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 improves up to $+25\%$ w.r.t. state-of-the-art interpretable concept-based models on challenging benchmarks, and discovers meaningful logic rules matching known ground truths even in the absence of concept supervision during training.

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). 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). 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* (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.

2. Preliminaries

Concept-based models Concept-based models $f : C \to Y$ 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 $g : X \to C$ 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

[*]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>.

Published at the Differentiable Almost Everything Workshop of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. July 2023. Copyright 2023 by the author(s).

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 $\hat{c}_i \in [0,1]$.[1] 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; van Krieken et al., 2022; Petersen et al., 2022) 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* $\perp$ (`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 1), the first interpretable concept-based model based on concept embeddings. In DCR, high-dimensional concept 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.

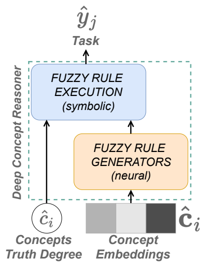

Figure 1. Deep Concept Reasoner (DCR) generates fuzzy logic rules using neural models on concept embeddings. Then DCR executes the rule using the concept truth degrees to evaluate the rule symbolically.

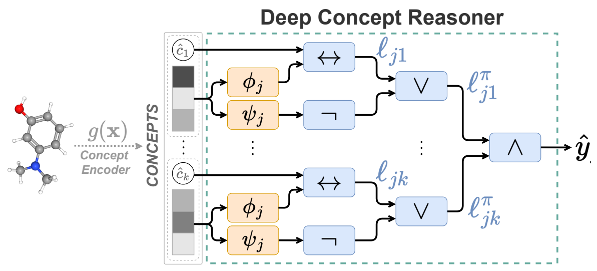

Figure 2. Schema of DCR modules: first neural models ϕ and ψ generate the rule, and then the rule is executed symbolically.

3.1. Rule syntax

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

Example: 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 $\hat{y}_j \Leftrightarrow \bigwedge_{i:\, r_{ji}=1} l_{ji}$. 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 $\neg\hat{c}_i$).

[1]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.

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. 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 $\phi_j : \mathbb{R}^m \rightarrow [0,1]$, 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.:

$$\ell_{ji} = (\phi_j(\hat{\mathbf{c}}_i) \Leftrightarrow \hat{c}_i) \tag{1}$$

Example: 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 $\psi_j : \mathbb{R}^m \rightarrow [0,1]$. 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:

$$\ell^r_{ji} = (\psi_j(\hat{\mathbf{c}}_i) \Rightarrow \ell_{ji}) = (\neg\psi_j(\hat{\mathbf{c}}_i) \vee \ell_{ji}) \tag{2}$$

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

Example: 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: $\hat{y}_j = \bigwedge_{i=1}^k \ell^r_{ji}$.

Example: 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. 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. (2020) 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 1 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 $\hat{y}_j = \min_{i=1}^k\{\max\{1 - \psi_j(\hat{\mathbf{c}}_i), \ell_{ji}\}\}$.

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:

$$\hat{y}_j^C = \bigvee_{\mathbf{x} \in \mathcal{X}_{\text{train}}} \hat{y}_j(\mathbf{x}) \tag{3}$$

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

4. Experiments

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.

DCR matches the accuracy of neural-symbolic systems trained using human rules (Table 2) 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 F). 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.

Table 1. 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.

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

DCR discovers semantically meaningful logic rules (Table 1) 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 of Booleanized DCR rules with the corresponding ground truth rules in Table 1. 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 3 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 F. 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, not only there is no supervision on the concepts, but we don't even know which are the concepts. Many of DCR's rules predicting mutagenic effects include functional groups such as phenols (Hättenschwiler and Vitousek, 2000) and dimethylamines (ACGIH®, 2016), which can be toxic.

5. Conclusion

This work presents the *Deep Concept Reasoner* (DCR), the new state-of-the-art of interpretable concept-based models. 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.

References

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.

Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Pietro Lió, Marco Gori, and Stefano Melacci. Entropy-based logic explanations of neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6046–6054, 2022.

Adrian Bussone, Simone Stumpf, and Dympna O'Sullivan. The role of explanations on trust and reliance in clinical decision support systems. In *2015 international conference on healthcare informatics*, pages 160–169. IEEE, 2015.

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

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, et al. Concept embedding models. *Advances in Neural Information Processing Systems*, 35, 2022.

Mateo Espinosa Zarlenga, Pietro Barbiero, Zohreh Shams, Dmitry Kazhdan, Umang Bhatt, Adrian Weller, and Mateja Jamnik. Towards robust metrics for concept representation evaluation. *AAAI*, 2023.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

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

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, 2019a.

Amirata Ghorbani, James Wexler, James Zou, and Been Kim. Towards automatic concept-based explanations. *arXiv preprint arXiv:1902.03129*, 2019b.

