GlanceNets: Interpretable, Leak-proof Concept-based Models

Emanuele Marconato
Department of Computer Science
University of Pisa & University of Trento
Pisa, Italy
emanuele.marconato@unitn.it

Andrea Passerini
Department of Computer Science
University of Trento
Trento, Italy
andrea.passerini@unitn.it

Stefano Teso
Department of Computer Science
University of Trento
Trento, Italy
stefano.teso@unitn.it

Abstract

There is growing interest in concept-based models (CBMs) that combine high-performance and interpretability by acquiring and reasoning with a vocabulary of high-level concepts. A key requirement is that the concepts be interpretable. Existing CBMs tackle this desideratum using a variety of heuristics based on unclear notions of interpretability, and fail to acquire concepts with the intended semantics. We address this by providing a clear definition of interpretability in terms of alignment between the model's representation and an underlying data generation process, and introduce GlanceNets, a new CBM that exploits techniques from disentangled representation learning and open-set recognition to achieve alignment, thus improving the interpretability of the learned concepts. We show that GlanceNets, paired with concept-level supervision, achieve better alignment than state-of-the-art approaches while preventing spurious concepts from unintentionally affecting its predictions.

1 Introduction

Concept-based models (CBMs) are an increasingly popular family of classifiers that combine the transparency of white-box models with the flexibility and accuracy of regular neural nets [1–5]. At their core, all CBMs acquire a vocabulary of concepts capturing high-level, task-relevant properties of the data, and use it to compute predictions and produce faithful explanations of their decisions [6].

The central issue in CBMs is how to ensure that the concepts are *semantically meaningful* and *interpretable* for (sufficiently expert and motivated) human stakeholders. Current approaches struggle with this. One reason is that the notion of interpretability is notoriously challenging to pin down, and therefore existing CBMs rely on different heuristics—such as encouraging the concepts to be sparse [1], orthonormal to each other [5], or match the contents of concrete examples [3]—with unclear properties and incompatible goals. A second, equally important issue is *concept leakage*, whereby the learned concepts end up encoding spurious information about unrelated aspects of the data, making it hard to assign them clear semantics [7]. Notably, even concept-level supervision is insufficient to prevent leakage [8], cf. Fig. 1.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

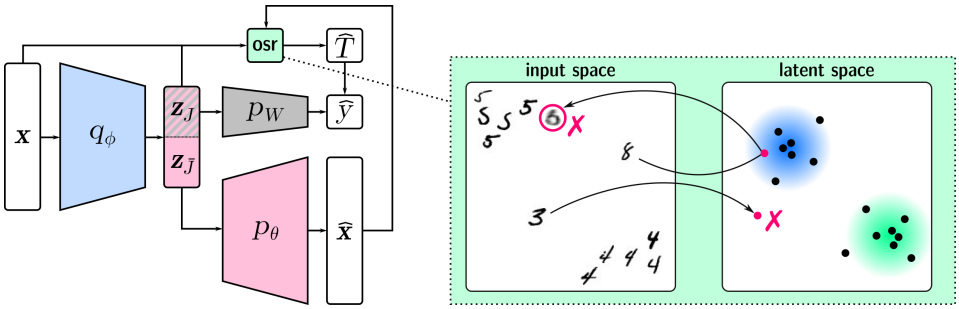


Figure 1: **Left**: Architecture of GlanceNets showing the encoder q_ϕ, decoder p_θ, classifier p_W, and open-set recognition step. **Right**: At test time, GlanceNets prevent leakage by identifying and rejecting out-of-distribution inputs using a combined strategy, shown here for a model trained on digits "4" and "5" only: the "3" is rejected as its embedding falls far away from prototypes of the two training classes (colored blobs), while the "8" is rejected as its reconstruction loss is too large.

Prompted by these observations, we define interpretability in terms of *alignment*: learned concepts are interpretable if they can be mapped to a (partially) interpretable data generation process using a transformation that preserves semantics. This is sufficient to unveil limitations in existing strategies, build an explicit link between interpretability and disentangled representations, and provide a clear and actionable perspective on concept leakage. Building on our analysis, we also introduce GlanceNets (aliGned LeAk-proof coNCEptual Networks), a novel class of CBMs that combine techniques from *disentangled representation learning* [9] and *open-set recognition* (OSR) [10] to actively pursue alignment – and guarantee it under suitable assumptions – and avoid concept leakage.

Contributions: Summarizing, we: (*i*) Provide a definition of interpretability as alignment that facilitates tapping into ideas from disentangled representation learning; (*ii*) Show that concept leakage can be viewed from the perspective of out-of-distribution generalization; (*iii*) Introduce GlanceNets, a novel class of CBMs that acquire interpretable representations and are robust to concept leakage; (*iv*) Present an extensive empirical evaluation showing that GlanceNets are as accurate as state-of-the-art CBMs while attaining better interpretability and avoiding leakage.

2 Concept-based Models: Interpretability and Concept Leakage

Concept-based models (CBMs) comprise two key elements: (i) A learned vocabulary of k high-level concepts meant to enable communication with human stakeholders [11], and (ii) a simulatable [12] classifier whose predictions depend solely on those concepts. Formally, a CBM $f : \mathbb{R}^d \to [c]$, with $[c] := \{1, \ldots, c\}$, maps instances $\mathbf{x}$ to labels y by measuring how much each concept activates on the input, obtaining an activation vector $\mathbf{z}(\mathbf{x}) := (z_1(\mathbf{x}), \ldots, z_k(\mathbf{x}))^T \in \mathbb{R}^k$, aggregating the activations into per-class scores $s_y(\mathbf{x})$ using a linear map [1, 3, 5], and then passing these through a softmax, i.e.,

$$s_y(\mathbf{x}) := \textstyle\sum_j w_{yj} \cdot z_j(\mathbf{x}), \qquad p(y \mid \mathbf{x}) := \mathrm{softmax}(\mathbf{s}(\mathbf{x}))_y. \tag{1}$$

Each weight $w_{yj} \in \mathbb{R}$ encodes the relevance of concept z_j for class y. The activations themselves are computed in a black-box manner, often leveraging pre-trained embedding layers, but learned so as to capture interpretable aspects of the data using a variety of heuristics, discussed below.

Now, *as long as the concepts are interpretable*, it is straightforward to extract human understandable local explanations disclosing how different concepts contributed to any given decision $(\mathbf{x}, y)$ by looking at the concept activations and their associated weights, thus abstracting away the underlying computations. This yields explanations of the form $\{(w_{yj}, z_j(\mathbf{x})) : j \in [k]\}$ that can be readily summarized[1] and visualized [13, 14]. Importantly, the score of class y is conditionally independent from the input $\mathbf{x}$ given the corresponding explanation, i.e., $s_y(\mathbf{x}) \perp\!\!\!\perp \mathbf{x} \mid \mathcal{E}(\mathbf{x}, y)$, ensuring that the latter is faithful to the model scores. GlanceNets inherit all of these features.

[1]For instance, by pruning those concepts that have little effect on the outcome to simplify the presentation.

Heuristics for interpretability. Crucially, CBMs are only interpretable insofar as their concepts are. Existing approaches implement special mechanisms to this effect, often pairing a traditional classification loss (such as the cross-entropy loss) with an auxiliary regularization term.

Alvarez-Melis and Jaakkola [1] acquire concepts using an autoencoder augmented with a sparsification penalty encouraging distinct concepts to activate on different instances. Chen et al. [5] apply geometric transforms to learn mutually orthonormal concepts that thus encode complementary information and attain comparable activation ranges. These mechanisms – sparsity and orthogonality, respectively – alone cannot prevent capturing features that are not semantic in nature.

A second group of CBMs tackle this issue by constraining the concepts to match *concrete* cases, in the hope that these are better aligned with human intuition [15]. For instance, prototype classification networks [2], part-prototype networks [3], and related approaches [16–18] model concepts using prototypes in embedding space that perfectly match training examples or parts thereof. Depending on the embedding space, which ultimately determines the distance to the prototypes, concepts learned this way may activate on elements unrelated to the example they match, leading to unclear semantics [19].

Closest to our work, concept bottleneck models (CBNMs) [20, 4] align the concepts using concept-level supervision – possibly obtained from a separate source, like ImageNet [21] – either sequentially or in tandem with the top-level dense layer. From a statistical perspective, this seems perfectly sensible: if the supervision is unbiased and comes in sufficient quantity, and the model has enough capacity, this strategy *appears* to guarantee the learned and ground-truth concepts to match.

Concept leakage in concept-bottleneck models. Unfortunately, concept-level supervision is *not* sufficient to guarantee interpretability. Mahinpei et al. [7] have demonstrated that concepts acquired by CBNMs pick up spurious properties of the data. In their experiment, they learn two concepts z_4 and z_5, meant to represent the 4 and 5 MNIST digits, using concept-level supervision, and then show that – surprisingly – these concepts can be used to classify all *other* digits (i.e., MNIST images that are neither 4's nor 5's) as even or odd significantly better than random guessing. This phenomenon, whereby learned concepts unintentionally capture information about unobserved concepts, is known as *concept leakage*.

Intuitively, leakage occurs because in CBNMs the concepts end up unintentionally capturing distributional information about unobserved aspects of the input, failing to provide well-defined semantics. However, a clear definition of leakage is missing, and so are strategies to prevent it. In fact, separating concept learning from classification and increasing the amount of supervision for the observed concepts (here, 4 and 5) is not enough [8]. A key contribution of our paper is showing that leakage can be understood from the perspective of domain shift and dealt with using open-set recognition [10].

3 Disentangling Interpretability and Concept Leakage

The main issue with heuristics used by CBMs is that they are based on unclear notions of interpretability. In order to develop effective algorithms, we propose to view interpretability as a form of *alignment* between the machine's representation and that of its user. This enables us to identify conditions under which interpretability can be achieved, build links to well-understood properties of representations, and leverage state-of-the-art learning strategies.

