Machine Learning (2026) 115:97
https://doi.org/10.1007/s10994-026-06999-y

If Concept Bottlenecks are the Question, are Foundation Models the Answer?

Nicola Debole[1] · Pietro Barbiero[2] · Francesco Giannini[3] · Andrea Passerini[1] · Stefano Teso[1,4] · Emanuele Marconato[1]

Received: 16 April 2025 / Revised: 2 January 2026 / Accepted: 19 January 2026

Abstract
Concept Bottleneck Models (CBMs) are neural networks designed to conjoin high performance with *ante-hoc* interpretability. CBMs work by first mapping inputs (e.g., images) to high-level concepts (e.g., visible objects and their properties) and then use these to solve a downstream task (e.g., tagging or scoring an image) in an interpretable manner. Their performance and interpretability, however, *hinge on the quality of the concepts they learn*. The go-to strategy for ensuring good quality concepts is to leverage expert annotations, which are expensive to collect and seldom available in applications. Researchers have recently addressed this issue by introducing "VLM-CBM" architectures that replace manual annotations with weak supervision from foundation models. It is however unclear what the impact of doing so is on the quality of the learned concepts. To answer this question, we put state-of-the-art VLM-CBMs to the test, analyzing their learned concepts empirically using a selection of significant metrics. Our results show that, depending on the task, VLM supervision can noticeably differ from expert annotations, and that concept accuracy and quality are not strongly correlated. Our code is available at https://github.com/debryu/CQA.

Keywords Explainable AI · Concept Bottleneck models · Foundation models

1 Introduction

Concept Bottleneck Models (CBMs) (Koh et al., 2020) are a popular class of neural networks that aim to resolve the traditional trade-off between interpretability and accuracy. In a nutshell, a CBM comprises two learnable modules: a *concept extractor* and an *inference layer*. The former maps the input into an activation vector of high-level concepts, while the latter is generally a white-box model, typically a (sparse) linear layer, that computes predictions from the extracted concepts. For instance, given the image of a red ball,

Stefano Teso and Emanuele Marconato are shared last author.

Editors: Riccardo Guidotti, Anna Monreale, Dino Pedreschi.

Extended author information available on the last page of the article

a CBM would encode it into a vector of concept activations – typically logits – for the events ("$shape'' = \text{sphere}$), ("$color'' = \text{red}$), and all other combinations established by a pre-defined vocabulary. This vector plays the role of a ***bottleneck*** encoding interpretable sufficient statistics for the final prediction. CBMs are *explainable-by-design* in the sense that they allow tracing any decision back to the concepts most relevant for it. At the same time, they support representation learning and can achieve high performance for complex data (e.g., images) that are beyond the reach of traditional white-box models. Their modular architecture also brings other benefits, such as support for *correcting* the model's predictions via user interventions (Koh et al., 2020; Espinosa Zarlenga et al., 2024; Shin et al., 2023; Steinmann et al., 2024) and for *debugging* the model itself using explanation-based feedback (Stammer et al., 2021; Bontempelli et al., 2023). For these reasons, the same two-step setup has been instantiated into a number of recent classifiers (Sawada & Nakamura, 2022a, b; Barbiero et al., 2023; Marconato et al., 2022; Kim et al., 2023; Vandenhirtz et al., 2024) and generative models (Ismail et al., 2023; Dominici et al., 2024).

It is however possible that the ***learned concepts may not match the semantics that humans attribute to them***. The more they deviate, the more opaque the model's decision process becomes, compromising all benefits that CBMs are designed to provide, including interpretability and intervenability (Mahinpei et al., 2021; Marconato et al., 2023; Furby et al., 2023; Raman et al., 2023; Lai et al., 2024).

The go-to strategy for aligning concepts to human semantics is to exploit manual concept annotations, however these are expensive to collect and thus often unavailable in applications. An increasingly prominent solution is to replace manual annotations with *weak supervision* obtained by querying Vision-Language Models (VLMs) with natural language descriptions of the desired concepts (Oikarinen et al., 2023; Yang et al., 2023; Srivastava et al., 2024; Rao et al., 2024). In essence, this procedure distills the concepts learned by a source VLM into a target CBM (cf. Figure 1), with the explicit goal of extending the reach of CBMs to tasks in which no manual annotations are available. The resulting ***VLM-CBMs*** offer competitive prediction performance and in many cases admit fast inference.[1]

VLMs, however, may be no silver bullet. In fact, although trained on very large corpora, they can suffer from hallucinations (Huang et al., 2023), shortcuts (Yuan et al., 2024), and subpar logical (Calanzone et al., 2025) and conceptual (Sahu et al., 2022) consistency, and

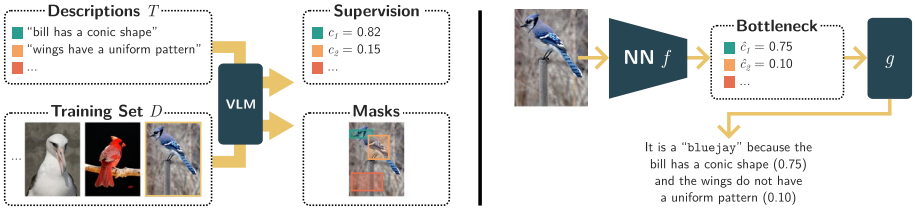


Fig. 1 A prototypical VLM-CBM. **Left**: given textual descriptions $\mathcal{T}$ of visual concepts, a VLM is used to annotate a training set $\mathcal{D}$ with per-concept labels or activation scores and (optionally) masks indicating where each concept activates. **Right**: the supervision is typically used to fine-tune a backbone f that extracts concepts from new inputs and a linear layer g for inferring predictions from the concepts. See Sect. 2 for architecture-specific differences

[1] This does not apply to VLM-CBMs that invoke the VLM during inference, see Sect. 2.

the distillation step risks to pass these defects onto the target CBM. Moreover, even highly accurate concepts can capture spurious contextual information – due to, e.g., *entanglement* (Higgins et al., 2018; Schölkopf et al., 2021; Suter et al., 2019; Eastwood & Williams, 2018) and *leakage* (Margeloiu et al., 2021; Mahinpei et al., 2021; Marconato et al., 2022; Havasi et al., 2022), illustrated in Fig. 2 – in which case the model's explanations become misleading. Unfortunately, most works on VLM-CBMs are chiefly concerned with downstream accuracy (Oikarinen et al., 2023; Yang et al., 2023; Srivastava et al., 2024; Rao et al., 2024; Yuksekgonul et al., 2023) rather than quality of the learned concepts. As a result, it is still unclear whether VLM-CBMs learn concepts of high enough quality.

We fill this gap by directly assessing the concepts learned by representative VLM-CBM architectures – namely, language-in-a-bottle (Yang et al., 2023), label-free CBMs (Oikarinen et al., 2023), and vision-language-guided CBMs (Srivastava et al., 2024); see Table 1 for a breakdown of different architectures – using a variety of metrics focusing on diverse, relevant aspects of concept quality. Our experiments on three datasets of increasing complexity indicate that: (*i*) *VLM concept supervision can differ from manual annotations*, even when textual descriptions $\mathcal{T}$ of concepts are provided by experts; (*ii*) in many cases *VLM-CBMs can output correct label predictions by leveraging low-quality concepts* and, more generally, prediction performance and concept quality are not strongly related. Overall, our results show that, while VLM-CBMs enjoy wider applicability compared to regular CBMs, ***replacing manual with VLM supervision can come at a substantial cost for concept quality***, and thus interpretability. With this work, we hope to prompt more stringent evaluation of concepts learned by state-of-the-art CBMs and to refocus the design of VLM-CBMs on robustness to annotation mistakes and interpretability.

2 The Family of Concept Bottleneck Models

Concept Bottleneck Models (CBMs) (Koh et al., 2020) combine two neural modules: a *concept extractor* $f : \mathcal{X} \to \mathbb{R}^k$ mapping an input $\mathrm{x} \in \mathcal{X}$ to a concept activation vector $\hat{\mathrm{c}} \in \mathbb{R}^k$, which encodes logits of k high-level concepts describing the input; and a white-box *infer-*

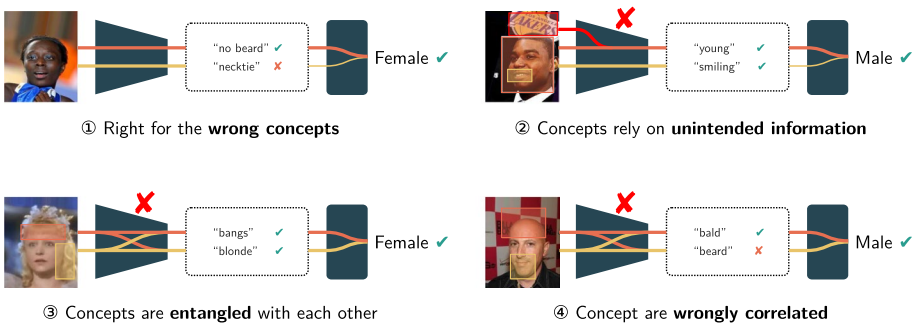


Fig. 2 Issues with CBM concepts. An illustration of the four issues with concept quality and usage affecting CBMs on the `CelebA` task (cf. Sect. 4): (1) Attaining high label accuracy does not prevent learning ***inaccurate concepts***. (2) CBMs can maximize label performance by learning ***leaky concepts*** that include irrelevant contextual cues. (3) ***Entangled concepts*** encode unwanted information about one another, affecting out-of-distribution behavior. (4) ***Impure concepts*** encode unwanted correlations that do not exist among ground-truth concepts

Table 1 Comparison of CBMs and VLM-CBMs

Model	VLM	Concept Sup.	External $\mathcal{T}$	Pretrained	Steps
`CBM` (Koh et al., 2020)	–	Labels	✓*	✗**	3
`LaBo` (Yang et al., 2023)	`CLIP` (Radford et al., 2021)	Similarity	✓	✓	1,3
`LF-CBM` (Oikarinen et al., 2023)	`CLIP` (Radford et al., 2021)	Similarity	✓	✗	1,2,3
`VLG-CBM` (Srivastava et al., 2024)	`G-DINO` (Liu et al., 2024)	Labels, Bboxes	✓	✗	1,2,3
`PH-CBM` (Yuksekgonul et al., 2023)	`CLIP` (Radford et al., 2021)	Similarity	✓	✓	1,3
`DN-CBM` (Rao et al., 2024)	`CLIP` (Radford et al., 2021)	Similarity	✗	✓	1~,3
`Naïve`	`LLaVa` (Contributors, 2023)	Labels	✓	✗	1,2,3

The columns which VLM is used, which kind of concept supervision (Concept Sup.) is required, whether the textual concept descriptions (that is, $\mathcal{T}$) can be supplied externally, and whether the concept extractor requires a pretrained model (Pretrained). To the right, we indicate which of the steps from Sect. 2 are specifically included in the models' training pipeline (Steps)

* The concept vocabulary can be customized in standard CBM too, but this comes at the cost of having to collect manual annotations for any new concepts

** Some variants of CBMs (Dominici et al., 2024; Laguna et al., 2024; Sawada & Nakamura, 2022b) require using pretrained modules

~ The VLM is used only to assign a name to each concept

ence layer $g : \mathbb{R}^k \to \mathcal{Y}$ mapping the concept activations to a prediction $\hat{y} \in \mathcal{Y}$. Combining f and g yields the complete predictor. Despite not being restricted to specific data or tasks, CBMs are often used for image classification. This is the use case we focus on.

While the concept extractor is an expressive (but black-box) neural backbone – enabling CBMs to achieve competitive *label* accuracy – the inference layer is white-box. This means that CBMs can naturally explain their own inferences, in that it is easy to identify what concepts contributed the most to each prediction. For instance, if the inference step is implemented – as customary – with a sparse linear layer, then a CBM *explanation* elucidates to what extent each concept activation $\hat{c}_j$ contributes to a prediction $\hat{y}$ by considering the weights $w_{\hat{y},j}$ of the linear layer (Kim et al., 2018). See Fig. 1 (right) for an example.

A major benefit of this setup is *intervenability*: whenever concept are mis-predicted, users can readily replace them with better estimates (Koh et al., 2020; Shin et al., 2023; Espinosa Zarlenga et al., 2024; Chauhan et al., 2023). It is also possible to teach CBMs to reapply past interventions to unseen instances (Steinmann et al., 2024) or to learn when interventions are required at training time (Espinosa Zarlenga et al., 2023), thus implement-

ing a form of interactive debugging (Teso et al., 2023). CBM architectures differ in how they define, extract and use their concepts, yet popular variants like Concept Embedding Models (Zarlenga et al., 2022) and GlanceNets (Marconato et al., 2022) all follow the same modular setup. We leave an in-depth discussion of other concept-based models to Sect. 5.

Learning with expert supervision CBM explanations are interpretable only as long as the concepts are *aligned* with the semantics that humans associate to them (Marconato et al., 2023; Fokkema et al., 2025). Most CBMs exploit dense concept-level annotations during training to ensure that this is the case. They typically do so by combining two cross-entropy losses, one for the labels $\mathcal{L}_Y = \mathbb{E}_{(\mathrm{x},y)}\big[-\log p(y \mid \mathrm{x})\big]$ and one for the concepts $\mathcal{L}_\mathrm{C} = \mathbb{E}_{(\mathrm{x},\mathrm{c})}\big[-\frac{1}{k}\sum_{j\in[k]}\log p(c_j \mid \mathrm{x})\big]$, although the exact formulations vary between architectures (Koh et al., 2020; Zarlenga et al., 2022; Marconato et al., 2022). Here, $p(c_j \mid \mathrm{x}) = \mathrm{sigmoid}(f(\mathrm{x})_j)$ is the predictive distribution of the j-th concept and the expectations run over the training data. The two modules can be trained either independently, sequentially or jointly (Koh et al., 2020), and the choice can impact the learned representations (Mahinpei et al., 2021). Most architectures discussed below follow a sequential setup: the concept extractor is learned first, by minimizing $\mathcal{L}_\mathrm{C}$, and it is then used to train the inference layer via $\mathcal{L}_Y$. Note also that, in most complex tasks, the concept extractor is pretrained and fine-tuned, rather than trained from scratch.

