A Appendix

A.1 Concept Alignment Score Implementation Details

As discussed in Section 4, there is lack of agreed-upon metrics to use for evaluating the interpretability of concept-based XAI models. For example, while concept predictive accuracy is well defined for scalar concept representations (e.g., vanilla CBMs), there seems to be no clear metric for evaluating the "concept accuracy" of an embedding representation. Therefore, in this work we build upon this gap and propose the CAS score as a generalization of the concept predictive accuracy. Intuitively, if a concept representation is able to capture a concept correctly, then we would expect that clustering samples based on that representation would result in coherent clusters where samples within the same cluster all have the concept active or inactive. The CAS attempts to capture this by looking at how coherent clusters are for each concept representation using the known concept labels for each sample as we change the size of each cluster. This is formally computed via Equation 2 throught a repeated evaluation of Rosenberg et al.'s homogeneity score [19] for different clusterings.

Following Rosenberg and Hirschberg [19], we compute the homogeneity score as described in Section 4 by estimating the conditional entropy of ground truth concept labels C_i w.r.t. cluster labels Π_i, i.e. $H(C_i, \Pi_i)$, using a contingency table. This table is produced by our selected clustering algorithm κ, i.e. $A = \{a_{u,v}\}$ where $a_{u,v}$ is the number of data points that are members of class $c_i = v \in \{0, 1\}$ and elements of cluster $\pi_i = u \in \{1, \cdots, \rho\}$:

$$H(C_i, \Pi_i) = -\frac{\rho}{N} \sum_{u=1}^{\rho} \left(a_{u,0} \log \frac{a_{u,0}}{a_{u,0} + a_{u,1}} + a_{u,1} \log \frac{a_{u,1}}{a_{u,0} + a_{u,1}} \right) \tag{1}$$

Similarly, we compute the entropy of the ground truth concept labels C_i, i.e. $H(C)$, as:

$$H(C_i) = -\left(\frac{\sum_{u=1}^{\rho} a_{u,0}}{2} \log \frac{\sum_{u=1}^{\rho} a_{u,0}}{2} + \frac{\sum_{u=1}^{\rho} a_{u,1}}{2} \log \frac{\sum_{u=1}^{\rho} a_{u,1}}{2} \right) \tag{2}$$

When evaluating the CAS, we use $\delta = 50$ to speed up its computation across all datasets.

A.2 Kernel Density Estimation of Mutual Information

Following the approach of [35, 36] we approximate the Mutual Information (MI) through the Kernel Density Estimation (KDE) method. Kolchinsky et al. [35] show that this method accurately approximates the MI computed through the binning procedure proposed by Tishby et al. [22]. The KDE approach assumes that the activity of the analysed layer (in this case, the concept encoding layer $\hat{C}$) is distributed as a mixture of Gaussians. This approximation holds true if the input samples used for evaluation are representative of the true input distribution. Therefore, we can consider the input distribution as delta functions over each sample in the dataset. Moreover, Gaussian noise is added to the layer activity to bound the mutual information w.r.t. the input – i.e., $\hat{C} = \hat{\mathbf{c}} + \epsilon$, where $\hat{\mathbf{c}}$ is the bottleneck activation vector and $\epsilon \sim N(0, \sigma^2 I)$ is a noise matrix with noise variance σ^2. In this setting, the KDE estimation of the MI with the input is:

$$I(\hat{C}; X) = H(\hat{C}) - H(\hat{C}|X) = H(\hat{C}) \leq \frac{\zeta}{2} - \frac{1}{n} \sum_{i=1}^{n} \log \left(\frac{1}{n} \frac{1}{2\pi\sigma^2} \sum_{j=1}^{n} e^{\frac{||\hat{c}^{(i)} - \hat{c}^{(j)}||_2^2}{2\sigma^2}} \right), \tag{3}$$

where n is the number of input samples and ζ is the dimension of the concept encoding layer $\hat{C}$ (e.g., $\zeta = m \cdot k$ for CEM). Notice that Shwartz-Ziv and Tishby [27] neglect the conditional entropy term arguing that the output of any neural network layer is a deterministic function of the input, which implies $H(\hat{C}|X) = 0$.

When considering instead the mutual information w.r.t. the downstream task label distribution Y, the conditional entropy is $H(\hat{C}|Y) \neq 0$ and the mutual information $I(\hat{C}; Y)$ can be estimated as:

$$I(\hat{C};Y) = H(\hat{C}) - H(\hat{C}|Y) \leq \frac{\zeta}{2} - \frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{1}{n}\frac{1}{2\pi\sigma^2}\sum_{j=1}^{n} e^{\frac{||\hat{c}^{(i)}-\hat{c}^{(j)}||_2^2}{2\sigma^2}}\right) - \sum_{l=1}^{L} p_l \left[\frac{\zeta}{2} - \frac{1}{P_l}\sum_{\substack{i \\ \text{s.t. } y^{(i)}=l}} \log\left(\frac{1}{P_l}\frac{1}{2\pi\sigma^2}\sum_{\substack{j \\ \text{s.t. } y^{(j)}=l}} e^{\frac{||\hat{c}^{(i)}-\hat{c}^{(j)}||_2^2}{2\sigma^2}}\right)\right],$$

where L is the number of downstream task labels, P_l the number of data with output label l, and $p_l = P_l/n$ is the probability of task label l.

When considering the concept labels C, however, the same estimation cannot be employed since it requires the labels to be mutually exclusive. While this holds true for the task labels Y in the considered settings, the concepts in C are generally multi-labeled — i.e., more than one concept can be true when considering a single sample $\mathbf{x}^{(i)}$. Therefore, in this case we compute the average of the conditional entropies $H(\hat{C}|C) = 1/k\sum_a H(\hat{C}|C_a)$ across all k concepts. More precisely,

$$\begin{aligned} I(\hat{C};C) &= H(\hat{C}) - H(\hat{C}|C) \\ &= H(\hat{C}) - \frac{1}{k}\sum_{a=1}^{k} H(\hat{C}|C_a) \\ &\leq \frac{\zeta}{2} - \frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{1}{n}\frac{1}{2\pi\sigma^2}\sum_{j=1}^{n} e^{\frac{||\hat{\mathbf{c}}^{(i)}-\hat{\mathbf{c}}^{(j)}||_2^2}{2\sigma^2}}\right) \\ &\quad - \frac{1}{k}\sum_{a=1}^{k}\sum_{m\in M_a} p_{a,m}\left[\frac{\zeta}{2} - \frac{1}{P_{a,m}}\sum_{\substack{i \\ \text{s.t. } c_a^{(i)}=m}} \log\left(\frac{1}{P_{a,m}}\frac{1}{2\pi\sigma^2}\sum_{\substack{j \\ \text{s.t. } c_a^{(j)}=m}} e^{\frac{||\hat{\mathbf{c}}^{(i)}-\hat{\mathbf{c}}^{(j)}||_2^2}{2\sigma^2}}\right)\right], \end{aligned}$$

where $P_{a,m}$ is the number of samples having the concept $c_a = m$, M_k is the set of possible values that the c_a concept can assume (generally $M_a = \{0,1\}$), and $p_{a,m} = P_{a,m}/n$ is the probability of concept label $c_a = m$.

