Integrating Local and Global Interpretability for Deep Concept-Based Reasoning Models

David Debot[1] and Giuseppe Marra[1]

Department of Computer Science, KU Leuven
{david.debot,giuseppe.marra}@kuleuven.be

Abstract. Two different approaches for interpretable Concept-Based Models (CBMs) exist: locally interpretable CBMs, which allow humans to understand the prediction of individual instances, and globally interpretable CBMs, which provide a broader understanding of their reasoning. In practice, the former focus on achieving high predictive accuracy, while the latter emphasize robustness and verifiability. To bridge this gap between extremes, we propose a hybrid model that integrates the strengths of both approaches. Our model, called Unified Concept Reasoner (UCR), leverages the high explainability of globally interpretable CBMs and high accuracy of locally interpretable CBMs, resulting in a powerful CBM with two heads that can be used for prediction. In our preliminary experimental evaluation, we show that UCR reaches comparable accuracy with competitors, converges to coherent global and local heads and is more stable w.r.t. hyperparameters.

Keywords: Concept-based models · Explainable AI · Neurosymbolic AI

1 Introduction

Within the broader field of explainable AI (xAI) [1], intrinsically explainable models are constructed in such a way that their decision-making processes can be understood directly, without the need for additional *post-hoc* techniques to interpret them. Concept-based models (CBMs) are a prominent group within this category [12]. CBMs are machine learning models that incorporate high-level, human-understandable features called *concepts* directly within their architecture. For example, when a CBM should predict whether an object in an image is an apple, some possible concepts might include whether the object is *round*, *red*, *soft*, and so on. The representation of human-understandable concepts allows CBMs to *explain* to users which concepts were used in making a decision. Recently, many CBMs have been developed, such as Concept Bottleneck Models (CBNMs) [7], Concept Embedding Models (CEMs) [15], as well as various other models [6, 10, 11, 13, 14].

Apart from explainability, a stronger desideratum in these models is *interpretability*, which assesses whether a human can discern *how* concepts are utilized to make predictions. Interpretability can be categorized into *local* and *global*, with global interpretability representing a stronger criterion [16]. Local interpretability allows for understanding the prediction of individual instances. It enables a human to trace how the concepts contribute to the specific output prediction for a given input. For example, many post-hoc xAI methods, such as saliency maps, provide local interpretability by highlighting which parts of an input influenced the prediction. In contrast, global interpretability provides a broader understanding of how concepts affect predictions across all possible inputs. This means a human can grasp the overall logic and rules governing the model's predictions, regardless of the specific input. A notable example of a globally interpretable model is a decision tree, where the entire decision-making process is transparent and comprehensible. Other examples include a logic program, where the logic rules used for prediction are visible to the human, and logistic regression, where the weights can be inspected.

Deep Concept Reasoner (DCR) models [2] are examples of CBMs that exhibit local interpretability. For a given input, a DCR model generates a logic rule that is evaluated on the current concepts to make a prediction. Since the logic rule describes how the concepts contribute to the prediction, the prediction can be interpreted locally - but not globally, as DCR does not offer a comprehensive view of all possible rules that can be used for prediction. Therefore, DCR's overall decision-making process and the full range of possible rules are not transparent when considering the model as a whole.

In contrast, Concept-based Memory Reasoner (CMR) models [3] are CBMs that provide global interpretability. A CMR model learns logic rules in a memory, and for a given input, selects one of these rules for evaluation using the concepts. CMR's global interpretability arises from the transparency of the rules stored in the memory: a human can inspect these rules to understand how the output can be predicted from the concepts.

Local CBMs, like DCR, and global CBMs, like CMR, usually belong to different classes of approaches with different foci. Local methods focus on building models that are as accurate as their black-box counterpart, while providing some understanding of the decision-making process. On the other hand, global methods focus on robustness and verifiability, as their global behaviour can be inspected and verified. However, current models focus on the two extremes of this local-global dimension.

In this paper, we try to fill this gap by bridging the previous two CBMs. In particular, we present the following contributions:

- We propose to project the probabilistic semantics of CMR onto DCR, providing a probabilistic perspective for DCR's local explanations, that we call Probabilistic DCR (PDCR).
- We demonstrate that PDCR can be interpreted using a variational approximation of CMR, offering a deeper understanding of the relationship between these two models.
- We leverage this insight to introduce a new CBM called Unified Concept Reasoner (UCR), which integrates both CMR and PDCR. This model features two distinct heads: one providing global interpretability and the other

offering local interpretability. We also present a preliminary empirical comparison of UCR with CMR and DCR.

2 Preliminaries

2.1 Concept Bottleneck Models

Concept Bottleneck Models (CBNMs) [7] are concept-based models that first predict the concepts from the input, and then predict the task solely from the concepts. Under probabilistic semantics, these models can be trained by optimizing the log-likelihood of the data:[1]

$$\max \sum_{(\hat{x},\hat{c},\hat{y})\in\mathcal{D}} \log p(\hat{y},\hat{c}|\hat{x}) = \max \sum_{(\hat{x},\hat{c},\hat{y})\in\mathcal{D}} \log p(\hat{c}|\hat{x}) + \log p(\hat{y}|\hat{c}) \tag{1}$$

where the $(\hat{x},\hat{c},\hat{y})$ are triplets of input, concept labels and task labels. Typically, $p(c|x)$ is a neural network (called the *concept predictor*) and $p(y|c)$ is either a neural network or logistic regression (called the *task predictor*). There is a distinction between Concept Bottleneck models (CBNMs) and more general Concept-Based Models (CBMs). While in CBNMs, only the concepts are used for task prediction, thus behaving as a bottleneck, in CBMs, other information can be passed to the task predictor. We will focus on the more general CBM class in this paper.

