# LogicCBMs: Logic-Enhanced Concept-Based Learning

## Supplementary Material

Our code will be made publicly available upon acceptance for further research and reproducibility. In this part of the paper, we provide additional details of our work, including the following information.

### Table of Contents



## A1. Dataset Details

- **CUB:** The Caltech-UCSD Birds-200-2011 (CUB) dataset [42] is a fine-grained bird species identification dataset. It consists of 11,788 images of 200 bird categories, 5,994 for training and 5,794 for testing. We use the 312 concepts expert-annotated in the dataset representing bird attributes like beak length and shape, body size, wing color, etc.
- **AwA2:** The Animals with Attributes (AwA2) dataset [43] is commonly used for zero-shot learning (ZSL) and attribute-based classification. It consists of 37322 images (26125 training, 11197 testing) of 50 animal classes, annotated with 85 numeric attribute values for each class and is class-level expert annotated.
- **CIFAR100:** CIFAR100 (Canadian Institute for Advanced Research) is a subset of the Tiny Images dataset [15] comprising 100 classes. It consists of 60000 images, 50000 for training and 10000 for testing. Since this dataset does not have expert annotated concepts, we query an LLM to get 925 concepts. We follow the process outlined by [25], where, an initial concept set is generated through queries like "List the most important features for recognizing something as a {class}". This is followed by a sequence of filtering steps, to improve the quality of the concept set.
- **SUN Attribute:** The SUN attribute dataset is build on top of the fine-grained SUN (Scene UNderstanding) categorical database [44] used for scene understanding and scene recognition. It contains 14340 images (we use a 80:10:10 train-val-test split) of 539 classes and 102 crowd sourced attributes.

### A1.1. CLEVR-Logic

As stated in the main paper, we introduce a dataset, CLEVR-Logic, a new variant of CLEVR which we generate to study logical relations among objects in images. We define concepts to be a certain set of *CLEVR* objects (sphere, cone, cube, cylinder) and specify the classes as logical operations among these objects. CLEVR images are then generated per class following the specified class-specific logic. Some example images and the logical relations expected to be learned from them are shown in Fig. 5. We provide some more images in Fig. A13. We found that generating about 20 images per class sufficed for the task considered herein. Our code will be made publicly upon acceptance, and we believe it can be used to generate CLEVR images of greater complexity to capture a wide range of logic operations.

## A2. Architecture and Implementation Details

The feature extractor $g$ used in all the Vanilla and Boolean CBM experiments was an Inception V3 [40], with the concept and classification layers being single linear layers. For all the other baselines, we use their code as is. For the CLEVR experiments, we use a ResNet-18 [8] as the feature extractor. We provide the architectures used for the synthetic dataset experiments in Fig. A12. For the XOR and 2XOR experiments, we train the models on the truth tables of both these operations respectively. All experiments were run on a single NVIDIA GeForce RTX 3090.

### A2.1. Additional Details on Metrics

**Further Details on CCG:** As described in the main paper, CCG is a metric computed to test the worst-case behavior of a model. Concretely, we choose the most misleading concept (or predicate) and turn it off (or on according to its ground truth) and measure the change in the confidence of the model. The most misleading concept (or predicate) is the one that has the most weight difference between the predicted class and ground truth class (Algorithm 1). The CCG computation process is described in Algorithm 2, where we first choose the most misleading concept (or predicate), replace it with its ground truth (or predicate activation computed on its ground truth concepts) and then pass the concept activations through the classifier again to check whether this leads to a correction in the models prediction.

**Algorithm 1** GET_MISLEADING_UNIT$(y, \hat{y}, W)$:

**Require:** Ground truth class $y$, predicted class distribution $\hat{y}$, classifier $f$'s weights $W$
**return** $\text{argmax}_i \, |W[y, i] - W[\hat{y}, i]|$

| Model | AwA2 | CIFAR100 | SUN |
|---|---|---|---|
| Vanilla CBM | 0.377 | 0.078 | 0.252 |
| LogicCBM | **0.485** | **0.410** | **0.616** |

Table A8. CCG comparison between Vanilla CBM and LogicCBM across different datasets.