In all the previous cases, since we use the natural logarithm, the MI is computed in NATS. To convert it into bits, we scale the obtained values by $\frac{1}{\log(2)}$.

The role of noise The variance σ^2 of the noise matrix ϵ, plays an important role in the computation of the MI. More precisely, low values of σ entail high negative values for $H(\hat{C}|X)$, and, consequently, high positive values for $I(\hat{C};X)$. In the extreme case where we do not add any noise, we have $H(\hat{C}|X) = -\inf$ and $I(\hat{C};X) \sim \inf$, as long as the entropy $H(\hat{C})$ is finite. Furthermore, as we can observe in the equations above, the dimensionality ζ of the concept representation also plays an important role in the computation of the MI, the latter being directly proportional to the dimensionality of concept representation layer $\hat{C}$. To mitigate this issue, we also consider the noise to be directly proportional to the dimension of $\hat{C}$, by setting $\sigma^2 = \zeta/100$.

A.3 Datasets

A.3.1 XOR problem

The first dataset used in our experiments is inspired by the exclusive-OR (XOR) problem proposed by [24] to show the limitations of Perceptrons. We draw input samples from a uniform distribution in the unit square $\mathbf{x} \in [0,1]^2$ and define two binary concepts $\{c_1, c_2\}$ by using the Boolean (discrete) version of the input features $c_i = \mathbb{1}_{x_i>0.5}$. Finally, we construct a downstream task label using the XOR of the two concepts $y = c_1 \oplus c_2$.

A.3.2 Trigonometric dataset

The second dataset we use in our experiments is inspired by that proposed by Mahinpei et al. [14] (see Appendix D of their paper). Specifically, we construct synthetic concept-annotated samples from three independent latent normal random variables $h_i \sim \mathcal{N}(0, 2)$. Each of the 7 features in each sample is constructed via a non-invertible function transformation of the latent factors, where 3 features are of the form $(\sin(h_i) + h_i)$, 3 features of the form $(\cos(h_i) + h_i)$, and 1 is the nonlinear combination $(h_1^2 + h_2^2 + h_3^2)$. Each sample is then associated with 3 binary concepts representing the sign of their corresponding latent variables, i.e. $c_i = (h_i > 0)$. In order to make this task Boolean-undecidable from its binary concepts, we modify the downstream task proposed by Mahinpei et al. [14] by assigning each sample a label $y = \mathbb{1}_{(h_1+h_2)>0}$ indicating whether $h_1 + h_2$ is positive or not.

A.3.3 Dot dataset

As much as the Trigonometric dataset is designed to highlight that fuzzy concept representations generalize better than Boolean concept representations, we designed the Dot dataset to show the advantage of embedding concept representations over fuzzy concept representations. The Dot dataset is based on four 2-dimensional latent factors from which concepts and task labels are constructed. Two of these four vectors correspond to fixed reference vectors $\mathbf{w}_+$ and $\mathbf{w}_-$ while the remaining two vectors $\{\mathbf{v}_i\}_{i=1}^2$ are sampled from a 2-dimensional normal distribution:

$$\mathbf{v}_{1,2} \sim \mathcal{N}(\mathbf{0}, 2\,\mathbf{I}) \qquad \mathbf{w}_+ = [1 \quad 1]^T \qquad \mathbf{w}_- = -\mathbf{w}_+ \tag{4}$$

We then create four input features as the sum and difference of the two factors $\mathbf{v}_i$:

$$\mathbf{x} = [(\mathbf{v}_1 + \mathbf{v}_2) \quad (\mathbf{v}_1 - \mathbf{v}_2)]^T \tag{5}$$

From this, we create two binary concepts representing whether or not the latent factors $\mathbf{v}_i$ point in the same direction as the reference vectors $\mathbf{w}_j$ (as determined by their dot products):

$$\mathbf{c} = \left[\mathbb{1}_{(\mathbf{v}_1\cdot\mathbf{w}_1)>0} \quad \mathbb{1}_{(\mathbf{v}_2\cdot\mathbf{w}_2)>0}\right]^T \tag{6}$$

Finally, we construct the downstream task as determining whether or not vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ point in the same direction (as determined by their dot product):

$$y = \mathbb{1}_{(\mathbf{v}_1\cdot\mathbf{v}_2)>0} \tag{7}$$

A.3.4 Real-world datasets

Furthermore, we evaluate our methods on two real-world vision tasks: (1) the Caltech-UCSD Birds-200-2011 dataset (CUB, [16]), as prepared by [9], and the Large-scale CelebFaces Attributes dataset (CelebA, [25]).

CUB [16] In CUB we construct a dataset with complete concept annotations by using the same $k = 112$ bird attributes selected by Koh et al. [9] as binary concept annotations (e.g., *beak_type*, *wing_color*, etc ...) and using the bird identity ($l = 200$) as the downstream task. All images are preprocessed in the same fashion as in [9] by normalizing and randomly flipping and cropping each image during training. This results in a dataset of around 6,000 RGB images with sizes $(3, 299, 299)$ which are split into test, validation, and training sets using the same splits by Koh et al. [9]. In our evaluation, we use CUB to test CBMs in real-world tasks where we have a complete set of concept annotations w.r.t. the downstream task.

CelebA [25] In CelebA, we select the 8 most balanced attributes $[a_1, \cdots a_8]$ out of each image's 40 binary attributes, as defined by how close their distributions are to a random uniform binary distribution, and use attributes $[a_1, \cdots, a_6]$ as concepts annotations for each sample. To simulate a task in which complete concept annotations are lacking, each image in CelebA is assigned a label corresponding to the base-10 representation of the number formed by the binary vector $[a_1, \cdots, a_8]$, resulting in a total of $l = 2^8 = 256$ classes. Note that concept annotations in this task are incomplete as attributes a_7 and a_8 are needed for predicting the downstream task but they are not provided during training. To improve resource utilization and training times, we further reduce the size of the

CelebA dataset by randomly subsampling the dataset and selecting every 12^{th} sample during training and we downsample every image to have shape $(3, 64, 64)$. This results in a dataset with around 16,900 RGB images from which a train, validation, and test datasets are generated using a traditional 70%-10%-20% split. In our experiments, we use CelebA to evaluate CBMs in scenarios where the bottleneck is extremely narrow and incomplete w.r.t. the downstream task.

