CS 957, System-2 AI
Neuro-Symbolic AI

Mohammad Hossein Rohban | Feb 2025

Sharif University of Technology

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]

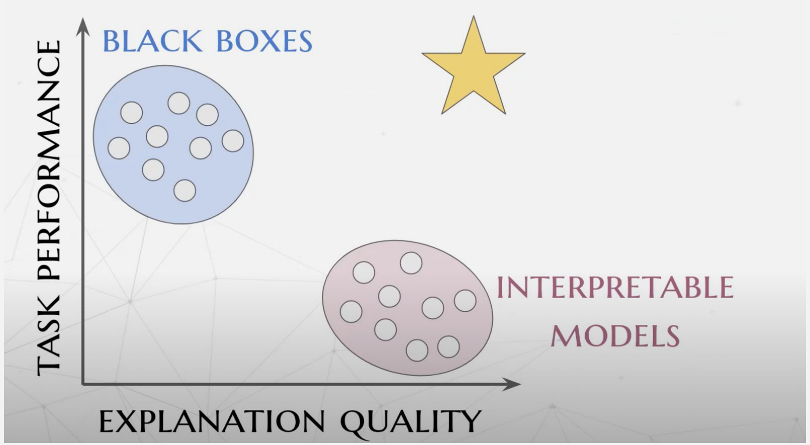
Motivation

Given an input to be classified.

Deep models are accurate but are blackbox and are not interpretable.

Inherently interpretable models are interpretable but are not as accurate.

Can we have both?



Courtesy: Author's slides

Motivation (cont.)

Want to build an

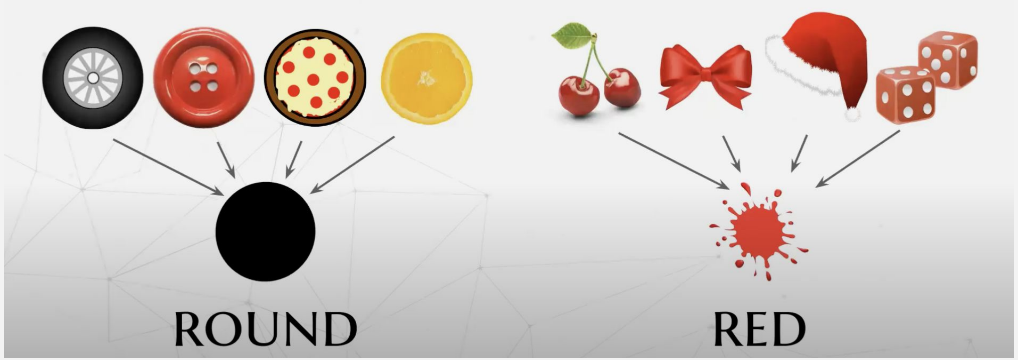
- inherently interpretable
- accurate
- robust

model.

Concepts

Each input can be decomposed into a set of concepts that together characterize the input semantic.