| | Neurosymbolic Programming Frameworks | Our Framework |
|---|---|---|
| **Logic specification** | Logic rules are pre-specified and fixed, reflecting a symbolic paradigm. | Logic gates are learnable components whose configurations emerge through training. |
| **Differentiability vs. learnability** | Logic is differentiable but not learnable; only rule probabilities are optimized (e.g., in DeepProbLog). | Logic is both differentiable and learnable; gates obtained through gradient-based training. |
| **Symbols / predicates** | Neural predicates need not be interpretable; they may correspond to arbitrary latent factors. | Concepts are explicitly human-interpretable by design. |
| **Interpretability source** | Interpretability arises from the structure of reasoning (e.g., through a domain-specific language). | Interpretability arises from human-defined concepts and their learned logical composition. |
| **Typical application domain** | Structured reasoning tasks, where constraints can be explicitly encoded as logical rules. | Continuous neural domains, such as classification tasks. |

Table A9. Comparison between neurosymbolic programming frameworks and our proposed framework.

**Algorithm 2** Concept Correction Gain Metric

**Require:** Classifier $f$, penultimate layer activations $a$, classifier $f$'s weights $W$, input image-concepts-label tuple ($x$, $c$, $y$)
$\hat{y} \leftarrow f(a)$
$i \leftarrow$ GET_MISLEADING_UNIT$(y, \hat{y}, W)$
**if** concept model **then**
  a[i] := c[i]
**else**
  a[i] := $\hat{z}$[i](c[a1], c[a2])
  ▷ Compute predicate on its constituent concept ground truths (a1, a2)
**end if**
$\breve{y} \leftarrow f(a)$

**Algorithm 3** Correction without concept ground truths

**Require:** Concept activation vector $\hat{c}$, chosen misleading unit $ind$, text concepts $concepts_{text}$, text classes $classes_{text}$):
**if** SIMILARITY($concepts_{text}[ind]$, $classes_{text}$[y]) $>$ $threshold$ **then**
  $\hat{c}[ind] \leftarrow \max \hat{c}$
**else**
  $\hat{c}[ind] \leftarrow \min \hat{c}$
**end if**
**return** $\hat{c}$

We also present some CCG results on the other datasets in Tab. A8 and see that LogicCBMs consistently do better than Vanilla CBMs.

**Performing corrections with no concept ground truths:** Some baseline methods (Posthoc CBMs [48] and Sparse CBMs [33]) in Tab. 1 do not use concept ground truths in their training process. Although the list of concepts considered overall is provided per dataset, there is no class-level or instance-level concept ground-truth. Since we need concept ground truths to perform a correction for CCG, we propose the process outlined in Algorithm 3 to perform interventions on these models. We use a pretrained SentenceTransformer [1] to get the embeddings of the selected (most misleading) text-concepts and the ground truth text-class, and measure the similarity between these embeddings. The concept is then set to the minimum/maximum concept activation using a threshold on this embedding similarity. We use *cosine similarity* to measure similarity and observe that a threshold of 0.5 worked well in our experiments.

## A2.2. Additional Experiments

**Correlated Concepts for Concept Pairing:** Beyond what was presented in the main paper for concept pairing, we also studied our framework by using correlated concepts for concept pairing. We describe this strategy in Algorithm 4, where $n$ random samples are chosen whose concept activations are obtained using the concept encoder $g$. These activations are aggregated and averaged per concept, of which the top two concepts are then estimated to be correlated. This strategy is compared with our original strategy (where we allowed random pairing of concept activations, which allows us to be efficient) in Tab. A10. As shown in the table, this approach does imply some features of the datasets. The accuracies for CUB and AwA2 are nearly the same as the original approach, although there is a drop in performance in CIFAR100. This could indicate that there aren't those many correlated concepts in CIFAR100. This may be an interesting direction to study further.

| Concept Pairing Strategy | CUB | AwA2 | CIFAR100 |
|---|---|---|---|
| Original | 81.55 | 90.08 | 68.97 |
| Correlated | 80.38 | 90.14 | 65.28 |

Table A10. Accuracy results using different concept pairing mechanisms.

