LogicCBMs: Logic-Enhanced Concept-Based Learning

Deepika SN Vemuri[1] Gautham Bellamkonda[1,3] Aditya Pola[1] Vineeth N Balasubramanian[1,2]
[1]Indian Institute of Technology, Hyderabad, [2]Microsoft Research, [3]KLA
{ai22resch11001, ai24mtech02001, vineethnb}@iith.ac.in, gauthambellamkonda@gmail.com

Abstract

Concept Bottleneck Models (CBMs) provide a basis for semantic abstractions within a neural network architecture. Such models have primarily been seen through the lens of interpretability so far, wherein they offer transparency by inferring predictions as a linear combination of semantic concepts. However, a linear combination is inherently limiting. So we propose the enhancement of concept-based learning models through propositional logic. We introduce a logic module that is carefully designed to connect the learned concepts from CBMs through differentiable logic operations, such that our proposed LogicCBM can go beyond simple weighted combinations of concepts to leverage various logical operations to yield the final predictions, while maintaining end-to-end learnability. Composing concepts using a set of logic operators enables the model to capture inter-concept relations, while simultaneously improving the expressivity of the model in terms of logic operations. Our empirical studies on well-known benchmarks and synthetic datasets demonstrate that these models have better accuracy, perform effective interventions and are highly interpretable [1].

1. Introduction

In recent times, building inherently interpretable models has gained prominence due to the drawbacks of post hoc explainability methods [19, 32, 36]. Concept-based models are a particularly promising direction [6, 14, 25, 30, 48], where classes are considered to be composed of concepts that are interpretable units of human-understandable abstraction. For example, the model could learn to look for concepts like {*huge, gray, mammal*} to classify an input as an *elephant*. In these models, the concept-to-class mapping is often deliberately kept simple (usually just a linear layer) for interpretability, so that the importance of each concept can be directly inferred by examining the weights. However, such an approach may be restrictive and prevent the model from learning and leveraging higher-order relations between concepts.

To illustrate this further, consider a model learning to recognize an *arctic fox* class. Arctic foxes have either *white fur* or *brown fur* depending on the environmental conditions (equivalent to an exclusive OR operation). Evidently, a concept-based model will need to go beyond a linear layer

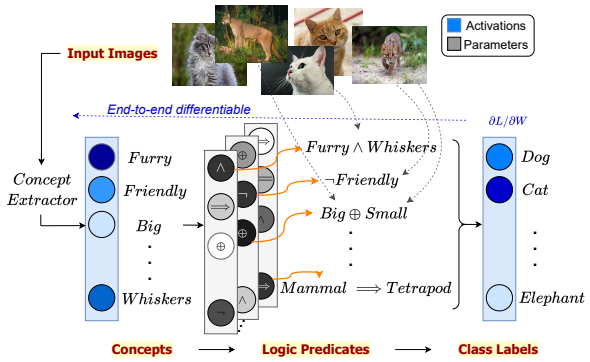

Figure 1. **LogicCBMs: Overview of Our Approach.** We enhance concept-based learning models by including differentiable logic gates in the network. The model now forms logical compositions of concepts while predicting the class label for a given input image. (Dark shades indicate higher strength in the figure; e.g. *Furry* is the strongest concept and *Cat* is the predicted class.)

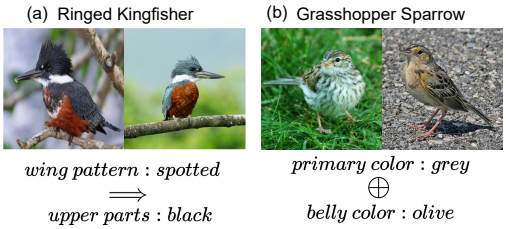

Figure 2. **Logical predicates capture intra-class variability.** Examples of predicates (bottom) captured by our method for classes (top) in the CUB dataset (test set). (a) A *Ringed Kingfisher* has black upper parts if it has a spotted wing pattern. This is captured by an IMPLIES operation. (b) A *Grasshopper Sparrow* has its primary color as grey or belly color as olive (not both, not neither), captured by an XOR operation in our method.

[1]https://github.com/deepikavemuri/LogicCBMs

to capture such nuances. We hence ask the question: *while concept-based models are becoming increasingly popular, how do we go beyond the linear layer in the concept-class connections to capture richer relationships, while retaining interpretability?*

Logic operations present a natural choice to model concept-class relations in a structured manner (e.g. *white fur* $\oplus$ *brown fur* could indicate an *arctic fox*). However, logic operations are non-differentiable by themselves, and are non-trivial to integrate inside neural network models in an end-to-end learnable manner. While there have been a few recent efforts on integrating logic into deep neural networks, they focus either on post-hoc analysis [3, 23, 29] or are focused on textual/tabular data [17]. We seek to address this need for integrating logic operations into concept-based models (CBMs) in an end-to-end learnable manner in this work. To this end, we propose LogicCBMs, a variant of CBMs where we introduce differentiable fuzzy logic gates that learn logic-based relationships between concepts and classes (as shown in Fig. 1). LogicCBMs are end-to-end learnable, thus not compromising on classification performance, while allowing for rich interpretability of the concept-class relationships in terms of logic gates (see Fig. 2). Our studies show that logic gates are *interpretable non-linearities*, and adding them in the network can be viewed as enabling the model with more capacity while retaining interpretability. Our method can be viewed as an initial effort that provides a pathway for integrating symbolic reasoning within concept-based learning architectures. Our key contributions can be summarized as follows:

- We introduce LogicCBMs, a technique to introduce logic operations into concept-based learning models in an end-to-end differentiable manner. The proposed approach builds logical compositions of concepts (predicates) and relates them to classes.
- We provide a simple and efficient methodology to implement LogicCBMs that provides promising quantitative and qualitative improvements over existing concept-based models.
- We perform comprehensive experiments on well-known benchmark datasets for concept-based learning (CUB, CIFAR100 and AWA2) that demonstrate the promise of learning logic operations and show improved performance, not only in terms of quantative metrics but also qualitative interpretability analysis. Additionally, we also introduce a new worst-case analysis metric, Concept Correction Gain, to corroborate the usefulness of adding logic to concept-based architectures. As part of our empirical studies, we also introduce a simple synthetic dataset: *CLEVR-Logic* to study logical relationships between concepts for validation of such methods in future.
- Our code and datasets will be made publicly available upon acceptance.

