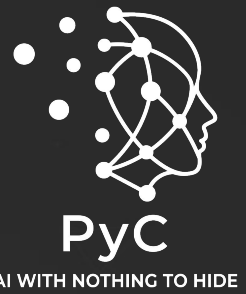
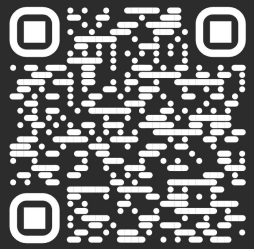
Foundations of Concept-Based Interpretable Deep Learning



Giuseppe Marra & Pietro Barbiero
giuseppe.marra@kuleuven.be pietro.barbiero@ibm.com

In collaboration with: Giovanni de Felice and Mateo Espinosa Zarlenga

ESSAi26



KU LEUVEN IBM Research

fwo Swiss National Science Foundation HASLERSTIFTUNG

Course Outline

I. Interpretability theory

II. Blueprint for interpretable models

+ interpretability in PyTorch

III. Concept representations

IV. Concept-based predictors

Mon 2:00 pm – 3:30 pm

Tue 2:00 pm – 3:30 pm

Course Outline

I. Interpretability theory

II. Blueprint for interpretable models

+ interpretability in PyTorch

III. Concept representations

IV. Concept-based predictors

Wed 2:00 pm – 3:30 pm

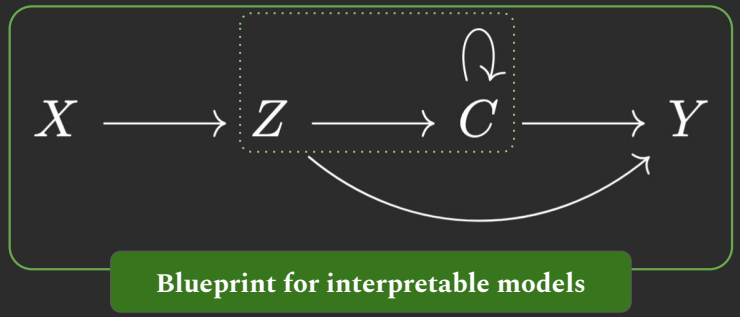
Fri 2:00 pm – 3:30 pm

Part III: Concept Representations

includes slides from: Mateo Espinosa Zarlenga, Luc De Raedt

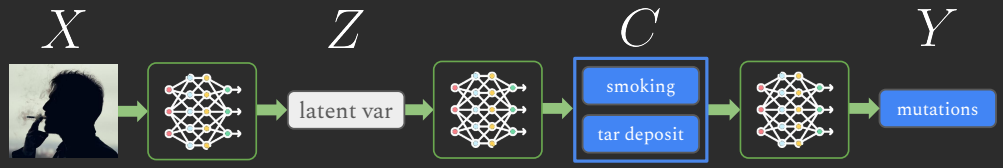
Recap: The Standard Interpretable Model

Recap: Blueprint for interpretable models

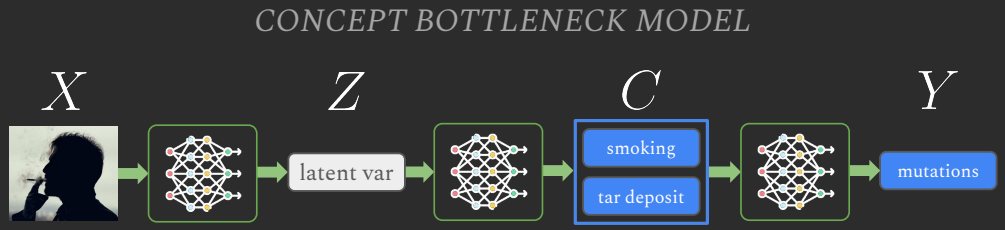



```python
embedding_encoder = torch.nn.Sequential(
    torch.nn.Linear(n_features, embedding_dims),
    torch.nn.LeakyReLU(),
)
c_encoder = pyc.nn.LinearEmbeddingToConcept(
    in_embeddings=embedding_dims,
    out_concepts=annotations
)
y_predictor = pyc.nn.LinearConceptToConcept(
    in_concepts=concept_dims,
    out_concepts=task_dims
)
model = torch.nn.ModuleDict({
    "embedding_encoder": embedding_encoder,
    "concept_encoder": c_encoder,
    "task_predictor": y_predictor
})
embeddings = model["embedding_encoder"](x_train)
c_pred = model["concept_encoder"](embeddings)
y_pred = model["task_predictor"](c_pred)
```

CONCEPT BOTTLENECK MODEL



Recap: Blueprint for interpretable models



$$\mathcal{D} = \{(x_i, \mathbf{c}_i, \mathbf{y}_i)\}_{i=1}^{N}$$

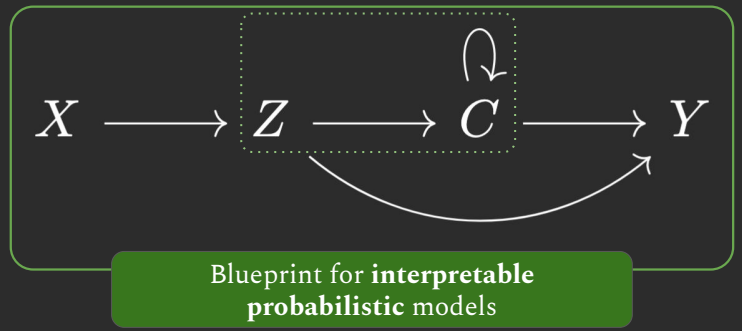
$$\mathcal{L}_{\text{concept}} = -\sum_{i=1}^{N}\sum_{j\in\mathcal{C}} [c_{i,j}\log\hat{c}_{i,j} + (1-c_{i,j})\log(1-\hat{c}_{i,j})]$$