2.2 Deep Concept Reasoner

In this section and the following ones, we consider rules that are conjunctions of concepts or their negation. For example, for a concept set $\{yellow, round\}$, some example rules are $y \leftarrow yellow \wedge round$, $y \leftarrow yellow \wedge \neg round$ and $y \leftarrow round$.

Deep Concept Reasoner (DCR) [2] is a CBM that first generates a fuzzy logic rule from the input and then evaluates this rule using concept predictions, allowing for local interpretability. Specifically, for each concept j, DCR's task predictor employs two neural networks $\phi_j : \mathbb{R}^m \to [0,1]$ and $\psi_j : \mathbb{R}^m \to [0,1]$ to predict from some embedding predicted from the input respectively the *role* and *relevance* of that concept. The relevance determines whether the concept is present in the rule or not, while the role determines whether it is present as a positive or negative literal. For example, consider the concept set $\{yellow, round, soft\}$. In the rule $y \leftarrow yellow \wedge \neg round$, the concept *yellow* is relevant $(\phi = 1)$ and has a positive role $(\psi = 1)$, *round* is relevant $(\phi = 1)$ has a negative role $(\psi = 0)$, and *soft* is irrelevant $(\phi = 0)$, making its role inconsequential. These semantics are formalized through the following logic formula used for task prediction:

$$y \Leftrightarrow \bigwedge_{j=1}^{n_C} (\neg\psi_j \vee (\phi_j \Leftrightarrow c_j)) \tag{2}$$

[1] When it is clear from the context, we abbreviate assignments to variables, e.g. $p(y = \hat{y})$ becomes $p(\hat{y})$.

where each ψ_j and ϕ_j are neural network outputs on some embedding and c represents the standard concept predictions. The Boolean semantics are relaxed into fuzzy semantics by employing some t-norm[2], relaxing the discrete binary values of each ψ, ϕ and c into continuous values between 0 and 1. This model can be trained like any other CBM by providing supervision on the task and the concepts. In case there are multiple tasks (i.e. multilabel or multiclass classification), the tasks are modelled independently of each other.

A notable issue with DCR is the degrees of freedom in the rule generation, as multiple rules can yield the same task prediction. For example, consider concept predictions $yellow = 1$ and $round = 0$. To predict $y = 1$, DCR can predict (and evaluate) several different rules: $y \leftarrow yellow$, $y \leftarrow yellow \land \neg round$, $y \leftarrow \neg round$, among others. This freedom exists because DCR can generate a distinct rule for each input, unlike traditional rule learners such as decision trees and ILP, for which the rule(s) do not depend on the input. In practice, the human needs to tune the model in order to learn rules that they consider meaningful and interpretable, as simply maximizing the likelihood of the data can result in any of these rules being learned.

In DCR, this tuning consists of a tunable bias for *simple* (i.e. short) rules. This is achieved by encoding a degree of competition for relevance among the concepts using a special activation function that incorporates a Softmax operation. This activation function has a *temperature* hyperparameter, which requires careful tuning to obtain meaningful rules in practice. However, a significant drawback of this model is that this temperature parameter is challenging for humans to interpret, complicating the tuning process.

2.3 Concept-based Memory Reasoner

Concept-based Memory Reasoner (CMR) [3] is a CBM that also evaluates a logic rule for task prediction, but it differs from DCR in several key aspects. Firstly, instead of directly predicting the rule, it learns a set of logic rules stored in a memory. For a given input, CMR learns to *select* one of these rules for evaluation. This approach ensures global interpretability, as all rules used for task prediction are transparent and accessible to the human. Secondly, CMR employs probabilistic semantics rather than fuzzy semantics. Thirdly, CMR represents rules differently. Instead of encoding the role and relevance of a concept as two separate variables, CMR uses a single three-valued categorical variable to signify a positive role, a negative role or irrelevance.

We will now explain CMR in more detail. CMR has a memory, also referred to as *rulebook*, comprising n_R embeddings, each representing a rule. Each rule embedding $\theta \in \mathbb{R}^k$ is decoded into a logic rule using a neural network $\rho : \mathbb{R}^k \to \mathbb{R}^{n_C \times 3}$, parametrizing the logits of n_C three-valued categorical distributions, with each distribution corresponding to a concept, indicating its role (positive, negative, or irrelevant) in the rule. Then, the set $\{\rho(\theta_i) \,|\, i \in [1, n_R]\}$ forms the

[2] An example is the Gödel t-norm, for which $a \land b = min(a, b)$, $a \lor b = max(a, b)$ and $\neg a = 1 - a$.

decoded rulebook, representing a set of logic rules, each of which can be evaluated. CMR includes a neural network $s : \mathbb{R}^l \rightarrow \mathbb{R}^{n_R}$ that functions as a selector mechanism over these rules, parametrizing for a given embedding predicted from the input n_R logits of a categorical distribution, one per rule in the memory. The task prediction then involves evaluating the selected logic rule. These semantics are captured in the following logic formula that is used for task prediction:

$$y \Leftrightarrow \bigvee_{i=1}^{n_R} \left((s = i) \bigwedge_{j=1}^{n_C} (\rho_{ij} = I) \vee (((\rho_{ij} = P) \wedge c_j) \vee ((\rho_{ij} = N) \wedge \neg c_j)) \right) \tag{3}$$

which is the disjunction over all rules in the memory, where each disjunct combines the selection of a rule $(s = i)$ and the corresponding rule body. The rule body evaluates to true if for each concept, it is either irrelevant $(\rho_{ij} = I)$ (i.e. not in the rule body), it has a positive role $(\rho_{ij} = P)$ and is true (c_j), or it has a negative role $(\rho_{ij} = N)$ and is false $(\neg c_j)$. CMR employs probabilistic semantics: s and each ρ_{ij} are categorical random variables, and the concepts are Bernoulli random variables. Under these semantics, computing the likelihood of the task being true using Eq. (3) corresponds with:

$$p(y|\hat{x}) = \sum_{\hat{s}=1}^{n_R} p(\hat{s}|\hat{x})\, p(y|\hat{s}, \hat{x}) \tag{4}$$

where

$$p(y = 1|\hat{s}, \hat{x}) = \prod_{j=1}^{n_C} (p(\rho_j = I|\hat{s}) + p(\rho_j = P|\hat{s})\, p(c_j = 1|\hat{x}) + p(\rho_j = N|\hat{s})\, p(c_j = 0|\hat{x})) \tag{5}$$