2. Related Work

Concept-based Models. Learning classes via concepts has been an actively growing area of research in recent years. Originally introduced in [14], later methods improved the method by addressing concept leakage [22], introducing a bypass concept layer [30], including uncertainty quantification [12], incorporating robustness [37], improving interactivity [2], and extracting concepts that are more amenable to composition [39]. More recent efforts attempted the use of LLMs for concept guidance and expert annotations in [25, 47]. Other efforts have included increasing model capacity by adding additional unsupervised concepts [20, 31], building concept bases for such models [48] and making black-box models intervenable [16]. Inter-concept relations in these models have been studied from a concept representation space perspective [28], from leveraging concept correlations [41], and from an energy-based modeling perspective [45]. However, none of these efforts model the interactions between concepts that drive the predictive performance of a network, we present the integration of logic operations as a solution.

Logic-based Explainability. Concurrently, away from concept-based literature (focus of our work), there have been efforts to extract the underlying logic of a given model. Logic-explained networks [3] operate on interpretable features, however, their logic explanations are obtained by building a truth table of binarized concepts from a fully trained model. SELOR [17] integrates logic into the model design via a probabilistic framework for logic rule generation; their work focuses however on tabular or textual data alone. [23, 29] proposed post-hoc techniques for generating logical compositions of explanations. In contrast to these efforts, our proposed framework works learns the underlying logic structure among intermediate concepts emergent during training via differentiable logic on visual inputs.

Differentiable Logic and Neurosymbolic Methods. Efforts in integrating logic into neural network models is a nascent area, with most previous efforts focusing on neurosymbolic approaches for tasks like inductive logic programming [34], reinforcement learning [50] and abstract reasoning [35]. Some of these efforts pre-specify a set of rules, learning their probabilities differentiably [18, 21], while a few others extract logic rules from examples by defining some kind of continuous relaxation over a discrete space such as the space of first-order logic (FOL) programs [49]. Fewer efforts attempt to use differentiable logic for better representation learning, viz. for better entity representation in knowledge graphs [5] and for learning FOL rules for knowledge-based reasoning [46]. On the other hand, our work focuses on integrating differentiable learnable logic into concept-based models, which allows a pathway to perform logic-based classification through concepts

that capture intermediate latent semantics in a given model. Appendix Tab. A9 provides a more comprehensive comparison.

3. LogicCBMs: Methodology

Preliminaries and Notation. We follow the setup introduced by [14] and define a concept-based model as a model that learns a mapping from $X \mapsto Y$ via an intermediate concept encoder $g(\cdot)$. These models learn from a three-tuple dataset $\{X, C, Y\}$ where $X \in \mathbb{R}^m$, $C \in \mathbb{R}^k$, $Y \in \mathbb{R}^n$ and m, k, n correspond to the dimensionalities of the image, concept and label spaces respectively. Each prediction is of the form $\hat{y} = f(g(x))$ where $g\colon X \mapsto C$ (e.g. *bird* image $\rightarrow$ {*white body*, *flat yellow bill*, ..., *orange legs*}) is the concept encoder, and $f\colon C \mapsto Y$ (e.g. {*white body*, *flat yellow bill*, ..., *orange legs*} $\rightarrow$ *Duck*) is an interpretable predictor network..

Conceptual Framework. We define a *predicate* to be a logical composition of concepts of the form z_i = ($c_1 \circ\mathrm{p}_1 c_2 \circ\mathrm{p}_2 c_3$ ...), where $c_i \in C$ is a concept and $\circ\mathrm{p}_j$ is a logic gate operation; for example, (*big* $\oplus$ *small*), (*orange* $\wedge$ *vegetable* $\wedge$ *healthy*). We are interested in learning the set of predicates logically entailed by each class:

$$y_i \leftarrow w_1.z_1 + w_2.z_2 + w_3.z_3 + ... \tag{1}$$

For example, as shown in Fig. 1, the *Cat* class could assign a higher weight to predicates such as (*Furry* $\wedge$ *Whiskers*), (*Mammal* $\implies$ *Tetrapod*) and $\neg$(*Soft Nails*).

Fig. 3 shows the overall architecture of our approach. As shown in the figure, we do not make any changes in $g(\cdot)$, the backbone concept extractor, thus following a standard vanilla CBM in how it obtains concepts from input image samples. We introduce a logic module (which can consist of multiple logic gate layers) following the concept encoder to extract logical predicates, which are then sent to a classifier. We now write each prediction as: $\hat{y} = f(h(g(x)))$, where $h\colon \{0,1\}^k \mapsto \mathbb{R}^p$ ({*white body*, *flat yellow bill*, *orange legs*} $\rightarrow$ {*white body* $\wedge$ *orange legs*, $\neg$ *mammal*}) and $f\colon \mathbb{R}^p \mapsto \mathcal{Y}$ ({*white body* $\wedge$ *orange legs*} $\rightarrow$ *'Duck'*), where p is the number of logic predicates.

Learning from Differentiable Predicates. For convenience of presentation, without loss of generality, we discuss the rest of our methodology for a logic module comprising one logic gate layer. Consider the logic gate layer to be composed of p logic gate neurons, where each neuron is one of q possible logic gates. There are two steps involved to learn predicates: (i) *Concept pairing*, and (ii) *Differentiable logic learning*. We describe each of these below.

(i) *Concept Pairing*: We introduce two weight matrices: a *concept pair matrix* $CP_{p\times k}$ and a *logic gate matrix* $G_{p\times q}$. In this work, for convenience, we consider each logic gate neuron to take in one pair of concepts and thus form binary predicates. Stacking successive logic gate layers enables us to compose binary predicates to form more intricate n-ary predicates (which we show some initial results on later in this work, and is also an interesting future direction). As shown in Fig. 3, in order to determine these pairs, we extract the two concepts with the highest weight per row, which gives us a set of p concept pairs.