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

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

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.

Dmitry Kazhdan, Botty Dimanov, Mateja Jamnik, Pietro Liò, and Adrian Weller. Now you see me (cme): concept-based model extraction. *arXiv preprint arXiv:2010.13233*, 2020.

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.

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

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.

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.

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.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31, 2018.

Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. Lyrics: A general interface layer to integrate logic inference and deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 283–298. Springer, 2020.

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

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

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.

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.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Deep differentiable logic gate networks. *arXiv preprint arXiv:2210.08277*, 2022.

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

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

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.

Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing differentiable fuzzy logic operators. *Artificial Intelligence*, 302:103602, 2022.

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

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.

A. Datasets & Experimental Setup

XOR dataset The first dataset used in our experiments is inspired by the exclusive-OR (XOR) problem proposed by (Minsky and 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 $c_i = \nVdash_{x_i > 0.5}$. 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 $y = \nVdash_{(h_1 + h_2) > 0}$ 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 and 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 and 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. Results Details

In this section we show the additional experimental results about the generalization capabilities of DCR in different datasets 3, and a comparison with neural-symbolic models on the MNIST-addition dataset 2.

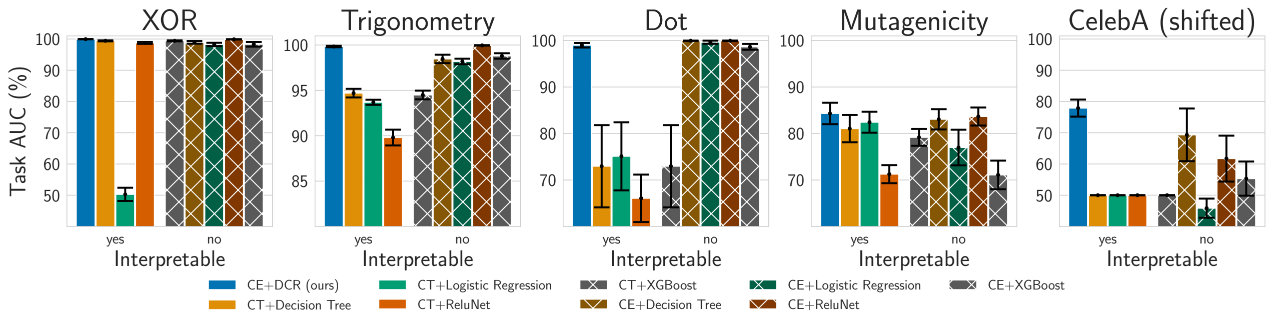

Figure 3. Mean ROC AUC for task predictions for all baselines across all tasks (the higher the better). DCR often outperforms interpretable concept-based models. *CE* stands for concept embeddings, while *CT* for concept truth degrees. Models trained on concept embeddings are not interpretable as concept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

Table 2. 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

D. 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 4 - 6 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 4, highlights the importance of the Carbon atom for the prediction that the molecule is mutagenic. In contrast, concepts 8 (Figure 4) and 28 (Figure 6) 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 7 and 8, highlighting in red the node corresponding to the closest concept. This highlights the size and variety of the molecules classified as different concepts.