CMR's method of tuning the model to acquire meaningful rules is significantly different from DCR's approach. Firstly, CMR employs regularization to make the rules as specific as possible, aiming to learn rules that contain the fewest irrelevant concepts. The complete likelihood to be optimized by the training objective of the task predictor corresponds with:[3]

$$p(\hat{y}|\hat{x}, \hat{c}) = \sum_{\hat{s}=1}^{n_R} p(\hat{s}|\hat{x})\, p(\hat{y}|\hat{s}, \hat{c})^{\beta_1}\, p_{reg}(\rho = \hat{c}|\hat{s})^{\hat{y}} \tag{6}$$

where $\hat{c}$ consists of the ground truth concepts, β_1 is a hyperparameter, p_{reg} is the regularization, and:

$$p(y = 1|\hat{s}, \hat{c}) = \prod_{j=1}^{n_C} (p(\rho_j = I|\hat{s}) + p(\rho_j = P|\hat{s})\, \mathbb{1}[\hat{c}_j = 1] + p(\rho_j = N|\hat{s})\, \mathbb{1}[\hat{c}_j = 0]) \tag{7}$$

which is similar to Eq. (5). Secondly, CMR encodes a degree of competition for relevance by setting an upper limit on the number of rules that can be learned,

[3] The concept predictor $p(c|x)$ can be trained like any other CBM.

defined by the hyperparameter called the *rulebook size* n_R. This hyperparameter is considerably more interpretable for a human than DCR's temperature, making it easier to tune the model effectively.

Example. Consider a scenario where the task y is predicting whether a car should stop based on two concepts r (signalling a red light) and c (signalling cloudy weather). If the light is red, the car should stop, and the weather being cloudy is irrelevant. We have four data points: $[r, c, y]$, $[r, \neg c, y]$, $[\neg r, c, \neg y]$ and $[\neg r, \neg c, \neg y]$. By choosing $n_R = 2$, CMR will learn the rules $y \leftarrow r \wedge c$ ($\rho_r = P$ and $\rho_c = P$) and $y \leftarrow r \wedge \neg c$ ($\rho_r = P$ and $\rho_c = N$). By choosing $n_R = 1$, CMR will learn the rule $y \leftarrow r$ ($\rho_r = P$ and $\rho_c = I$), where c is irrelevant.

3 Method

3.1 Probabilistic Deep Concept Reasoner

In this section, we propose two changes to DCR. We call the variant of DCR that incorporates these changes Probabilistic DCR (PDCR). The first change we propose is to use CMR's representation of a rule instead of DCR's. This means that instead of having two independent variables per concept representing its role and relevance, the model uses a single three-valued categorical variable per concept, signifying a positive or negative role, or irrelevance. Consequently, the logic formula DCR uses for task prediction (Eq. (2)) changes to:

$$y \Leftrightarrow \bigwedge_{j=1}^{n_C} ((\rho_j = I) \vee (((\rho_j = P) \wedge c_j) \vee ((\rho_j = N) \wedge \neg c_j))) \tag{8}$$

which closely corresponds to CMR's logic formula (Eq. (3)), the difference being that there is no disjunction over rules in a memory. The second change we propose is to employ probabilistic semantics instead of fuzzy semantics. Under probabilistic semantics, computing the likelihood of the task using the above logic formula corresponds with[4]:

$$p(y=1|\hat{x}) = \prod_{j=1}^{n_C} (p(\rho_j = I|\hat{x}) + p(\rho_j = P|\hat{x})\, p(c_j = 1|\hat{x}) + p(\rho_j = N|\hat{x})\, p(c_j = 0|\hat{x})) \tag{9}$$

which is similar to Eq. (5) of CMR, the difference being that the role and relevance of each concept more generally depends on the input ($p(.|\hat{x})$) instead of only on which rule has been selected ($p(.|\hat{s})$).

3.2 Interpreting PDCR using a variational approximation of CMR

In this section, we demonstrate that we can interpret PDCR within a variational approximation of CMR. In particular, we show that PDCR can be seen as an

[4] This can be derived similarly as for CMR (see [3]) by exploiting the independence between the different conjuncts.

approximate distribution of CMR and that its training objective is a partial estimate of the standard lower bound of variational inference. The intuition behind this derivation is that both PDCR and CMR compute a rule embedding that is decoded into a logic rule. While the way these embeddings are obtained is different (i.e. a neural encoder for PDCR and a selection from a trainable memory for CMR), PDCR can be thought of as using a neural model to approximate the harder selection of CMR.

In order to show this, we first make the rule embeddings θ explicit inside Eq. (5) of CMR:

$$p(\hat{y}|\hat{s},\hat{x}) = \int p(\hat{\theta}|\hat{s})\, p(\hat{y}|\hat{\theta},\hat{x})\, d\hat{\theta} \tag{10}$$

where

$$p(y=1|\hat{\theta},\hat{x}) = \prod_{j=1}^{n_C} (p(\rho_j = I|\hat{\theta}) + p(\rho_j = P|\hat{\theta})\, p(c_j = 1|\hat{x}) + p(\rho_j = N|\hat{\theta})\, p(c_j = 0|\hat{x})) \tag{11}$$

Here, it becomes explicit that selecting a rule ($s = \hat{s}$) corresponds to selecting a distribution over rule embeddings θ, each of which gets decoded into roles and relevances.