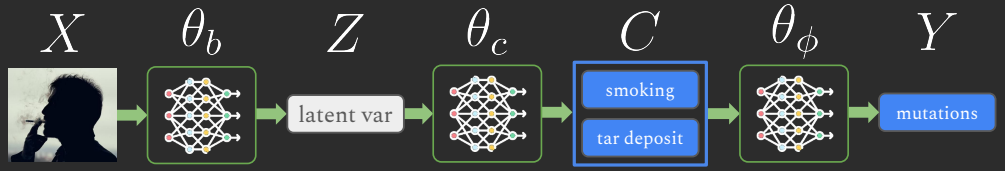
$$\mathcal{L}_{\text{task}} = -\sum_{i=1}^{N}\sum_{k\in\mathcal{Y}} [y_{i,k}\log\hat{y}_{i,k} + (1-y_{i,k})\log(1-\hat{y}_{i,k})]$$

```python
# generate concept and task predictions
latent = model["latent_encoder"](x_train)
c_pred = model["concept_encoder"](embeddings=latent)
y_pred = model["task_predictor"](concepts=c_pred)

# compute loss
concept_loss = loss_fn(c_pred, c_train)
task_loss = loss_fn(y_pred, y_train)
loss = concept_loss + concept_reg * task_loss

loss.backward()
optimizer.step()
```

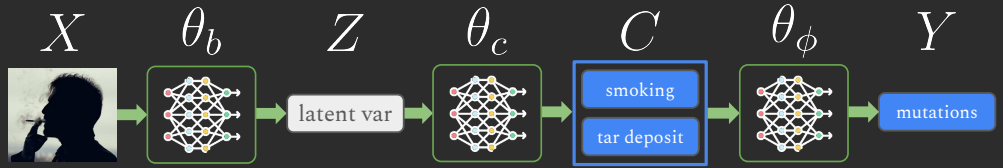
Recap: Interpretable probabilistic models

PROBABILISTIC CONCEPT BOTTLENECK MODEL

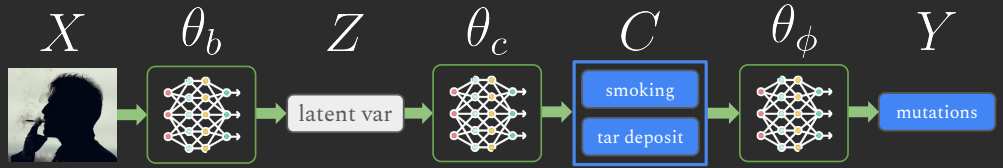




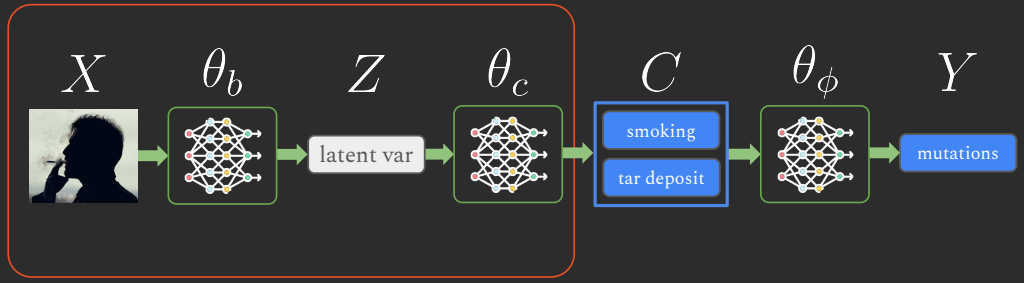
$$p(y, c \mid x; \theta_\phi, \theta_c, \theta_b) = \int p(y \mid c; \theta_\phi) p(c \mid z; \theta_c) p(z \mid x; \theta_b) \mathrm{d}z$$



The bottleneck derives from our assumption about the target user of the machine learning model,

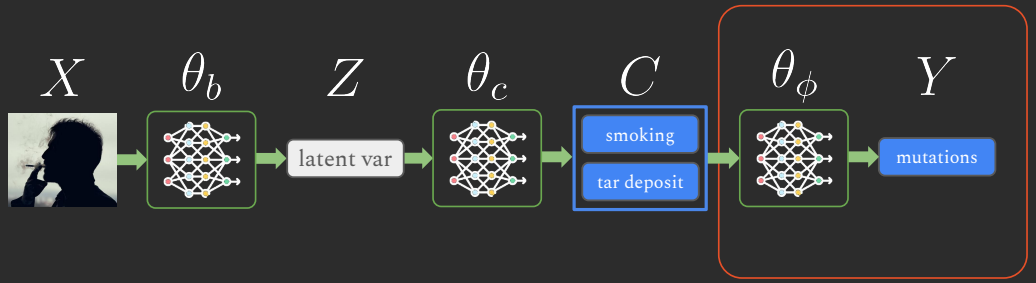


But concepts are actually more than just a bottleneck
The are a **semantic API** consumed by (at least) three actors:



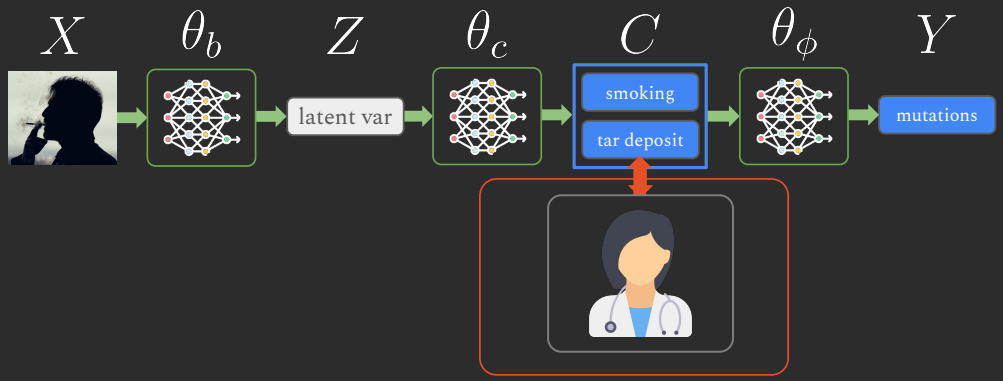
But concepts are actually more than just a bottleneck
The are a **semantic API** consumed by (at least) three actors:

- a machine learning encoder model



But concepts are actually more than just a bottleneck
The are a **semantic API** consumed by (at least) three actors:

- a machine learning encoder model
- a task predictor



But concepts are actually more than just a bottleneck
The are a **semantic API** consumed by (at least) three actors:

- a machine learning encoder model
- a task predictor
- a target user

This semantic interface needs a contract

- How do define the interface such that these three actors communicate efficiently?

- Natural language is appealing, but it weakens our control over the interaction
  - a key reason for using interpretable models.
  - but see [1]

- **Formal languages,** e.g. logic, form a middle ground:
  - precise enough for machines to manipulate
  - still close enough to human meaning

[1] Yamaguchi, Shin'ya, and Kosuke Nishida. "Explanation bottleneck models." AAAI 2025

Assumption: bounded and formal users

Bounded formal user

apple

red

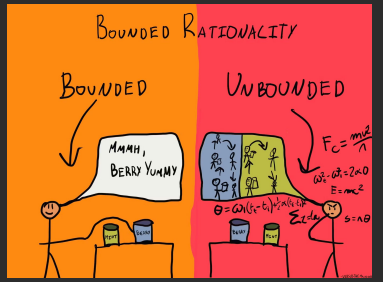
gray

Fixed vocabulary of symbols (syntax)

$$c^{[h]}_{\texttt{red}}(🍎) = 0.9$$

$$c^{[h]}_{\texttt{red}}(🍋) = 0.2$$

Mechanism to assign meaning to symbols (semantics)

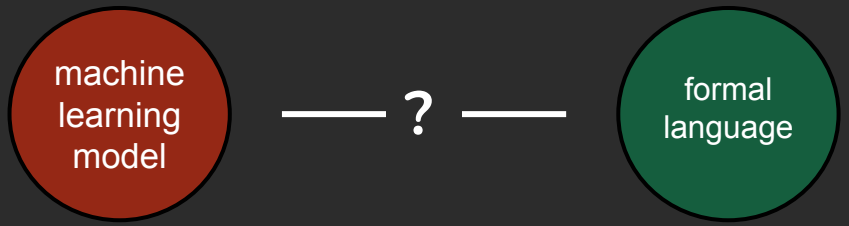


Time / computational limits (bounded rationality)

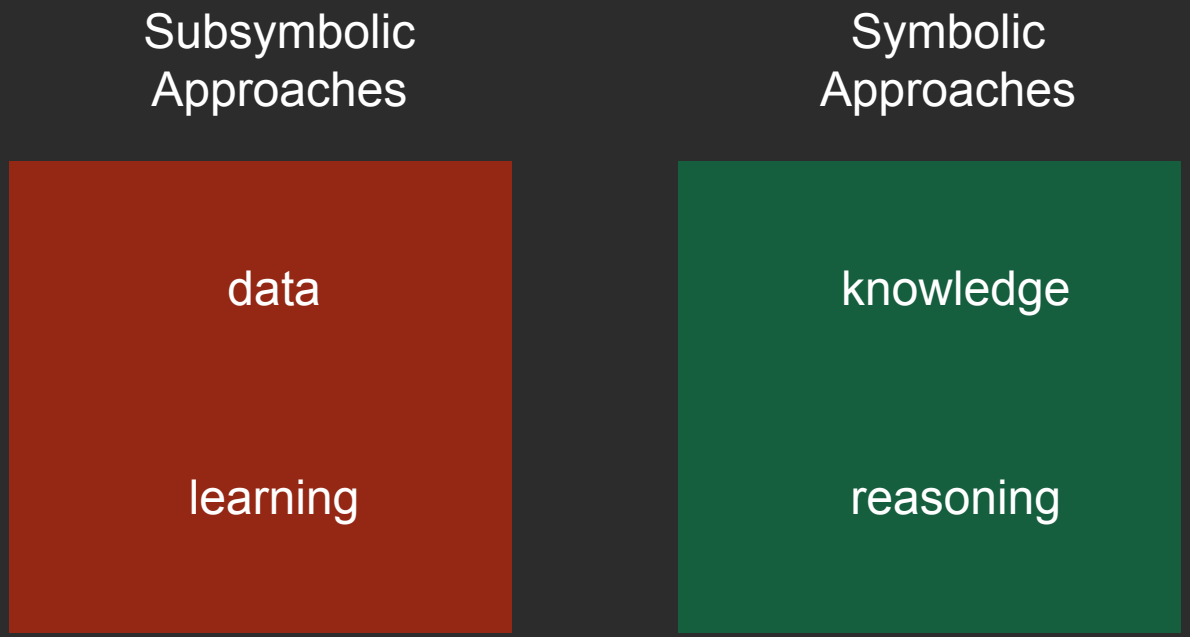
Alfred Tarski

Herbert Simon

A central question



The neurosymbolic integration quest



Learning and Reasoning

Learning:

Neural/statistical generalization from data

Learning: adapting neural networks to map inputs to useful outputs (including large language models and broader foundation models)

Examples:

- images -> labels
- text → embeddings
- prompts → generated responses
- data → learned representations
- large-scale corpora → foundation models

Reasoning:

Deriving conclusions from knowledge, constraints, or uncertainty

Reasoning means using some structured mechanism to infer what follows from what we know. This can be logical, probabilistic, or both.