A.4 Effect of Concept Encoder Capacity

Different concept encoders will have different approximation capabilities, and the resulting concept representations will be affected by the architectural choices. To test whether the choice of a specific model might bias our results, here we show that the relative rankings across methods in our real-world tasks (CUB and CelebA) are preserved when using backbones with significantly different capacities i.e., a ResNet18 vs a ResNet34. Specifically, Figure A.1 compares the concept and task predictive accuracies of our baselines in CUB and CelebA when using different backbone capacities (trained while fixing all other hyperparamters are described in Appendix A.6). Notice that although we observe a drop in performance when using a ResNet18 backbone, this drop is similar across all baselines and therefore leads to our results having the same ranking as observed when using a ResNet34 backbone.

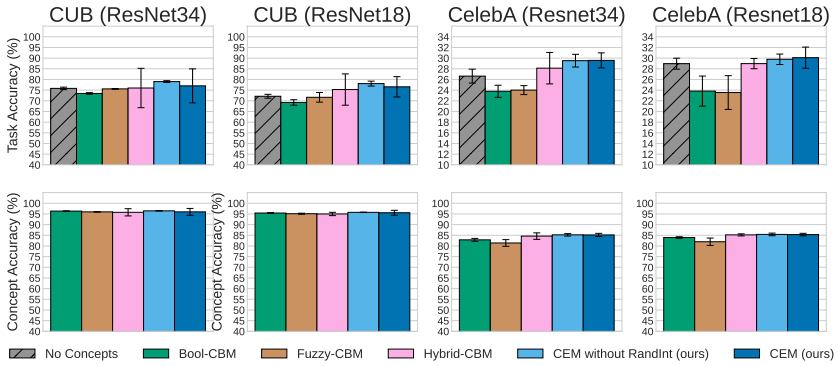


Figure A.1: Task and average concept accuracy when using a ResNet18 backbone vs a ResNet34 backbone in CUB and CelebA.

Similarly, Figure A.2 shows that the same rankings and results observed in Figure 6, where a ResNet34 backbone was used, can be seen when performing interventions in the baselines which use a ResNet18 backbone.

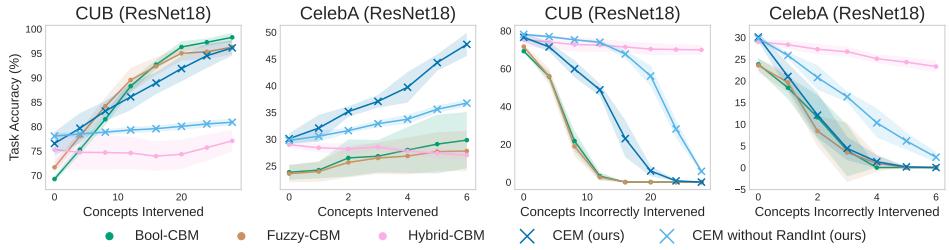
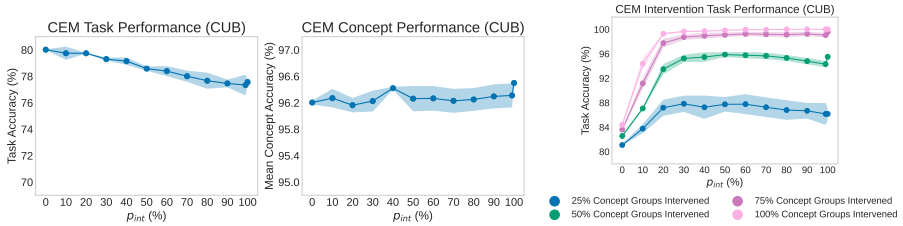


Figure A.2: Effects of performing positive random concept interventions (left and center left) and incorrect random interventions (center right and right) for different models with a ResNet18 backbone in CUB and CelebA. As in [9], when intervening in CUB we jointly set groups of mutually exclusive concepts.

A.5 RandInt Probability Ablation Study

Figure A.3 shows the results of varying p_{int} for CEMs trained on CUB (using the same training setup as defined in Appendix A.6). We observe that although there is a slight trade-off in validation task accuracy as we increase p_{int}, this trade-off is eclipsed compared to the concept intervention

capabilities which come by increasing p_{int}. Because of this, in our work we settle with $p_{\text{int}} = 0.25$ as this study shows that this value leverages good performance without interventions while enabling effective interventions.



(a) (b)

Figure A.3: Ablation study for p_{int} in CUB. (a) Task and concept validation accuracy of CEMs trained with different values of p_{int}. (b) Task validation accuracy when intervening on an increasing number of concept groups for CEMs trained with different values of p_{int}.

A.6 Training Details

Model Architectures For simplicity, we use the same DNN architectures across all synthetic tasks (i.e., *XOR*, *Trig*, *Dot*) unless specified otherwise. Specifically, we use an MLP with hidden layer sizes $\{128, 128\}$ and LeakyReLU activations for latent code generator ψ in CEM and concept encoder g in all CBM variants. When learning concept embedding representations in synthetic datasets, we learn embeddings with $m = 128$ activations.

In both CUB and CelebA, for latent code generator ψ in CEM and concept encoder g in all CBM variants we use a pretrained ResNet-34 model [37] with its last layer modified to output $n_{\text{hidden}} = m$ activations. When using CEM, we learn embeddings with $m = 16$ activations, smaller than in the synthetic datasets given the larger number of concepts in these tasks (see Appendix A.14 for an ablation study showing how the embedding size affects performance in CEM).

Across all datasets we always use a single fully connected layer for label predictor f and, for the sake of fairness, set $\gamma = k \cdot (m-1)$ when evaluating Hybrid CBMs. This is done so that the overall bottleneck of Hybrid-CBM has size $k+\gamma = k+k(m-1) = km$, just as in an equivalent CEM model. Notice therefore that the dimensionality of $\hat{\mathbf{c}}$ is k for Bool and Fuzzy CBMs while it is $k \cdot m$ for our Hybrid-CBM and CEM baselines. When training end-to-end models without concept supervision (i.e., our "No Concepts" baseline), we use the exact same architecture as in the Hybrid-CBM but provide no concept supervision in its bottleneck (equivalent to setting the weight for the concept loss to 0 during training). Finally, when using RandInt, we set $p_{\text{int}} = 0.25$, as empirically we observe that this yields good results across all datasets (see Appendix A.5 above).