apple = red + round



Courtesy: Author’s slides

Initial Idea

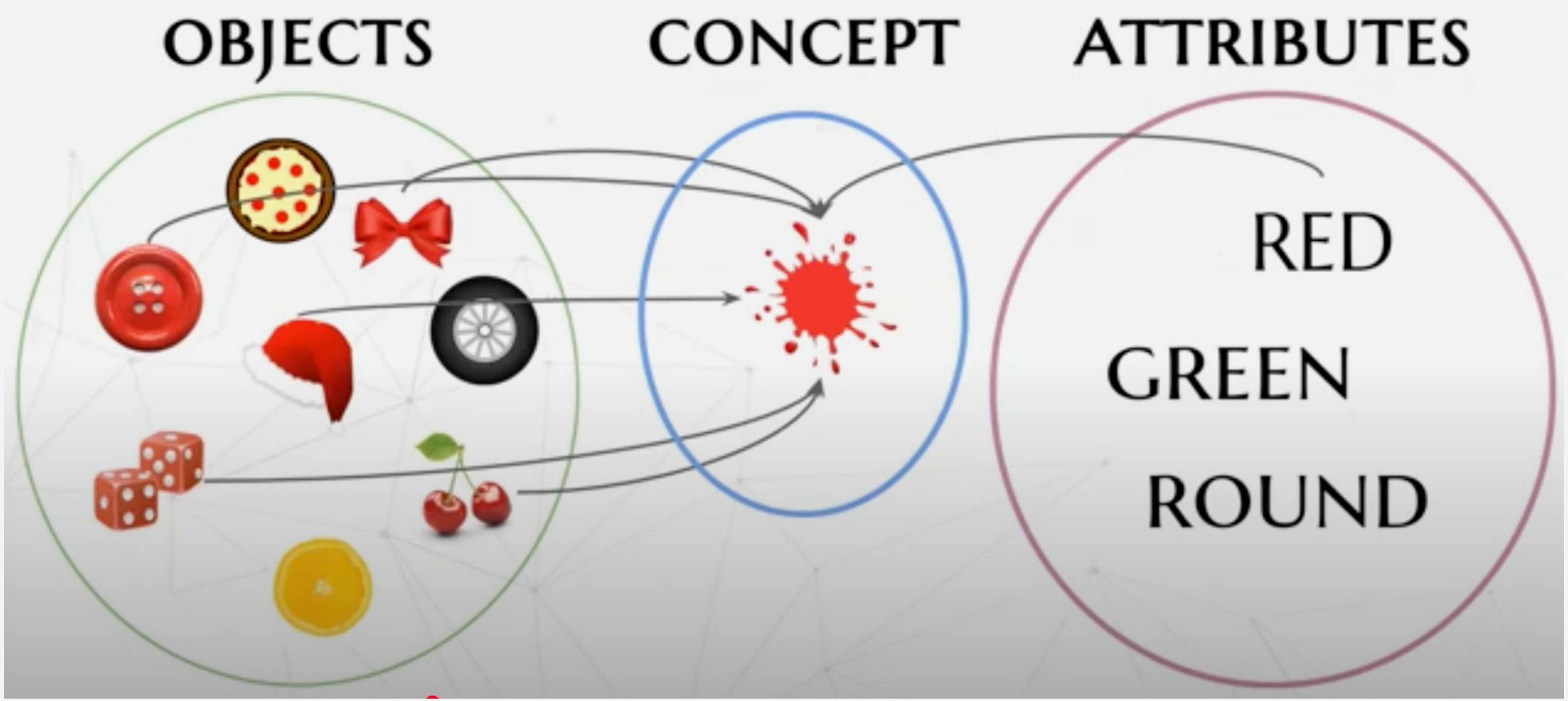
Describe each class as logical AND of concepts:

$$\text{Apple} = \text{Red} \wedge \text{Round}$$

This is definitely interpretable, but a whole lot of questions arise:

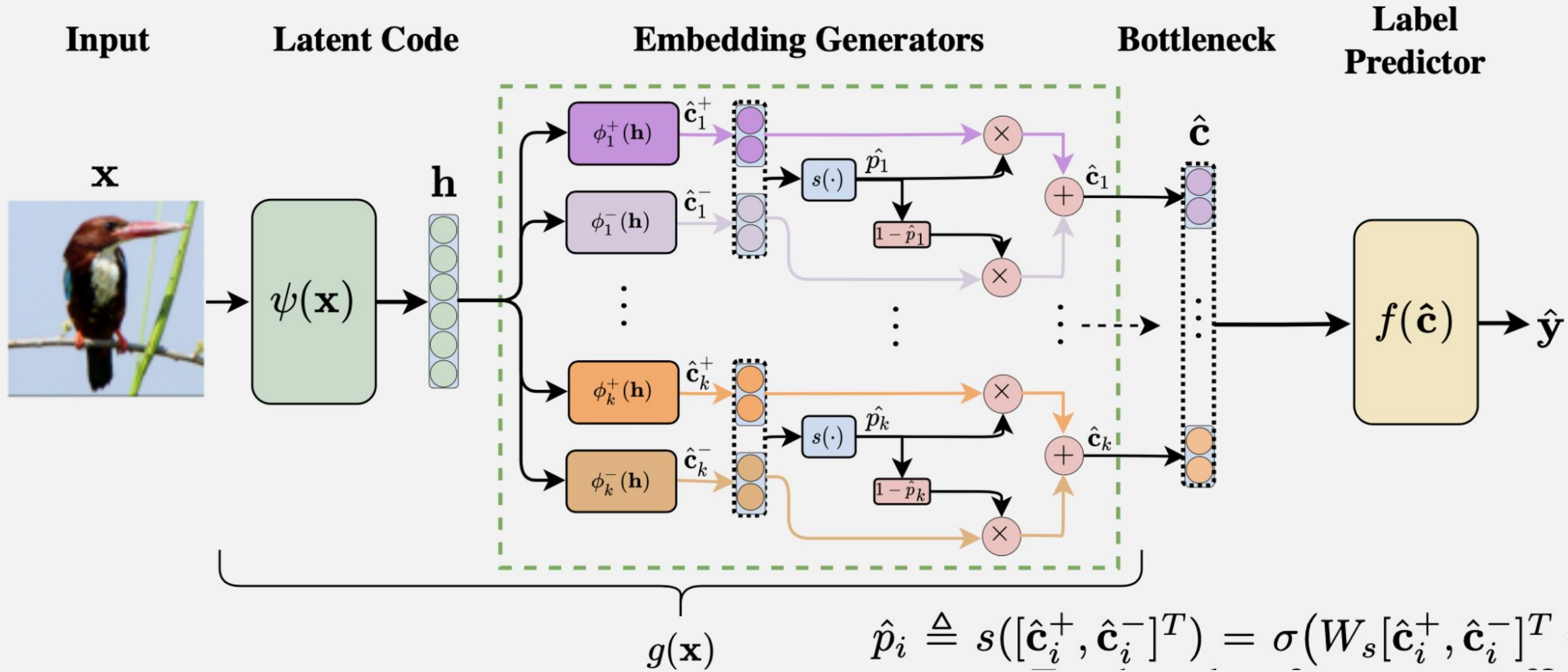
- Where do concepts come from?
- Where do logical rules come from?
- Crisp rules tend to result in inaccurate classification.

But what are concepts?



Courtesy: Author’s slides

We can try to learn them ...



$$\hat{p}_i \triangleq s([\hat{\mathbf{c}}_i^+, \hat{\mathbf{c}}_i^-]^T) = \sigma\left(W_s[\hat{\mathbf{c}}_i^+, \hat{\mathbf{c}}_i^-]^T + \mathbf{b}_s\right)$$

$$\mathcal{L} \triangleq \mathbb{E}_{(\mathbf{x},y,\mathbf{c})}\left[\mathcal{L}_{\text{task}}\Big(y, f\big(g(\mathbf{x})\big)\Big) + \alpha\mathcal{L}_{\text{CrossEntr}}\Big(\mathbf{c}, \hat{\mathbf{p}}(\mathbf{x})\Big)\right]$$

Courtesy: M. Zarlenga et al, “Concept Embedding Models: Beyond the Accuracy-Explainability Trade-Off,” NeurIPS 2022.

We we have up to so far?

For each concept i, an embedding vector $\mathbf{c}_i$.

A deep model that takes as input x, and outputs existence, in form of confidence, of each concept i.