Learning with VLM supervision A key issue of CBMs is that concept annotations are *not* available in most applications. In these cases, $\mathcal{L}_\mathrm{C}$ cannot be used during learning, and weak supervision coming from the label annotations via $\mathcal{L}_\mathrm{Y}$ can be insufficient to prevent concepts from deviating, even radically, from the ground-truth ones (Bortolotti et al., 2025). In an attempt to widen their applicability, researchers have recently proposed several "VLM-CBMs" that forego expert annotations in favor of a pre-trained Vision-Language Model (VLM). The idea is that, when asked to detect a set of preselected concepts, identified by textual descriptions $\mathcal{T} = \{t_1, \ldots, t_k\}$, in any image, a sufficiently large VLM trained on a wealth of diverse data will likely produce a high-quality response. Next, we introduce the key steps involved in learning a VLM-CBM (Fig. 1).

Step 1: Obtaining concept supervision from the VLM. A popular option is to obtain concept activations by matching textual concept descriptions and training images in the VLM's *embedding space*. For instance, Label-free CBMs (`LF-CBM` s) (Oikarinen et al., 2023), Language-in-a-Bottle (`LaBo`) (Yang et al., 2023) and Post-hoc CBMs (`PH-CBM`) (Yuksekgonul et al., 2023) use CLIP's visual encoder E_V and text encoder E_T (Radford et al., 2021) to compute (variants of) the cosine similarity $c_j = \mathrm{sim}(E_V(\mathrm{x}), E_T(t_j))$ between image x and text t_j.[2]

A more direct route is to collect binary annotations c_j by *prompting* the VLM: one simply feeds a training input x to the VLM and asks whether a concept described by t_j is present. Vision-Language-Guided CBMs (`VLG-CBM` s) (Srivastava et al., 2024) do so with Grounding Dino (Liu et al., 2024). As a bonus, they also collect per-concept bounding-boxes and use them to filter out low-quality hits. In either case, one obtains similarity scores or binary labels $\mathrm{c} \in \mathbb{R}^k$ for each data point.

Finally, Discover-then-Name (`DN-CBM`) (Rao et al., 2024) stands out in that it extracts concept activations using sparse autoencoders. Since the extracted concepts are anonymous, they are assigned a name by decoding the image representation to text using CLIP. One downside is that this makes it impossible to prespecify the vocabulary $\mathcal{T}$.

[2] We use the same notation for both scores and labels, for simplicity.

Step 2: Training the concept extractor with VLM supervision. `LF-CBM` updates the backbone using a modified concept loss $\mathcal{L}_{\mathrm{C}}$ that regresses on the similarity scores, while `VLG-CBM` uses the concept annotations to power the cross-entropy loss $\mathcal{L}_{\mathrm{C}}$. It also optionally augments the training set by adding cut-outs of the input image identified by the concepts' bounding boxes, so as to encourage learned concepts to be independent of their background. `LaBo` and `PH-CBM` are special as they have no backbone[3] and apply the inference layer directly to the similarity scores.[4]

Step 3: Training the top linear layer. Once the concepts have been learned, the top layer is trained by optimizing $\mathcal{L}_{\mathrm{Y}}$. The size of concept explanations is kept under control by incorporating a sparsity penalty – such as ℓ_1-regularization or group lasso (Oikarinen et al., 2023; Srivastava et al., 2024) – or by normalizing the weights (Yang et al., 2023).

Issues with VLM-CBMs VLM-CBMs extend the reach of CBMs to tasks that lack dense annotations, yet they are not devoid of risks in that their concepts may not always be high quality. For instance, some concepts learned by popular VLM-CBMs can be irrelevant and/or non-visual (cf. Table 5) and still affect the model's decision making (Srivastava et al., 2024; Yang et al., 2023). The VLMs themselves are likely imperfect. It is well known that CLIP can exhibit erroneous agreements when using cosine similarity (Li et al., 2024), and more generally that foundation models are subject to hallucinations (Huang et al., 2023) and shortcut learning (Yuan et al., 2024). Ultimately, *it is unclear how VLM-provided supervision affects the learned concepts*. This is precisely what our study aims to find.

The concept vocabulary Before proceeding, we briefly mention another important aspect of VLM-CBMs. While most of these models can work with any prespecified vocabulary $\mathcal{T}$, recent works suggest to generate it automatically by consulting a Large Language Model (LLM), such as ChatGPT (Hurst et al., 2024). The LLM is simply asked to describe concepts that it deems useful for identifying specific classes (Oikarinen et al., 2023; Yang et al., 2023; Srivastava et al., 2024; Feng et al., 2024). Even filtering out low-quality answers, there is still a chance that invalid concepts may crop up in the vocabulary, as shown in (Yang et al., 2023) and in A.3.1. Since we aim to assess the impact of VLM-supplied concepts, *we use gold standard annotations as reference*. Therefore, in our experiments we use the corresponding (gold standard) vocabulary for all VLM-CBMs, excluding `DN-CBM`s in the process as they do not support this mode of operation.

3 Metrics for Concept Quality

We aim to assess to what extent VLM-CBM learn high-quality concepts. Establishing formal conditions for concepts to be deemed interpretable is a difficult open problem (Marconato et al., 2023; Zarlenga et al., 2023; Barbiero et al., 2025) and beyond the scope of our paper. We take a different route and ask whether the learned concepts satisfy a minimal set of reasonable desiderata, namely they are accurate (Sect. 3.1), avoid leakage (Sect. 3.2), are disentangled (Sect. 3.3) and avoid unwanted correlations (Sect. 3.4).

[3] When viewed as VLM-CBMs, they are in fact equivalent.

[4] The downside is the non-negligible runtime cost of running CLIP at inference time.

3.1 Accurate Concept Predictions

The most natural requirement is that the learned concepts are ***accurate*** (Koh et al., 2020), i.e., that they match their ground-truth values, irrespective of whether they were learned from expert or VLM supervision. Inaccurate concepts can still lead to correct decision-making, but they undermine the credibility of the CBMs' explanations. For instance, if a VLM-CBM wrongly recognizes the presence of a *"necktie"* and yet uses it to distinguish between male and female celebrities, cf. Figure 2 (1), it leads to poor explanations of the decision-making. This issue is well known, and indeed, poor concept predictions are the primary target for debugging, which can be done by manually labeling or intervening on concepts for misclassified instances (Koh et al., 2020; Steinmann et al., 2024; Espinosa Zarlenga et al., 2024).

While most works on VLM-CBMs measure accuracy (Oikarinen et al., 2023; Yang et al., 2023; Srivastava et al., 2024; Rao et al., 2024), this is unsuitable for unbalanced concepts (that, e.g., occur rarely) and restricted to binary concepts. However, as mentioned in Sect. 2, different architectures use different concept supervision, from binary labels (`VLG-CBM`) to similarity scores (`LF-CBM` and `LaBo`). Hence, we assess concept predictions by measuring the Area under the ROC curve, denoted $\mathrm{AUC}(C)$, averaged over all concepts in the bottleneck. An additional benefit is that the AUC is robust to unbalanced concept classes, while the accuracy is not.

3.2 Avoiding Concept Leakage

A natural desideratum is that each learned concept does not carry spurious information about the label compared to its ground-truth counterpart, as this hinders interpretability. Prior work has shown that (even accurate) concepts can suffer from ***leakage***,[5] i.e., they may unintentionally carry information that facilitates label predictions but is semantically incongruent with the concept itself (Mahinpei et al., 2021; Margeloiu et al., 2021; Lockhart et al., 2022; Havasi et al., 2022; Marconato et al., 2022). For instance, in Fig. 2 (2) the model maximizes the score of the correct ("*male*") class by encoding the name of a basketball team within the concept *"young"*.[6]

Unfortunately, leakage cannot be easily detected by monitoring model accuracy alone. Prior works (Mahinpei et al., 2021; Lockhart et al., 2022; Marconato et al., 2022; Havasi et al., 2022) suggest measuring the retained amount of label accuracy when restricting the bottleneck to concepts that are known to be *irrelevant* for the prediction task. Doing so requires knowing what concepts ought to be irrelevant and this information is not available in all tasks. Other metrics evaluate leakage by considering the sparsity of weights learned by the inference layer Srivastava et al. (2024), which depends on empirical factors (such as training regime of the inference layer) that are orthogonal to concept leakage. We solve both issues by developing a new metric that estimates concept leakage independently of

[5] We are interested in leakage as it can be detrimental to interpretability Mahinpei et al. (2021), however, it is still not clear if it should be avoided at all costs. For instance, in certain settings, leakage has been demonstrated to enhance the model's robustness to interventions at test time Espinosa Zarlenga et al. (2023).

[6] Our example considers a single concept mistake, for simplicity. We stress that, however, while individual mistakes can *sometimes* help inferring the right label (as in our example), concept mistakes only count as leakage when they *systematically* improve label performance (over what is achievable with the ground-truth concepts) over the whole dataset.

the inference layer. It does so by measuring the gap in label performance between the best achievable inference layers fed on predicted vs. ground-truth concepts. The algorithm works as follows. First, we sort all concepts by their *ground-truth* Pearson correlation with the task label y, least correlated first. Second, we train two linear SVMs (Cortes & Vapnik, 1995) to predict the ground-truth label y from the first ℓ concepts, one feeding on the ground-truth concept labels c and the other on the predicted concepts $\hat{\mathrm{c}}$, and measure the gap in their F_1 -scores, i.e.,

$$\mathrm{gap}_\ell = F_1^{CBM}(Y;\ell) - F_1^{GT}(Y;\ell) \tag{1}$$

Intuitively, a linear classifier predicting the label from the first $\ell \ll k$ (i.e., least correlated) concepts should perform very poorly; however, *this is not the case if they leak label information, e.g., when being spuriously correlated with other concepts that are informative of the label, or by subsuming non-conceptual stylistic factors that contain predictive information of the labels*. We repeat this step for $\ell = 1, \ldots, k$ and then record the average of all gaps:

$$\mathrm{LEAK} = \frac{1}{k \cdot Z} \sum_{\ell=1}^{k} \max(\mathrm{gap}_\ell, 0) \tag{2}$$

here, $Z = 1 - \sum_{\ell=1}^{k} F_1^{GT}(Y;\ell)/k$ is a normalizing constant that ensures LEAK has a maximum value equal to 1. LEAK is zero if and only if all performance gaps are equal to or less than zero, i.e., the learned concepts provide the same or less information about the label as the ground-truth ones. Vice versa, a value close to 1 indicates that the learned concepts leak task-relevant information.

Additional details, including pseudo-code, are provided in Appendix C.3. We also performed a sanity check in Appendix C.4: our tests indicate that the leakage metric behaves as intended. Specifically, noisy or random model concepts do not improve prediction accuracy on the downstream task. Therefore, any systematic gain in predictive accuracy over all *test* set is indicative of information leakage. We stress, however, that a model producing low-accuracy labels will naturally exhibit smaller (or even negative) gaps with respect to the use of ground-truth concepts, and therefore lower (or potentially no) leakage as measured by our metric. Our results in Sect. 4 have to be interpreted in light of this observation.

3.3 Disentanglement of Concepts

Leakage occurs when concepts unintentionally encode spurious task-relevant information. In some cases, this happens because they encode information about one another, as doing so can improve label accuracy. Therefore, a key requirement is that learned concepts are ***disentangled*** (Kazhdan et al., 2021; Marconato et al., 2023), i.e., manipulating the input to alter one concept should not affect the prediction of the others (Suter et al., 2019). E.g., in Fig. 2 (3) the presence of *"bang"* is entangled with the *"hair color"*. Concepts that mix irrelevant factors are problematic as they do not match the semantics of their ground-truth counterpart, complicating interpretation (Marconato et al., 2023). For example, if the concept *"bang"* contains information about the *"hair color"*, it can behave unpredictably when this is switched on or off in the data (Montero et al., 2022). Disentanglement is also a

prerequisite for concept reuse, generalization, and supporting post-hoc alignment with users (Bortolotti et al., 2025; Fokkema et al., 2025).

We measure disentanglement with DCI, a popular metric that estimates the inter-relations between predicted and ground-truth concepts (Eastwood & Williams, 2018). It does so by training a non-linear model, e.g., a random forest, predicting the ground-truth concepts from the learned ones and then computing relevance scores R_{ij} that reflect how predictive the *i*-th learned concept is for the *j*-th ground-truth concept. The DCI disentanglement score D_i for the concept c_i is then defined as:

$$D_i := 1 - \big(- \sum_{j=1}^{k} P_{ij} \log_k P_{ij} \big) \tag{3}$$

where $P_{ij} = R_{ij} / \sum_{\ell} R_{i\ell}$. The degree of disentanglement of the model is then given by computing a weighted average of all disentanglement scores, see Appendix C.2. DIS has a maximum score of 1 and occurs when concepts are completely disentangled; instead, the score is 0 (worst case) if learned concepts equally depend on all ground-truth ones.

3.4 Unwanted Correlations Among Concepts

When ground-truth concepts are not statistically independent, a key desideratum is that the learned concepts reflect the same correlations in the ground-truth concepts: if correlation among learned concepts exceeds the one in the data, the CBM may incur situations where *"beard"* is predicted because of the absence of hair, as portrayed in Fig. 2 (4). Correlation mismatch may arise in the learned space due to the model exploiting indirect statistical shortcuts, which are not necessarily present in the original ground-truth concept correlations. This can impact the model prediction in out-of-distribution data, e.g., when changes in *"bald"* lead to an undesired effect on *"beard"*, as well as to the responsiveness to manual intervention on concepts (Zarlenga et al., 2023), e.g., when the classifier leverages this unwanted correlation to predict correct labels, but does not respond to user manipulations as intended. If inter-concept correlations decrease, concept predictions may not match the ground-truth. For example, assume that the concepts *"makeup"* and *"lipstick"* are highly correlated in the data, but the corresponding learned concepts reveal a much weaker correlation. This can happen when at least one concept does not match the ground-truth value, e.g., *"lipstick"* is not always accurate, causing a significant difference in co-occurrences and in overall incorrect concept predictions. In this context, if final class labels depends on both ground-truth concepts being active, the CBM may learn to use only the concept *"makeup"* correctly activating to solve the task, while not depending of whether the *"lipstick"* concept is present in the input. This eventually leads to implausible explanations of the decision-making process.

