LABEL-FREE CONCEPT BOTTLENECK MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Concept bottleneck model (CBM) are a popular way of creating more interpretable neural networks by having hidden layer neurons correspond to human-understandable concepts. However, existing CBMs and their variants have two crucial limitations: first, the need to collect labeled data for each of the predefined concepts, which is time consuming and labor intensive; second, the accuracy of a CBM is often significantly lower than that of a standard neural network, especially on more complex datasets. This poor performance creates a barrier for adoption in practical real world applications. Motivated by these challenges, we propose *Label-free* CBM which is a framework to transform any neural network into an interpretable CBM without labeled concept data, while retaining a high accuracy. Our Label-free CBM has many advantages, it is: *scalable* - we present the first CBM scaled to ImageNet, *efficient* - creating a CBM takes only a few hours even for very large datasets, and *automated* - training it for a new dataset requires minimal human effort.

1 INTRODUCTION

Deep neural networks (DNNs) have demonstrated unprecedented success in a wide range of machine learning tasks such as computer vision, natural language processing, and speech recognition. However, due to their complex and deep structures, they are often regarded as *black-box* models that are difficult to understand and interpret their behaviors. Interpretable models are important for many reasons such as creating calibrated trust in our models, i.e. understanding when we should trust the models, and making deep learning models interpretable and explainable is an active yet challenging research topic.

One approach to make deep learning more interpretable is Concept Bottleneck Models (CBMs) (Koh et al., 2020). CBMs typically have a concept bottleneck layer before the (last) fully connected layer of the neural network. The concept bottleneck layer is trained to have each neuron correspond to a single human understandable *concept*. This makes the final decision a linear function of interpretable concepts, greatly increasing our understanding of the decision making. Importantly, CBMs have been shown to be useful in a variety of applications, including model debugging and timely

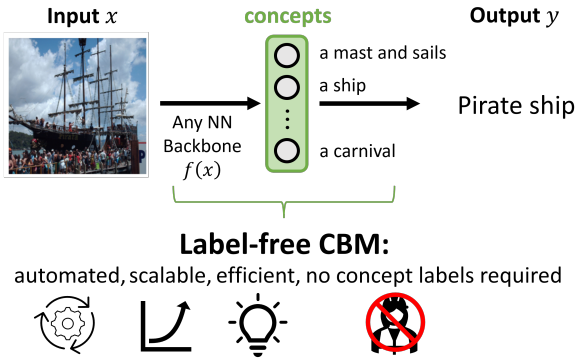

Figure 1: Our proposed **Label-free CBM** has many desired features of where existing CBMs lack and can transform any neural network backbone into an interpretable concept bottleneck model.

intervention on the DNNs. However, there are two crucial limitations of existing CBMs and their variants (Koh et al., 2020; Yuksekgonul et al., 2022; Zhou et al., 2018): (i) the labeled data is required for each of the predefined concept, which is time consuming and expensive to collect; (ii) the accuracy of a CBM is often significantly lower than the accuracy of a standard neural network, especially on more complex datasets.

To address the above two main challenges, we propose a new framework named **Label-free** CBM, which is capable of transforming any neural network into an interpretable CBM *without* labeled concept data while *preserving* comparable accuracy of the original neural network model by leveraging foundation models (Bommasani et al., 2021). Our Label-free CBM has many advantages:

- it is *scalable* - to our best knowledge, it is the first CBM that scales to ImageNet
- it is *efficient* - creating a CBM takes only a few hours even for very large datasets
- it is *automated* - training it for a new dataset requires minimal human effort

2 Related work

Post-hoc explanations (Samek et al., 2021). The approach of post-hoc explanations includes some classic methods such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg & Lee, 2017), which try to explain individual model decisions by identifying which parts of the input data (e.g. pixels) are the most important for a given decision. However, these methods are based on local approximations of the DNN model and as such are not always accurate. Further, the explanations at the granularity of input pixels may not always be helpful and could require substantial subjective analysis from human. In contrast, our explanations in Section 4.4 are correct(not approximated) and explain predictions in terms of human-understandable concepts.

More interpretable final layer: (Wong et al., 2021) proposes making the FC layer sparse, and develop an efficient algorithm for doing so. They show that sparse models are more interpretable in many ways, but it still suffers from the fact the previous layer features are not interpretable. NBDT Wan et al. (2020) propose replacing the final layer with a neural backed decision tree for another form of more interpretable decisions. Other approaches to make NNs more interpretable include Concept Whitening Chen et al. (2020) and Concept Embedding Models Zarlenga et al. (2022).

CBM: Most related to our approach are Concept Bottleneck Models (Koh et al., 2020; Losch et al., 2019) which means creating a layer before the last fully connected layer where each neuron corresponds to a human interpretable concept. CBMs have been shown to be beneficial by allowing for human test-time intervention for improved accuracy, as well as being easier to debug. To reduce the training cost of a CBM in (Koh et al., 2020), a recent work (Yuksekgonul et al., 2022) proposed a post-hoc CBM that only need to train the last FC layer along with an optional residual fitting layer without the need to train the backbone from scratch by leveraging Test Concept Activation Vectors (TCAV) (Kim et al., 2018) or the multi-modal CLIP model (Radford et al., 2021). However, the post-hoc CBM does not fully address the problems of the original CBM as using TCAV still requires collecting annotated/labelled concept data and their use of CLIP model can only be applied to the NN backbone having the same architecture of CLIP image encoder which limits the application of post-hoc CBM. Besides, the performance of post-hoc CBMs without residual fitting layers is often significantly lower than the standard DNNs; although adding the residual layer can help to improve the accuracy, the residual layer is not interpretable and is thus undesired. Similarly, an earlier work Interpretable Basis Decomposition Zhou et al. (2018) proposes learning a concept bottleneck layer based on labeled concept data for explanable decisions, even though they do not call themselves a CBM. Extensive comparison of our method and existing approaches is shown in Table 1.

Model editing/debugging: Our approach is related to a range of works proposing ways to edit networks, such as (Bau et al., 2020; Wang et al., 2022) for generative vision models, (Bau et al., 2020) for classifiers, or language models (Meng et al., 2022; Mitchell et al., 2021). In addition (Abid et al., 2021) propose a way to debug model mistakes using TCAV activation vectors.

CLIP-Dissect: (Oikarinen & Weng, 2022). CLIP-Dissect is a recent method for understanding the roles of hidden layer neurons by leraging the CLIP multimodal model (Radford et al., 2021). It can provide a score of how close any neuron is to representing any given concept, which makes it useful as an optimization target for learning our interpretable projection in Step 3.

	Flexibility		Interpretability		Performance	
Method:	Without labeled concept data	Any network architecture	Sparse final layer	All interpretable features	Preserves accuracy	Extends to ImageNet scale
CBM	No	**Yes**	No	**Yes**	No	No
IBD	No	**Yes**	No	No	**Yes**	No
P-CBM	No	**Yes**	Yes	**Yes**	No	No
P-CBM (CLIP)	**Yes**	No	**Yes**	**Yes**	No	No*
P-CBM-h	No	**Yes**	**Yes**	No	**Yes**	No
P-CBM-h (CLIP)	**Yes**	No	**Yes**	No	**Yes**	No*
Label-free CBM (This work)	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

Table 1: Comparison of our method against existing methods for creating Concept Bottleneck models, CBM (Koh et al., 2020), IBD (Zhou et al., 2018) and 4 versions of P-CBM (Yuksekgonul et al., 2022), where '-h' indicates the hybrid model that uses uninterpretable residual term, '(CLIP)' means models using CLIP concepts and '*' indicates that their method might be able to scale if using our proposed GPT-3 procedure for generating concept set.

3 Label-free CBM: A new framework to build CBM

In this section, we propose **Label-free CBM**, a novel framework that builds a concept bottleneck model (CBM) in an automated, scalable and efficient fashion, which addresses the core limitations of existing CBMs (Koh et al., 2020; Yuksekgonul et al., 2022). Given a neural network backbone, **Label-free CBM** transforms the backbone into an interpretable CBM without the need of concept labels in the following 4 key steps as illustrated in Fig 2: (Step 1) Create the initial concept set and filter undesired concepts; (Step 2) Compute embeddings from the backbone and the concept matrix on the training dataset; (Step 3) Learn W_c to create a Concept Bottleneck Layer; (Step 4) Learn the weights W_F of the sparse final FC layer to make predictions.

Note that the backbone model here can either be a model trained on our target task, or some other task. In the following, we provide the details of **Label-free CBM** for Step 1 in Sec 3.1, Step 2 and 3 in Sec 3.2, and finally Step 4 in Sec 3.3 respectively.

3.1 Step 1: Concept set creation and filtering

In this step, we will describe how to create a concept set to serve as the basis of human-interpretable concepts in the concept bottleneck layer that will be constructed in the Step 3. This step consists of two sub-steps: **A. Initial concept set creation** and **B. Concept set filtering** below.

