Concept Bottleneck Models Evaluation

Diego Belfiore - 245633

February 2, 2025

Abstract

Taking as a reference the 2020 work by Wei Koh et al. [1] I aim to evaluate the concepts quality and the explanations outputs of a Concept Bottleneck Model (CBM): an architecture that learns mappings from raw input to high-level concepts used for final predictions. The authors leverage the CBM possibility of factual explanation with the expressivity of Neural Networks (NNs), proposing a technique that obtains both high concept prediction and final classification accuracy. The realm of explainability is central to real world AI application. Finance and medicine are among the fields where a chance of indicating to stakeholders how models get to their results is pivotal. My personal interest in the topic evolved exploring the Cognitive Science literature. In recent years we have seen an evident acceleration in the use of NNs for multiple tasks, ranging from emotional face recognition (Brooks et al., 2024 [2]), to 3D shape computation (Olikka et al., 2024 [3]), often lacking depth in explaining how models behave. I believe that implementing CBMs could equip the field with useful tools for the next research.

1 Technique

1.1 Architecture

Koh et al. propose different kinds of CBMs. I focused on the architecture comprehending an encoding Convolutional NN called InceptionV3[4] and a logistic regressor as classifier. The authors modify the fully connected layer of InceptionV3 aligning it with the concepts' number. Four possible models are presented. The independent bottleneck model has the encoder that learns vectors representing the concepts of each image, the classifier is trained on the ground truth concepts and tested with the encoder's vectors. The sequential bottleneck is identical as the one before but directly uses the encoder's vectors to train the classifier. The joint bottleneck, introduces a weighting of the concepts during the training, it is probably the most expressive among the models because it allows different levels of concepts supervision for the final prediction. The standard model ignores concepts learning. These four models can be expressed through loss functions combining and optimizing $\hat{f}$ (final predictions) and $\hat{g}$ (concepts predictions).
The repository proposed by the authors was solid and outside minor modifications, the only functional change I made on an already existing file has been on "inference.py" where I implemented a features extractor from the bottleneck. All code changes are listed in the Appendix.

1.2 Dataset

I used the Caltech-UCSD Birds-200-2011 (CUB) dataset (Wah et al., 2011 [5]). It is composed of 11,788 photographs of birds from 200 species, with 312 annotated concepts for each image. The authors implement a filtering process that reduces the concepts to 112, only classes that majorly represent concepts are retained.
Data preprocessing for training followed best practices proposed by Cui et al. in 2018 [6], each picture was augmented with random color jittering, random horizontal flip and randomly cropped with dimensions 299 x 299. While the material already includes predefined dataset splits, I ensured that I could generate them myself. Running "preprocessing.py", I obtained the same output files provided by the authors. It was noticeable that the code sums training and validation data to achieve a bigger training set, in this context the technique makes sense: every time a new epoch starts the images are re-augmented.

1.3 Evaluation Metrics

Evaluating the quality of the learned concepts of these models is nothing trivial. Do models with good classification performance leverage plausible features? Are the concepts distinct or do they noisily overlap? Also, we know from the 2021 work

of Mahinpei et al.[7] that CBMs are potentially susceptible to concept leakage, where concept vectors may not be as clean as intended, incorporating latent information that is not explicitly related to the defined concepts, thus undermining models explainability potential. To address these questions I focused on three main metrics.

- Fidelity: the level in which concepts do preserve relevant information for the final prediction. I propose to compute the Fidelity as the difference between the prediction accuracy of the standard model (no concept supervision) and the accuracy of the sequential bottleneck model (full concept supervision)
- Diversity: represents how much concepts are distinct. I have chosen to compute it through the Spearman correlation among all the predicted concepts. Ranging from -1 to 1, values around 0 would indicate poor correlation among the features, then better results for the purpose of this research.
- Grounding: indicates the alignment with human categorization. Since the CUB dataset is human annotated we can consider the accuracy of predicted concepts as a suitable indicator of human-machine alignment.
- Top1Class Accuracy: reported for understanding models final classification performance.