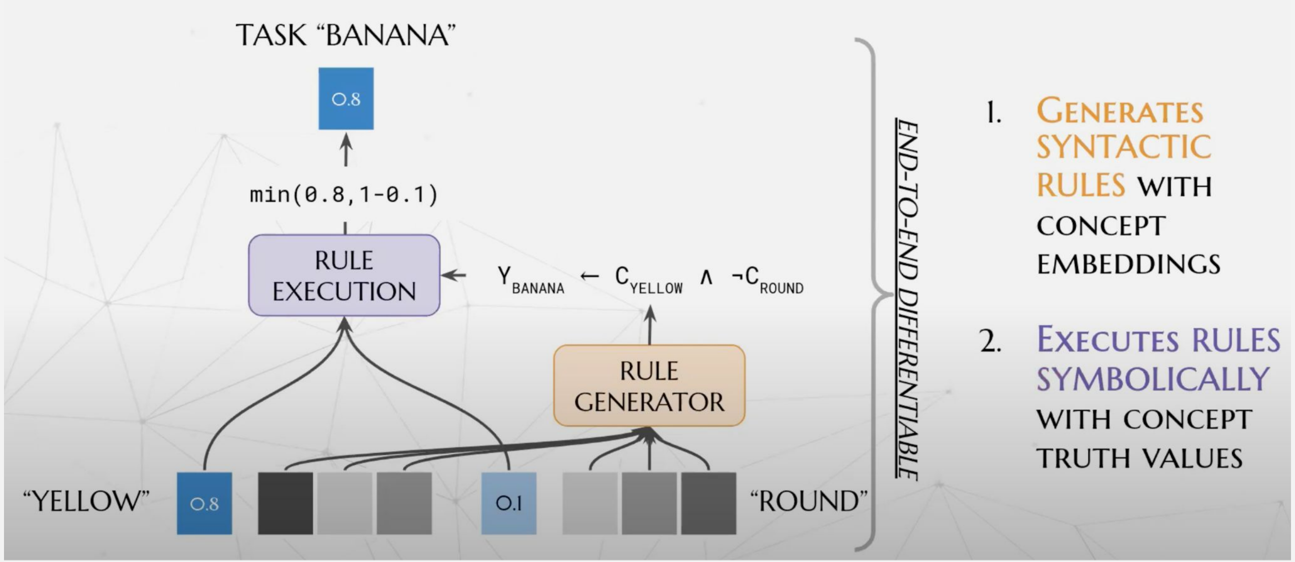
Let’s call it g(x).

How to form rules then?

Rule Learning

Use a neural network to output a rule.

Then, rules are executed.