(ii) *Differentiable Logic Learning*: Logic gates are by themselves non-differentiable operations, and hence pose a constraint in incorporating them into end-to-end learnable architectures. In order to overcome this constraint, we take inspiration from fuzzy logic. In particular, we use t-norms, as in [13, 27], which are fuzzy versions of logic gates to allow differentiability and backpropagation while passing the pairwise activations of the concept encoder through the logic layer. For example, if $z_1 = c_a \oplus c_b$ (XOR operation) and c_a, c_b are concept activations, we can approximate this with the t-norm $c_a + c_b - 2 \cdot c_a c_b$. We use $q = 16$ logic gate operations (as in [27]) in our work herein (these gates are listed in Tab. A16 in the Appendix). (While this list of gate operations is inherited from [27], our technical contributions lie in carefully designing and connecting the concept layer with the logic operations in a seamless learnable manner.)

For each of the p logic neurons, we compute all q fuzzy logic operations on the concept pair assigned to it and learn a probability distribution to represent how important a logic gate is for that concept pair. The probability distribution for each logic neuron is captured by the G matrix. The activation of a logic neuron then is the maximum of the weighted logic operation outputs, given by:

$$\hat{z_i} = \max_{i=1,\ldots,q} (g_i \cdot z_i(c_a, c_b)), \quad \sum_{i=1}^{q} g_i = 1,\ g_i \geq 0 \tag{2}$$

where, c_a and c_b are the concept activations input to the i^{th} logic neuron, g_i is weight distribution and $\hat{z_i}$ is the activation of the i^{th} logic neuron (from the G matrix). Once the predicate activations are obtained, the model learns a predicate-class mapping using a linear layer.

$$F_i = \sum_{j=1}^{p} V_{ij} \cdot \hat{z_j} \text{ where, } 1 \leq i \leq n,$$

$$\hat{\mathbf{y}} = \sigma(F) \quad \text{where, } F_i \text{ is a logical formula.} \tag{3}$$

where $\hat{z_j}$ is the j^{th} logic neuron's activation. V denotes the linear layer's parameters, and σ is the softmax activation function. Each F_i is a weighted sum of the learned predicates, which we refer to as a logic formula.

Overall Training Procedure. Our proposed architecture with the logic module does not entail any other loss terms, beyond the standard ones. This is in line with our objective to integrate logic learning seamlessly inside a concept-based learning model. Similar to existing efforts, the overall model is trained for concept classification using a binary

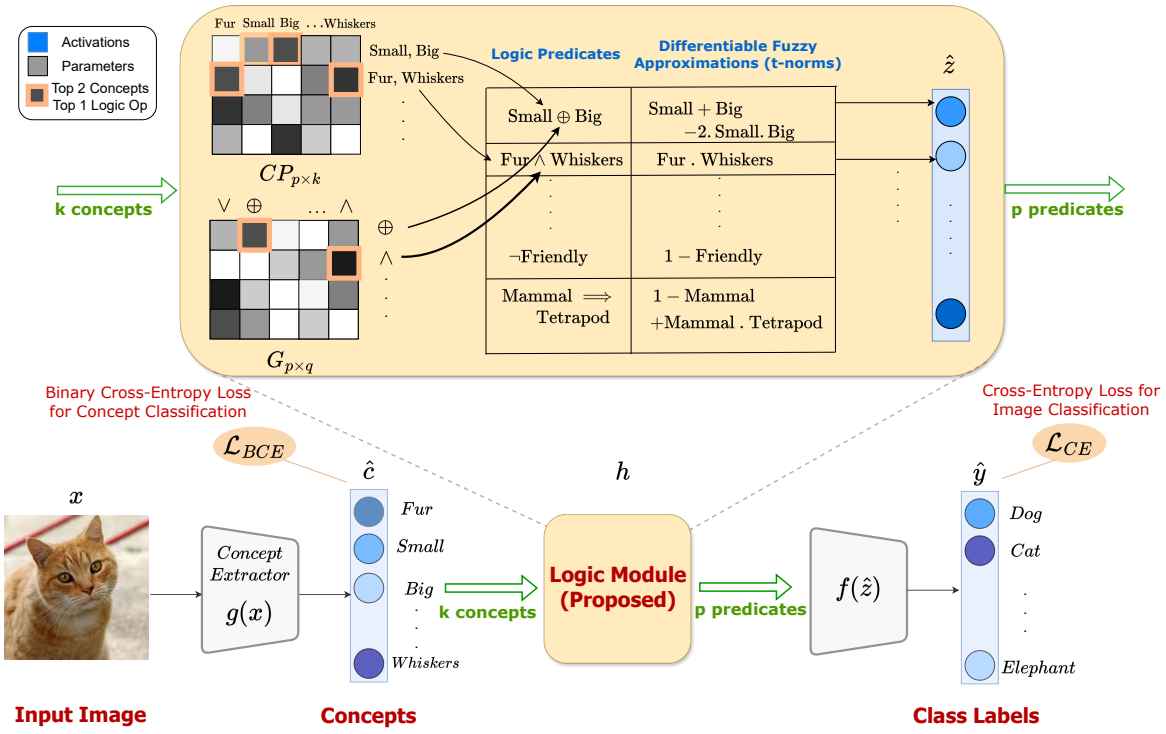

Figure 3. **LogicCBM Architecture.** The proposed logic module/layer is added to a CBM after the concept layer $g(\cdot)$. To implement the logic module, we use two matrices: CP (Concept Pairs) and G (Logic Gates) after the concept layer to learn predicates using differentiable fuzzy logic operations. Our framework is end-to-end differentiable with a subsequent linear layer $f(\cdot)$ that learns the final predicate-class mapping to output the class label prediction.

cross-entropy loss ($\mathcal{L}_{BCE}$) (at the concept layer), and a final standard prediction cross-entropy loss ($\mathcal{L}_{CE}$) (at the output classification layer). Our overall loss thus remains similar to a CBM, given by:

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha \cdot \mathcal{L}_{BCE} \tag{4}$$

where α is a weighting hyperparameter.

