Interpretable Neural-Symbolic Concept Reasoning

Pietro Barbiero [*1] **Gabriele Ciravegna** [*2] **Francesco Giannini** [*3] **Mateo Espinosa Zarlenga** [1]
Lucie Charlotte Magister [1] **Alberto Tonda** [4] **Pietro Lió** [1] **Frederic Precioso** [2] **Mateja Jamnik** [1]
Giuseppe Marra [*5]

Abstract

Deep learning methods are highly accurate, yet their opaque decision process prevents them from earning full human trust. Concept-based models aim to address this issue by learning tasks based on a set of human-understandable concepts. However, state-of-the-art concept-based models rely on high-dimensional concept embedding representations which lack a clear semantic meaning, thus questioning the interpretability of their decision process. To overcome this limitation, we propose the *Deep Concept Reasoner* (DCR), the first interpretable concept-based model that builds upon concept embeddings. In DCR, neural networks do not make task predictions directly, but they build syntactic rule structures using concept embeddings. DCR then executes these rules on meaningful concept truth degrees to provide a final interpretable and semantically-consistent prediction in a differentiable manner. Our experiments show that DCR: (i) improves up to $+25\%$ w.r.t. state-of-the-art interpretable concept-based models on challenging benchmarks (ii) discovers meaningful logic rules matching known ground truths even in the absence of concept supervision during training, and (iii), facilitates the generation of counterfactual examples providing the learnt rules as guidance.

1. Introduction

The opaque decision process of deep learning (DL) models has failed to inspire human trust despite their state-of-the-art performance across multiple tasks (Rudin, 2019; Bussone et al., 2015), raising ethical (Durán & Jongsma, 2021; Lo Piano, 2020) and legal (Wachter et al., 2017; EUGDPR, 2017) concerns. For this reason, interpretability is now a core research topic in the field of responsible AI (Rudin, 2019).

Concept-based models (Kim et al., 2018; Chen et al., 2020) aim to increase human trust in deep learning models by using human-understandable concepts to train interpretable models—such as logistic regression or decision trees (Rudin, 2019; Koh et al., 2020; Kazhdan et al., 2020) (Figure 1). This approach significantly increases human trust in the AI predictor (Rudin, 2019; Shen, 2022) as it allows users to clearly understand a model's decision process. However, state-of-the-art concept-based models, which rely on concept embeddings (Yeh et al., 2020; Kazhdan et al., 2020; Mahinpei et al., 2021; Espinosa Zarlenga et al., 2022) to attain high performance, are not completely interpretable. Indeed, concept embeddings lack clear semantics on individual dimensions, e.g., $\hat{\mathbf{c}}_{\text{yellow}} = [2.3, 0.3, -3.5, \dots]^T$ does not have semantics assigned to each of its dimensions. This sacrifice of interpretability in favour of model capacity leads to a possible reduction in human trust when using these models, as argued by Rudin (2019); Mahinpei et al. (2021).

In this paper, we propose the *Deep Concept Reasoner*[1] (DCR, Section 3), the first interpretable concept-based model building on concept embeddings. DCR applies differentiable and learnable modules on concept embeddings to build a set of fuzzy rules which can then be executed on semantically meaningful concept truth degrees to provide a final interpretable prediction. Our experiments (Section 4) show that DCR: (i) attains better task accuracy than state-of-the-art interpretable concept-based models (Figure 1), (ii) discovers meaningful logic rules, matching known ground truths even in absence of training concept supervision, and (iii) facilitates the generation of counterfactual examples thanks to the highly-interpretable learnt rules.

[*]Equal contribution [1]University of Cambridge, Cambridge, UK [2]Université Côte d'Azur, Inria, CNRS, I3S, Maasai, Nice, France [3]University of Siena, Siena, Italy [4]INRA, Université Paris-Saclay, Thiverval-Grignon, France [5]KU Leuven, Leuven, Belgium. Correspondence to: Pietro Barbiero <pb737@cam.ac.uk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]Code available in public repository: `https://github.com/pietrobarbiero/pytorch_explain`.

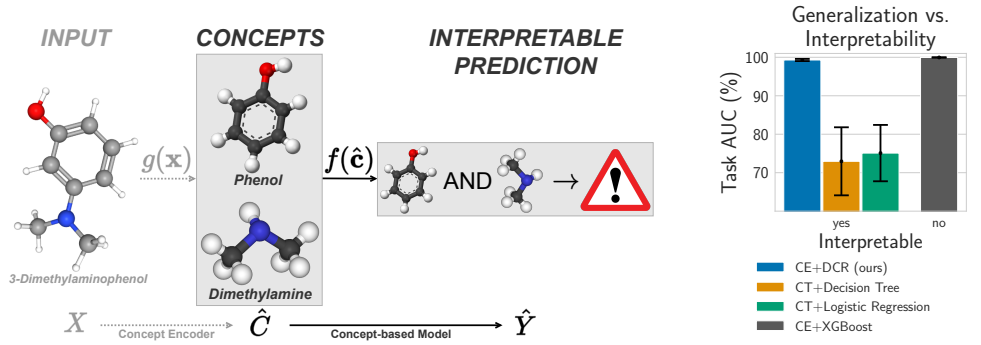

Figure 1. (a) An interpretable concept-based model f maps concepts $\hat{C}$ to tasks $\hat{Y}$ generating an interpretable rule. When input features are not semantically meaningful, a concept encoder g can map raw features to a concept space. (b) The proposed approach (DCR) outperforms interpretable concept-based models in the *Dot* dataset. *CE* stands for concept embeddings and *CT* for concept truth values.

2. Preliminaries

Concept-based models Concept-based models $f : C \rightarrow Y$ learn a map from a concept space C to a task space Y (Yeh et al., 2020). If concepts are semantically meaningful, then humans can interpret this mapping by tracing back predictions to the most relevant concepts (Ghorbani et al., 2019a). When the features of the input space are hard for humans to reason about (such as pixel intensities), concept-based models work on the output of a concept-encoder mapping $g : X \rightarrow C$ from the input space X to the concept space C (Ghorbani et al., 2019b; Koh et al., 2020). In general, training a concept-based model may require a dataset where each sample consists of input features $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ (e.g., an image's pixels), k ground truth concepts $\mathbf{c} \in \mathcal{C} \subseteq \{0,1\}^k$ (i.e., a binary vector with concept annotations, when available) and o task labels $\mathbf{y} \in \mathcal{Y} \subseteq \{0,1\}^o$ (e.g., an image's classes). During training, a concept-based model is encouraged to align its predictions to task labels i.e., $\mathbf{y} \approx \hat{\mathbf{y}} = f(g(\mathbf{x}))$. Similarly, a concept encoder can be supervised when concept labels are available i.e., $\mathbf{c} \approx \hat{\mathbf{c}} = g(\mathbf{x})$. When concept labels are not available, they can still be extracted from pre-trained models associating concept labels to clusters found in their embeddings as proposed by Ghorbani et al. (2019b); Magister et al. (2021). We indicate concept and task predictions as $\hat{c}_i = (g(\mathbf{x}))_i$ and $\hat{y}_j = (f(\hat{\mathbf{c}}))_j$ respectively.

Concept truth values vs. concept embeddings Usually, concept-based models represent concepts using their truth degree, that is, $\hat{c}_1, \ldots, \hat{c}_k \in [0,1]$. However, this representation might significantly degrade task accuracy as observed by Mahinpei et al. (2021) and Espinosa Zarlenga et al. (2022). To overcome this issue, concept-based models may represent concepts using concept embeddings $\hat{\mathbf{c}}_i \in \mathbb{R}^m$ alongside their truth degrees $\hat{c}_i \in [0,1]$.[2] While this increases task accuracy of concept-based models (Espinosa Zarlenga et al., 2022), it also weakens their interpretability as concept embeddings lack clear semantics.

Fuzzy logic rules Continuous fuzzy logics (Hájek, 2013) extend Boolean logic by relaxing discrete truth-values in $\{0,1\}$ to truth degrees in $[0,1]$, and Boolean connectives to (differentiable) real-valued operators. In particular, a t-norm $\wedge : [0,1] \times [0,1] \rightarrow [0,1]$ generalises the Boolean conjunction while a t-conorm $\vee : [0,1] \times [0,1] \rightarrow [0,1]$ generalises the disjunction. These two operators are connected by the strong negation $\neg$, defined as $\neg x = 1 - x$. For example, the product (fuzzy) logic can be defined by the operators $x \wedge y := x \cdot y$ and $x \vee y := x + y - xy$. As in Boolean logic, the syntax of a t-norm fuzzy rule includes: (i) Atomic formulas consisting of *propositional variables* z, and logical *constants* $\bot$ (`false`, "0") and $\top$ (`true`, "1"), (ii) *Literals* representing atomic formulas or their negation, and (iii) Logical *connectives* $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ joining formulas in arbitrarily complex compound formulas.

3. Deep Concept Reasoning

Here we describe the "Deep Concept Reasoner" (DCR, Figure 2), the first interpretable concept-based model based on concept embeddings. Similarly to existing models based on concept embeddings, DCR exploits high-dimensional representations of the concepts. However, in DCR, such representations are only used to compute a logic rule. The final prediction is then obtained by evaluating such rules on the concepts' truth values and not on their embeddings, thus maintaining clear semantics and providing a totally interpretable decision. Being differentiable, DCR is trainable as an independent module on concept databases, but it can also be trained end-to-end with differentiable concept encoders. In the following section, we describe (1) the

[2] With an abuse of notation, we use the same symbol for a concept embedding and its corresponding truth degree, with the former in bold to distinguish it.

syntax of the rules we aim to learn (Section 3.1), (2) how to (neurally) generate and execute learnt rules to predict task labels (Section 3.2), (3) how DCR learns simple rules in specific t-norm semantics (Section 3.2), and (4) how we can generate global and counterfactual explanations with DCR (Section 3.4). We provide Figure 2 as a reference to graphically follow the discussion.

3.1. Rule syntax

To understand the rationale behind DCR's design, we begin with an illustrative toy example:

Example 3.1. *Consider the problem of defining the fruit "banana" given the vocabulary of concepts "soft", "round", and "yellow". A simple definition can be* $y_{banana} \Leftrightarrow \neg c_{round} \wedge c_{yellow}$. *From this rule we can deduce that (i) being "soft" is irrelevant for being a "banana" (indeed bananas can be both soft or hard), and (ii) being both "not round" and "yellow" is relevant to being a "banana".*

As in this example, DCR rules can express whether a concept is ***relevant*** or not (e.g., "soft"), and whether a concept plays a positive (e.g., "yellow") or negative (e.g., "not round") ***role***. To formalize this description of rule syntax, we let l_{ji} denote the literal of concept c_i (i.e., $\hat{c}_i$ or $\neg\hat{c}_i$) representing the *role* of the concept i for the j-th class. Similarly, we let $r_{ji} \in \{0, 1\}$ representing whether $\hat{c}_i$ is *relevant* for predicting the class y_j. For each sample $\mathbf{x}$ and predicted class $\hat{y}_j$, DCR learns a rule with the following syntax[3]:

$$\hat{y}_j \Leftrightarrow \bigwedge_{i:\, r_{ji}=1} l_{ji} \tag{1}$$

Such a rule defines a logical statement for why a given sample is predicted to have label $\hat{y}_j$ using a conjunction of relevant concept literals (i.e., $\hat{c}_i$ or $\neg\hat{c}_i$).

3.2. Rule generation and execution

Having defined the syntax of DCR rules, we describe how to *generate* and *execute* these rules in a differentiable way. To generate a rule we use two neural modules ϕ_j and ψ_j which determine the role and relevance of each concept, respectively. Then, we execute each rule using the concepts' truth degrees of a given sample. We split this process into three steps: (i) learning each concept's roles, (ii) learning each concept's relevance, and (iii) predicting the task using the relevant concepts.