Training Hyperparameters In all synthetic tasks, we generate datasets with 3,000 samples and use a traditional 70%-10%-20% random split for training, validation, and testing datasets, respectively. During training, we then set the weight of the concept loss to $\alpha = 1$ across all models. We then train all models for 500 epochs using a batch size of 256 and a default Adam [38] optimizer with learning rate 10^{-2}.

In CUB, we set the concept loss weight to $\alpha = 5$ in all models and, as in [9], we use a weighted cross entropy loss for concept prediction to mitigate imbalances in concept labels. All models in this task are trained for 300 epochs using a batch size of 128 and an SGD optimizer with 0.9 momentum and learning rate of 10^{-2}.

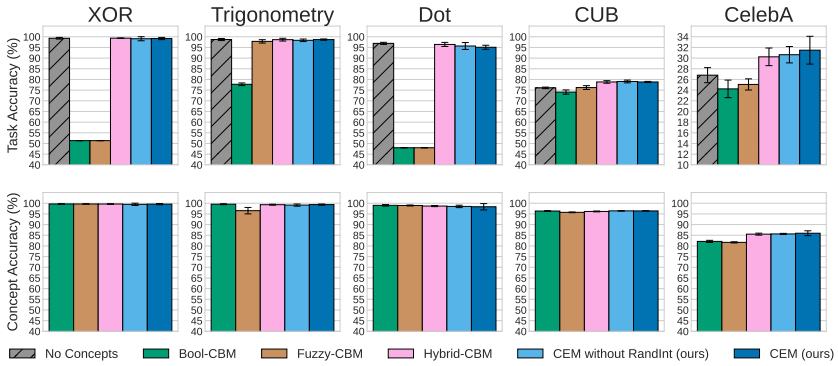
In our CelebA task, we fix the concept loss weight to $\alpha = 1$ in all models and also use a weighted cross entropy loss for concept prediction to mitigate imbalances in concept labels. All models in this task are trained for 200 epochs using a batch size of 512 and an SGD optimizer with 0.9 momentum and learning rate of 5×10^{-3} (different from CUB to avoid instabilities observed if the initial learning rate was too high).

In all models and tasks, we use a weight decay factor of $4e-05$ and scale the learning rate during training by a factor of 0.1 if no improvement has been seen in validation loss for the last 10 epochs.

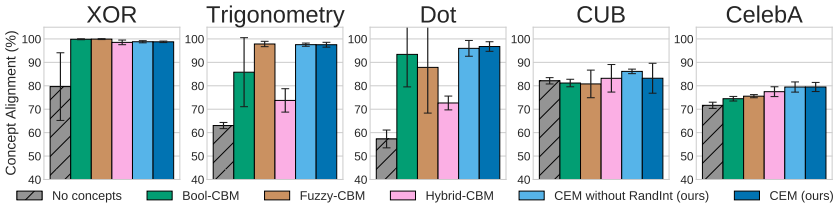
Furthermore, all models are trained using an early stopping mechanism monitoring validation loss and stopping training if no improvement has been seen for 15 epochs.

A.7 Task and Mean Concept Performance

In Figure A.4a we show the task and mean concept predictive performance of all of our baselines. Notice that as claimed in Section 5, all baselines are able to achieve a very similar mean concept accuracy but they have very distinct task accuracies, suggesting a that the interpretability-vs-accuracy trade-off is different across different models. For further clarity and to facilitate cross-comparison across methods and datasets, we also show our concept alignment scores in a bar-plot format in Figure A.4b.



(a)



(b)

Figure A.4: (a) Task and mean concept accuracy for all methods across all tasks. (b) Concept alignment scores for all methods across all tasks.

In Table 2 and Table 1 we report the same results in tabular form for clarity's sake. Notice how in CUB the baseline model without concept suppervision (i.e., "No Concepts") has a better CAS mean compared to Bool and Fuzzy CBMs. We hypothesize that because certain concepts in CUB tend to be activated only for specific classes (e..g, there is a very high imbalance in how concepts are activated across classes), clusters produced from the intermediate representations of a DNN trained to predict said classes will be highly coherent with respect to concepts that are class-specific, leading to high CAS scores. The same cannot be said of e.g., CelebA (where concept activations are highly balanced across different classes by design), which is why we observe the CAS in black-box DNNs being lower than that in CBM models.

Table 1: Task accuracy for all methods across all tasks reported with the mean and 95% confidence interval.

	No concepts	Boolean-CBM	Fuzzy-CBM	Hybrid-CBM	CEM (ours)
XOR	**99.33,** (99.01, 99.66)	51.33, (51.33, 51.33)	51.42, (51.42, 51.42)	**99.23,** (99.23, 99.23)	**99.17,** (98.71, 99.57)
Trigonometry	98.47, (98.47, 98.47)	77.77, (77.52, 77.99)	98.37, (98.37, 98.37)	**98.67,** (98.42, 98.90)	**98.43,** (97.79, 99.01)
Dot	**97.57,** (97.01, 98.09)	48.00, (48.00, 48.00)	48.17, (48.02, 48.31)	**96.67,** (96.67, 96.67)	**97.13,** (97.13, 97.13)
CUB	73.41, (71.83, 74.70)	67.11, (65.29, 68.56)	**72.98,** (70.39, 76.30)	**70.70,** (64.28, 77.68)	**77.11,** (75.89, 78.10)
CelebA	26.80, (25.90, 27.84)	24.23, (24.23, 24.23)	25.07, (24.36, 25.81)	**30.24,** (29.13, 31.41)	**30.63,** (29.62, 31.74)

Table 2: Concept alignment scores for all methods across all tasks reported with the mean and 95% confidence interval.