4. Experimental Results and Analysis

Datasets: We perform our experiments on a total of seven datasets, comprising standard natural benchmark datasets for CBMs (*CUB200, AwA2, CIFAR100*), a large scale scene recognition dataset (*SUN*) as well as synthetic datasets explicitly used to study the learned logic (*XOR, 2XOR, CLEVR-Logic*). Our natural datasets capture different levels of concept supervision: Caltech-UCSD Birds (CUB, [42]), Animals with Attributes (AwA2, [43]) and CIFAR100 [15]. CUB and AwA2 have concepts for the classes annotated at an instance-level and class-level respectively, while for CIFAR100, we acquire class-level annotations from an LLM following the procedure outlined by [25]. The SUN attribute dataset [26] is also instance-level annotated. More dataset details are in Sec. A1 in the Appendix.

Baselines: We compare our approach with well-known concept-based learning models, in particular: (1) Vanilla [14], MLP and Boolean CBMs [7] (2) Label-Free CBMs [25], (3) Posthoc CBMs [48], (4) Sparse CBMs [33] and (5) VLG CBMs [38]. Vanilla and Boolean CBMs use soft and hard concepts [7] in the training process of a CBM. An MLPCBM is a variant of a Vanilla CBM that replaces the linear concept-to-class mapping with a multilayer perceptron (MLP); one would expect this approach to capture richer concept-class relationships than a simple linear layer. Label-Free CBMs propose an LLM-based class-level concept annotation method, where they use CLIP-Dissect [24] for concept alignment. Posthoc CBMs propose a method for converting a blackbox model into a CBM using concept activation vectors [11]. Sparse CBMs propose training CBMs using contrastive learning and self-supervision. Finally, VLG-CBMs use grounded open-domain object detectors to enhance the faithfulness of concept learning.

Results: Tab. 1 shows the experimental results on standard benchmark datasets. Our method provides an end-to-end training pipeline while outperforming prior approaches as shown in the table. We note that our implementation uses a single logic module/layer (we stick to binary predicates herein; extending to n-ary predicates would be an interesting future direction), and thus use consistently lesser pa-

MODEL	CUB	AWA2	CIFAR100
VANILLA CBM [14]	75.20 ± 0.79	88.81 ± 0.52	55.39 ± 0.62
MLP CBM [7]	72.63 ± 0.25	89.15 ± 0.23	65.00 ± 0.31
BOOLEAN CBM [6]	63.57 ± 0.27	82.97 ± 0.33	47.40 ± 0.82
LFCBM [25]	74.29 ± 0.24	89.76 ± 0.20	65.16 ± 0.14
POSTHOC CBM [48]	64.65 ± 0.08	89.14 ± 0.04	51.33 ± 0.02
SPARSE CBM [33]	70.28 ± 1.05	84.34 ± 0.12	61.68 ± 1.00
VLG-CBM [38]	60.38 ± 0.00	-	65.73 ± 0.00
LOGICCBM (OURS)	**81.13** ± 0.42	**90.04** ± 0.05	**68.46** ± 0.45

Table 1. Validation accuracies (%) on the CUB, AwA2, and CIFAR100 datasets averaged over 3 seeds. Results for VLG-CBMs were taken from their paper and don't report results on AwA2.

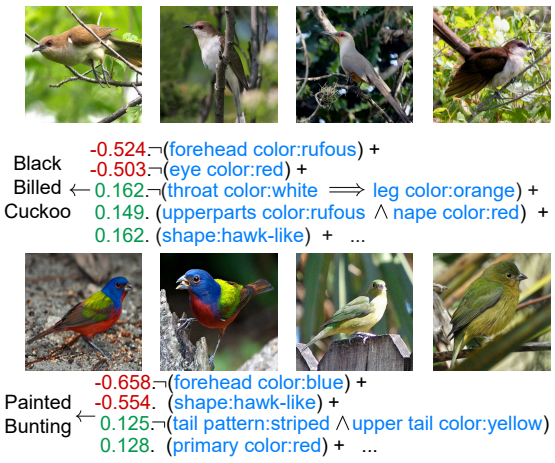

Figure 4. Qualitative results showing examples of the class-level logic captured by LogicCBM on the CUB dataset

rameters than a Vanilla CBM across datasets (please see Tab. A13 in the Appendix). Our experiments with the concept pairing (CP) matrix also showed that both learned concept pairings and random concept pairings performed equivalently across datasets; we hence use the random pairing for its efficiency in practice. We hypothesize this works due to two key factors: (i) the subsequent learning process builds on this concept pairing to learn suitable predicate-class relationships; and (ii) there is sufficient redundancy in the logic module which allows appropriate logic relations to be learned. Fig. 4 shows qualitative results of our method on the CUB dataset (more such results are included in Sec. A3 in the Appendix). As observed from the figure, our logic module provides a concise, yet expressive set of interpretable units (predicates), which get the highest weight in our LogicCBM.

For completeness of this discussion, we also performed a focused study on comparing our LogicCBMs with a different family of recent methods that extract logic explanations from interpretable features – LENs [3]. Note that this method is not intended for end-to-end logic-based predictions (and hence is different from our core focus); however, for completeness of comparison, we train a ψ-net [3]

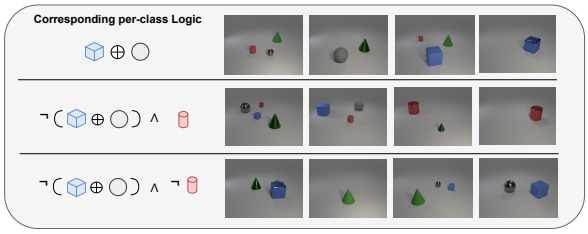

Figure 5. Sample images from our CLEVR-Logic dataset along with corresponding logic used to generate them.

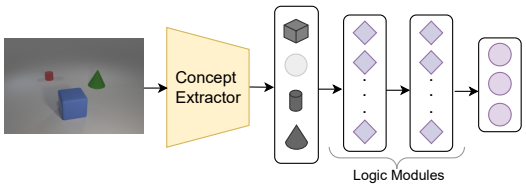

Figure 6. LogicCBM architecture for the CLEVR-Logic dataset. Polyhedra denote concepts, diamonds denote logic neurons, circles denote classes.

and a LogicCBM directly on concept ground truths (without any backbone network for concept extraction) from the CUB dataset. LogicCBMs outperformed the ψ-net (64% vs 51%), validating its usefulness for effectively learning concept-class relationships.

