# Enhancing Concept-based Learning with Logic

Deepika Vemuri [1] Gautham Bellamkonda [1] Vineeth N Balasubramanian [1]

## Abstract

Concept-based models promote learning in terms of high-level transferrable abstractions. These models offer one level more of transparency compared to a black box model, as the predictions are a weighted combination of concepts. The relations between concepts are a rich source of information that would compliment learning. We propose using the propositional logic derived from the concepts to model these relations and to address the expressivity-vs-interpretability tradeoff in these models. Three architectural variants that give rise to logic-enhanced models are introduced. We analyse several ways of training them and experimentally show that logic-enhanced concept-based models perform better than or on par with the base models, with the additional benefit of having better concept alignment and interpretability. These models allow for a richer formal expression of predictions, paving the way for logical reasoning with symbolic concepts.

## 1. Introduction

Humans learn concepts and their relations to each other, rather than learning each new object they encounter independently. Besides, sub-symbolic features like raw pixels are difficult to reason with. So, when a model is learning to classify, it is useful to learn the high level transferable concepts that make up the classes. Say the model has learnt that the class *elephant* is made up of the concepts {*huge, grey, mammal, etc*}. Now that the model has some understanding of what *huge* and *grey* mean, it could use these concepts to learn about an object like a *boulder*. Concept-based models perform classification using such concepts in a two-step process; they learn to predict the concepts from the inputs and then use the concepts to predict classes. The concept to class step is deliberately kept simple (it's usually just a linear layer) so that we can infer the importance of each concept present in a particular class. However, there are more complex interactions that these models fail to capture. One of which would be to model the concepts' relations to each other. Besides, with atomic [1] concepts we can only model so much and can only induce so much interpretability. Working with combinations of concepts increases the expressivity of the layer, enabling the layer to learn more complicated functions. Giving the model the ability to represent such relations could also help it learn knowledge at varying degrees of abstraction and give way to an architecture amenable to performing more complicated tasks like reasoning. So how can we give our model a relation learning capability while retaining, or better still, improving its interpretability?

Logic rules can be used to express such relations. They enable consistency and provide a framework to perform symbolic reasoning. Since concepts are, in a sense, "symbols"; i.e. the units of information we want to reason with, we need a mechanism to extract the logic that maps these symbols to classes. We show that this can be done by adding a learnable logic module that takes in concepts and outputs meaningful combinations of them (Figure 1).

Logic gates are interpretable non-linearities. This means we give the model more capacity with the added benefit of better interpretability. Each neuron, belonging to a layer in this module, learns a distribution over a number of possible logic gates for the two concepts that are selected to be input to it. We propose two concept pair selection mechanisms, to select the concepts pairs to input to the logic layer - (1) a large, sparse layer with fixed pairings or (2) a shorter, denser, fully connected layer where the pairings are learnt. Since we have access to concept ground truths, the logic module can either be incorporated into the existing architecture and be trained end-to-end or it can be trained as an independent component and then fine-tuned with any concept-based architecture.

**Contributions:**

1. We introduce a module to extract meaningful logical combinations, which we refer to as predicates, from atomic concepts.

2. We demonstrate that adding such a logic module on

[1] Indian Institute of Technology Hyderabad, India. Correspondence to: Deepika Vemuri <ai22resch11001@iith.ac.in>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1] Used in the *propositional logic* sense of the word

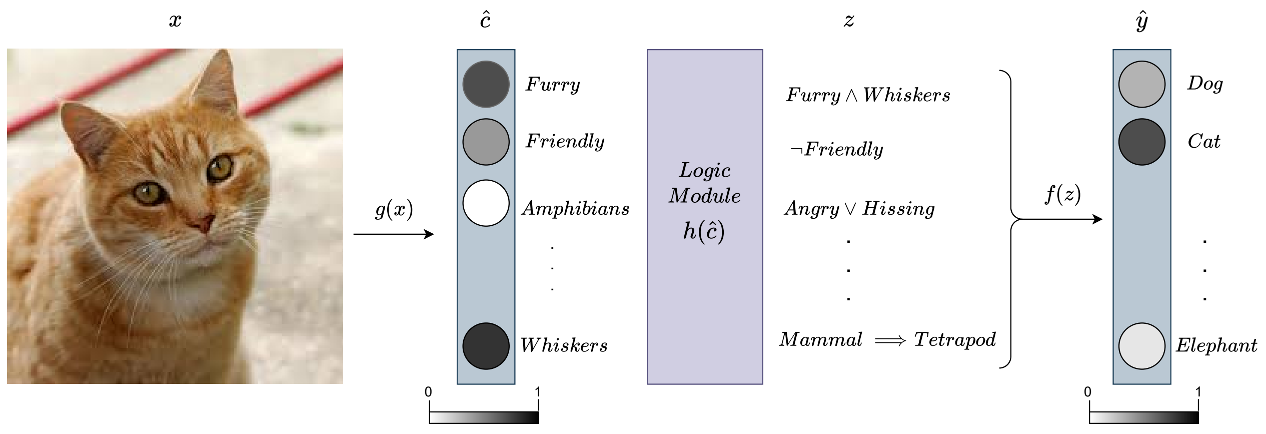


*Figure 1.* **A high-level overview of how our method works.** Concept-based models can be improved by giving them the ability to extract logic and form meaningful combinations of the concepts at their disposal. We introduce a logic module for this purpose. The scale under the layers, showing the degree of greyness in the neurons is used to indicate the strength of the logit in that respective layer. For ex. *Whiskers* is the strongest concept and *cat* is the strongest class.

top of several existing concept-based architectures consistently improves performance and propose metrics to quantify this.