	No concepts	Boolean-CBM	Fuzzy-CBM	Hybrid-CBM	CEM (ours)
XOR	79.65, (71.32, 89.12)	99.86, (99.86, 99.86)	99.92, (99.92, 99.92)	98.53, (97.88, 99.10)	98.79, (98.50, 99.06)
Trigonometry	63.02, (62.18, 63.66)	85.80, (85.80, 85.80)	97.84, (97.84, 97.84)	73.75, (73.75, 73.75)	97.55, (97.11, 97.93)
Dot	57.31, (53.80, 57.31)	93.40, (85.22, 99.57)	87.86, (75.03, 98.24)	72.66, (70.68, 74.26)	95.98, (94.90, 97.16)
CUB	82.12, (81.49, 82.69)	81.18, (80.12, 82.09)	80.79, (79.36, 82.75)	83.19, (79.81, 85.78)	86.14, (85.50, 86.68)
CelebA	71.66, (71.66, 71.66)	74.48, (73.87, 75.08)	75.56, (75.16, 75.91)	77.48, (77.48, 77.48)	79.47, (78.43, 80.33)

A.8 Concept Subsampling in CUB

All concept bottleneck models require datasets containing concept annotations, which may be costly to acquire. Here we compare a CBM's robustness when concept annotations are scarce. We simulate this scenario by randomly selecting a random subsample of the 112 concepts in our CUB task which we then use as annotations for all models during training (all models are trained using the same architecture and training hyperparameters as our CUB model in Section 5). As we observe in Figure A.5, the task and concept accuracy of both CEMs and Hybrid-CBMs are only mildly affected by the reduction in concept supervisions, as opposed to Bool and Fuzzy CBMs. In both CEMs and Hybrid-CBMs this robustness allows a dramatic reduction of required concept annotations and the costs related to acquiring such annotations. Nevertheless, as seen in Section 5.4, notice that although Hybrid-CBM performs well in concept scarsity, it is unable to effectively react to human concept interventions (a crucial limitation that CEM is able to overcome).

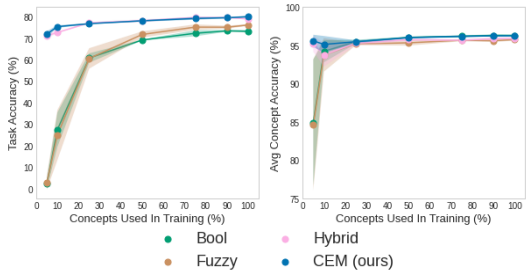


Figure A.5: Task and average concept accuracies when using a percentage of the available concepts in our CUB task during training. All points are generated by sampling, uniformly at random, 5 different concept subsets at training time and averaging all metrics.

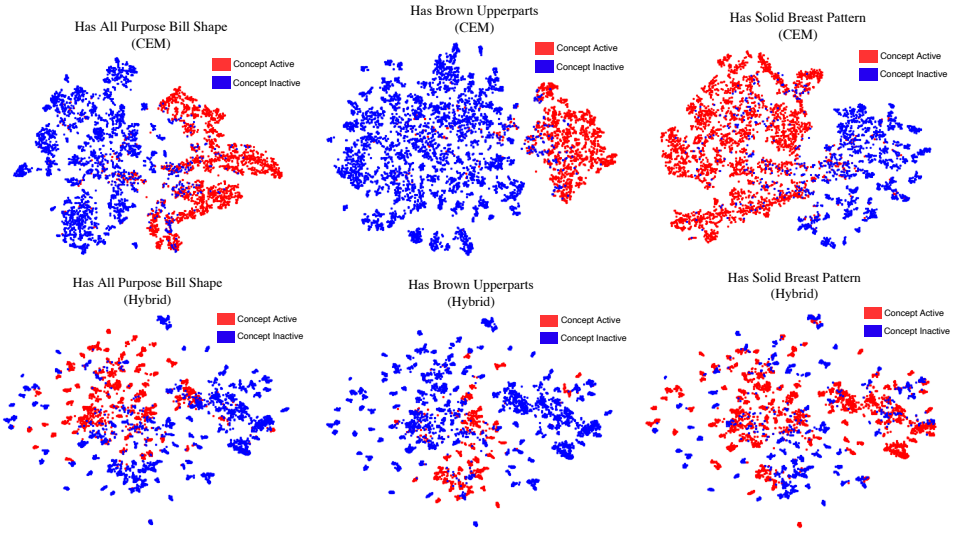
A.9 Bottleneck Representation Experiment Details

To explore our hypothesis that the high alignment observed in CEM's representations may lead to its embeddings forming more interpretable representations than Hybrid's embeddings, we evaluate the power of their learnt bottlenecks as representations for different tasks. With this aim, we train a Hybrid-CBM and a CEM, both with the same architecture as described for models trained on CUB in Appendix A.6, on a variation of CUB with only 25% of its concept annotations randomly selected before training. This results on a total of $k = 28$ concepts being randomly selected to be provided as supervision for both models. We then train these models to convergence using the same training setup as in CUB models described in Appendix A.6 resulting in a Hybrid-CBM with $77.15\% \pm 0.33\%$ test task accuracy and $95.3\% \pm 0.31\%$ test mean concept accuracy. In contrast, its CEM counterpart achieved $76.76\% \pm 0.27\%$ test task accuracy and $95.47\% \pm 0.19\%$ test mean concept accuracy.

Once trained, we use the bottleneck representations learn by both the Hybrid-CBM and the CEM to predict the remaining 75% of the concept annotations in CUB using a simple logistic linear model. In other words, for each concept not used to train each of these models (of which there are $112 - 28 = 84$ of them) we train a linear probe to predict the concept's true value from the entire bottleneck representations learnt by both our Hybrid-CBM and CEM models. We do this for a total of 5 randomly initialized Hybrid-CBMs and CEMs and observe that the probes trained using the Hybrid-CBM's bottleneck have a mean concept accuracy of $91.83\% \pm 0.51\%$ while the probes trained using CEM's bottleneck have a mean concept accuracy of $94.33\% \pm 0.88\%$.

A.10 More Qualitative Results

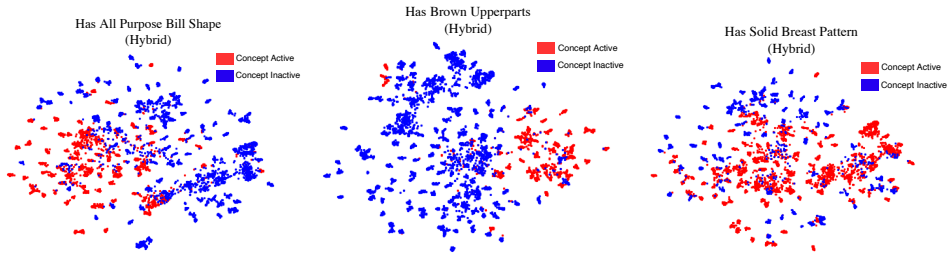
In this section we show further qualitative results which highlight the same trends observed in Section 5. Specifically, we see via the t-SNE [28] plots shown in Figure A.6 that the concept representations learnt by Hybrid-CBMs are more visually entangled than those learnt by CEM. Notice that because in Hybrid-CBM we use $\hat{\mathbf{c}}_i = \hat{\mathbf{c}}_{[k:k+\gamma]}$ as the embedding learnt for concept c_i, all Hybrid-CBM t-SNE plots shown in Figure A.6 have a very similar arrangement and differ only in their coloring.