4.1. Do these models learn meaningful logic?

Since real-world datasets do not have explicit logic supervision, it is not straightforward to assess the correctness of the logic. We hence examine our method on synthetic datasets where the logic used to generate the data is known, allowing for validation of correctness of the model's learned logic.

Datasets, Baselines and Metrics: We use three synthetic datasets: *XOR* (proposed in [4]), *2XOR* (a more complex version of XOR which we created that computes the XOR operation on three inputs) and *CLEVR-Logic*, a new variant of CLEVR [10] which we generated to study logical relations among objects in images. In *CLEVR-Logic* we define concepts to be a set of *CLEVR* objects (sphere, cone, cube, cylinder) and specify classes as logical operations among these objects, which defines what is in the image (for example, *sphere* $\oplus$ *cone* could be one class which has images that exclusively contain either a *sphere* or a *cone*). CLEVR images are then generated per class following the class-specific logic. Sample images and their corresponding logical relationships are shown in Fig. 5. More details are included in Sec. A2 in the Appendix. All our code and datasets will be made publicly available upon acceptance.

Implementation Details: For the *XOR* and *2XOR* datasets, our objective in the training process is to learn the correct logic predicate among the options (XOR is one of the possible logic gates in our logic module). To this end, we train

Dataset	DCR[1]	Our Method (LogicCBMs)
XOR	$(c_1 \wedge \sim c_2) \vee (\sim c_1 \wedge c_2)$	$c_1 \oplus c_2$
	# P : 180	# P : 16
2XOR	$(\sim c_0 \wedge \sim c_1 \wedge c_2)$ $\vee (\sim c_0 \wedge c_1 \wedge \sim c_2)$ $\vee \ldots$	$c_1 \oplus c_2 \oplus c_3$
	# P : 783	# P : 48

Table 2. Logic rules learned (along with number of parameters used: #P) by DCR [1] and our method on *XOR* and *2XOR* datasets

GT Rule	DCR[1] Error	LCBM Error	LCBM Rule
$c1 \oplus c2$	0.4 % $\pm$ 0.06	**0 %** $\pm$ **0.00**	$c1 \oplus c2$
$\neg(c1 \oplus c2) \wedge c3$	0.5 % $\pm$ 0.02	**0 %** $\pm$ **0.00**	$\neg(c1 \oplus c2) \wedge c3$
$\neg(c1 \oplus c2) \wedge \neg c3$	1.7 % $\pm$ 1.5	**0 %** $\pm$ **0.00**	$\neg(c1 \oplus c2) \wedge \neg c3$

Table 3. Error rate results for DCR and our method (LCBM = LogicCBM) on *CLEVR-Logic* dataset (all models trained on 2 seeds). We also show the ground truth logic and logic learned by our model, which matches the ground truth on all these experiments.

single-layer and two-layer LogicCBM models for the two datasets respectively. For *CLEVR-Logic*, we train a concept encoder to capture various CLEVR objects as concepts. The extracted concepts are then passed through two logic modules (since some ground truth logic requires 3-ary predicates, as shown in Fig. 5) containing 15 logic neurons each. Fig. 6 shows our model architecture. The predicates themselves are the class labels here. We also train DCR [1] models as a baseline on all three datasets for comparison.

Results: Tab. 2 presents the results for the *XOR* and *2XOR* datasets. As evident in the table, our models learn more succinct and expressive logic, while also using significantly lesser parameters. The DCR models generate explanations only in terms of AND, OR and NOT, necessitating lengthy explanations, especially evident in the case of *2XOR*. Tab. 3 shows the results (including the ground truth and learned predicates) for multiple runs on the *CLEVR-Logic* dataset. Our LogicCBM model is reliably able to learn the underlying class-level logic. The DCR model misclassifies some input samples, as seen in the error rates shown in the table.

4.2. More Analysis

4.2.1. Logic leads to better concept alignment

Concept activations of samples belonging to the same class are expected to be close to each other. We capture this aspect of a model's behavior using *concept alignment*, where we measure the average degree of similarity of concept activations among samples belonging to the same class. Let n_i denote the number of samples belonging to class i and g indicate the model's concept encoder. We compute concept alignment using cosine similarity as below:

$$CA(i) = \sum_{j_1=1}^{n_i} \sum_{j_2=j_1+1}^{n_i} \cos(g(x_{j_1}), g(x_{j_2})) \qquad (5)$$

We compare Vanilla and LogicCBMs, the results of which are shown in Tab. 4. Our logic-based models have a consistently higher concept alignment across datasets. We provide some class-level analysis of this in the Appendix (Sec. A3).

Method	CUB	AwA2	CIFAR100
Vanilla CBM	0.8619 $\pm$ 0.001	0.9754 $\pm$ 0.021	0.9936 $\pm$ 0.007
LogicCBM	**0.9284** $\pm$ 0.001	**0.9810** $\pm$ 0.026	**0.9996** $\pm$ 0.000

Table 4. Mean concept alignment scores on Vanilla CBM and LogicCBM. The higher the better.

4.2.2. Logic can be used for finetuning

We studied the possibility of using our logic module to finetune a pre-trained concept-based model. In certain use cases, it is possible that we may not want to train a LogicCBM from scratch (or architecturally change an existing concept-based model), but may be interested in enhancing an existing model's performance with additional logic. To this end, as shown in Fig. 7, we propose the use of our logic module in a separate logic classification head in the architecture (f_2), which takes in concept activations during finetuning. The baseline CBM's backbone can now leverage the logic head's gradient as well. The training objective in this case hence is given by:

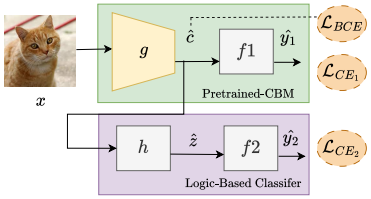

Figure 7. Proposed architecture to use logic for finetuning an existing CBM.

$$\mathcal{L} = \mathcal{L}_{CE_1} + \alpha \mathcal{L}_{CE_2} + \beta \cdot \mathcal{L}_{BCE} \qquad (6)$$

where a cross-entropy loss is used over both the heads (original and logic head), where α and β are weighting hyperparameters. Our results on Vanilla CBMs using this approach are reported in Tab. 5 and show consistent gains and promise in such an approach.