3. We investigate the effect of logic layer size, number of connections and logic gates used on performance.

4. We propose a mechanism to perform better feature extraction using the logic learnt by our module.

## 2. Background: Concept-based Learning

A concept-based model (Koh et al., 2020) learns a mapping from $x \in X$ to $y \in Y$ via an intermediate concept encoder. Hence, it requires a dataset of three-tuples $\{X, C, Y\}$ where $X \in \mathbb{R}^h$, $C \in \mathbb{R}^k$, $Y \in \mathbb{R}^l$ and $h, k, l \in \mathbb{Z}$, correspond to the image, concept and label space respectively. Each prediction is of the form $\hat{y} = f(g(x))$ where, $g\colon \mathbb{R}^h \mapsto \mathbb{R}^k$ maps a sample from the image space to a set of concepts in the concept space (bird image $\rightarrow$ {white body, flat yellow bill, etc}) and $f\colon \mathbb{R}^k \mapsto \mathbb{R}^l$ maps a set of concepts in the concept space to a label in the label space ({white body, flat yellow bill, etc} $\rightarrow$ "Duck"). Concept-based models can be trained in several ways: independently, jointly and sequentially. We use jointly trained CBMs for all the CBM models in this work.

The bottleneck in these models is the availability of concept ground truths - which is expensive since we often require expert annotations. We alleviate this by following the recent approach (Oikarinen et al., 2023; Yang et al., 2023) of using an LLM to query for, filter and get class-level concept annotations. This lets us test our method on larger datasets with LLM obtained class-level concept annotations.

## 3. Learning Logic from Concepts

With the goal of going beyond the atomic concept strengths obtained from linear layer weights, we extract the propositional logic that maps concepts to classes. In order to do this, we add a learnable logic gate layer (Petersen et al., 2022) after the concept layer, where each neuron takes two concepts as inputs. This layer is parameterized by the choice of logic gate at each neuron. Now, each prediction becomes $\hat{y} = f(h(g(x)))$, where $h\colon \mathbb{R}^k \mapsto \mathbb{R}^t$, $t \in \{m, p\}$, based on the two architectural variants discussed subsequently.

Let there be $p$ logic gate neurons in the logic layer. Each logic gate neuron learns a distribution over $q$ possible logic gates. In order to make the logic gates learnable through gradient-based methods, they have to be made (1) differentiable and (2) continuous. To make them differentiable, we use the fuzzy logic alternative of each logic gate (Table 9). The obtained concept activations are passed through the logic gate neurons, where the $q$ fuzzy logic alternatives are computed for the $p$ neurons. To make them continuous, a weighted sum of all these operations is computed per neuron, which gives us a *relaxed* logic gate output. This is then passed to the next layer.

$$Z_k = \sum_{i=0}^{q} \frac{e^{\tilde{w}_i}}{\sum_j e^{\tilde{w}_j}} \cdot z_i(c_a, c_b)$$

where, $\tilde{w} \in \mathbb{R}^q$ and $c_a$ and $c_b$ are atomic concepts. The output of each neuron in this layer can be thought of as a relaxed logical predicate $Z_k$ $(0 \leq k < p)$.

But how do we decide which concepts to input to which logic gate neuron in the first place? We describe two concept selection mechanisms next:

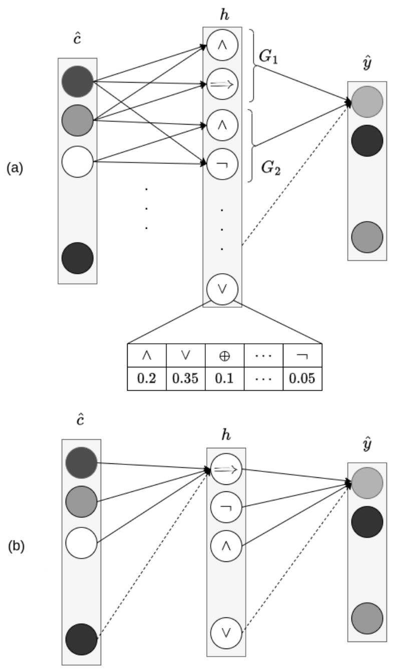


*Figure 2.* The concept selection mechanisms (a) The fixed pairings variant, where the logic layer is large, sparse and connections are random and fixed, (b) The learnt pairings variant, where the logic layer is short, dense and the connections are learnt

1. **Fixed pairings**: Here we follow the paradigm of a deep differentiable logic gate network (Petersen et al., 2022), where the connections to the logic layer are fixed, i.e. they don't have weights and the layer is large and sparse. The connections are pseudo randomly initialised such that for each logic gate neuron, two concepts $\{c_a, c_b\}$ are randomly chosen. Since the layer is large, we subsequently assign contiguous neurons into $m$ groups and then have a linear layer onto the classifier (Figure 2 (a)). So, the extra parameters that the model learns here are $W_{p\times q}$ and $W_{m\times l}$.