Interpretability. We henceforth focus on the (rather general) generative process shown in Fig. 2: the observations $\mathbf{X} \in \mathbb{R}^d$ are caused by n generative factors $\mathbf{G} \in \mathbb{R}^n$, themselves caused by a set of confounds $\mathbf{C}$ (including the label Y [22]). Notice that the generative factors *can* be statistically dependent due to the confounds $\mathbf{C}$, but as noted by Suter et al. [23], the total causal effect [24, Def. 6.12] between G_i and G_j is zero for all $i \neq j$. The generative factors capture all information necessary to determine the observation [23, 25], so the goal is to learn concepts $\mathbf{Z} \in \mathbb{R}^k$ that recover them. The variable T will be introduced later on.

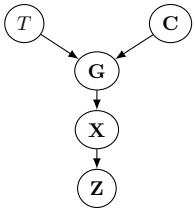


Figure 2: The data generation process.

We posit that a (learned) representation is only interpretable if it supports *symbolic communication* between the model and the user, in the sense that it shares the same (or

similar enough) semantics to the user's representation. The latter is however generally unobserved. We therefore make a second, critical assumption that *some* of the generative factors $\mathbf{G}_I \subseteq \mathbf{G}$ are interpretable to the user, meaning that they can be used as a proxy for the user's internal representation. Naturally, not all generative factors are interpretable [26], but in many applications some of them are. For instance, in dSprites [27] the generative factors encode the position, shape and color of a 2D object, and in CelebA [28] the hair color and nose size of a celebrity. Human observers have a good grasp of such concepts.

Interpretability as alignment. Under this assumption, if the variables $\mathbf{Z}_J \subseteq \mathbf{Z}$ are *aligned* to the generative factors $\mathbf{G}_I$ by a map $\alpha : \mathbf{g} \mapsto \mathbf{z}_J$ that preserves semantics, they are themselves interpretable. Now, defining what a semantics-preserving map should look like is challenging, but constructing one is not: the identity is clearly one such map, and so are maps that permute the indices and independently rescale the individual variables. One desirable property is that α does not "mix" multiple G's into a single Z. E.g., if Z blends together head tilt, hair color, and nose size, users will have trouble pinning down what it means.[2] This property can be formalized in terms of *disentanglement* [29, 23, 9]. This is however insufficient: we wish the map between G_i and its associated factor Z_j to be "simple", so as to *conservatively* guarantee that it preserves semantics. This makes alignment strictly stronger than disentanglement.

Motivated by these desiderata, we say that $\mathbf{Z}_J$ is *aligned* to $\mathbf{G}_I$ if it satisfies:

(*i*) **Disentanglement.** There exists an injective map between indices $\pi : [n_I] \to [k]$, where $[n_I]$ identifies the subset of generative factors indexes in $\mathbf{G}_I$, such that, for all $i, i' \in [n_I]$, $i \neq i'$, and $j = \pi(i)$, it holds that fixing G_i is enough to fix Z_j regardless of the value taken by the other generative factors $G_{i'}$, and

(*ii*) **Monotonicity.** The map α can be written as $\alpha(\mathbf{g}) = (\mu_1(g_{\pi(1)}), \ldots, \mu_n(g_{\pi(n_I)}))^T$, where the μ_i's are monotonic transformations. This holds, for instance, for linear transformations of the form $A\ (g_{\pi(1)}, \ldots, g_{\pi(n_I)})^T$, where $A \in \mathbb{R}^{n_I \times k}$ is a matrix with no non-zero off-diagonal entries. This second requirement can be relaxed depending on the application.

Notice that we do not require each G_i to map to a *single* Z_j (a property known as *completeness* [29]): $\mathbf{Z}_J$ is interpretable even if it contains multiple – perhaps slightly different, but aligned – transformations of the same G_i.

Measuring alignment with DCI. Disentanglement can be measured in a number of ways [30], but most of them provide little information about how simple the map α is. In order to estimate alignment, we repurpose DCI, a measure of disentanglement introduced by Eastwood and Williams [29], see also Appendix B. According to this metric, a representation $\mathbf{Z}_J$ is disentangled if there exists a regressor that, given $\mathbf{z}_J$, can predict $\mathbf{g}_I$ with high accuracy using few z_i's to predict each g_i. Following [29], we use a linear regressor with parameters $B \in \mathbb{R}^{k \times n_I}$ on the test set – assuming that it is annotated with the interpretable generative factors and corresponding learned representations – and then measure how diffuse the weights associated to each latent factor are. We do this by normalizing them and computing their average Shannon entropy over all G_i's, i.e.,

$$-\sum_{j \in [k]} \rho_j \left(\sum_{i \in [n_I]} \bar{b}_{ji} \log \bar{b}_{ji} \right), \quad \text{where } \bar{b}_{ji} = b_{ji} / \sum_{j' \in [k]} b_{j'i} \quad \text{and} \quad \rho_j = \sum_i b_{ji} / \sum_{j'i} b_{j'i} \tag{2}$$

Hence, DCI gauges the degree of mixing that a linear map can attain using the learned representation $\mathbf{Z}$, and as such it indirectly measures alignment, with B approximating the inverse of A.

Achieving alignment with concept-level supervision. It has been shown that disentanglement cannot be achieved in the purely unsupervised setting [31]. This immediately entails that alignment is also impossible in that setting, highlighting a core limitation of approaches like self-explainable neural networks [1]. However, disentanglement can be attained if supervision about the generative factors is available, even only for a small percentage of the examples [32]. As a matter of fact, supervision is used in representation learning to achieve *identifiability*, a stronger condition than – and that entails both of – disentanglement *and* alignment [33]. Thus, following CBNMs, we seek alignment by leveraging concept-level supervision.

[2] The converse is not true: interpretable concepts with *compatible* semantics can be mixed without compromising interpretability. E.g., rotating a coordinate system gives another intuitive coordinate system. Our point is that conservatively avoiding mixing helps to preserve semantics.

Interpretability and concept leakage. Intuitively, concept leakage occurs when a model is trained on a data set on which (*i*) some generative factors $\mathbf{G}_V \subset \mathbf{G}$ vary, while the others $\mathbf{G}_F = \mathbf{G} \setminus \mathbf{G}_V$ are fixed, and (*ii*) the two groups of factors are statistically dependent. For instance, in the even vs. odd experiment 4 and 5 play the role of $\mathbf{G}_V$ and the other digits of $\mathbf{G}_F$. CBNMs with access to supervision on $\mathbf{G}_V$ tend to acquire a latent representation that approximates these factors. But, because of (*ii*), this representation correlates with the fixed factors $\mathbf{G}_F$. This immediately explains why additional supervision on $\mathbf{G}_V$ cannot prevent leakage, but rather has the opposite effect: the better a latent representation matches $\mathbf{G}_V$, the more information it conveys about $\mathbf{G}_F$.

In contrast with previous assessments [7, 8], we notice that this phenomenon can be viewed as a special form of domain shift: the training examples are sampled from a ground-truth distribution $p(\mathbf{X}, \mathbf{G} \mid T = 1)$ in which $\mathbf{G}_F$ is approximately fixed, e.g., $p(\mathbf{G}_F \mid T = 1) = \delta(\mathbf{g}'_F)$ for some vector $\mathbf{g}'_F$, while in the test set, the data is sampled from a different distribution $p(\mathbf{X}, \mathbf{G} \mid T = 0)$ in which $\mathbf{G}_F$ is no longer fixed. In the MNIST task, for instance, when $T = 1$ no concept besides 4 and 5 can occur, while all concepts *except* 4 and 5 can occur when $T = 0$. Here, $T \in \{0, 1\}$ selects between training and test distribution, see Fig. 2. Now, CBMs have no strategy to cope with domain shift and thus cannot disambiguate between known training and unknown test concepts.

Motivated by this, we propose then to tackle concept leakage by designing a CBM specifically equipped with strategies for detecting – at *inference* time – instances that do not belong to the training distribution using OSR [10]. The idea is to estimate the value of the variable T at inference time, essentially predicting whether an input was sampled from a distribution similar enough to the training distribution, and therefore can be handled by a model learned on this distribution, or not. This strategy proves very effective in practice, as shown by our empirical evaluation (Section 5.2).

4 Addressing Alignment and Leakage with GlanceNets

GlanceNets combine a VAE-like architecture [34, 35] for learning disentangled concepts with a prior and classifier designed for open-set prediction [36]. In order to accommodate for non-interpretable factors, the latent representation of GlanceNets $\mathbf{Z}$ is split into two: (i) k concepts $\mathbf{Z}_J$, aligned to the *interpretable* generative factors $\mathbf{G}_I$, that are used for prediction, and (ii) $\bar{k}$ *opaque* factors $\mathbf{Z}_{\bar{J}}$ that are only used for reconstruction. Specifically, a GlanceNet comprises an encoder $q_\phi(\mathbf{Z} \mid \mathbf{X})$ and a decoder $p_\theta(\mathbf{X} \mid \mathbf{Z})$, both parameterized by deep neural networks, as well as a classifier $p_W(Y \mid \mathbf{Z}_J)$ feeding off the interpretable concepts only. The overall architecture is shown in Fig. 1.