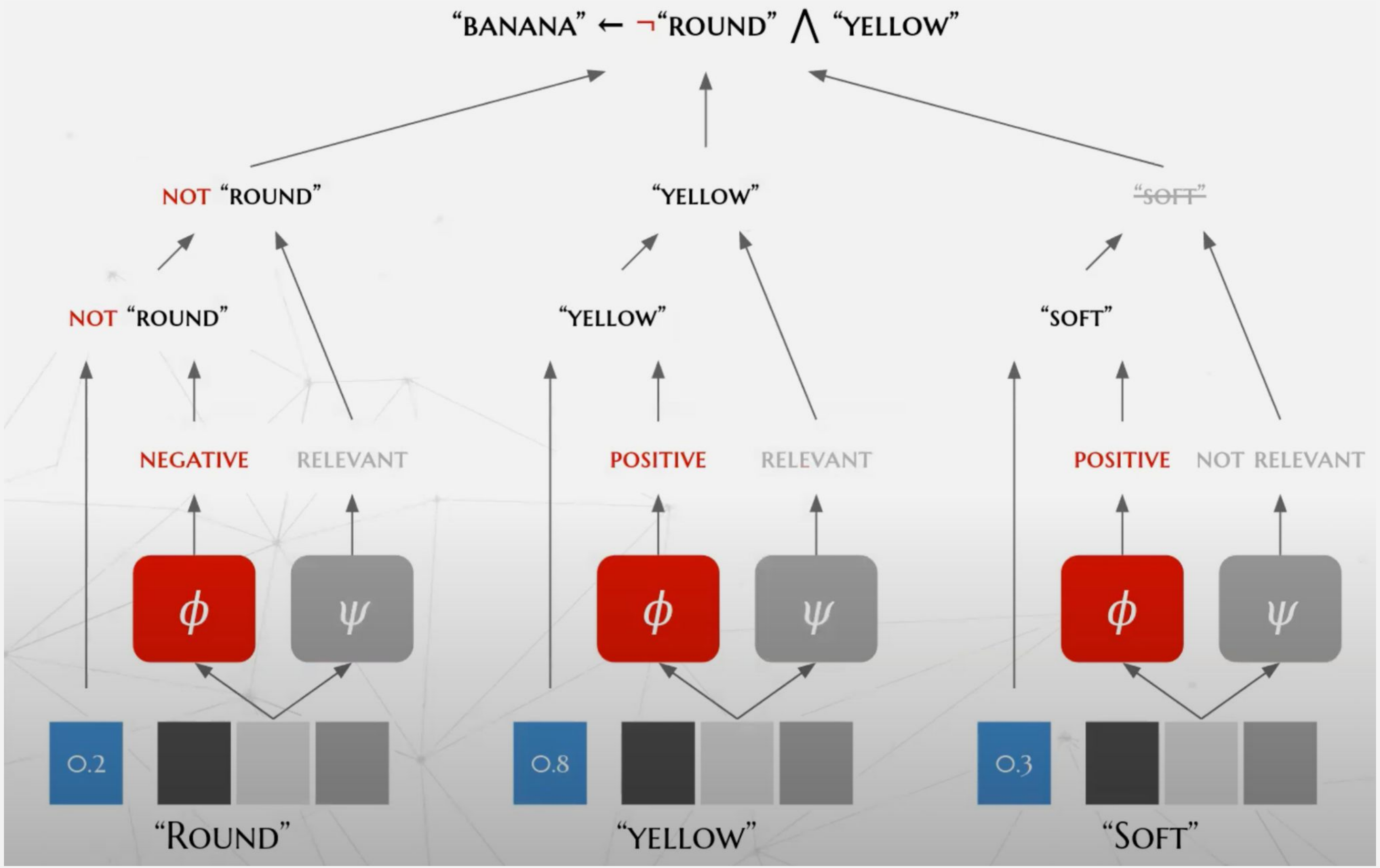


Courtesy: Author's slides

Rule Learning (cont.)

Courtesy: Author's slides

Rule Learning (cont.)