$$F_i = \sum_{j=1}^{m} w_{ij} G_j$$

$$\text{where, } G_j = \sum_{k=jp/m+1}^{(j+1)p/m} Z_k$$

2. **Learned pairings**: Although the logic layer with fixed pairings has a larger subspace, since the combinations are chosen pseudo randomly, not all of them might be relevant. Intuitively, we would like the number of predicates to be less than the number of concepts themselves. Since we would like to consider only the relevant combinations of concepts, we use a fully connected layer between the concept and logic gate layer and learn this relevance. As logic gates take binary inputs, we select the two highest weighted concepts to pass to the logic gate neuron they are connected to at each iteration. Since this layer is small and dense, we discard the grouping of the previous variant and simply have a linear layer onto the classifier (Figure 2(b)). So, the extra - parameters being learnt here are $W_{k\times p}$, $W_{p\times q}$ and $W_{p\times l}$.

$$F_i = \sum_{j=1}^{p} w_{ij} Z_j$$

Once the concept selection mechanism has been chosen, we pass the learnt predicates through a softmax to get the class logits.

$$\hat{y}_i = \frac{e^{F_i}}{\sum_{j=1}^{l} e^{F_j}}$$

where, $F_i$ denotes a logical formula.

Since we have access to concept ground truths, we use a binary cross-entropy loss ($\mathcal{L}_{BCE}$) to train the model to perform concept classification. After the predicates are obtained from these concepts, we use them to perform classification using a standard cross-entropy loss ($\mathcal{L}_{CE}$). So, to train the model, we use a weighted combination of these two losses.

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

where, $\alpha$ is the weighting hyperparameter. We would like to highlight that the addition of the logic module does not necessitate the usage of any extra loss, as it does not require any ground-truth logic.

**Using logic to learn better features:** Logic not only improves performance and alignment, but we find that it can also be used as feedback, to improve backbone features and hence concepts. Some kinds of logical relations like implications may not strictly help in improving performance, but rather help in *understanding* (e.g. mammal → dog). This could be thought of as a sort of *common sense*, which isn't task specific - which motivated a knowledge-based variant of our framework.

We use two classifier heads (Figure 3) - one taken from a CBM ($clf1$) and the other from the logic module ($clf2$). The backbone and concept part of the architecture essentially branch out into two parts - one that has the logic module and hence works with predicates, and the other that works with atomic concepts. The logic module is trained independently using the concept and class ground truths at its disposal and acts as the *knowledge* [2].

[2]Check appendix A for other knowledge-based works

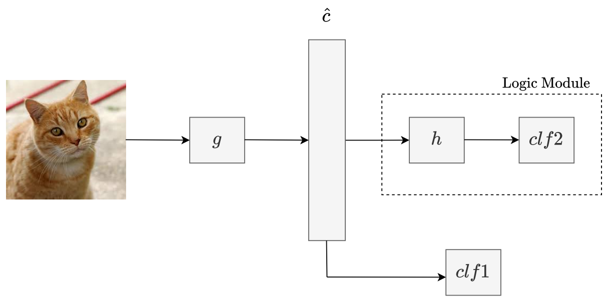


*Figure 3.* We introduce an auxiliary classifier to get a knowledge-based architecture that uses logic as feedback to get better features from the backbone

## 4. Experiments

We empirically study our method in the following two aspects:

- **Performance**: How does logic impact performance? Do different ways of training these models lead to different levels of generalization? Can logical relation understanding lead to better predictive performance?
- **Interpretability**: Do logic-enhanced models offer better interpretability? Are they more stable and reliable?

### 4.1. Experimental Setup

**Datasets and Baselines:** We perform analysis on two datasets: CUB200 and CIFAR100, to evaluate our method on both instance-level and class-level concept annotated datasets. The class-level concept annotation acquisition procedure is described in the appendix D.

We compare our approach with existing works that also work with concepts and show that adding a logic layer to them consistently improves performance. The term concept here is used in a general sense. It could be used to indicate a prototype, a text-based phrase or an LLM-generated description. The works we consider are the following: (1) BotCL (Wang et al., 2023) and (2) Label-Free CBMs (Oikarinen et al., 2023). BotCL proposes a self-supervised slot attention based (Locatello et al., 2020) concept discovery method, learning to represent an image solely through such concepts and Label-Free CBMs propose an LLM-based class-level concept annotation method which uses CLIP-Dissect (Oikarinen & Weng, 2023) for concept alignment.

### 4.2. Logic works well on instance-level concepts

We experiment with several ways of training based on the concept selection mechanism chosen. These include pre-training and fine-tuning different parts of the architecture, the details of which are provided in the appendix (B). A comparison among these methods is done, the best results of which are reported per mechanism in Table 1. We observe that our methods consistently perform on par or better than the base models on both datasets. We also see that the confidence[3] of the model significantly increases on CUB when a logic layer is added (shown for CBMs in Table 2).

| | Variants | CUB | CIFAR100 |
|---|---|---|---|
| CBM | Standard | 75.37 | 83.33 |
| | Fixed | **80.13 (+4.76)** | **84.13 (+0.80)** |
| | Learned | **77.12 (+1.75)** | **84.55 (+1.22)** |
| LF CBM | Standard | 72.94 | 65.29 |
| | Fixed | 70.55 | 65.09 |
| | Learned | 72.11 | **65.82 (+0.53)** |
| BotCL | Standard | 58.29 | 74.51 |
| | Fixed | 49.5 | 72.69 |
| | Learned | **58.59 (+0.3)** | 74 |