**Extension to n-ary Predicates:** In all our experiments in the main paper, we used only one logic gate layer. A key advantage of using our framework is that we can stack multiple logic modules to form more expressive predicates, which can further enhance the models' performance. We studied the use

[1] https://huggingface.co/sentence-transformers

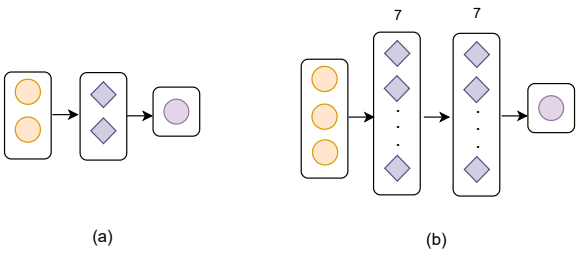


Figure A12. In continuation to Fig. 6 in the main paper, we provide the logic model architectures used for the *XOR* and *2XOR* synthetic dataset experiments. Yellow circles are concepts, diamonds are logic neurons and purple circles are final outputs.

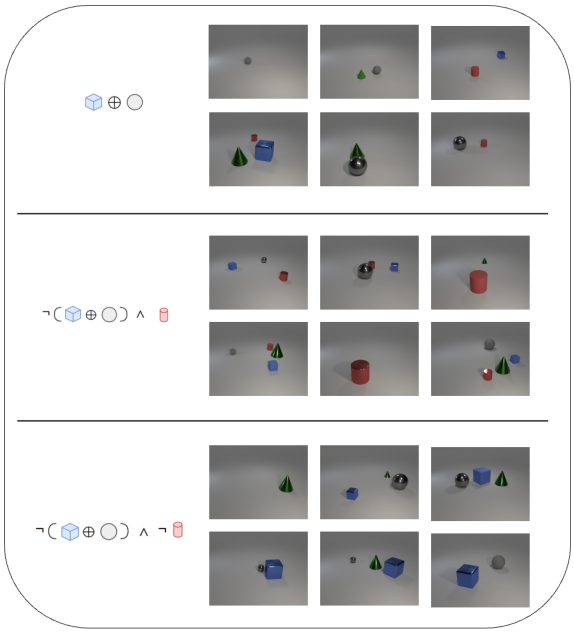

Figure A13. More images from our CLEVR-Logic dataset

**Algorithm 4** Correlated Concept Selection

**Require:** Random samples $S = \{x_1, x_2, \ldots, x_n\}$, concept encoder $g$, concept activation (act) matrix $A \in \mathbb{R}^{n \times h_2}$.
Initialize $C = \mathbf{0} \in \mathbb{R}^m$
**for** $i = 1$ to $n$ **do**
  $A[i,:] \leftarrow g(x_i)$ ▷ Get concept act for sample $x_i$.
  $C \leftarrow C + A[i,:]$ ▷ Accumulate concept acts.
**end for**
$C \leftarrow C/n$ ▷ Average acts across all samples.
$k_1, k_2 \leftarrow$ argtop2$(C)$
**return** $(k_1, k_2)$

of two and three logic modules (specific architectures provided in Tab. A11) and we report results in Tab. A12. The #L = 1 row also corresponds to the architectures of the logic gate layers used in Tab. 1. These results show that extensions to multiple logic modules, while feasible, require more careful tuning and provide interesting directions for future work.

| #L | CUB | AwA2 | CIFAR100 |
|---|---|---|---|
| 1 | 250 × 1 | 60 × 1 | 600 × 1 |
| 2 | 125 × 2 | 30 × 2 | 300 × 2 |
| 3 | 60 × 3 | 15 × 3 | 150 × 3 |

Table A11. Multi-layer logic architectures. #L indicates the number of layers.

| #L | CUB | AwA2 | CIFAR100 |
|---|---|---|---|
| 1 | 81.55 | 89.9 | 68.95 |
| 2 | 77.62 | 89.39 | 67.48 |
| 3 | 58.76 | 86.3 | 67.67 |

Table A12. Results of extending LogicCBMs to $n$-ary predicates with multiple logic layers. (Note that the single-module results are our method's performance in Tab. 1).

| Model | CUB | AwA2 | CIFAR100 |
|---|---|---|---|
| Fuzzy CBM | 312×200 | 85×50 | 925×100 |
| MLP CBM | 312×250×200 | 85×60×50 | 925×150×100 |
| Boolean CBM | 312×200 | 85×50 | 925×100 |
| LFCBM | 290×200 | 72×50 | 892×100 |
| Sparse CBM | 370×200 | 85×50 | 944×100 |
| Posthoc CBM | 312×200 | 85×50 | 440×200 |
| LogicCBM | 250×200 | 60×50 | 600×100 |

Table A13. The number of parameters in the concept to class layer in all baselines. #concepts x #classes.