The Oracle Impurity Score (OIS) (Zarlenga et al., 2023) measures this "impurity" by penalizing whether unwanted correlations appear among the learned concepts. It does so by comparing the inter concept relations among ground-truth concepts with the inter concept relations among learned ones. Similar to DCI, a non-linear model, e.g., a random forest, is trained to predict the same ground-truth concepts from ground-truth concepts, obtaining a relevance matrix $R_{ij}^{(gt)}$. The same procedure is done by training a model from the same fam-

ily to predict the ground-truth concepts from the model ones. This results to the same relevance matrix R_{ij} accounted in DCI. The divergence between these two matrices serves as an impurity metric, quantifying the amount of undesired information in the learned concepts:

$$\text{OIS} = \frac{2}{k} \sum_{ij=1}^{k} (R_{ij} - R_{ij}^{(gt)})^2 \tag{4}$$

An OIS of 1 indicates a complete misalignment between ground-truth and learned concepts, i.e., the *i*-th concept representation can predict one or many other concepts, except the ones it supposed even when the concepts are independent, while an OIS of 0 denotes perfect alignment, meaning that the *i*-th learned concept does not capture any unwanted correlation.

Relationships between Metrics We point out that some of the metrics may share mutual dependencies; however, the clear relationship between them is not yet known. While concept accuracy is not a strong indicator of concept quality – in the sense that even if it is perfect this does not rule out leakage (Mahinpei et al., 2021; Margeloiu et al., 2021), entanglement (Marconato et al., 2022; Havasi et al., 2022), or impure correlations (Zarlenga et al., 2023) – the amount of leakage does partly depend on whether concepts are entangled, and partly on whether they capture non-conceptual stylistic information (Marconato et al., 2023).[7] The two differ as leakage assesses whether model concepts encode spurious label-relevant information. Disentanglement, in contrast, focuses on dependencies between the modeled concepts themselves, regardless of any outside influence. Entanglement may occur without leakage, and vice versa: for instance, if a concept (e.g., “blurry") unintentionally captures stylistic cues predictive of the label, leakage may increase even if the concepts remain disentangled.

Finally, it is important to note that a reliable measurement of disentanglement assumes the availability of data with many independent variations across ground-truth factors. Disentanglement, as measured with DCI, decreases with the degree to which learned concepts contain information about other concepts. Nonetheless, in practice, learned concepts can be naturally correlated (Zarlenga et al., 2023). When this happens, complementary tools such as OIS (Zarlenga et al., 2023) can be used. By construction, OIS and DIS depend on the importance matrix of learned concepts; however, they do not measure the same quantity and their combination can provide useful insights. DIS provides a global measure of inter-concept dependencies but it naturally assumes that all ground-truth concept variations are observed to give a sound estimate. On the other hand, OIS is more sensible to lack of full variations and measures impurities in the representations compared to the ground-truth. Moreover, OIS can spot unwanted correlations in the learned concepts even when DIS is good.[8]

In practice, especially in difficult classification tasks, the concept extractor may optimize for accuracy by exploiting wrongly correlated, entangled, or leaky concepts, precisely because doing so maximizes the amount of task-relevant information in the bottleneck (Zhang et al., 2024). Still, we empirically evaluate all four metrics so as to provide a fuller

[7] Specifically, concepts that are both disentangled and independent from style entail no leakage, but the converse may not hold: if a concept c_i is *non-linearly* entangled into another c_j (according to Eq. 3), a *linear* layer may unable to extract useful information carried about the label by c_i from c_j, thereby leading to zero leakage.

[8] An example of this behavior is illustrated in (Zarlenga et al., 2023).

picture of VLM-CBMs concepts. To ground the discussion, in Sect. 4 we estimate and analyze the correlations between these metrics across models and datasets. Our analysis reveals that leakage is not correlated with either disentanglement or OIS, but shows that concept accuracy, disentanglement, and OIS are all strongly correlated. Despite that, the fact that these metrics are not perfectly correlated indicates that they all measure slightly different aspects of the learned concepts.

4 Empirical Analysis

We tackle empirically the following research questions:

Q1 Are expert and VLMs annotations comparable?

Q2 Do VLM-CBMs attain both high label accuracy and high concept accuracy?

Q3 Are VLM-CBM concepts leak-proof, disentangled, and pure?

Data sets We consider three concept-annotated data sets for image classification. Shapes3d Kim and Mnih (2018) is a synthetic data set of renderings of 3d objects with varying shape, color, orientation and background, for a total of 42 different binary concepts. The ground-truth binary label $y \in \{0, 1\}$ indicates whether a specific object (a *red pill*) is present in the image. In total, we use 48k samples for training, 5k for validation, and 30k for testing. CelebALiu et al. (2015) contains about 200k images (of size $178 \times 218 \times 3$) of celebrity faces and provides manual annotations for 40 binary concepts (hair color, presence of glasses, whether the person is smiling, *etc*.). The model is asked to predict the "gender" of the celebrity from the remaining 39 concepts. We selected 25k examples for training, 5k for validation, and 20k for testing. CUBWah et al. (2011) is a bird classification task spanning 200 different classes and 112 concepts (bill shape, body color, feather pattern, *etc*.). It consists of 4.8k images for training, 1.2k for validation, and 5.8k for testing.

Architectures We evaluate three representative VLM-CBMs, namely LaBo (Yang et al., 2023), LF-CBM (Oikarinen et al., 2023), and VLG-CBM (Srivastava et al., 2024). As baselines, we consider a standard CBM (Koh et al., 2020) and Naïve, a CBM trained on concept labels gathered from a recent VLM balancing performance and efficiency. Specifically, `Naïve` first queries LLAVA-Phi3 (Contributors, 2023), for each training image, about whether each concept in $\mathcal{T}$ is present or not, and then fits a sparse linear layer on the resulting binary answers with a cross-entropy loss; see Appendix B.1 for details.

For all models except `LaBo`, which has no backbone, we implement the image encoder using CLIP (Radford et al., 2021) (`ViT-B16` and `RN50` versions) or ResNet-18(He et al., 2016) in both `Shapes3d` and `CelebA` (the former for `LF-CBM` and `VLG-CBM`, the latter for `CBM` and `Naïve`). Following (Oikarinen et al., 2023; Srivastava et al., 2024), in `CUB` we use a ResNet-18 version pre-trained on CUB instead. As we will show, this seemingly innocuous choice noticeably affects both label and concept quality. In all cases, we implement the inference layer as a linear layer feeding on concept logits. Following the original implementations, `LF-CBM` and `VLG-CBM` incentivize sparsity by using `GLM-SAGA` (Wong et al., 2021), while `CBM` and `Naïve` minimize the unregularized cross entropy loss and `LaBo` regularizes the linear layer by applying a softmax operator to the weight rows (Yang et al., 2023). All details regarding architectures and hyperparameter tuning are reported in Appendices B.2 and B.3.

Concept vocabularies

All VLM-CBMs allow generating a task-specific concept vocabulary by querying an LLM (see Sect. 2). This procedure may yield invalid concepts (Srivastava et al., 2024) and more generally complicates comparing against gold-standard concept annotations. To ensure a fair evaluation, in each dataset we fix the same vocabulary $\mathcal{T}$ for all competitors. For comparison, we report the size of concept dictionaries from the original implementation of `CBM`, `LaBo`, `LF-CBM` and `VLG-CBM` in Appendix A.3.2, while all our concept dictionaries can be found in Appendix A.3.3. The generation process is in Fig. 5.

Experimental setup We perform two separate evaluations: one to assess the performance of VLMs as annotators (results in Table 2) and a second one to evaluate VLM-CBMs through metrics from Sect. 3 (results in Table 3). A sketch of the entire evaluation process can be found in Appendix A.1.

4.1 Q1: VLMs Supervision Does Not Match Expert Annotation

We begin by assessing the quality of the VLM annotations. Since `G-DINO` and `LLaVa` return hard binary annotations, while `CLIP` returns a numerical similarity score, we assess annotation quality in terms of average macro precision and recall (see Appendix A.2 for more details) with respect to the ground-truth concept annotations.[9] Results of this evaluation are reported in Table 2. Overall, *annotation quality differs substantially across VLMs and data sets*. The best annotations are obtained on `CUB` (112 concepts), with most VLMs achieving at least 53% precision and 58% recall – in fact, `G-DINO` attains high precision and recall above 90% and `LLaVa` 92% precision above and medium recall (about 66%). VLM annotations in `CelebA` (39 concepts) are noticeably worse – the best option being `LLaVa`, with precision and recall above 67% – and very poor in `Shapes3d` (42 concepts): `CLIP`, the best VLM, achieves precision and recall barely above 30% and suffers from high variance.

We hypothesize that annotation quality depends on a combination of factors: (*i*) Whether the VLM has encountered concepts closely matching the queried textual description during training (e.g., hair colors, eyeglasses).[10] (*ii*) Whether the input is in-distribution. E.g., VLMs may struggle to annotate `Shapes3d` precisely because its synthetic images fall out-of-distribution. It is in fact unlikely that VLMs would have trouble identifying concepts such as colors or shapes that likely appear in their training data. While bigger VLM, such as Llama-vision 3.2 (Grattafiori et al., 2024), might yield better annotations, they are not typically used for VLM-CBMs as they substantially increase computational costs. We leave

Table 2 Agreement between ground-truth and VLMs annotations in terms of (macro) precision and recall, averaged across all concepts and examples

	`Shapes3d`		`CelebA`		`CUB200`	
	MPr(↑)	MRc(↑)	MPr(↑)	MRc(↑)	MPr(↑)	MRc(↑)
`CLIP`	0.32 ± 0.23	0.30 ± 0.23	0.58 ± 0.08	0.65 ± 0.08	0.53 ± 0.04	0.58 ± 0.04
`G-DINO`	0.18 ± 0.07	0.64 ± 0.07	0.56 ± 0.11	0.54 ± 0.11	0.97 ± 0.07	0.90 ± 0.07
`LLaVa`	0.13 ± 0.06	0.13 ± 0.06	0.69 ± 0.13	0.67 ± 0.13	0.92 ± 0.08	0.66 ± 0.08

[9] We do not use $\mathrm{AUC}(C)$, as G-DINO and LLaVa provide binary answers, making thresholding meaningless.

[10] We remark that understanding what conditions enable VLMs to acquire high-quality concepts is still an open problem (Rajendran et al., 2024).

Table 3 Results averaged over 5 runs

	Model	$F_1(Y)$ (↑)	AUC(C) (↑)	LEAK (↓)	DIS (↑)	OIS (↓)
Shapes3d	CBM	0.99 ± 0.01	0.99 ± 0.01	0.18 ± 0.06	0.99 ± 0.01	0.08 ± 0.01
	Naïve	0.54 ± 0.06	0.17 ± 0.01	$\underline{0.14} \pm 0.01$	0.20 ± 0.01	0.16 ± 0.01
	LaBo	0.76 ± 0.01	$\underline{0.31} \pm 0.01$	0.81 ± 0.01	$\underline{0.28} \pm 0.01$	$\underline{0.15} \pm 0.01$
	LF-CBM	$\underline{0.96} \pm 0.01$	$\underline{0.31} \pm 0.01$	0.64 ± 0.01	$\underline{0.28} \pm 0.01$	$\underline{0.15} \pm 0.01$
	VLG-CBM	0.52 ± 0.01	0.24 ± 0.01	0.06 ± 0.10	0.24 ± 0.01	0.16 ± 0.01
CelebA	CBM	0.95 ± 0.01	0.75 ± 0.01	0.41 ± 0.01	0.74 ± 0.01	0.13 ± 0.01
	Naïve	0.95 ± 0.01	$\underline{0.51} \pm 0.01$	0.99 ± 0.01	$\underline{0.39} \pm 0.01$	$\underline{0.16} \pm 0.01$
	LaBo	0.97 ± 0.01	0.38 ± 0.01	$\underline{0.42} \pm 0.01$	0.28 ± 0.01	0.18 ± 0.01
	LF-CBM	0.99 ± 0.01	0.41 ± 0.01	0.53 ± 0.01	0.29 ± 0.01	0.17 ± 0.01
	VLG-CBM	$\underline{0.98} \pm 0.01$	0.32 ± 0.01	0.92 ± 0.01	0.21 ± 0.01	0.20 ± 0.01
CUB	CBM	0.69 ± 0.01	0.91 ± 0.01	0.10 ± 0.02	0.75 ± 0.01	0.09 ± 0.01
	Naïve	0.66 ± 0.01	0.43 ± 0.03	$\underline{0.02} \pm 0.01$	0.27 ± 0.02	0.15 ± 0.01
	LaBo	0.27 ± 0.01	0.25 ± 0.01	0.01 ± 0.01	0.18 ± 0.01	0.15 ± 0.01
	LF-CBM	$\underline{0.68} \pm 0.01$	0.25 ± 0.01	0.01 ± 0.01	0.19 ± 0.01	0.16 ± 0.01
	VLG-CBM	0.60 ± 0.01	$\underline{0.79} \pm 0.01$	$\underline{0.02} \pm 0.01$	$\underline{0.60} \pm 0.01$	$\underline{0.10} \pm 0.01$

Best results are in **bold** and second best are underlined

a more detailed study thereof to future work, while we provide few examples of automated annotations in Fig. 9b. More details on the experiment can be found in Appendix A.2.

4.2 Q2: VLM-CBMs Output Good Labels but Inaccurate Concepts

Next, we evaluate VLM-CBMs predictions and concepts using F_1 score for the former (denoted $F_1(Y)$) and $\mathrm{AUC}(C)$ for the latter (Sect. 3.1). The results for all models and data sets are reported in Table 3.