Concept role Generation: To determine the *role* (positive/negative) of a concept, we use a feed-forward neural network $\phi_j : \mathbb{R}^m \to [0, 1]$, with m being the dimension of each concept embedding. The neural model ϕ_j takes as input a concept embedding $\hat{\mathbf{c}}_i \in \mathbb{R}^m$ and returns a soft indicator representing the role of the concept in the formula, that is, whether in literal l_{ji} the concept should appear negated (e.g., $\phi_{banana}(\hat{\mathbf{c}}_{round}) = 0$) or not (e.g., $\phi_{banana}(\hat{\mathbf{c}}_{yellow}) = 1$). Execution: When we execute the rule, we need to compute the actual truth degree of a literal l_{ji} given its role $\phi(\hat{\mathbf{c}}_i)$. We define this truth degree $\ell_{ji} \in [0, 1]$. In particular, we want to (i) forward the same truth degree of the concept, i.e. $\ell_{ji} = \hat{c}_i$, when $\phi(\hat{\mathbf{c}}_i) = 1$, and (ii) negate it, i.e. $\ell_{ji} = \neg\hat{c}_i$, when $\phi(\hat{\mathbf{c}}_i) = 0$. This behaviour can be generalized by a fuzzy equality $\Leftrightarrow$ when both ϕ_j and $\hat{c}$ are fuzzy values, i.e.:

$$\ell_{ji} = (\phi_j(\hat{\mathbf{c}}_i) \Leftrightarrow \hat{c}_i) \tag{2}$$

Example 3.2. *For a given object consider* $\hat{c}_{round} = 0$ *and* $\phi_{banana}(\hat{\mathbf{c}}_{round}) = 0$. *Then we get* $\ell_{banana,round} = (\phi_{banana}(\hat{\mathbf{c}}_{round}) \Leftrightarrow \hat{c}_{round}) = \neg\hat{c}_{round} = 1$. *If instead we had* $\phi_{banana}(\hat{\mathbf{c}}_{round}) = 1$, *then* $\ell_{banana,round} = (\phi_{banana}(\hat{\mathbf{c}}_{round}) \Leftrightarrow \hat{c}_{round}) = 0$.

Concept relevance. Generation: To determine the *relevance* of a concept $\hat{\mathbf{c}}_i$, we use another feed-forward neural network $\psi_j : \mathbb{R}^m \to [0, 1]$. The model ψ_j takes as input a concept embedding $\hat{\mathbf{c}}_i \in \mathbb{R}^m$ and returns a soft indicator representing the likelihood of a concept being relevant for the formula (e.g., $\psi_{banana}(\hat{\mathbf{c}}_{soft}) = 1$) or not (e.g., $\psi_{banana}(\hat{\mathbf{c}}_{yellow}) = 0$). Execution: When we execute the rule, we need to compute the truth degree of a literal given its relevance r_{ji}. We define the truth degree of a relevant literal as $\ell^r_{ji} \in [0, 1]$, where r stands for "relevant". In particular, we want to (i) filter irrelevant concepts when $\psi_j(\hat{\mathbf{c}}_i) = 0$ by setting $\ell^r_{ji} = 1$, and (ii) retain relevant literals when $\psi_j(\hat{\mathbf{c}}_i) = 1$ by setting $\ell^r_{ji} = \ell_{ji}$. This behaviour can be generalized to fuzzy values of ψ_j as follows:

$$\ell^r_{ji} = (\psi_j(\hat{\mathbf{c}}_i) \Rightarrow \ell_{ji}) = (\neg\psi_j(\hat{\mathbf{c}}_i) \vee \ell_{ji}) \tag{3}$$

Note that setting $\ell^r_{ji} = 1$ makes the literal l_{ji} irrelevant since "1" is neutral w.r.t. the conjunction in Equation 4.

Example 3.3. *For a given object of type "banana", let the concept "soft" be irrelevant, that is* $\psi_{banana}(\hat{\mathbf{c}}_{soft}) = 0$. *Then we get* $\ell^r_{banana,soft} = (\psi_{banana}(\hat{\mathbf{c}}_{soft}) \Rightarrow \ell_{banana,soft}) = 1$, *independently from the content of* $\hat{c}_{soft}$ *or* $\ell_{banana,soft}$. *Conversely, let the concept "yellow" by relevant, that is* $\psi_{banana}(\hat{\mathbf{c}}_{yellow}) = 1$, *and let its concept literal be* $\ell_{banana,yellow} = \hat{\mathbf{c}}_{yellow} = 1$. *As a result, we get* $\ell^r_{banana,yellow} = (\psi_{banana}(\hat{\mathbf{c}}_{yellow}) \Rightarrow \ell_{banana,yellow}) = 1$.

Task prediction Finally, we conjoin the relevant literals ℓ^r_{ji} to obtain the task prediction $\hat{y}_j$:

$$\hat{y}_j = \bigwedge_{i=1}^{k} \ell^r_{ji} \tag{4}$$

[3] Here and in all equations we omit the explicit dependence on $\mathbf{x}$ for simplicity, i.e., we write $\hat{y}_j$ for $\hat{y}_j(\mathbf{x})$.

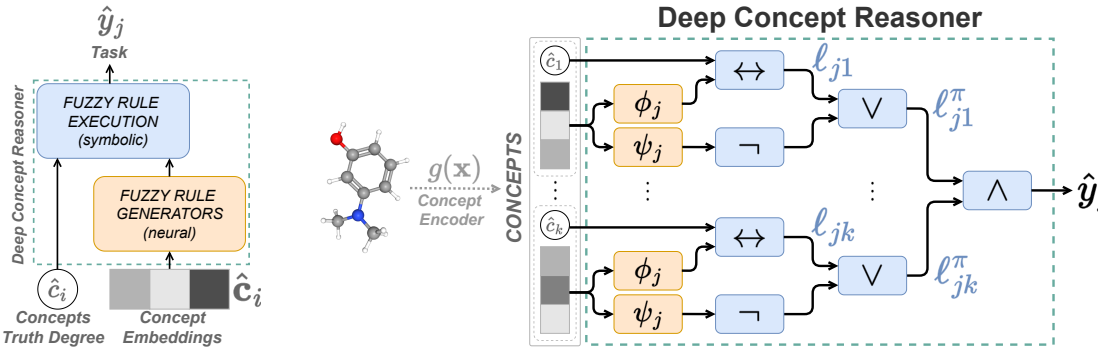

Figure 2. (left) Deep Concept Reasoner (DCR) generates fuzzy logic rules using neural models on concept embeddings. Then DCR executes the rule using the concept truth degrees to evaluate the rule symbolically. (right) Schema of DCR modules: first neural models ϕ and ψ generate the rule, and then the rule is executed symbolically.

Example 3.4. *For a given object of type "banana", consider the following truth degrees for the concepts:* $\hat{c}_{soft} = 1, \hat{c}_{round} = 0, \hat{c}_{yellow} = 1$. *Consider also the following values for the role and relevance of the class "banana":* $\phi_{banana}(\hat{\mathbf{c}}_i) = [0, 0, 1]$ *and* $\psi_{banana}(\hat{\mathbf{c}}_i) = [0, 1, 1]$ *for* $i \in \{soft, round, yellow\}$. *Then, we obtain the final prediction for class banana as:*

$$\begin{aligned}\hat{y}_{banana} = \textstyle\bigwedge_{i=1}^{3} (\neg\psi_{banana}(\hat{\mathbf{c}}_i) \vee (\phi_{banana}(\hat{\mathbf{c}}_i) \Leftrightarrow \hat{c}_i)) = \\ = (1 \vee (0 \Leftrightarrow 1)) \wedge (0 \vee (0 \Leftrightarrow 0)) \wedge (0 \vee (1 \Leftrightarrow 1)) = \\ = (1 \vee 0) \wedge (0 \vee 1) \wedge (0 \vee 1) = 1 \wedge 1 \wedge 1 = 1\end{aligned}$$

We remark that the models ϕ_j and ψ_j: (a) generate fuzzy logic rules using concept embeddings which might hold more information than just concept truth degrees, and (b) do not depend on the number of input concepts which makes them applicable—without retraining—in testing environments where the set of concepts available differs from the set of concepts used during training. We also remark that the whole process is differentiable as the neural models ϕ_j and ψ_j are differentiable as well as the fuzzy logic operations as we will see in the next section.

3.3. Rule parsimony and fuzzy semantics

Rule parsimony Simple explanations and logic rules are easier to interpret for humans (Miller, 1956; Rudin, 2019). We can encode this behaviour within the DCR architecture by enforcing a certain degree of competition among concepts to make only relevant concepts survive. To this end, we design a special activation function for the neural network ψ_j rescaling the output of a log-softmax activation:

$$\gamma_{ji} = \log\left(\frac{\exp(\text{MLP}_j(\hat{\mathbf{c}}_i))}{\sum_{i'=1}^{k}\exp(\text{MLP}_j(\hat{\mathbf{c}}_{i'}))}\right) \tag{5}$$

$$r_{ji} = \psi_j(\hat{\mathbf{c}}_i) = \sigma\left(\gamma_{ji} - \frac{1}{k}\sum_{i'=1}^{k}\gamma_{ji'}\right) \tag{6}$$

This way, if the scores γ_{ji} are uniformly distributed, then we expect the network ψ_j to select half of the concepts. We can also parametrise this function by introducing a parameter $\tau \in [-\infty, \infty]$ that allows a user to bias the default behaviour of the activation function: $r_{ji} = \sigma(\gamma_{ji} - \frac{\tau}{k}\sum_{i'=1}^{k}\gamma_{ji'})$. A user can increase τ to get more relevance scores closer to 1 (more complex rules) or decrease it to get more relevance scores closer to 0 (simpler rules).

Fuzzy semantics To create a semantically valid model, we enforce the same semantic structure in all logic and neural operations. Moreover, to train our model end-to-end, we need these semantics to be differentiable in all its operations, including logic functions. Marra et al. (2020c) describe a set of possible t-norm fuzzy logics which can serve the purpose. In our experiments, we use the Gödel t-norm. With this semantics, we can rewrite Equation 2 as:

$$\begin{aligned}\ell_{ji} &= \phi_j(\hat{\mathbf{c}}_i) \Leftrightarrow \hat{c}_i = (\phi_j(\hat{\mathbf{c}}_i) \Rightarrow \hat{c}_i) \wedge (\hat{c}_i \Rightarrow \phi_j(\hat{\mathbf{c}}_i)) = \\ &= (\neg\phi_j(\hat{\mathbf{c}}_i) \vee \hat{c}_i) \wedge (\neg\hat{c}_i \vee \phi_j(\hat{\mathbf{c}}_i)) = \\ &= \min\{\max\{1 - \phi_j(\hat{\mathbf{c}}_i), \hat{c}_i\}, \max\{1 - \hat{c}_i, \phi(\hat{\mathbf{c}}_i)\}\}\end{aligned}$$

and Equation 4 as: $\hat{y}_j = \min_{i=1}^{k}\{\max\{1 - \psi_j(\hat{\mathbf{c}}_i), \ell_{ji}\}\}$

3.4. Global and counterfactual explanations

Interpreting global behaviour In general, DCR rules may have different weights and concepts for different samples. However, we can still globally interpret the predictions of our model without the need for an external post-hoc explainer. To this end, we collect a batch of (or all) fuzzy rules generated DCR on the training data $\mathcal{X}_{\text{train}}$. Following Barbiero et al. (2022), we then Booleanize the collected rules and aggregate them with a global disjunction to get a single logic formula valid for all samples of class j:

$$\hat{y}_j^C = \bigvee_{\mathbf{x}\in\mathcal{X}_{\text{train}}} \hat{y}_j(\mathbf{x}) \tag{7}$$

This way we obtain a global overview of the decision process of our model for each class.

Counterfactual explanations Logic rules clearly reveal which concepts play a key role in a prediction. This transparency, typical of interpretable models, facilitates the extraction of simple counterfactual explanations without the need for an external algorithm as in Abid et al. (2021). In DCR we extract simple counter-examples $x^\star$ using the logic rule as guidance. Following Wachter et al. (2017), we generate counter-examples as close as possible to the original sample $|x - x^\star| < \epsilon$. In particular, Wachter et al. (2017) proposes to perturb the input features of a model starting from the most relevant features. As the decision process depends mostly on the most relevant features, perturbing a small set of features is usually enough to find counter-examples. To this end, we first rank the concepts present in the rule according to their relevance scores. Then, starting from the most relevant concept, we invert their truth value until the prediction of the model changes. The new rule represents a counterfactual explanation for the original prediction.

4. Experiments

4.1. Research questions

In this section, we analyze the following research questions:

- **Generalization —** How does DCR generalize on unseen samples compared to interpretable and neural-symbolic models? How does DCR generalize when concepts are unsupervised?
- **Interpretability —** Can DCR discover meaningful rules? Can DCR re-discover ground-truth rules? How stable are DCR rules under small perturbations of the input compared to interpretable models and local post-hoc explainers? How long does it take to extract a counterfactual explanation from DCR compared to a non-interpretable model?