Following other CBMs, the classifier is implemented using a dense layer with parameters $W \in \mathbb{R}^{v \times k}$ followed by a softmax activation, i.e., $p_W(Y \mid \mathbf{z}_J) := \mathrm{softmax}(W\mathbf{z}_J)$, and the most likely label is used for prediction. The class distribution is obtained by marginalizing over the encoder's distribution:

$$p(Y \mid \mathbf{x}) := \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[p(Y \mid \mathbf{z}, \mathbf{x})] = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[p_W(Y \mid \mathbf{z}_J)] \tag{3}$$

Equality holds because $Y \perp\!\!\!\perp \mathbf{X} \mid \mathbf{Z}_J$. In order to expedite the computation, we follow the general practice of approximating the integral as $\mathrm{softmax}(W\,\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\mathbf{z}_J]) = \mathrm{softmax}(W[\boldsymbol{\mu}_\phi(\mathbf{x})]_J)$.

In contrast to regular VAEs, GlanceNets associate each class to a prototype in latent space through the prior $p(\mathbf{Z} \mid \mathbf{Y})$, which is conditioned on the class and modelled as a *mixture of gaussians* with one component per class. The encoder, decoder, and prior are fit on data so as to maximize the evidence lower bound [37], defined as $\mathbb{E}_{p_D(\mathbf{x},y)}[\mathcal{L}(\theta, \mathbf{x}, y; \beta)]$ with:

$$\mathcal{L}(\theta, \mathbf{x}, y; \beta) := \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z}) + \log p_W(y \mid \mathbf{z}_J)] - \beta \cdot \mathsf{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z} \,|\, y)) \tag{4}$$

Here, $p_D(\mathbf{x}, y)$ is the empirical distribution of the training set $D = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, m\}$. The first term of Eq. (4) is the likelihood of an example after passing it through the encoder distribution.

The second term penalizes the latent vectors based on how much their distribution differs from the prior and encourages disentanglement. As mentioned in Section 3, learning disentangled representations is impossible in the unsupervised i.i.d. setting [31]. Following Locatello et al. [32], and similarly to CBNMs, we assume access to a (possibly separate) data set $\tilde{D} = \{(\mathbf{x}_\ell, \mathbf{g}_{I,\ell})\}$ containing supervision about the *interpretable* generative factors $\mathbf{G}_I$ and integrate it into the ELBO by replacing the per-example loss $\mathcal{L}$ in Eq. (4) with:

$$\mathbb{E}_{p_{\mathcal{D}}(\mathbf{x},y)}\big[\mathcal{L}(\theta, \mathbf{x}, y; \beta)\big] + \gamma \cdot \mathbb{E}_{p_{\tilde{D}}(\mathbf{x},\mathbf{g})}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\Omega(\mathbf{z}, \mathbf{g})\big] \tag{5}$$

where $\gamma > 0$ controls the strength of the concept-level supervision. Following [32], the term $\Omega(\mathbf{z}, \mathbf{g})$ penalizes encodings sampled from $q_\phi(\mathbf{z} \,|\, \mathbf{x})$ for differing from the annotation $\mathbf{g}$. Specifically, we implement this term using the average cross-entropy loss $\Omega(\mathbf{z}, \mathbf{g}) := -\sum_k g_k \log \sigma(z_k) + (1 - g_k) \log(1 - \sigma(z_k))$, where the annotations g_k are rescaled to lie in $[0, 1]$ and σ is the sigmoid function.

In order to tackle concept leakage, GlanceNets integrate the OSR strategy of Sun et al. [36]. This strategy identifies out-of-class inputs by considering the class prototype $\mu_y := \mathbb{E}_{p(\mathbf{z}|y)}[\mathbf{z}]$ in $\mathbb{R}^k$ defined by the prior distribution and the decoder $p_\theta(\mathbf{x}|\mathbf{z})$. Once the encoder has been learned, GlanceNets use the training data to estimate: (*i*) a distance threshold η_y, which defines a spherical subset in the latent space $\mathcal{B}_y = \{\mathbf{z} : \|\mu_y - \mathbf{z}\| < \eta_y \}$ centered around the prototype of class y, and (*ii*) a maximum threshold on the reconstruction error η_{thr}. If new data points have reconstruction error above η_{thr} or they do not belong to any subset $\mathcal{B}_y$, they are inferred as open-set instances, i.e., $\hat{T} = 0$. This procedure is illustrated in Fig. 1. In practice, we found that choosing the thresholds as to include the 95% of training examples to work well in our experiments. Further details are available in Appendices A and C.

4.1 Benefits and Limitations

GlanceNets can naturally be combined with different VAE-based architectures for learning disentangled representations [38], including β-TCVAEs [39], InfoVAEs [40], DIP-VAEs [41], and JL1-VAEs [42]. Since our experiments already show substantial benefits for GlanceNets building on β-VAEs [43], we leave a detailed study of these extensions to future work.

Like CBNMs, GlanceNets foster alignment by leveraging supervision on the interpretable generative factors [32], possibly derived from an external data set [20]. However, GlanceNets can be readily adapted to a variety of different kinds of supervision used for VAE-based models, including *partially* annotated examples [26], group information [44], pairings [45, 46] and other kinds of weak supervision [47, 48], as well as feedback from a domain expert [49]. On the other hand, CBNMs are incompatible with these approaches.

One limitation inherited from VAEs by GlanceNets is the assumption that the interpretable generative factors are disentangled from each other [23]. In practice, GlanceNets work even when this does not hold (as in our even vs. odd experiment, see Section 5.2). However, one direction of future work is to integrate ideas from hierarchical disentanglement [50].

5 Empirical Evaluation

In this section, we present results on several tasks showing that GlanceNets outperform CBNMs [20] in terms of alignment and robustness to leakage, while achieving comparable prediction accuracy. All experiments were implemented using Python 3 and Pytorch [51] and run on a server with 128 CPUs, 1TiB RAM, and 8 A100 GPUs. GlanceNets were implemented on top of the `disentanglement-pytorch` [52] library. All alignment and disentanglement metrics were computed with `disentanglement_lib` [31]. Code for the complete experimental setup is available in the Supplementary Material and will be released on GitHub. Additional details on architectures and hyperparameters can also be found in the Supplementary Material.

5.1 GlanceNets achieve better alignment than CBNMs

In a first experiment, we compared GlanceNets with CBNMs on three classification tasks for which supervision on the generative factors is available. In order to evaluate the impact of this supervision on the different competitors, we varied the amount of training examples annotated with it from 1% to 100%. For each increment, we measured prediction performance using accuracy, and alignment using the linear variant of DCI [29] discussed in Section 3.

Data sets. We carried out our evaluation on two data sets taken from the disentanglement literature and a very challenging real-world data set. *dSprites* [27] consists of 64×64 black-and-white images of sprites on a flat background, where each sprite is determined by one categorical and four generative factors, namely `shape`, `size`, `rotation`, `position_x`, and `position_y`. The images were obtained by discretizing and enumerating the generative factors, for a total of $3 \times 6 \times 40 \times 32 \times 32$ images. *MPI3D* [53] consists of 64×64 RGB rendered images of 3D shapes held by a robotic

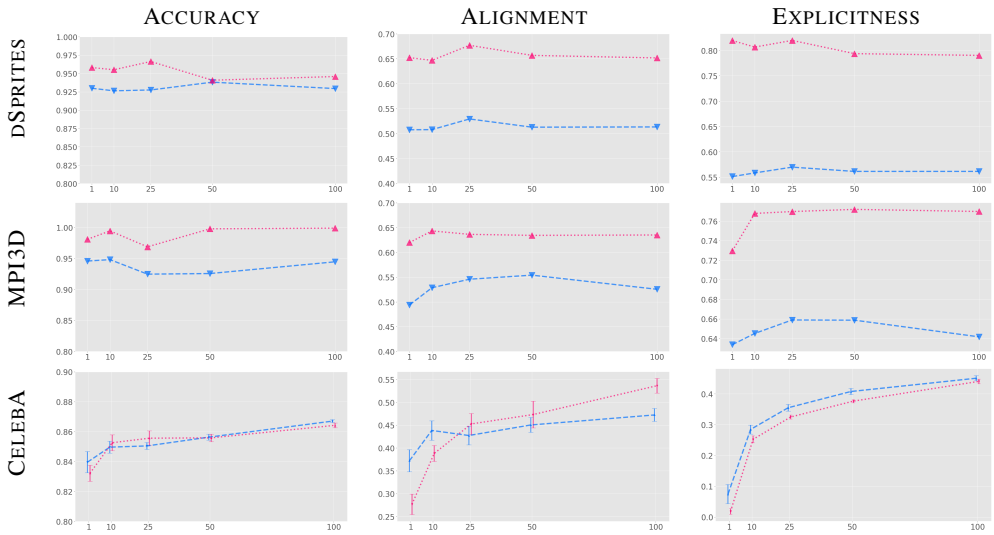


Figure 3: **GlanceNets are better aligned than CBNMs.** Each row is a data set and each column reports a different metric. The horizontal axes indicate the $\%$ of training examples for which supervision on the generative factors is provided. Remarkably, in all data sets **GlanceNets** achieve substantially better alignment than **CBNMs** for the same amount of supervision, and achieve comparable accuracy in 14 cases out of 15.

arm. The generative factors are `object_color`, `object_shape`, `object_size`, `camera_height`, `background_color`, and the `horizontal` and `vertical` position of the arm. The data contains $6 \times 6 \times 2 \times 3 \times 4 \times 40 \times 40$ examples. *CelebA* [28] is a collection of 178×218 RGB images of over 10k celebrities, converted to 64×64 by first cropping them to 178×178 and then rescaling. Images are annotated with 40 binary generative factors including hair color, presence of sunglasses, *etc*. Since we are interested in measuring alignment, we considered only those 10 factors that CBNMs can fit well (in the Appendix). We also dropped all those examples for which hair color is not unique (e.g., annotated as both `blonde` and `black`), obtaining approx. $127k$ examples. CelebA is more challenging than dSprites and MPI3D, as it does not include all possible factor variations and the generative factors – although disentangled – are insufficient to completely determine the contents of the images. For dSprites and MPI3D, we used a random 80/10/10 train/validation/test split, while for CelebA we kept the original split [28].