Examples:

- facts + rules → conclusions
- constraints → allowed solutions
- evidence + probabilities → updated beliefs
- formulas/programs → consequences

Learning and Reasoning

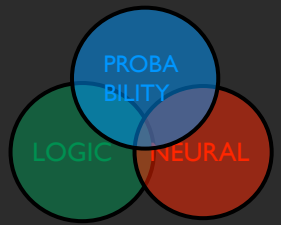
Learning:

Neural/statistical generalization from data

Learning: adapting neural networks to map inputs to useful outputs (including large language models and broader foundation models)

Examples:

- images -> labels
- text → embeddings
- prompts → generated responses
- data → learned representations
- large-scale corpora → foundation models



Reasoning:

Deriving conclusions from knowledge, constraints, or uncertainty

Reasoning means using some structured mechanism to infer what follows from what we know. This can be logical, probabilistic, or both.

Examples:

- facts + rules → conclusions
- constraints → allowed solutions
- evidence + probabilities → updated beliefs
- formulas/programs → consequences

Learning:

Neural/statistical generalization from data

NEURAL

Reasoning:

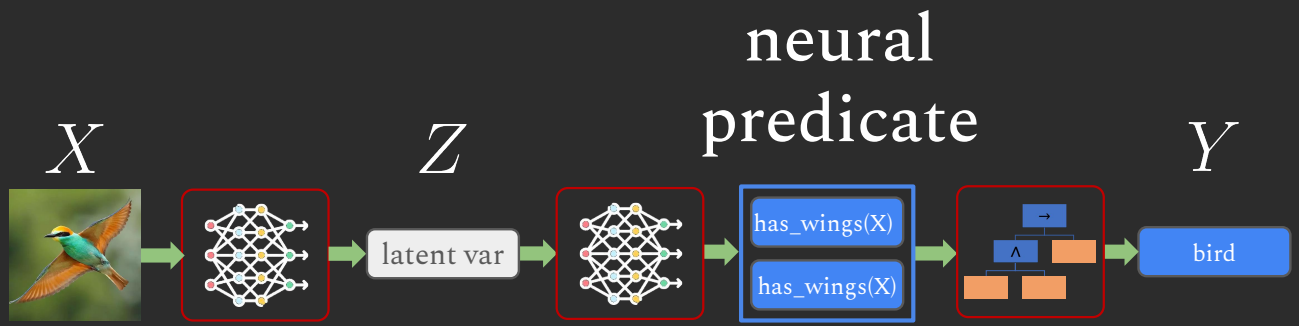
Deriving conclusions from knowledge, constraints, or uncertainty

0.8::has_wings(x) ∧ has_beak(x) → bird(x)

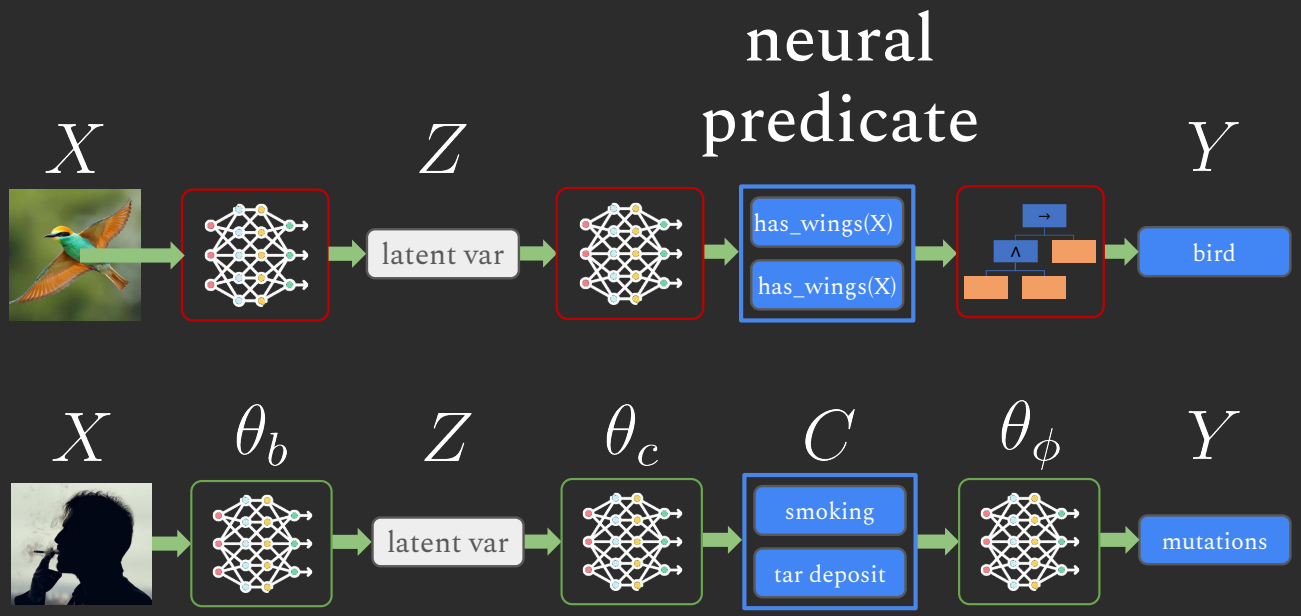
LOGIC PROBABILITY

raw input $\xrightarrow{?}$ symbolic predicates

The neural predicate



The neural predicate



Learning:

Neural/statistical generalization from data

NEURAL

Reasoning:

Deriving conclusions from knowledge, constraints, or uncertainty

0.8::has_wings(x) ∧ has_beak(x) → bird(x)

LOGIC PROBABILITY

raw input $\xrightarrow{?}$ symbolic predicates

Same big challenge that CBM

The formal language contract

We need to specify:

1) Syntax:
    a) Vocabulary of symbols (predicates and entities)
    b) The arities of the predicates

e.g. predicates = {sunny, tall, funny, fatherOf}
entities = {bart, homer, lisa, ...}
arities: sunny/0, tall/1, funny/1, fatherOf/2

The formal language contract

We need to specify:

1) Syntax
2) Semantics:
    a) what values do we allow?
    b) which values do we assign at each symbol?

e.g. True/False, [0,1], (-∞, +∞), $\mathbb{R}^n$

sunny = 0.8

fatherOf(homer,bart) = True

The formal language contract

We need to specify:

1) Syntax
2) Semantics

The neurosymbolic twist: the conditional semantics

is sunny=True in this particular context:

The formal language contract

We need to specify:

1) Syntax
2) Semantics

The neurosymbolic twist: the conditional semantics

is happy(girl)=True in this particular context:
and is happy(dog)=True?

The formal language contract

We need to specify:

1) Syntax
2) Semantics
3) Connectives / Operators / Algebra *(mostly relevant tomorrow)*
    a) how do we compose the symbols to create more complex sentences?

The Concept Bottleneck Model contract

1) Syntax:
    a) Vocabulary of symbols (predicates and entities)
    b) The arities of the predicates
2) Semantics:
    a) what values do we allow?
    b) which values do we assign at each symbol?

The Concept Bottleneck Model contract

1) Syntax:
   a) Vocabulary of symbols (predicates and entities)

      Given and Fixed

   b) The arities of the predicates

      Only properties (0-arity; propositional)
2) Semantics:
   a) what values do we allow?

      Unclear, at the base Boolean

   b) which values do we assign at each symbol?

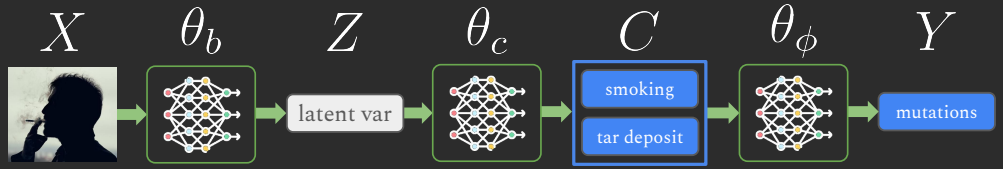
      Fully Supervised

The Concept Bottleneck Model contract

1) Syntax:
   a) Vocabulary of symbols (predicates and entities)

      Given and Fixed

   b) The arities of the predicates
2) Semantics:
   a) what values do we allow?
   b) which values do we assign at each symbol?

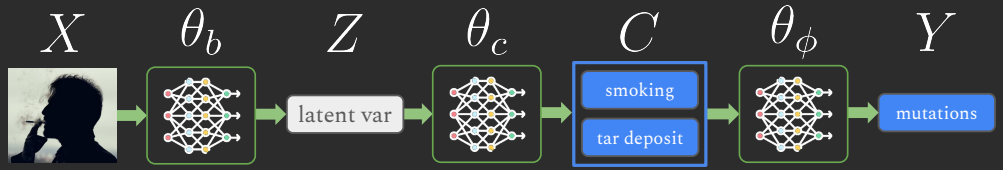


The Concept Bottleneck Model contract

1) Syntax:
   a) Vocabulary of symbols (predicates and entities)

      Given and Fixed

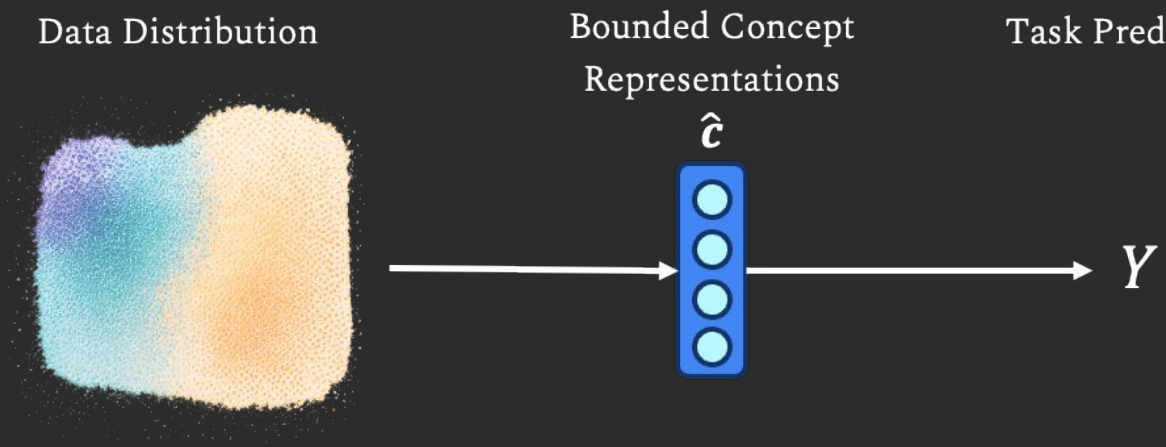
   b) The arities of the predicates
2) Semantics:
   a) what values do we allow?
   b) which values do we assign at each symbol?