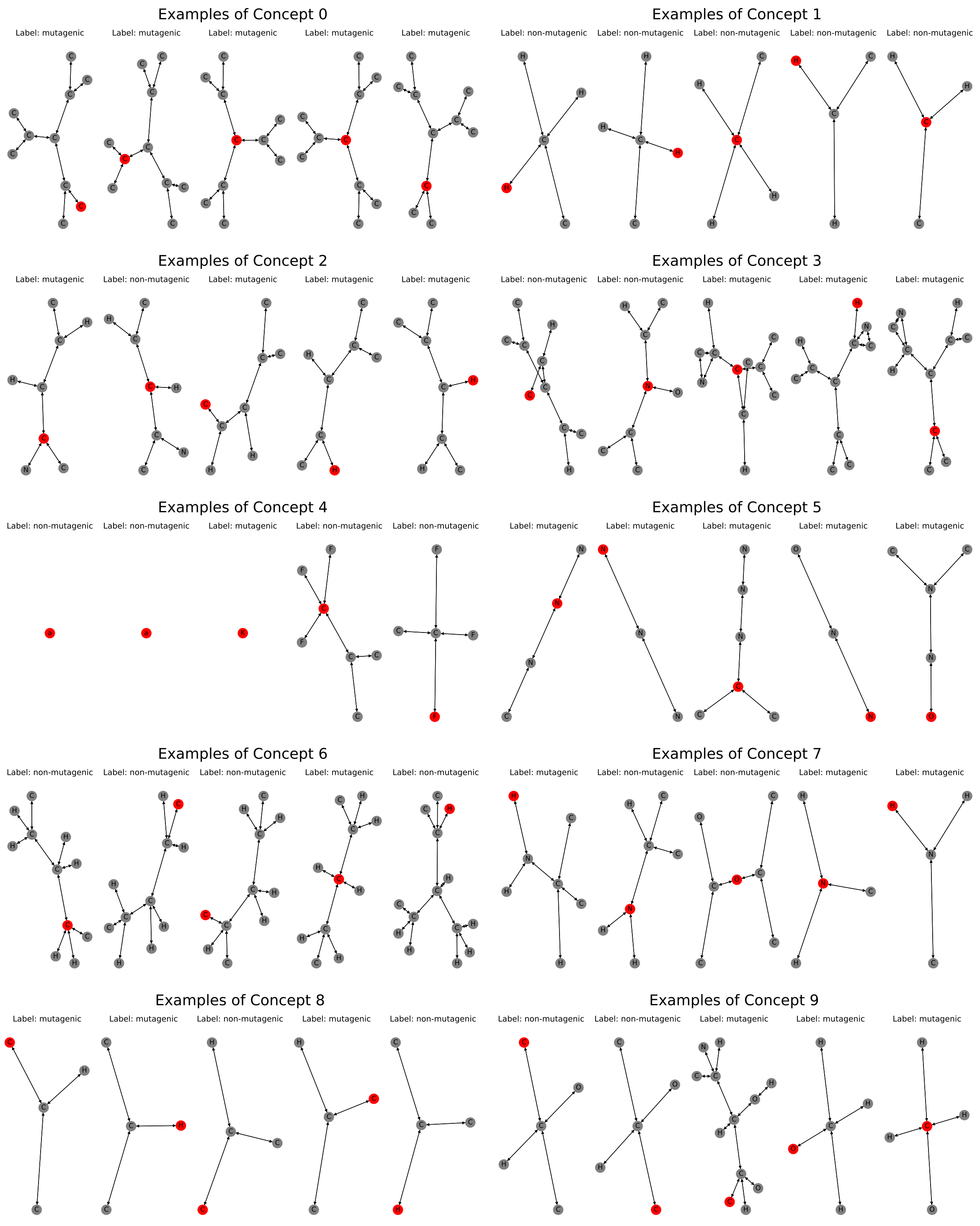

Figure 4. Concept discovered by the graph concept explainer. Part I.

