic," in *ICLR*, 2021.

[225] Y. Yang, I. G. Morillo, and T. M. Hospedales, "Deep neural decision trees," *arXiv preprint arXiv:1806.06988*, 2018.

[226] S. Kim, J. Nam, and B. C. Ko, "Vit-net: Interpretable vision transformers with neural tree decoder," in *ICML*, 2022, pp. 11 162–11 172.

[227] Z. Zhang, P. Chen, M. McGough, F. Xing, C. Wang, M. Bui, Y. Xie, M. Sapkota, L. Cui, J. Dhillon *et al.*, "Pathologist-level interpretable whole-slide cancer diagnosis with deep learning," *Nat. Mach. Intell.*, vol. 1, no. 5, pp. 236–245, 2019.

[228] Y. Xu, X. Yang, L. Gong, H.-C. Lin, T.-Y. Wu, Y. Li, and N. Vasconcelos, "Explainable object-induced action decision for autonomous vehicles," in *CVPR*, 2020, pp. 9523–9532.

[229] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum, "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding," *NeurIPS*, vol. 31, 2018.

[230] Q. Zhang, X. Wang, Y. N. Wu, H. Zhou, and S.-C. Zhu, "Interpretable cnns for object classification," *TPAMI*, pp. 3416–3431, 2020.

[231] X. Shen, Y. Luo, Z. Liu, and L. Song, "Interpretable compositional convolutional neural networks," in *IJCAI*, 2021, pp. 2971–2978.

[232] H. Liang, Z. Ouyang, Y. Zeng, H. Su, Z. He, S.-T. Xia, J. Zhu, and B. Zhang, "Training interpretable convolutional neural networks by differentiating class-specific filters," in *ECCV*, 2020, pp. 622–638.

[233] W. Guo, J. Yang, H. Yin, Q. Chen, and W. Ye, "Picnn: A pathway towards interpretable convolutional neural networks," in *AAAI*, 2024, pp. 2003–2012.

[234] K. Greff, S. Van Steenkiste, and J. Schmidhuber, "Interpretable neural predictions with differentiable binary variables," *Artif. Intell.*, vol. 289, p. 103396, 2019.

[235] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *NAACL*, 2016, pp. 1480–1489.

[236] R. Ghaeini, X. Z. Fern, and P. Tadepalli, "Interpreting recurrent and attention-based neural models: a case study on natural language inference," *ACL*, 2018.

[237] A. Subramanian, D. Pruthi, H. Jhamtani, T. Berg-Kirkpatrick, and E. Hovy, "Spine: Sparse interpretable neural embeddings," in *AAAI*, 2018.

[238] M. Du, N. Liu, F. Yang, and X. Hu, "Learning credible deep neural networks with rationale regularization," in *ICDM*, 2019, pp. 150–159.

[239] S. O. Arik and T. Pfister, "Protoattend: Attention-based prototypical learning," *JMLR*, vol. 21, no. 210, pp. 1–35, 2020.

[240] A. Chan, J. Qin, Y. Chen, V. Srivastava, and P. Agrawal, "When can models learn from explanations? a formal framework for understanding the roles of explanation data," in *NeurIPS*, 2022.

[241] N. Zhang, Y. Xiao, and B. Yu, "Summarize-then-answer: Generating concise explanations for multi-hop reading comprehension," in *EMNLP*, 2023.

[242] A. Jacovi and Y. Goldberg, "Aligning faithful interpretations with their social attribution," *TACL*, vol. 9, pp. 294–310, 2021.

[243] S. Kumar and P. Talukdar, "NILE: Natural language inference with faithful natural language explanations," in *ACL*, 2022, pp. 8664–8684.

[244] N. F. Rajani, B. McCann, C. Xiong, and R. Socher, "Explain Yourself! Leveraging language models for commonsense reasoning," in *ACL*, 2019, pp. 4932–4942.

[245] Y. Sun, Y. Ji, Y. Zhang, and Z. Wang, "LIREx: Augmenting language inference with relevant explanation," in *AAAI*, 2023, pp. 13 944–13 952.

[246] O. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom, "Towards interpretable natural language understanding with explanations as latent variables," in *NeurIPS*, 2018, pp. 4864–4873.

[247] S. Wiegreffe, A. Marasović, and N. A. Smith, "Measuring association between labels and free-text rationales," in *EMNLP*, 2021, pp. 10 266–10 284.