We use variational inference to approximate the distribution $p(\theta|\hat{s})$ with another distribution $q(\theta|\hat{x})$. We first take the logarithm of Eq. (10):

$$\log p(\hat{y}|\hat{s},\hat{x}) = \log \int p(\hat{\theta}|\hat{s})\, p(\hat{y}|\hat{\theta},\hat{x})\, d\hat{\theta} \tag{12}$$

which is equivalent to:

$$\log p(\hat{y}|\hat{s},\hat{x}) = \log \int p(\hat{\theta}|\hat{s})\, p(\hat{y}|\hat{\theta},\hat{x}) \left(\frac{q(\hat{\theta}|\hat{x})}{q(\hat{\theta}|\hat{x})}\right) d\hat{\theta} \tag{13}$$

Because of Jensen's inequality, we obtain:

$$\log p(\hat{y}|\hat{s},\hat{x}) \geq \int q(\hat{\theta}|\hat{x}) \log \left(\frac{p(\hat{\theta}|\hat{s})\, p(\hat{y}|\hat{\theta},\hat{x})}{q(\hat{\theta}|\hat{x})}\right) d\hat{\theta} \tag{14}$$

Then, we can split the multiplication within the logarithm into a sum of logarithms, and split the integral accordingly:

$$\log p(\hat{y}|\hat{s},\hat{x}) \geq \int q(\hat{\theta}|\hat{x}) \log p(\hat{y}|\hat{\theta},\hat{x})\, d\hat{\theta} + \int q(\hat{\theta}|\hat{x}) \log \left(\frac{p(\hat{\theta}|\hat{s})}{q(\hat{\theta}|\hat{x})}\right) d\hat{\theta} \tag{15}$$

which results into the following lower bound:

$$\log p(\hat{y}|\hat{s},\hat{x}) \geq \mathbb{E}_{q(\theta|\hat{x})}[\log p(\hat{y}|\theta,\hat{x})] - KL(q(\theta|\hat{x})\,||\,p(\theta|\hat{s})) \tag{16}$$

This inequality indicates that maximizing the log-likelihood of the data can be done by maximizing the log-likelihood of the expected prediction under the approximate distribution and by minimizing the KL divergence between the approximate and the exact distribution. Intuitively, this criterion states that in order to maximize the likelihood of the data, we can compute the rule embedding θ using PDCR and then use the decoding strategy of CMR. At the same time, the KL divergence between PCDR's distribution $q(\theta|\hat{x})$ and CMR's distribution $p(\theta|\hat{s})$ should be minimized. While this minimization will provide the entry point for our joint model in Sec. 3.3, for PDCR, $p(\theta|\hat{s})$ represents just a learnable prior acting as regularizer. We will therefore neglect it in the rest of the discussion.

After substituting this in Eq. (4) of CMR, we have:

$$p(\hat{y}|\hat{x}) \gtrapprox \sum_{\hat{s}=1}^{n_R} p(\hat{s}|\hat{x})\, p(\hat{y}|\hat{s},\hat{x}) \gtrapprox \sum_{\hat{s}=1}^{n_R} p(\hat{s}|\hat{x})\, e^{\mathbb{E}_{q(\theta|\hat{x})}[\log p(\hat{y}|\theta,\hat{x})]} \tag{17}$$

where the factor with the exponential is not dependent on $\hat{s}$. Therefore, this simplifies to:

$$p(\hat{y}|\hat{x}) \gtrapprox e^{\mathbb{E}_{q(\theta|\hat{x})}[\log p(\hat{y}|\theta,\hat{x})]} \tag{18}$$

or in log-space,

$$\log p(\hat{y}|\hat{x}) \gtrapprox \mathbb{E}_{q(\theta|\hat{x})}[\log p(\hat{y}|\theta,\hat{x})] \tag{19}$$

Normally, this expectation has to be approximated by sampling; however, we approximate the expectation by evaluating the expression in the distribution's maximum a posteriori estimate $\theta_{\hat{x}}$, which we parametrize with a neural network f.[5] This results in:

$$\log p(\hat{y}|\hat{x}) \gtrapprox \log p(\hat{y}|\theta_{\hat{x}},\hat{x}) \qquad \text{where} \quad \theta_{\hat{x}} = f(\hat{x}) \tag{20}$$

Lastly, as, the likelihood $p(\hat{y}|\theta,\hat{x})$ is the same for PDCR as for CMR (given by Eq. (11)), we can interpret this as follows: this variational approximation of CMR can be optimized by optimizing the objective of PDCR.

3.3 Unified Concept Reasoner: combining PDCR and CMR

We propose to exploit this connection between PDCR and CMR to define a new CBM that combines both models. This hybrid model features two heads for task prediction: one that is locally interpretable and works by generating and evaluating a logic rule (as in PDCR), and one that is globally interpretable and works by selecting a rule from rules learned in a memory and evaluating it (as in CMR). We call this model Unified Concept Reasoner, abbreviated UCR. Fig. 1 shows an example task prediction at test time, where the human can choose between the two heads.

Training objective. UCR is trained by using the variational approximation outlined in the previous section, incorporating Eq. (16) in the CMR likelihood

[5] This can be justified by considering a very peaked normal distribution, in the limit becoming a Dirac delta. In this case, the network f would be parameterizing the mean of the distribution or, similarly, the point mass of the delta.

