# Concept Bottleneck Models - Machine Learning project

Niccolò Zenaro
niccolo.zenaro@studenti.unipd.it

February 2025

## Abstract

Nowadays state-of-the-art models are trained end-to-end to predict targets given raw inputs. We refer to them as black box models because they can't simply manipulate concepts as we intend them, for example "the presence of a forked tail that has the same color of the forehead", but instead they are trained to retain information (species of birds) from the inputs (pixels). What if we could feed and train a network with images and concepts, in order to obtain a classification refined on those concepts? Concept Bottleneck Models enable interpretation of the results allowing a richer human-model interaction because we can intervene on the predicted concept values and propagate these changes to the final prediction.

## 1 Introduction

In recent years, neural networks have been successful in tasks that could require human-level intelligence, such as understanding and generating images or text, or controlling robots to follow instructions. However, the decision-making process of these networks is not often *explainable*, causing problems in user trust in sensitive or critical environments such as medicine, finance, or automation. This occurs because state-of-the-art models nowadays are trained end-to-end to go directly from an input $x$ to a desired target or output $y$, and we cannot easily interact with them using our high-level concepts.

*Concept Bottleneck Models* approach this problem with a simple idea, first predicting an intermediate set of human-specified concepts $c$ and then using this to predict the target $y$. In fact concepts can be represented by a unique neuron in the bottleneck layer $f^c$ of a model $f$. These models are trained on data points *($x$, $c$, $y$)*, where $x$ is the input annotated with concept $c$ and target $y$. During testing, they take an input $x$, predict concepts $\hat{c}$ and then use those to predict a target $\hat{y}$. The main achievement of *CBM* is that we can intervene on those concepts $\hat{c}$, editing them directly, and then propagating those changes to the final prediction $\hat{y}$. This intervention enhances the model-human interaction and the interpretability of the methods, and is what explainable Artificial Intelligencence (*XAI*) is trying to achieve.

In this work I tried to build end-to-end models that simulate what the authors did in [6], obviously with a smaller capacity but anyway following the same concepts explained.

## 2 Related Work

### 2.1 Neuron-Level explanations

The smallest entity in a neural network that can represent a concept is a neuron, which could be also a *unit* or a *filter* in a convolutional neural network [1]. The *Network dissection approach* analyzes neuron representations by comparing activations to predefined concept masks. Then it uses *IoU* metrics to quantify whether a convolutional filter represents a specific concept.

### 2.2 Layer-Level explanations

As stated before, concepts can also be represented by a whole *layer*. If we use a pre-trained model, passing a set of concepts $\mathcal{C}$ and extracting the activation of a specific layer, we can train a concept classifier. This is done in

two approaches: *Concept Activation Vectors* [4] or *probing*. *CAV* is a continous vector that corresponds to a concept represented by a layer of a neural network $f$, and it allows to determine how much an image $x$ is correlated with a concept $C$ by measuring the probability of it to have a positive influence on predicting a class label $l \in \{1, ..., L\}$.

# 3 Dataset

The dataset used is the Caltech-UCSD Birds-200-2011 (CUB)[8], which has $n$ = 11,788 bird images. Each image contains $k$ = 312 binary attributes, the task is to classify correctly the bird among the 200 possible species. This is not an easy datasets to work on, because an image $x_i$ labeled as $y_m$, can have different concepts $c$ with respect to an image $x_j$ with the same label $y_m$, leading to poor classification results.

# 4 Methods

## 4.1 Concept Bottleneck Models

We can see *CBM*s as a composition of functions predicting a tartget $y \in \mathbb{R}^n$ from an input $x \in \mathbb{R}^m$, but obviously we can adapt this to a regression problem. Training data are presented as $\{x^i, y^i, c^i\}$, where $c \in \mathbb{R}^k$ is a vector of concepts of dimensionality $k$. A model of this type is described with a function $f(g(x))$, where the function $g : \mathbb{R}^m \to \mathbb{R}^k$ maps the input $x$ to a set of binary concepts $c$ and the function $f : \mathbb{R}^k \to \mathbb{R}^n$ maps the concept space to the final prediction. The name "*concept bottleneck*" derives from the fact that the prediction $\hat{y} = f(g(x))$ relies on the bottleneck, that is the prediction of concept $\hat{c}$ by the function $g(x)$, which is trained to align with true concepts $c$. Two losses are defined for the models, $\mathcal{L}_c$ measures the discrepancy between the predicted concept $\hat{c}$ and the true concept $c$, while $\mathcal{L}_y$ is the loss for the predicted and true target.