*Table 1.* Accuracy comparison on CUB and CIFAR100. *Standard* indicates the vanilla base model, *Fixed* and *Learnt* are the two logic-based architectural variants

| Variant | CUB | | CIFAR100 | |
|---|---|---|---|---|
| | ✓ | × | ✓ | × |
| CBM | 92.12 | 42.15 | 94.64 | 57.07 |
| Fixed | **96.61** | **26.65** | **99.97** | 84.32 |
| Learned | **96.83** | **34.47** | **99.99** | 80.41 |

*Table 2.* Confidence improvement. ✓ means right predictions (higher is better), and × means wrong predictions (lower is better)

**Logic as Knowledge:** The motivation here is to use logic as knowledge for better feature learning. We follow the architecture of Figure 3, where the logic module of the architecture is frozen, while the rest of the network is finetuned using this knowledge. We see that this way of learning the backbone, does better than a standard CBM. Additionally, we try training the whole network end-to-end to assess how the auxiliary classifier could help improve performance. These methods are compared with a standard CBM in Table 3.

| Model | CUB | CIFAR100 |
|---|---|---|
| CBM | 75.37 | 83.33 |
| KBV1 | 80.21 | 75.5 |
| KBV2 | 81.97 | 85.32 |

*Table 3.* Accuracy comparison with knowledge-based variants. KBV1 uses frozen logic, while KBV2 has a trainable one

### 4.3. Logic leads to better concept alignment

We would like the concept activations of inputs of the same class to be similar to each other. We measure this using concept-alignment by computing the average degree of simi-

[3]See Appendix C

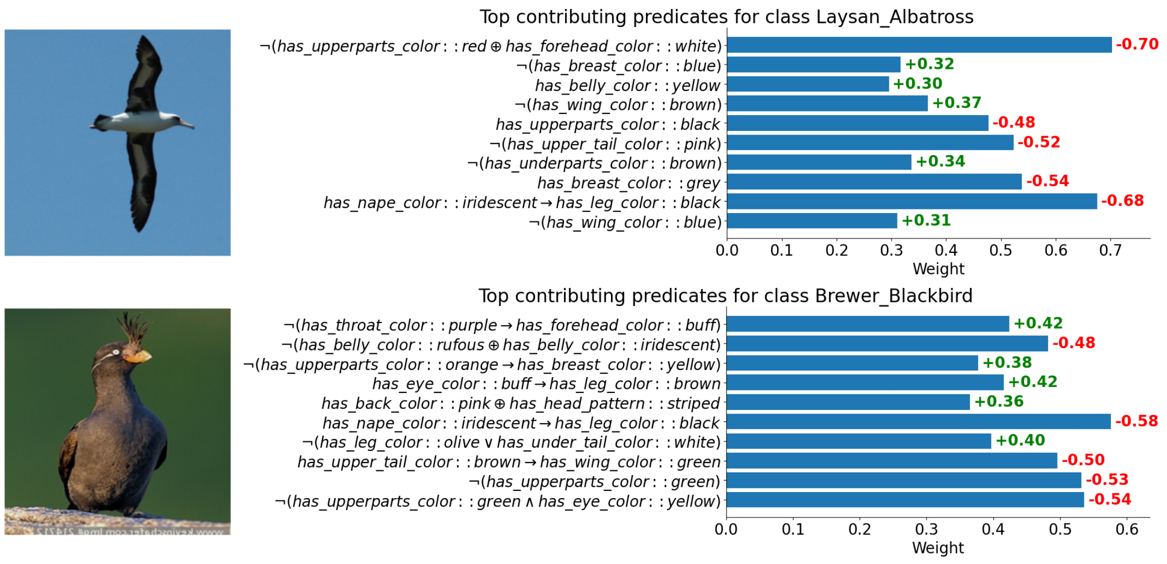


*Figure 4.* Top contributing predicates for 2 different classes on the CUB dataset - **green** indicates a positive contribution, while **red** indicates a negative contribution

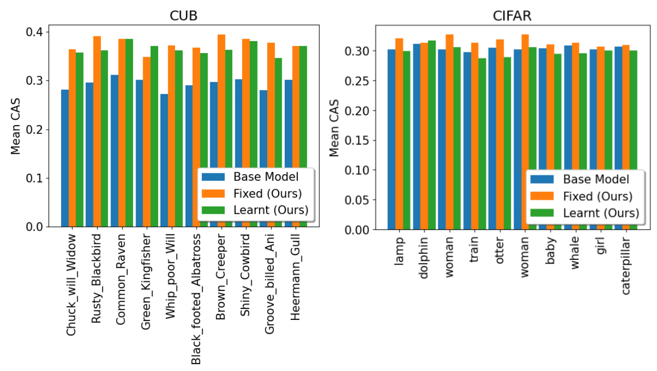


*Figure 5.* Concept alignment scores on 10 random classes (x-axis) of CUB and CIFAR100

larity in concept activations of images belonging to the same class. Let $n_i$ denote the number of samples belonging to class $i$ and $g$ indicate the concept encoder of the model we are measuring concept-alignment for.