(a) (b) (c)

Figure A.6: t-SNE visualisations of CEM and Hybrid-CBM concept embeddings for concepts (a) "has all purpose bill shape", (b) "has brown upperparts", and (c) "has solid breast pattern". Each visualised test sample point is coloured red if the concept is active in that sample and blue otherwise. Concepts displayed in this figure were selected at random. All t-SNE plots are generated using a perplexity of 30 and running the optimization for $1,500$ iterations.

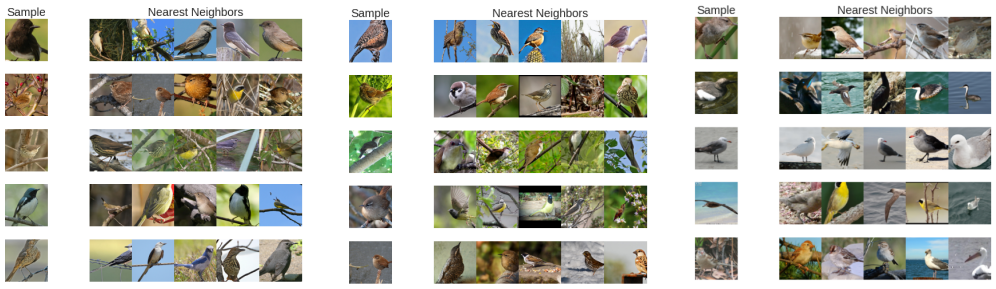
Moreover, Figure A.7 shows that even when we include the concept probability as part of a concept's embedding in the Hybrid model (i.e., we let $\hat{\mathbf{c}}_i = \left[\hat{\mathbf{c}}_{[k:k+\gamma]}, \hat{\mathbf{c}}_{[i:(i+1)]}\right]^T$ rather than $\hat{\mathbf{c}}_i = \hat{\mathbf{c}}_{[k:k+\gamma]}$ as before), we still observe similar entanglement within the latent space learnt for each concept in Hybrid-CBMs. This suggests that even when one includes a highly-discriminative feature, such as the probability of a concept being activated as part of the Hybrid-CBM's embeddings, the resulting representation is far from being easily separable w.r.t. its ground truth concept activation.



(a) (b) (c)

Figure A.7: t-SNE visualisations of Hybrid-CBM concept embeddings for concepts (a) "has all purpose bill shape", (b) "has brown upperparts", and (c) "has solid breast pattern". In contrast to the t-SNE plots shown in Figure A.6, when producing these results we include the concept probability as part of the concept embedding learnt by Hybrid-CBM. All t-SNE plots are generated using a perplexity of 30 and running the optimization for $1,500$ iterations.

Finally, Figure A.8 shows that the coherency observed in Figure 5c is seen across different learnt concept representations.



(a) (b) (c)

Figure A.8: Five nearest Euclidean neighbours to random test samples for concept embeddings (a) "has all purpose bill shape", (b) "has brown upperparts", and (c) "has solid breast pattern".

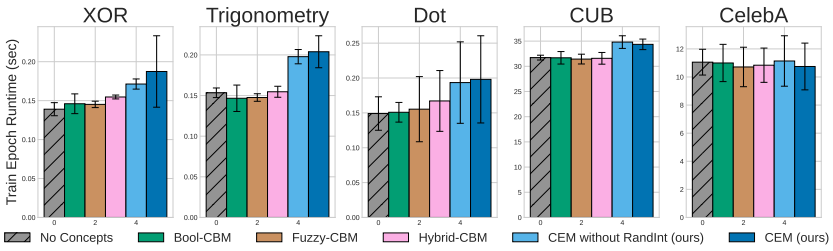
A.11 Computational Cost of CEM

As mentioned in our Conclusion, CEM's use of an three linear layers (two for producing $\hat{\mathbf{c}}_i^+$ and $\hat{\mathbf{c}}_i^-$ and one for generating $\hat{p}_i$) leads to CEM requiring more FLOPs than vanilla CBMs per training epoch. Therefore, in this section we compare the computational cost of training CEM w.r.t. standard CBMs, by studying (i) the average runtime of one training epoch (Figure A.9a) and (ii) the average number of epochs taken for each method until convergence as dictated by our early stopping mechanism (Figure A.9b). We observe that overall CEM does not incur in statistically significantly different training convergence times than other baselines. Similarly, as expected we see that a training step in CEM does require more FLOPs than vanilla CBMs (we empirically observe less than 10% time increases in large datasets such as CUB and CelebA). Nevertheless, given its performance improvements showcased in Section 5, and its positive reaction to interventions, we believe that these small computational costs are justified.

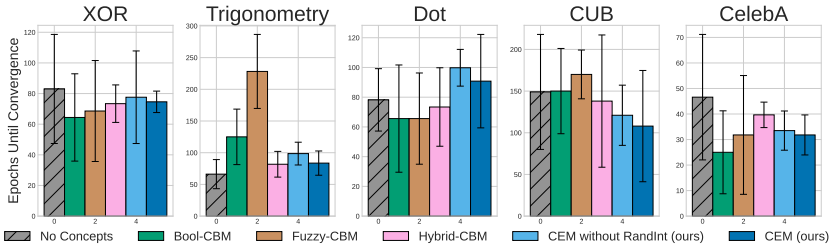
Furthermore, we note that including RandInt in CEM does not significantly increase the training time in practice. This is due to the fact that its subroutine can be implemented using a simple multiplicative Bernoulli mask of the predictive concept probability vector.

A.12 Effect of RandInt in standard CBMs

RandInt is a form of regularization that we specifically designed to applicable to CEM's use of a positive and negative concept embedding. Its purpose is to incentivize each embedding to be better aligned with the ground truth semantics it represents so that their use in interventions is more effective. Nevertheless, as it is formulated in Section 3.2, it is possible to apply it to other kinds of CBMs (e.g., Fuzzy and Hybrid CBMs). When applied to other kinds of models, however, it may not have the intended effect. For example, in vanilla CBMs where there is no extra capacity in the bottleneck, RandInt will behave in a similar way to a dropout regularizer and may instead force the label predictor to depend less on a specific concept activation when the concepts are an incomplete description of the task (therefore leading to possibly worse responses to concept interventions). Notice that this does not happen in CEM as during training RandInt still allows gradients to flow and update the weights that generate the "correct" embedding, letting the model modify this embedding so that it is aligned with its intended semantics. On the other hand, if the concepts are a complete description of the downstream task, then, as p_{int} approaches 1, we expect RandInt's use in a CBM to behave similarly to how a independently-trained CBM behaves (where the concept encoder and label predictor models are trained separately). This means that, as shown in [9], it may lead to some improvements in how effective interventions are. To verify this, and for a fair comparison across methods, we train all CBM baselines with our RandInt regularizer ($p_{\text{int}} = 0.25$ as in the rest of experiments). As hypothesized, we observe in Figure A.10 that RandInt seems in fact to hurt the performance of standard CBMs in concept-incomplete tasks (e.g., CelebA) while it adds small performance improvements in concept-complete tasks (e.g., CUB). More importantly, however, notice that our main result of our intervention results in Section 5.4 still hold: CEM still significantly