Like the authors did in the original paper [6], I implemented three different types of CBMs:

- The *independent model* learns $\hat{f}$ and $\hat{g}$ independently, $\hat{f} = \arg\min \sum_i \mathcal{L}_y(f(c^{(i)}); y^{(i)})$, and function $\hat{g} = \arg\min \sum_{i,j} \mathcal{L}_c(g_j(x^{(i)}); c_j^{(i)})$. This model has $\hat{f}$ trained using true $c$, but at test time takes as input $\hat{g}(x)$.
- The *sequential model* learns $\hat{g}$ as the *independent*, but uses the concept prediction $\hat{g}(x)$ to learn the new loss for $\hat{f} = \arg\min_f \sum_i \mathcal{L}_Y(f(\hat{g}(x^{(i)})); y^{(i)})$.
- The *joint model* learns a weighted sum for the losses of $\hat{f}, \hat{g}$, producing a joint total loss of type $\hat{f}, \hat{g} = \arg\min_{f,g} \sum_i [\mathcal{L}_y(f(c^{(i)}); y^{(i)}) + \sum_j \lambda \mathcal{L}_c(g_j(x^{(i)}); c_j^{(i)})]$ for some $\lambda > 0$.

## 4.2 Testing with intervention

The ability to intevene on predicted concepts $\hat{c}$ is what makes the difference for these models with respect to end-to-end ones, making them explainable and enhancing human-model interaction. Intervention during test-time is useful because it shows how much a concept can be useful to describe an input object; for example for bird images we can infer how much the concept "`wing color`" can affect the prediction of the bird species.

# 5 Experiments

## 5.1 Dataset preprocessing

CUB Dataset [8] is composed of $n = 11788$ images, each one is labeled among 200 possible labels that correspond to bird species. For each image we have 312 binary attributes or concepts that can be predicted. The dataset has been transformed in different ways among training set *Tr*, validation set *Vs* and testing set *Te*. *Ts* comprises the $75\%$ of the entire dataset, *Te* the $20\%$ and *Vs* the $5\%$; for *Ts* I applied several transformations with `torchvision.transforms`, not only resizing the image for the pre-trained model, but also *random cropping*, *random horizontal flip* and *random color jittering*; for the others *Vs, Te* only resizing has been applied.

## 5.2 The models

For the three types of *CBMs* described above, the authors of the original paper used Inception-V3 [7] for concepts prediction and a Multi-layer Perceptron for classification. With the same approach, I tryed to

emulate them but with different architectures; transfer learning was achieved with *ResNet-50* [3] trained on *IMAGENET*, fine-tuning the last layer for the concept prediction task. This was done on 15 *epochs* with a batch size of 32 elements, using `Binary Cross-Entropy with logits loss`, Adam [5] optimizer with `learning_rate` $= 0.0001$, and `weight_decay` $= 0.0005$ for regularization. The loss has also adjusted weights as defined in the original paper, because of the class imbalance for an individual concept that is roughly $1:9$ on average; this encourages the model to learn to predict positive concept labels that, as aforementioned, have a ratio of 1 over 9 with respect to negative concept labels.

The Multi-layer Perceptron has three *hidden layers* of `dim` $= [400, 500, 300]$ and was trained on 30 *epochs*, using *SGD Optimizer* with `learning_rate` $= 0.001$, `momentum` $= 0.9$, *Nesterov Momentum*[2] and `weight_decay` $= 0.0005$ for regularization. Being a multi-class classification problem, the loss used was `Cross Entropy Loss`. Early stopping prevents the model from overfitting the concept's dataset.

These are the base models used for building the three types of *CBMs*; *Joint* model sees also the addition of a $\lambda$ factor on loss calculation during training, that regulates the contribution of the loss from $\hat{g}$ predictions, i.e. generated by fine-tuned *ResNet50*.

### 5.3 Intervention during Testing

Concept groups are determined by having a common prefix in the file `attributes.txt`, creating a simple dictionary of concepts. During testing, we can tune the number of group concepts that will substitute the predicted concepts $\hat{c}$, ideally simulating what an expert can do. It's important to say that during the experiments the concepts are randomly selected, while an expert could decide the concepts to substitute based on its knoweledge.

## 6 Conclusions

*CBMs* can compete on task accuracy while supporting intervention and interpretation, enabling more human-model interaction. However, one of the main limitations of these models is the need for annotated concepts that cannot be available for specific tasks. If the set of concepts is good enough, then fewer training examples might be required to achieve a good accuracy level on task.

## References


[1] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations, 2017.

[2] Aleksandar Botev, Guy Lever, and David Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. *arXiv preprint arXiv:1706.08298*, 2017.

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

[4] 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*, pages 2668–2677. PMLR, 2018.

[5] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] 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.

[7] 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.

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