2 Experimental area

2.1 Setup

To set up, I followed the guidelines proposed by Koh et al. The initial weights of the architectures have been the ones of "inception_v3_google-1a9a5a14", learned on ImageNet. The number of epochs of each training has been set up by default to 1000; an early stopping technique was already implemented in the code. The default early stopping was set to block the execution after one hundred epochs without performance increase. This epochs amount may be considered somehow conservative but clearly oriented to train models for a fine grained expressivity. Starting some trial runs I noticed that all CBMs rapidly achieved near ceiling performance during the first epochs, then the increase started happening after thousands of batches. To optimize the available resources I fixed the early stopping function to block after 5 epochs without improvement. Weight_decay, learning rate and scheduler step values were originally searched through a set of multiple experiments ranging among different hyperameters configurations. The authors propose a standard configuration of -weight_decay 0.0004 -lr 0.001 -scheduler_step 1000 which I applied too. The pivotal weighting parameter for the Joint Models is called attr_loss_weight, the authors claim that the best balance between performance on classes and concepts happens when the value is set to 0.01. Attr_loss_weight set to a narrower 0.001 has instead shown to positively affect class performance. In my work I considered all these different configuration and trained each model once. Model Accuracy on Concepts and Top1 Class resulted mostly coherent with the ones found in the author's reference paper (See Table 2 and Table 3).

2.2 Concepts Evaluation

Fidelity: The average Top1Class accuracy of the standard model is 82.27 while the sequential bottleneck achieves 75.62 accuracy. Their difference amounts to 6.65. This may indicate that full concept supervised training leads to the loss of important information for the final prediction.

Diversity: Independent and sequential CBM encoder output vectors show a Spearman correlation of .3 suggesting low correlation among the concepts. The standard model bottleneck vectors, that neglects concept supervision, show a Spearman correlation of .2. Both the Joint models show a .25 Spearman correlation, that would suggest again a good differentiation among the features.

Grounding: Fully supervised Inception shows concept accuracy of 96.81, the model appears reliable in terms of human-machine alignment. This value suggests that the model's predicted concepts align with the human-annotated concepts in the CUB dataset. The joint .01 and .001 models reports an accuracy of .93 and .94 respectively. This may suggest less alignment with human annotated data than full concept supervised models. The standard model shows concept accuracy of .50 which is in any case more than chance (chance = 1/112). All concept evaluation results are schematized in Table 1.

Models	Diversity (r)	Accuracy % ($\hat{g}$)
Ind	.3	96.81
Seq	.3	96.81
Joint .01	.25	93.46
Joint .001	.25	94.11
Std	.2	50.7

Table 1: Concepts Acc. Comparison

3 Discussion

3.1 Joint models differences

The training of the joint models brought to fuzzy results that do not replicate the authors metrics. Joint .01 model should have better concept accuracy than the reached one and given the higher supervision on concepts should also perform better than the Joint .001 that instead obtains similar results. To catch a possible explanation for this phenomena it becomes necessary to explain the Joint model loss function.

$$\hat{f}, \hat{g} = \arg\min_{f,g} \sum_i [L_Y\big(f(g(x^{(i)})); y^{(i)}\big) + \sum_j \lambda L_{C_j}\big(g(x^{(i)}); c^{(i)}\big)]$$

The loss function jointly minimizes the class prediction loss $L_Y\big(f(g(x^{(i)})); y(i)$ and the the concept prediction loss $L_{C_j}(g(x^{(i)})); c(i)$. The λ value - to which before we referred to through "attr_loss_weight" - controls how much emphasis the optimization process places on concept prediction relative to the class prediction. Setting the early stopping to block the training after 5 epochs of not increasing accuracy may have been a too aggressive solution for this kinds of optimization that probably had no time to fully express the characteristics of the λ balancing. More epochs of training would have probably led the Joint .001 to sacrifice the accuracy on concepts for a better classification performance and the Joint .01 to manifest its potential on both tasks.