Failure: Insufficient vocabulary

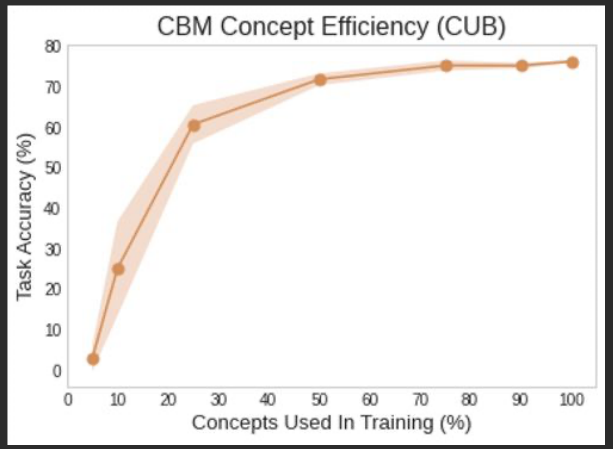
Failure: Incomplete vocabulary

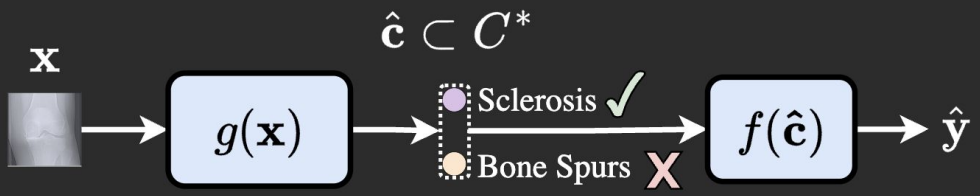
Concepts lead to an information bottleneck in the model if the concept set is incomplete



Failure: Incomplete vocabulary

Concepts lead to an information bottleneck in the model if the concept set is incomplete





Failure: Incomplete vocabulary
Repair: automated concept discovery

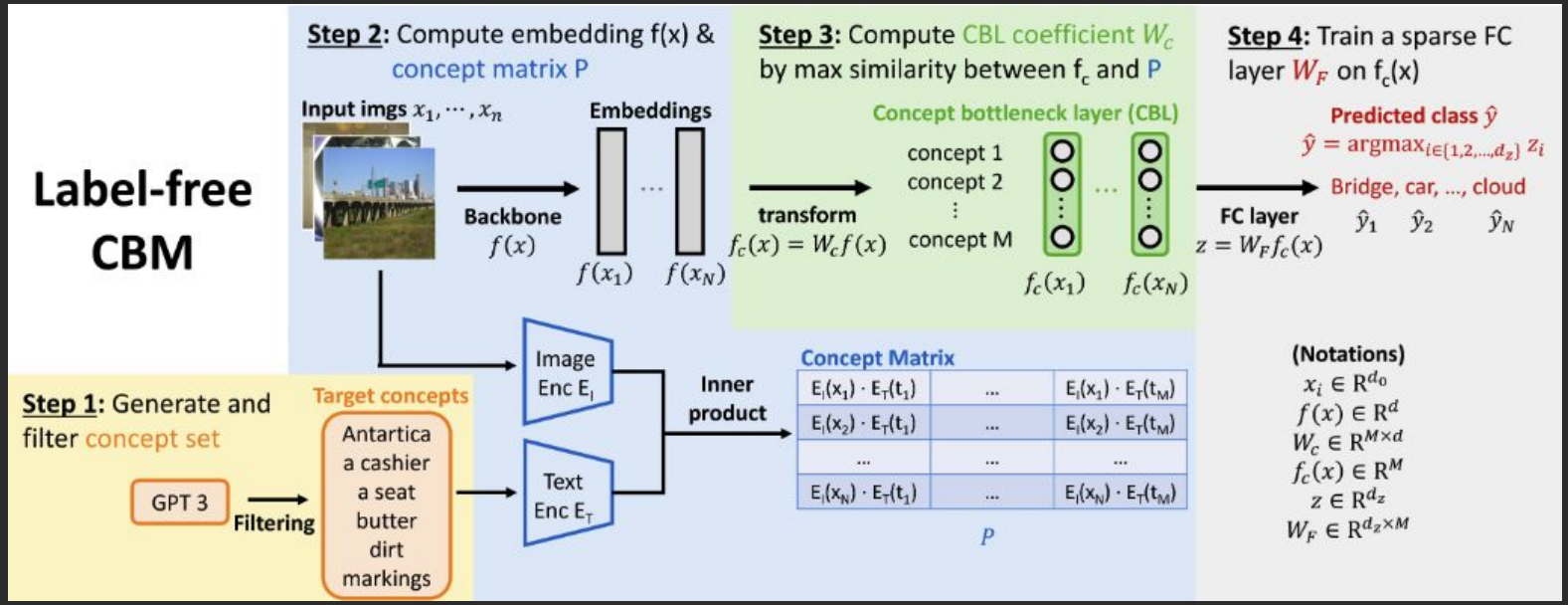
This can be turned into a fully automated pipeline if we exploit language models as knowledge bases for extracting concept descriptions:

"List the most important features for recognizing something as a {class}:"

[1] Oikarinen et al. "Label-Free Concept Bottleneck Models." ICLR (2023).
[2] Yang, Yue, et al. "Language in a bottle: Language model guided concept bottlenecks for interpretable image classification." IEEE/CVF 2023.

Automating concept discovery

Label-free CBMs achieve this by introducing *filtering rules* and a *projection matrix* that produces sparse and discriminative scores aligned to CLIP's similarity scores

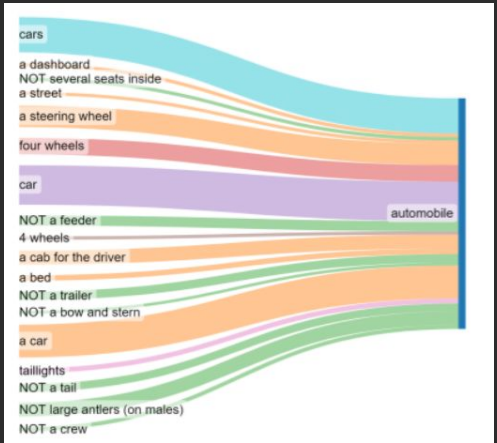


[1] Oikarinen et al. "Label-Free Concept Bottleneck Models." ICLR (2023).