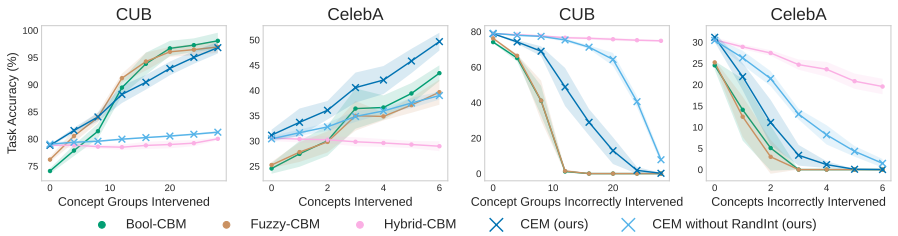
(a)



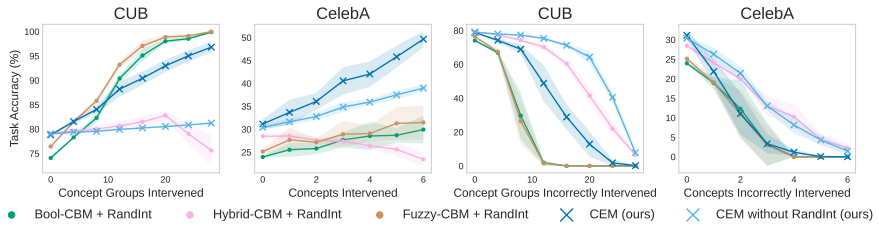
(b)

Figure A.9: Computational cost of CEM compared to other baselines. (a) Average wall-clock runtime (in seconds) for one training epoch of each model. (b) Average number of training epochs performed until early stopping concluded the training run (recall we use a patience of 15 epochs).

outperforms Hybrid-CBMs, its closest competitor, even when the Hybrid model is trained with RandInt.



(a)



(b)

Figure A.10: Task accuracy after interventions with and without RandInt for all methods. (a) Task accuracy after both "correct" and "incorrect" interventions for models trained without RandInt. (b) Task accuracy after both "correct" and "incorrect" interventions for models trained with RandInt.

A.13 Intervention Experiment Details

Setup For our intervention results discussed in Section 5, for each method we train 5 different models using different random seeds. Then, when intervening on a model $\mathcal{M}$ by correcting d of its concepts at test-time, we select the same random subset of d concepts we will intervene on for all

models trained with the same initial seed as $\mathcal{M}$. Given that several CUB concept annotations are mutually exclusive (e.g., "has white wings" and "has brown wings"), following [9] when intervening in models trained in this task we jointly set groups of mutually exclusive concepts to their ground truth values. This results in a total of 28 groups of mutually exclusive concepts in CUB which we intervene on.

Exploring effects of different training procedures in CBM interventions Previous work by Koh et al. [9] suggests that CBMs trained sequentially (where the concept encoder is trained first and then frozen when training the label predictor) or independently (where the concept encoder and label predictor are trained independently of each other and then composed at the end to produce a CBM) can sometimes outperform jointly trained CBMs when expert interventions are introduced. In this section we explore whether the results shown in Figure 6 would differ if one compares our model against sequentially and independently trained Fuzzy-CBMs.

Figure A.11 shows how CEMs react to interventions compared to sequentially and independently trained CBMs. Notice that the observed trends in these results are not so different than those seen when comparing CEMs against jointly-trained CBMs: in concept completeness (e.g., CUB), Fuzzy-CBMs (with the exception of Sequential-CBMs which seem to underperform) tend to react better to correct interventions than CEM but can quickly drop their performance if these interventions are not correct. In stark contrast, however, in concept-incomplete settings such as in CelebA, we see that Sequential and Independent CBMs experience mild performance improvements when correct interventions are performed, leading to CEMs outperforming these models by a large margin. These results suggest that our observations in Section 5 hold even if one changes the training process for a Fuzzy-CBM and highlight that CEMs are the only models in our evaluation capable of maintain high performance both in concept-complete and concept-incomplete settings.

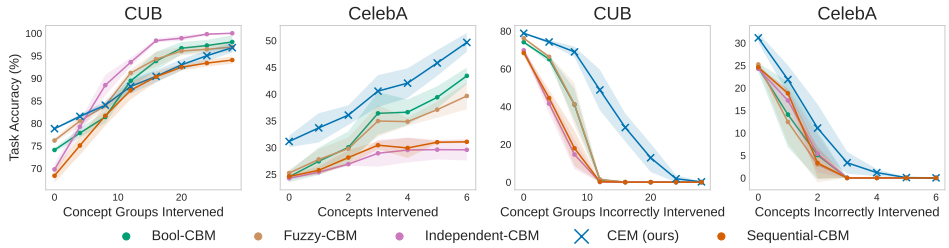
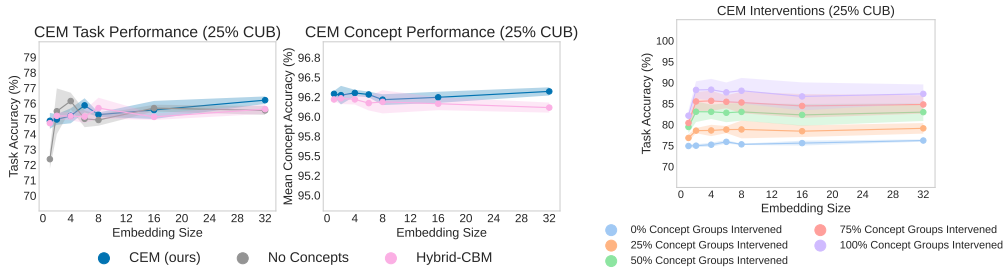


Figure A.11: Effects of performing positive random concept interventions (left and center left) and incorrect random interventions (center right and right) for different training regimes for CBMs (Joint, Sequential, and Independent). For clarity, Hybrid is not included in this plot (see Figure 6 for those results).