4.2. Experimental setup

Data & task setup We investigate our research questions using six datasets spanning three of the most common data types used in deep learning: tabular, image, and graph-structured data. We use the three benchmark datasets (*XOR*, *Trigonometry*, and *Dot*) proposed by Espinosa Zarlenga et al. (2022) as they capture increasingly complex concept-to-label relationships, therefore challenging concept-based models. To test the DCR's ability to re-discover ground-truth rules we use the *MNIST-Addition* dataset (Manhaeve et al., 2018), a standard benchmark for neural-symbolic systems where one aims to predict the sum of two digits from the MNIST's dataset. Furthermore, we evaluate our methods on two real-world benchmark datasets: the Large-scale CelebFaces Attributes (*CelebA*, (Liu et al., 2015)) and the *Mutagenicity* (Morris et al., 2020) dataset. In particular, we define a new *CelebA* task to simulate a real-world condition of concept "shifts" where train and test concepts are correlated (e.g., "beard" and "mustaches") but do not match exactly. To this end, we split the set of *CelebA* attributes defined by Espinosa Zarlenga et al. (2022) in two partially disjoint sets and use one set of attributes for training models and one for testing. Finally, we use *Mutagenicity* as a real-world scenario the concept encoder is unsupervised. As *Mutagenicity* does not have concept annotations, we first train a graph neural network (GNN) on this dataset, and then we use the Graph Concept Explainer (GCExplainer, (Magister et al., 2021)) to extract a set of concepts from the embeddings of the trained GNN. For dataset with concept labels instead, we generate concept embeddings and truth degrees by training a Concept Embedding Model (Espinosa Zarlenga et al., 2022).

Baselines We compare DCR against interpretable models, such as logistic regression (Verhulst, 1845), decision trees (Breiman, 2017), as well as state-of-the-art black-box classifiers, such as extreme gradient boosting (XGBoost) (Chen & Guestrin, 2016), and locally-interpretable neural models, such as the Relu Net (Ciravegna et al., 2023). We train all baseline models in two different conditions mapping concepts to tasks either using concept truth degrees or using concept embeddings (baselines marked with *CT* and *CE* in figures, respectively). We consider interpretable only baselines trained on concept truth degrees only, as concept embeddings lack of clear semantics assigned to each dimension. However, baselines trained on concept embeddings still provide a strong reference for task accuracy w.r.t. interpretable models. On the *MNIST-Addition* dataset we compare DCR with state-of-the-art neural-symbolic baselines including: DeepProbLog (Manhaeve et al., 2018), DeepStochLog (Winters et al., 2022), Logic Tensor Networks (Badreddine et al., 2022), and Embed2Sym (Aspis et al., 2022). This is possible as the *MNIST-Addition* dataset provides access to the full set of ground-truth rules, allowing us to train these neural-symbolic systems. Finally, we compare DCR interpretability with interpretable models, such as logistic regression and decision trees, and with local post-hoc explainers, such as the Local Interpretable Model-agnostic Explanations (LIME, (Ribeiro et al., 2016)) applied on XGBoost.

Evaluation We assess each model's performance and interpretability based on four criteria. First, we measure task generalization using the Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores (Hand & Till, 2001) (the higher the better). Second,

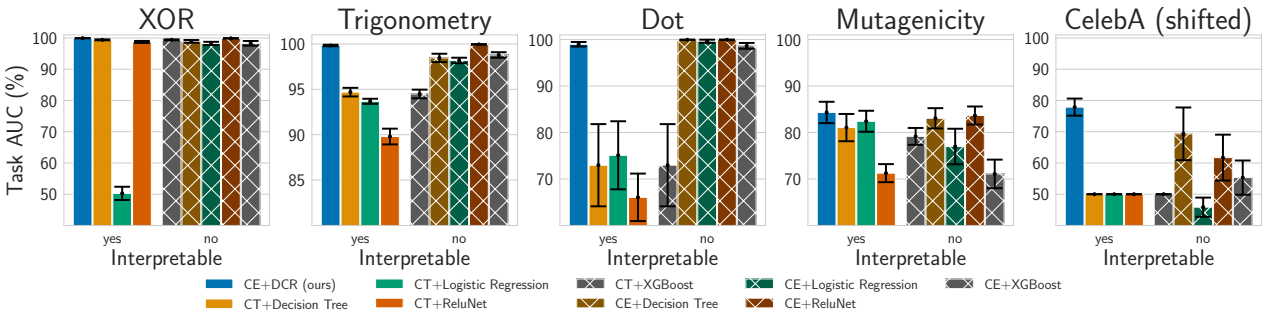

Figure 3. Mean ROC AUC for task predictions for all baselines across all tasks (the higher the better). DCR often outperforms interpretable concept-based models. *CE* stands for concept embeddings, while *CT* for concept truth degrees. Models trained on concept embeddings are not interpretable as concept embeddings lack a clear semantic for individual embedding dimensions.

we evaluate DCR interpretability by comparing the learnt logic formulae with ground-truth rules in *XOR*, *Trigonometry*, and *MNIST-Addition* datasets, and indirectly on *Mutagenicity* by checking whether the learnt rules involve concepts corresponding to functional groups known for their harmful effects, as done by Ying et al. (2019). Third, to further assess interpretability, we measure the sensitivity of the predictions under small perturbations following Yeh et al. (2019) (the lower the better). Finally, we measure how receptive our model is to extract meaningful counterfactual examples from its rules by computing the number of concept perturbations required to obtain a counterfactual example following Wachter et al. (2017) (the lower the better). For each metric, we report their mean and 95% Confidence Intervals (CI) on our test sets using 5 different initialization seeds. We report further details and results in the appendix.

4.3. Task generalization

DCR outperforms interpretable models (Figure 3) Our experiments show that DCR generalizes significantly better than interpretable benchmarks in our most challenging datasets. This improvement peaks when concept embeddings hold more information than concept truth degrees, as in the *CelebA* and *Dot* tasks where this deficit of information is imposed byconstruction (Espinosa Zarlenga et al., 2022). This grants DCR a significant advantage (up to $\sim 25\%$ improvement in ROC-AUC) over the other interpretable baselines. This phenomenon confirms the findings by Mahinpei et al. (2021) and Espinosa Zarlenga et al. (2023). In particular, the concept shift in *CelebA* causes interpretable models to behave almost randomly as the set of test concepts is different from the set of train concepts (despite being correlated). DCR however still generalizes well as the mechanism generating rules only depends on concept embeddings and the embeddings hold more information on the correlation between train and test concepts w.r.t. concept truth degrees. To further test this hypothesis, we compare DCR against XGBoost, decision trees (DTs), and logistic regression trained on concept embeddings. In most cases, concept embeddings allow DTs and logistic regression to improve task generalization, but the predictions of such models are no longer interpretable. In fact, even a logic rule whose terms correspond to dimensions of a concept embedding is not semantically meaningful as discussed in Section 2. In contrast, DCR uses concept embeddings to assemble rules whose terms are concept truth degrees, which makes it possible to keep the rules semantically meaningful.

DCR matches the accuracy of neural-symbolic systems trained using human rules (Table 1) Our experiments show that DCR generates rules that, when applied, obtain accuracy levels close to neural-symbolic systems trained using human rules, currently representing the gold standard to benchmark rule learners. We show this result on the *MNIST-Addition* dataset (Manhaeve et al., 2018), a standard benchmark in neural-symbolic AI, where the labels on the concepts are not available. We learn concepts without supervision by adding another task classifier, which only uses very crisp $\hat{c}_i$ to make the task predictions (see Appendix H). DCR achieves similar performance to state-of-the-art neural-symbolic baselines (within 1% accuracy from the best baseline). However, DCR is the only system discovering logic rules directly from data, while all the other baselines are trained using ground-truth rules. Therefore, this experiment indicates how DCR can learn meaningful rules also without concept supervision while still maintaining state-of-the-art performance.

4.4. Interpretability

DCR discovers semantically meaningful logic rules (Table 2) Our experiments show that DCR induces logic rules that are both accurate in predicting the task and formally correct when compared to ground-truth logic rules. We evaluate the formal correctness of DCR rules on the *XOR*, *Trigonometry*, and *MNIST-Addition* datasets where we have access to ground-truth logic rules. We report a selection

Table 1. Task accuracy on the *MNIST-addition* dataset. The neural-symbolic baselines use the knowledge of the symbolic task to distantly supervise the image recognition task. DCR achieves similar performances even though it learns the rules from scratch.

MODEL	ACCURACY (%)
With ground truth rules	
DeepProbLog	97.2 ± 0.5
DeepStochLog	97.9 ± 0.1
Embed2Sym	97.7 ± 0.1
LTN	98.0 ± 0.1
Without ground truth rules	
DCR(ours)	97.4 ± 0.2

of Booleanized DCR rules with the corresponding ground truth rules in Table 2. Our results indicate that DCR's rules align with human-designed ground truth rules, making them highly interpretable. For instance, DCR predicts that the sum of two MNIST digits is 17 if either the first image is a (i.e., c'_9) and the second is an (i.e., c''_8) or vice-versa which we can interpret globally using Equation 7 as: $y_{17} \Leftrightarrow (c'_9 \wedge c''_8) \vee (c'_8 \wedge c''_9)$. We list all logic rules discovered by DCR on the *MNIST-Addition* dataset in Appendix H. It is interesting to investigate the potential of DCR also in settings where we do not have access to the ground-truth logic rules, such as the *Mutagenicity* dataset. Here, unlike the *MNIST addition* dataset, not only there is no supervision on the concepts, but we don't even know which are the concepts. We use GCExplainer (Magister et al., 2021) to generate a set of concept embeddings from the embeddings of a trained GNN. We then use these embeddings to train DCR. In this setting, we can only evaluate the correctness of a DCR rules indirectly by checking whether the concepts appearing in the rules correspond to functional groups known for their harmful effects within the *Mutagenicity* dataset following Ying et al. (2019). Interestingly, many of DCR's rules predicting mutagenic effects include functional groups such as phenols (Hättenschwiler & Vitousek, 2000) and dimethylamines (ACGIH®, 2016), which can be highly toxic when combined in molecules such as 3-Dimethylaminophenols (Sabry et al., 2011). This suggests that DCR has the potential to unveil semantically meaningful relations among concepts and to make them explicit to humans by means of the learnt rules. We provide experimental details with the full list of concepts and rules discovered in *Mutagenicity* in Appendix C.

DCR rules are stable under small perturbations (Figure 4) An important characteristic of local explanations is to be stable under small perturbations (Yeh et al., 2019). Indeed, users do not trust explanations if they change significantly on very similar inputs for which the model makes the same prediction. This metric, also known as explanation sensitivity, is generally computed as the maximum change

Table 2. Error rate of Booleanised DCR rules w.r.t. ground truth rules. Error rate represents how often the label predicted by a Booleanised rule differs from the fuzzy rule generated by our model. The error rate is reported with the mean and standard error of the mean. A full list of logic rules for MNIST is in Appendix H.

GROUND-TRUTH RULE	PREDICTED RULE	ERROR (%)
XOR		
$y_0 \leftarrow \neg c_0 \wedge \neg c_1$	$y_0 \leftarrow \neg c_0 \wedge \neg c_1$	0.00 ± 0.00
$y_0 \leftarrow c_0 \wedge c_1$	$y_0 \leftarrow c_0 \wedge c_1$	0.00 ± 0.00
$y_1 \leftarrow \neg c_0 \wedge c_1$	$y_1 \leftarrow \neg c_0 \wedge c_1$	0.02 ± 0.02
$y_1 \leftarrow c_0 \wedge \neg c_1$	$y_1 \leftarrow c_0 \wedge \neg c_1$	0.01 ± 0.01
Trigonometry		
$y_0 \leftarrow \neg c_0 \wedge \neg c_1 \wedge \neg c_2$	$y_0 \leftarrow \neg c_0 \wedge \neg c_1 \wedge \neg c_2$	0.00 ± 0.00
$y_1 \leftarrow c_0 \wedge c_1 \wedge c_2$	$y_1 \leftarrow c_0 \wedge c_1 \wedge c_2$	0.00 ± 0.00
MNIST-Addition		
$y_{18} \leftarrow c'_9 \wedge c''_9$	$y_{18} \leftarrow c'_9 \wedge c''_9$	0.00 ± 0.00
$y_{17} \leftarrow c'_9 \wedge c''_8$	$y_{17} \leftarrow c'_9 \wedge c''_8$	0.00 ± 0.00
$y_{17} \leftarrow c'_8 \wedge c''_9$	$y_{17} \leftarrow c'_8 \wedge c''_9$	0.00 ± 0.00

in the explanation of a model $\Phi(f)$ on a slightly perturbed input ($x^\star$), that is, $|\Phi(f(\mathbf{x}^\star)) - \Phi(f(\mathbf{x}))|, |\mathbf{x} - \mathbf{x}^\star|_\infty < \epsilon$. We compare the DCR explanations w.r.t. our interpretable baselines as well as w.r.t. LIME (Ribeiro et al., 2016) explaining the output of XGBoost. Since we are using different types of models, we use a normalised version of the sensitivity $|\Phi(f(\mathbf{x}^\star)) - \Phi(f(\mathbf{x}))|/|\Phi(f(\mathbf{x}))|$. We compute the distance between two explanations considering the feature importance of the original explanation w.r.t. to the feature importance of the explanation for the perturbed example. For decision tree's rules, we consider the distance between the original path and the path of the perturbed example. As highlighted in Figure 4, in all datasets the explanations provided by DCR are very stable, particularly w.r.t. LIME and ReluNet. Notice that the figure does not report the explanation sensitivity of logistic regression and decision tree because it is trivially zero as they learn fixed rules for the entire dataset. The area under the sensitivity curves of all methods together with further details concerning this experiment has been reported in Appendix F.