A. Initial concept set creation: To be specific, a concept set refers to the set of concepts represented in the Concept Bottleneck layer. In the original CBM paper (Koh et al., 2020), this is decided by

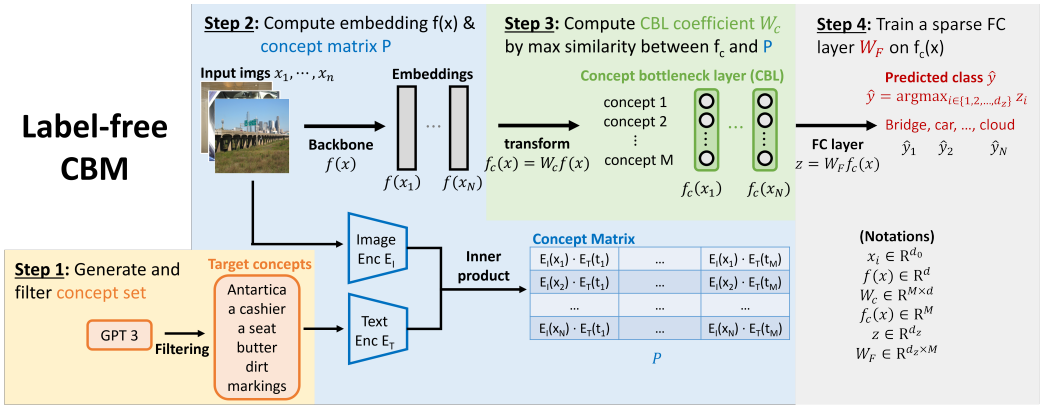

Figure 2: Overview of our pipeline for creating label-free CBM.

domain experts as the set of concepts that are important for the given task. However, since our objective is to automate the entire process of generating concept bottleneck models, it is not desirable to rely on having human experts. Instead, we propose generating the concept set via GPT-3 (Brown et al., 2020) using the OpenAI API. Somewhat surprisingly, GPT-3 has a good amount of domain knowledge of which concepts are important for detecting each class when prompted in the right way. Specifically, we ask GPT-3 the following:

- List the most important features for recognizing something as a {class}:
- List the things most commonly seen around a {class}
- Give superclasses for the word {class}:

Note that the {class} in here refers to the class name in the machine learning tasks. For GPT-3 to perform well on the above prompt, we provide two examples of the desired outputs for few-shot adaptation. Note that those two examples can be shared across all datasets, so no additional user input is needed for generating concept set for a new dataset. Full prompts and example outputs are illustrated in the Appendix Figures 7 and 8. To reduce variance we run each prompt twice and combine the results. This gives us a large, somewhat noisy set of initial concepts, which we improve by filtering in **B. Concept set filtering**. Note that Posthoc-CBM (Yuksekgonul et al., 2022) proposes using a knowledge graph known as ConceptNet (Speer & Havasi, 2013) to generate initial concept set, but it doesn't work for class names that consist of multiple words (i.e. ImageNet classes such as *leatherback sea turtle*), and in general we found GPT-3 concepts to be of higher quality. See Appendix B.2 for comparison against ConceptNet generated concepts.

B. Concept set filtering: Several filters are employed to improve the quality and reduce the size of our concept set, as stated below:

1. *Concept length*: We delete any concept longer than 30 characters in length, to keep concepts simple and avoid unnecessary complication.
2. *Remove concepts too similar to classes*: We don't want our CBM to contain the output classes themselves as that would defeat the purpose of the explanation. To avoid this we remove all concepts that are too similar to the names of target classes. We measure this with cosine similarity in a text embedding space. In particular we use an ensemble of similarities of the CLIP ViT-B/16 text encoder (Radford et al., 2021) (denoted as CLIP cos) as well as the all-mpnet-base-v2 sentence encoder, so our measure is a combination of visual and textual similarity. For all datasts we deleted concepts with similarity > 0.85 to any target class.
3. *Remove concepts too similar to each other*: Similar to above, we don't want duplicate or too similar concepts in the bottleneck layer. We use the same embedding space as above, and remove any concept that has another concept with > 0.9 cosine similarity already in concept set.
4. *Remove concepts not present in training data*: To make sure our concept layer accurately presents its target concepts, we remove any concept that doesn't activate CLIP highly. Note that this cut-off is a dataset specific metric and we delete all concepts with average top-5 activation less than the cut-off.
5. *Remove concepts we can't project accurately*: This step is actually performed after step 3 and is described in section 3.2.

We discuss the reasoning behind the filters and their effects on our results in detail in Appendix B.1.

3.2 Step 2 and 3: Learning the Concept Bottleneck Layer (CBL)

Once the concept set is obtained, the next step is to learn a projection from the backbone model's feature space into a space where axis directions correspond to interpretable concepts. Here, we present a way of learning the projection weights W_c without any labeled concept data using CLIP-Dissect (Oikarinen & Weng, 2022). To start, we need the set of target concepts that the bottleneck layer is expected to represent as $\mathcal{C} = \{t_1, ..., t_M\}$, as well as the training dataset (e.g. images) $\mathcal{D} = \{x_1, ..., x_N\}$ of the original task, where $x_i \in R^{d_0}$. Next we calculate and save the CLIP concept activation matrix P where $P_{i,j} = E_I(x_i) \cdot E_T(t_j)$ and E_I and E_T are the CLIP image and text

encoders respectively. W_c is initialized as a random $M \times d_0$ matrix where d_0 is the dimensionality of backbone features $f(x)$. We define $f_c(x) = W_c f(x)$, where $f_c(x_i) \in R^M$. We use k to denote a neuron of interest in the projection layer, and its activation pattern is denoted as q_k where $q_k^\top = [f_{c,k}(x_1), \ldots, f_{c,k}(x_N)]^\top$, with $q_k \in R^N$.

To make the neurons in the CBL interpretable, we need to enforce the projected neurons to activate in correlation with the target concept, which can be formulated as optimizing W_c to maximize the CLIP-Dissect similarity between activation pattern and the target concept. To optimize this similarity, we have designed a new and fully differentiable similarity function $\text{sim}(q_i, t_i)$ that can be applied to CLIP-Dissect, called *cosine cubed*, that still achieves very good performance in explaining the neuron functionality. Our goal is to minimize the following objective L over W_c in Equation (1):

$$L(W_c) = \sum_{i=1}^{M} -\text{sim}(q_i, t_i) := \sum_{i=1}^{M} -\frac{\bar{q_i}^3 \cdot \bar{P_{:,i}}^3}{||\bar{q_i}^3||_2 ||\bar{P_{:,i}}^3||_2}. \tag{1}$$

Here $\bar{q}$ indicates vector q normalized to have mean 0 and standard deviation 1, and the *cosine cubed* similarity $\text{sim}(q_i, t_i)$ is simply the cosine similarity between two activation vectors after both have been normalized and raised to third power element-wise. The third power is necessary to make the similarity more sensitive to highly activating inputs. As this is still a cosine similarity, it takes values between $[-1, 1]$. We optimize L using the Adam optimizer on training data, with early stopping when similarity on validation data starts to decrease. Finally to make sure our concepts are truthful, we drop all concepts j with $\text{sim}(q_j, t_j) < 0.45$ on validation data at the end of training W_c. This is the concept set filter #5 from Sec 3.1. This cutoff was selected manually as a good indicator of a neuron being interpretable. During this filtering, the number of concept is reduced: $M \leftarrow M - \Delta$, where Δ is non-negative integer representing the number of concepts being removed in this step. Note that matrix W_c has also to be updated accordingly by removing the rows that corresponds to the removed concepts, and with our notation, $W_c \in R^{M \times d}$. To simplify Figure 2, we omit plotting this concept removal step.

3.3 Step 4: Learning the sparse final layer

Now that the Concept Bottleneck Layer is learned, the next task is to learn the final predictor with the fully connected layer $W_F \in R^{d_z \times M}$ where d_z is the number of output classes. The goal is to keep W_F sparse, since sparse layers have been demonstrated to be much more interpretable, as shown in (Wong et al., 2021). Given that both the backbone $f(x)$ and concept projection W_c are fixed, this is a problem of learning a sparse linear model. This can be solved efficiently with the elastic net penalty:

$$\min_{W_F, b_F} ||W_F f_c(X_{\text{train}}) + b_F - y||_2^2 + \lambda R_\alpha(W_F) \tag{2}$$

where $R_\alpha = (1-\alpha)\frac{1}{2}||W_F||_2^2 + \alpha||W_F||_1$ and b_F denotes the bias of the FC layer. We optimize Equation (2) using the GLM-SAGA solver created by (Wong et al., 2021). For the sparse models, we used $\alpha = 0.99$ and λ was chosen such that each model has 25 to 35 nonzero weights per output class. This level was found to still result in interpretable decisions while retaining good accuracy. Depending on the number of features/concepts in the previous layer this corresponds to 0.7-15% of the weights of the model being nonzero.

4 Results

4.1 Evaluation setup

To evaluate our approach, we train Label-free CBMs on 5 datasets. These are CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), CUB (Wah et al., 2011), Places365 (Zhou et al., 2017) and ImageNet (Deng et al., 2009). This is a diverse set of tasks, where CIFAR-10/100 and ImageNet are general image classification datasets, CUB is a fine-grained bird-species classification dataset and Places365 is focused on scene recognition. Their sizes vary greatly, with CUB having 5900 training samples, CIFAR datasets 50,000 each and ImageNet and Places365 have 1-2 million training images. CUB contains annotations with 312 concepts for each datapoint, such as *has wing color:blue* or *has head pattern:spotted*. We used neither the concept names or the concept annotations to train our LF-CBM models to showcase our ability to perform without labels, yet our method discovered similar concepts and is competitive with methods that utilize the available concept information.

		Dataset				
Model	Sparse final layer	CIFAR10	CIFAR100	CUB200	Places365	ImageNet
Standard	No	88.80%	70.10%	76.70%	48.56%	76.13%
Standard (sparse)	Yes	82.96%	58.34%	**75.96%**	38.46%	**74.35%**
P-CBM*	Yes	70.50%	43.20%	59.60%	N/A	N/A
P-CBM (CLIP)*	Yes	84.50%	56.00%	N/A	N/A	N/A
Label-free CBM (Ours)	Yes	**86.40%** $\pm$ 0.06%	**65.13%** $\pm$ 0.12%	74.31% $\pm$ 0.29%	**43.68%** $\pm$ 0.10%	71.95% $\pm$ 0.05%

Table 2: Accuracy comparison, best performing sparse model bolded. We can see our method outperforms Posthoc-CBM and matches or outperforms a sparse standard model. *Indicates reported accuracy. The results for our method are average over three training runs, displayed together with standard deviation.