We generated the ground-truth labels y as follows. For dSprites, we labeled images according to a random but fixed linear separator defined over the *continuous* generative factors, chosen so as to ensure that the classes are balanced. For MPI3D and CelebA, we focused on the *categorical* factors instead. Specifically, we clustered all images using the algorithm of [54], for a total of 10 and 4 clusters for MPI3D and CelebA respectively, and then labeled all examples based on their reference cluster. This led to slightly unbalanced classes containing different percentages of examples, ranging from 5% to 16% in MPI3D and from 21% to 29% in CelebA.

Architectures. For dSprites and MPI3D, we implemented the encoder as a six layer convolutional neural net, while for CelebA we adapted the convolutional architecture of Ghosh et al. [55]. We employed a six layer convolutional architecture for the decoder in all cases, for simplicity, as changing it did not lead to substantial differences in performance. In all cases, as for all CBMs (see Section 2), the classifier was implemented as a dense layer followed by a softmax activation. The very same architectures were used for both GlanceNets and CBNMs, for fairness. For each data set, we chose the latent space dimension as the total number of generative factors, where categorical ones are one hot encoded. In particular, we used 7 latent factors for dSprites, 21 for MPI3D and 10 for CelebA. Further details are included in the Supplementary Material.

Results and discussion. The results of this first experiment are reported in Fig. 3. The behavior of both competitors on dSprites and MPI3D was extremely stable, owing to the fact that these data sets cover an essentially exhaustive set of variations for all generative factors, so we report their hold-out performance on the test set. Since for CelebA variance was non-negligible, we ran both methods

7 times varying the random seed used to initialize the network and report the average performance across runs and its standard deviation.

In addition to alignment, we also report explicitness [29], which measures how well the linear regressor employed by DCI fits the generative factors. The higher, the better. Details on its evaluation are included in Suppl. Material.

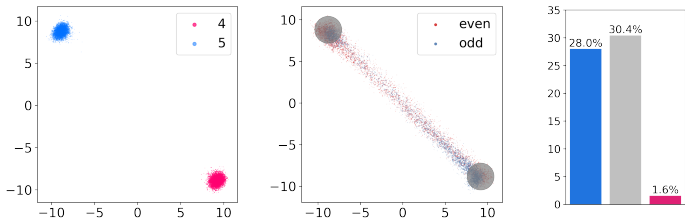
The plots clearly show that, although the two methods achieve high and comparable accuracy in all settings, GlanceNets attain better alignment in all data sets and for all supervision regimes than CBNMs, with a single exception in CelebA using low values of supervision, for a total of 13 wins out of 15 cases. In all *disentanglement* data sets, there is a clear margin between the alignment achieved by GlanceNets and that of CBNMs: performances vary up to maximum of 15% in dSprites, and a minumum of 8% in MPI3D. In CelebA, the gap is evident with full supervision (almost 8% of difference in alignment), and GlanceNets still attain overall better scores in the 25% and 50% regime. On the other hand, performance are lower, but comparable, with 10% supervision. The case at 1% refers to an extreme situation where both CBNMs and GlanceNets struggle to align with generative factors, as is clear also from the very low explicitness.

In dSprites and MPI3D, both GlanceNets and CBNMs quickly achieve very high alignment at 1% supervision, as expected [32], whereas better results in CelebA are obtained with growing supervision. Also, both models display similar stability on this data set, as shown by the error bars in the plot.

5.2 GlanceNets are leak-proof

Next, we evaluated robustness to concept leakage in two scenarios that differ in whether the unobserved generative factors are disentangled with the observed ones or not, see Section 3. In both experiments, we compare GlanceNets with a CBNM and a modified GlanceNet where the OSR component has been removed (denoted CG-VAE).

Leakage due to unobserved entangled factors. We start by replicating the experiment of Mahinpei et al. [7]: the goal is to discriminate between even and odd MNIST images using a latent representation $\mathbf{Z} = (Z_4, Z_5)$ obtained by training (with complete supervision on the generative factors) *only* on examples of 4's and 5's. Leakage occurs if the learned representation can be used to solve the prediction task better than random on a test set where all digits except 4 and 5 occur.[3] During training, we use the digit label for conditioning the prior $p(\mathbf{Z} \mid \mathbf{Y})$ of the GlanceNet. More qualitative results are collected in Appendix D.



(a) (b) (c)

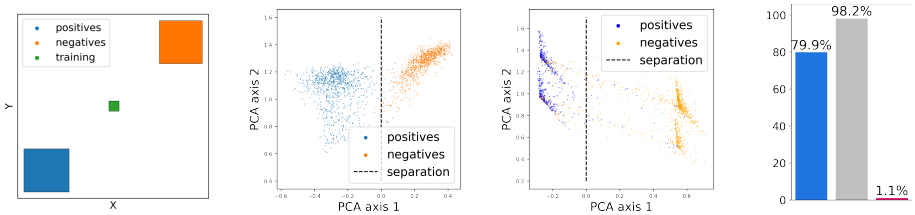
Figure 4: **GlanceNets are leak-proof on MNIST.** (a) Training set embedded by GlanceNet with $\beta = 100$; axes indicate z_4 and z_5 and color the concept label, i.e., 4 vs. 5. (b) Latent representations of the test images, divided in even vs. odd. Every ball in light gray denotes the region $|\mu_y - \mathbf{z}| < \eta_y$ for each class prototype y. For more details, refer to Section 3. (c) Information Leakage performances of the considered models: CBNM, CG-VAE and GlanceNet.

Fig. 4 (a, b) illustrates the latent representations of the training and test set output by a GlanceNet: since the two digits are mutually exclusive, the model has learned to map all instances along the (z_4, z_5) diagonal. This is where OSR kicks in: if an input is identified as open-set, T is predicted as 0 by the OSR component and the input is rejected. In all leakage experiments, we implement rejection by predicting a random label. Since MNIST is balanced, we measure leakage by computing

[3] Margeloiu et al. [8] perform classification using a multi-layer perceptron on top of $\mathbf{z}$. Following the CBM literature, we use a linear classifier instead. Leakage occurs regardless.

the difference in accuracy between the classifier and an ideal random predictor, i.e., $2 \cdot |\text{acc} - \frac{1}{2}|$: the smaller, the better. The results, shown in Fig. 4 (c), show a substantial difference between GlanceNet and the other approaches. Consistently with the values reported in [7], CBNMs are affected by a considerable amount of leakage, around 28%. This is not the case for our GlanceNet: most (approx. 85%) test images are correctly identified as open-set and rejected, leading to a very low (about 2%) leakage, 26% less than CBNMs. The results for CG-VAE also indicate that removing the open-set component from GlanceNets dramatically increases leakage back to around 30%. This shows that alignment and disentanglement alone are not sufficient, and that the open-set component plays a critical role for preventing leakage.

Leakage due to unobserved disentangled factors. Next, we analyze concept leakage between *disentangled* generative factors using the dSprites data set. To this end, we defined a binary classification task in which the ground-truth label depends on `position_x` and `position_y` only. In particular, instances within a fixed distance from $(0, 0)$ are annotated as positive and the rest as negative, as shown in Fig. 5 (a). In order to trigger leakage, all competitors are trained (using full concept-level supervision, as before) on training images where `shape`, `size` and `rotation` vary, but `position_x` and `position_y` are almost constant (they range in a small interval around $(0.5, 0.5)$, cf. Fig. 5). leakage occurs if the learned model can successfully classify test instances where `position_x` and `position_y` are no longer fixed. More qualitative results can be found in Appendix E.



(a) (b) (c) (d)

Figure 5: **GlanceNets are leak-proof on dSprites.** (*a*) The variations over `pos_x` and `pos_y` for the training set, and for the test set, divided in positives vs. negatives. (*b*) PCA reduction for GlanceNet over the its five latent factors. (*c*) PCA reduction for CBNM; the dotted line indicates the separating hyperplane predicted in the second phase. (*d*) Leakage % for CBNM, CG-VAE and GlanceNet.

For both competitors, we encode `shape` using a 3D one-hot encoding and `size` and `rotation` as continuous variables. During training, we use the `shape` annotation for conditioning the prior $p(\mathbf{Z} \mid \mathbf{Y})$ of the GlanceNet. The first two PCA components of the latent representations acquired by our GlanceNet and by a CBNM are shown, rotated so as to be separable on the first axis, in Fig. 5 (b, c): in both cases, it is possible to separate positives from negatives based on the obtained representations in the five latent dimensions. As shown in Fig. 5 (d), this means that both CBNM and CG-VAE suffer from very large leakage, 80% and 98%, respectively. In contrast, OSR allows us to correctly identify and reject almost all test instances, leading to negligible leakage even in this disentangled setting.

6 Related Work

Concept-based explainability. Concepts lie at the heart of AI [56] and have recently resurfaced as a natural medium for communicating with human stakeholders [11]. In explainable AI, this was first exploited by approaches like TCAV [57], which extract local concept-based explanations from black-box models using concept-level supervision to define the target concepts. Post-hoc explanations, however, are notoriously unfaithful to the model's reasoning [58–60]. CBMs, including GlanceNets, avoid this issue by leveraging concept-like representations directly for computing their predictions. Existing CBMs model concepts using prototypes [2, 3, 16, 17] or other representations [1, 20, 4, 5], but they seek interpretability using heuristics, and the quality of concepts they acquire has been called into question [61, 19, 7, 8]. We show that disentangled representation learning helps in this regard.