DCR explains prediction mistakes In DCR, task predictions are obtained by executing the logic rules. In this sense, the rules transparently represent the model behavior, and they can explain misclassifications at the level of tasks. For example, reading DCR rules we can observe that a task was mispredicted because some concepts have been predicted wrongly, or the relevance scores are selecting a suboptimal set of concepts. To test this, we analyze the mispredicted test samples in datasets where we have access to ground truth rules as reference (XOR and Trigonometry). Interestingly, DCR is able to identify a mislabeled sample in the XOR dataset (first row of Table 3), highlighting an error in the data generation process. In fact, the rule learnt by DCR is correct $y = 0 \leftarrow \neg c_0 \wedge \neg c_1$ but the ground truth label was incorrect $y = 1$. In the Trigonometric dataset instead,

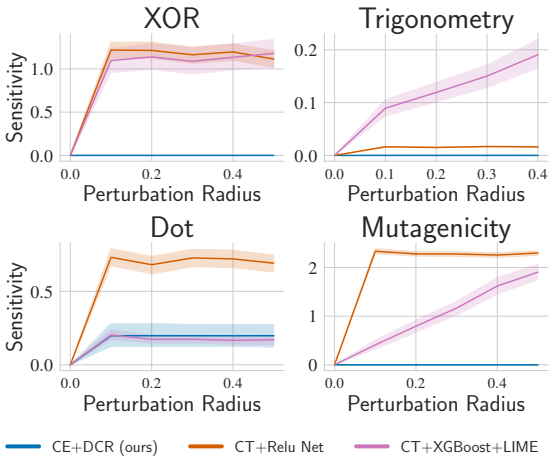

Figure 4. Sensitivity of model explanation when changing the radius of the input perturbation. The lower, the better. DCR explanations engender trust as they are stable under small perturbations of the input. The same does not hold generally for LIME explanations of XGBoost or Relu Net decision rules.

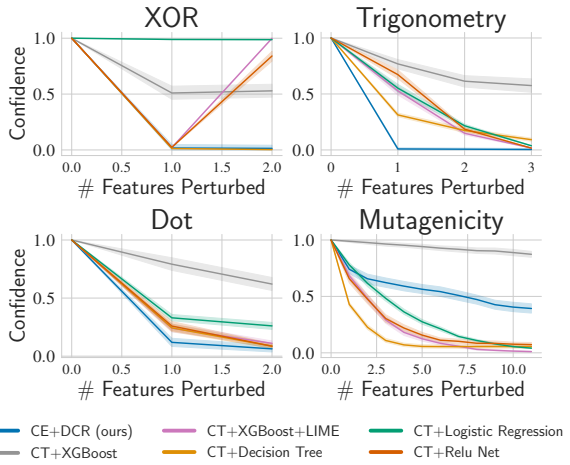

Figure 5. Model confidence as a function of the number of perturbed features on counterfactual examples. The lower, the better. Similarly to interpretable methods, DCR prediction confidence quickly drops after inverting the truth degree of a small set of relevant concepts, facilitating the discovery of counterfactual examples.

concepts were mispredicted, thus leading to incorrect rules.

Table 3. DCR explains prediction errors.

Dataset	Concepts	DCR rule	Ground truth label
XOR	[0.0, 0.0]	$y = 0 \leftarrow \neg c_0 \wedge \neg c_1$	$y = 1$
Trigonometry	[0.0, 1.0, 1.0]	$y = 1 \leftarrow \neg c_0 \wedge c_1 \wedge c_2$	$y = 0$
Trigonometry	[0.0, 1.0, 0.0]	$y = 0 \leftarrow \neg c_0 \wedge c_1 \wedge \neg c_2$	$y = 1$
Trigonometry	[0.0, 1.0, 1.0]	$y = 1 \leftarrow \neg c_0 \wedge c_1 \wedge c_2$	$y = 0$
Trigonometry	[0.0, 1.0, 1.0]	$y = 1 \leftarrow \neg c_0 \wedge c_1 \wedge c_2$	$y = 0$

DCR enables discovering counterfactual examples (Figure 5) Besides being stable, DCR rules can be used to find simple counterfactual examples, as introduced in Section 3.4. In Figure 5 we show a model's confidence in its predictions as we increase the number of concept perturbations. In making perturbations, we sort concepts from the most relevant to the least using DCR rules, as suggested by Wachter et al. (2017). Our results show that DCR confidence in its predictions drops quickly when we perturb the most relevant concepts according to a given rule. This enables us to discover counterfactual examples where the concept literals are very similar to the original one rule. This behaviour is emblematic of interpretable models such as decision trees and logistic regression, for which similar conclusions can be drawn. We also observe how in *Mutagenicity* DCR confidence is a bit higher than interpretable baselines. We can explain this behavior as for this challenging dataset DCR rules give equal relevance to a larger set of concepts. Still, DCR confidence is much lower than a black box such as XGBoost. Local explainers such as LIME can only partially explain the decision process of black box models such as XGBoost: LIME areas under the model confidence curve are generally higher than the other methods. We report the actual values for all methods in Table 5 together with further details and counterfactual examples.

5. Key findings & significance

Limitations One of the main limitations of DCR is that its global behavior may not be directly interpretable, which means that global rules may not perfectly align with the exact reasoning of the model. This could be an issue in cases where a user requires a precise understanding of the global model behavior. Also, the complexity of DCR rules may increase significantly when the difference between two tasks can only be determined by using a very high number of concepts. However, in most real-world cases, and in current benchmark datasets for concept-based models, this issue rarely arises. Finally, DCR requires concept embeddings as inputs, which assumes the existence of concept-based datasets or high-quality concept-discovery methods.

Relations with concept-based methods Interpretable concept-based models (Koh et al., 2020) address the lack of human trust in AI systems as they allow their users to understand their decision process (Rudin, 2019; Ghorbani et al., 2019a; Barbiero et al., 2023). These approaches come with several advantages over other explainability methods as they circumvent the brittleness of post-hoc methods (Adebayo et al., 2018; Kindermans et al., 2019) and provide a semantic advantage in settings where input features are naturally hard to reason about (e.g., raw image pixels) by providing explanations in terms of human-interpretable concepts (Ghorbani

et al., 2019a; Georgiev et al., 2022; Azzolin et al., 2022; Magister et al., 2022; Xuanyuan et al., 2022). However, Espinosa Zarlenga et al. (2022) and Mahinpei et al. (2021) emphasise how state-of-the-art concept-based models either struggle to efficiently solve real-world tasks using concept truth-values only or they weaken their interpretability using concept embeddings to increase their learning capacity. This is true even when concept-based models use a simple logistic regression or decision tree to map concept embeddings to tasks because concept embedding dimensions do not have a clear semantic meaning, and models composing such dimensions generate prediction rules that are not human-interpretable. Our work solves this issue by introducing the first interpretable concept-based model that learns logic rules from concept embeddings.

Relations with neural-symbolic methods A common paradigm in neural-symbolic is to exploit deep learning models to map subsymbolic information (e.g. images) to an intermediate logical representation, which is then manipulated using weighted logic formalisms, such as probabilistic logic (DeepProbLog (Manhaeve et al., 2018), NeurASP (Yang et al., 2020)), fuzzy logic (Lyrics(Marra et al., 2020c), LTN (Badreddine et al., 2022; Wagner & Garcez, 2021)) or both (DLM (Marra et al., 2020b), RNM(Marra et al., 2020a)). This sets DCR between concept-based and neural-symbolic models. However, while these neural symbolic models focus on how to maximally exploit available logic knowledge (e.g. a logic program) to improve neural predictions, DCR focuses on learning such logical knowledge. Other neural symbolic approaches, such as Neuro-Symbolic Concept Learner (Mao et al., 2019), the Neural Logic Machines (Dong et al., 2019), and the Neural State Machine (Hudson & Manning, 2019), are actually closer in spirit to concept based models as they exploit intermediate symbolic representations. However, the decision-making process on top of the concepts/symbols still relies on (or is uniquely) an uninterpretable neural component. In contrast, DCR encodes its decision process in a logical rule that is executed explicitly giving the user full knowledge and control over the concept-to-task decision-making process. This would be impossible in these neural symbolic approaches, as the decision process is implicit in the weights of the networks.

Key advantages of DCR The main advantage of DCR w.r.t. existing interpretable and black-box methods arises when dealing with challenging tasks where both interpretability and accuracy should be maximized. For simpler tasks, existing interpretable methods, such logistic regression, could be enough. On the other side, when interpretability is not a hard user requirement, then a simple black-box model would be easier to set up (e.g., it does not require concept labels or concept encoders). However, in all cases where interpretability plays a crucial role for the end user and existing interpretable models fail, then DCR could be preferable. Finally, compared to existing neural-symbolic approaches, DCR has an edge in all settings where the rules are unknown, while other methods (like DeepProbLog) might be more stable when the full set of rules is known in advance. For other limitations/drawbacks, please see our reply to common questions.

Conclusion This work presents the *Deep Concept Reasoner* (DCR), the new state-of-the-art of interpretable concept-based models. To achieve this, DCR builds for each sample a weighted logic rule combining neural and symbolic algorithms on concept embeddings in a unified end-to-end differentiable system. In our experiments, we compare DCR with state-of-the-art interpretable concept-based models and black-box models using datasets spanning three of the most common data types used in deep learning: tabular, image, and graph data. Our experiments show that Deep Concept Reasoners: (i) attain better task accuracy w.r.t. state-of-the-art interpretable concept-based models, (ii) discover meaningful logic rules, and (iii) facilitate the generation of counterfactual examples. While the global behaviour of the model is still not directly interpretable, our results show how aggregating Boolean DCR rules provides an approximation for the global behaviour of the model which matches known ground truth relationships. As a result, our experiments indicate that DCR represents a significant advance over the current state-of-the-art of interpretable concept-based models, and thus makes progress on a key research topic within the field of explainability.

Acknowledgements

The authors would like to thank Nikola Simidjievski for his insightful comments on earlier versions of this manuscript. PB acknowledges support from the European Union's Horizon 2020 research and innovation programme under grant agreement No 848077. GC and FP acknowledges support from the EU Horizon 2020 project AI4Media, under contract no. 951911 and by the French government, through Investments in the Future projects managed by the National Research Agency (ANR), 3IA Cote d'Azur with the reference number ANR-19-P3IA-0002. MEZ acknowledges support from the Gates Cambridge Trust via a Gates Cambridge Scholarship. FG was supported by TAILOR and by HumanE-AI-Net projects funded by EU Horizon 2020 research and innovation programme under GA No 952215 and No 952026, respectively.

References

Abid, A., Yuksekgonul, M., and Zou, J. Meaningfully explaining model mistakes using conceptual counterfactuals.

arXiv preprint arXiv:2106.12723, 2021.

ACGIH®. American conference of governmental industrial hygienists: Tlvs and beis based on the documentation of the threshold limit values for chemical substances and physical agents and biological exposure indices. American Conference of Governmental Industrial Hygienists Washington, DC, USA, 2016.

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

Aspis, Y., Broda, K., Lobo, J., and Russo, A. Embed2sym-scalable neuro-symbolic reasoning via clustered embeddings. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, pp. 421–431, 2022.

Azzolin, S., Longa, A., Barbiero, P., Liò, P., and Passerini, A. Global explainability of gnns via logic combination of learned concepts. *arXiv preprint arXiv:2210.07147*, 2022.

Badreddine, S., Garcez, A. d., Serafini, L., and Spranger, M. Logic tensor networks. *Artificial Intelligence*, 303: 103649, 2022.

Barbiero, P., Ciravegna, G., Giannini, F., Lió, P., Gori, M., and Melacci, S. Entropy-based logic explanations of neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6046–6054, 2022.

Barbiero, P., Fioravanti, S., Giannini, F., Tonda, A., Lio, P., and Di Lavore, E. Categorical foundations of explainable ai: A unifying formalism of structures and semantics. *arXiv preprint arXiv:2304.14094*, 2023.

Breiman, L. *Classification and regression trees*. Routledge, 2017.

Bussone, A., Stumpf, S., and O'Sullivan, D. The role of explanations on trust and reliance in clinical decision support systems. In *2015 international conference on healthcare informatics*, pp. 160–169. IEEE, 2015.

Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Chen, Z., Bei, Y., and Rudin, C. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.

Ciravegna, G., Barbiero, P., Giannini, F., Gori, M., Lió, P., Maggini, M., and Melacci, S. Logic explained networks. *Artificial Intelligence*, 314:103822, 2023.