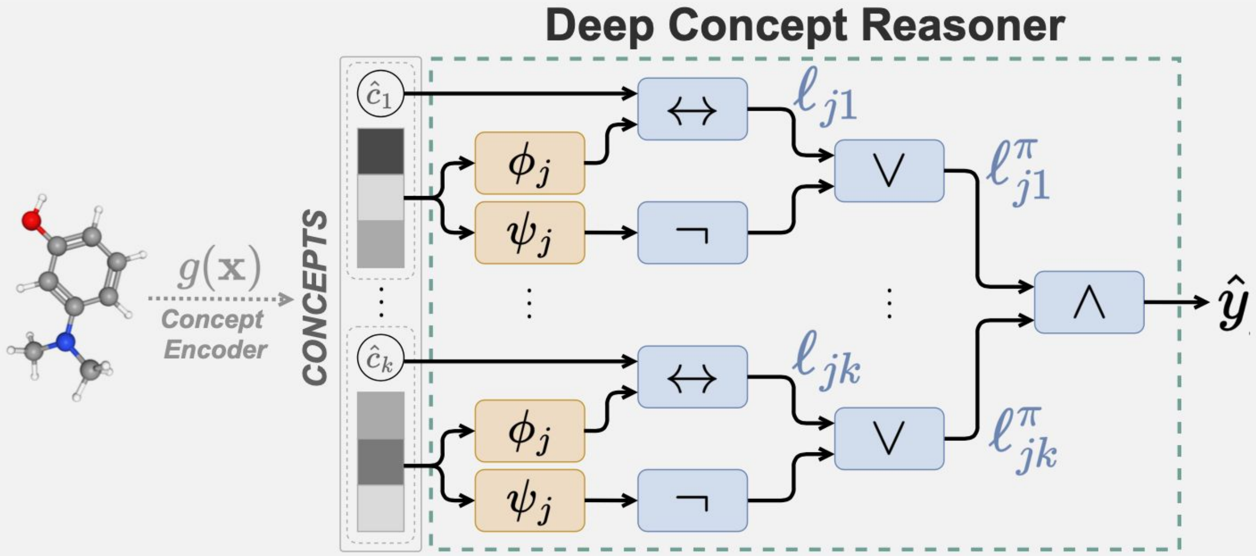
Courtesy: Author's slides

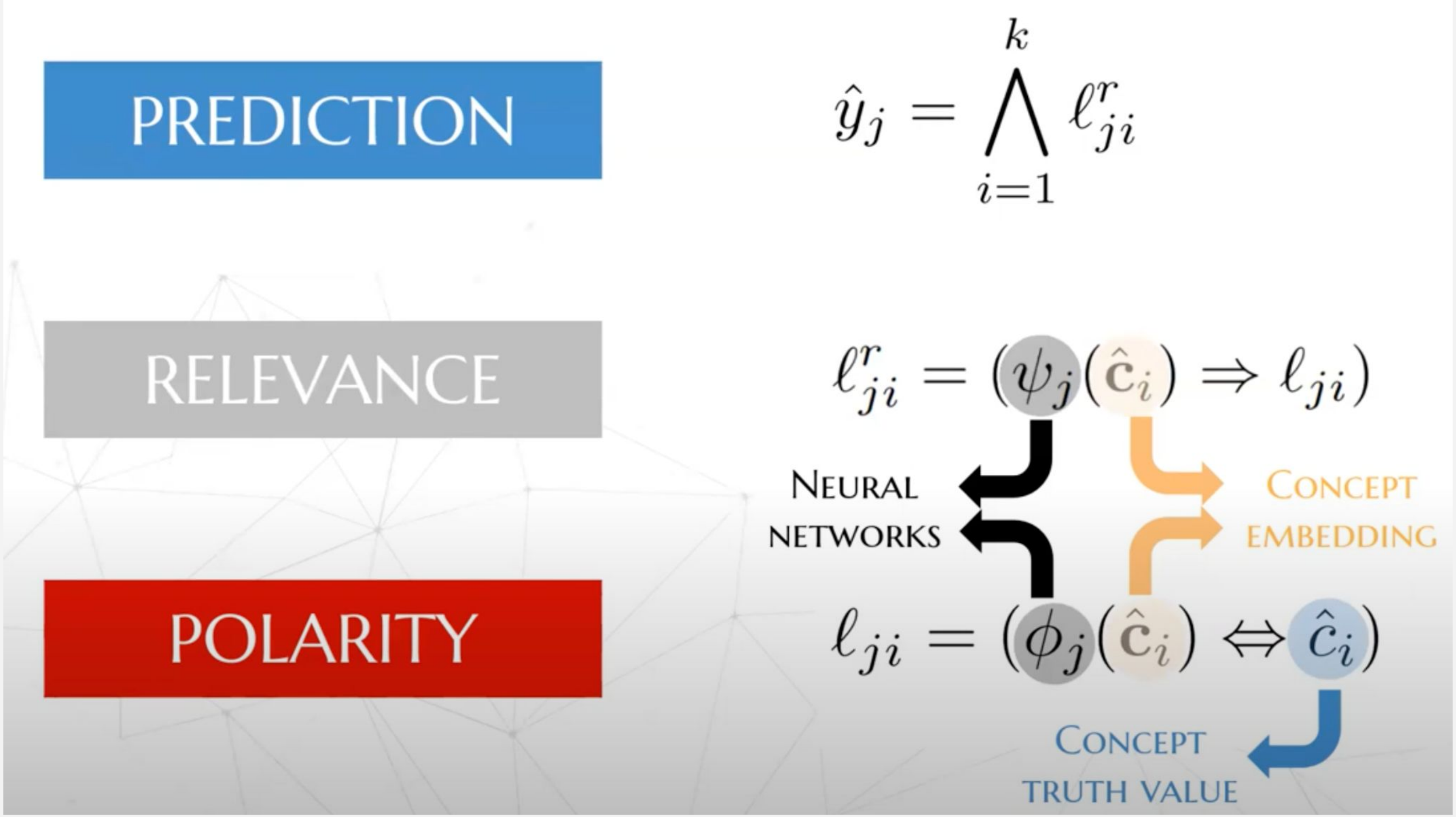
Rule Learning (cont.)

Literal l_{ij}: for a given x, concept i status is in favor of class j: $\ell_{ji} = (\phi_j(\hat{\mathbf{c}}_i) \Leftrightarrow \hat{c}_i)$

Literal l^r_{ij}: if concept i is relevant to class j, then l_{ij}, otherwise 1.

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





Courtesy: Author's slides

Making it differentiable

Make discontinuous logical binary operators continuous.