## A2.3. Hyperparameter Details

All models were trained with a batch size of 64. The number of parameters in $f$ (concept-class layer) are provided for all baselines in Tab. A13. We describe the hyperparameters used for each baseline.

- *Vanilla CBMs:* On CUB, the Vanilla CBMs were trained for 40 epochs, with a learning rate 0.001, weight decay of 1e-4 and $\alpha$ as 0.01 using an SGD optimizer. The AwA2 models were trained for 30 epochs, with the other configuration being the same as CUB. The CIFAR100 models were trained for 40 epochs, with a learning rate of 0.001, weight decay of 1e-4 and $\alpha$ as 0.001 using an Adam optimizer. The SUN models were trained for 50 epochs, with a learning rate of 0.01, weight decay of 1e-4 using an SGD optimizer.
- *LogicCBMs:* All the LogicCBM models were trained with the same corresponding configuration as the Vanilla CBM models for fairness.
- *MLP CBMs:* We trained these models for roughly double the number of epochs as the Vanilla CBMs with all other configurations remaining the same.
- *Boolean CBMs:* All the Boolean CBM models were trained using the same configuration as the corresponding Vanilla

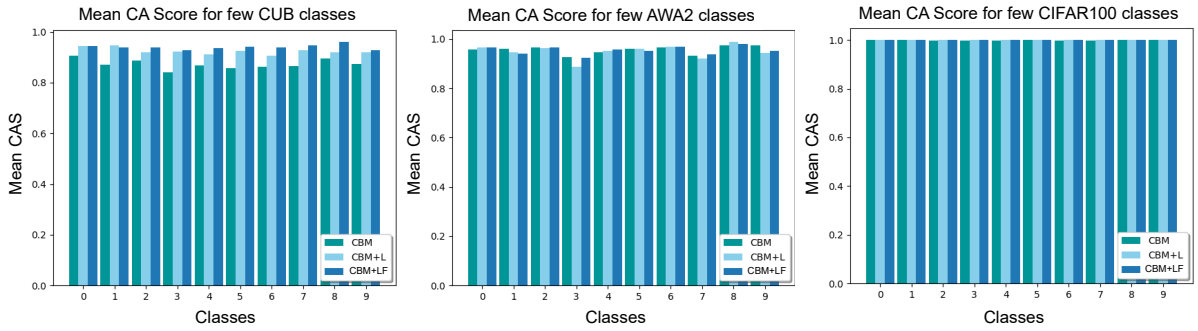


Figure A14. Mean concept alignment scores on 10 random classes (x-axis) of CUB, CIFAR100 and AwA2, on a Vanilla CBM, LogicCBM (CBM+L) and a Vanilla CBM+LF (CBM+LF).