Dong, H., Mao, J., Lin, T., Wang, C., Li, L., and Zhou, D. Neural logic machines. *arXiv preprint arXiv:1904.11694*, 2019.

Durán, J. M. and Jongsma, K. R. 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.

Espinosa Zarlenga, M., Barbiero, P., Ciravegna, G., Marra, G., Giannini, F., Diligenti, M., Shams, Z., Precioso, F., Melacci, S., Weller, A., et al. Concept embedding models. *Advances in Neural Information Processing Systems*, 35, 2022.

Espinosa Zarlenga, M., Barbiero, P., Shams, Z., Kazhdan, D., Bhatt, U., Weller, A., and Jamnik, M. Towards robust metrics for concept representation evaluation. *AAAI*, 2023.

EUGDPR. GDPR. General data protection regulation, 2017.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Forgy, E. W. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, 21: 768–769, 1965.

Georgiev, D., Barbiero, P., Kazhdan, D., Veličković, P., and Liò, P. Algorithmic concept-based explainable reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6685–6693, 2022.

Ghorbani, A., Abid, A., and Zou, J. Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 3681–3688, 2019a.

Ghorbani, A., Wexler, J., Zou, J., and Kim, B. Towards automatic concept-based explanations. *arXiv preprint arXiv:1902.03129*, 2019b.

Hájek, P. *Metamathematics of fuzzy logic*, volume 4. 2013.

Hand, D. J. and Till, R. J. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.

Hättenschwiler, S. and Vitousek, P. M. The role of polyphenols in terrestrial ecosystem nutrient cycling. *Trends in ecology & evolution*, 15(6):238–243, 2000.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hudson, D. and Manning, C. D. Learning by abstraction: The neural state machine. *Advances in Neural Information Processing Systems*, 32, 2019.

Kazhdan, D., Dimanov, B., Jamnik, M., Liò, P., and Weller, A. Now you see me (cme): concept-based model extraction. *arXiv preprint arXiv:2010.13233*, 2020.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 267–280. Springer, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. Concept bottleneck models. In *International Conference on Machine Learning*, pp. 5338–5348. PMLR, 2020.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

Lo Piano, S. 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.

Magister, L. C., Kazhdan, D., Singh, V., and Liò, P. Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks. *arXiv preprint arXiv:2107.11889*, 2021.

Magister, L. C., Barbiero, P., Kazhdan, D., Siciliano, F., Ciravegna, G., Silvestri, F., Jamnik, M., and Lio, P. Encoding concepts in graph neural networks. *arXiv preprint arXiv:2207.13586*, 2022.

Mahinpei, A., Clark, J., Lage, I., Doshi-Velez, F., and Pan, W. Promises and pitfalls of black-box concept learning models. *arXiv preprint arXiv:2106.13314*, 2021.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31, 2018.

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., and Wu, J. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

Marra, G., Diligenti, M., Giannini, F., Gori, M., and Maggini, M. Relational neural machines. *arXiv preprint arXiv:2002.02193*, 2020a.

Marra, G., Giannini, F., Diligenti, M., and Gori, M. Integrating learning and reasoning with deep logic models. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part II*, pp. 517–532. Springer, 2020b.

Marra, G., Giannini, F., Diligenti, M., and Gori, M. Lyrics: A general interface layer to integrate logic inference and deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 283–298. Springer, 2020c.

Miller, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

Minsky, M. and Papert, S. A. *Perceptrons: An introduction to computational geometry*. MIT press, 1969.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Ribeiro, M. T., Singh, S., and Guestrin, C. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

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

Sabry, N. M., Mohamed, H. M., Khattab, E. S. A., Motlaq, S. S., and El-Agrody, A. M. Synthesis of 4h-chromene, coumarin, 12h-chromeno [2, 3-d] pyrimidine derivatives and some of their antimicrobial and cytotoxicity activities. *European journal of medicinal chemistry*, 46(2):765–772, 2011.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

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

Verhulst, P. F. Resherches mathematiques sur la loi d'accroissement de la population. *Nouveaux memoires de l'academie royale des sciences*, 18:1–41, 1845.

Wachter, S., Mittelstadt, B., and Russell, C. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

Wagner, B. and Garcez, A. d. Neural-symbolic integration for interactive learning and conceptual grounding. *arXiv preprint arXiv:2112.11805*, 2021.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The caltech-ucsd birds-200-2011 dataset, 2011.

Winters, T., Marra, G., Manhaeve, R., and De Raedt, L. Deepstochlog: Neural stochastic logic programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 10090–10100, 2022.

Xuanyuan, H., Barbiero, P., Georgiev, D., Magister, L. C., and Lió, P. Global concept-based interpretability for graph neural networks via neuron analysis. *arXiv preprint arXiv:2208.10609*, 2022.

Yang, Z., Ishay, A., and Lee, J. Neurasp: Embracing neural networks into answer set programming. In *29th International Joint Conference on Artificial Intelligence (IJCAI 2020)*, 2020.

Yeh, C.-K., Hsieh, C.-Y., Suggala, A., Inouye, D. I., and Ravikumar, P. K. On the (in) fidelity and sensitivity of explanations. *Advances in Neural Information Processing Systems*, 32, 2019.

Yeh, C.-K., Kim, B., Arik, S., Li, C.-L., Pfister, T., and Ravikumar, P. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33:20554–20565, 2020.

Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

A. Datasets & Experimental Setup

XOR dataset The first dataset used in our experiments is inspired by the exclusive-OR (XOR) problem proposed by (Minsky & Papert, 1969) 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 $y = c_1 \oplus c_2$.

Trigonometric dataset The second dataset we use in our experiments is inspired by that proposed by Mahinpei et al. (2021) (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. (2021) by assigning each sample a label $y = \mathbb{1}_{(h_1+h_2)>0}$ indicating whether $h_1 + h_2$ is positive or not.

Vector dataset As much as the Trigonometric dataset is designed to highlight that fuzzy concept representations generalize better than Boolean concept representations, we designed the Vector dataset to show the advantage of embedding concept representations over fuzzy concept representations. The Vector 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. We then create four input features as the sum and difference of the two factors $\mathbf{v}_i$. 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). 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).

MNIST Addition In the MNIST addition dataset (Manhaeve et al., 2018), MNIST images are paired and the pair is labelled with the sum of the two corresponding digits. There are 30000 labelled pairs. The two images are given as two separate inputs to the model (i.e. they are not concateneted).

Mutagenicity The *Mutagenicity* dataset (Morris et al., 2020) is a labelled graph classification dataset, where a graph represents a molecule. The task is to predict whether the molecule is mutagenic or non-mutagenic. The dataset has 4337 graphs. We use the version available as part of the PyTorch Geometric (Fey & Lenssen, 2019) library.

CelebA We use the CelebA dataset to simulate a real-world condition where the set of training and test concepts is not the same, though the embeddings of training and test concepts are still correlated. To this end, we work using pre-trained embeddings generated by a Concept Embedding Model in the setting described by Espinosa Zarlenga et al. (2022). We then select the 3 most frequent concepts and train DCR and all the other baseline models on these concepts. However, at test time shift the set of concepts and we use the 3rd, 4th, and 5-th most frequent concept to make predictions. While all the first 5 concepts are highly correlated being attributes in human face images, the shift in distribution is quite significant. DCR can cope with this shift without any modification. However, usually AI models require a fixed number of features at training and test time. For this reason, we use zero-padding on training and test concepts to allow the other baselines to be trained and tested.

B. Training details

B.1. Deep Concept Reasoner

For all datasets we train DCR using a Godel t-norm semantics. We also implement the neural modules ϕ and ψ as with two-layer MLPs with a number of hidden layers given by the size of the concept embeddings.

For all synthetic datasets (i.e., *XOR*, *Trig*, *Dot*) and for CelebA we train DCR for 3000 epochs using a temperature of $\tau = 100$. In *Mutagenicity* we train DCR for 7000 epochs using a temperature of 100.

B.2. Concept Embedding Generators

To generate concept embeddings on synthetic datasets (i.e., *XOR*, *Trig*, *Dot*), we use a Concept Embedding Model (Espinosa Zarlenga et al., 2022) implemented as an MLP with hidden layer sizes $\{128, 128\}$ and LeakyReLU activations. When learning concept embedding representations in synthetic datasets, we learn embeddings with $m = 128$ activations.

In CelebA, we use a Concept Embedding Model on top of a pretrained ResNet-34 model (He et al., 2016) with its last layer modified to output $n_{\text{hidden}} = m$ activations. In this case, we learn embeddings with $m = 16$ activations, smaller than in the synthetic datasets given the larger number of concepts in these tasks.

In *Mutagenicity*, we use a Graph Convolutional Network (Scarselli et al., 2008; Morris et al., 2019) to map input graphs to the given task. We then extract concept embeddings using GCExplainer (Magister et al., 2021; **?**), a graph-based variant of the Automated Concept-based Explanation proposed by Ghorbani et al. (2019b) for image data. We implement the GNN with four layers of graph convolutions with 40 hidden neurons followed by leaky ReLU activation function each. We then apply mean pooling on node embeddings produced by the preceding graph convolutions and extract predictions via a linear readout function with 10 hidden units. We train these networks for 20 epochs with a learning rate of 0.001 and a batch size of 16 graphs, where we use an 80:20 split for the training and testing set. After training, we run GCExplainer on the node embeddings computed before pooling and extract 30 concepts using k-Means(Forgy, 1965), where each concept corresponds to a cluster of graph nodes in the embedding space. We encode these cluster labels as one-hot binary arrays and associate each node with the binary label of the closest cluster. We then obtain the concept truth values of a given graph by aggregating the binary labels of its nodes. To generate concept embeddings, we consider the node embeddings closest to the cluster centroids for active concepts.

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 (Kingma & Ba, 2014) optimizer with learning rate 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}.

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.

B.3. Hyperparameter search for benchmark classifiers

xWe run a grid search using an internal 3-fold cross-validation to find the optimal settings for benchmark classifiers. The parameter grid we use is:

- decision tree
  - max depth: [2, 4, 10, all leaves pure]
  - min_samples_split: [2, 4, 10]
  - min_samples_leaf: [1, 2, 5, 10]
- logistic regression
  - penalty: [l1, l2, elasticnet]
- XGBoost
  - booster: [tree, linear, dart]

C. Mutagenicity: Extracted Concepts

Here we report the visualization of the concepts extracted in *Mutagenicity* by GCExplainer. Following Magister et al. (2021) we represent the concept of a node by expanding and visualizing its p-hop neighborhood. In this experiment we set $p = 4$ as

we used four graph convolutional layers. Figures 6 - 8 show the 30 concepts extracted using GCExplainer when $k = 30$ in k-Means, where the red nodes are the nodes clustered together for a given concept. A human can identify the concept present by reasoning about which features and structures are repeated across the five sample subgraphs, representative of a concept. Using this approach, a number of concepts can be clearly identified. For example, concept 0 (Figure 6, highlights the importance of the Carbon atom for the prediction that the molecule is mutagenic. In contrast, concepts 8 (Figure 6) and 28 (Figure 8) highlight the importance of the star structure in both the prediction of the molecule being mutagenic and non-mutagenic. Concept 11 clearly identified a complex structure of carbon, nitrogen and hydrogen atoms for predicting the label 'mutagenic'. For a complete overview, we visualise the full molecule of the medoids of each cluster in Figures 9 and 10, highlighting in red the node corresponding to the closest concept. This highlights the size and variety of the molecules classified as different concepts.