Fuzzy logic: each statement has a degree of truth

Logical operators on statements are functions of their truth values.

op \ t-norm	Product	Lukasiewicz	Gödel
$x \wedge y$	$x \cdot y$	$\max(0, x + y - 1)$	$\min(x, y)$
$x \vee y$	$x + y - x \cdot y$	$\min(1, x + y)$	$\max(x, y)$
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \Rightarrow y$	$x \leq y ? 1 : \frac{y}{x}$	$\min(1, 1 - x + y)$	$x \leq y ? 1 : y$

Differentiable Expression

Adopting Godel's t-norm:

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

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

Making rules parsimonious

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

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

DCR outperforms interpretable models

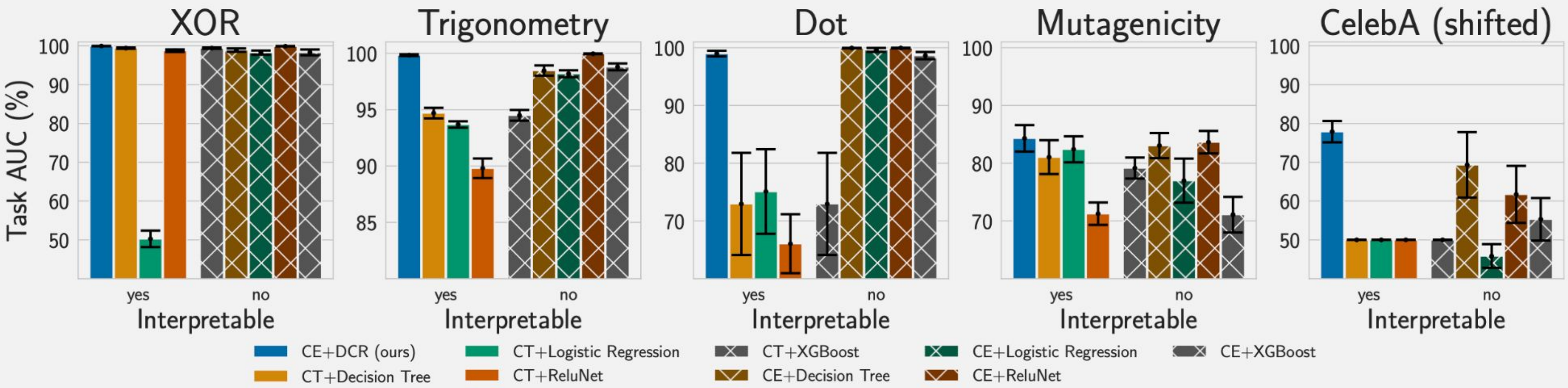


Figure 3. Mean ROC AUC for task predictions for all baselines across all tasks (the higher the better). DCR often outperforms interpretable concept-based models. *CE* stands for concept embeddings, while *CT* for concept truth degrees. Models trained on concept embeddings are not interpretable as concept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

DCR matches the accuracy of neural-symbolic systems trained using human rules

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

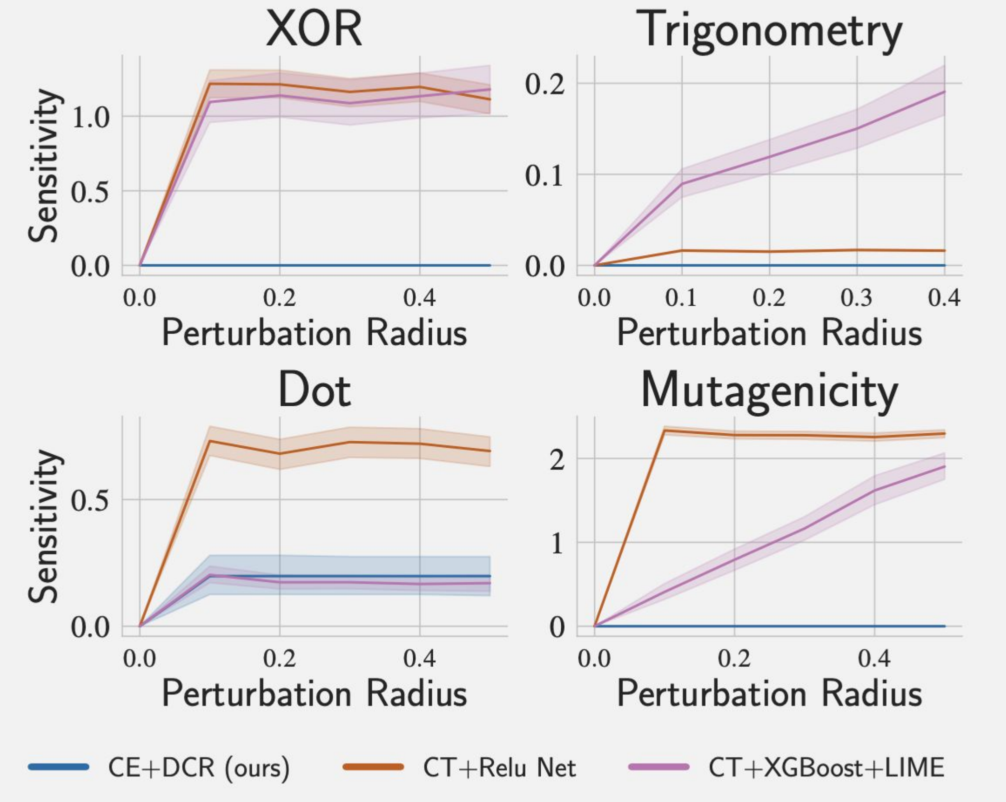
MODEL	ACCURACY (%)
With ground truth rules	
DeepProbLog	97.2 ± 0.5
DeepStochLog	97.9 ± 0.1
Embed2Sym	97.7 ± 0.1
LTN	98.0 ± 0.1
Without ground truth rules	
DCR(ours)	97.4 ± 0.2