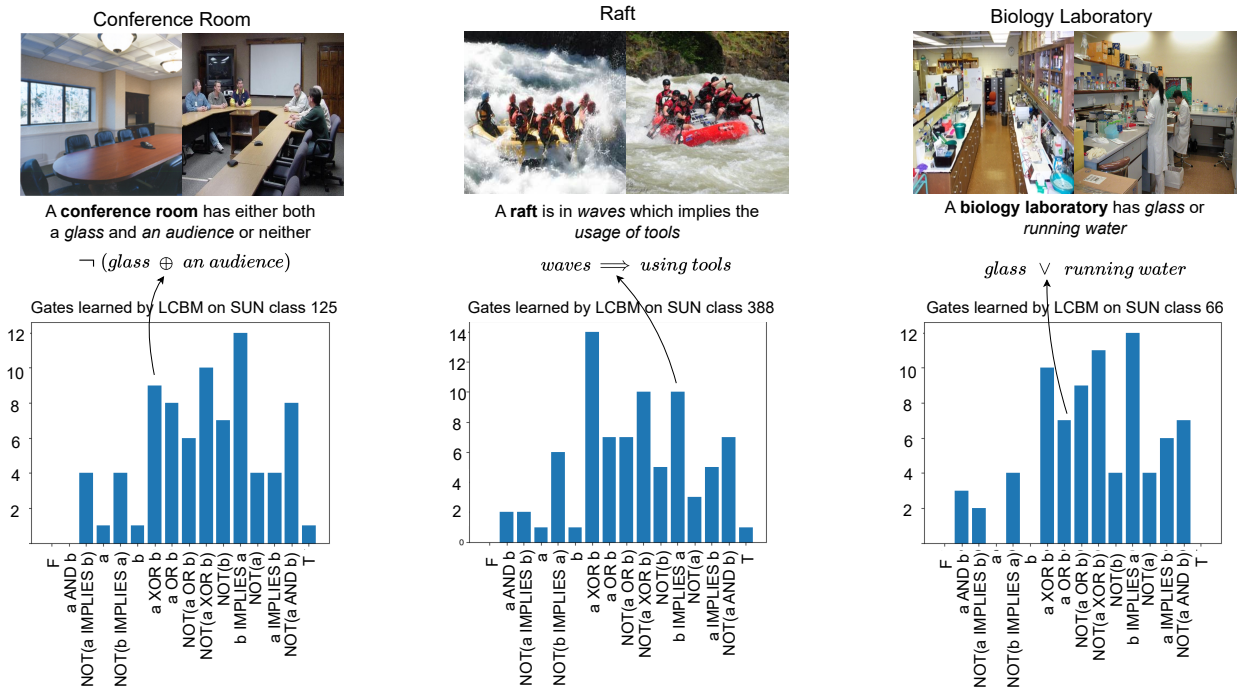


Figure A15. More examples of predicates learned on a LogicCBM for some classes from the SUN attribute dataset, along with the distribution of logic gates learned for those respective classes.

CBM models with the only extra step being that the concept activations are thresholded and made binary before sending them to the classifier.

- *Label-Free CBMs:* We used the default hyperparameters given in the paper for CUB and CIFAR100. For AwA2 the models were trained for 1000 epochs.
- *Posthoc CBMs:* On CUB, the models were trained for 40 epochs, with a learning rate of 0.01. On CIFAR100, the models were trained for 100 epochs, with the same learning rate of 0.001. On AwA2, the models were trained for 100 epochs, with a learning rate of 0.0001.
- *Sparse CBMs:* On CUB, the models were trained for 100 epochs, with a learning rate of 3e-4. On CIFAR100, the models were trained for 40 epochs, with the same learning rate of 3e-4. On AwA2, the models were trained for 20 epochs, with a learning rate of 3e-3.
- *Logic for Finetuning:* For the results presented in Tab. 5, we use the Vanilla CBM models from row 1 of Tab. 1. We finetune the models using our logic module for 15 epochs on CUB, 40 epochs on AwA2 and 10 epochs on CIFAR100.

## A3. More Results and Ablation Studies

**Concept Alignment Score.** In Table 4, we presented results on the mean concept-alignment scores per benchmark averaged over all their respective classes, in order to measure the average degree of similarity among the concept activations of samples belonging to the same class. Here, we present some mean concept-alignment scores per class in Figure A14.

**Logic Layer Size.** In Fig. A17, we present some analysis on the number of logic neurons used in our single logic layer LogicCBM models. As shown in Fig. A17, we see a consistent drop in performance across datasets when $p$ is decreased. The drop is higher on CUB indicating the complexity of the dataset.

**Choice of $\alpha$ and $\beta$** We perform ablation studies on the two hyperparameters: $\alpha$ (LogicCBM) and $\beta$ (Vanilla CBM+LF). The best LogicCBM and best Vanilla CBM+LF models per dataset are considered, and the values of these hyperparameters are varied among [0.001, 0.01, 0.1, 1]. As seen from the results from Tables A14, A15, there is a trend towards a drop in performance as the concept weight is increased.