$$CA(i) = \sum_{j_1=1}^{n_i} \sum_{j_2=j_1+1}^{n_i} g(x_{j_1}) \cdot g(x_{j_2})$$

We compute this score on a standard CBM and the two logic-based variants , on CUB200 and CIFAR100. The logic variants consistently help better align concepts on CUB200. Although on CIFAR100, the alignment scores seem to be slightly noisier, similar to the change in confidence we see. We show the alignment scores on both datasets for a subset of classes in Figure 5 and across classes in Table 4.

| DATASET | STANDARD | FIXED | LEARNED |
|---|---|---|---|
| CUB | 0.30 | **0.37** | **0.35** |
| CIFAR100 | 0.30 | **0.31** | **0.30** |

*Table 4.* Mean concept alignment score across all the classes

### 4.4. Logic leads to better interpretability

Some examples of the logical predicates learnt by our framework are shown in Figure 4. Clearly, getting predictions in terms of predicates is much more expressive than atomic concepts. We do further analysis on the logic gates in the appendix E.

## 5. Conclusion

In this work, we presented how concept-based models can be enhanced with logic, showing that these models perform better or on par with the base models while offering better concept alignment (logic leads to better concept learning), interpretability (the predictions are in terms of propositional logic predicates which are much more expressive) and flexibility (the logic module can be trained as an independent component). We described three variants of the logic module and analysed them.

Overall, we see that logic works best with instance-level concept annotations. The logic module seems to be slightly noisier on class-level annotations, although it might require some further analysis to confirm this behaviour. While there is still room for improvement on the goodness of the logic learnt by each class, we see this as an initial effort under the broader goal of improving a model's predictive performance by giving it the ability to logically reason.

## References

Barbiero, P., Ciravegna, G., Giannini, F., Zarlenga, M. E., Magister, L. C., Tonda, A. P., Lio', P., Precioso, F., Jamnik, M., and Marra, G. Interpretable neural-symbolic concept reasoning. *ArXiv*, abs/2304.14068, 2023. URL https://api.semanticscholar.org/CorpusID:258352760.

Ciravegna, G., Barbiero, P., Giannini, F., Gori, M., Liò, P., Maggini, M., and Melacci, S. Logic explained networks. *Artificial Intelligence*, 314:103822, January 2023. ISSN 0004-3702. doi: 10.1016/j.artint.2022.103822. URL http://dx.doi.org/10.1016/j.artint.2022.103822.

Huang, Y., Tang, J., Chen, Z., Zhang, R., Zhang, X., Chen, W., Zhao, Z., Zhao, Z., Lv, T., Hu, Z., and Zhang, W. Structure-clip: Towards scene graph knowledge to enhance multi-modal structured representations, 2023.

Kim, E., Jung, D., Park, S., Kim, S., and Yoon, S.-H. Probabilistic concept bottleneck models. *ArXiv*, abs/2306.01574, 2023. URL https://api.semanticscholar.org/CorpusID:259063823.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. Concept bottleneck models. pp. 5338–5348. PMLR, 2020.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

Lee, S., Wang, X., Han, S., Yi, X., Xie, X., and Cha, M. Self-explaining deep models with logic rule reasoning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=8SY8ete3zu.

Li, L., Wang, W., and Yang, Y. Logicseg: Parsing visual semantics with neural logic learning and reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4122–4133, October 2023.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 11525–11538. Curran Associates, Inc., 2020.

Marconato, E., Passerini, A., and Teso, S. Glancenets: Interpretabile, leak-proof concept-based models. In *Neural Information Processing Systems*, 2022. URL https://api.semanticscholar.org/CorpusID:249209924.

Margeloiu, A., Ashman, M., Bhatt, U., Chen, Y., Jamnik, M., and Weller, A. Do concept bottleneck models learn as intended? *ArXiv*, abs/2105.04289, 2021. URL https://api.semanticscholar.org/CorpusID:234339919.

Oikarinen, T. and Weng, T.-W. CLIP-dissect: Automatic description of neuron representations in deep vision networks. In *The Eleventh International Conference on Learning Representations*, 2023.

Oikarinen, T., Das, S., Nguyen, L. M., and Weng, T.-W. Label-free concept bottleneck models. 2023.

Petersen, F., Borgelt, C., Kuehne, H., and Deussen, O. Deep differentiable logic gate networks. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Rajaby Faghihi, H., Nafar, A., Zheng, C., Mirzaee, R., Zhang, Y., Uszok, A., Wan, A., Premsri, T., Roth, D., and Kordjamshidi, P. Gluecons: A generic benchmark for learning under constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37 (8):9552–9561, Jun. 2023. doi: 10.1609/aaai.v37i8.26143. URL https://ojs.aaai.org/index.php/AAAI/article/view/26143.

Stein, A., Naik, A., Wu, Y., Naik, M., and Wong, E. Towards compositionality in concept learning. URL https://api.semanticscholar.org/CorpusID:268932793.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL http://arxiv.org/abs/1512.00567.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. J. The caltech-ucsd birds-200-2011 dataset. 2011. URL https://api.semanticscholar.org/CorpusID:16119123.