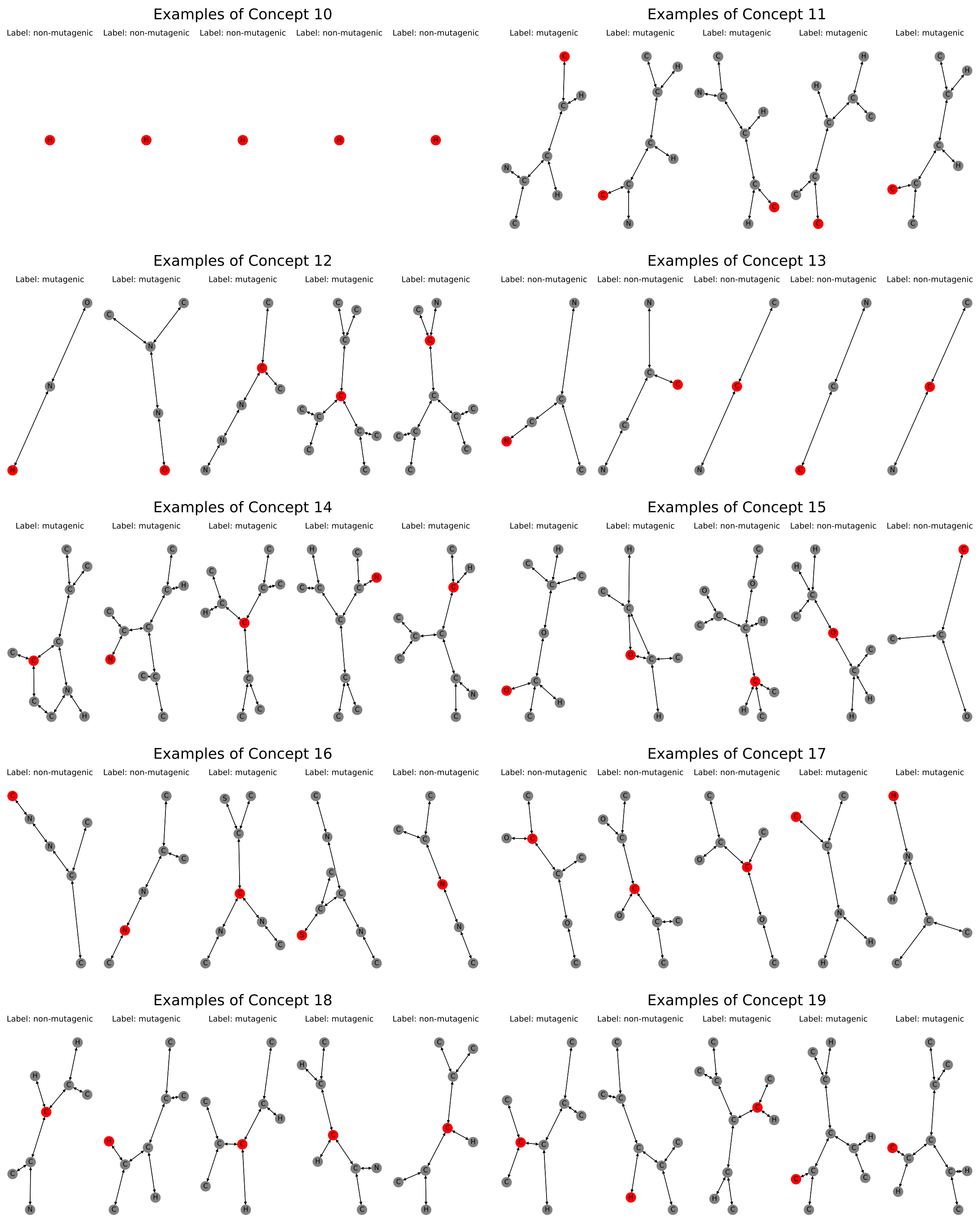

Figure 5. Concept discovered by the graph concept explainer. Part II.

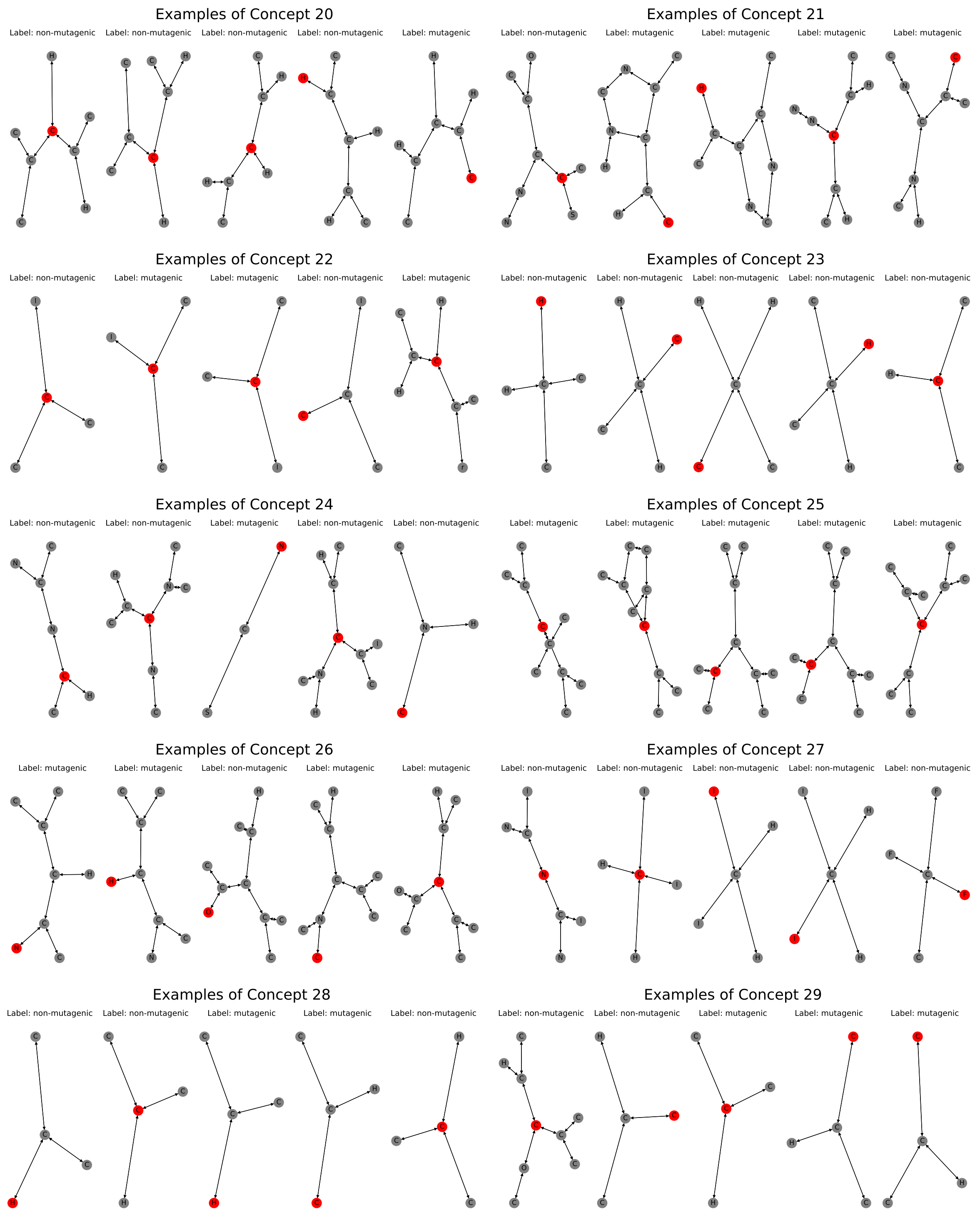

Figure 6. Concept discovered by the graph concept explainer. Part III.