**More qualitative results.** We provide the distribution of logic gates learned for the CIFAR100 and SUN attribute dataset in Fig. A16. More qualitative results on pre- and post-correction confidences on samples from the three datasets are presented in Fig. A19. We also show some examples of the class-level logic learned for specific classes in Fig. A18, where the learned logic is presented in the form of Equation 1. Finally, we provide some more example predicates on some classes from the SUN attribute dataset in Fig. A15 along with their corresponding learned class-level logic. These results show the promise of our framework.

## A4. Limitations and Future Work

The logic module in all our logic-enhanced models (from Table 1) works with a vector of concept activations (scalars), through the presence of an explicit concept layer. Extending it to work with concept embeddings or multi-modal concepts (a vector of vectors), could be a useful future direction. We hypothesize that this may cause a drop in some interpretability (as there is no longer a well-defined notion of concept activation in such models), and addressing this challenge would be interesting future work. As shown in our results in Table A10, the use of more intelligent concept pairing mechanisms may yield better outcomes, although this might require further studying. Finally, future work could also further explore the two extensions we propose - logic for finetuning and multi-layered logic (that we do some initial experimentation with in this work). We believe that our work herein paves the path for future such efforts in this direction.

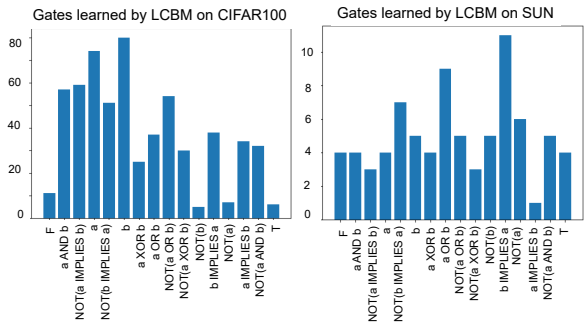


Figure A16. The distribution of logic gates learned by our LogicCBM models on CIFAR100 and SUN.

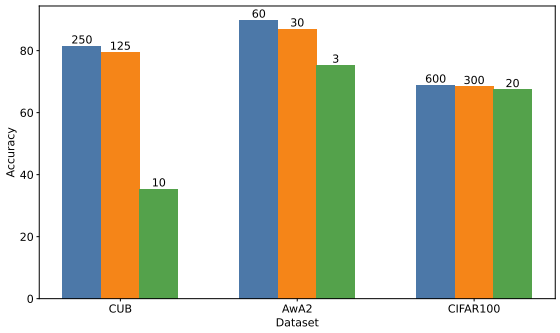


Figure A17. Analysis on the effect of logic layer size on classification accuracy on all datasets.

| Dataset | $\alpha$ | CBM+L |
|---|---|---|
| CUB | 0.01 | 81.55 |
| | 0.1 | 47.16 |
| | 1 | 41.9 |
| AwA2 | 0.001 | 90.05 |
| | 0.1 | 87.64 |
| | 1 | 61.47 |
| CIFAR100 | 0.001 | 68.95 |
| | 0.1 | 60.39 |
| | 1 | 31.48 |

Table A14. Effect of $\alpha$ on CBM+L on accuracy.

| Dataset | Beta | LCBM |
|---|---|---|
| CUB | 0.01 | 81.55 |
| | 0.1 | 65.02 |
| | 1 | 50 |
| AwA2 | 0.001 | 90.05 |
| | 0.1 | 87.18 |
| | 1 | 65.86 |
| CIFAR100 | 0.001 | 68.95 |
| | 0.1 | 56.96 |
| | 1 | 26.94 |

Table A15. Effect of $\beta$ on LCBM (logic for finetuning) on accuracy.