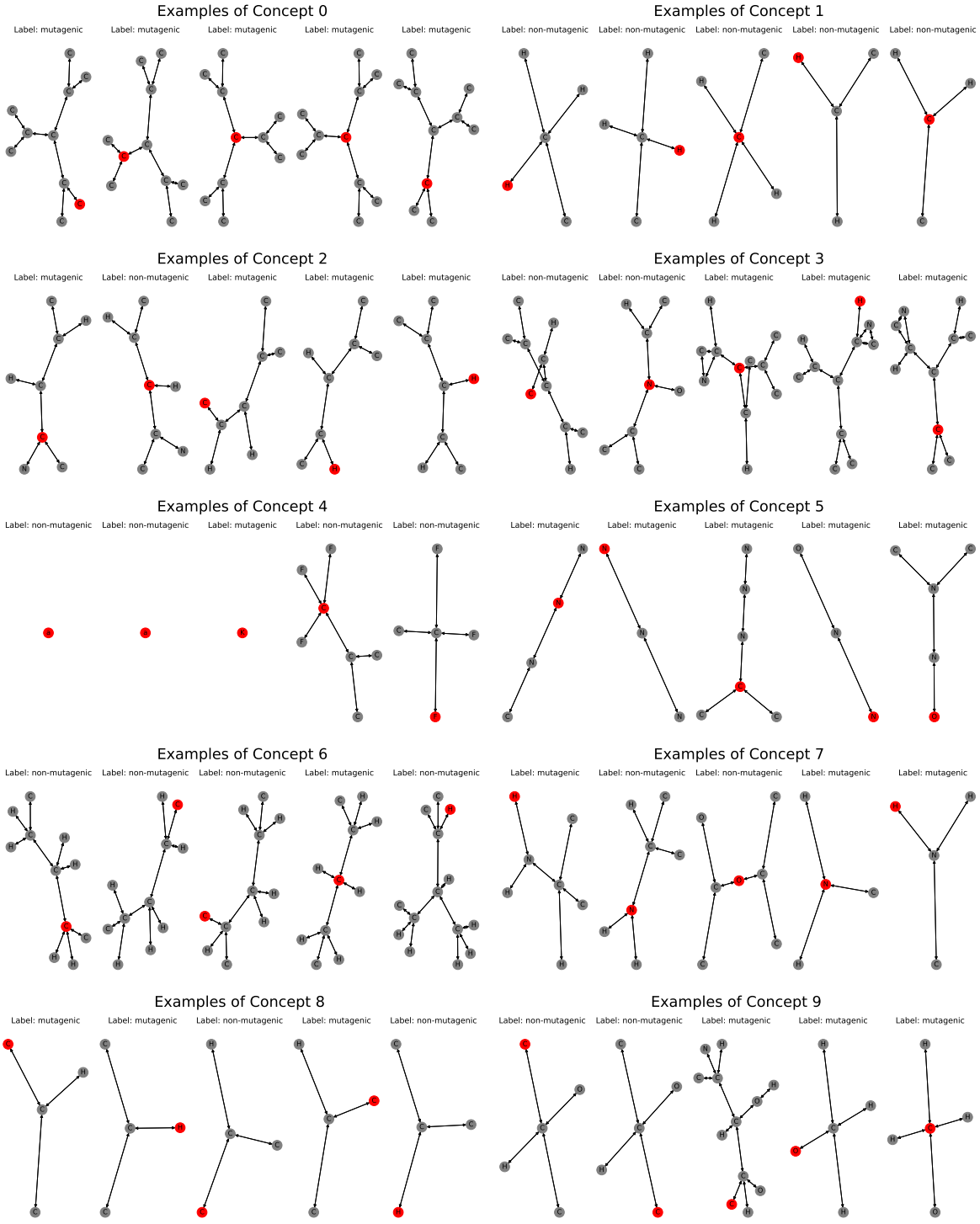

Figure 6. Concept discovered by the graph concept explainer. Part I.

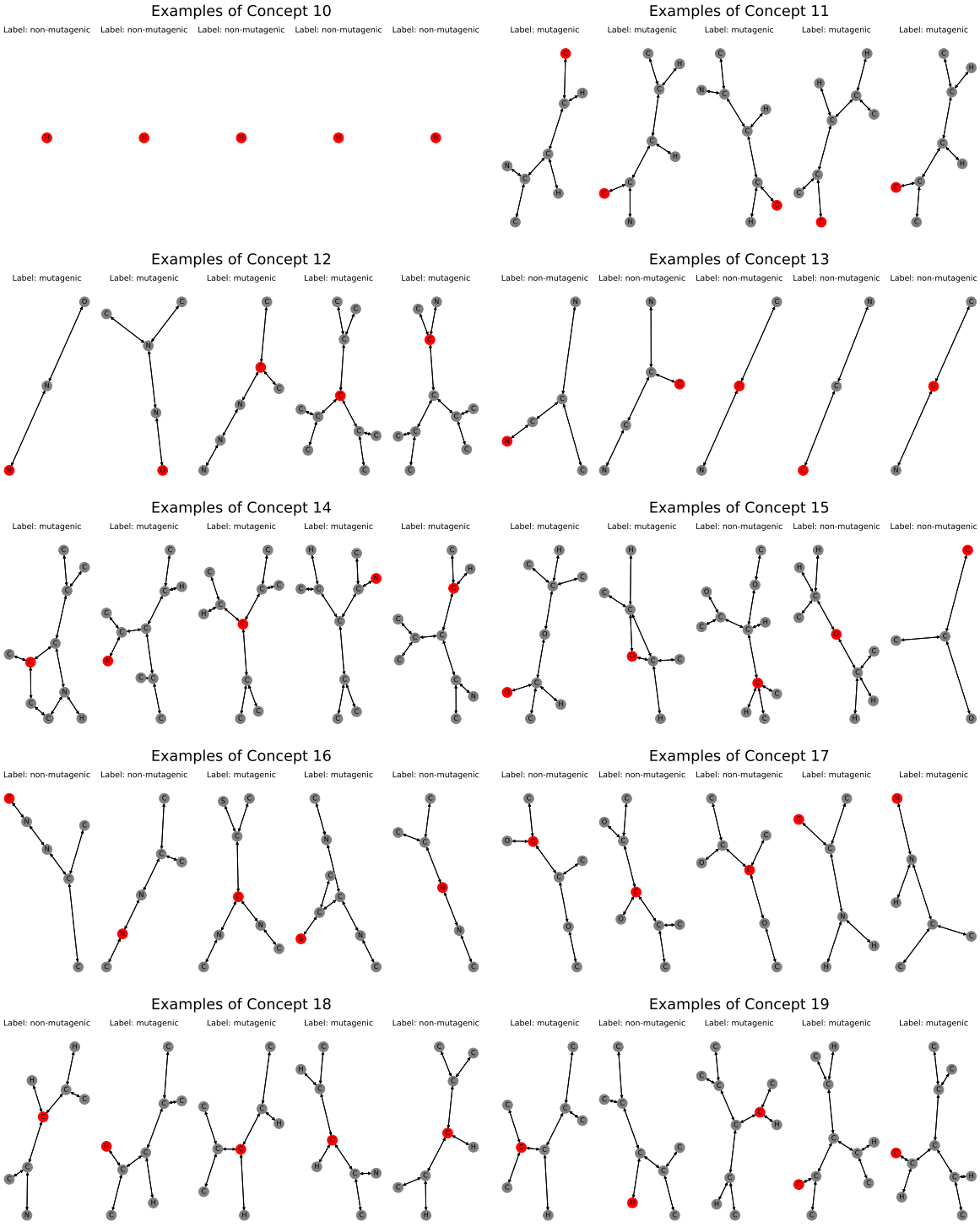

Figure 7. Concept discovered by the graph concept explainer. Part II.

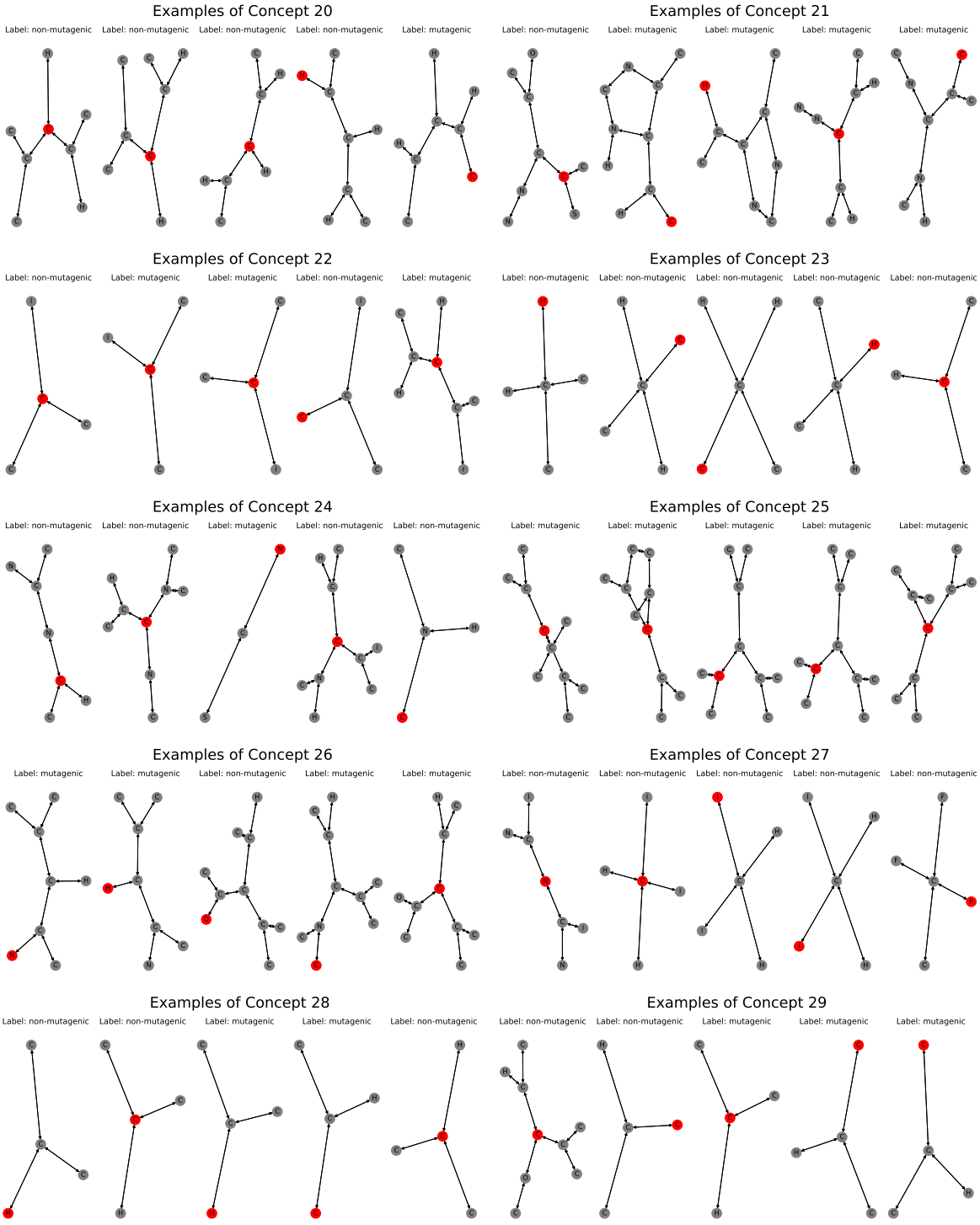

Figure 8. Concept discovered by the graph concept explainer. Part III.

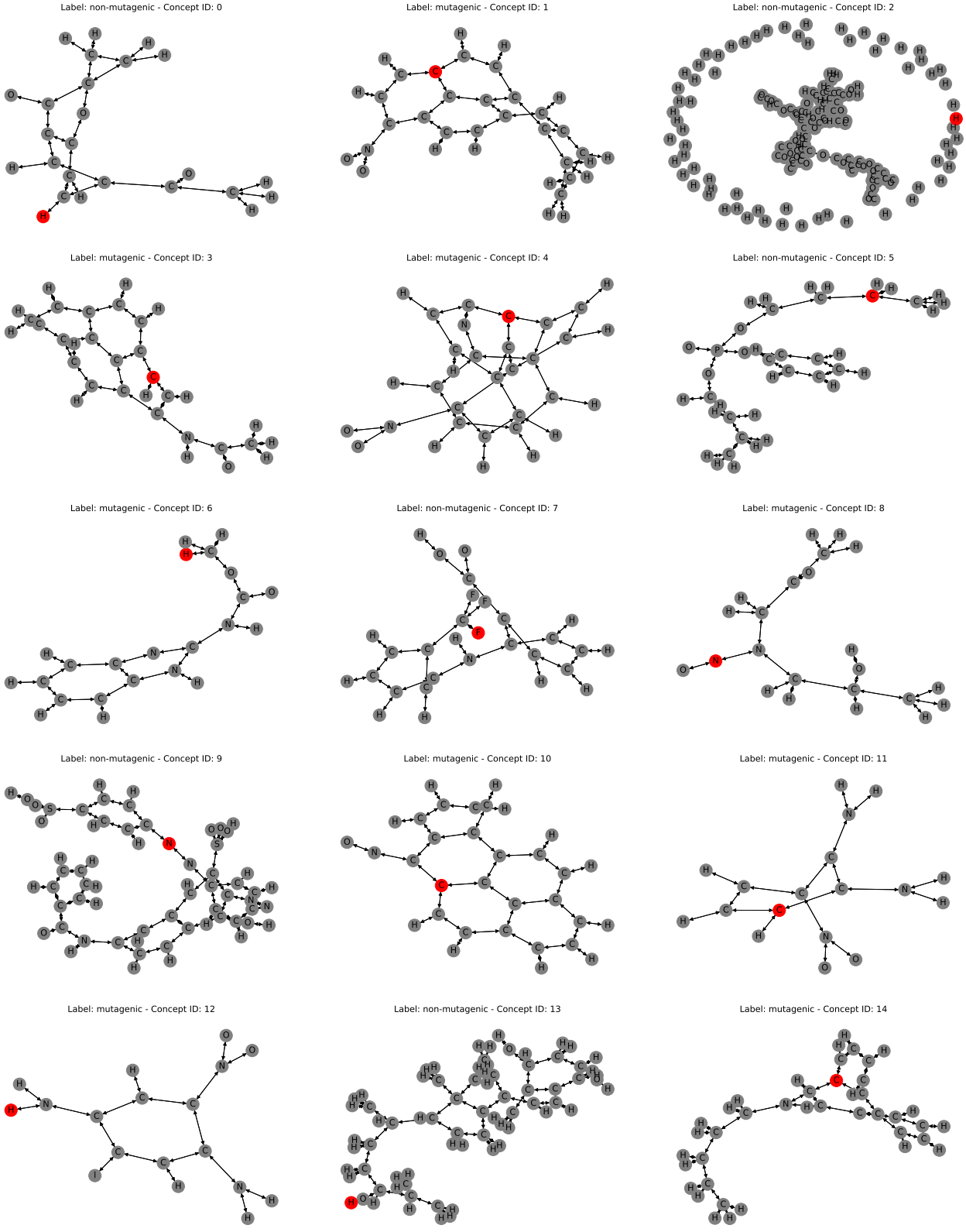

Figure 9. Full molecule corresponding to the closest node embedding to the concept centroid. Part I.