Wang, B., Li, L., Nakashima, Y., and Nagahara, H. Learning bottleneck concepts in image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Yang, Y., Panagopoulou, A., Zhou, S., Jin, D., Callison-Burch, C., and Yatskar, M. 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*, pp. 19187–19197, 2023.

Zarlenga, M. E., Barbiero, P., Ciravegna, G., Marra, G., Giannini, F., Diligenti, M., Shams, Z., Precioso, F., Melacci, S., Weller, A., Lio', P., and Jamnik, M. Concept embedding models. *ArXiv*, abs/2209.09056, 2022. URL https://api.semanticscholar.org/CorpusID:252367901.

## Appendix

In this part of the paper, we provide additional details of our work. Related work is discussed in Section A. Section B provides details about the architecture and implementation of our models. Section D gives details about the datasets used in our experiments. Finally, we conclude with an analysis of the logic layer in Section E.

## A. Related Work

**Concept-based Models:** This family of models introduced the idea of predicting labels in terms of concepts. However, owing to the criticism of whether these models actually learn as intended (Margeloiu et al., 2021) lead to several works that address various limitations in their concept learning process. These include addressing concept leakage (Marconato et al., 2022), uncertainty quantification (Kim et al., 2023) and extracting concepts that are more amenable to composition (Stein et al.). Our work, on the other hand, aims to extract combinations of concepts and use them as the basis for classification.

**Logic-based Interpretability:** Several works attempt to derive logic from a model. Logic-explained networks (Ciravegna et al., 2023) introduce a family of such models that have a mapping between an input concept space and an output concept space, the SELOR (Lee et al., 2022) framework gives a probabilistic formulation for logic rule generation that work with interpretable features and the DCR (Barbiero et al., 2023) model that builds upon concept embeddings (Zarlenga et al., 2022) by giving explanations in terms of logical rules. However, all of these works use logic as a means of *explanation*, whereas we use logic gates with the aim of getting meaningful concept combinations and incorporate it as a learnable module.

**Knowledge-augmented methods:** These methods use knowledge-base to augment their learning. Some works use a scene-graph (Huang et al., 2023) as external knowledge. Logic rules are also often used to represent knowledge - as a way of constraining predictions. GLUECons (Rajaby Faghihi et al., 2023) present a benchmark for constraint integration with deep neural networks. LogicSeg (Li et al., 2023) does this for the task of semantic segmentation, where the relations derived from a class-hierarchy are enforced in the form of logic rules. To the best of our knowledge, we are the first to learn and represent knowledge as a part of a network and use it to improve the feature learning process.

## B. Architecture and Implementation Details

The feature extractor $g$ used in all the experiments was an Inception V3 (Szegedy et al., 2015) and the concept and classifiers are single linear layers. We extend the difflogic [4] library for implementing the logic layer variants.
*Hyperparameter details:* For both CUB200 and CIFAR100, we train the models for 40 epochs, with a learning rate of 0.001, a weight decay of 0.0004 and a batch size of 64. We take $m = 2l$ wherever applicable.

*Training details:* We experiment with several ways of training the 2 variants: (1) *Naive* - where the whole backbone + concept + logic module + classifier - architecture is trained end-to-end. We try this method with the fixed pairings variant and henceforth refer to it as LCBM1. (2) *Pretrained CBM* - where the model is trained in a two-step fashion; the CBM is trained separately using the concept and class ground truths and the logic layer is then learnt using the backbone and concept parameters loaded from the trained CBM. We try this method only on learned pairings and refer to it as LCBM3 henceforth. (3) *Finetuning both components* - also a two-step training process, where the CBM is trained separately, and the logic module is trained separately using the concept and class ground truths, after which both components are finetuned together. We try this method with both variants henceforth referred to as LCBM2 (fixed pairings) and LCBM4 (learned pairings).

| Training Method | bb+c (pretrained) | logic (pretrained) | bb+c (finetune) | logic (finetune) |
|---|---|---|---|---|
| LCBM1 | | | | |
| LCBM3 | ✓ | | ✓ | |
| LCBM2,4 | ✓ | ✓ | ✓ | ✓ |

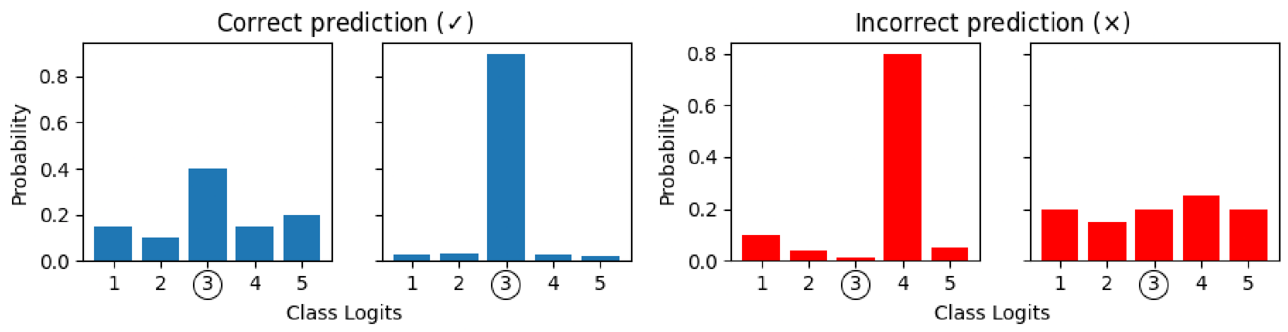
*Table 5.* Training schemes experimented with. bb+c - (backbone + concept) part of the architecture