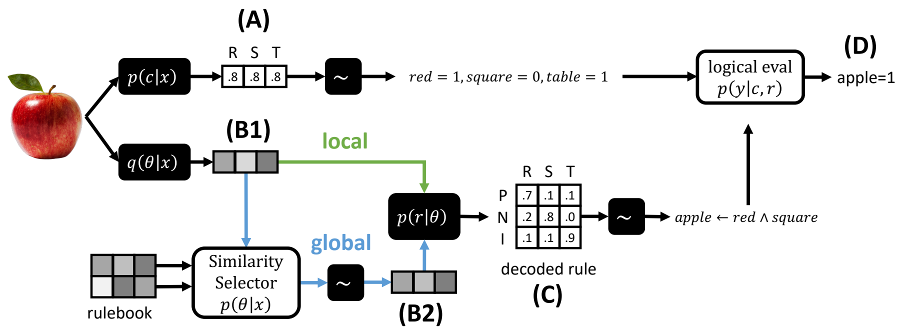


Fig. 1: Example task prediction, adapted from Figure 2 in [3]. In this figure, we sample from each distribution ($\sim$) for clarity, but in practice we compute every probability exactly. Each black box containing a probability distribution is parametrized by a neural network. (A) As in other CBMs, concepts are predicted from the input. (B1-**local**) A single local rule embedding is predicted from the input. (B2-**global**) A global rule embedding is selected from the global rulebook according to a similarity score with the local embedding. The rule embedding is decoded into a rule (C) which is then evaluated with the predicted concepts to predict the task (D).

equation (Eq. (6)) without ignoring the term with the KL divergence. This results in the following likelihood being optimized during training:

$$p(\hat{y}|\hat{x},\hat{c}) \geq \sum_{\hat{s}=1}^{n_R} p(\hat{s}|\hat{x})\, e^{\beta_1(\mathbb{E}_{q(\theta|\hat{x})}[\log p(\hat{y}|\theta,\hat{c})]-KL(q(\theta|\hat{x})\,||\,p(\theta|\hat{s})))} p_{reg}(\rho=\hat{c}|\hat{s})^{\hat{y}} \quad (21)$$

Optimizing the lower bound of this likelihood achieves three objectives. First, it ensures that using the approximate distribution $q(\theta|\hat{x})$ for rule prediction results in accurate task predictions. Second, it promotes the specificity of the decoded rule by minimizing the inclusion of irrelevant concepts. Third, it ensures that the approximate distribution $q(\theta|\hat{x})$ approximates the true distribution $p(\theta|\hat{x})$, which is defined by $p(s|\hat{x})$ and $p(\theta|\hat{s})$.

Intuitively, when training UCR, the local path is used to predict rules that make correct task predictions and are as specific as possible, and the global path with its limited rule capacity acts as a regularizer, introducing a degree of competition for relevance. *This provides a key advantage over DCR when designing the model*: to tune UCR to learn meaningful rules, CMR's rulebook size hyperparameter n_R is used instead of DCR's temperature hyperparameter, which is arguably one of the main drawbacks of DCR [3]. In terms of advantages over CMR, the use of UCR's local head helps navigating the rule embeddings space (i.e. θ) and avoids CMR's need to reinitialize parameters of the distribution $p(s|x)$ to escape local minima during training.

Two prediction heads at test time. At test time, we can choose to use either the approximate distribution $q(\theta|\hat{x})$ or the distribution $p(\theta|\hat{x})$ to provide an embedding that will be decoded into a rule for task prediction, representing respectively the locally and globally interpretable head; the local one corresponds to the PDCR approach, while the global one corresponds to the CMR approach. These two heads can be computed in the following way:

$$\begin{aligned} p_{local}(\hat{y}|\hat{x}) &= \sum_{\hat{s}=1}^{n_R} p(\hat{s}|\hat{x})\, e^{\mathbb{E}_{q(\theta|\hat{x})}[\log p(\hat{y}|\theta,\hat{x})]} \\ p_{global}(\hat{y}|\hat{x}) &= \sum_{\hat{s}=1}^{n_R} p(\hat{s}|\hat{x})\, e^{\mathbb{E}_{p(\theta|\hat{s})}[\log p(\hat{y}|\theta,\hat{x})]} \end{aligned} \tag{22}$$

For the local head, we can omit the sum over $\hat{s}$ because the terms are independent of $\hat{s}$. Moreover, we approximate the expectation in $p_{local}(\hat{y}|\hat{x})$ by evaluating the expression in the distribution's maximum a posteriori estimate $\theta_{\hat{x}}$, similarly to what was done in the previous section:

$$\mathbb{E}_{q(\theta|\hat{x})}[\log p(\hat{y}|\theta,\hat{x})] \approx \log p(\hat{y}|\theta_{\hat{x}},\hat{x}) \qquad \text{where} \quad \theta_{\hat{x}} = f(\hat{x}) \tag{23}$$

Thus, the local head simplifies to:

$$p_{local}(\hat{y}|\hat{x}) \approx p(\hat{y}|\theta_{\hat{x}},\hat{x}) \qquad \text{where} \quad \theta_{\hat{x}} = f(\hat{x}) \tag{24}$$

which corresponds to PDCR. We call the embedding $\theta_{\hat{x}}$ the *local rule embedding*.

While in CMR, the distribution $p(\theta|\hat{s})$ is a Dirac delta, in the previous section, we left this design choice open. Similarly as for the local head, at test time, we approximate $p(\theta|\hat{s})$ by evaluating the expression in the distribution's maximum a posteriori estimate to obtain CMR's semantics. Therefore, we obtain:

$$p_{global}(\hat{y}|\hat{x}) \approx \sum_{\hat{s}=1}^{n_R} p(\hat{s}|\hat{x})\, p(\hat{y}|\theta_{\hat{s}},\hat{x}) \tag{25}$$

which corresponds to CMR. We call the embeddings $\theta_{\hat{s}}$ the *global rule embeddings*. These embeddings are learnable parameters and stored in a memory, as in CMR.