A first key finding is that VLM-CBMs *often achieve $F_1(Y)$ comparable to the* `CBM` *baseline using substantially less accurate concepts*, as indicated by $\mathrm{AUC}(C)$.[11] The gap is particularly noticeable in `Shapes3d`, where all VLM-CBMs have much lower $\mathrm{AUC}(C)$ than `CBM`, yet `LF-CBM` (and to a lesser extent `LaBo`) surprisingly achieve good F_1. On the other hand, `Naïve` and `VLG-CBM` produce poor label predictions, showing that the learned concepts carry very little to no information about the label. The other two data sets paint a similar picture: in `CelebA`, $\mathrm{AUC}(C)$ never crosses 50% while $F_1(Y)$ for all models is always above 95%, matching or even surpassing the `CBM` baseline; and in `CUB`, the VLM-CBMs fare poorly at $\mathrm{AUC}(C)$– the only exception being `VLG-CBM`, with $\mathrm{AUC}(C) = 79\%$ – despite all of them except `LaBo` achieving competitive $F_1(Y)$ ($\geq 60\%$, close the `CBM` 's 69%). Again, this show a disconnect between label and concept accuracy, and that maximizing the former provides no guarantees about the latter.

We readily acknowledge that for `LF-CBM`, `LaBo`, and `VLG-CBM`, our $F_1(Y)$ results on `CUB` are lower than those reported in the original publications (Oikarinen et al., 2023; Yang et al., 2023; Srivastava et al., 2024). This is because the original evaluation used a broad set of LLM-generated concepts whose number can be up to ten thousand (a comparison of bottleneck sizes on `CUB` dataset can be found in Table 6a). In contrast, *we always employ the gold-standard concept vocabulary*. Note that this is sufficient for inferring the correct labels, and indeed the associated manual concept annotations enable `CBM` to perform well on all data sets in terms of $F_1(Y)$. Our results indicate that *restricting these models to the gold-standard vocabulary does affect label accuracy compared to using larger vocabularies*. When contrasted with existing results (Oikarinen et al., 2023; Yang et al., 2023; Srivastava et al., 2024), this leads us to hypothesize that a more extensive concept vocabulary allows the bottleneck to retain more information about label predictions.[12]

Finally, we attribute the lackluster performance of `LaBo` to the fact that it is the only model without a backbone pretrained on `CUB`. We believe such task-specific pre-training, which is common practice for this data set (Oikarinen et al., 2023; Srivastava et al., 2024), brings a substantial performance advantage and as such it should be employed whenever feasible.

4.3 Q3: VLM-CBMs Often Leverage Leaky, Entangled, and Impure Concepts

We now turn to evaluating specific aspects of concept quality, beginning with leakage (Sect. 3.2). Compatibly with existing findings (Mahinpei et al., 2021; Margeloiu et al.,

[11] The poor concept accuracy is likely due to poor VLM annotations, see Sect. 4.1.

[12] This observation is compatible with other results in the field, which show that it is possible to increase the label accuracy adopting a larger bottleneck both when either concepts do not correspond to anything meaningful (Srivastava et al., 2024; Mahinpei et al., 2021).

2021; Havasi et al., 2022; Marconato et al., 2022), the results in Table 3 indicate that CBM *is affected by leakage in all datasets*, with a maximum of 41% in CelebA and a minimum of 10% in CUB.

In Shapes3d, both LF-CBM and LaBo (CLIP-based models) are affected by leakage. VLG-CBM is the only method that considerably avoids leakage in this dataset, despite faring poorly in label predictions. Note that a low $F_1(Y)$ does not imply leakage is absent, as demonstrated by the results of Naïve,[13]

The trend is flipped in CelebA: LaBo fares similarly to CBM, LF-CBM drops to 53%, Naïve obtains maximum leakage despite achieving an $\mathrm{AUC}(C)$ higher than other VLM-CBMs, and VLG-CBM is highly affected by it, with 92% leakage. In Fig. 3 (left), we report the gaps for CelebA, which better spot on which subset of concepts (VLM) CBMs suffer leakage more. For example, the plots show that even if CBM and LaBo have similar LEAK (41% vs 42%), they display different behavior: LaBo displays a small amount of leakage in every concept, while CBM reveals that only a few concepts are more leaky. In particular, the concepts *"big lips"* and *"bald"* appear to introduce the most substantial amount of leakage on CBM. Interestingly, we observe that gaps of all models are above zero when considering all concepts in the bottleneck.[14]

The evaluation on CUB portrays a different picture where only CBM and Naïve are sensibly affected by leakage. The high gaps for CBM in Fig. 3 (right) show that a big proportion of label information is spread onto few concepts. VLG-CBM and LaBo instead do not leak label information in the concepts, as visible for the always small or negative gaps.

These leakage results across datasets suggest that VLMs (such as LaBo, LF-CBM in Shapes3d and Naïve, LaBo, LF-CBM, VLG-CBM in CelebA) can learn concepts whose semantics are less aligned with the ground-truth (i.e., that contain extra information),

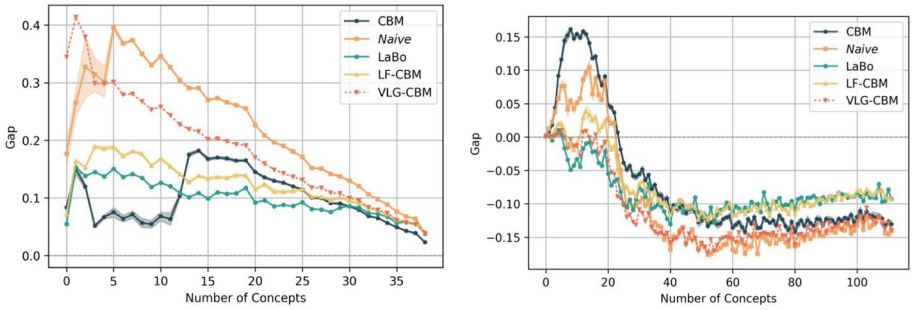


Fig. 3 Gaps of the concept leakage test in **CelebA** and **CUB**. The higher the gap, the more the subset of learned concepts leaks information for label prediction. (Left) Gaps on CelebA upon varying the subset of concepts from 1 to 39. All model concepts always display a non-negative gain in label F_1 compared to ground-truth concepts. (Right) Gaps on CUB upon varying the subset of concepts from 1 to 112. Only CBM is affected by leakage when considering smaller subsets of concepts

[13] This can be noted from the plot on gaps in Fig. 9 where Naïve leaks a small amount of label information at early stages with a subset of a few irrelevant concepts. The gaps reduce at later stages, especially when adding all concepts, where, eventually, Naïve can no longer achieve label F_1 above the ground-truth.

[14] This shows that *extra, stylistic information* about labels is encoded in the bottleneck beyond the predefined ground-truth concepts. For example, as in Fig. 2 (2), the model could exploit textual information within the image (that is correlated to the label) to improve the prediction.

making their classification process less interpretable. We postulate this occurs because – just like regular CBMs – VLM-CBMs can optimize for task performance by exploiting spurious information available in concept annotations and inputs, even more so when ground-truth concepts are insufficient to perfectly predict the label (Yeh et al., 2020; Havasi et al., 2022).

Next, we look at disentanglement and OIS (Sects. 3.3 and 3.4). Overall, all tested models that achieve high $\mathrm{AUC}(C)$ also display higher disentanglement and lower OIS. In all datasets, `CBM` attains the highest DIS and the lowest OIS thanks to expert annotation, whereas VLM-CBMs fare below the baseline `Naïve` in `Shapes3d` and `CelebA`. On `CUB`, `VLG-CBM` achieve higher DIS and lower OIS compared to other models, thanks to the higher quality of annotation of `G-DINO`. We analyze the DCI importance matrices R_{ij} for `CelebA` on a single, optimal $F_1(Y)$ run of each competitor, cf. Figure 12. All VLM-CBMs drastically deviate from the behavior of `CBM`, showing different entangled concepts based on the VLM they leverage. For example, `Naïve` displays high entanglement for the concept *"goatee"* and *"wearing lipstick"*, CLIP-based approaches show that *"goatee"*, *"no beard"*, and *"smiling"* are the most entangled concepts, whereas `VLG-CBM` entangles several concepts, the most evident being *"gray hair"*, and *"pale skin"*. For all models, except `VLG-CBM` in `CUB`, low performance in DIS and high OIS indicate that learned concepts often depends on variations of other, unrelated concepts. This behavior muddles the semantics of individual learned concepts, which do not respect the ground-truth dependencies, resulting in overall non-interpretable predictions (e.g., `VLG-CBM` in `CelebA` attains 0.21 DIS and 0.20 OIS).

Finally, we analyze the Spearman rank correlation coefficient between different metrics across runs and datasets (see Appendix C.1 for details). Each model seed run constitutes a point used in this correlation analysis. We observe in Fig. 4 that both DIS and OIS are highly correlated with $\mathrm{AUC}(C)$ of the model (0.86 for $\mathrm{AUC}(C)$ and DIS with p-value of $1.1 \cdot 10^{-22}$ and -0.76 for $\mathrm{AUC}(C)$ and OIS with p-value $5.1 \cdot 10^{-15}$), and that DIS and OIS are negatively correlated (correlation coefficient of -0.73 with p-value $9.8 \cdot 10^{-14}$). Therefore, the low values of $\mathrm{AUC}(C)$ leave room for models to learn concepts with poor dependencies between them (DIS and OIS). Interestingly, we also find that $F_1(Y)$ and LEAK are positively correlated (coefficient of 0.64 with p-value $7.8 \cdot 10^{-10}$). Specializing the analysis to datasets separately, we find that $F_1(Y)$ and LEAK are extremely correlated in `CUB` (correlation coefficient of 0.97 with p-value of $6.7 \cdot 10^{-15}$), while they are not in `Shapes3d` and `CelebA`, see Fig. 10. This correlation shows that the label performance of all CBMs and VLM-CBMs we tested can be boosted by exploiting unintended information that leaks in the learned concepts and ignoring actual concepts predictions of the model.

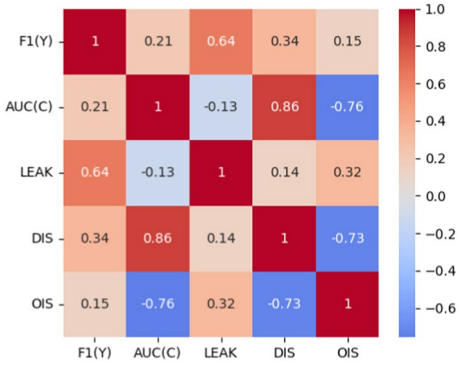


Fig. 4 Correlations among metrics across datasets and models. The matrix displays the Spearman correlation coefficient between metrics for 75 trained models

Overall, these results highlight that, in our experiments, some metrics are mutually dependent; however, a more in-depth analysis to reveal their precise correlations is left for future work.

Conclusions on concept quality analysis. Our results indicate that overall, across datasets, VLM-CBMs often leverage poor-quality concepts, mostly due to inaccurate annotations from replacing human annotations with VLM-guided concepts. The only positive exception is `VLG-CBM` in `CUB` that, thanks to more accurate concept annotations, fares better than other competitors.

5 Related Work

Concept-based models are a heterogeneous class of models that leverage learned concepts to glue together low-level perception with transparent high-level reasoning, see (Schwalbe, 2022; Poeta et al., 2023; Ji et al., 2025) for recent overviews. *Unsupervised* models rely on concept discovery techniques, and foster interpretability via penalty terms architectural biases, such as *sparsity* (Alvarez Melis & Jaakkola, 2018), *orthogonality* (Chen et al., 2020), *reconstruction* (Li et al., 2018) and *prototypes* (Chen et al., 2019). Supervised CBMs differ in what information the encode in the concept bottleneck: *concept activations* (Koh et al., 2020), *embeddings* (Zarlenga et al., 2022), *probabilities* (Marconato et al., 2022; Kim et al., 2023; Vandenhirtz et al., 2024), or *discrete activations* (Havasi et al., 2022; Lockhart et al., 2022). VLM-CBMs relax the need for expert supervision by leveraging pre-trained VLMs.

The issue of interpretability in CBMs is well-known. Surprisingly, most works on VLM-CBMs do not focus on ***concept quality*** (Oikarinen et al., 2023; Yang et al., 2023; Srivastava et al., 2024; Rao et al., 2024; Schrodi et al., 2024). It is well-known that, however, the semantics of concepts learned from data can be misleading (Furby et al., 2023) and that even achieving high concept accuracy is not sufficient to ensure concepts are interpretable (Mahinpei et al., 2021; Margeloiu et al., 2021; Havasi et al., 2022; Marconato et al., 2022; Barbiero et al., 2025). A number of metrics have been introduce to establish desirable properties of learned concepts. We focus on *leakage* (Havasi et al., 2022; Mahinpei et al., 2021), *disentanglement* (Marconato et al., 2022), *OIS* (Zarlenga et al., 2023), but other works have looked at *completeness* (Yeh et al., 2020; Havasi et al., 2022), *locality* (Raman et al., 2023), or *post-hoc alignment* (Mikriukov et al., 2023; Bortolotti et al., 2025). Techniques for improving concept quality build on causal insights (Bahadori & Heckerman, 2021) and on side-information (Sawada & Nakamura, 2022a; Havasi et al., 2022) obtained from either experts (Bontempelli et al., 2023; Lertvittayakumjorn et al., 2020; Stammer et al., 2021; Teso et al., 2023) or foundation models (Srivastava et al., 2024). Provided that the extracted concepts are high-quality, ***the inference layer must satisfy a set of properties*** to guarantee correct decision making and be high-quality. Several works have indicated that *sparsity* is a central property (Alvarez Melis & Jaakkola, 2018; Barbiero et al., 2022; Oikarinen et al., 2023) and that can lead to least leakage of information (Srivastava et al., 2024), while other works focus on making the classifier decision mechanism more *understandable* by learning logic rules as in decision trees (Barbiero et al., 2023, 2024), or *verifiable* (Debot et al., 2024), or *causally transparent* (De Felice et al., 2025; Dominici et al., 2024; Moreira et al., 2024). Since we focus specifically on concept quality, we leave to future work to examine how VLM-CBMs stimulate classifier quality.

Beyond CBMs. The issue of concept quality extends beyond CBMs to other branches of AI, including Neuro-Symbolic AI and Foundation Models. For instance, it was shown that the concepts learned by Neuro-Symbolic (NeSy) architectures can be compromised by shortcuts (Marconato et al., 2024), and these findings were recently extended to CBMs (Bortolotti et al., 2025). A popular solution is, just like for CBMs, to leverage foundation models (Steinmann et al., 2024). Our work highlights that this strategy comes with its own set of pitfalls and cannot strictly guarantee the trustworthiness of the predictor.