[4] https://github.com/Felix-Petersen/difflogic

## C. Confidence Improvement

Using the logic module allows us to improve the confidence of the model. We calculate the confidence of a prediction as the maximum of the softmax probabilities. Confidence of a model is then defined as the average of the confidence of all the predictions. Ideally, we would want the model to be confident when it makes a correct prediction (✓) and less confident when it makes an incorrect prediction ($\times$), as shown in Figure 6

$$\text{Confidence} = \frac{1}{n}\sum_{i=1}^{n} \max(\text{logits}(x_i))$$



(a) Confidence should increase for correct predictions (b) Confidence should decrease for incorrect predictions

*Figure 6.* Visualizations describing expected confidence change

## D. Dataset Details

- **CUB**: The Caltech-UCSD Birds-200-2011 (CUB) dataset (Wah et al., 2011) 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, size, wing color, etc.
- **CIFAR100**: CIFAR100 (Canadian Institute for Advanced Research) is a subset of the Tiny Images dataset (Krizhevsky, 2009) comprising 100 classes. It consists of 60000 images, 50000 for training and 10000 for testing. Since this dataset does not have manually annotated concepts, we query an LLM to get 925 concepts (Oikarinen et al., 2023).

Some example concepts from both datasets are given in Table 7.

## E. Analysis of the Logic Layer

### E.1. Effect of logic layer size

Since the fixed pairings variant works with a wide logic layer, we study the effect of the size of the logic layer on the performance of the model. We vary the size of the logic layer from 4000 logic gate neurons to 16000 logic gate neurons and observe that the performance of the model increases with the size of the logic layer. These results on LCBM1 and LCBM2 for both CUB and CIFAR100 are shown in Figure 7. We do a study on increasing the layer size for the learned pairings model as well. For this variant, the logic layer size was chosen such that $\#concepts \leq logic\ layer\ size \leq \#classes$. These results are reported in Table 8.

### E.2. Effect of number of logic gate types used

As shown in Figure 9, the distribution of logic gates learnt is quite spread out. We analyse this behaviour by reducing the number of logic gate types used and checking whether the model is able to learn good logic with a reduced logical operation set. We observe that the number of logic gate types used have a marked impact on performance. The number of logic gates

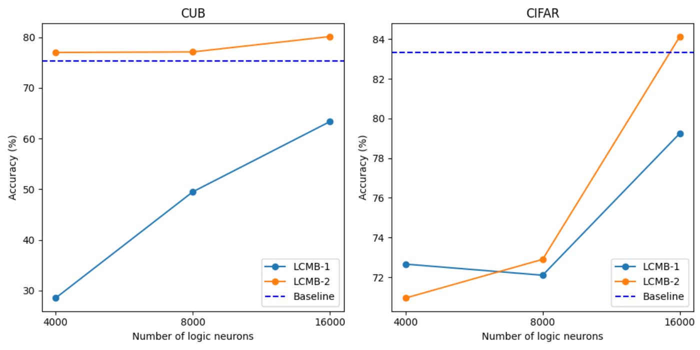


*Figure 7.* Effect of logic layer size on performance

| Model | CUB | | CIFAR100 | |
|---|---|---|---|---|
| | 200 | 250 | 500 | 600 |
| LCBM3 | 74.87 | 77.12 | 84.55 | 83.94 |
| LCBM4 | 73.46 | 76.96 | 80.98 | 73.27 |

*Figure 8.* Effect of number of neurons in feedback models

used in all the experiments in the main results are 16. These gates along with their fuzzy logic relaxations are shown in Table 9. We run an ablation by reducing the number of logic gates to 8. These consist of only the following operations - $\wedge, \vee, 1, 0, c_1, c_2, \neg c_1, \neg c_2$. As evidenced from Table 6, fine-grained datasets seem to require more expressive logic gates like $implication$ and $xor$, as there is a significant performance drop without them; whereas class-level annotated datasets seem to perform well even without such gates, indicating that they do not need very complicated logic.

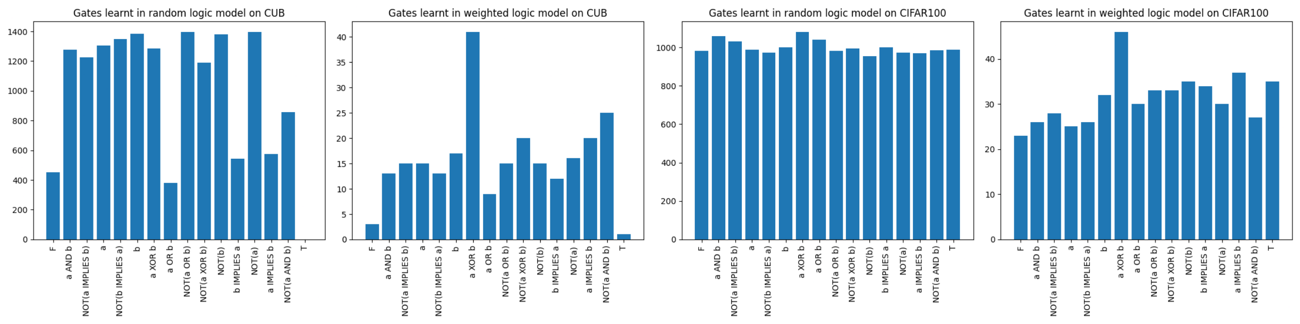


*Figure 9.* Distribution of gates learnt by the models

| Dataset | Gates | 16k | 8k | 4k |
|---|---|---|---|---|
| CUB | 8 | 18.44 | 9.26 | 3.58 |
| | 16 | 63.35 | 49.49 | 28.54 |
| CIFAR100 | 8 | 81.54 | 79.32 | 79.71 |
| | 16 | 84.13 | 72.9 | 70.95 |