Automating concept discovery

Two ideas:

1. Concept filtering: LLMs generate non-visual, redundant, and ambiguous concepts. Filter them before use.



1. remove concepts that are **too long**
2. remove concepts that are **too similar to the target** class names
3. remove concepts that are **too similar to each other**
4. remove concepts that are **not activated** in the training data.

[1] Oikarinen et al. "Label-Free Concept Bottleneck Models." ICLR (2023).
[2] Yang, Yue, et al. "Language in a bottle: Language model guided concept bottlenecks for interpretable image classification." IEEE/CVF 2023.

Automating concept discovery

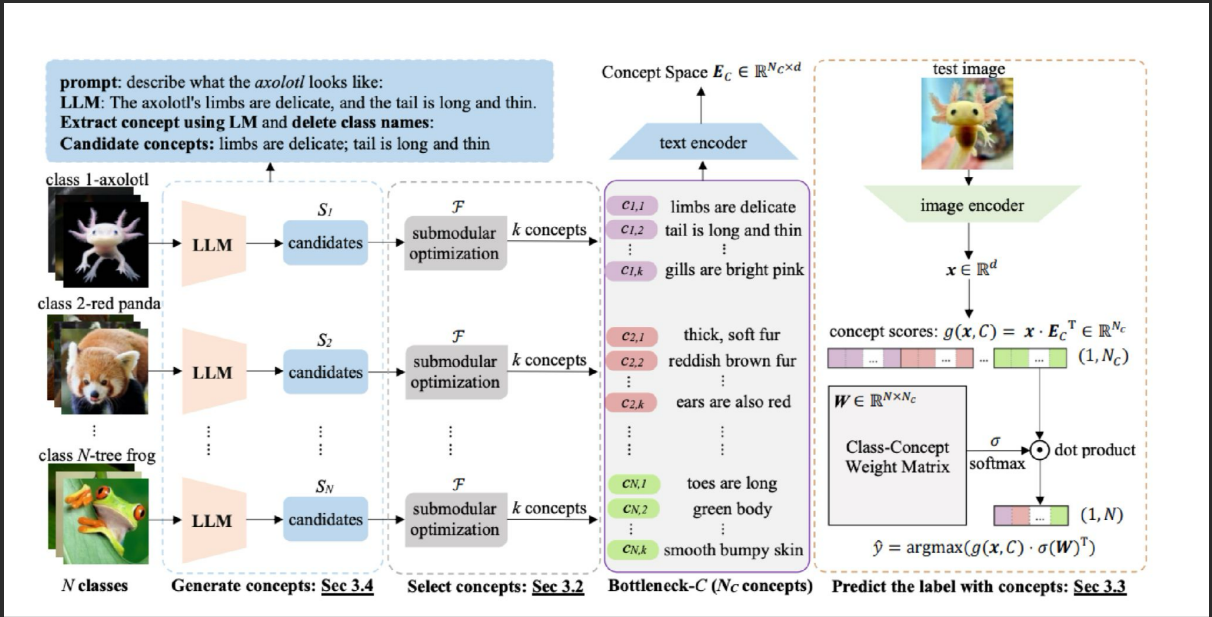
Two ideas:

1. Concept filtering: LLMs generate non-visual, redundant, and ambiguous concepts
2. Discriminative text-image scores: CLIP scores are naturally saturated and therefore are not ideal concept scores on their own. Use them only as supervision signal.

[1] Oikarinen et al. "Label-Free Concept Bottleneck Models." ICLR (2023).
[2] Yang, Yue, et al. "Language in a bottle: Language model guided concept bottlenecks for interpretable image classification." IEEE/CVF 2023.

Automating concept discovery

Language Guided Bottlenecks (LaBo) has a very similar pipeline (same CLIP backbone)



[1] Yang et al. "Language in a bottle: Language model guided concept bottlenecks for interpretable image classification." CVPR (2023).

Automating concept discovery

Alternatively, one may filter the concept bank via submodular optimization:

$$\mathcal{F}(C_y) = \underbrace{\alpha \cdot \sum_{c \in C_y} D(c)}_{\text{discriminability}} + \underbrace{\beta \cdot \sum_{c_1 \in S_y} \max_{c_2 \in C_y} \phi(c_1, c_2)}_{\text{coverage}}$$

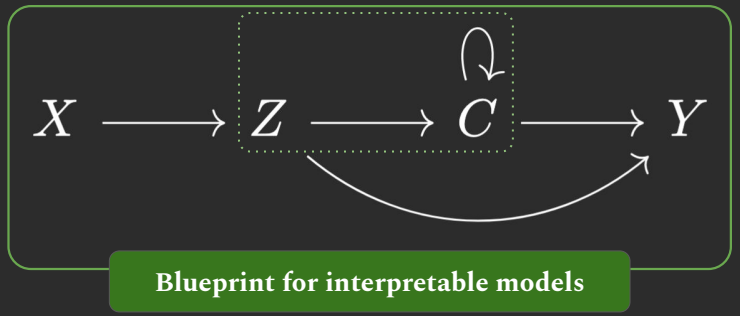
Concepts to select for class y

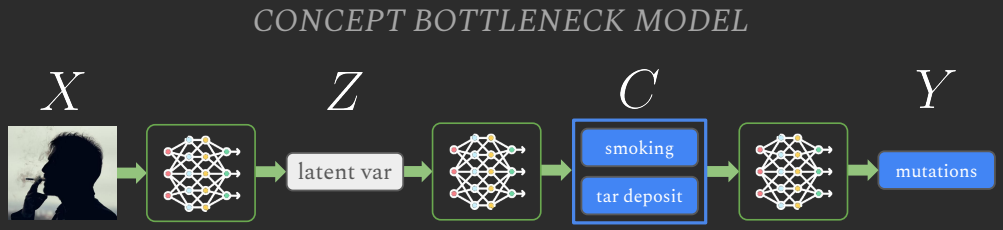
How good is the concept at discriminating only a handful of task labels?