6 Conclusion

Our results suggest that, despite significantly broadening the applicability of CBMs, VLMs are unfortunately not yet an accurate substitute for high-quality expert annotations. Annotation quality translates not only to worse concept accuracy, but it also results in higher leakage, worse entanglement, and more impure concepts, depending on the dataset. While it remains to be seen whether the quality of machine-generated concepts could match man-made ones, our results highlight that leakage remains an open problem even when using predefined (and sufficient) textual description of concepts for both CBMs and VLM-CBMs. Even though the results do not show a direct correlation between annotation quality and increased leakage, we note that VLM-CBMs tend to be less interpretable than CBMs, while exhibiting (depending on the dataset) comparable levels of leakage.

With this work, we hope to help understand potential limitations of VLM-CBMs and to prompt researchers to place more emphasis on concept quality when developing future architectures. We will conclude by pointing out possible ways forward. One natural solution is, of course, to simply wait for VLMs to improve. While this is not unlikely given recent trends, leveraging bigger models necessarily comes at a substantial computational cost. We suggest that a more future-proof strategy, not necessarily orthogonal to the former, is to design VLM-CBMs that are more robust to low-quality concept annotations. This could be achieved, for instance, by leveraging sound confidence estimates of the VLM's annotations during training (Abbasi Yadkori et al., 2024), exploiting well-known regularization terms and learning strategies which proved effective for regular CBMS (Bahadori & Heckerman, 2021; Fokkema et al., 2025; Marconato et al., 2022), and actively seeking manual annotations for low-quality concepts (Espinosa Zarlenga et al., 2024; Steinmann et al., 2024).

Appendix A: Experimental Details

All experiments were implemented using Python 3 and Pytorch (Ansel et al., 2024). The code is available at https://github.com/debryu/CQA.

Pipeline

Concept vocabulary: Every dataset used in our experiments comes with annotated attributes. For example, `CUB` has for each image a vector of 112 values (we refer to them as *ground-truth concepts*, Fig. 5 green box) that are each associated with a textual description (Fig. 5 orange box at the top).

To query the concept in the VLM, we preprocess the original text: this step is done once per dataset, maintaining the same number of concepts and semantic meaning. In Fig. 5 bottom orange box, there is an example of the output, and we refer to this as *concept vocabulary*. **All models use the same concept vocabulary**. In contrast to methods that rely on LLMs to generate the concept descriptions, this step makes sure that for each attribute, there is one and only one ground-truth value.

Replacing human annotations: VLM-CBMs rely on a VLM to generate concept annotation that are then used to train a Concept Bottleneck Model (CBM). `Naïve` and `VLG-CBM` produce boolean concept annotations $\hat{c}_i \in \{0, 1\}$(Fig. 6 blue box) by using `LLaVa-Contributors` (2023) and `G-Dino` Liu et al. (2024), respectively.

`LaBo` and `LF-CBM` leverage `CLIP`Radford et al. (2021) (through cosine similarity of textual and visual embeddings) to produce continuous values $\hat{c}_i \in [-1, 1]$ as concept annotations. More implemention details can be found in Sect. 4 and Appendix B.2.

Training and evaluation: During training of VLG-CBMs, the bottleneck is supervised to match the VLM annotated concepts (with the sole exception of `CBM` model, which uses ground-truth concepts as supervision). The training is performed sequentially: (1) first, we train the bottleneck, then (2) the inference layer. At the end, we collect activations of the bottleneck, i.e., the learned concepts (red box in Fig. 7), and predicted class (blue box in Fig. 7) for evaluations.

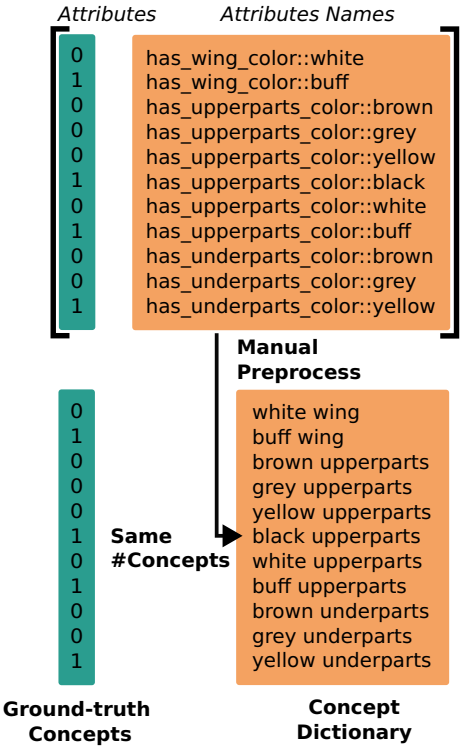


Fig. 5 Generation of the concept vocabulary

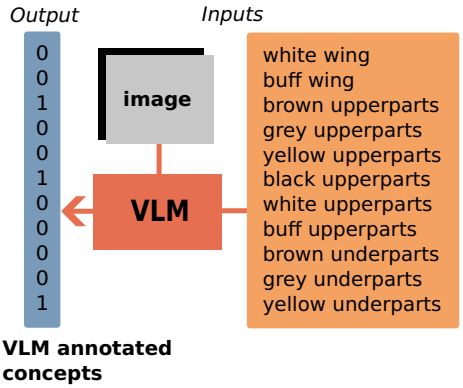


Fig. 6 VLM data annotation

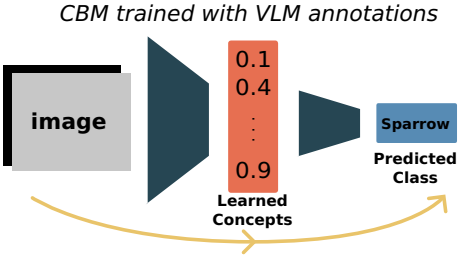


Fig. 7 CBM evaluation after training. The model is run on the entire test set to store learned concepts and predicted classes

Evaluating VLM Annotations

To evaluate VLM annotations we use macro average precision and recall measured on predicted concepts versus ground-truth ones. While `G-DINO` and `Naïve` produce boolean annotations (i.e. a concept is either 0 or 1 as in Fig. 6), this is not true for `CLIP` scores. To overcome this, we train a logistic regressor model for each concept (using `CLIP` embeddings as input), and then assign 0 or 1 to the concept based on the output of that model. In this phase, we consider leveraging ground-truth concept annotations in a small portion of the test set $(\mathrm{x}, \mathrm{c}) \in \mathcal{D}_{test}$. For each predicted concept $\hat{c}_i$, we fit a weight $w_i \in \mathbb{R}$ and a bias $b_i \in \mathbb{R}$, to minimize the cross-entropy loss on the scores obtained from CLIP vision encoding $E_V(\mathrm{x})$ of an image x and the textual encoding $E_T(t_i)$ of the concept string t_i:

$$\ell_i = \frac{1}{|\mathcal{D}_{test}|} \sum_{(\mathrm{x},c_i)\in\mathcal{D}_{test}} c_i \cdot \log\Big(\sigma\big(w_i \cdot S_c(E_V(\mathrm{x}), E_T(t_i)) + b_i\big)\Big) + (1-c_i)\cdot \log\Big(1-\sigma\big(w_i S_c(E_V(\mathrm{x}), E_T(t_i)) + b_i\big)\Big)$$

where $S_c(\cdot,\cdot)$ denotes the cosine similarity and $\sigma(\cdot)$ is the sigmoid function. Then, to convert CLIP scores to predicted concepts, we apply the step-function using the values of the weights and biases:

$$\hat{c}_i = \begin{cases} 1 & \text{if } \sigma\big(w_i S_c(E_V(\mathrm{x}), E_T(t_i)) + b_i\big) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Once we have boolean concept annotations generated from `CLIP`,`G-DINO`, and `LLaVa`, we evaluate per-concept macro average precision and recall. In Table 2, we report the average for all concepts.

We also take into consideration whether some concepts are mutually exclusive, such as in `Shapes3d` where concept 1 through 10 are the one-hot encoding of the color, so only one will be active. With that in mind, when evaluating `CLIP` annotations on `Shapes3d` we evaluate concept activation separately for each group of concepts (wall color, background color, object color, size and shape), setting as *active* only the concept with highest activation withing the group.

VLMs Evaluated Using AUC

In Table 4, we report $\mathrm{AUC}(C)$ of annotations generated from `CLIP`, `G-DINO`, and `LLaVa`. We observe that, similar to results in Table 2, `G-DINO` and `LLaVa` perform well on `CUB` and fare poorly on `Shapes3d` and `CelebA`. `CLIP`, on the other hand, achieves poor $\mathrm{AUC}(C)$ in all datasets. Focusing on `CLIP`, we observe it exhibits lower performance on `CUB` compared to the precision and recall analysis in Table 2. This happens because there exist specific choices of the weights and biases for the similarity scores of `CLIP` that result in good concept predictions. This issue is sidestepped when optimizing the weights and biases for concept predictions as in Appendix A.2.

Concept Vocabulary

Concept Vocabulary Generation

When the concept vocabulary is not available, a common choice is to query a Large Language Model to create a set of textual descriptions of the concepts $\mathcal{T}$. However, one known limitation is that an LLM can produce invalid concepts (Yang et al., 2023). In general, a high-quality concept vocabulary generation is difficult because the concept names need to avoid being overly specific or narrow for the task. In Table 5, we report some examples of invalid concepts in the generated vocabularies for `CUB`. These concept descriptions are taken from the original evaluation of the methods we tested.

Table 4 Concept $\mathrm{AUC}(\uparrow)$ on VLMs concept annotations

	Shapes3D	CelebA	CUB200
CLIP	0.30	0.38	0.25
G-DINO	**0.56**	**0.48**	**0.98**
LLaVa	**0.37**	**0.54**	**0.74**

Bold values indicate cases in which varying the threshold gives flat regions

Table 5 Examples of invalid concepts from the CUB vocabulary generated using LLMs

Method	Generated concepts	
	Irrelevant	Non-visual
LF-CBM (Oikarinen et al., 2023)	"A Mohs hardness of 10"	"Loud, rasping voice"
		"A loud, harsh cry"
		"A loud, high-pitched song"
LaBo (Yang et al., 2023)	"Hollow trees or stumps"	"Series of high, piping notes"
	"Adds color and life to our forests"	"Song is a trill"
	"Named after anna mary robertson, better known as grandma moses"	"Cawing can be quite loud and annoying"
VLG-CBM (Srivastava et al., 2024)	"A trackpad or mouse"	"A loud, melodious song"
		"Loud, rasping voice"
		"A loud caw"

The concepts are taken from the filtered set of the original implementation (Oikarinen et al., 2023; Srivastava et al., 2024; Yang et al., 2023)

Table 6 Concept vocabulary size used in original works (a) versus ours (b)

(a) Number of selected concepts for CUB dataset based on the original implementations. We also report the relative increase in vocabulary size compared to ours.

Model	N. Concepts CUB	Increase
CBM Koh et al. (2020)	112	$\times 1$
LF-CBM Oikarinen et al. (2023)	370	$\times 3$
LaBo Yang et al. (2023)	$\geq$ 2000	$\times 18$
VLG-CBM Srivastava et al. (2024)	723	$\times 6$

(b) Our concept vocabulary size across datasets. The bottleneck size is constant to enable fair comparison.

	N. Concepts		
	Shapes3d	CelebA	CUB
CBM Koh et al. (2020)	42	38	112
LF-CBM Oikarinen et al. (2023)	42	38	112
LaBo Yang et al. (2023)	42	38	112
VLG-CBM Srivastava et al. (2024)	42	38	112

Concept Vocabulary in Other Works

As a reference, we provide the number of concepts used in the original implementation of the methods we use in our experiments. We report the relative increase Table 6(a) in the number of concepts, which is the rounded multiplier of the original vocabulary size compared to ours. In previous works, the size of the vocabulary is significantly bigger, making it impossible to perform a fair comparison in label accuracy between the performance of our VLM-CBMs with the original versions.

Our Fixed Concept Vocabulary

Our objective is to use a vocabulary of concepts that can be shared across different models, which also includes access to ground-truth values. With that in mind, we took the textual description of the ground-truth concepts, with some minor preprocessing such as removing underscores for better understanding. We list below the complete sets of concepts used in our experiments for the `CUB`, `CelebA`, and `Shapes3d` datasets.

`CUB` concept vocabulary			
Dagger beak	White back	Yellow forehead	Multi-colored back
Hooked seabird beak	Buff back	Black forehead	Solid tail
All-purpose beak	Notched tail	White forehead	Striped tail
Cone beak	Brown upper-tail	Brown under-tail	Multi-colored tail
Brown wing	Grey upper-tail	Grey under-tail	Solid belly
Grey wing	Black upper-tail	Black under-tail	Brown primary color
Yellow wing	White upper-tail	White under-tail	Grey primary color
Black wing	Buff upper-tail	Buff under-tail	Yellow primary color
White wing	Eyebrow head	Brown nape	Black primary color
Buff wing	Plain head	Grey nape	White primary color
Brown upperparts	Brown breast	Yellow nape	Buff primary color
Grey upperparts	Grey breast	Black nape	Grey leg
Yellow upperparts	Yellow breast	White nape	Black leg
Black upperparts	Black breast	Buff nape	Buff leg
White upperparts	White breast	Brown belly	Grey beak
Buff upperparts	Buff breast	Grey belly	Black beak
Brown underparts	Grey throat	Yellow belly	Buff beak
Grey underparts	Yellow throat	Black belly	Blue crown
Yellow underparts	Black throat	White belly	Brown crown
Black underparts	White throat	Buff belly	Grey crown
White underparts	Buff throat	Rounded wings	Yellow crown
Buff underparts	Black eye	Pointed wings	Black crown
Solid breast	Beak length about–	Size small	White crown
Striped breast	– The same as head	Size medium	Solid wing
Multi-colored breast	Beak length shorter–	Very small size	Spotted wing
Brown back	– Than head	Duck-like	Striped wing
Grey back	Blue forehead	Perching-like	Multi-colored wing
Yellow back	Brown forehead	Solid back	
Black back	Grey forehead	Striped back	

`Shapes3d` concept vocabulary		
Red floor	Lime wall	Purple object
Orange floor	Azure wall	Pink object
Yellow floor	Blue wall	Miniscule object
Green floor	Navy blue wall	Very small object
Lime floor	Purple wall	Small object
Azure floor	Pink wall	Medium size object
Blue floor	Red object	Slightly big object
Navy blue floor	Orange object	Big object
Purple floor	Yellow object	Very big object
Pink floor	Green object	Enormous object

Shapes3d concept vocabulary		
Red wall	Lime object	Cube shaped object
Orange wall	Azure object	Cylinder shaped object
Yellow wall	Blue object	Sphere shaped object
Green wall	Navy blue object	Pill shaped object

CelebA concept vocabulary		
5 o clock shadow	Chubby	Pointy nose
Arched eyebrows	Double chin	Receding hairline
Attractive	Eyeglasses	Rosy cheeks
Bags under eyes	Goatee	Sideburns
Bald	Gray hair	Smiling
Bangs	Heavy makeup	Straight hair
Big lips	High cheekbone	Wavy hair
Big nose	Mouth slightly open	Wearing earrings
Black hair	Mustache	Wearing hat
Blond hair	Narrow eyes	Wearing lipstick
Blurry	No beard	Wearing necklace
Brown hair	Oval face	Wearing necktie
Bushy eyebrows	Pale skin	Young

Appendix B: Models Details

Naïve

We consider a fixed concept vocabulary $\mathcal{T}$ consisting of a total of k concept textual descriptions. According to the description of VLM-CBMs training Sect. B.2, our proposed baselines consist of the following three steps:

1. Gathering concept annotation for all input–output pairs from the training set by querying a LLaVa model to provide a binary label if any of the concepts in $\mathcal{T}$ is present.
2. We use the collected binary concept annotation to train the concept extractor f.
3. By freezing the concept extractor, a linear layer g is trained to predict the classes from concept activations.