Model	CUB	AwA2	CIFAR100
Vanilla CBM	75.20 $\pm$ 0.79	88.81 $\pm$ 0.52	55.39 $\pm$ 0.62
Vanilla CBM + L	79.21 $\pm$ 0.08	89.56 $\pm$ 0.07	65.89 $\pm$ 0.72

Table 5. Classification accuracy of CBMs obtained by finetuning using our logic module (LF: Logic for Finetuning).

4.2.3. Logic makes interventions more effective

One way to study the correctness of the learned logic using LogicCBMs is to intervene on concepts in test samples, and observe the outcomes. In order to study this, we perform interventions on misclassified samples at test time, where we randomly choose k concepts (output of concept layer) obtained from the respective data sample and replace them with their ground truths. On all baseline models, we perform $k \in \{4, 8, 10\}$ such test-time interventions; and on our LogicCBM model, we perform k/2 predicate interventions (i.e. $\{2, 4, 5\}$) for fairness of comparison, as each logic predicate is composed of two concepts.

Also, as logic predicates do not have direct ground truths available, we manually replace the predicate in the misclassified sample with the expected logic neuron operation on the corresponding ground truth concepts. We perform this evaluation on CUB, as it is relatively the most difficult dataset. Fig. 8 shows the the ratio of intervention successes as the fraction of the number of misclassified samples that change to the correct prediction (after this intervention), among all misclassified samples for a given method. Ideally, a higher ratio would indicate a better concept-class relationship learned. As the plot shows, we see that LogicCBMs have the most effective successful interventions. Some baseline methods (Posthoc and Sparse CBMs) don't have concept ground truths at a sample/class level, and hence could not be reported herein.

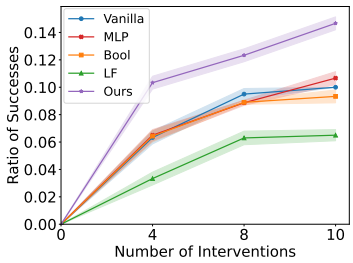

Figure 8. Ratio of test-time intervention successes on baselines vs our CBM method on the CUB dataset. x-axis indicates number of interventions performed.

4.2.4. The diversity of logic gates used matters

As stated earlier, the number of logic gates used in LogicCBMs are $q = 16$ (as listed in Tab. A16 in the Appendix). Fig. 9 shows the distribution across logic gates learned by the models on the CUB and AwA2 datasets (plots for CIFAR100 is in Appendix Sec. A3). The bar plots show the use of all gates in each of the datasets. Some gates like NOT are relatively less used on some of the datasets, possibly because the other predicates had a stronger impact on the prediction. To study this further, we reduce the number of logic gate types used to 8, thus reducing the diversity of possible logic operations, and study the accuracy on each of these datasets. In particular, the set of 16 logic gates is pruned to 8 by retaining only the simpler operations

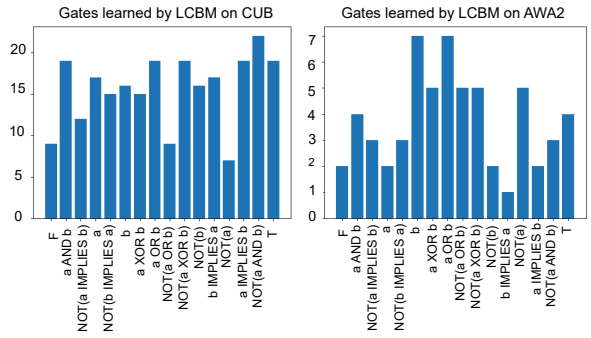

Figure 9. The distribution of logic gates learned by our LogicCBM models on the CUB and AwA2 datasets. See Appendix Sec. A3 for results on CIFAR100.

#Logic Types	CUB	AwA2	CIFAR100
16	**81.13** ± 0.42	**90.04** ± 0.05	**68.46** ± 0.45
8	63.32 ± 0.62	86.39 ± 0.61	59.45 ± 0.11

Table 6. Drop in accuracy on reducing q (number of logic operations considered) from 16 to 8. Results indicate the importance of diversity of logic gates used.

$(\wedge, \vee, 1, 0, c_1, c_2, \neg c_1, \neg c_2)$ and removing the more expressive ones. Tab. 6 shows the results of this study, where we observe that the reduced number of logic gate types used has a marked impact on performance with a considerable drop in accuracy across datasets, particularly evident on CUB which we attribute to its fine-grained nature. This highlights the use of a diverse set of logic gates for better performance. Exploring an optimal set of logic gates for a given task can be another interesting direction of future work.

4.2.5. Scaling to 500+ classes: LogicCBMs for scene recognition

In order to study how LogicCBMs scale, we conduct experiments on the SUN attribute dataset [26], a large-scale scene recognition dataset with 500+ categories. LogicCBMs match the performance of a Vanilla CBM ($\approx$ 85% validation accuracy on both models). Importantly, we observed that the qualitative results showed improved performance; an example of the logic captured by our model is shown in Fig. 10 along with the learned class-level logic gate distribution (more such results are in Appendix Sec. A3). This is also reflected in our other metric in Appendix Tab. A8. We obtain the class-level logic distribution by running inference on all samples belonging to a chosen class and averaging the maximum activating logic gate for each logic neuron over all samples.

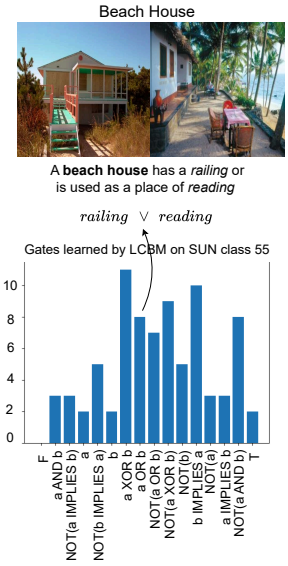

Figure 10. An example predicate learned by a LogicCBM on the SUN Attribute dataset for the *Beach House* class. The bar plot indicates the distribution of logic gates learned for the class.