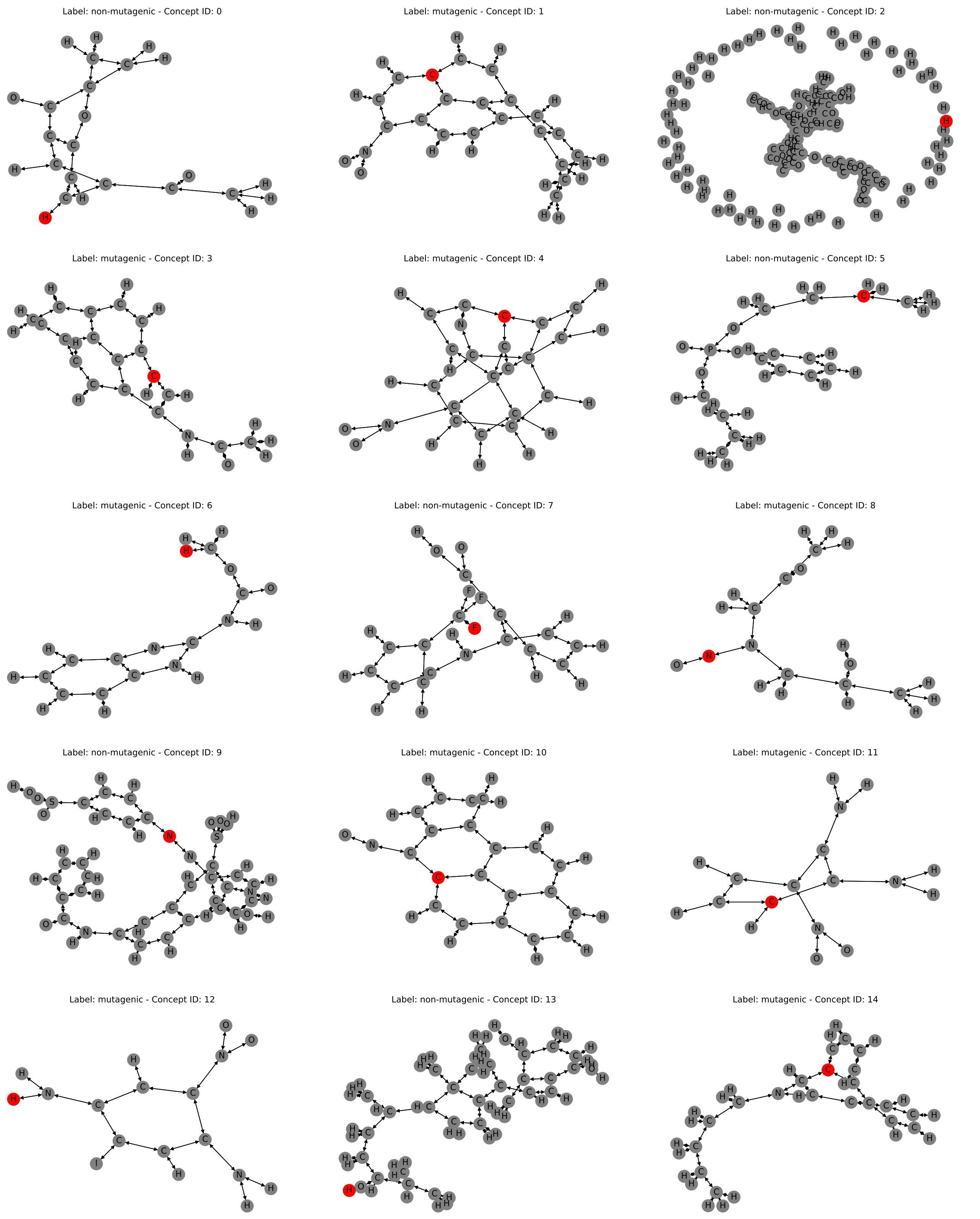
Label: non-mutagenic - Concept ID: 0
Label: mutagenic - Concept ID: 1
Label: non-mutagenic - Concept ID: 2
Label: mutagenic - Concept ID: 3
Label: mutagenic - Concept ID: 4
Label: non-mutagenic - Concept ID: 5
Label: mutagenic - Concept ID: 6
Label: non-mutagenic - Concept ID: 7
Label: mutagenic - Concept ID: 8
Label: non-mutagenic - Concept ID: 9
Label: mutagenic - Concept ID: 10
Label: mutagenic - Concept ID: 11
Label: mutagenic - Concept ID: 12
Label: non-mutagenic - Concept ID: 13
Label: mutagenic - Concept ID: 14

Figure 7. Full molecule corresponding to the closest node embedding to the concept centroid. Part I.