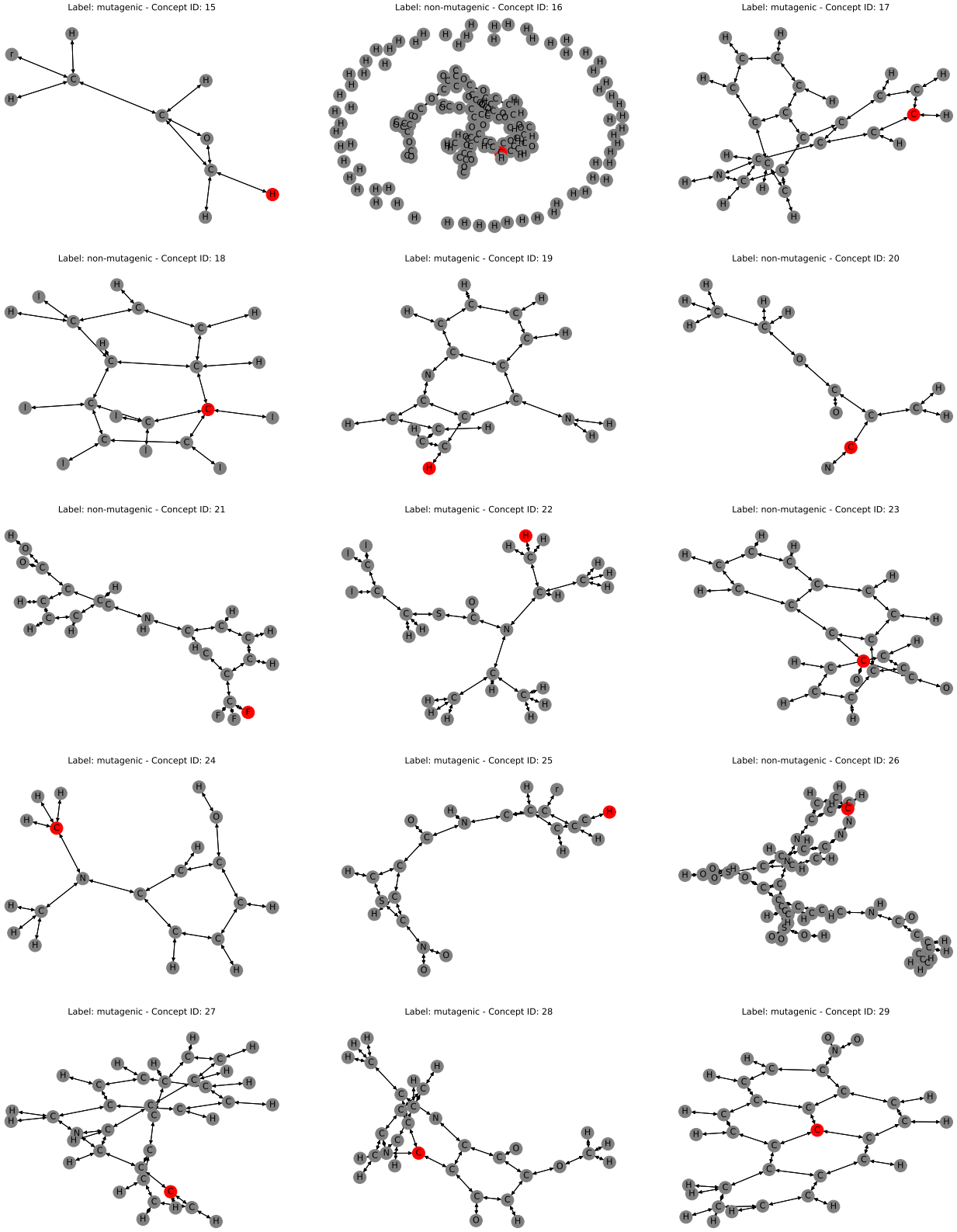

Figure 10. Full molecule corresponding to the closest node embedding to the concept centroid. Part II.

D. Softmax temperature effect on relevant concepts

We perform an ablation study on the temperature hyperparameter of DCR. This hyperparameter controls the number of concepts selected by DCR to generate rules in the activation function of Equation 6. A low temperature biases DCR towards simpler rules composing fewer concepts, while a high temperature biases DCR towards more complex rules composing many concepts. To assess this we train DCR on the embeddings of a pre-trained Concept Embedding Model on the Caltech-UCSD Birds-200-2011 dataset (Wah et al., 2011) as it contains a large number of concepts. We test 7 temperature ranges $\tau \in [0.1, 10]$ and we train DCR using 5 different initialization seeds.

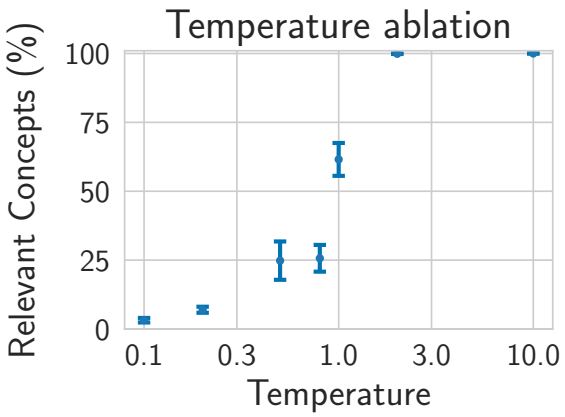

Figure 11. Temperature ablation on pre-trained concept embeddings from the CUB dataset.

E. Number of concepts effect on training and test time

We evaluate the computational cost of DCR as a function of the number of training concepts. To this end, we train DCR on the embeddings of a pre-trained Concept Embedding Model on the Caltech-UCSD Birds-200-2011 dataset (Wah et al., 2011) as it contains a large number of concepts. We then randomly select 10, 50, 100, and 150 concepts to train DCR. We train DCR using 5 different initialization seeds. We observe that the computational time increases linearly when the number of concepts is small, and then it becomes almost constant.

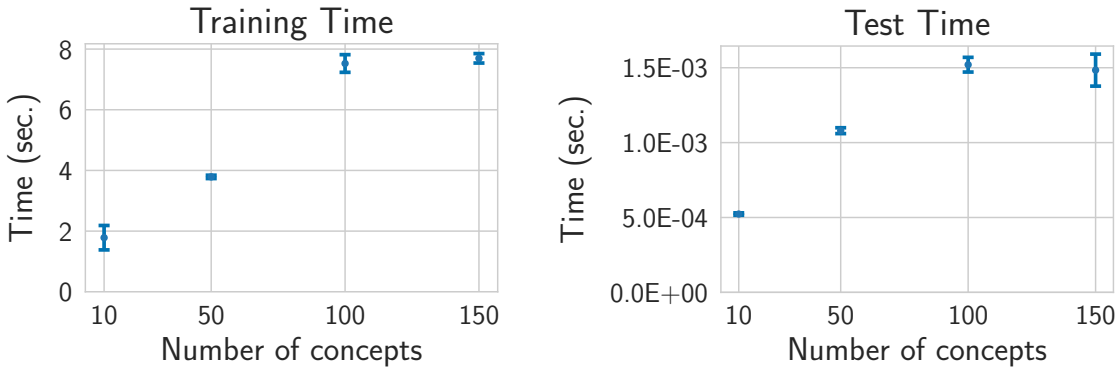

Figure 12. DCR computational time on pre-trained concept embeddings from the CUB dataset.

F. Sensitivity analysis

In Table 4, we report the results of the sensitivity analysis comparing DCR with interpretable models and local post-hoc explainers. More precisely, we report the area under the sensitivity curves of Figure 4, when increasing the perturbation radius. The lower the values, the more stable the local explanations are on similar samples. These samples $x^\star$ correspond to

randomly perturbed sample x drawn by from the test set. More precisely, we draw them from a Gaussian distribution with maximum radius ϵ. These perturbations must be non-significant, i.e., the model prediction must not change. We can see how DCR sensitivity is typically close to existing interpretable models. On the contrary, the compared explanation-based methods Lime and ReluNet have higher sensitivity, strongly reducing the user trust in these explanation methods. Indeed, since samples are very similar and the model predictions do not change, a user expects that also the corresponding explanation should not change, which does not happen for these methods.

Table 4. AUC of the explanation sensitivity curves when increasing the perturbation radius ϵ. The lower, the better.

Model	XOR	Trig	Vec	Mutag
DT	**0.000**$_{\pm 0.000}$	**0.000**$_{\pm 0.000}$	**0.000**$_{\pm 0.000}$	**0.000**$_{\pm 0.000}$
LR	**0.000**$_{\pm 0.000}$	**0.000**$_{\pm 0.000}$	**0.000**$_{\pm 0.000}$	**0.000**$_{\pm 0.000}$
ReluNet	0.939$_{\pm 1.301}$	0.110$_{\pm 0.181}$	0.148$_{\pm 0.247}$	0.995$_{\pm 1.480}$
LIME	0.984$_{\pm 0.885}$	0.013$_{\pm 0.009}$	0.592$_{\pm 0.534}$	1.900$_{\pm 0.969}$
DCR	**0.000**$_{\pm 0.000}$	**0.000**$_{\pm 0.000}$	0.165$_{\pm 0.614}$	**0.000**$_{\pm 0.000}$

G. Counterfactual explanations

In table 5 we report the Areas under the model confidence curves of Figure 5 when increasing the number of perturbed features. The lower the values, the easier it is to find a counterfactual sample. By comparing DCR with existing interpretable models and local post-hoc explainers, we can see how DCR provides the lowest values in three datasets out of four, confirming that the provided explanations are very precise as they correctly indicate the most important features for a given prediction.

In Table 6, instead, we report some examples of counterfactual rules provided by DCR on some benchmarked datasets.

Table 5. Area under the model confidence curves reported in Figure 5 against counterfactual samples when increasing the number of perturbed features. The lower, the better. Our model reports the lowest value on three datasets out of four, confirming that DCR explanations can be effectively used to find counterfactual examples.

Model	XOR	Trig	Vec	Mutag
DT	**0.339**$_{\pm 0.468}$	**0.395**$_{\pm 0.380}$	**0.443**$_{\pm 0.442}$	**0.185**$_{\pm 0.311}$
LR	0.992$_{\pm 0.015}$	0.451$_{\pm 0.402}$	0.530$_{\pm 0.391}$	0.347$_{\pm 0.351}$
ReluNet	0.622$_{\pm 0.476}$	0.469$_{\pm 0.429}$	0.448$_{\pm 0.457}$	**0.279**$_{\pm 0.387}$
LIME	0.674$_{\pm 0.462}$	0.424$_{\pm 0.422}$	0.450$_{\pm 0.438}$	0.249$_{\pm 0.372}$
XGBoost	0.680$_{\pm 0.460}$	0.739$_{\pm 0.431}$	0.804$_{\pm 0.426}$	0.924$_{\pm 0.226}$
DCR	**0.344**$_{\pm 0.505}$	**0.255**$_{\pm 0.436}$	**0.394**$_{\pm 0.489}$	0.705$_{\pm 0.467}$

Table 6. Counterfactuals (Boolean)

Dataset	**Old Concepts**	**Old prediction**	**New Concepts**	**New Prediction**
XOR	$\neg f0$, $\neg f1$	$y = 0$	$\neg f0$, $f1$	$y = 1$
XOR	$f0$, $\neg f1$	$y = 1$	$\neg f0$, $\neg f1$	$y = 0$
Trigonometry	$\neg f0$, $\neg f1$, $\neg f2$	$y = 0$	$\neg f0$, $f1$, $f2$	$y = 1$
Trigonometry	$f0$, $f1$, $\neg f2$	$y = 1$	$\neg f0$, $\neg f1$, $\neg f2$	$y = 0$
Vector	$f0$, $\neg f1$	$y = 0$	$\neg f0$, $f1$	$y = 1$
Vector	$\neg f0$, $\neg f1$	$y = 1$	$f0$, $f1$	$y = 0$
CelebA	$\neg f0$, $\neg f1$, $\neg f2$	$y = 0$	$f0$, $f1$, $f2$	$y = 4$

H. MNIST addition experiment

In this experiment, we tested DCR in a task where it is not provided with any label on the concepts. In the MNIST addition dataset (Manhaeve et al., 2018), pairs of MNIST images are labelled with the sum of the corresponding digit. The single images are, therefore, never labelled. The idea behind the task is that an image classifier can still be asked to predict the class of the single images, while a differentiable symbolic program can be used to map the class of the images to their sum. In terms of learning, the knowledge of both the label on the addition and the symbolic program provides a distant supervision signal to the image classifier.