4.3. CCG: A Worst-Case Metric for CBMs

Concept-based interpretable models are especially of importance in high-risk scenarios where there may be a high cost associated with an erroneous decision. We hence propose a metric to study CBM-related models in such a worst-case setting to assess the quality of concept/predicate-class

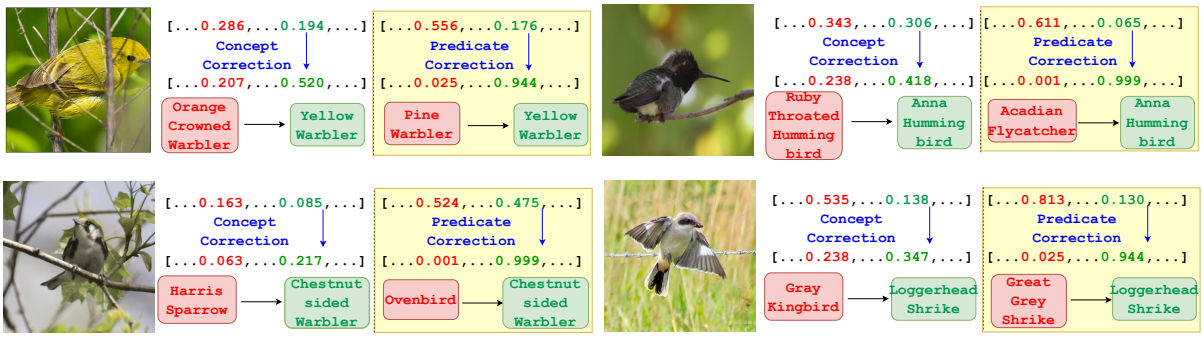

Figure 11. Some examples of pre- and post-correction confidences of a Vanilla CBM (left) and a LogicCBM (right, highlighted in yellow). Note the significantly improved confidences of the LogicCBM model post correction.

relationships they have learned. Our evaluation is based on test-time corrections of concepts (or predicates). For each sample, we identify the most misleading concept (or predicate) and turn it off (or on, depending on ground truth for a given example). We then measure the resulting change in the model's prediction confidence, where confidence is measured as the softmax output of the ground truth class. For example, certain species of dogs have pointy ears, although it is more common in cats. If a model incorrectly predicts *'cat'* for such an image, we can test whether turning off the misleading concept *pointy ears* increases the model's confidence in the correct class (*'dog'*). A significant positive change in model confidence represents correctness of the learned concept-class relationships. We define this metric as *Concept Correction Gain* (CCG). Ideally, the CCG would be high, indicating sensitivity to the right semantic cues. CCG is similar in principle to activation patching [9]. While activation patching is a mechanistic interpretability technique to analyze the behavior of a model's components, CCG quantifies a model's confidence when some misleading information is removed. Formally:

$$\text{CCG} = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \left[\sigma(\check{y}_i)_{y_i^*} - \sigma(\hat{y}_i)_{y_i^*} \right] \tag{7}$$

where, y_i^* is the ground truth class from the i^{th} sample, $\check{y}_i$ and $\hat{y}_i$ are the pre and post correction probability distributions over the classes and $|\mathcal{C}|$ is the number of samples that get corrected when the chosen concept (or predicate) is replaced by its ground truth. In baseline concept-based models, we replace the most misleading concept by its ground truth presence (or absence). In our LogicCBM, since we do not have predicate ground truths, we consider the logic gate of the most misleading logic neuron and compute the selected logic operation on the ground truths of its constituent concepts. This is further described in the Appendix (Sec. A2.1) along with how the misleading concept/predicate is estimated. Note that when we make a single correction in a LogicCBM predicate, for fairness of comparison, we make two corrections in other baseline methods. Tab. 7 reports the CCG metric for LogicCBM and baseline models (other than VLG-CBMs since we don't have access to their models) on the CUB dataset (more results are in Appendix Sec. A2). LogicCBMs significantly outperform other methods, showing their utility in potentially high-risk scenarios that require interpretability. While these numbers provide a population-level view, we also show some qualitative results at a sample level that examine the pre- and post-correction confidences of these models in Fig. 11.

Model	CCG
Vanilla CBM	0.2102
MLP CBM	0.117
Boolean CBM	0.308
LFCBM	0.2491
Posthoc CBM	0.293
Sparse CBM	0.358
LCBM (Ours)	**0.5228**

Table 7. Our CCG metric values on different baseline models on the CUB dataset. Higher the value, the more responsive the model is to corrections.

5. Conclusion

In this work, we presented LogicCBMs, a new pathway to integrate logic into concept-based learning models. We leverage differentiable fuzzy logic operations integrated into such models to predict a model's classification output through logical compositions over intermediate semantics defined by concepts. To study the performance of our logic-enhanced concept-based models, we perform comprehensive experiments on well-known standard benchmarks as well as on syntheic datasets including a new one we introduce, *CLEVR-Logic*. We also provide a worst-case analysis metric (CCG) that can help support further studies in this area. Logic gates are a way of giving the model more modeling capacity without losing interpretability. Our experiments show that these models improve on multiple model metrics beyond accuracy including better concept alignment, effective interventions and receptivity to corrections. We believe that our work provides a new dimension to concept-based learning and can improve a model's overall performance by giving it the ability to logically express its predictions in terms of semantic symbols.

Acknowledgments. Deepika SN Vemuri would like to thank PMRF for the fellowship support. We thank the anonymous reviewers for their helpful feedback in improving the presentation of the paper.

References

[1] Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Mateo Espinosa Zarlenga, Lucie Charlotte Magister, Alberto Tonda, Pietro Lió, Frederic Precioso, Mateja Jamnik, and Giuseppe Marra. Interpretable neural-symbolic concept reasoning. In *International Conference on Machine Learning*, pages 1801–1825. PMLR, 2023. 6

[2] Kushal Chauhan, Rishabh Tiwari, Jan Freyberg, Pradeep Shenoy, and Krishnamurthy Dvijotham. Interactive concept bottleneck models. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 2023. 2

[3] Gabriele Ciravegna, Pietro Barbiero, Francesco Giannini, Marco Gori, Pietro Liò, Marco Maggini, and Stefano Melacci. Logic explained networks. *Artificial Intelligence*, 314:103822, 2023. 2, 5

[4] Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, et al. Concept embedding models: Beyond the accuracy-explainability trade-off. *Advances in Neural Information Processing Systems*, 35:21400–21413, 2022. 5