A.14 Embedding Size Ablation Study

In this section we explore the effects of the embedding size m in CEMs and compare their performance as m varies against that of Hybrid-CBMs and end-to-end black box models with equal capacity. For this, we train CEMs, Hybrid-CBMs, and end-to-end black box models on CUB (with only 25% of its concept annotations being selected) and CelebA using the same architectures and training configurations as described in Appendix A.6. We chose to reduce the number of concept annotations in CUB to better study how our model behaves when the raw number of activations in its bottleneck (which is equal to $(k \cdot m)$ in CEMs) is severely constrained. We show our results in Figure A.12 and Figure A.13.

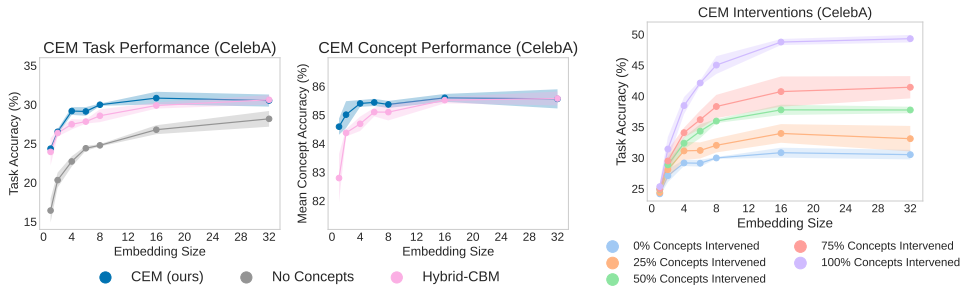
Our study shows that, for both tasks, after enough capacity is provided to CEMs (which for our particular datasets seems to be around 8-16 activations per embedding), our models are able to perform better or competitively against end-to-end black box models and Hybrid-CBMs. In particular, we see that with the exception of very small embedding sizes, CEM tends to outperform Hybrid-CBM models with equal capacity, suggesting that introducing a fully supervised bottleneck can in fact help in both task and mean concept performance. Similarly, we see that with the exception of when the embedding size is $m = 2$ in CUB, CEMs are able to perform equally as well or better than end-to-end black box architectures with equal capacity. Furthermore, notice that even in the case



(a) (b)

Figure A.12: Ablation study for m in CUB when only 25% of its concept annotations are used during training. (a) Task and concept validation accuracy of CEMs with different embedding sizes. For comparison, we include Hybrid-CBMs and end-to-end black box models with equal bottleneck capacity as their CEM counterpart for each value of m. (b) Task validation accuracy when intervening on an increasing number of concept groups for CEMs with different embedding sizes.

where end-to-end black box models outperform CEM (as in $m = 2$ for CUB), the difference in task accuracy is less than 1.5%, a hit which may not be detrimental if one takes into account the fact that CEM produces highly-accurate concept-based explanations and it is able to significantly surpass the performance of end-to-end black box model if interventions in its concept bottleneck are allowed. Finally, we similarly see for both tasks that interventions have similar effects on models after a moderately sized embedding is used, therefore suggesting there is no benefit in increasing the embedding size significantly if one is interested in interventions. These two studies suggest that unless the embedding size is drastically constrained (e.g., $m \leq 4$), CEM's performance is stable with respect to the embedding size used, aiding with hyperparameter selection and allowing CEMs to be more easily integrated into other architecture designs.



(a) (b)

Figure A.13: Ablation study for m in CelebA. (a) Task and concept validation accuracy of CEMs with different embedding sizes. (b) Task validation accuracy when intervening on an increasing number of concepts for CEMs with different embedding sizes.

A.15 Code, Licences, and Resources

Libraries For our experiments, we implemented all baselines and methods in Python 3.7 and relied upon open-source libraries such as PyTorch 1.11 [39] (BSD license) and Skelearn [40] (BSD license). To produce the plots seen in this paper, we made use of Matplotlib 3.5 (BSD license). We have released all of the code required to recreate our experiments in an MIT-licensed public repository[5].

Resources All of our experiments were run on a private machine with 8 Intel(R) Xeon(R) Gold 5218 CPUs (2.30GHz), 64GB of RAM, and 2 Quadro RTX 8000 Nvidia GPUs. We estimate that approximately 240-GPU hours were required to complete all of our experiments.

[5]Code can be found at https://github.com/mateoespinosa/cem/

References

[1] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[2] Michelle Goddard. The EU General Data Protection Regulation (GDPR): European regulation that has a global impact. *International Journal of Market Research*, 59(6):703–705, 2017.

[3] Gabriëlle Ras, Marcel van Gerven, and Pim Haselager. Explanation methods in deep learning: Users, values, concerns and challenges. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 19–36. Springer, 2018.

[4] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *arXiv preprint arXiv:2103.11251*, 2021.

[5] Juan Manuel Durán and Karin Rolanda Jongsma. Who is afraid of black box algorithms? On the epistemological and ethical basis of trust in medical AI. *Journal of Medical Ethics*, 47(5): 329–335, 2021.

[6] Samuele Lo Piano. Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward. *Humanities and Social Sciences Communications*, 7(1): 1–7, 2020.

[7] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, 31:841, 2017.

[8] EUGDPR. GDPR. General data protection regulation, 2017.

[9] 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.

[10] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3681–3688, 2019.

[11] 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.

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

[13] Max W Shen. Trust in AI: Interpretability is not necessary or sufficient, while black-box interaction is necessary and sufficient. *arXiv preprint arXiv:2202.05302*, 2022.

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

[15] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33:20554–20565, 2020.

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

[17] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[18] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[19] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[20] Leonard Kaufman and Peter J Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, 344:68–125, 1990.

[21] Lucie Charlotte Magister, Dmitry Kazhdan, Vikash Singh, and Pietro Liò. Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks. *arXiv preprint arXiv:2107.11889*, 2021.

[22] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[23] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE information theory workshop (ITW)*, pages 1–5. IEEE, 2015.

[24] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 1969.

[25] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[26] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*, 2018.

[27] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[28] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

[29] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[30] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[31] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[32] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

[33] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

[34] Chi Li, M Zeeshan Zia, Quoc-Huy Tran, Xiang Yu, Gregory D Hager, and Manmohan Chandraker. Deep supervision with intermediate concepts. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1828–1843, 2018.

[35] Artemy Kolchinsky, Brendan D Tracey, and David H Wolpert. Nonlinear information bottleneck. *Entropy*, 21(12):1181, 2019.

[36] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=ry_WPG-A-`.

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

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] 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. *arXiv preprint arXiv:1912.01703*, 2019.

[40] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.