Disentanglement and interpretability. Interpretability is one of the main driving factors behind the development of disentangled representation learning [62–64]. These approaches however make

no distinction between interpretable and non-interpretable generative factors and generally focus on properties *of the world*, like independence between causal mechanisms [9] or invariances [43]. Interpretability, however, depends on human factors that are not well understood and therefore usually ignored [12, 65]. The link between disentanglement and interpretability has never been made explicit. Importantly, in contrast to alignment, disentanglement does not require that the map between matching generative and learned factors preserves semantics. We remark that other VAE-based classifiers either do not tackle disentanglement or are unconcerned with concept leakage [66, 67, 36].

Disentanglement and CBMs. Neither the literature on disentanglement nor the one on CBMs have attempted to formalize the notion of interpretability or to establish a proper link between the latter and disentanglement. The work of Kazhdan et al. [68] is the only one to compare techniques for disentangled representation learning and concept acquisition, however it makes no attempt at linking the two notions. Our work fills this gap.

Acknowledgments and Disclosure of Funding

The research of ST and AP was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

References

[1] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7786–7795, 2018.

[2] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[3] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems*, 32:8930–8941, 2019.

[4] Max Losch, Mario Fritz, and Bernt Schiele. Interpretability beyond classification output: Semantic Bottleneck Networks. *arXiv preprint arXiv:1907.10882*, 2019.

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

[6] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[7] Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models. In *International Conference on Machine Learning: Workshop on Theoretic Foundation, Criticism, and Application Trend of Explainable AI*, volume 1, pages 1–13, 2021.

[8] Andrei Margeloiu, Matthew Ashman, Umang Bhatt, Yanzhi Chen, Mateja Jamnik, and Adrian Weller. Do concept bottleneck models learn as intended? *arXiv preprint arXiv:2105.04289*, 2021.

[9] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[10] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35 (7):1757–1772, 2012.

[11] Subbarao Kambhampati, Sarath Sreedharan, Mudit Verma, Yantian Zha, and Lin Guan. Symbols as a Lingua Franca for Bridging Human-AI Chasm for Explainable and Advisable AI Systems. In *Proceedings of Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[12] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[13] Peter Hase and Mohit Bansal. Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5540–5552, 2020.

[14] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[15] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.

[16] Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. ProtoPShare: Prototypical Parts Sharing for Similarity Discovery in Interpretable Image Classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, page 14201430, 2021.

[17] Meike Nauta, Ron van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14933–14943, 2021.

[18] Gurmail Singh and Kin-Choong Yow. These do not look like those: An interpretable deep learning model for image recognition. *IEEE Access*, 9:41482–41493, 2021.

[19] Adrian Hoffmann, Claudio Fanconi, Rahul Rade, and Jonas Kohler. This looks like that... does it? shortcomings of latent space prototype interpretability in deep networks. *arXiv preprint arXiv:2105.02968*, 2021.

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

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[22] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. *arXiv preprint arXiv:1206.6471*, 2012.

[23] Raphael Suter, Djordje Miladinovic, Bernhard Schölkopf, and Stefan Bauer. Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In *International Conference on Machine Learning*, pages 6056–6065. PMLR, 2019.

[24] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. 2017.

[25] Abbavaram Gowtham Reddy, L Benin Godfrey, and Vineeth N Balasubramanian. On causally disentangled representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[26] Aviv Gabbay, Niv Cohen, and Yedid Hoshen. An image is worth more than a thousand words: Towards disentanglement in the wild. *Advances in Neural Information Processing Systems*, 34, 2021.

[27] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[28] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[29] Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

[30] Julian Zaidi, Jonathan Boilard, Ghyslain Gagnon, and Marc-André Carbonneau. Measuring disentanglement: A review of metrics. *arXiv preprint arXiv:2012.09276*, 2020.

[31] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pages 4114–4124, 2019.

[32] Francesco Locatello, Michael Tschannen, Stefan Bauer, Gunnar Rätsch, Bernhard Schölkopf, and Olivier Bachem. Disentangling factors of variations using few labels. In *International Conference on Learning Representations*, 2020.

[33] Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pages 2207–2217. PMLR, 2020.

[34] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International conference on machine learning*. PMLR, 2014.

[35] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*. PMLR, 2014.

[36] Xin Sun, Zhenning Yang, Chi Zhang, Keck-Voon Ling, and Guohao Peng. Conditional gaussian distribution learning for open set recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13480–13489, 2020.

[37] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[38] Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, Narayanaswamy Siddharth, Brooks Paige, Dana H Brooks, Jennifer Dy, and Jan-Willem Meent. Structured disentangled representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2525–2534. PMLR, 2019.

[39] Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in vaes. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2615–2625, 2018.

[40] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Balancing learning and inference in variational autoencoders. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 5885–5892, 2019.

[41] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*, 2018.

[42] Travers Rhodes and Daniel Lee. Local disentanglement in variational auto-encoders using jacobian l_1 regularization. *Advances in Neural Information Processing Systems*, 34:22708–22719, 2021.

[43] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

[44] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[45] Rui Shu, Yining Chen, Abhishek Kumar, Stefano Ermon, and Ben Poole. Weakly supervised disentanglement with guarantees. In *International Conference on Learning Representations*, 2020.

[46] Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning*, pages 6348–6359. PMLR, 2020.

[47] Aviv Gabbay and Yedid Hoshen. Latent optimization for non-adversarial representation disentanglement. *arXiv preprint arXiv:1906.11796*, 2019.

[48] Junxiang Chen and Kayhan Batmanghelich. Weakly supervised disentanglement by pairwise similarities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3495–3502, 2020.

[49] Wolfgang Stammer, Marius Memmel, Patrick Schramowski, and Kristian Kersting. Interactive disentanglement: Learning concepts by interacting with their prototype representations. *arXiv preprint arXiv:2112.02290*, 2021.

[50] Andrew Ross and Finale Doshi-Velez. Benchmarks, algorithms, and metrics for hierarchical disentanglement. In *International Conference on Machine Learning*, pages 9084–9094. PMLR, 2021.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[52] Amir H. Abdi, Purang Abolmaesumi, and Sidney Fels. Variational learning with disentanglement-pytorch. *arXiv preprint arXiv:1912.05184*, 2019.

[53] Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[54] Zhexue Huang. Clustering large data sets with mixed numeric and categorical values. In *In The First Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 21–34, 1997.

[55] Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019.

[56] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

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

[58] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *Advances in Neural Information Processing Systems*, 32:13589–13600, 2019.

[59] Stefano Teso. Toward faithful explanatory active learning with self-explainable neural nets. In *Proceedings of the Workshop on Interactive Adaptive Learning (IAL 2019)*, pages 4–16, 2019.

[60] Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified bp attributions fail. In *International Conference on Machine Learning*, pages 9046–9057. PMLR, 2020.

[61] Meike Nauta, Annemarie Jutte, Jesper Provoost, and Christin Seifert. This looks like that, because... explaining prototypes for interpretable image recognition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 441–456. Springer, 2021.

[62] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

[63] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. *Advances in neural information processing systems*, 28, 2015.

[64] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

[65] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[66] Sjoerd van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. Are disentangled representations helpful for abstract visual reasoning? In *NeurIPS*, 2019.

[67] Weidi Xu and Haoze Sun. Semi-supervised variational autoencoders for sequence classification. *ArXiv*, abs/1603.02514, 2016.

[68] Dmitry Kazhdan, Botty Dimanov, Helena Andres Terre, Mateja Jamnik, Pietro Liò, and Adrian Weller. Is disentanglement all you need? comparing concept-based & disentanglement approaches. *arXiv preprint arXiv:2104.06917*, 2021.

[69] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein autoencoders. *arXiv preprint arXiv:1711.01558*, 2017.

[70] Adam Kingma. Adam: a method for stochastic optimization, 3rd, int. conf. learn. represent. iclr 2015-conf. *Track Proc*, (1-15).

A Implementation details

A.1 GlanceNet and CBNM Architectures

In all experiments, we used exactly the same architecture and number of latent variables for both GlanceNets and CBNMs to ensure a fair comparison.

Encoder architectures:

- *dSprites*: We chose a rather standard architecture [52]. It comprises six 2D convolutional layers of depth 32, 32, 64, 128, 256, and 256, respectively, all with a kernel of size 4, stride 2, and padding 1, and followed by ReLU activations. The output is flattened to a vector and passed through a dense layer to obtain the mean $\boldsymbol{\mu}(\mathbf{x})$ and (diagonal) variance $\boldsymbol{\sigma}(\mathbf{x})$ of the encoder distribution $\mathcal{N}(\mathbf{Z} \mid \boldsymbol{\mu}(\mathbf{x}), \mathrm{diag}(\boldsymbol{\sigma}(\mathbf{x})))$.
- *MPI3D*: We used the same architecture with slightly different convolutional depths of 32, 32, 64, 64, 128, and 256, changing also the kernel size to 3 and removing padding, as per [52].
- *CelebA*: We leveraged the architecture of Ghosh et al. [55], which is a common reference for VAE models on CelebA-64 [69]. The encoder is composed of four convolutions of depth 128, 256, 512, 1024 respectively, all with kernel size of 5, stride of 2, followed batch normalization and ReLU activation.

The models had exactly as many latent variables as generative factors for which supervision is available, which in our three data sets are 7, 21, and 10, respectively.

Decoder architecture: All models share the same decoder architecture, obtained by stacking:

- A 2D convolution on the latent space with a filter depth of 256, kernel size of 1, and stride of 2, followed by the ReLU activation;
- Five transposed 2D convolutions of depth 256, 256, 128, 128, 64, 64, and `num_channels`, respectively, all with kernel of size 4 and stride 2.

Here, `num_channels` is either 1 (dSprites) or 3 (MPI3D and CelebA). The shape of the last layer was chosen so as to match the dimension of the input image. Additional details can be found in the various Tables in this appendix.