To execute the first step, similarly to (Srivastava et al., 2024), we prompt the VLM with the following script:

```
"This is an image of {class_name}.
Does the image contain {prefix}{obj}{suffix}?
Please reply only with 'Yes' or 'No'."
```

where class_name is the class associated to the input sample and obj is the concept being queried. To adapt to different datasets, we added prefix and suffix, which can be empty strings or customized to enrich the prompt. For example, in CelebA we use the prefix *"a person with"*, so that the full prompt ends up being "Does the image contain a person with Moustache?".

Table 7 CBM architecture

Input	Type	Output	Produces	Trained	With
(224, 224, 3)	ResNet18	(1000)		✓	
(1000)	Linear	(#Conc.)	Concept logits	✓	XENT
(#Conc.)	Sigmoid	(#Conc.)	Concept probs	✗	
(#Conc.)	Linear	(#Class)	Class logits	✓	GLM-SAGA

Table 8 LaBo architecture

Input	Type	Output	Produces	Trained	With
(224, 224, 3)	CLIP ViT-B/16	(768)		✗	
(768),(768, $\#Conc.$)[15]	Dot product	(#Conc.)		✗	
(#Conc.)	Normal-ization	(#Conc.)	Concept logits	✗	
(#Conc.)	Linear	(#Class)	Class logits	✓	XENT

(768) is the CLIP image embedding, while (768, #Conc.) are the textual concepts encoded using CLIP

When the model does not follow the prompt with its answer, we append after the response

```
"Given this information, my 'Yes' or 'No' answer is:"
```

If the response from the model is still not satisfactory, we set the concept annotation value to 0. The VLM is run locally, using https://ollama.com.

Architectures

We list here the description of all architectures used in our experiments, where, for ease of reading, we omit the details about the backbones we used. For each component we specify its name, input, output, what it produces (i.e. the output of a layer could simply be an intermediate process, and if so the field is empty, or return important information such as logits or probabilities) and lastly if the layer/component is frozen or being trained. We also specify how a particular component is trained, where the options are:

1. XENT: standard PyTorch cross entropy loss computed between ground-truth and predicted values.
2. SVM: we use Support Vector Machine Schölkopf et al. (2000) (specifically the scikit learn implementation) to fit a linear classifier.
3. GLM-SAGA: we minimize an elastic net loss using the GLM-SAGA solver created by Wong et al. (2021), in order to obtain a sparse linear classifier (Tables 7, 8, 9, 10, 11).

For CUB, the ResNet18 output shape is 200, matching the number of classes, and so the following linear layer will have input of dimension 200 while keeping unvaried the other components of the architecture.

Pre-trained models are dowloaded from https://pytorch.org/vision/main/models.html and https://github.com/osmr/imgclsmob.

Table 9 `Naïve` architecture

Input	Type	Output	Produces	Trained	With
(224, 224, 3)	ResNet18	(1000)		✓	
(1000)	Linear	(#Conc.)	Concept logits	✓	XENT
(#Conc.)	Sigmoid	(#Conc.)	Concept probs	✗	
(#Conc.)	Linear	(#Class)	Class logits	✓	SVM

Table 10 `LF-CBM` architecture

Input	Type	Output	Produces	Trained	with
(224, 224, 3)	CLIP ViT-B/16	(768)		✗	
(768)	Linear	(#Conc.)	Concept logits	✓	XENT
(#Conc.)	Linear	(#Class)	Class logits	✓	GLM-SAGA

Table 11 `VLG-CBM` architecture

Input	Type	Output	Produces	Trained	with
(224, 224, 3)	CLIP ViT-B/16	(768)		✗	
(768)	Linear	(#Conc.)		✓	XENT
(#Conc.)	Normalization	(#Conc.)	Concept logits	✗	
(#Conc.)	Linear	(#Class)	Class logits	✓	GLM-SAGA

Hyperparameter Selection

During training, we apply early stopping when a model's validation loss does not decrease after 16 consecutive epochs.

We conducted a grid search over different sets of hyperparameters and determined the best combination by monitoring the label F_1 score on the validation set with wandb. As for the optimizer, we used Adam (Kingma & Ba, 2014). Besides more common hyperparameters, we use a flag `unfreeze` to control the number of layers to unfreeze from the ResNet backbone (starting from the last one). The flag `balanced` is used to handle dataset imbalances when training the concept bottleneck.

CBM model parameters are set as:`-epochs=20,-unfreeze=5,-lr=0.001,-batch_size=512,-backbone=resnet18,-optimizer=adamw,-balanced`.

Naïve model parameters are set as:`-epochs=20,-unfreeze=5,-lr=0.001,-predictor=svm,-c_svm=1,-batch_size=512,-backbone=resnet18,-optimizer=adamw,-balanced,-ollama_model=llava-phi3`.

LaBo model parameters are set as:`-lr=0.01,-batch_size=258,-backbone=clip_RN50,-feature_layer=layer4,-clip_name=ViT-B/16`.

LF-CBM model parameters are set as:`-lr=0.01,-batch_size=258,-backbone=clip_RN50,-feature_layer=layer4,-clip_name=ViT-B/16,-proj_steps=1000, clip_cutoff=0.0,-interpretability_cutoff=0.0`.[15]

VLG-CBM model parameters are set as:`-backbone=clip_RN50,-feature_layer=layer4,-crop_to_concept_prob=0.01,-cbl_confidence_threshold=0.15,-cbl_hidden_layers=3,-cbl_epochs=8,-cbl_lr=0.001,-val_split=0.1`.

For the models that require **GLM-SAGA** (Wong et al., 2021) we use the following parameters:`-glm_alpha=0.99,-glm_step_size=0.1,-n_iters=2000,-lam=0.0007,-saga_batch_size=256`.

Lastly, when running experiments on `CUB` we use`-backbone=resnet18_cub` for `CBM`, `Naïve`, `LF-CBM` and `VLG-CBM`. For `LF-CBM` and `VLG-CBM` we also specify`-feature_layer=features.final_pool` which is the last layer of the pretrained ResNet backbone. All the other hyperparameters are kept the same, except for **VLG-CBM parameters** which are:`-backbone=resnet18_cub,-feature_layer=features.final_pool,-crop_to_concept_prob=0.1,-cbl_confidence_threshold=0.15,-cbl_hidden_layers=0,-cbl_epochs=35,-cbl_lr=0.0005.-lam=0.0002`.

Appendix C: Metrics Details

Spearman Rank Correlation Coefficient

In Fig. 4 and Sect. D.2, we evaluate the Spearman rank correlation coefficient between label performance and concept quality metrics of the trained models. Here we provide details on the evaluation performed. Each trained model on a specific dataset is evaluated according to $F_1(Y)$, $\mathrm{AUC}(C)$, LEAK, DIS, and OIS on the test set. We group all runs (5 seeds for each model in `Shapes3d`, `CelebA`, `CUB`), obtaining a total of 75 points where we evaluate the correlation coefficient among metrics. For each possible combination of metrics $m_A, m_b \in \{F_1(Y), \mathrm{AUC}(C), \mathrm{LEAK}, \mathrm{DIS}, \mathrm{OIS}\}$, with $m_A \neq m_B$, we evaluate the Spearman rank coefficient on the 75 values obtained. We also evaluate the correlation coefficients among metrics for each dataset individually, by restricting to models trained on that data. In this case, 25 points are considered for each dataset. The results are reported in Fig. 10.

DCI Framework

Given n ground-truth concepts $c_1, \ldots, c_n$ (or generative factors) and k learned concepts $\hat{c}_1, \ldots, \hat{c}_k$ (or code variables), the disentanglement metric is defined as:

$$D = \sum_{i=1}^{k} \rho_i D_i, \quad \text{where } \rho_i = \frac{\sum_{j=1}^{n} R_{ij}}{\sum_{\ell=1}^{k} \sum_{j=1}^{n} R_{\ell j}} \tag{5}$$

[15] The last two hyperparameters are set to zero to prevent unwanted concept filtering.

corresponding to the weighted average over all disentanglement scores D_i appearing in Eq. 3, where $i \in \{1, \ldots, k\}$. In general k and n can be different, however in our experiments we consider $k = n$, i.e., the number of ground-truth concepts and model concepts coincide in all dataset (39 for `CelebA`, 42 for `Shapes3d` and 112 for `CUB`).

The **matrix of relative importance** R is provided by the regressor, trained to predict the ground-truth concept c_j from the learned concept $\hat{c}_i$. It represents the relative importance of $\hat{c}_i$ in predicting c_j. As in the original implementation, we use a random forest regressor. The entries $R_{i,j}$ of the matrix count the number of times a tree splits on $\hat{c}_i$ over the total number of splits when predicting c_j (Breiman et al., 1984). That is, the more the predicted concept $\hat{c}_i$ creates a decision split in a tree for predicting the ground-truth concept c_j, the greater the importance of R_{ij}. For more details, we refer the reader to (Eastwood and Williams (2018), Sections 2 and 4.3).

When considering a lasso regressor instead of a random forest (with concepts normalized to have zero mean and unitary variance), the matrix of relative importance R can be obtained by the weights of the trained regressor:

$$R_{ij} = |W_{ij}| \tag{6}$$

where $|W_{ij}|$ is the absolute value of the weight applied to $\hat{c}_i$ in predicting c_j.

LEAK **Measure**

To establish each concept's relevance to label prediction, we compute the Pearson correlation coefficient between each ground-truth concept and the label. Then, both ground-truth and predicted concepts are sorted by increasing correlation with the label, so that highly correlated concepts are placed last. Next, we use the iterative algorithm in Algorithm 1 to implement the procedure detailed in Sect. 3.2.

Input:
$(\hat{C}_{train}, \hat{C}_{test}) \leftarrow$ predicted concepts on train/test set
$(C_{train}, C_{test}) \leftarrow$ ground truth concepts on train/test set
$(y_{train}, y_{test}) \leftarrow$ ground truth labels on train/test set
gaps = [ ]
for $i = 1$ to k **do**
    SVM.train($\hat{C}_{train}[:, 1:i], y_{\text{train}}$)
    predictions $\leftarrow$ SVM.predict($\hat{C}_{test}[1:i]$)
    F1_predicted $\leftarrow$ compute_F1(predictions, y_{test})
    SVM.train($C_{train}[:, 1:i], y_{\text{train}}$)
    predictions $\leftarrow$ SVM.predict($C_{test}[1:i]$)
    F1_ground_truth $\leftarrow$ compute_F1(predictions, y_{test})
    gaps.append(F_1_predicted $-$ F_1_ground_truth)
end for
return: gaps ▷ List of F_1 gaps for each step

Algorithm 1 Compute F1 gap between predicted and ground-truth concepts

The F_1 gap provides a principled measure of how much unintended label information is encoded in the learned concepts. Notice that, previous works (Mahinpei et al., 2021; Marconato et al., 2022) have proposed to measure the gap between the best label predictors

trained on irrelevant model concepts with a random label predictor. However, this requires indicating what concepts are irrelevant, needing highly dataset-specific knowledge Marconato et al. (2023). In contrast, our method defines a data-driven baseline (avoiding manual labeling of relevant and irrelevant concepts) by training a predictor on an increasing subset of concepts, starting from low-label-correlated concepts that are assumed to be largely irrelevant. In our setting, using random label predictors risks overestimating leakage, since we do not have information about which concepts are irrelevant.