For the model backbone, for CIFAR and CUB we use the same backbones as (Yuksekgonul et al., 2022) for fair comparison, so we use CLIP(RN50) image encoder as the backbone for CIFAR, and ResNet-18 trained on CUB from imgclsmob for CUB. For both ImageNet and Places365 we use ResNet-50 trained on ImageNet as the backbone network. The number of concepts each model uses is roughly proportional to the number of output classes for that task, as each class adds more initial concepts. The number of concepts for our models are as follows: 128 for CIFAR-10, 824 for CIFAR-100, 211 for CUB, 2202 for Places-365 and 4505 for ImageNet. CUB has a smaller number of concepts because we only used the *important features* prompt. All models are trained on a single Nvidia Tesla P100 GPU, and the full training run takes anywhere from few minutes to 20 hours depending on the dataset size. The majority of runtime is taken by step 2, where we save activations for both the backbone and CLIP over the entire training dataset. Fortunately, these results only need to be calculated once and can be reused for training new CBMs on the same dataset. Once the activations have been saved, learning the model takes less than 4 hours on all datasets.

4.2 Accuracy

Table 2 shows the performance of **Label-free CBM** on all 5 datasets. We can see that our method can create a CBM with a sparse final layer on all datasets with little loss in accuracy even on very large datasets, including a model with 72% top-1 accuracy on the challenging ImageNet. Our **Label-free CBM** noticeably outperforms Post-hoc CBM on the datasets evaluated, but some rows for P-CBM are N/A as they do not provide results for those datasets and it is unclear how to scale their method to larger datasets. We did not include methods with non-interpretable components (i.e. P-CBM-h or IBD) for comparison as we want to focus on fully interpretable CBM models. Note that P-CBM uses the expert provided concept set for CUB200, yet we can outperform it using our fully GPT-3 derived set of features. The standard sparse models were finetuned by us by learning a sparse final layer directly after feature layer $f(x)$ as described in Wong et al. (2021) and also have 25-35 nonzero weights per class. Sparse standard models have accuracies comparable to our CBM, indicating the CBM does not cause loss in accuracy.

4.3 Explainable decision rules

Perhaps the biggest benefit of Concept bottleneck models is that their decisions can be explained as a simple linear combination of understandable features. To showcase this, in Figure 3 we visualize the final layer weights for the classes "Orange" and "Lemon" on ImageNet, and classes "Mountain" and "Mountain Snowy" on Places365. The visualization is a Sankey diagram of the final layer weights for two output classes, with the weight between a concept and output class displayed as the width of the line connecting them. We have only included weights with absolute value greater than 0.05 (for comparison the largest weights are usually 0.5-1). Negative weights are denoted as NOT concept. The learned decision rules largely align with our intuitions. For ImageNet, the concept "citrus" fruit is highly connected to both "Orange" and "Lemon", while orange colors activate the "Orange" class and yellow colors and limes activate the "Lemon" class. On Places, concepts like "a high elevation"

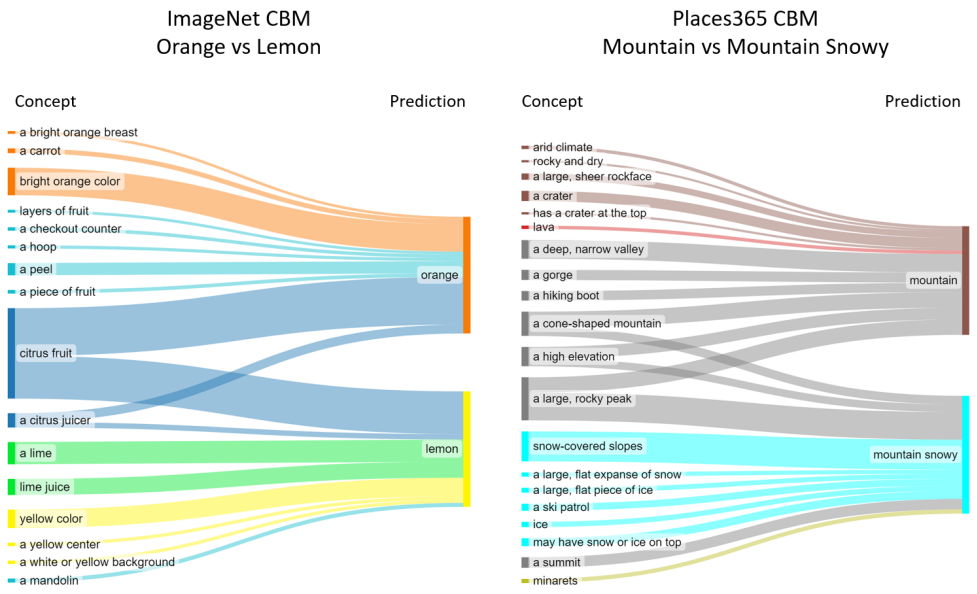

Figure 3: Visualization of the final layer weights of our **Label-free CBM**. Showcasing how our models differentiate between two similar classes. The width of the line is proportional to connecting weight.

activate both classes, while "Mountain Snowy" is activated by snow and ice related concepts and "Mountain" is activated by concepts related to volcanoes like "lava" and "crater". Visualizations like this allows us to gain a *global* understanding of how our model behaves.

4.4 Explainable individual decisions

In addition to global understanding, Concept Bottleneck models allow us to understand the reasoning behind individual decisions. Since our decisions are linear functions of interpretable features they can be explained in a simple and accurate way. After learning the Concept Bottleneck Layer, we normalize the activations of each concept to have mean 0 and standard deviation 1 on the training data. Given normalized features, the contribution of feature j to output i on input x_k can be naturally calculated as $\textit{Contrib}(x_k, i, j) = W_{F[ij]} f_c(x_k)_j$. Since our W_F is sparse, most contributions will be 0 and the important contributions can be easily visualized with a bar plot. Example visualizations for explaining the models predicted class are shown in Figure 4 and more visualizations are available in Appendix B.5. These visualizations show the concepts with highest absolute value contribution. Concepts with negative activation can still be important contributors, and they are shown as "NOT concept" in our visualizations. We can see for example the CUB model correctly identifies "a red head" as the most important feature in recognizing this image as "Red headed Woodpecker", while the fact that there is no "a rosy breast" present also increases models confidence.

5 Case study: Manually improving an ImageNet model

In this section we take a deep dive into our ImageNet **Label-free CBM** – we analyze the errors it makes and design ways to debug/improve our network based on these findings. In particular, we show how we can inspect a handful of individual incorrect decisions and manually change our model weights to not only fix those predictions but also multiple other predictions, improving the networks overall accuracy. As far as we know, this is the first example of manually editing a large well-trained neural network in a way that improves test accuracy on the same distribution.

Figure 4: Visualization of two correct decisions made by our **Label-free CBM**.

5.1 Types of model errors

During the course of our experiment, we were able to identify 4 different types of mistakes our model makes, with different strategies for fixing them:

Type 1: Incorrect/ambiguous label: Despite looking like errors in terms of accuracy, these are simply a consequence of having noisy labels and are unavoidable. We will not attempt to fix this type of mistakes.

Type 2: No sufficient concept in CBL: Despite having a large set of concepts, our CBL does not include all concepts important for detecting certain classes. For example, detecting the difference between two species of snakes may require recognizing very fine-grained patterns that can't be easily explained in words. While these could often be fixed by adding new concepts to the concept set, we do not focus on it as it requires retraining the projection $W_{c'}$ and the final layer.

Type 3: Incorrect concept activations: Some of the time, the activations in our CBL are incorrect and do not match the image, causing the prediction to become incorrect. Since the network up to the Concept Bottleneck Layer is mostly a black box, we can't easily improve the predictions, but predictions can be improved with test time interventions as seen in Figure 5.

Type 4: Incorrect final layer weight W_F: Sometimes even if all the concept activations are correct, the models final layer weights cause it to still make an erroneous prediction. As the final layer is fully interpretable, fixing these errors is the main focus of our study described in section 5.2.

See Figure 10 in the Appendix for examples of Type 1 and Type 2 errors.

5.2 Editing final layer weights

In this section we describe our procedure for manually editing final layer weights of our model. This is done with the following procedure:

1. **Find an incorrect prediction belonging to Type 4 errors**: This is done by visualizing incorrect model predictions and their explanations, and identifying Type 4 errors, i.e. the highly activating concepts looks correct and the label is correct and unambiguous.
2. **Identify a concept to edit.**: Perhaps the most tricky step, to flip a prediction we need to select a concept that is highly activated in this image, important for the ground truth class and not important for the incorrect class using our domain knowledge, and will have a

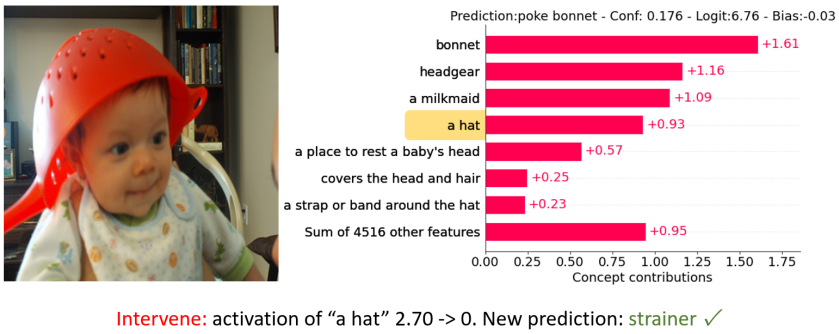

Figure 5: Here we fix an incorrect model prediction on ImageNet by simply zeroing out the incorrect activation for concept "a hat". Model originally predicted "poke bonnet" which is a type of hat.