[5] Chi Han, Qizheng He, Charles Yu, Xinya Du, Hanghang Tong, and Heng Ji. Logical entity representation in knowledge-graphs for differentiable rule learning. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[6] Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. In *Advances in Neural Information Processing Systems*, pages 23386–23397. Curran Associates, Inc., 2022. 1, 5

[7] Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. In *Advances in Neural Information Processing Systems*, 2022. 4, 5

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[9] Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*, 2024. 8

[10] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017. 5

[11] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning*, pages 2668–2677. PMLR, 2018. 4

[12] Eunji Kim, Dahuin Jung, Sangha Park, Siwon Kim, and Sung-Hoon Yoon. Probabilistic concept bottleneck models. *ArXiv*, abs/2306.01574, 2023. 2

[13] George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*. Prentice hall New Jersey, 1995. 3

[14] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. pages 5338–5348. PMLR, 2020. 1, 2, 3, 4, 5

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4, 1

[16] Sonia Laguna, Ričards Marcinkevičs, Moritz Vandenhirtz, and Julia E Vogt. Beyond concept bottleneck models: How to make black boxes intervenable? *arXiv preprint arXiv:2401.13544*, 2024. 2

[17] Seungeon Lee, Xiting Wang, Sungwon Han, Xiaoyuan Yi, Xing Xie, and Meeyoung Cha. Self-explaining deep models with logic rule reasoning. In *Advances in Neural Information Processing Systems*, 2022. 2

[18] Ziyang Li, Jiani Huang, and Mayur Naik. Scallop: A language for neurosymbolic programming. *Proceedings of the ACM on Programming Languages*, 7(PLDI):1463–1487, 2023. 2

[19] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc. 1

[20] Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models. *arXiv preprint arXiv:2106.13314*, 2021. 2

[21] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31, 2018. 2

[22] Emanuele Marconato, Andrea Passerini, and Stefano Teso. Glancenets: Interpretabile, leak-proof concept-based models. In *Neural Information Processing Systems*, 2022. 2

[23] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2020. Curran Associates Inc. 2

[24] Tuomas Oikarinen and Tsui-Wei Weng. CLIP-dissect: Automatic description of neuron representations in deep vision networks. In *The Eleventh International Conference on Learning Representations*, 2023. 4

[25] Tuomas Oikarinen, Subhro Das, Lam M Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. *arXiv preprint arXiv:2304.06129*, 2023. 1, 2, 4, 5

[26] Genevieve Patterson, Chen Xu, Hang Su, and James Hays. The sun attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision*, 108(1-2):59–81, 2014. 4, 7

[27] Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Deep differentiable logic gate networks. In *Advances in Neural Information Processing Systems*, 2022. 3, 6

[28] Naveen Raman, Mateo Espinosa Zarlenga, and Mateja Jamnik. Understanding inter-concept relationships in concept-based models. *arXiv preprint arXiv:2405.18217*, 2024. 2

[29] Biagio La Rosa, Leilani H. Gilpin, and Roberto Capobianco. Towards a fuller understanding of neurons with clustered compositional explanations. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2

[30] Anirban Sarkar, Deepak Vijaykeerthy, Anindya Sarkar, and Vineeth N Balasubramanian. A framework for learning ante-hoc explainable models via concepts. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10276–10285, 2022. 1, 2

[31] Yoshihide Sawada and Keigo Nakamura. Concept bottleneck model with additional unsupervised concepts. *IEEE Access*, 10:41758–41765, 2022. 2

[32] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017. 1

[33] Andrei Semenov, Vladimir Ivanov, Aleksandr Beznosikov, and Alexander Gasnikov. Sparse concept bottleneck models: Gumbel tricks in contrastive learning, 2024. 4, 5, 2

[34] Prithviraj Sen, Breno WSR de Carvalho, Ryan Riegel, and Alexander Gray. Neuro-symbolic inductive logic programming with logical neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, pages 8212–8219, 2022. 2

[35] Hikaru Shindo, Viktor Pfanschilling, Devendra Singh Dhami, and Kristian Kersting. Learning differentiable logic programs for abstract visual reasoning. *Machine Learning*, 113(11):8533–8584, 2024. 2

[36] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 3145–3153. PMlR, JMLR.org, 2017. 1

[37] Sanchit Sinha, Mengdi Huai, Jianhui Sun, and Aidong Zhang. Understanding and enhancing robustness of concept-based models. In *AAAI Conference on Artificial Intelligence*, 2022. 2

[38] Divyansh Srivastava, Ge Yan, and Lily Weng. Vlg-cbm: Training concept bottleneck models with vision-language guidance. In *Advances in Neural Information Processing Systems*, pages 79057–79094. Curran Associates, Inc., 2024. 4, 5

[39] Adam Stein, Aaditya Naik, Yinjun Wu, Mayur Naik, and Eric Wong. Towards compositionality in concept learning. 2

[40] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 1

[41] Moritz Vandenhirtz, Sonia Laguna, Ričards Marcinkevičs, and Julia Vogt. Stochastic concept bottleneck models. In *Advances in Neural Information Processing Systems*, pages 51787–51810. Curran Associates, Inc., 2024. 2

[42] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge J. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 4, 1

[43] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis ; Machine Intelligence*, 41(09):2251–2265, 2019. 4, 1

[44] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010. 1

[45] Xinyue Xu, Yi Qin, Lu Mi, Hao Wang, and Xiaomeng Li. Energy-based concept bottleneck models: Unifying prediction, concept intervention, and probabilistic interpretations. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[46] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 2

[47] Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19187–19197, 2023. 2

[48] Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023. 1, 2, 4, 5

[49] Matthieu Zimmer, Xuening Feng, Claire Glanois, Zhaohui Jiang, Jianyi Zhang, P. Weng, Hao Jianye, Li Dong, and Liu Wulong. Differentiable logic machines. *Trans. Mach. Learn. Res.*, 2023, 2021. 2

[50] Matthieu Zimmer, Xuening Feng, Claire Glanois, Zhaohui JIANG, Jianyi Zhang, Paul Weng, Dong Li, Jianye HAO, and Wulong Liu. Differentiable logic machines. *Transactions on Machine Learning Research*, 2023. 2