Synthetic Evaluation of the LEAK Measure

We create simulated concepts to perform sanity checks on LEAK measure. We construct 16 ground-truth concepts c by independently sampling them from the uniform distribution $\mathrm{Unif}([-1,1])$. We consider the first 4 concept components to predict a binary label $y \in \{0,1\}$ by using a random matrix $W \in \mathbb{R}^{2\times 4}$, whose entries are randomly sampled iid from the normal distribution $\mathcal{N}(\mu=0,\sigma^2=5)$. Ground-truth labels are given by:

$$y_i \leftarrow \underset{y\in\{0,1\}}{\operatorname{argmax}} W_y^\top \mathrm{c}_{i,1:4} \tag{7}$$

where $\mathrm{c}_{i,1:4}$ is the vector containing to the first 4 entries of c_i. We generate in total $N = 10000$ pairs (c_i, y_i). We consider the following variants of the learned concepts:

1. Random: Model concepts are sampled iid from $\mathrm{Unif}([-1,1])$. This baseline should display zero leakage since concepts are not related to labels.
2. Noise: Model concepts are constructed from the ground-truth ones by adding noise $\boldsymbol{\varepsilon}$, where the components are sampled iid from the normal distribution $\mathcal{N}(0,\sigma^2)$. Formally $\hat{\mathrm{c}} = \mathrm{c} + \boldsymbol{\varepsilon}$. We consider three cases for $\sigma^2 \in \{0.1, 0.5, 1\}$. These variants should not present an increase in leakage because no extra information about labels is encoded.
3. Leak: We consider model concepts, where each component depends on the ground-truth label y, i.e., $\hat{c}_j = 2y - 1$, for all $j \in [16]$. To these, we add a noise variable $\boldsymbol{\varepsilon}$, whose components are sampled iid from the normal distribution $\mathcal{N}(\mu=0,\sigma^2)$, varying $\sigma^2 \in \{1,2,3,4\}$. As variance decreases, we expect concepts to be more informative about the label and to increase the amount of leakage.
4. Entangled: We consider a random matrix $\mathrm{O} \in \mathbb{R}^{16\times 16}$, where the components are sampled iid from the normal distribution $\mathcal{N}(\mu=0,\sigma^2=5)$. The model concepts are constructed as $\hat{\mathrm{c}}_i = \lambda \mathrm{O}\mathrm{c}_i + (1-\lambda)\mathrm{c}_i$, where $\lambda \in [0,1]$ is a parameter regulating the convex combination between the original concepts and the linearly transformed ones. We consider $\lambda \in \{0.25, 0.5, 0.75, 1\}$. For non-vanishing entanglement, we expect to observe leakage in the concepts.

For all four variants, we generate N model concepts, which together with ground-truth concepts and labels, are used to evaluate the LEAK of the model. The results for this simulation are reported in Table 12 and the gaps are displayed in Fig. 8. We observe that the ***Random*** baseline and the Noise ones fare very low in LEAK (always below or equal to 0.02) as expected. For the ***Random*** variant, we observe an increase in the gap when considering 3 concepts, probably due to a spurious effect and to N not being extremely high.

Table 12 Leakage scores over simulated data

	LEAK		LEAK		LEAK
Random	0.021	Leaky($\sigma^2 = 4.00$)	0.577	Ent.($\lambda = 0.25$)	0.588
Noisy($\sigma^2 = 1.00$)	0.001	Leaky($\sigma^2 = 3.00$)	0.665	Ent.($\lambda = 0.50$)	0.578
Noisy($\sigma^2 = 0.50$)	0.001	Leaky($\sigma^2 = 2.00$)	0.777	Ent.($\lambda = 0.75$)	0.600
Noisy($\sigma^2 = 0.10$)	0.001	Leaky($\sigma^2 = 1.00$)	0.962	Ent.($\lambda = 1.00$)	0.581

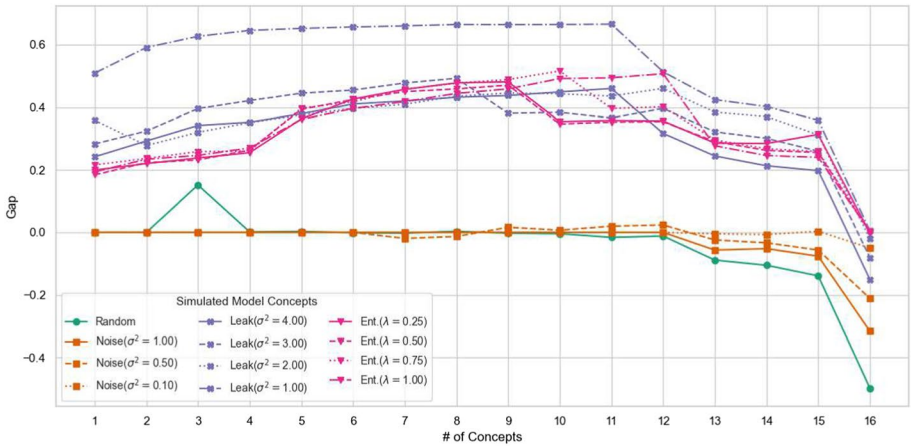


Fig. 8 Gaps in LEAK for simulated concepts. Leaky models, in purple, show the highest gaps resulting in a high amount of leakage when noise is reduced. Entangled models, in pink, also display positive gaps, increasing with the size of the learned concepts considered. Random (green) and Noise (orange), instead, reveal small or negative gaps, leading to a negligible amount of leakage

Both the ***Random*** and ***Noise*** ablations show decreasing gaps when the last three concepts are added, showing a deterioration in the carried information about labels. This naturally happens because little to none information about the label is kept in the model concepts, especially when considering the Random baseline and highly noisy concepts ($\sigma^2 = 1$). Conversely, the ***Leak*** ablations display a drastic increase when lowering the noise variance σ^2. With variance $\sigma^2 = 1$, labels can be accurately predicted from the first model concepts, reaching LEAK $\approx 97\%$. This is also as expected, since model concepts are very informative about the labels. The gaps in Fig. 8 are sensibly higher than all other 3 variants for all subsets of concepts. For the ***Entangled*** ablations, we observe that varying the value of λ does not sensibly affect the score on LEAK, which remains above 57% for all of them. This is as expected, since even small values of λ result in spreading label information across the bottleneck, which can be flexibly retrieved by training SVMs. Notice that $\lambda = 0$ would give LEAK equal to zero, because model concepts coincide with ground-truth ones. As expected, mixing ground-truth concepts and entangled ones (through the linear combination with O) yields a sensible amount of leakage. Interestingly, the gaps show a positive increase at the beginning, reflecting that a leak of information about labels is gained when including more entangled concepts, and the information about labels is spread among the model concepts.

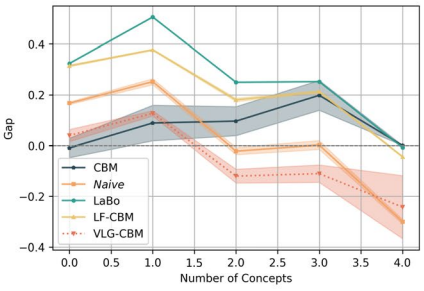
Appendix D: Additional Experimental Results

Additional Results on Leakage and Visualization of G-DINO

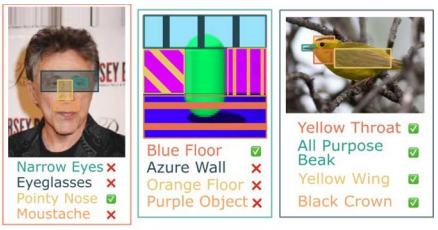
We report in Fig. 9a the gaps for models trained on Shapes3d. The plot shows a high amount of leakage for LaBo and LF-CBM, while LEAK in CBM remains low. We observe that the Naïve and VLG-CBM have often negative gaps, because the learned concepts are not as predictive for labels as the ground-truth ones. We also report in Fig. 9b the mask and concept annotations obtained with G-DINO on all datasets.

Correlations Among Metrics

We also evaluate the Spearman correlation coefficient across metrics, but specialized for datasets, see Fig. 10. The results reveal that AUC(C), DIS, and OIS are highly correlated



(a) Gaps in concept leakage test on Shapes3d dataset. Negative values indicate that the classifier using predicted concepts has lower F_1 score than the same which has access to ground-truth concept annotations.



(b) Annotations obtained using G-DINO on the 3 different datasets. On CelebA (left) and Shapes3d (center) the VLM makes several mistakes highlighted by red crosses. On the other hand, the model does a good job annotating CUB (right).

Fig. 9 **a** Gaps in concept leakage test on Shapes3d dataset. Negative values indicate that the classifier using predicted concepts has lower F_1 score than the same which has access to ground-truth concept annotations. **b** Annotations obtained using G-DINO on the 3 different datasets. On CelebA (left) and Shapes3d (center) the VLM makes several mistakes highlighted by red crosses. On the other hand, the model does a good job annotating CUB (right)

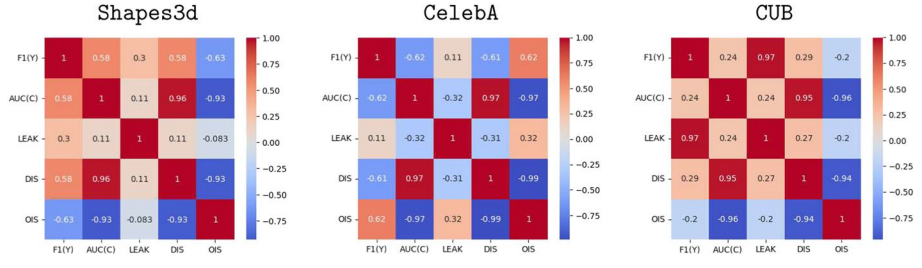


Fig. 10 Correlations between metrics across datasets. We evaluate the Spearman correlation coefficient between metrics for a model trained on separate datasets. Each evaluation is done considering 25 points

in all datasets. On the other hand, it appears that $F_1(Y)$ seems positively (negatively) correlated to $\mathrm{AUC}(C)$ in `Shapes3d` (resp. `CelebA`). We also observe that trained models in `CUB` manifest a high correlation between $F_1(Y)$ and LEAK.

Disentanglement

We provide in Figs. 11, 12, and 13 the DCI matrices obtained from the different models on `Shapes3d`, `CelebA`, and `CUB`, respectively. The images represent the $k \times k$ DCI matrices, where k is the number of concepts. Each row refers to the importances of a distinct learned concept, whereas columns refer to distinct ground-truth concepts. E.g., the leftmost pixel at the top represents the importance that the first learned concept has in predicting the first ground-truth concept. A column of high activated pixels is a signal that a specific ground-truth concept is encoded in multiple learned concepts. On the other hand, a row of high activated pixels shows that the learned concept is entangled as it encodes information of multiples ground-truth concepts.

We note that DCI matrices of `LF-CBM` and `LaBo` are very similar in all datasets, because they both exploit the CLIP similarities to supervise the concepts. We detail below specific aspects of the importance matrices across datasets and methods.

- **Shapes3d** (Fig. 11). The **CelebA** importance matrix exhibits the intended diagonal pattern, showing that learned concepts are highly disentangled. `Naïve` does not learn

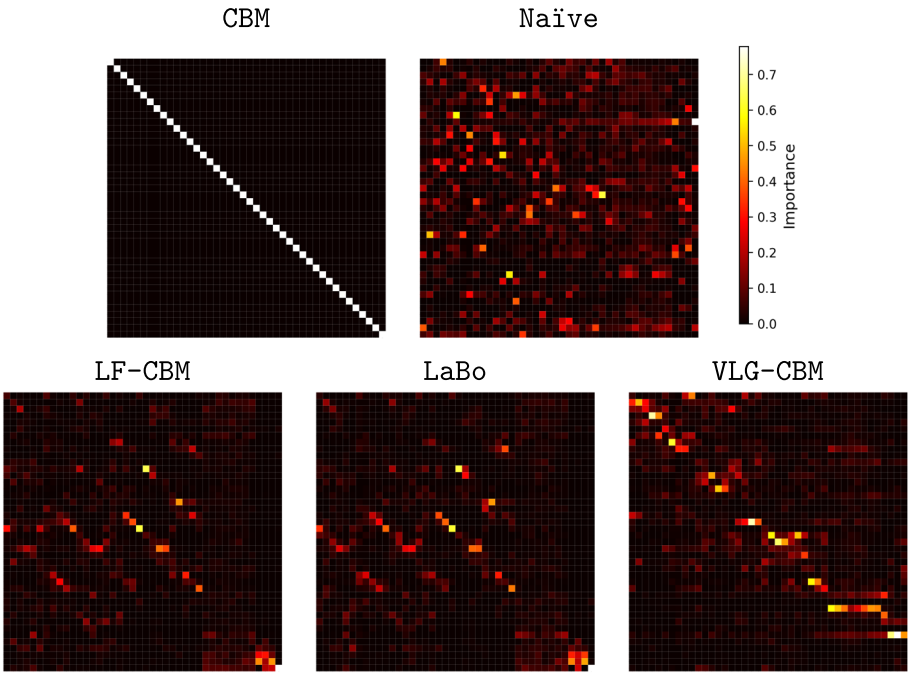


Fig. 11 Importance matrices in DCI on **Shapes3d**. High disentanglement is achieved when, for each row, only one square is active. For reference, a perfect DCI matrix is a diagonal matrix

meaningful concepts as shown from off-diagonal importance values. Moreover, there is no one-to-one link between learned and ground-truth concepts. `LF-CBM` and `LaBo` show similar importance matrices, with entanglement on several concepts (e.g., row 24 *"green object"* encodes also variations of *"green floor"*, *"lime floor"*, *"green wall"*, *"lime wall"*,*"green object"*, and *"lime object"*). From the mostly plain dark columns at the right, we see that learned concepts do not encode much information about the object shape (concepts $30-38$). In `VLG-CBM`, the learned concepts are less entangled (except for the concept 33, which brings information about all shapes) and the blank 5 bottom rows shows that those learned concepts do not contain any information about ground-truth concepts.

- CelebA (Fig. 12) **CBM** has disentangled concepts. **Naïve** presents some entangled concepts (e.g. c. 36 *"Wearing lipstick"* that encodes for *"Arched eyebrows"*,*"Arched eyebrows"*, *"Pointy nose"*,*"Rosy cheeks"* and *"Wearing necklace"*). The first two learned concepts (top 2 rows) are not influenced by any ground-truth concept, and this is true also for other rows in the matrix. Overall, this means that most concepts are not meaningful as they are either entangled or not encoding anything, while few ones are correctly linked to their ground-truth concept. `LF-CBM` and `LaBo` do have similar pattern, featuring a high entangled concept 24 *"No Beard"* and some few other less entangled concepts. There are also a lot of high importance pixels out of the diagonal indicating that some concepts are not correctly aligned to their ground-truth counterpart. `VLG-CBM` shows a very bad DCI matrix where there is no diagonal pattern and multiple rows present high value pixels, which means entangled concepts.