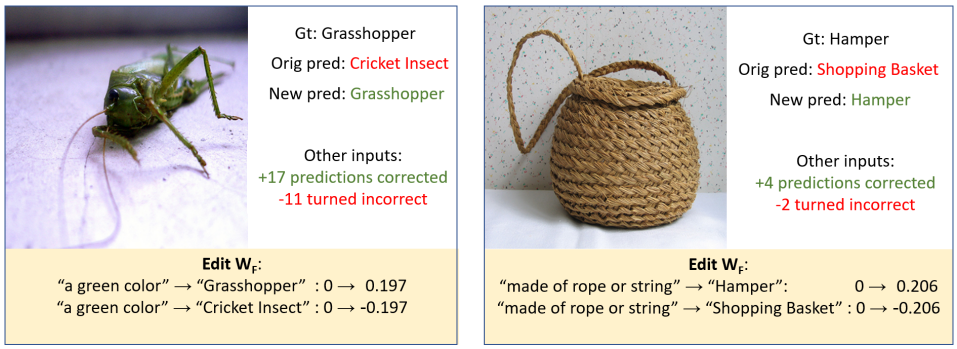

Figure 6: Examples of two edits we performed on our model predictions. Both edits fix the error we found, and fix more other predictions than they cause mistakes on other unseen inputs.

small impact on other predictions. For example to flip a prediction from "cricket insect" to "grasshopper" we chose the concept "a green color".

3. **Change weights by sufficient magnitude**: Next we choose magnitude of weight change Δw, and edit the weights in the following way: $W_{F[gt,concept]} \leftarrow W_{F[gt,concept]} + \Delta w$ and $W_{F[pred,concept]} \leftarrow W_{F[pred,concept]} - \Delta w$. Typically we want Δw to be slightly more than required to flip the prediction on our example, but understanding how it affect other predictions is hard and should be checked with a validation dataset.

Figure 6 shows two examples of model edits we performed, and their effects on model predictions on the validation set. In total we identified 5 such beneficial edits during our quick exploration, the other 3 are shown in Figure 9. After applying all of them, the total validation accuracy of our model goes from 71.98% to 72.02%, correcting 38 predictions while turning 17 incorrect. While this might seem like a small difference, it is worth noting we only change 10/4.5 million total weights in the final layer, of which 34,550 are non-zero. Our edits only affect 10 classes which have a total of 500 validation examples, so in effect we increase accuracy on this subset by 4.2% which is a significant boost. While labor-intensive, we believe this editing approach is very promising for future use, where practitioners could for example notice an incorrect prediction in a production system, quickly devise a fix for that prediction, and check how it affects other predictions using validation data.

6 Conclusion

We have presented **Label-free CBM**, a fully automated and scalable method for generating concept bottleneck models and used it to create the first high performing CBM on ImageNet. In addition, we have demonstrated how our models are easily understandable in terms of both global decision rules and individual predictions, and how to use this understanding to manually improve model weights in a fully trained ImageNet model in a way that improves validation accuracy.

7 REPRODUCIBILITY

We have described our algorithm in detail, including hyperameters and other details in sections 3 and 4. In addition we will release our code publicly prior to publication.

REFERENCES

Abubakar Abid, Mert Yuksekgonul, and James Zou. Meaningfully explaining model mistakes using conceptual counterfactuals. *arXiv preprint arXiv:2106.12723*, 2021.

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *European conference on computer vision*, pp. 351–369. Springer, 2020.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2021. URL https://arxiv.org/abs/2108.07258.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pp. 5338–5348. PMLR, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Max Losch, Mario Fritz, and Bernt Schiele. Interpretability beyond classification output: Semantic bottleneck networks, 2019. URL https://arxiv.org/abs/1907.10882.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *arXiv preprint arXiv:2202.05262*, 2022.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale, 2021. URL https://arxiv.org/abs/2110.11309.

Tuomas Oikarinen and Tsui-Wei Weng. Clip-dissect: Automatic description of neuron representations in deep vision networks. *arXiv preprint arXiv:2204.10965*, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, 2021.

Robyn Speer and Catherine Havasi. Conceptnet 5: A large semantic network for relational knowledge. *The people's web meets NLP, theory and applications of natural language processing*, pp. 161–176, 02 2013. doi: 10.1007/978-3-642-35085-6_6.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Henry Jin, Suzanne Petryk, Sarah Adel Bargal, and Joseph E. Gonzalez. Nbdt: Neural-backed decision trees, 2020.

Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. Rewriting geometric rules of a gan. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022.

Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In *International Conference on Machine Learning*, pp. 11205–11216. PMLR, 2021.

Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. *arXiv preprint arXiv:2205.15480*, 2022.

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. *arXiv preprint arXiv:2209.09056*, 2022.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 119–134, 2018.

List the things most commonly seen around a "tench":

-a pond
-fish
-a net
-a rod
-a reel
-a hook
-bait

List the things most commonly seen around a "beer glass":

-beer
-a bar
-a coaster
-a napkin
-a straw
-a lime
-a person

List the things most commonly seen around a "mountain":

-a peak
-a valley
-a forest
-rocks
-dirt
-a trail
-an animal

Give superclasses for the word "tench":

-fish
-vertebrate
-animal

Give superclasses for the word "beer glass":

-glass
-container
-object

Give superclasses for the word "grasshopper":

-insect
-arthropod
-animal

Figure 7: Full prompts used for GPT-3 concept set creation. Text in green generated by GPT, rest is our prompt.

A APPENDIX

A.1 LIMITATIONS

While our model performs well in terms of both interpretability and accuracy, it still has some limitations. For example, using a model like GPT-3 to produce concepts is stochastic, and sometimes may fail to generate important concepts for detecting a certain class. However our model is quite robust to small changes in the concept set as shown in sections B.1 and B.3. In general, automatic ways of generating concept set may sometimes lack the required domain knowledge to include some important concepts, and would perhaps be best used in collaboration with a human expert.

Second, our model is aimed to extend Concept bottleneck models to large datasets where labeled concept data is not available. However it works best on domains where CLIP performs well and as such may not perform as well on small datasets that require specific domain knowledge and labeled concept data is available, such as medical datasets. For such datasets we believe it is still better to use a method that can effectively leverage the labels available such as original Concept Bottleneck Models Koh et al. (2020) or P-CBMYuksekgonul et al. (2022).

A.2 ADDITIONAL FIGURES

Figures 7, 8 provide examples of the full prompts we used for GPT-3, as well as GPT outputs.

Figure 9 shows the additional model edits performed in our ImageNet CBM experiment, and Figure 10 showcases Type 1 and Type 2 errors made by our ImageNet CBM.

List the most important features for recognizing something as a "goldfish":

-bright orange color
-a small, round body
-a long, flowing tail
-a small mouth
-orange fins

List the most important features for recognizing something as a "beerglass":

-a tall, cylindrical shape
-clear or translucent color
-opening at the top
-a sturdy base
-a handle

List the most important features for recognizing something as a "scuba diver":

-a person wearing a scuba mask and breathing apparatus
-a wet suit
-flippers
-a diving tank

Figure 8: Full prompts used for GPT-3 concept set creation. Text in green generated by GPT, rest is our prompt.