[248] A. Marasović, I. Beltagy, D. Downey, and M. E. Peters, "Few-shot self-rationalization with natural language prompts," in *NAACL*, 2022, pp. 410–424.

[249] S. Narang, C. Raffel, K. Lee, A. Roberts, N. Fiedel, and K. Malkan, "Wt5?! training text-to-text models to explain their predictions," *arXiv preprint arXiv:2004.14546*, 2020.

[250] H. Liu, Q. Yin, and W. Y. Wang, "Towards explainable nlp: A generative explanation framework for text classification," in *ACL*, 2019, pp. 5570–5581.

[251] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom, "e-snli: Natural language inference with natural language explanations," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[252] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *NeurIPS*, vol. 35, pp. 24 824–24 837, 2022.

[253] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *NeurIPS*, vol. 36, 2024.

[254] A. Madsen, S. Chandar, and S. Reddy, "Are self-explanations from large language models faithful?" in *ACL*, 2024, pp. 295–337.

[255] W. J. Yeo, R. Satapathy, R. S. M. Goh, and E. Cambria, "How interpretable are reasoning explanations from prompting large language models?" in *ACL*, 2024, pp. 1800–1815.

[256] J. Yu, J. Cao, and R. He, "Improving subgraph recognition with variational graph information bottleneck," in *CVPR*, 2022, pp. 19 396–19 405.

[257] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, and R. He, "Graph information bottleneck for subgraph recognition," in *ICLR*, 2021.

[258] Y. Chen, Y. Bian, B. Han, and J. Cheng, "How interpretable are interpretable graph neural networks?" in *ICML*, 2024.

[259] E. Dai and S. Wang, "Towards self-explainable graph neural network," in *CIKM*, 2021, pp. 302–311.

[260] A. Feng, C. You, S. Wang, and L. Tassiulas, "Kergnns: Interpretable graph neural networks with graph kernels," in *AAAI*, 2022, pp. 6614–6622.

[261] F. Giannini, S. Fioravanti, O. Keskin, A. Lupidi, L. C. Magister, P. Lió, and P. Barbiero, "Interpretable graph networks formulate universal algebra conjectures," *NeurIPS*, 2024.

[262] Y. Wu, X. Wang, A. Zhang, X. He, and T.-S. Chua, "Discovering invariant rationales for graph neural networks," in *ICLR*, 2022.

[263] Y. Nian, Y. Chang, W. Jin, and L. Lin, "Globally interpretable graph learning via distribution matching," in *WWW*, 2024, pp. 992–1002.

[264] Q. Lin, J. Liu, R. Mao, F. Xu, and E. Cambria, "Techs: Temporal logical graph networks for explainable extrapolation reasoning," in *ACL*, 2023, pp. 1281–1293.

[265] J. Shi, S. Guo, H. Dai, T. Luo, Z. Zheng, J. Wang, and S.-T. Xia, "Sig: Efficient self-interpretable graph neural network for continuous-time dynamic graphs," in *Advances in Neural Information Processing Systems*, 2023.

[266] Y. Pan, J. Liu, T. Zhao, L. Zhang, Y. Lin, and J. S. Dong, "A symbolic rule integration framework with logic transformer for inductive relation prediction," in *WWW*, 2024, pp. 2181–2192.

[267] Y. Yang, Z. Guan, J. Li, W. Zhao, J. Cui, and Q. Wang, "Interpretable and efficient heterogeneous graph convolutional network," *TKDE*, pp. 1637–1650, 2021.

[268] J. Yin, C. Li, H. Yan, J. Lian, and S. Wang, "Train once and explain everywhere: Pre-training interpretable graph neural networks," *NeurIPS*, pp. 35 277–35 299, 2023.

[269] Z. Huang, K. Li, S. Wang, Z. Jia, W. Zhu, and S. Mehrotra, "Ses: Bridging the gap between explainability and prediction of graph neural networks," in *ICDE*, 2024, pp. 2945–2958.

[270] W. Jin, R. Barzilay, and T. Jaakkola, "Multi-objective molecule generation using interpretable substructures," in *ICML*, 2020, pp. 4849–4859.

[271] T. Lanciano, F. Bonchi, and A. Gionis, "Explainable classification of brain networks via contrast subgraphs," in *KDD*, 2020, pp. 3308–3318.