Specializing for Gaussian parametrization. In order to compute the training objective in Eq. (21), we need to compute the KL divergence between $q(\theta|\hat{x})$ and $p(\theta|\hat{s})$. Let us take as distributions $p(\theta|\hat{s})$ and $q(\theta|\hat{x})$ two normal distributions with fixed isotropic covariance matrices (i.e. diagonal and all diagonal elements are the same), and means parameterized by neural networks. Then, for the univariate case, it is known that:

$$KL(q(\theta|\hat{x})\,||\,p(\theta|\hat{s})) = -\log\left(\frac{\sigma_p}{\sigma_q}\right) - \frac{1}{2} + \frac{\sigma_q^2 + (\theta_{\hat{x}} - \theta_{\hat{s}})^2}{2\sigma_p^2} \tag{26}$$

with σ_p and σ_q the standard-deviations and $\theta_{\hat{s}}$ and $\theta_{\hat{x}}$ the means of respectively $p(\theta|\hat{s})$ and $q(\theta|\hat{x})$. As we do not parametrize the standard-deviations, the term with the logarithm is constant. During optimization, we can ignore the constant terms:

$$KL(q(\theta|\hat{x})\,||\,p(\theta|\hat{s})) \propto \frac{(\theta_{\hat{x}} - \theta_{\hat{s}})^2}{2\sigma_p^2} \tag{27}$$

After neglecting the constant terms during optimization, we note that σ_p can be used as a hyperparameter β signifying the importance of the KL term in the training objective. We use this in Eq. (27) and generalize to multivariate distributions, obtaining:

$$KL(q(\theta|\hat{x}) \,||\, p(\theta|\hat{s})) \propto \beta \cdot ||\theta_{\hat{x}} - \theta_{\hat{s}}||_2^2 \tag{28}$$

In CMR, the categorical distribution $p(s|\hat{x})$ is parametrized by a neural network that takes $\hat{x}$ as input. We propose to instead parametrize the logits of $p(s|\hat{x})$ using a similarity between the local rule embedding $\theta_{\hat{x}}$ and each global rule embedding $\theta_{\hat{s}}$:

$$p(\hat{s}|\hat{x}) = \frac{e^{\alpha \cdot sim(\theta_{\hat{x}}, \theta_{\hat{s}})}}{\sum_{\bar{s}=1}^{n_R} e^{\alpha \cdot sim(\theta_{\hat{x}}, \theta_{\bar{s}})}} \tag{29}$$

where α is a hyperparameter, and the similarity measure is based on the mean squared error distance metric:

$$sim(\theta_1, \theta_2) = \frac{1}{1 + ||\theta_1 - \theta_2||_2^2} \tag{30}$$

with $||.||_2$ denoting the L2-norm. This way, the distribution $p(s|\hat{x})$ is non-parametric except for parameters shared with $q(\theta|\hat{x})$ and each $\theta_{\hat{s}}$. As a consequence, the global head can select a global rule embedding only if it is the one that is the closest to the local rule embedding, which is ideally the global rule the most similar to the local rule.

Inherited properties. UCR inherits the powerful property from CMR and DCR of being a universal binary classifier [5], a characteristic it shares with neural networks. Consequently, with proper tuning, UCR can obtain black-box accuracy for any concept set, which is a property that sets it apart from many other CBMs like Concept Bottleneck Models. The proof is the same as for CMR (see [3]). Another notable property shared with CMR is that the likelihood computation is tractable in the number of rules and concepts, more specifically $O(n_C \cdot n_R)$. This follows from Eqs. (11), (21) and (28) to (30).

Benefits over CMR and DCR. We restate some of the advantages of UCR over both DCR and CMR. Compared to DCR, UCR offers a globally interpretable head, while DCR only offers local interpretability. Additionally, UCR simplifies the tuning process by adopting CMR's number of rules hyperparameter, rather than relying on DCR's temperature hyperparameter, which is more difficult to interpret. UCR also shares CMR's benefit of interpreting rules as data prototypes, as well as its probabilistic as opposed to fuzzy semantics, which have been shown to be less intuitive in the learning setting [8]. Compared to CMR, UCR holds the potential for higher accuracy due to its less constrained, locally interpretable head. Moreover, UCR avoids CMR's need for reinitialization during training to escape local optima.

4 Experiments

Our preliminary experiments are aimed at answering the following research questions. (1) Does UCR obtain similar accuracy as DCR and CMR? (2) Do the rules utilized by UCR's local and global head correspond at test time? (3) Does UCR learn meaningful rules? (4) How does UCR's sensitivity to hyperparameters w.r.t. the learned rules compare to DCR's?

4.1 Experimental setting

We describe essential information about the experiments.

Datasets. We use the dataset MNIST-addition [9], where the input consists of two MNIST images, each denoting a digit, and where there are 19 tasks, one per possible sum of these digits. The concepts are the possible digits per image.

Evaluation. We measure classification performance on the tasks using subset accuracy, and evaluate both heads of UCR. We also measure *rule correspondence* between both heads, which we define as the percentage of data points where the local rule and selected global rule correspond (averaged over the tasks). We report the mean and standard-deviation of these metrics over three runs.

Baselines. We compare UCR's performance with CMR and DCR.

4.2 Results and discussion

UCR achieves similar task accuracy as DCR and CMR. Tab. 1 presents the accuracy of UCR, CMR and DCR. The local head of UCR shows accuracy on par with CMR and outperforms[6] DCR for all choices of n_R. Additionally, the reduction in accuracy when switching from the local to the global head in UCR is minimal for certain choices of n_R. In contrast, the performance of DCR is only marginally influenced by the choice of temperature.