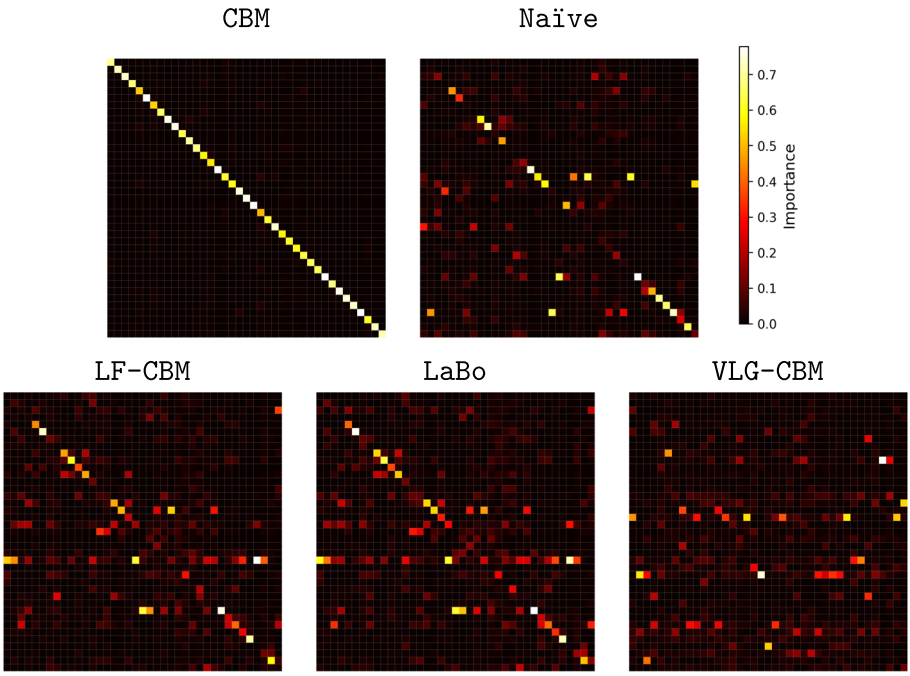


Fig. 12 Importance matrices in DCI on `CelebA`. High disentanglement is achieved when, for each row, only one square is active. For reference, a perfect DCI matrix is a diagonal matrix

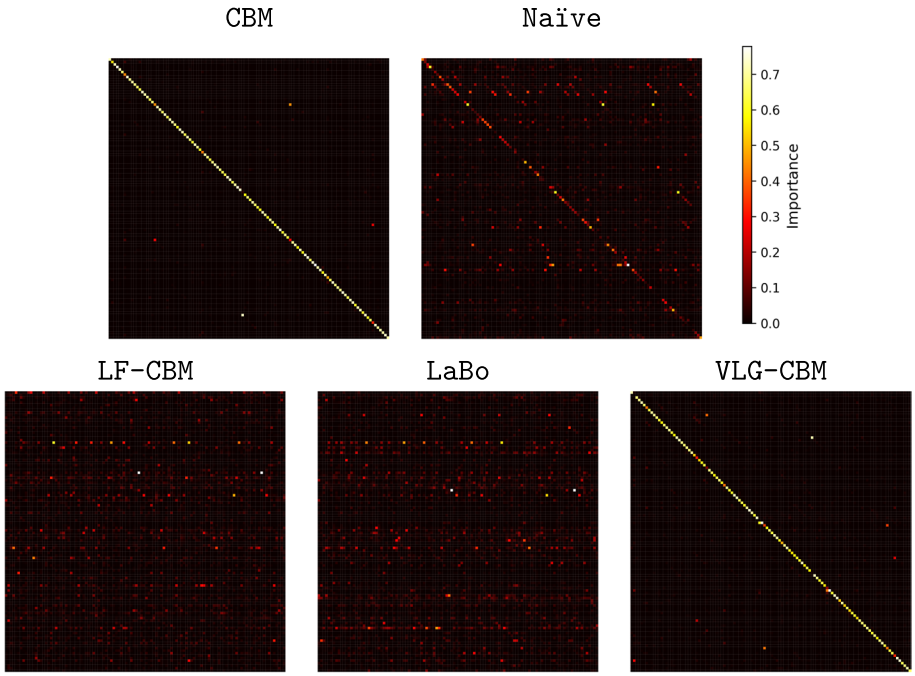


Fig. 13 Importance matrices in DCI on **CUB**. High disentanglement is achieved when, for each row, only one square is active. For reference, a perfect DCI matrix is a diagonal matrix

- CUB (Fig. 13) `CBM` and `VLG-CBM` are very similar and almost match a diagonal matrix, indicating that concepts are mostly disentangled. `Naïve` does have few entangled concepts especially in the top rows, however a diagonal shape is still partially visible. `LF-CBM` and `LaBo` only show pixels activating in horizontal lines, indicating high entanglement on almost all concepts.

Acknowledgements The authors are grateful to the anonymous reviewer, whose comments strongly improved the current manuscript. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HaDEA). Neither the European Union nor the granting authority can be held responsible for them. Grant Agreement no. 101120763 - TANGO.

Funding Open access funding provided by Università degli Studi di Trento within the CRUI-CARE Agreement.

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

Abbasi Yadkori, Y., et al. (2024). To believe or not to believe your LLM: Iterative prompting for estimating epistemic uncertainty. NeurIPS
Alvarez Melis, D., & Jaakkola, T. (2018). Towards robust interpretability with self-explaining neural networks. NeurIPS
Ansel, J., et al. (2024). PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In: ASPLOS
Bahadori, M.T., & Heckerman, D. (2021). Debiasing concept-based explanations with causal analysis. In: ICLR
Barbiero, P., et al. (2022). Entropy-based logic explanations of neural networks. In: AAAI
Barbiero, P., et al. (2023). Interpretable neural-symbolic concept reasoning. In: ICML
Barbiero, P., et al. (2024). Relational concept bottleneck models. NeurIPS
Barbiero, P., et al. (2025). Neural interpretable reasoning. arXiv:2502.11639
Bontempelli, A., et al. (2023) Concept-level debugging of part-prototype networks. In: ICLR
Bortolotti, S., et al. (2025). Shortcuts and identifiability in concept-based models from a neuro-symbolic lens. arXiv:2502.11245
Breiman, L., et al. (1984). Classification and regression trees. CRC Press
Calanzone, D., et al. (2025). Logically consistent language models via neuro-symbolic integration. In: ICLR
Chauhan, K., et al. (2023). Interactive concept bottleneck models. In: AAAI
Chen, C., et al. (2019). This looks like that: deep learning for interpretable image recognition. NeurIPS
Chen, Z., et al. (2020). Concept whitening for interpretable image recognition. Nature Machine Intelligence
Contributors, X. (2023). XTuner: A toolkit for efficiently fine-tuning LLM. https://github.com/InternLM/xtuner
Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning
De Felice, G., et al. (2025). Causally reliable concept bottleneck models. arXiv:2503.04363
Debot, D., et al. (2024). Interpretable concept-based memory reasoning. arXiv:2407.15527
Dominici, G., et al. (2024a). AnyCBMs: How to turn any black box into a concept bottleneck model
Dominici, G., et al. (2024b). Causal concept graph models: Beyond causal opacity in deep learning. arXiv:2405.16507
Dominici, G., et al. (2024c). Counterfactual concept bottleneck models. arXiv:2402.01408
Eastwood, C., & Williams, C.K. (2018). A framework for the quantitative evaluation of disentangled representations. In: ICLR
Espinosa Zarlenga, M., et al. (2023). Learning to receive help: Intervention-aware concept embedding models. NeurIPS
Espinosa Zarlenga, M., et al. (2024). Learning to receive help: Intervention-aware concept embedding models. NeurIPS
Feng, J., et al. (2024). Bayesian concept bottleneck models with llm priors. arXiv:2410.15555
Fokkema, H., et al. (2025). Sample-efficient learning of concepts with theoretical guarantees: From data to concepts without interventions. arXiv:2502.06536
Furby, J., et al. (2023). Towards a deeper understanding of concept bottleneck models through end-to-end explanation. In: Workshop on representation learning for responsible human-centric AI @ AAAI
Grattafiori, A., et al. (2024). The llama 3 herd of models. arXiv preprint arXiv:2407.21783
Havasi, M., et al. (2022). Addressing leakage in concept bottleneck models. In: NeurIPS
He, K., et al. (2016). Deep residual learning for image recognition. In: CVPR
Higgins, I., et al. (2018). Towards a definition of disentangled representations. arXiv:1812.02230
Huang, L., et al. (2023). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. ACM TOIS
Hurst, A., et al. (2024). Gpt-4o system card. arXiv:2410.21276
Ismail, A.A., et al. (2023). Concept bottleneck generative models. In: ICLR
Ji, Y., et al. (2025). A comprehensive survey on self-interpretable neural networks. arXiv:2501.15638
Kazhdan, D., et al. (2021). Is disentanglement all you need? Comparing concept-based & disentanglement approaches. arXiv:2104.06917
Kim, H., & Mnih, A. (2018). Disentangling by factorising. In: ICML

Kim, B., et al. (2018). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors. In: ICML
Kim, E., et al. (2023). Probabilistic concept bottleneck models. In: ICML
Kingma, D.P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv:1412.6980
Koh, P.W., et al. (2020). Concept bottleneck models. In: ICML
Laguna, S., et al. (2024). Beyond concept bottleneck models: How to make black boxes intervenable? NeurIPS
Lai, S., et al. (2024). Faithful vision-language interpretation via concept bottleneck models. In: ICLR
Lertvittayakumjorn, P., et al. (2020) Find: human-in-the-loop debugging deep text classifiers. In: EMNLP
Li, O., et al. (2018). Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In: AAAI
Li, S., et al. (2024). On erroneous agreements of clip image embeddings. arXiv:2411.05195
Liu, Z., et al. (2015). Deep learning face attributes in the wild. In: ICCV
Liu, S., et al. (2024). Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In: ECCV
Lockhart, J., et al. (2022). Towards learning to explain with concept bottleneck models: Mitigating information leakage. arXiv:2211.03656
Mahinpei, A., et al. (2021). Promises and pitfalls of black-box concept learning models. In: Workshop on theoretic foundation, criticism, and application trend of explainable AI @ ICML
Marconato, E., et al. (2022). *GlanceNets: Interpretabile*. Leak-proof Concept-based Models. In: NeurIPS.
Marconato, E., et al. (2023). Interpretability is in the mind of the beholder: A causal framework for human-interpretable representation learning. Entropy
Marconato, E., et al. (2024). Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts. NeurIPS
Margeloiu, A., et al. (2021). Do concept bottleneck models learn as intended? arXiv:2105.04289
Mikriukov, G., et al. (2023). Evaluating the stability of semantic concept representations in CNNs for robust explainability. In: World conference on explainable artificial intelligence
Montero, M., et al. (2022). Lost in latent space: Examining failures of disentangled models at combinatorial generalisation. NeurIPS
Moreira, R., et al. (2024). Diconstruct: Causal concept-based explanations through black-box distillation. arXiv:2401.08534
Oikarinen, T., et al. (2023). Label-free concept bottleneck models. In: ICLR
Poeta, E., et al. (2023). Concept-based explainable artificial intelligence: A survey. arXiv:2312.12936
Radford, A., et al. (2021). Learning transferable visual models from natural language supervision. In: ICML
Rajendran, G., et al. (2024). From causal to concept-based representation learning. NeurIPS
Raman, N., et al. (2023). Do concept bottleneck models obey locality? In: XAI in action: Past, present, and future applications
Rao, S., et al. (2024). Discover-then-name: Task-agnostic concept bottlenecks via automated concept discovery
Sahu, P., et al. (2022). Unpacking large language models with conceptual consistency. arXiv:2209.15093
Sawada, Y., & Nakamura, K. (2022a). Concept bottleneck model with additional unsupervised concepts. IEEE Access
Sawada, Y., & Nakamura, K. (2022b). C-senn: Contrastive self-explaining neural network. arXiv:2206.09575
Schölkopf, B., et al. (2000). New support vector algorithms. Neural Computation
Schölkopf, B., et al. (2021). Toward causal representation learning. IEEE
Schrodi, S., et al. (2024). Concept bottleneck models without predefined concepts. arXiv:2407.03921
Schwalbe, G. (2022). Concept embedding analysis: A review. arXiv:2203.13909
Shin, S., et al. (2023). A closer look at the intervention procedure of concept bottleneck models. In: ICML
Srivastava, D., et al. (2024). VLG-CBM: Training concept bottleneck models with vision-language guidance. In: NeurIPS
Stammer, W., et al. (2021). Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. In: CVPR
Steinmann, D., et al. (2024). Learning to intervene on concept bottlenecks. In: ICML
Suter, R., et al. (2019). Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In: ICML
Teso, S., et al. (2023). Leveraging explanations in interactive machine learning: An overview. Frontiers in Artificial Intelligence
Vandenhirtz, M., et al. (2024) Stochastic concept bottleneck models. arXiv:2406.19272
Wah, C., et al. (2011). The caltech-ucsd birds-200-2011 dataset
Wong, E., et al. (2021). Leveraging sparse linear layers for debuggable deep networks. In: ICML

Yang, Y., et al. (2023). Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In: CVPR
Yeh, C.-K., et al. (2020). On completeness-aware concept-based explanations in deep neural networks. NeurIPS
Yuan, Y., et al. (2024). Do LLMs overcome shortcut learning? An evaluation of shortcut challenges in large language models. arXiv:2410.13343
Yuksekgonul, M., et al. (2023). Post-hoc concept bottleneck models. In: ICLR
Zarlenga, M.E., et al. (2022). Concept embedding models: Beyond the accuracy-explainability trade-off. In: NeurIPS
Zarlenga, M.E., et al. (2023). Towards robust metrics for concept representation evaluation. In: AAAI
Zhang, R., et al. (2024). The decoupling concept bottleneck model. PAMI

Authors and Affiliations

Nicola Debole[1] · Pietro Barbiero[2] · Francesco Giannini[3] · Andrea Passerini[1] · Stefano Teso[1,4] · Emanuele Marconato[1]

✉ Emanuele Marconato
emanuele.marconato@unitn.it

Nicola Debole
nicola.debole@unitn.it

Pietro Barbiero
pietro.barbiero@ibm.com

Francesco Giannini
francesco.giannini@sns.it

Andrea Passerini
andrea.passerini@unitn.it

Stefano Teso
stefano.teso@unitn.it

[1] DISI, University of Trento, Trento, Italy

[2] IBM Research Zurich, Rüschlikon, Switzerland

[3] DI, University of Pisa, Pisa, Italy

[4] CIMeC, University of Trento, Trento, Italy