A.2 Supervision and Training

Concept-level supervision. Depending on the supervision provided, only a fraction of the inputs was made available during training with their generative factors. In dSprites and MPI3D all generative factors are matched by the models, whereas in the case of CelebA we restricted learning to those 10 attributes that are best fit by the CBNMs, namely: `bald`, `black hair`, `brown hair`, `blonde hair`, `eyeglasses`, `gray hair`, `male`, `no beard`, `smiling`, and `wearing hat`. Both CBNMs and GlanceNets are jointly trained, meaning that optimization steps for the concepts and label supervision are taken simultaneously. Whenever concept supervision is lower than 100%, for those examples without concept annotations we trained both models using label supervision only. We did not evaluate other training strategies available for CBNMs (e.g., sequential training [20]) as these appear to bring no benefit in terms of either performance nor leakage.

Optimization setup. In all experiments, we used the Adam optimizer [70] with default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For dSprites, we used a batch size of 64 and fixed learning rate to $\eta = 4 \cdot 10^{-4}$, while for MPI3D and CelebA we used a batch size of 100 and annealed the learning rate from 10^{-7} to $\eta_{MPI} = 10^{-3}$ and $\eta_{CelebA} = 10^{-4}$, respectively. To prevent overfitting, in CelebA we multiplied the learning rate by a factor of 0.95 in each epoch and apply early stopping on the validation set, with a patience of 10 epochs.

Prior to training, we selected a reasonable value for the following hyper-parameters:

- β: the weight of the KL divergence in Eq. (4).
- γ: the weight of the loss on the generative factors in Eq. (5).
- λ: the weight of the cross-entropy loss over the label, which is left implicit in Eq. (4).

Table 1: Structure of the encoder network used for dSprites.

INPUT SHAPE	LAYER TYPE	PARAMETERS	ACTIVATION
$(64, 64, 1)$	Convolution	depth=32, kernel=4, stride=2, padding=1	ReLU
$(32, 32, 32)$	Convolution	depth=32, kernel=4, stride=2, padding=1	ReLU
$(16, 16, 32)$	Convolution	depth=64, kernel=4, stride=2, padding=1	ReLU
$(8, 8, 64)$	Convolution	depth=128, kernel=4, stride=2, padding=1	ReLU
$(4, 4, 128)$	Convolution	depth=256, kernel= 4, stride=2, padding=1	ReLU
$(2, 2, 256)$	Convolution	depth=256, kernel=4, stride=2, padding=1	ReLU
$(1, 1, 256)$	Flatten		
$(1, 256)$	Linear	dim=7+7, bias = True	

Table 2: Structure of the encoder network used for MPI3D.

INPUT SHAPE	LAYER TYPE	PARAMETERS	ACTIVATION
$(64, 64, 3)$	Convolution	depth=32, kernel=3, stride=2	ReLU
$(32, 32, 32)$	Convolution	depth=32, kernel=3, stride=2	ReLU
$(16, 16, 32)$	Convolution	depth=64, kernel=3, stride=2	ReLU
$(8, 8, 64)$	Convolution	depth=64, kernel=3, stride=2	ReLU
$(4, 4, 64)$	Convolution	depth=128, kernel= 3, stride=2	ReLU
$(2, 2, 128)$	Convolution	depth=256, kernel=3, stride=2	ReLU
$(1, 1, 256)$	Flatten		
$(1, 256)$	Linear	dim=21+21, bias = True	

Table 3: Structure of the encoder network used for CelebA.

INPUT SHAPE	LAYER TYPE	PARAMETERS	FILTER	ACTIVATION
$(64, 64, 3)$	Convolution	depth=128, kernel=5, stride=2	BatchNorm	ReLU
$(30, 30, 128)$	Convolution	depth=256, kernel=5, stride=2	BatchNorm	ReLU
$(13, 13, 256)$	Convolution	depth=512, kernel=5, stride=2	BatchNorm	ReLU
$(5, 5, 512)$	Convolution	depth=1028, kernel=5, stride=2	BatchNorm	ReLU
$(1, 1, 1028)$	Flatten			
$(1, 1028)$	Linear	dim=10+10, bias = True		

Table 4: Structure of the decoder network.

INPUT SHAPE	LAYER TYPE	PARAMETERS	ACTIVATION
$(\dim(\mathbf{z}))$	Unsqueeze		
$(\dim(\mathbf{z}), 1, 1)$	Convolution	depth=256, kernel=1, stride=2	ReLU
$(256, 1, 1)$	Deconvolution	depth=256, kernel=4, stride=2	ReLU
$(256, 2, 2)$	Deconvolution	depth=128, kernel=4, stride=2	ReLU
$(128, 6, 6)$	Deconvolution	depth=128, kernel=4, stride=2	ReLU
$(128, 14, 14)$	Deconvolution	depth=64, kernel=4, stride=2	ReLU
$(64, 30, 30)$	Deconvolution	depth=64, kernel=4, stride=2	ReLU
$(64, 62, 62)$	Deconvolution	depth=`num_channels`, kernel=4, stride=2	

For dSprites, we found a good balance for $\lambda = \gamma = 100$, while for MPI3D we achieved good performance with $\lambda = 10^3$ and $\gamma = 7 \cdot 10^3$. We adopted the same hyperparameters choice for CelebA, with the exception that we reduced the reconstruction error by 0.01. For all data sets, we cross-validated over different values of β but we obtained better alignment performances with $\beta \approx 1$. This happens because we inject supervision on the latent factors (which is absent in regular β-VAEs [43]).

A.3 Implementation of leakage tests

MNIST. For this dataset, we considered only Multi-Latyer Perceptrons instead of convolutions. Both the encoder and the decoder are composed by two linear layers with depth 128, and a dense layer connected to the latent space and to the input space, respectively. Further details are on Table 5.

For the GlanceNet we considered a latent space of dimension 10 where the supervision on the 4 and 5 digits is used to fit the $\{z_4, z_5\}$ latent factors. These two, constitute the latent subspace where leakage occurs, while the other are useful only for reconstruction. Conversely, for the CBNM we considered only two latent factors.

During training of the latent encodings, we used stochastic gradient descent with learning rate $\eta = 0.001$, reducing it by 0.95 in each epoch for both CBNMs and GlanceNets. The training was performed only on the 4 and 5 digits (in the usual training set partition for MNIST), for almost 50 epochs. Afterwards, we considered the open-set representations, restricted to $\{z_4, z_5\}$, as inputs for training a logistic regression for parity recognition. During the training, only the digits in the MNIST training set partition (exception made for 4 and 5) are considered, while performance are calculated on the test set.

dSprites. We adopted the same architecture in the upper section, except that we reduced the latent space to 5 dimensions. As a reminder, during training all sprites are almost fixed at the center, therefore additional factors of variations for its position are needless. The training was performed over 300 epochs for both GlanceNets and CBNMs, with $\eta = 4 \cdot 10^{-4}$. After training, the representations of the open-set sprites (in which position is no longer fixed) are used to fit a logistic regression. In this case, the labels depend on whether the sprite is located at bottom-left corner or at the upper-right one, for more information refer to Fig. 5. The classification performance was evaluated on a held-out test set for both models, under an 80/20 train/test split.

Table 5: Encoder and Decoder structures for MNIST

TYPE	INPUT SHAPE	LAYER TYPE	PARAMETERS	ACTIVATION
ENCODER				
	$(28, 28)$	Flatten		
	(784)	Linear	dim=128, bias=True	ReLU
	(128)	Linear	dim=128, bias=True	ReLU
	(128)	Linear	dim=10+10, bias=True	
DECODER				
	$(\dim(\mathbf{z}))$	Linear	dim=128, bias=True	ReLU
	(128)	Linear	dim=128, bias=True	ReLU
	(128)	Linear	dim=728, bias=True	
	(728)	Unsqueeze		

B DCI framework

In our case study, we are interested into DCI in [29]) maps that linearly connect the $\mathbf{z}'s$ to the $\mathbf{g}'s$. In order to evaluate alignment performances, the inverse map $\alpha^{-1} : \mathbb{R}^k \to \mathbb{R}^{n_I}$ is constructed from the latent space to the span of the n_I generative factors. The latent representations and generative factors were normalized in the $[0, 1]$ interval prior to learning.

B.1 Alignment and explicitness

The importance weights of this map are the absolute-values of the weights in the linear matrix of α^{-1}, indicated as $B \in \mathbb{R}^{k \times n_I}$ in the main text, see Section 3. Then, the importance weights are used

to evaluate the dispersion of the learned weights. To this end, we measure each Shannon entropy H_j on all k latent factors:

$$H_j = -\sum_{i \in 1}^{n_I} \bar{b}_{ji} \log_n \bar{b}_{ji} \quad \text{where} \;\; \bar{b}_{ji} = b_{ji} \Big/ \sum_{\ell=1}^{n_I} b_{j\ell} \tag{6}$$

Then, the average alignment is calculated as:

$$\text{alignment} = 1 - \sum_{j=1}^{k} \rho_j H_j \quad \text{where} \;\; \rho_j = \sum_{i=1}^{n_I} b_{ji} \Big/ \sum_{j'=1, i=1}^{k, n_I} b_{j'i} \tag{7}$$

and ranges in $[0, 1]$. Similarly, the quantity:

$$\tilde{b}_{ji} = b_{ji} \Big/ \sum_{\ell=1}^{k} b_{\ell i} \quad \text{and} \quad \tilde{H}_i = \sum_{j=1}^{k} \tilde{b}_{ji} \log_k \tilde{b}_{ji} \tag{8}$$

is the *completeness* of the latent representation, a measure akin to alignment (Eq. (7)) that quantities the degree to which each generative factor correlates with *distinct* latent factors. Alignment and completeness relate to different properties of the map: the higher the *alignment*, the more each Z_j depends on variations of only a single G_i. On the other hand, learning multiple Z_j's capturing a single G_i reduces the *completeness*. As an illustrative example, consider the matrix:

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \end{pmatrix}$$

From the above definitions, one gets $alignment = 1$ and $completeness < 1$. This follows since each Shannon entropy for the alignment score is zero (as it is related to the rows), whereas the Shannon entropy for the completeness is greater than zero (it refers to the columns). Moreover, each latent variable z_i depends only on the variations of a single generative factor g_j.