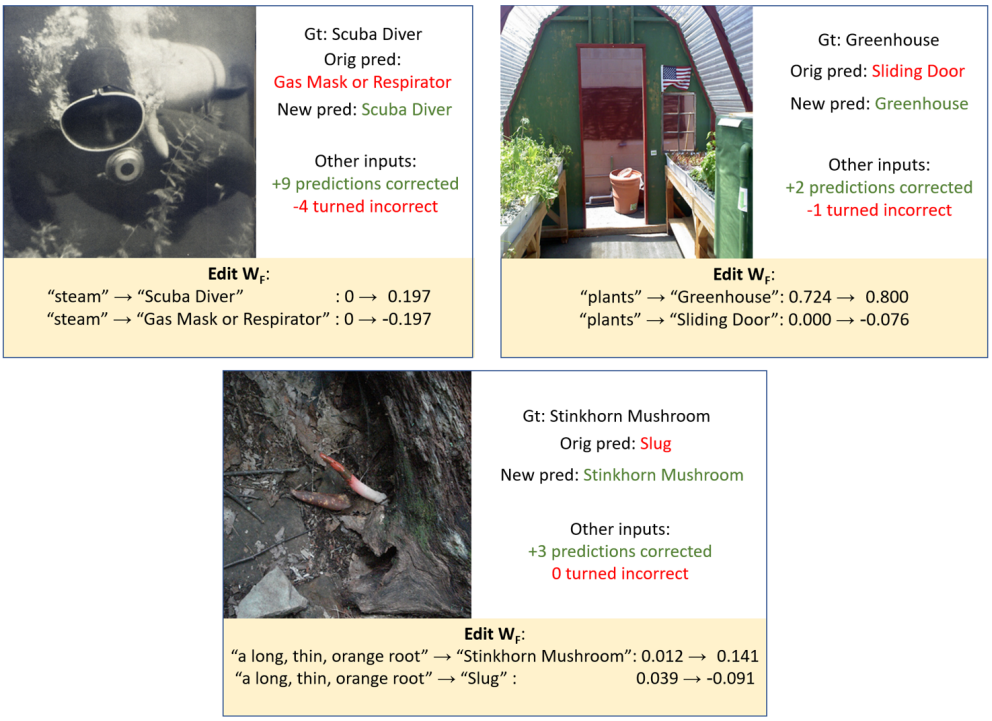

Figure 9: The other 3 model edits we performed in our experiment

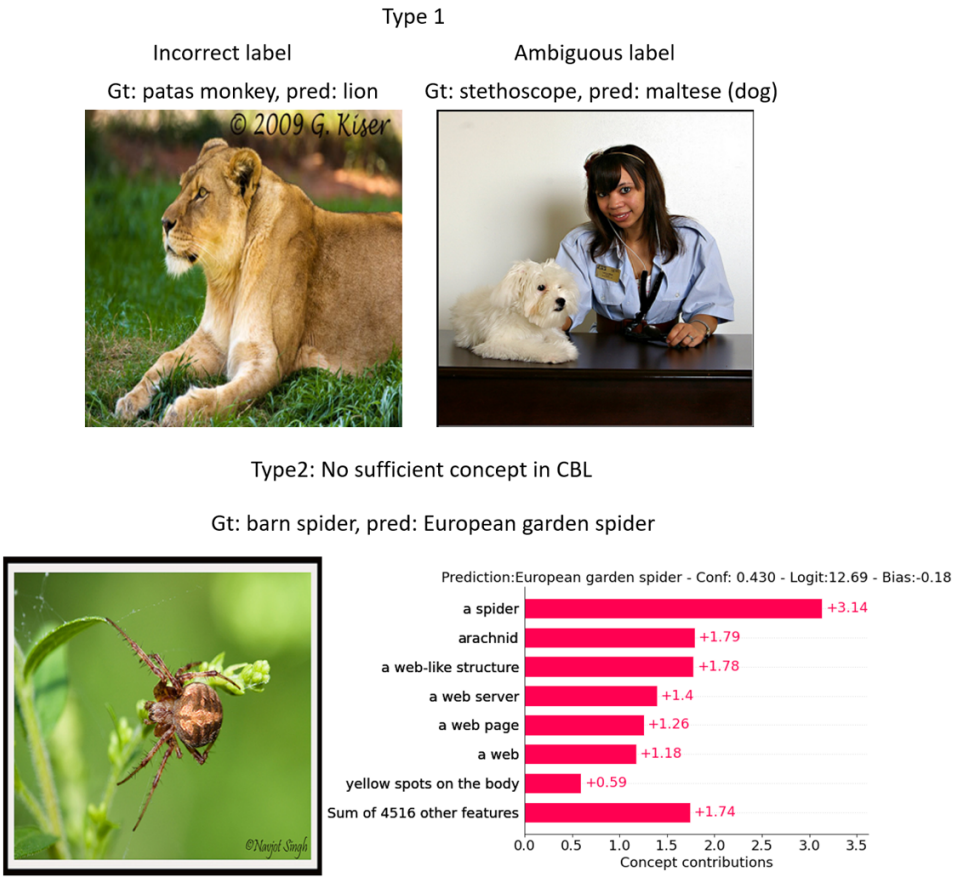

Figure 10: Examples of Type 1 and type 2 model errors.

B APPENDIX: NEW REBUTTAL EXPERIMENTS

B.1 ABLATION STUDY AND FURTHER DISCUSSION ON CONCEPT FILTERING IN SEC 3.1

In this section we study how each step in our proposed concept filtering affect the results of our method. In general our use of filters has two main aims:

- To improve the interpretability of our models
- Improve computational efficiency and complexity by reducing the number of concepts

The filters are not designed to improve performance of the models in terms of accuracy, in fact we would expect them to slightly reduce accuracy as a model with more concepts is larger and more powerful. To evaluate the effect each individual filter has we trained models on both CIFAR10 and ImageNet while deactivating one filter at a time, as well as one without using any filters at all. The results are displayed in Table 3. We can see that the accuracy of our models is not very sensitive to the filters at all, with ImageNet accuracy remaining constant and CIFAR10 slightly increasing its accuracy with less filters. Two ImageNet results are marked as N/A as the large number of concepts caused some memory/computational issues for training the models which we were unable to resolve in the rebuttal timeframe.

Filters	CIFAR10 Accuracy	CIFAR10 #concepts	ImageNet Accuracy	ImageNet #concepts
All filters	86.26%	142	71.89%	4380
No length filter	86.55%	146	71.88%	5347
No similarity to classes filter	86.74%	148	71.93%	4671
No similarity to other concepts filter	86.42%	151	N/A	6515
No CLIP activation filter	86.41%	147	71.92%	4478
No projection accuracy filter	86.33%	147	71.88%	4462
No filters at all	86.56%	177	N/A	9087

Table 3: Effect of our individual concept filters on the final accuracy and number of concepts used by our models.

To further show the effects of each filter, we will showcase the full filtering procedure for our CIFAR10 model with all filters. We start with a freshly generated initial concept set from GPT-3, which has a total of 177 concepts. We then take the following steps to filter down concepts:

1. **Delete concepts that are too long.** This step is important to make sure concepts are easy to visualize and simple non-convoluted concepts. For CIFAR-10 this leads to deletion of 3 concepts:
   - *white spots on the fur (in some cases)*, *feline features (e.g., whiskers, ears)* and *legs with two toes pointing forward and two toes pointing backwar*.
   174 concepts remain.
2. **Delete concepts too similar to output classes.** This step is required for the explanations to be informative, as it helps avoid trivial explanations such as: "This is image is a cat because it is a cat". This step removes the following CIFAR-10 concepts (with the class name it is too close to shown in brackets):
   - a cat (cat), a deer (deer), a horse (horse), a plane (airplane), a car (automobile), car (automobile), cars (automobile), vehicle (automobile), animal (dog), truck driver (truck)
   164 concepts remain
3. **Delete concepts too similar to other concepts.** In this step we aim to delete duplicate concepts from the concept set, i.e. two concepts that have the same semantic meaning should not both be part of the concept set. Duplicate concepts will cause unnecessary computational cost and confusing explanations. For CIFAR-10 we delete the following concepts (with the concept it is too similar to in brackets):
   - 4 wheels (four wheels), a food bowl (a bowl), a furry, four-legged animal (a large, four-legged mammal), a gasoline station (a gas station), a large boxy body (a large body), a

large, muscular body (a large body), a street (a road), a strong engine (an engine), large, bulging eyes (large eyes), the ocean (the sea)
154 concepts remain

4. **Remove concepts not present in training data.** In this step we remove all concepts that CLIP thinks are not really present in training data. The purpose is to avoid learning neurons that don't correctly represent their target concepts, as concepts missing from the dataset are unlikely to be learned correctly in the Concept Bottleneck Layer. For CIFAR-10 this step deletes the following concepts:
- a bit, a mechanic, legs, long legs, passengers, several seats inside, windows all around
147 concepts remain

5. **Remove concepts we can't project accurately.** The purpose of this step is similar to the previous step, we want to remove all concepts that are not faithfully represented by the CBL. To do this we evaluate the similarity score between the target concept and the activations of our new neuron using CLIP-Dissect Oikarinen & Weng (2022) on the validation data, and delete concepts with similarity less than 0.45 which was determined to be a good indicator of faithful concept representation. For CIFAR-10 we delete the following concepts:
- a bed, a coffee mug, a crew, a house, a wheel
We are left with our 142 final concepts.

In total our method has 5 different cutoff parameters that can be tuned. Since the main purpose of these filters is to make the network more interpretable, their values were mainly chosen through trial and error to find ones that produce the most useful and explainable models. Most of these cutoffs are independent of the dataset chosen, and the only one we changed from one dataset to another was the value of cutoff for images being present in the dataset (filter 4). We used a cutoff of 0.25 for CIFAR-10 and CIFAR-100, 0.26 for CUB-200 and 0.28 for Places and ImageNet. We found we had to change this cutoff as CLIP doesn't activate very highly on low resolution images of CIFAR, and in general we get higher top5 activations on larger datasets. Since the other 4 cutoffs are fixed, using our method for a new dataset won't require a large hyperparameter search, and as seen in Table 3 the overall performance is not very sensitive to specific filters.

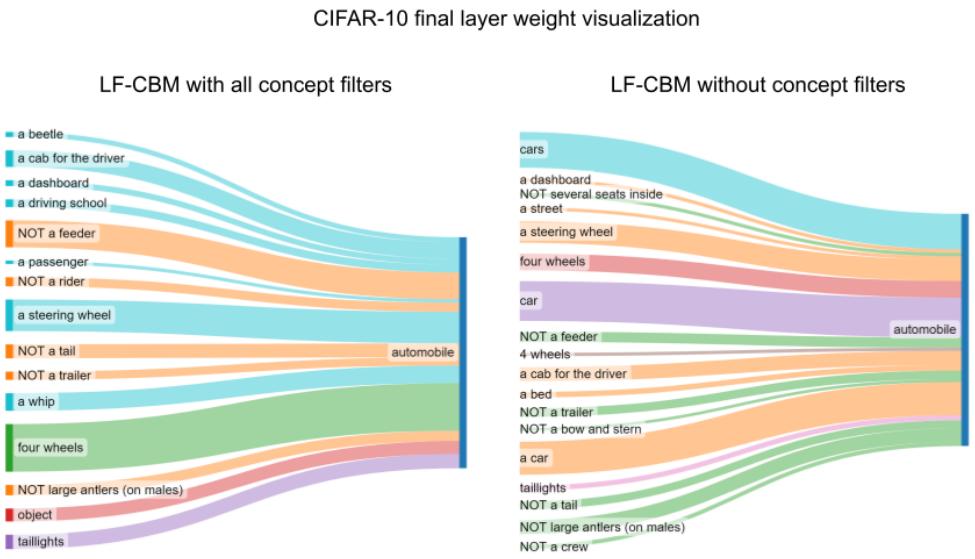

Figure 11: *Comparison of the final layer weights between CIFAR-10 model trained with and without filters.*

To visualize the effect of our filters, in Figure 11 we visualize the final layer weights for CIFAR-10 class "automobile" trained with and without filters. We can see the model with filters has found a reasonable decision rule, with largest weights corresponding to *four wheels* and *a steering wheel*.