Table 1: Task accuracy on the training and test set. The results of CMR are taken from [3]. For DCR, we consider different values for the temperature τ; for UCR, we consider different rulebook sizes n_R, and we report the accuracy of both heads.

	CMR	DCR			UCR					
		$\tau = .1$	$\tau = 1$	$\tau = 10$	$n_R = 2$		$n_R = 20$		$n_R = 50$	
					local	global	local	global	local	global
TRAIN	-	96.3$\pm$.7	95.4$\pm$.9	96.9$\pm$1.	99.9$\pm$.1	13.3$\pm$.6	99.9$\pm$.1	98.0$\pm$1.	99.9$\pm$.0	97.6$\pm$.4
TEST	97.5$\pm$.3	93.2$\pm$.6	93.1$\pm$1.	94.1$\pm$.6	97.7$\pm$.2	13.2$\pm$.7	97.8$\pm$.2	95.7$\pm$.8	97.6$\pm$.4	95.4$\pm$.5

[6] Notice that this setting is picked from [3] which differs from [2]. In the former, concepts are thresholded before task prediction to avoid leakage [4]. Moreover, in the MNIST-addition dataset, the regularization term p_{reg} of CMR and UCR provides a strong inductive bias, which helps explain why DCR performs worse.

UCR learns meaningful global rules, and their quality can be tuned by comparing both heads' accuracy. Fig. 2 shows the learned global rules of UCR for different values of n_R and for learned local rules of DCR for different values of the temperature τ. When comparing these rules with the accuracy data in Tab. 1, a clear correspondence emerges between the difference in accuracy on the training set of UCR's heads and the quality of the global rules. This indicates that the quality of the global rules can be fine-tuned by simply considering the difference in accuracy between the heads. In contrast, for DCR, no such correspondence exists, meaning that the rule quality can only be tuned through manual inspection of the rules, requiring the human to already have a clear understanding of what constitutes meaningful rules for the current task.

(a) UCR ($n_R = 1$)
$y_3 \leftarrow \Omega$

(b) UCR ($n_R = 2$)
$y_3 \leftarrow c_{1,1} \wedge c_{2,2} \wedge \Omega$
$y_3 \leftarrow c_{1,2} \wedge c_{2,1} \wedge \Omega$

(c) UCR ($n_R = 20$)
$y_3 \leftarrow c_{1,1} \wedge c_{2,2} \wedge \Omega$
$y_3 \leftarrow c_{1,2} \wedge c_{2,1} \wedge \Omega$
$y_3 \leftarrow c_{1,0} \wedge c_{2,3} \wedge \Omega$
$y_3 \leftarrow c_{1,3} \wedge c_{2,0} \wedge \Omega$

(d) UCR ($n_R = 50$)
$y_3 \leftarrow c_{1,1} \wedge c_{2,2} \wedge \Omega$
$y_3 \leftarrow c_{1,2} \wedge c_{2,1} \wedge \Omega$
$y_3 \leftarrow c_{1,0} \wedge c_{2,3} \wedge \Omega$
$y_3 \leftarrow c_{1,3} \wedge c_{2,0} \wedge \Omega$
$y_3 \leftarrow c_{1,2} \wedge \Omega$

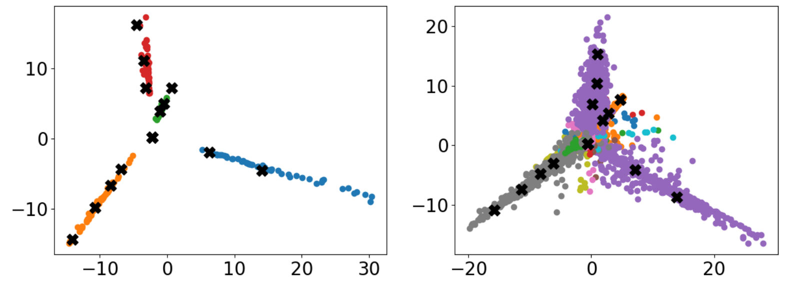
(e) DCR ($\tau = 0.1$)
$y_3 \leftarrow c_{1,3}$
$y_3 \leftarrow \neg c_{1,7}$
$y_3 \leftarrow \neg c_{2,2}$
$y_3 \leftarrow \neg c_{1,7} \wedge \neg c_{2,2}$
$y_3 \leftarrow \neg c_{1,7} \wedge c_{2,2}$
[... 5 more]

(f) DCR ($\tau = 1$)
$y_3 \leftarrow \Omega_{\{(1,0),(1,7),(1,9),(2,3),(2,4)\}}$
$y_3 \leftarrow \Omega_{\{(1,1),(1,9),(2,0),(2,2),(2,8)\}}$
$y_3 \leftarrow c_{2,0} \wedge \Omega_{\{(1,1),(1,3),(1,5),(1,9),(2,8)\}}$
$y_3 \leftarrow c_{2,1} \wedge \Omega_{\{(1,2),(2,6)\}}$
$y_3 \leftarrow c_{2,1} \wedge \Omega_{\{(1,7),(1,9),(2,6)\}}$
[... 20 more]

(g) DCR ($\tau = 10$)
$y_3 \leftarrow c_{1,1} \wedge c_{2,2} \wedge \Omega$
$y_3 \leftarrow c_{1,2} \wedge c_{2,1} \wedge \Omega$
$y_3 \leftarrow c_{1,0} \wedge c_{2,3} \wedge \Omega$
$y_3 \leftarrow c_{1,3} \wedge c_{2,0} \wedge \Omega$

Fig. 2: Rules learned by UCR and DCR with different hyperparameters for task y_3 (rulebook size n_R and temperature τ, respectively). $c_{i,j}$ denotes that the i-th digit is j. For brevity, we denote with Ω_S the conjunction of the negation of the remaining concepts except $c_{i,j}$ where $(i, j) \in S$. For UCR, we give the rules learned by the global head that are selected for at least one example where the label is true. For DCR, we give the local rules accumulated over the test set for examples where the label is true.