[272] W. Du, S. Zhang, D. Wu, J. Xia, Z. Zhao, J. Fang, and Y. Wang, "Mmgnn: A molecular merged graph neural network for explainable solvation free energy prediction," in *IJCAI*, 2024, pp. 5808–5816.

[273] R. M. Annasamy and K. Sycara, "Towards better interpretability in deep q-networks," in *AAAI*, 2019, pp. 4561–4569.

[274] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. Jimenez Rezende, "Towards interpretable reinforcement learning using attention augmented agents," *NeurIPS*, vol. 32, 2019.

[275] J. Wang, Y. Zhang, K. Tang, J. Wu, and Z. Xiong, "Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks," in *SIGKDD*, 2019, pp. 1900–1908.

[276] W. Shi, G. Huang, S. Song, Z. Wang, T. Lin, and C. Wu, "Self-supervised discovering of interpretable features for reinforcement learning," *TPAMI*, vol. 44, no. 5, pp. 2712–2724, 2020.

[277] G. Liu, X. Sun, O. Schulte, and P. Poupart, "Learning tree interpretation from object representation for deep reinforcement learning," *NeurIPS*, pp. 19 622–19 636, 2021.

[278] Z. Yu, J. Ruan, and D. Xing, "Explainable reinforcement learning via a causal world model," in *IJCAI*, 2023, pp. 4540–4548.

[279] Y. Zhang, Y. Du, B. Huang, Z. Wang, J. Wang, M. Fang, and M. Pechenizkiy, "Interpretable reward redistribution in reinforcement learning: a causal approach," *NeurIPS*, 2024.

[280] H. Yau, C. Russell, and S. Hadfield, "What did you think would happen? explaining agent behaviour through intended outcomes," *NeurIPS*, pp. 18 375–18 386, 2020.

[281] A. Verma, H. M. Le, Y. Yue, and S. Chaudhuri, "Interpretable and explainable logical policies via neurally guided symbolic abstraction," in *ICLR*, 2023.

[282] J. Guo, R. Zhang, S. Peng, Q. Yi, X. Hu, R. Chen, Z. Du, X. Zhang, L. Li, Q. Guo, and Y. Chen, "Efficient symbolic policy learning with differentiable symbolic expression," in *NeurIPS*, 2023.
[283] X. Li, H. Lei, L. Zhang, and M. Wang, "Differentiable logic policy for interpretable deep reinforcement learning: A study from an optimization perspective," *TPAMI*, vol. 45, no. 10, pp. 11 654–11 667, 2023.
[284] C.-K. Yeh, B. Kim, S. Arik, C.-L. Li, T. Pfister, and P. Ravikumar, "On completeness-aware concept-based explanations in deep neural networks," *NeurIPS*, pp. 20 554–20 565, 2020.
[285] W. Qian, C. Zhao, Y. Li, F. Ma, C. Zhang, and M. Huai, "Towards modeling uncertainties of self-explaining neural networks via conformal prediction," in *AAAI*, 2024, pp. 14 651–14 659.
[286] M. E. Zarlenga, P. Barbiero, Z. Shams, D. Kazhdan, U. Bhatt, A. Weller, and M. Jamnik, "Towards robust metrics for concept representation evaluation," in *AAAI*, vol. 37, no. 10, 2023, pp. 11 791–11 799.
[287] Y. Rong, T. Leemann, T.-T. Nguyen, L. Fiedler, P. Qian, V. Unhelkar, T. Seidel, G. Kasneci, and E. Kasneci, "Towards human-centered explainable ai: A survey of user studies for model explanations," *TPAMI*, 2023.
[288] S. Ghosh, K. Yu, F. Arabshahi, and K. Batmanghelich, "Dividing and conquering a blackbox to a mixture of interpretable models: route, interpret, repeat," in *ICML*, 2023.
[289] Q. Zhang, Y. Yang, H. Ma, and Y. N. Wu, "Interpreting cnns via decision trees," in *CVPR*, 2019, pp. 6261–6270.
[290] J. Song, H. Zhang, X. Wang, M. Xue, Y. Chen, L. Sun, D. Tao, and M. Song, "Tree-like decision distillation," in *CVPR*, 2021, pp. 13 488–13 497.
[291] Y. Dang, K. Huang, J. Huo, Y. Yan, S. Huang, D. Liu, M. Gao, J. Zhang, C. Qian, K. Wang *et al.*, "Explainable and interpretable multimodal large language models: A comprehensive survey," *arXiv preprint arXiv:2412.02104*, 2024.