On the other hand, the weights for the model without filters are quite problematic. First, the 3 largest weights are *cars*, *car* and *a car*. Not only do they have the same meaning as the final class itself (filter 2), thus lacking any explanatory power, they are duplicates of each other (filter 3), making the explanation unnecessarily compilicated. Finally we see the unrelated concept *bed* which is not present in CIFAR-10 having a small positive weight, which should be removed by either filter 4 or 5.

B.2 Ablation: Effect of Concept set

In this section we evaluate the effect our initial concept set generator has on our results. In our paper we proposed a new method of generating descriptions using GPT-3 Brown et al. (2020), and in this section we compare those results to the case where we generate the initial concept set using ConceptNet Speer & Havasi (2013) as proposed by Yuksekgonul et al. (2022). In general ConceptNet only works well when prompted with relatively common single word phrases, but many of class names for CUB200, Places365 and ImageNet consist of multiple words, such as the ImageNet class *great white shark*. To overcome this issue we split multi-word class names into single word components and add concepts related to each word to the concept set, in this case *great*, *white* and *shark*. After creating the initial concept set we use the same filtering procedure described in section 3.1B for both models. Table 4 displays the accuracy comparison between models trained on different initial concept setst. In general using GPT3 seems to provide a small (0.1-1.5%) accuracy boost on all datasets, with the exception of CUB200 where the ConceptNet model fails completely. This is because ConceptNet is unable to generate concepts for the highly specific class names in CUB200, such as *Groove billed Ani*, while GPT-3 is not troubled by this.

Model\Dataset	CIFAR10	CIFAR100	CUB200	Places365	ImageNet
LF-CBM (GPT-3) (original)	**86.40%**	**65.13%**	**74.31%**	**43.68%**	**71.95%**
LF-CBM (ConceptNet)	86.30%	63.62%	2.19%	42.49%	71.52%

Table 4: Accuracy comparison between using different methods for generating initial concept set.

B.3 Consistency of concept set generation

Since the initial concept set is generated using GPT-3, it is a random process and in this section we explore how much this noise effects our final results. When regenerating initial concept set and running our concept filtering pipeline for our ImageNet model, we got 4380 concepts, compared to the original 4523 concepts. In terms of accuracy, the model learned with regenerated concept set reached an accuracy of 71.89%, compared to the average of 71.95% with the original concept set. This indicates that the difference is very small in terms of accuracy. However, the exact concepts used and model weights can be pretty different as shown in Figure 12, despite all being relevent (e.g. there are many concepts that appear on both concept sets, e.g. Image 49962 has important concepts of "wipes", "toiletries" on both, though they have different corresponding contributions). Figure 12 shows the explanations for decisions on two random images for the original model and model trained with regenerated concept set.

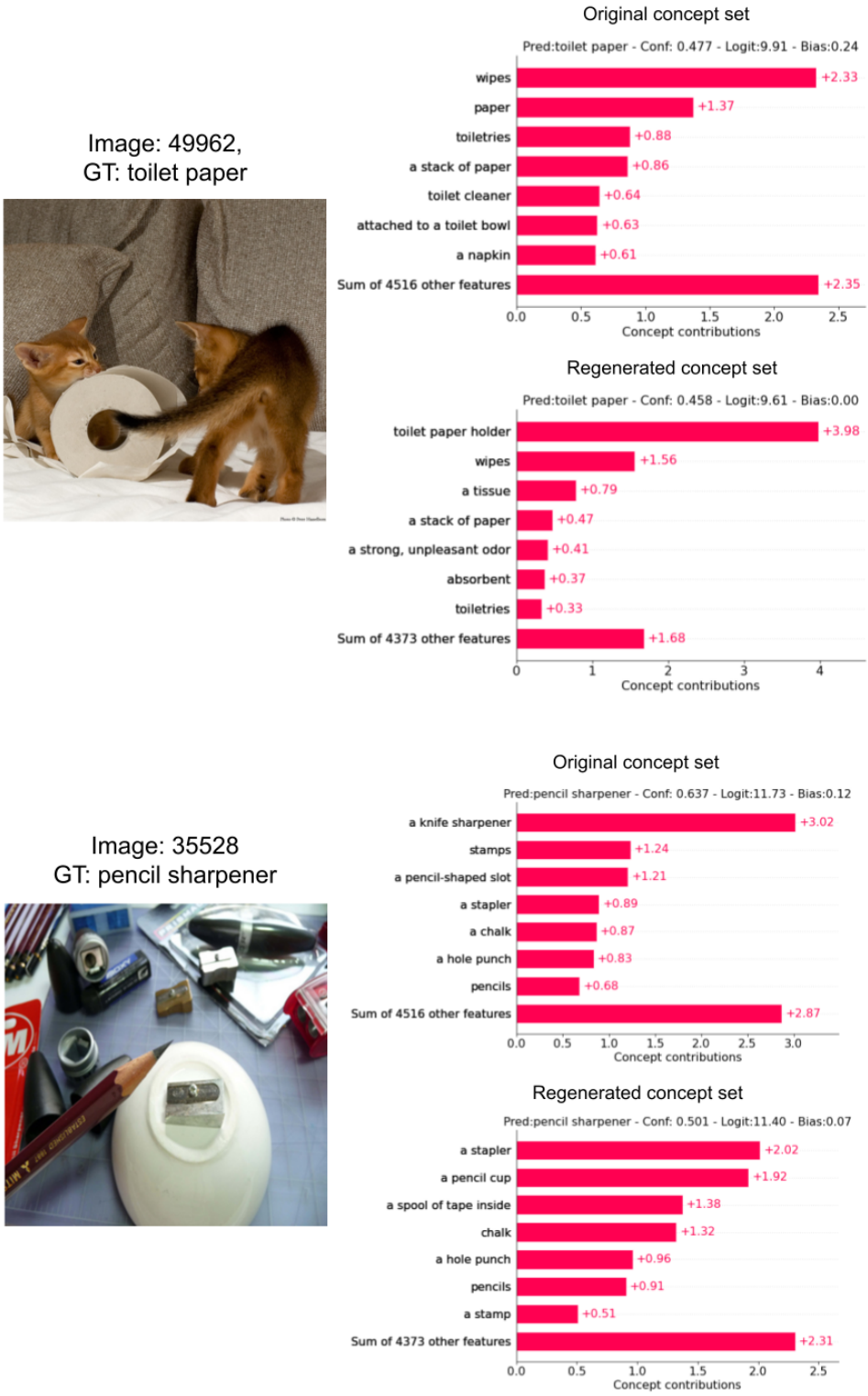

Figure 12: Difference in concepts used for a decision and their weights for models trained with original vs regenerated concept set, shown on two random images.

B.4 ADDITIONAL WEIGHT VISUALIZATIONS FOR RANDOM CLASSES

In figures 13 and 14 we have visualized the final layer weights leading to 4 randomly chosen output classes for both Places365 and ImageNet. These were created using the same procedure as Figure 3, with minor differences such as focus on single class at a time and no manual editing of concept colors according to semantic similarity. We can see that the concepts that have large weights are indeed relevant to the class, and overall the weights look reasonable.

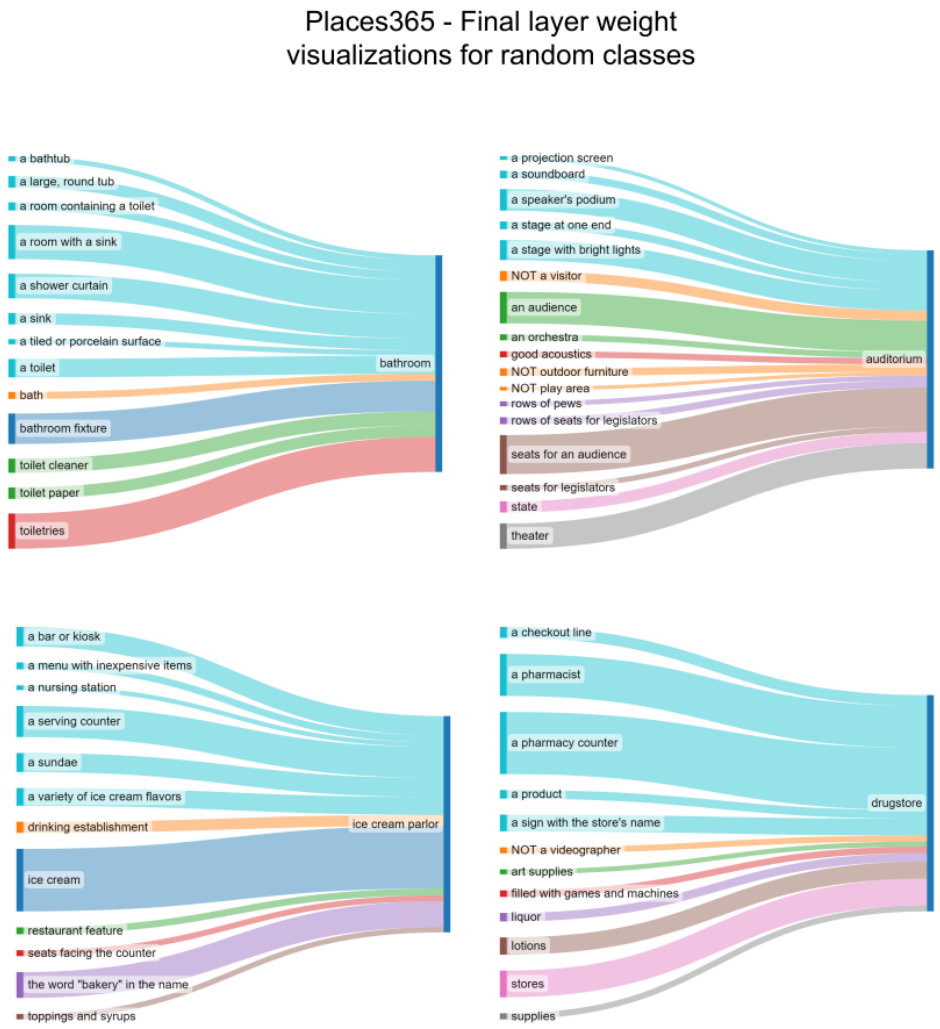

Figure 13: Weight visualizations for 4 randomly chosen output classes for our CBM trained on Places365.