| ID | Operator | real-valued | 00 | 01 | 10 | 11 |
|---|---|---|---|---|---|---|
| 0 | False | $0$ | 0 | 0 | 0 | 0 |
| 1 | $A \wedge B$ | $A \cdot B$ | 0 | 0 | 0 | 1 |
| 2 | $\neg(A \Rightarrow B)$ | $A - AB$ | 0 | 0 | 1 | 0 |
| 3 | $A$ | $A$ | 0 | 0 | 1 | 1 |
| 4 | $\neg(A \Leftarrow B)$ | $B - AB$ | 0 | 1 | 0 | 0 |
| 5 | $B$ | $B$ | 0 | 1 | 0 | 1 |
| 6 | $A \oplus B$ | $A + B - 2AB$ | 0 | 1 | 1 | 0 |
| 7 | $A \vee B$ | $A + B - AB$ | 0 | 1 | 1 | 1 |
| 8 | $\neg(A \vee B)$ | $1 - (A + B - AB)$ | 1 | 0 | 0 | 0 |
| 9 | $\neg(A \oplus B)$ | $1 - (A + B - 2AB)$ | 1 | 0 | 0 | 1 |
| 10 | $\neg B$ | $1 - B$ | 1 | 0 | 1 | 0 |
| 11 | $A \Leftarrow B$ | $1 - B + AB$ | 1 | 0 | 1 | 1 |
| 12 | $\neg A$ | $1 - A$ | 1 | 1 | 0 | 0 |
| 13 | $A \Rightarrow B$ | $1 - A + AB$ | 1 | 1 | 0 | 1 |
| 14 | $\neg(A \wedge B)$ | $1 - AB$ | 1 | 1 | 1 | 0 |
| 15 | True | $1$ | 1 | 1 | 1 | 1 |

Table A16. List of real-valued binary logic operations considered in this work [27].

| **Dataset** | **Class** | **Concepts** |
|---|---|---|
| CUB | Black-footed Albatross | back pattern: solid, under tail color: rufous, wing shape: long-wings, belly color: red, wing color: red, upperparts color: brown, breast pattern: multi-colored, upperparts color: rufous, bill shape: cone, tail shape: notched tail, back color: blue |
| | American Crow | back pattern: solid, wing shape: long-wings, upperparts color: brown, bill shape: cone, tail shape: notched tail, back color: blue, under tail color: grey, wing shape: tapered-wings, belly color: iridescent, wing color: iridescent |
| | Lazuli Bunting | back pattern: solid, under tail color: rufous, throat color: pink, wing shape: long-wings, wing color: red, upper tail color: pink, upperparts color: brown, breast pattern: multi-colored, bill shape: cone |
| AwA2 | Raccoon | black, white, gray, patches, spots, stripes, furry, small, pads, paws, tail, chewteeth, meatteeth, claws, walks, fast, quadrapedal, active, nocturnal, hibernate, agility |
| | Cow | black, white, brown, patches, spots, furry, toughskin, big, bulbous, hooves, tail, chewteeth, horns, smelly, walks, slow, strong, quadrapedal, active, inactive |
| | Dolphin | white, blue, gray, hairless, toughskin, big, lean, flippers, tail, chewteeth, swims, fast, strong, muscle, active, agility, fish, newworld, oldworld, coastal, ocean, water |
| CIFAR100 | Chair | furniture, a person, object, legs to support the seat, an office, a computer, a desk, four legs, a backrest, armrests on either side |
| | House | windows, building, object, structure, a yard, a chimney, a door, a wall, siding or brick exterior, a garage, roof |
| | Kangaroo | a grassland, short front legs, an animal, a safari, mammal, a long, powerful tail, brown or gray fur, marsupial, long, powerful hind legs, Australia |

Table A17. Some sample classes and a subset of their corresponding concepts for all the real-world datasets.

Vermilion Flycatcher ← 0.303 . (belly_color:orange ∧ leg_color:black) + -0.06 . (back color:blue) +
0.321 . (upper parts:olive ⟹ breast pattern:spotted) +
-0.295 . ¬ (belly pattern:striped) + ...

CUB

Anna Hummingbird ← 0.01 . (upper parts:olive ⟹ breast pattern:spotted) +
0.05 . (bill length:longer-than-head) +
-0.257 . (belly color:red) + ...

Blue Whale ← -0.266 . ¬ (weak ⊕ blue) + 0.391 . (tail) +
0.358 . ¬ (bipedal ∨ fierce) + 0.357 . (big ∧ fish) + ...

AWA2

Cow ← 0.122 . (white) + 0.206 . ¬ (agility) +
0.114 . (coastal ∧ tree) + 0.120 . ¬ (hops ∨ mountains) + ...