3.2 Final considerations

Koh et al. claim "little trade-off" between task and concept errors; through the analysis of their results and my replications it appears more appropriate to say that it is possible to build models that allow for high level concept explanations sacrificing some amount of accuracy. In any case this still represents a promising result. By analyzing the models' behavior in depth, I found that all bottleneck architectures — including the 0 concept supervision Standard Bottleneck — show a Spearman correlation of approximately 0.2, indicating good diversity among their concept representations. We should look to these results in the context of their training: Joint and Indipendent models are explicitly set up to catch different and most representative concepts for each class, then it is great having a value that moves toward the goal of these architectures. The similar r value obtained by the Standard model could be misleading: while it shows "diversity", these vectors are not representing human-aligned concepts. The result reflects the tendency of CNNs to leverage different features of the images in order to allow their final prediction. It is plausible that the 112 feature bottleneck, that in the "Standard fashion" is basically a last layer before regression, still learns relevant and distinct features that partially align with the ground truth concepts. Personally, the most exciting aspect of these architecture is how they show that it is actually feasible to obtain high level human-understandable concept predictions and use them to achieve a final classification. Citing Koh et al., it would be virtually possible to build algorithms that allow for high level concept intervention by non experts.

Models	Replica	Authors
Ind	75.56	76
Seq	75.63	76
Joint .01	80.66	81
Joint .001	81.04	83
Std	82.27	82.50

Table 2: Top1Class Accuracy

Models	Replica	Authors
Ind	.97	.97
Seq	.97	.97
Joint .01	.93	.97
Joint .001	.94	.85
Std	.5	.5

Table 3: Concepts Accuracy

3.3 Limitations and Future Directions

I acknowledge that performing more training runs on each model could have been beneficial to strengthen the results. Compared to the authors, I drastically decreased the number of epochs needed for early stopping. While some of the results do align with the original ones, the joint models might have expressed better performance with longer trainings due to their optimization characteristics. This essay evaluates the quality of concepts learned by different CBMs and relates the impact of the concepts on the final classification but doesn't explore the issue of Concept Leakage cited in this paper. Testing for it leveraging different tasks and datasets could be beneficial for better understanding the behavior of these models and may be a good starting point for future explorations.

References

[1] 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*, pages 5338–5348. PMLR, 2020.

[2] Jeffrey A Brooks, Lauren Kim, Michael Opara, Dacher Keltner, Xia Fang, Maria Monroy, Rebecca Corona, Panagiotis Tzirakis, Alice Baird, Jacob Metrick, et al. Deep learning reveals what facial expressions mean to people in different cultures. *Iscience*, 27(3), 2024.

[3] Netta Ollikka, Amro Kamal Mohamed Abbas, Andrea Perin, Markku Kilpeläinen, and Stephane Deny. A comparison between humans and ai at recognizing objects in unusual poses. *Transactions on Machine Learning Research*, 2024.

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

[5] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *California Institute of Technology*, 2011.

[6] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4109–4118, 2018.

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

A Appendix

A.1 Code

1. "data_processing.py" line 19 set "val_file" to "None" for redundancy solving

2. "generate_new_data.py" Line 178, weights_only set to False due to pytorch constraints

3. "dataset.py‘" line 192, implemented "num_workers=8" and "pin_memory=True" to ease the transfer of training data to GPU

4. "train.py‘" line 248, brought early stopping from 100 to 5

5. "inference.py‘" line 91, created "extracted_attributes" list. Line 118, added torch.cat to concatenate output tensors. Line 238 now extracts attributes and create a .csv file. Line 288, added directory creation if not existing.

6. To compute the Spearman correlation among the concepts I created "spearman.py".

   The code is available at: https://github.com/DiegoBelfiore1/MachineLearningII.git