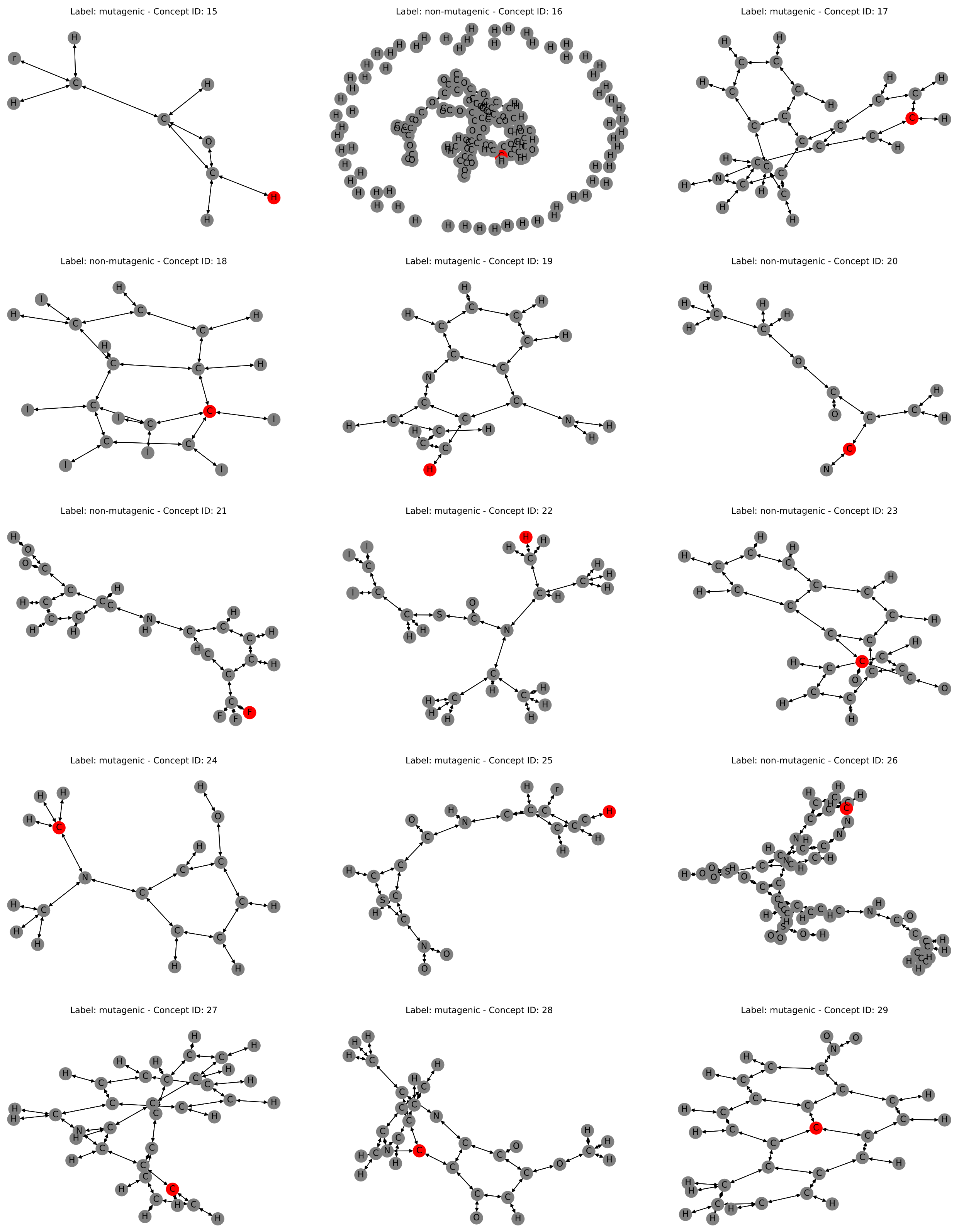
Label: mutagenic - Concept ID: 15
Label: non-mutagenic - Concept ID: 16
Label: mutagenic - Concept ID: 17
Label: non-mutagenic - Concept ID: 18
Label: mutagenic - Concept ID: 19
Label: non-mutagenic - Concept ID: 20
Label: non-mutagenic - Concept ID: 21
Label: mutagenic - Concept ID: 22
Label: non-mutagenic - Concept ID: 23
Label: mutagenic - Concept ID: 24
Label: mutagenic - Concept ID: 25
Label: non-mutagenic - Concept ID: 26
Label: mutagenic - Concept ID: 27
Label: mutagenic - Concept ID: 28
Label: mutagenic - Concept ID: 29

Figure 8. Full molecule corresponding to the closest node embedding to the concept centroid. Part II.