We also calculate the explicitness of the map α, which is related to the mean squared error (MSE) of the prediction. Since the MSE for random guessing for a variable in the $[0, 1]$ interval is equal to 1/6, the explicitness becomes:

$$\text{explicitness} = 1 - 6 \cdot \text{MSE}$$

B.2 Empirical evaluation

For dSprites and MPI3D, all DCI quantities were calculated with the built-in evaluation code provided by `disentanglement_lib`, [31]. For CelebA, since the 40 attributes in CelebA are not exhaustive for the image generation, we implemented computed DCI as follows: *(i)* we first converted the J attributes $\mathbf{z}_J$ and $\mathbf{g}_J$ connected to `hair type` to a single concept h and fit the model with Lasso regression to predict g_h from $\mathbf{z}$. Then, *(ii)* we trained a Logistic Regression with $l1$ penalty to predict the remaining $\mathbf{g}'s$. Finally, we took both weights in *(i)* and in *(ii)* to compute the matrix $B \in \mathbb{R}^{6 \times 6}$. In this way, we determined alignment and explicitness for CelebA. We chose the lasso coefficient $\lambda = 0.01$ for both regressions.

C Open-Set Recognition Mechanism

In this section, we provide additional details on the OSR mechanism introduced in Section 3. Our method adapts the one of Sun et al. [36], which distinguishes between closed-set and open-set data points by combining a reconstruction check Γ_r with a localization check Γ_{ls}. The overall OSR check is given by:

$$\hat{T} = \Gamma_r \wedge \Gamma_{ls} \tag{9}$$

After completing the training process, all the training instances are passed to the model to evaluate the thresholds:

- The reconstruction threshold η_r is the maximum real number such that a fixed percentage of training examples have reconstruction error less or equal to it. At test time, given an instance $\mathbf{x}$, let $\hat{\eta} = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$ be the reconstruction error. Then, $\Gamma_r = 1$ (i.e., the check passes) if the empirical reconstruction error is less than the threshold, $\hat{\eta} < \eta_r$, otherwise $\Gamma_r = 0$.
- The latent-space distance thresholds are evaluated for each class-prototype embedded in the latent space $\mu_y = \mathbb{E}_{p(\mathbf{z}|y)}[\mathbf{z}]$. For each of them, we first evaluated the relative distance between point belonging to the class y and the prototype μ_y. Then, we evaluated a threshold η_y on the distances, as to include a fixed percentage of training instances into the set $\mathcal{B}_y = \{\mathbf{z} : \|\mu_y - \mathbf{z}\| < \eta_y\}$. At test time, those points that do not belong to any set $\mathcal{B}_y$ are predicted as open-set instances, i.e. $\Gamma_{ls} = 0$, otherwise $\Gamma_{ls} = 1$.

In our experiments, the threshold are obtained by fixing both reconstruction and latent space distance to keep the 95% of training data. In the case of η_y, this quota has been reached singularly for each $\mathcal{B}_y$, thus obtaining different values η_y's from one another. Finally, combining both rejection methods we are sure the model would predict as closed-set at least the 90% of training instances.

D Concept Leakage in MNIST

We report here additional details for the concept leakage test on MNIST, which has been originally introduced by Margeloiu et al. [8]. The experiment has two stages:

1. At train time, the model is trained to align its representations to the concepts of 4 and 5, by passing full supervision on them. Both CBNMs and GlanceNets are allotted two latent concepts, which we denote (Z_4, Z_5). There is no downstream classification task in this stage.
2. At test time, all MNIST images, excluding those of 4's and 5's, are encoded using the learned encoder and used to learn a classifier of even vs. odd digits. The performance of the resulting classifier, applied to non-$\{4, 5\}$ images, is then computed.

In this experiment, concept leakage occurs if the accuracy on the downstream task is above the 50%.

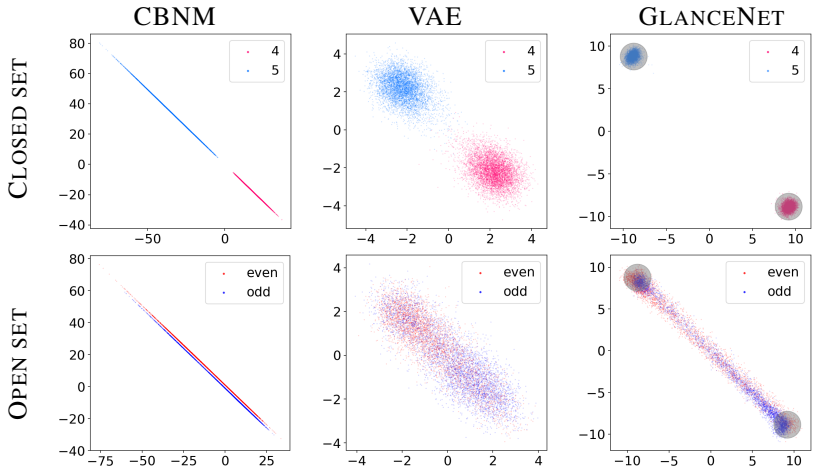


Figure 6: **Latent space representation for MNIST.** On the first row, we report the representations for 4 and 5 as fitted by CBNM, VAE and GlanceNet, respectively. On the second row, we display the scattering plot for points only belonging to the open set. For CBNM, we separated even and odd instances by $\Delta y = 2$, since their representations strongly overlap. All plots comprise only the z_4, z_5 axes.

D.1 Qualitative results

In Fig. 6, we show the latent space representations for different models on the MNIST leakage test, for both closed-set and open-set data points. To illustrate the contribution of our mixture prior, in

addition to the CBNM and GlanceNet models, we also considered a simpler supervised VAE model. This model has the same encoder, decoder, and classifier as the GlanceNet, but uses a regular Gaussian prior[4]. We found that this model achieved a similar level of leakage to CG-VAE. We display in Fig. 7 the reconstruction of a few random examples output by GlanceNet: the reconstructions of all instances belonging to the open-set greatly deviate from the original.

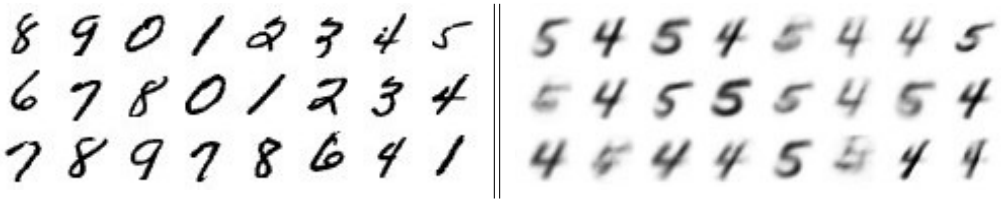


Figure 7: **MNIST reconstruction with GlanceNet.** On the left we reported the original digits, whereas on the right the reconstruction with the learned decoder. All images have been inverted in the black and white scale.

E Concept Leakage on dSprites

In this section, we report additional material for the dSprites concept leakage experiment. This experiment resembles the previous one on MNIST:

1. At training time, a CBM learns the representations of `shape`, `size` and `rotation` by receiving supervision on all possible variations of these factors. On the other hand, no variation of factors `pos_x` and `pos_y` are observed, in the sense that the position of the training sprites is fixed to the center of the image. We fit the CBNM and the GlanceNet with 5 latent factors to learn the representations. Again, no downstream classification task appears at this stage.
2. At test time, the encoder is kept fixed for different variations varying of the factors `pos_x` and `pos_y` are observed. The downstream task in this phase amounts to recognizing whether a sprite lies in top-right or in the bottom-left corner of the image.

In this experiment, concept leakage occurs if the accuracy on the downstream task is above the 50%.

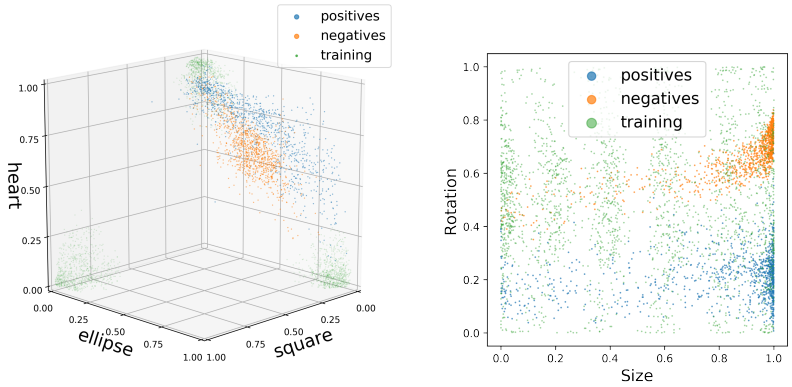


Figure 8: **Concept space representation of GlanceNet for dSprites.** On the left, we show the projections on the one-hot encoded `shape` subspace, whereas on the right we project on the {`size`, `rotation`} subspace. We include the representations for training points, positive and negative ones.

E.1 Qualitative results

We also include qualitative results for GlanceNet and on dSprites for closed set and open set data points. In Fig. 8 we display the projections of train and test points on the two different latent

[4]For the VAE model, we chose the Gaussian prior in [34], i.e., $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, 1)$.

subspaces (see caption). In both of them, positives and negatives representations are well separated from each other, implying substantia leakage. We also evaluated the reconstruction quality during training and testing and reported some of them in Fig. 9. Notably, almost all points are recognized to be open set instances thanks to the reconstruction threshold.