*Table 6.* Effect of number of gates on performance

### E.3. Feedback as a way of incorporating more logic layers:

We find that incorporating more than one logic layer hurts the classifier's accuracy in general, on both the variants. Using multi-layered logic in our knowledge-based variant gives us a means to use more complicated relational logic to improve the base classifier. We compare performance with and without the feedback loop and observe that the accuracy of the base model CBM significantly increases when the feedback loop is added. Our results are shown in Table 8 along with additional results in Table 10.

| **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 |
| CIFAR100 | Bicycle | a tire, object, a helmet, a handlebar, a bicycle seat, pedals attached to the frame, mode of transportation, two wheels of equal size, a seat affixed to the frame, a chain |
| | Chair | furniture, a person, object, legs to support the seat, an office, a computer, a desk, four legs, a backrest, armrests on either side |
| | 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 7.* Some sample classes and a subset of their corresponding concepts for both the datasets

| Dataset | CBM | 4k x 2 | 2k x 3 | 4k x 2 with Feedback | 2k x 3 with Feedback |
|---|---|---|---|---|---|
| CUB | 75.37 | 2.92 | 3.12 | **79.54** | **80.78** |
| CIFAR100 | 83.33 | 65.91 | 1.6 | **85.05** | **83.47** |

*Table 8.* Effect of feedback loop on performance

| 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 9.* List of all real-valued binary logic ops.

| | CBM | 16, 16k x 1 | 16, 8k x 1 | 16, 4k x 1 | 16, 4k x 2 | 16, 2k x 3 |
|---|---|---|---|---|---|---|
| CUB | 75.37 | 80.38 | 79.04 | 79.54 | 79.54 | 78.96 |
| CIFAR100 | 83.33 | 85.12 | 85.37 | 84.23 | 85.05 | 83.47 |

*Table 10.* Effect of number of gates and number of layers on performance using the knowledge-based variant

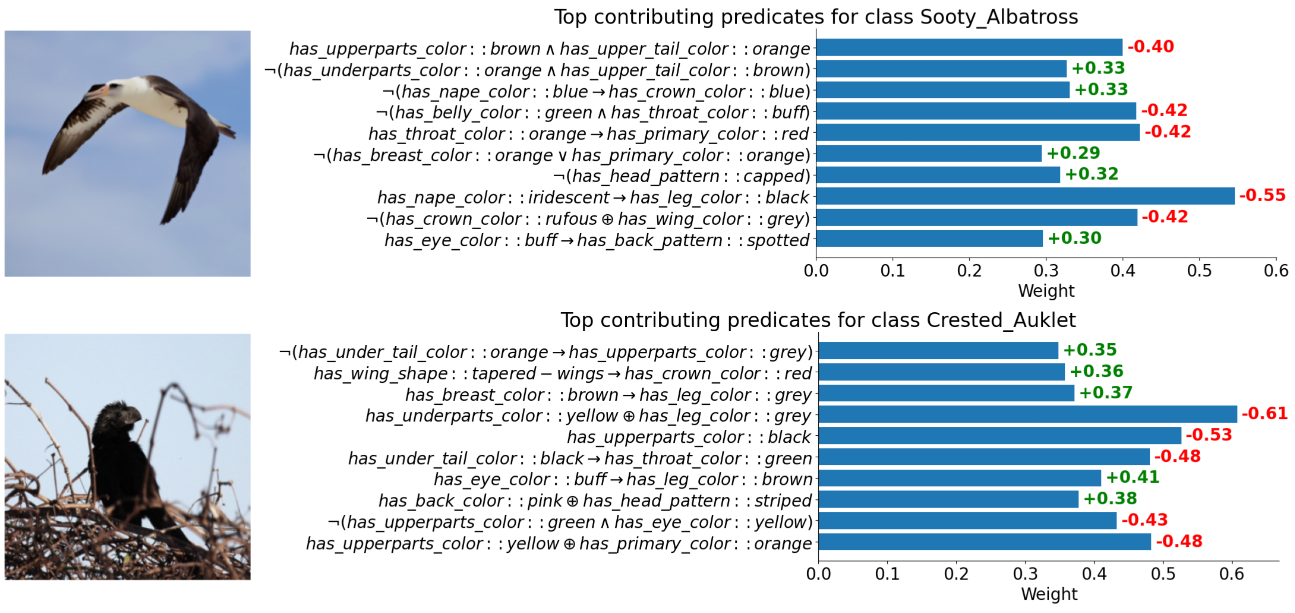


*Figure 10.* Logical predicates on 2 more classes