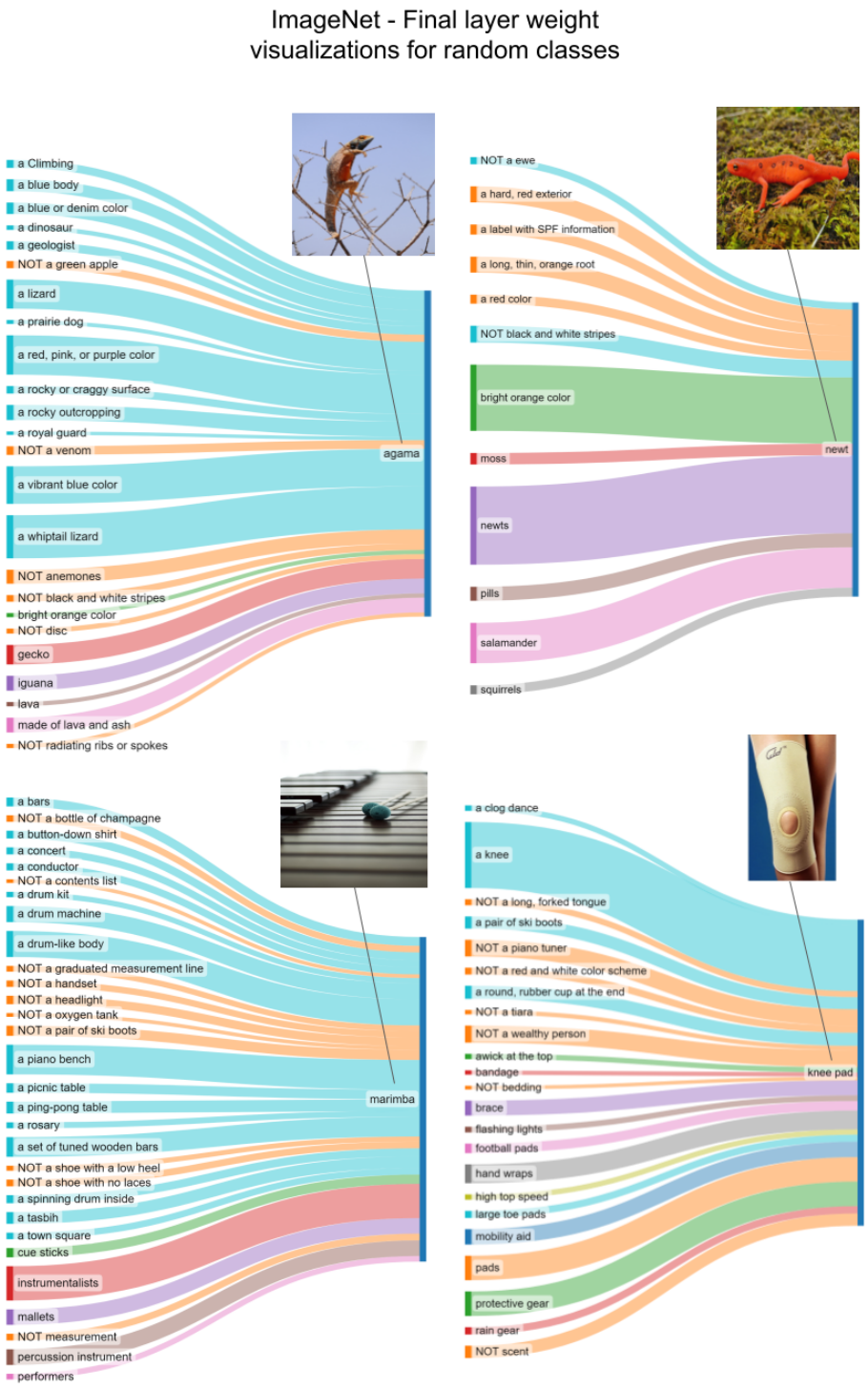

Figure 14: Weight visualizations for 4 randomly chosen output classes for our CBM trained on ImageNet, as well as an example image of each class to clarify their meaning.

B.5 ADDITIONAL EXPLANATIONS FOR RANDOM IMAGES

In figures 15, 16, 17 we display additional samples of decision explanations for 4 randomly chosen input images for each of Places365, ImageNet and CUB-200 respectively. Overall, it can be seen that the concepts and explanations are of good quality, even with the CUB-200 dataset which may require very specific knowledge and concepts to create CBMs. This result demonstrates the effectiveness of our Lf-CBM, which does not use the knowledge-specific concepts sets unlike existing works on CBMs.

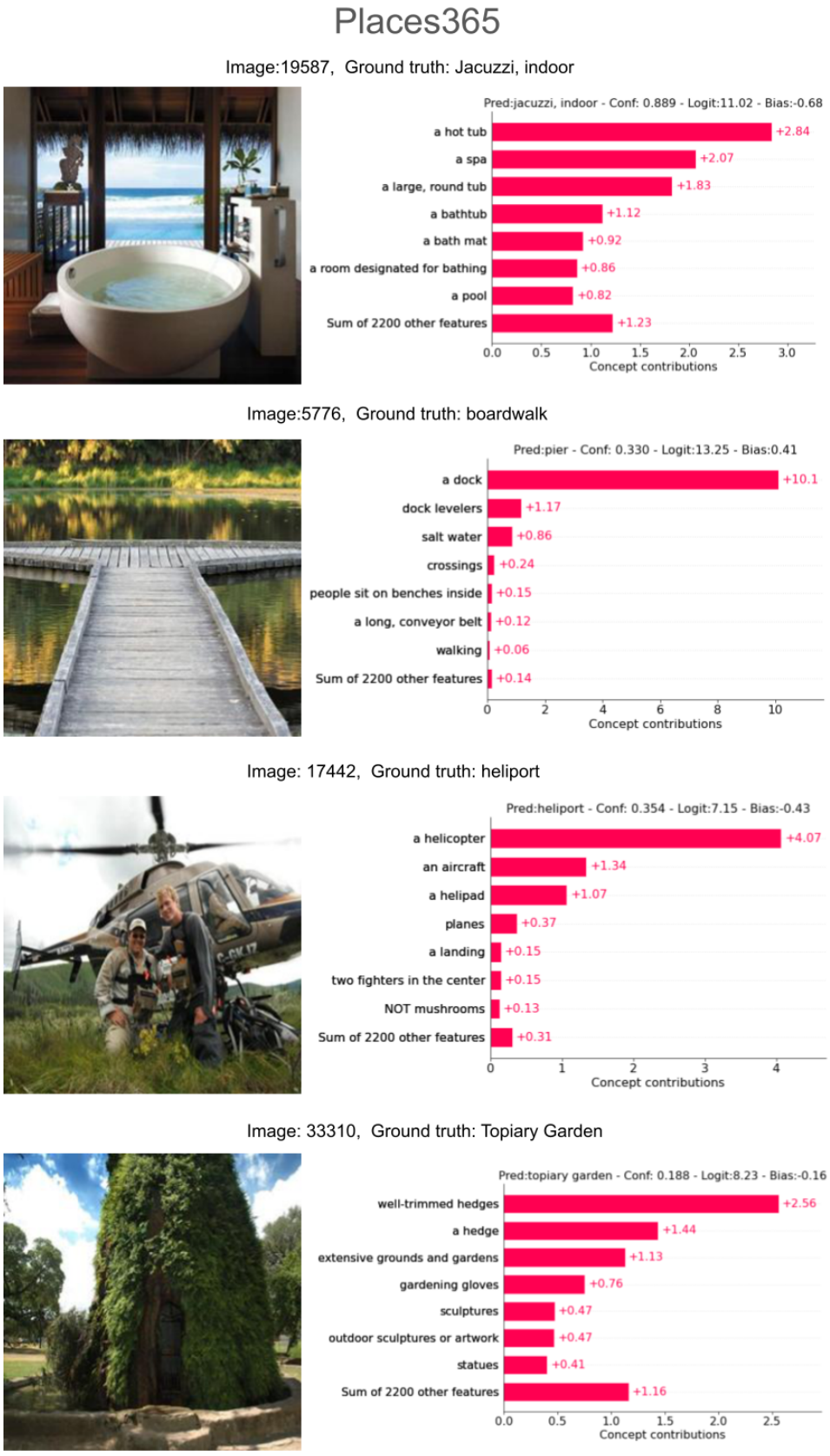

Figure 15: Explanations for 4 randomly chosen input images for our CBM trained on Places365.