DCR discovers semantically meaningful logic rules

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

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

DCR rules are stable under small perturbations



DCR explains prediction mistakes

Table 3. DCR explains prediction errors.

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

Counterfactual Examples

- generate counter-examples as close as possible to the original sample $|x - x^\star| < \varepsilon$
- first rank the concepts present in the rule according to their relevance scores
-

DCR enables discovering counterfactual examples

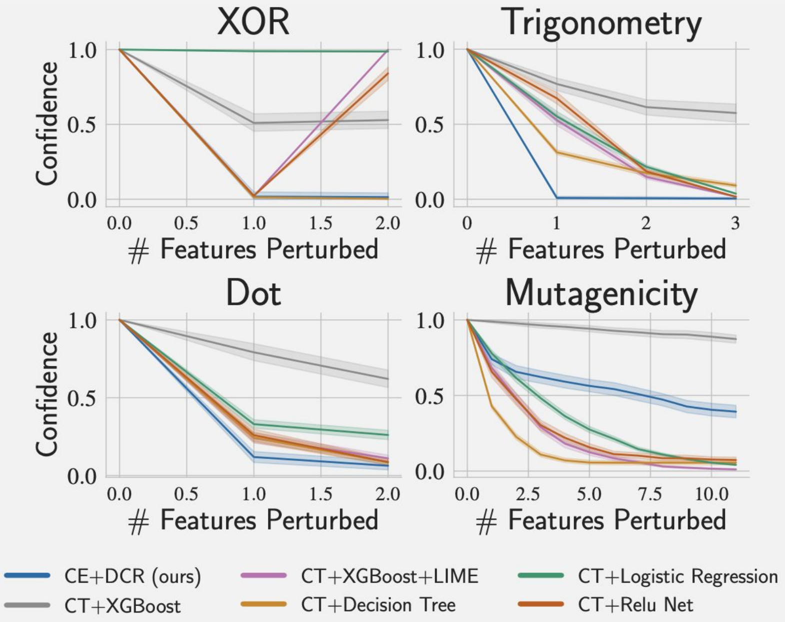


Figure 5. Model confidence as a function of the number of perturbed features on counterfactual examples. The lower, the better. Similarly to interpretable methods, DCR prediction confidence quickly drops after inverting the truth degree of a small set of relevant concepts, facilitating the discovery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

Neuro-Symbolic Architecture

This method was based on Neural$_{\text{Symbolic}}$.

This uses a neural net that is generated from symbolic rules. An example is the Neural Theorem Prover, which constructs a neural network from an AND-OR proof tree generated from knowledge base rules and terms. Logic Tensor Networks also fall into this category [Wikipedia].

Let's Discuss Limitations of this work and how to improve it

Brainstorm