This task can be easily mapped in terms of a concept-based model. The output of the classifier for the two images constitutes a set of 20 concepts (i.e. 10 class predictions for each of the two images). The set of all possible additions constitutes a set of 19 tasks. The MNIST addition task could be considered a first example of a more structured (i.e. relational) setting, where the input is a list of two images. However, it is still simple enough not to require any specific modelling.

The absence of direct supervision on the concepts puts our system in a different regime. In fact, there is no loss that forces the concept probabilities to represent crisp decisions. The softmax activation function tends to crisp decisions only when coupled with a categorical cross-entropy loss. In the absence of such loss, the network can still exploit the entire categorical distribution as an embedding to latently encode the identity of the digits.

Our solution to the absence of a concept loss is made of two ingredients. First, the softmax output distribution is substituted with a Gumbel-softmax sampling layer. The Gumbel-softmax forces the network to always make crisp decisions by sampling from the corresponding categorical distribution. Notice that a categorical distribution and its one-hot samples coincide when the distribution becomes very peaked on its prediction (e.g. at the end of the learning). Second, we introduce a second task predictor function $f_{NN} : C \rightarrow Y$, that akin to standard concept bottleneck models, predicts the task only from the probabilities, and we add a corresponding loss encouraging $f_{NN}(g(\mathbf{x})) = \mathbf{y}$. The goal here is to force the model to exploit (and thus learn) the concept probabilities $\hat{c}_i$ and not to rely only on their embeddings $\hat{\mathbf{c}}_i$.

In Table **??**, we show the comparison with state-of-the-art Neural Symbolic frameworks, as described in the main text. Moreover, in Table 7, we show the entire list of global rules learned by DCR, showing that it actually captured perfectly the semantics of the addition relation.

Table 7. MNIST addition global rules for 10000 training examples. f_{ij} reads "class of the digit in position i is j. Therefore, the rule $y_0 \leftarrow f_{00} \wedge f_{10}$ means that if the first digit is a 0 and the second digit is a 0 then the sum is a 0. The semantics is correct except for a single rule $y_8 \leftarrow f_{03} \wedge f_{16}$, which is easily identifiable as having a count of 1. Notice that we had to map the network concept IDs to the corresponding human digits, as there was no supervision on concepts during training.

RULE	COUNT	RULE	COUNT
$y_0 \leftarrow f_{00} \wedge f_{10}$	93	$y_9 \leftarrow f_{03} \wedge f_{16}$	89
$y_1 \leftarrow f_{00} \wedge f_{11}$	110	$y_9 \leftarrow f_{09} \wedge f_{10}$	100
$y_1 \leftarrow f_{01} \wedge f_{10}$	102	$y_9 \leftarrow f_{07} \wedge f_{12}$	110
$y_2 \leftarrow f_{00} \wedge f_{12}$	89	$y_9 \leftarrow f_{02} \wedge f_{17}$	102
$y_2 \leftarrow f_{01} \wedge f_{11}$	119	$y_9 \leftarrow f_{05} \wedge f_{14}$	89
$y_2 \leftarrow f_{02} \wedge f_{10}$	101	$y_{10} \leftarrow f_{01} \wedge f_{19}$	115
$y_3 \leftarrow f_{01} \wedge f_{12}$	124	$y_{10} \leftarrow f_{06} \wedge f_{14}$	97
$y_3 \leftarrow f_{03} \wedge f_{10}$	96	$y_{10} \leftarrow f_{09} \wedge f_{11}$	100
$y_3 \leftarrow f_{02} \wedge f_{11}$	115	$y_{10} \leftarrow f_{08} \wedge f_{12}$	100
$y_3 \leftarrow f_{00} \wedge f_{13}$	100	$y_{10} \leftarrow f_{07} \wedge f_{13}$	113
$y_4 \leftarrow f_{03} \wedge f_{11}$	121	$y_{10} \leftarrow f_{04} \wedge f_{16}$	94
$y_4 \leftarrow f_{04} \wedge f_{10}$	84	$y_{10} \leftarrow f_{03} \wedge f_{17}$	89
$y_4 \leftarrow f_{01} \wedge f_{13}$	137	$y_{10} \leftarrow f_{02} \wedge f_{18}$	103
$y_4 \leftarrow f_{02} \wedge f_{12}$	105	$y_{10} \leftarrow f_{05} \wedge f_{15}$	75
$y_4 \leftarrow f_{00} \wedge f_{14}$	112	$y_{11} \leftarrow f_{08} \wedge f_{13}$	89
$y_5 \leftarrow f_{01} \wedge f_{14}$	104	$y_{11} \leftarrow f_{03} \wedge f_{18}$	105
$y_5 \leftarrow f_{03} \wedge f_{12}$	105	$y_{11} \leftarrow f_{07} \wedge f_{14}$	94
$y_5 \leftarrow f_{04} \wedge f_{11}$	113	$y_{11} \leftarrow f_{09} \wedge f_{12}$	97
$y_5 \leftarrow f_{00} \wedge f_{15}$	95	$y_{11} \leftarrow f_{04} \wedge f_{17}$	111
$y_5 \leftarrow f_{02} \wedge f_{13}$	90	$y_{11} \leftarrow f_{05} \wedge f_{16}$	86
$y_5 \leftarrow f_{05} \wedge f_{10}$	95	$y_{11} \leftarrow f_{02} \wedge f_{19}$	105
$y_6 \leftarrow f_{02} \wedge f_{14}$	92	$y_{11} \leftarrow f_{06} \wedge f_{15}$	104
$y_6 \leftarrow f_{05} \wedge f_{11}$	96	$y_{12} \leftarrow f_{03} \wedge f_{19}$	98
$y_6 \leftarrow f_{00} \wedge f_{16}$	109	$y_{12} \leftarrow f_{04} \wedge f_{18}$	87
$y_6 \leftarrow f_{04} \wedge f_{12}$	91	$y_{12} \leftarrow f_{06} \wedge f_{16}$	105
$y_6 \leftarrow f_{01} \wedge f_{15}$	86	$y_{12} \leftarrow f_{07} \wedge f_{15}$	96
$y_6 \leftarrow f_{03} \wedge f_{13}$	107	$y_{12} \leftarrow f_{09} \wedge f_{13}$	106
$y_6 \leftarrow f_{06} \wedge f_{10}$	92	$y_{12} \leftarrow f_{05} \wedge f_{17}$	94
$y_7 \leftarrow f_{00} \wedge f_{17}$	100	$y_{12} \leftarrow f_{08} \wedge f_{14}$	87
$y_7 \leftarrow f_{04} \wedge f_{13}$	108	$y_{13} \leftarrow f_{06} \wedge f_{17}$	106
$y_7 \leftarrow f_{01} \wedge f_{16}$	103	$y_{13} \leftarrow f_{08} \wedge f_{15}$	85
$y_7 \leftarrow f_{02} \wedge f_{15}$	81	$y_{13} \leftarrow f_{09} \wedge f_{14}$	82
$y_7 \leftarrow f_{07} \wedge f_{10}$	103	$y_{13} \leftarrow f_{07} \wedge f_{16}$	118
$y_7 \leftarrow f_{06} \wedge f_{11}$	137	$y_{13} \leftarrow f_{05} \wedge f_{18}$	79
$y_7 \leftarrow f_{05} \wedge f_{12}$	87	$y_{13} \leftarrow f_{04} \wedge f_{19}$	100
$y_7 \leftarrow f_{03} \wedge f_{14}$	117	$y_{14} \leftarrow f_{06} \wedge f_{18}$	105
$y_8 \leftarrow f_{05} \wedge f_{13}$	72	$y_{14} \leftarrow f_{07} \wedge f_{17}$	98
$y_8 \leftarrow f_{01} \wedge f_{17}$	122	$y_{14} \leftarrow f_{05} \wedge f_{19}$	78
$y_8 \leftarrow f_{03} \wedge f_{15}$	99	$y_{14} \leftarrow f_{09} \wedge f_{15}$	74
$y_8 \leftarrow f_{02} \wedge f_{16}$	97	$y_{14} \leftarrow f_{08} \wedge f_{16}$	101
$y_8 \leftarrow f_{06} \wedge f_{12}$	90	$y_{15} \leftarrow f_{09} \wedge f_{16}$	107
$y_8 \leftarrow f_{08} \wedge f_{10}$	96	$y_{15} \leftarrow f_{08} \wedge f_{17}$	95
$y_8 \leftarrow f_{07} \wedge f_{11}$	116	$y_{15} \leftarrow f_{07} \wedge f_{18}$	103
$y_8 \leftarrow f_{04} \wedge f_{14}$	106	$y_{15} \leftarrow f_{06} \wedge f_{19}$	111
$y_8 \leftarrow f_{00} \wedge f_{18}$	100	$y_{16} \leftarrow f_{07} \wedge f_{19}$	115
$y_8 \leftarrow f_{03} \wedge f_{16}$	1	$y_{16} \leftarrow f_{09} \wedge f_{17}$	100
$y_9 \leftarrow f_{04} \wedge f_{15}$	87	$y_{16} \leftarrow f_{08} \wedge f_{18}$	84
$y_9 \leftarrow f_{08} \wedge f_{11}$	112	$y_{17} \leftarrow f_{09} \wedge f_{18}$	100
$y_9 \leftarrow f_{06} \wedge f_{13}$	76	$y_{17} \leftarrow f_{08} \wedge f_{19}$	86
$y_9 \leftarrow f_{01} \wedge f_{18}$	113	$y_{18} \leftarrow f_{09} \wedge f_{19}$	102
$y_9 \leftarrow f_{00} \wedge f_{19}$	94		

Our solution to the MNIST addition task shows that DCR can be enhanced with an unsupervised (or distantly supervised) criterion for the learning of meaningful concepts. This creates interesting links with generative models for learning representations, but we leave such interpretation for future works.

The architecture of the image classifiers is those in (Manhaeve et al., 2018). The additional task network is MLP with 1 hidden layer of 30 hidden neurons and relu activations. We searched over the following grid of parameters (bold selected): embedding size [10, 20, **30**, 50]; gumbel-softmax temperature [1, 1.25, 1.50, **1.75**, 2.0].

I. Complexity of logic rules

We compute rule complexity as the average size of the learnt logic rules. Table 8 summarizes the main outcomes comparing DCR rules with decision tree rules. In most datasets, such as Trigonometry, Dot, or CelebA, the rule complexity of DCR matches that of decision tree rules while providing superior task performance. However, in Mutagenicity, there is a tradeoff between performance and complexity compared to decision trees. Nevertheless, we don't observe a significant increase in rule complexity as shown in the plot, partly because DCR rules are "per sample." However, if we were to learn global rules, the complexity would likely increase, especially if multiple combinations of concepts could result in the same task prediction. It is worth noting that overly complex rules may not be a machine error, but rather a limitation of the human side. For example, asking a model to explain complex tasks using raw features like pixel intensities as concepts would lead to complex rules.

Table 8. Complexity of logic rules

	CE+DCR (ours)	**CT+Decision Tree**	**CE+Decision Tree**
XOR	2.00 ± 0.00	2.00 ± 0.00	1.40 ± 0.16
Trigonometry	3.00 ± 0.00	3.00 ± 0.00	1.40 ± 0.16
Dot	2.00 ± 0.00	2.00 ± 0.00	1.93 ± 0.07
Mutagenicity	13.57 ± 0.62	4.84 ± 0.74	2.35 ± 0.35
CelebA	1.00 ± 0.00	1.00 ± 0.00	5.86 ± 0.56

J. Code, Licences, 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 (Paszke et al., 2019) (BSD license) and Scikit-learn (Pedregosa et al., 2011) (BSD license). To produce the plots seen in this paper, we made use of Matplotlib 3.5 (BSD license). We will release all of the code required to recreate our experiments in an MIT-licensed public repository.

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.