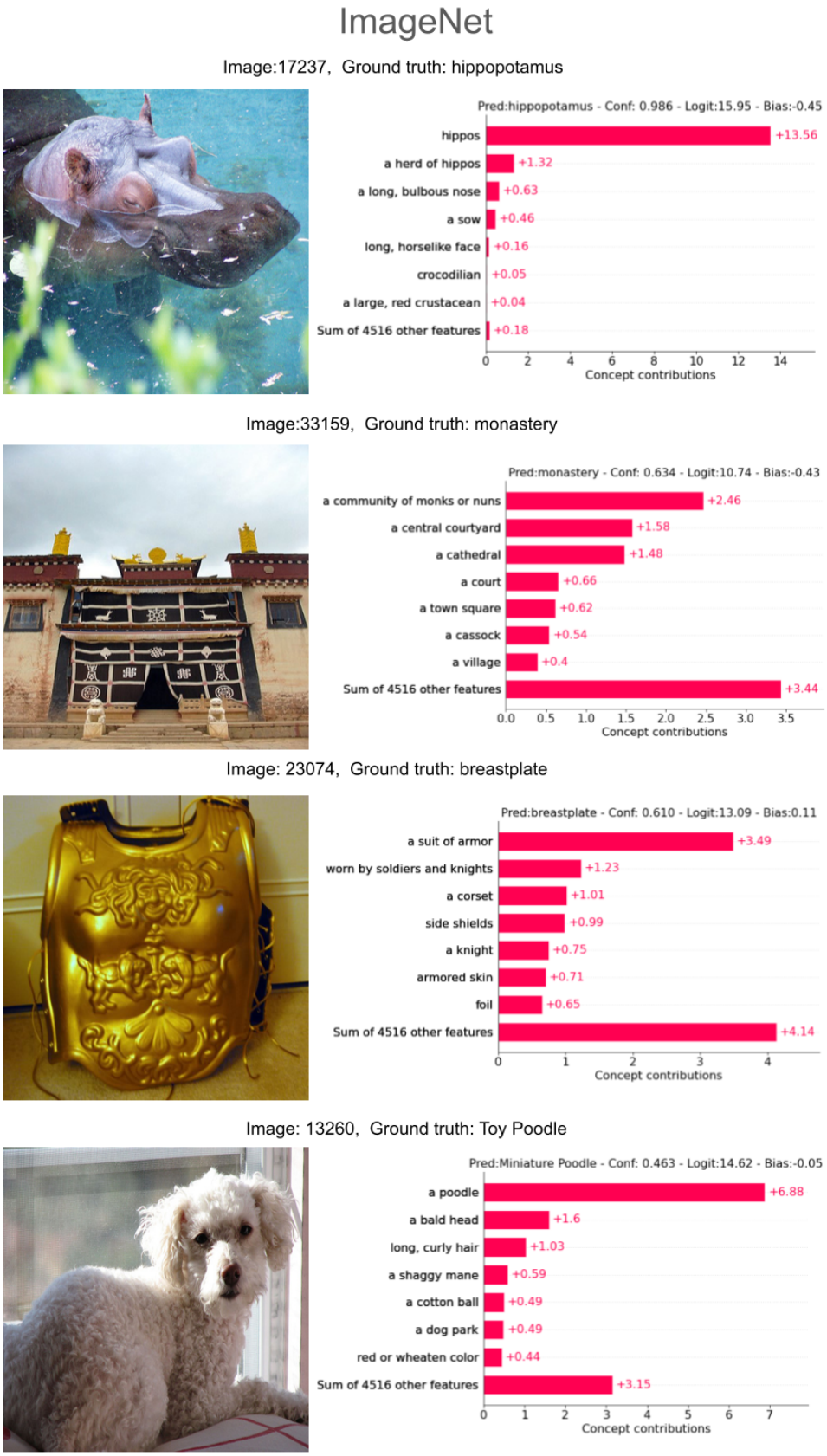

Figure 16: Explanations for 4 randomly chosen input images for our CBM trained on ImageNet.

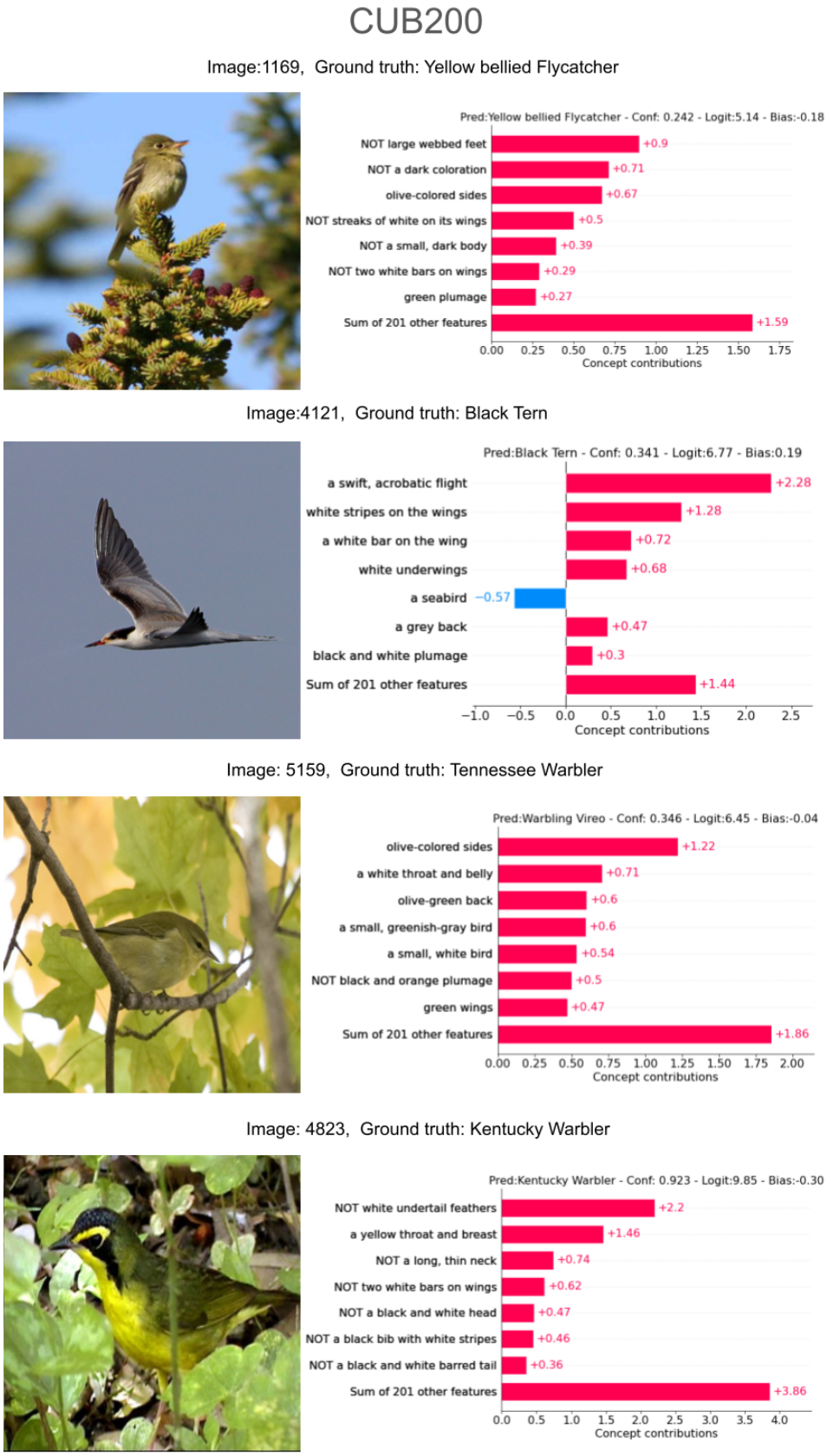

Figure 17: Explanations for 4 randomly chosen input images for our CBM trained on CUB-200.

B.6 ABLATION: LF-CBM WITHOUT SPARSITY CONSTRAINT

To estimate the effect sparsity has on our models, we show the accuracy of LF-CBM models trained without sparsity constraints (LF-CBM (dense)) in Table 5. We can see that not using a sparsity constraint greatly improves accuracy (except for CUB200 which starts to overfit), but it comes at a very large cost for interpretability as can be seen in Figure 18, where almost none of the decision is explained by the 10 most highly contributing concepts.

Model\Dataset	CIFAR10	CIFAR100	CUB200	Places365	ImageNet
LF-CBM (sparse) (original)	86.40%	65.13%	74.31%	43.68%	71.95%
LF-CBM (dense)	87.50%	67.93%	74.25%	48.25%	74.09%

Table 5: Results of training our CBM without sparisity constraints on the final layer.

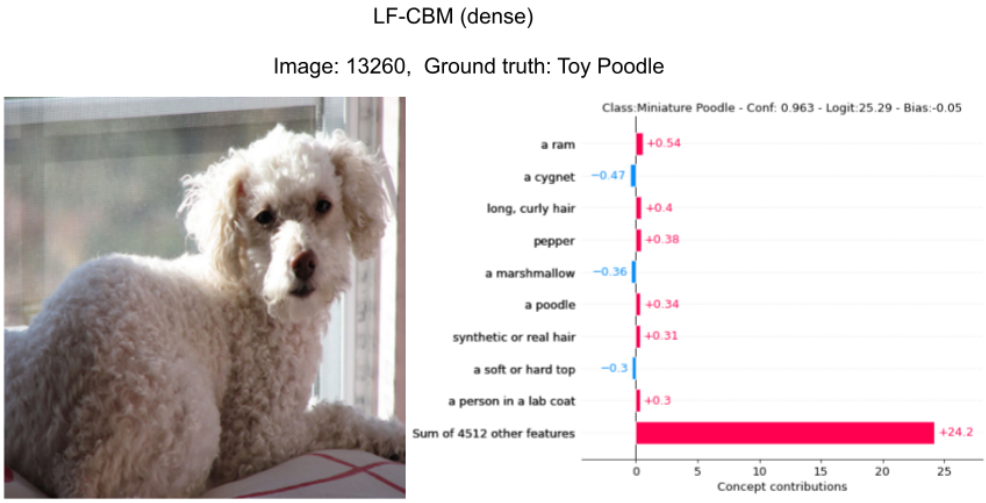

Figure 18: Dense CBM explanation for the same image as the fourth image in Figure 16. We can see the dense model is practically uninterpretable.