Figure A18. More examples of the class-level logic learned by logic-enhanced CBMs.

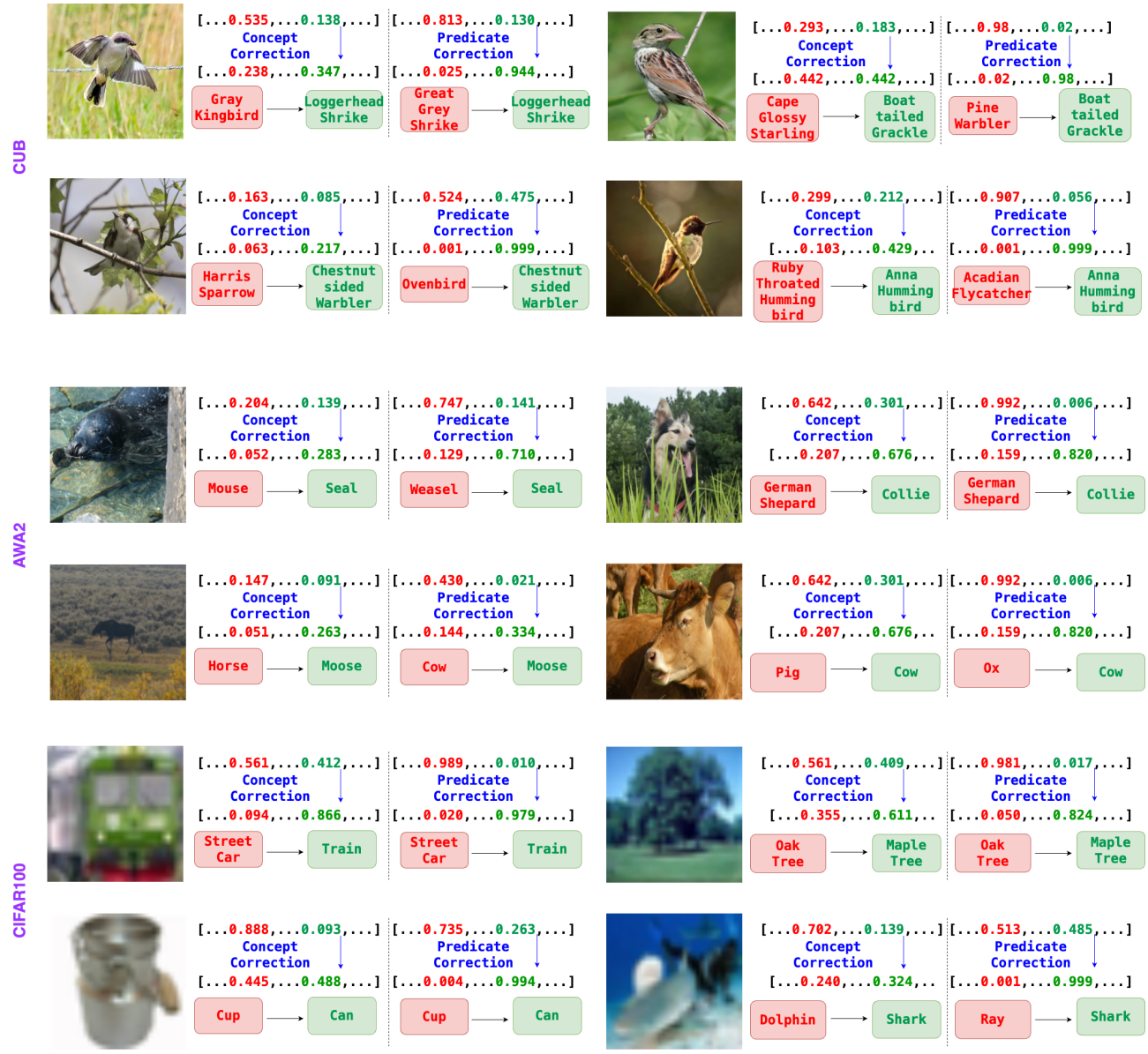


Figure A19. More examples of pre- and post-correction confidences on a CBM and a logic-enhanced CBM across three datasets.