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.

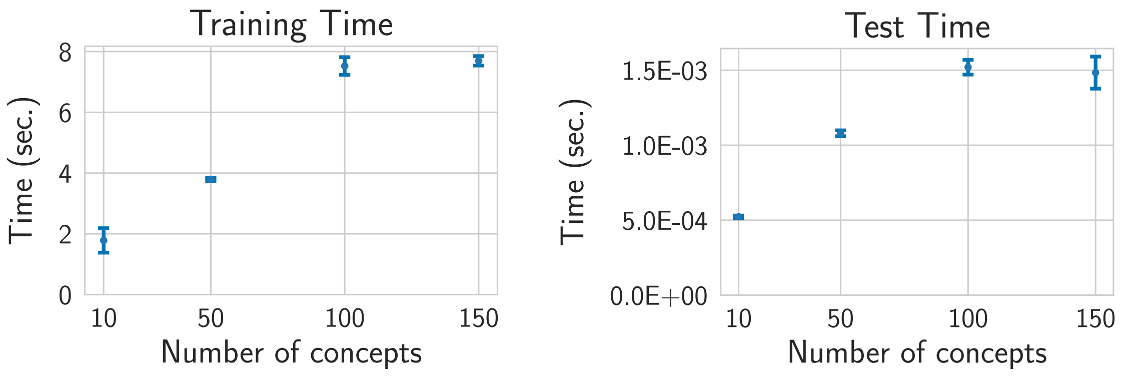

Figure 9. DCR computational time on pre-trained concept embeddings from the CUB dataset.

F. 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 2, we show the comparison with state-of-the-art Neural Symbolic frameworks, as described in the main text. Moreover, in Table 3, we show the entire list of global rules learned by DCR, showing that it actually captured perfectly the semantics of the addition relation.