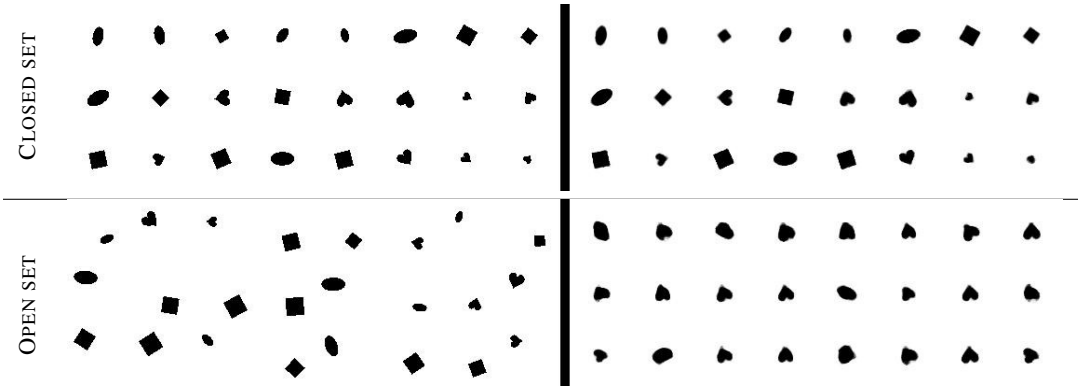


Figure 9: **Reconstruction for dSprites on train and test with GlanceNet.** On the upper panel, we report the reconstructions of the sprites belonging to the closed set. On the lower one, the reconstructions of the open set points. Like MNIST, all images have been inverted in the black and white scale.

F Additional results for GlanceNets and CBNMs in CelebA

In this section, we discuss additional results for CBNMs vs. GlanceNets on the CelebA dataset. We first report the accuracy of the learned concepts on the supervised latent factors for both CBNMs and GlanceNets in CelebA. Then, we examine two variants of GlanceNets varying the dimension of the unsupervised factors in the latent space: a β-VAE with 20 latent factors and a β-TCVAE with 40 latent factors, [39]. This variant includes an additional loss term to Eq. (5) given by the Total Correlation (TC) of the model posterior $q_\phi(\mathbf{z}) = \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{x})]$:

$$(\beta - 1) \cdot \mathsf{KL}\big(q_\phi(\mathbf{z}) || \prod_{i=1}^{k} q_\phi(z_i)\big) \tag{10}$$

where β denotes the strength hyper-parameter. Both the β-VAE and the β-TCVAE receive supervision only on the 10 generative factors that are fitted in the CBNM. A the end of the section, we report traversals for the models with 40 latent factors.

F.1 Concepts Accuracy

We report the concepts accuracy for both CBNMs and GlanceNets in Fig. 10, with 10 latent dimensions and the TCVAE variant. The difference in concept accuracy between GlanceNet (both variants) and CBNMs is negligible, with GlanceNets showing slightly higher variance when the percentage of concepts supervision is very small. This highlights how, in terms of concept accuracy, the two classes of models are essentially indistinguishable, even though they are in terms of alignment.

F.2 Performances upon variations of the latent space dimension

Similarly to the analysis in Section 5.1, we show the behavior of the metrics upon increasing the dimension of the latent space. The first variant of GlanceNets, based on β-VAE, was fitted with $\beta \approx 1$, with a latent space of dimension 20. The second variant is a TCVAE, trained with a weight of the total correlation $\beta = 10$ for all concepts supervisions, exception made for the 100% run, where we found better results with $\beta = 0.5$. We measured alignment and explicitness for both variants of GlanceNets by restricting on only those 10 latent factors where supervision were provided. This is in line with the notion of alignment, since we are interested in measuring the interpretability of the model, not the disentanglment among different components.

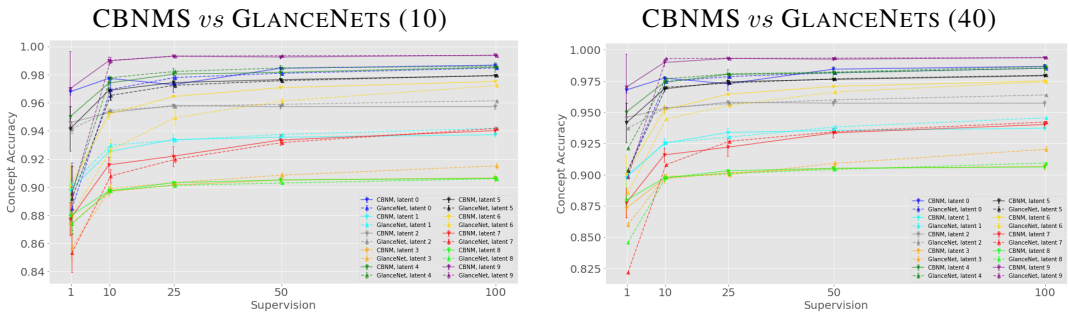


Figure 10: Concepts accuracy for CBNMs vs GlanceNets. Different colors refer to the distinct attributes for which supervision is provided. The solid line is reserved to CBNMs, whether GlanceNets are displayed with a dotted line. On the left, CBNMs vs GlanceNets with a latent space of dimension 10. On the right, CBNMs vs GlanceNets with TCVAE variant and a latent space of 40.

In Fig. 11 we report the results obtained, including the original variant with 10 latent dimensions. For the β-VAE (20) and TCVAE (40) we can see the improvement provided by extending the latent space. The latter achieves particular high values of alignment.

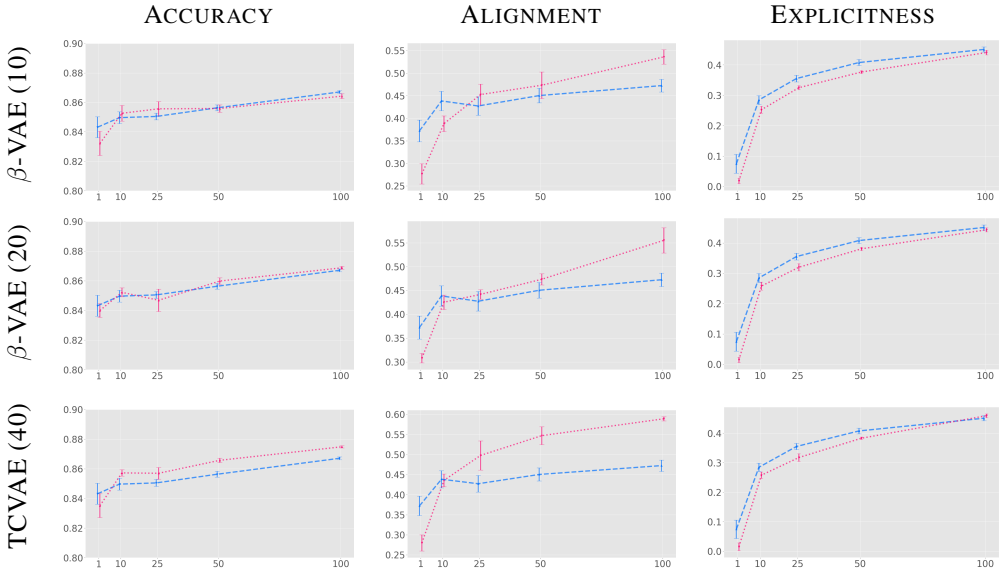


Figure 11: Accuracy, Alignment and Explicitness metrics for CBNMs vs GlanceNets. For each row we vary the comparison with variants of GlanceNets: β-VAE (10) refers to the model we reported in the main text, β-VAE (20) is a variant with 20 latent dimensions, and TCVAE (40) is the model based on a TCVAE with 40 latent dimensions.

F.3 Latent traversals

We finally report in the traversals for some of the supervised attributes, obtained by the GlanceNet TCVAE with full supervision on the concepts. We excluded the traversals of the attributes HAT and BALD since the generator failed to reproduce them faithfully. The others are well captured by the model, as we reported in Fig. 12.

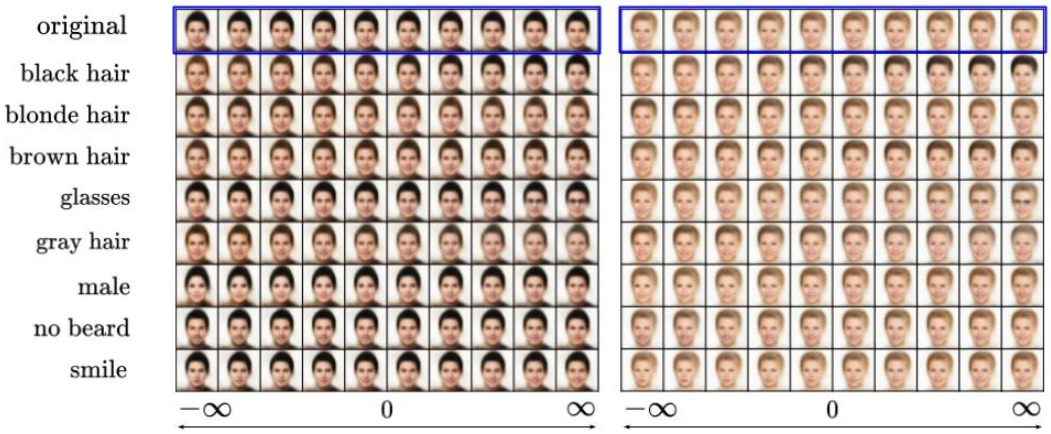


Figure 12: Latent traversals on two test images. In each row, we report the result of changing a single latent factor Z_i (from -5 to $+5$) while keeping fixed the others.