UCR's local and selected global rule often correspond. We calculate the rule correspondence on the test set between the local and global heads for $n_R = 20$, separately for data points where the task label is true vs false. When the label is true, the heads almost fully correspond ($98.86{\pm}0.25$), while when the label is false, the rules still often correspond ($74.21{\pm}2.01$). This is further illustrated in Fig. 3, where we do a Principal Component Analysis on the local and global rule embeddings for task y_3 on the test set. For data points where the ground truth $y_3 = 1$, the local rule embeddings form distinct clusters based on the rules they decode into, and each cluster has corresponding global rule embeddings that are only close to that cluster in the embedding space.[7] For data points where $y_3 = 0$, although the local rule embeddings are quite clustered, not every cluster has its own global rule embedding, explaining the difference in rule correspondence between $y = 1$ and $y = 0$.



(a) $y_3 = 1$ **(b)** $y_3 = 0$

Fig. 3: First two dimensions of a Principal Component Analysis of local rule embeddings (coloured dots) and global rule embeddings (black crosses) for task y_3 ($n_R = 20$) on the test set. Data points are split based on whether the label is true or false. We only show global rule embeddings that are selected for at least one example. In each figure, local rule embeddings that share the same colour decode into the same rule.

5 Conclusion

We presented a probabilistic perspective for DCR, a locally interpretable concept-based model originally utilizing fuzzy semantics. Following this, we introduced the insight that this probabilistic version of DCR can be interpreted using a variational approximation of CMR, a globally interpretable concept-based model. Building on this insight, we introduced a new concept-based model called UCR, which integrates both DCR and CMR, resulting in a model with two heads: one being globally interpretable and the other locally interpretable. We conducted a preliminary empirical investigation of this model, showing that UCR reaches comparable accuracy with competitors, converges to coherent global and local heads and is more stable w.r.t. hyperparameters.

For future works, further evaluation of UCR is required to fully assess its performance across a broader range of realistic datasets. Additionally, it would be interesting to find mathematical bounds on the correspondence between local and global rules, providing a formal understanding of their relationship.

[7] Note that, because the selector $p(s|x)$ selects based on similarity between global and local rule embeddings, it is guaranteed that, for a given input, the global rule embedding closest to the local one gets selected.

Acknowledgements

This work was supported by the KU Leuven Research Fund (STG/22/021, CELSA/24/008).

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (xai). IEEE access **6**, 52138–52160 (2018)
2. Barbiero, P., Ciravegna, G., Giannini, F., Zarlenga, M.E., Magister, L.C., Tonda, A., Lió, P., Precioso, F., Jamnik, M., Marra, G.: Interpretable neural-symbolic concept reasoning. In: International Conference on Machine Learning. pp. 1801–1825. PMLR (2023)
3. Debot, D., Barbiero, P., Giannini, F., Ciravegna, G., Diligenti, M., Marra, G.: Interpretable concept-based memory reasoning. arXiv preprint arXiv:2407.15527 (2024)
4. Havasi, M., Parbhoo, S., Doshi-Velez, F.: Addressing leakage in concept bottleneck models. Advances in Neural Information Processing Systems **35**, 23386–23397 (2022)
5. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural networks **2**(5), 359–366 (1989)
6. Kim, E., Jung, D., Park, S., Kim, S., Yoon, S.: Probabilistic concept bottleneck models. arXiv preprint arXiv:2306.01574 (2023)
7. Koh, P.W., Nguyen, T., Tang, Y.S., Mussmann, S., Pierson, E., Kim, B., Liang, P.: Concept bottleneck models. In: International conference on machine learning. pp. 5338–5348. PMLR (2020)
8. van Krieken, E., Acar, E., van Harmelen, F.: Analyzing differentiable fuzzy logic operators. Artificial Intelligence **302**, 103602 (2022)
9. Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., De Raedt, L.: Deepproblog: Neural probabilistic logic programming. Advances in neural information processing systems **31** (2018)
10. Oikarinen, T., Das, S., Nguyen, L.M., Weng, T.W.: Label-free concept bottleneck models. arXiv preprint arXiv:2304.06129 (2023)
11. Pittino, F., Dimitrievska, V., Heer, R.: Hierarchical concept bottleneck models for vision and their application to explainable fine classification and tracking. Engineering Applications of Artificial Intelligence **118**, 105674 (2023)
12. Poeta, E., Ciravegna, G., Pastor, E., Cerquitelli, T., Baralis, E.: Concept-based explainable artificial intelligence: A survey. arXiv preprint arXiv:2312.12936 (2023)
13. Sawada, Y., Nakamura, K.: Concept bottleneck model with additional unsupervised concepts. IEEE Access **10**, 41758–41765 (2022)
14. Xu, X., Qin, Y., Mi, L., Wang, H., Li, X.: Energy-based concept bottleneck models: unifying prediction, concept intervention, and conditional interpretations. arXiv preprint arXiv:2401.14142 (2024)
15. Zarlenga, M.E., Barbiero, P., Ciravegna, G., Marra, G., Giannini, F., Diligenti, M., Precioso, F., Melacci, S., Weller, A., Lio, P., et al.: Concept embedding models. In: NeurIPS 2022-36th Conference on Neural Information Processing Systems (2022)
16. Zhang, Y., Tiňo, P., Leonardis, A., Tang, K.: A survey on neural network interpretability. IEEE Transactions on Emerging Topics in Computational Intelligence **5**(5), 726–742 (2021)