Table 3. 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
$y_0 \leftarrow f_{00} \wedge f_{10}$	93
$y_1 \leftarrow f_{00} \wedge f_{11}$	110
$y_1 \leftarrow f_{01} \wedge f_{10}$	102
$y_2 \leftarrow f_{00} \wedge f_{12}$	89
$y_2 \leftarrow f_{01} \wedge f_{11}$	119
$y_2 \leftarrow f_{02} \wedge f_{10}$	101
$y_3 \leftarrow f_{01} \wedge f_{12}$	124
$y_3 \leftarrow f_{03} \wedge f_{10}$	96
$y_3 \leftarrow f_{02} \wedge f_{11}$	115
$y_3 \leftarrow f_{00} \wedge f_{13}$	100
$y_4 \leftarrow f_{03} \wedge f_{11}$	121
$y_4 \leftarrow f_{04} \wedge f_{10}$	84
$y_4 \leftarrow f_{01} \wedge f_{13}$	137
$y_4 \leftarrow f_{02} \wedge f_{12}$	105
$y_4 \leftarrow f_{00} \wedge f_{14}$	112
$y_5 \leftarrow f_{01} \wedge f_{14}$	104
$y_5 \leftarrow f_{03} \wedge f_{12}$	105
$y_5 \leftarrow f_{04} \wedge f_{11}$	113
$y_5 \leftarrow f_{00} \wedge f_{15}$	95
$y_5 \leftarrow f_{02} \wedge f_{13}$	90
$y_5 \leftarrow f_{05} \wedge f_{10}$	95
$y_6 \leftarrow f_{02} \wedge f_{14}$	92
$y_6 \leftarrow f_{05} \wedge f_{11}$	96
$y_6 \leftarrow f_{00} \wedge f_{16}$	109
$y_6 \leftarrow f_{04} \wedge f_{12}$	91
$y_6 \leftarrow f_{01} \wedge f_{15}$	86
$y_6 \leftarrow f_{03} \wedge f_{13}$	107
$y_6 \leftarrow f_{06} \wedge f_{10}$	92
$y_7 \leftarrow f_{00} \wedge f_{17}$	100
$y_7 \leftarrow f_{04} \wedge f_{13}$	108
$y_7 \leftarrow f_{01} \wedge f_{16}$	103
$y_7 \leftarrow f_{02} \wedge f_{15}$	81
$y_7 \leftarrow f_{07} \wedge f_{10}$	103
$y_7 \leftarrow f_{06} \wedge f_{11}$	137
$y_7 \leftarrow f_{05} \wedge f_{12}$	87
$y_7 \leftarrow f_{03} \wedge f_{14}$	117
$y_8 \leftarrow f_{05} \wedge f_{13}$	72
$y_8 \leftarrow f_{01} \wedge f_{17}$	122
$y_8 \leftarrow f_{03} \wedge f_{15}$	99
$y_8 \leftarrow f_{02} \wedge f_{16}$	97
$y_8 \leftarrow f_{06} \wedge f_{12}$	90
$y_8 \leftarrow f_{08} \wedge f_{10}$	96
$y_8 \leftarrow f_{07} \wedge f_{11}$	116
$y_8 \leftarrow f_{04} \wedge f_{14}$	106
$y_8 \leftarrow f_{00} \wedge f_{18}$	100
$y_8 \leftarrow f_{03} \wedge f_{16}$	1
$y_9 \leftarrow f_{04} \wedge f_{15}$	87
$y_9 \leftarrow f_{08} \wedge f_{11}$	112
$y_9 \leftarrow f_{06} \wedge f_{13}$	76
$y_9 \leftarrow f_{01} \wedge f_{18}$	113
$y_9 \leftarrow f_{00} \wedge f_{19}$	94
$y_9 \leftarrow f_{03} \wedge f_{16}$	89
$y_9 \leftarrow f_{09} \wedge f_{10}$	100
$y_9 \leftarrow f_{07} \wedge f_{12}$	110
$y_9 \leftarrow f_{02} \wedge f_{17}$	102
$y_9 \leftarrow f_{05} \wedge f_{14}$	89
$y_{10} \leftarrow f_{01} \wedge f_{19}$	115
$y_{10} \leftarrow f_{06} \wedge f_{14}$	97
$y_{10} \leftarrow f_{09} \wedge f_{11}$	100
$y_{10} \leftarrow f_{08} \wedge f_{12}$	100
$y_{10} \leftarrow f_{07} \wedge f_{13}$	113
$y_{10} \leftarrow f_{04} \wedge f_{16}$	94
$y_{10} \leftarrow f_{03} \wedge f_{17}$	89
$y_{10} \leftarrow f_{02} \wedge f_{18}$	103
$y_{10} \leftarrow f_{05} \wedge f_{15}$	75
$y_{11} \leftarrow f_{08} \wedge f_{13}$	89
$y_{11} \leftarrow f_{03} \wedge f_{18}$	105
$y_{11} \leftarrow f_{07} \wedge f_{14}$	94
$y_{11} \leftarrow f_{09} \wedge f_{12}$	97
$y_{11} \leftarrow f_{04} \wedge f_{17}$	111
$y_{11} \leftarrow f_{05} \wedge f_{16}$	86
$y_{11} \leftarrow f_{02} \wedge f_{19}$	105
$y_{11} \leftarrow f_{06} \wedge f_{15}$	104
$y_{12} \leftarrow f_{03} \wedge f_{19}$	98
$y_{12} \leftarrow f_{04} \wedge f_{18}$	87
$y_{12} \leftarrow f_{06} \wedge f_{16}$	105
$y_{12} \leftarrow f_{07} \wedge f_{15}$	96
$y_{12} \leftarrow f_{09} \wedge f_{13}$	106
$y_{12} \leftarrow f_{05} \wedge f_{17}$	94
$y_{12} \leftarrow f_{08} \wedge f_{14}$	87
$y_{13} \leftarrow f_{06} \wedge f_{17}$	106
$y_{13} \leftarrow f_{08} \wedge f_{15}$	85
$y_{13} \leftarrow f_{09} \wedge f_{14}$	82
$y_{13} \leftarrow f_{07} \wedge f_{16}$	118
$y_{13} \leftarrow f_{05} \wedge f_{18}$	79
$y_{13} \leftarrow f_{04} \wedge f_{19}$	100
$y_{14} \leftarrow f_{06} \wedge f_{18}$	105
$y_{14} \leftarrow f_{07} \wedge f_{17}$	98
$y_{14} \leftarrow f_{05} \wedge f_{19}$	78
$y_{14} \leftarrow f_{09} \wedge f_{15}$	74
$y_{14} \leftarrow f_{08} \wedge f_{16}$	101
$y_{15} \leftarrow f_{09} \wedge f_{16}$	107
$y_{15} \leftarrow f_{08} \wedge f_{17}$	95
$y_{15} \leftarrow f_{07} \wedge f_{18}$	103
$y_{15} \leftarrow f_{06} \wedge f_{19}$	111
$y_{16} \leftarrow f_{07} \wedge f_{19}$	115
$y_{16} \leftarrow f_{09} \wedge f_{17}$	100
$y_{16} \leftarrow f_{08} \wedge f_{18}$	84
$y_{17} \leftarrow f_{09} \wedge f_{18}$	100
$y_{17} \leftarrow f_{08} \wedge f_{19}$	86
$y_{18} \leftarrow f_{09} \wedge f_{19}$	102

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].

G. Complexity of logic rules

We compute rule complexity as the average size of the learnt logic rules. Table 4 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 4. 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

H. 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.