How good does the filtered concept set represent the original set of candidates?

[1] Yang et al. "Language in a bottle: Language model guided concept bottlenecks for interpretable image classification." CVPR (2023).

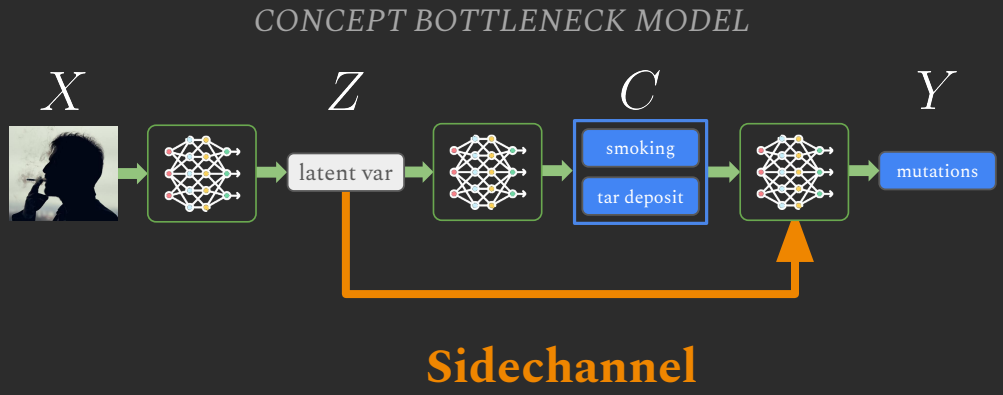
Blueprint for interpretable models



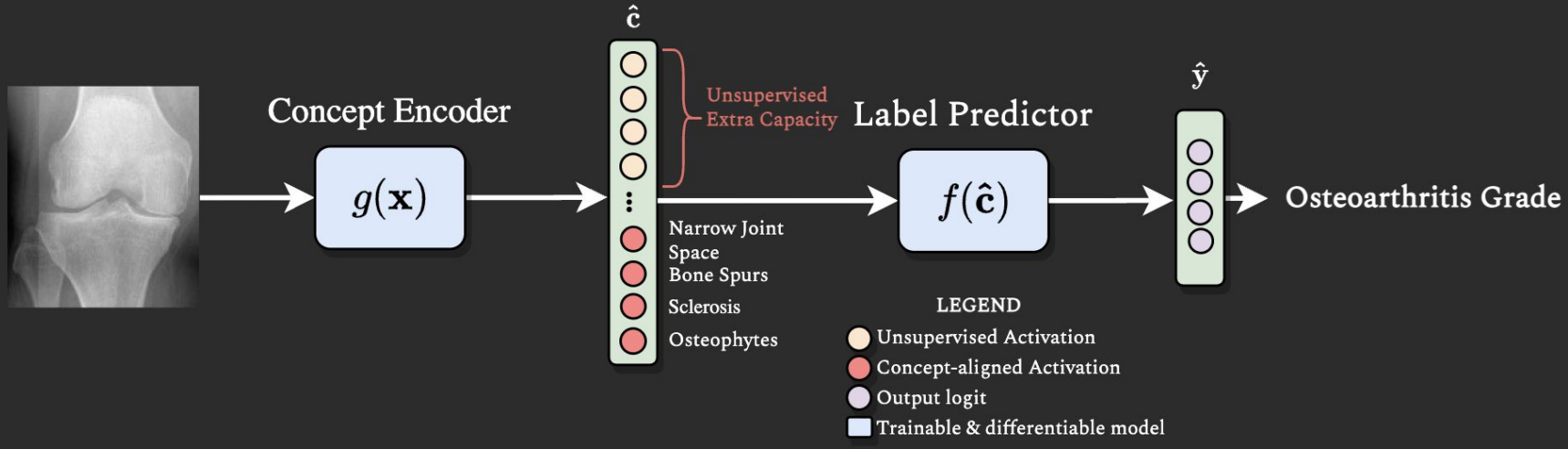


What if do not look for a complete set at all?

Failure: Incomplete vocabulary
Repair: Adding extra (unsupervised) capacity



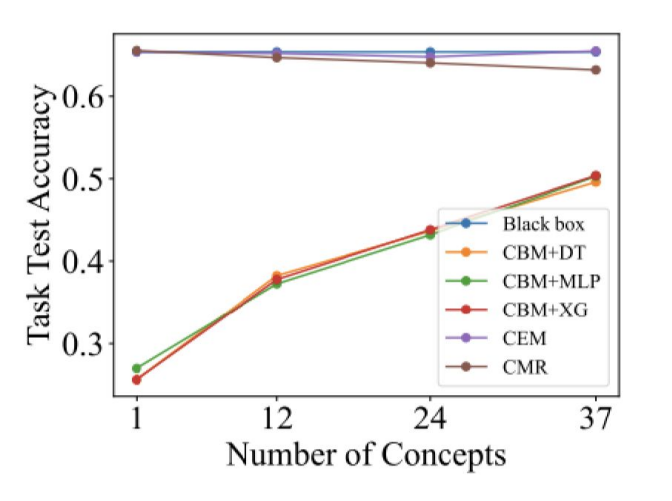
Failure: Incomplete vocabulary
Repair: Adding extra (unsupervised) capacity



A **"Hybrid" CBM** adds extra unsupervised capacity to the CBM's bottleneck

[1] Mahinpei et al. "Promises and pitfalls of black-box concept learning models." arXiv preprint (2021).

Failure: Incomplete vocabulary
Repair: Adding extra (unsupervised) capacity



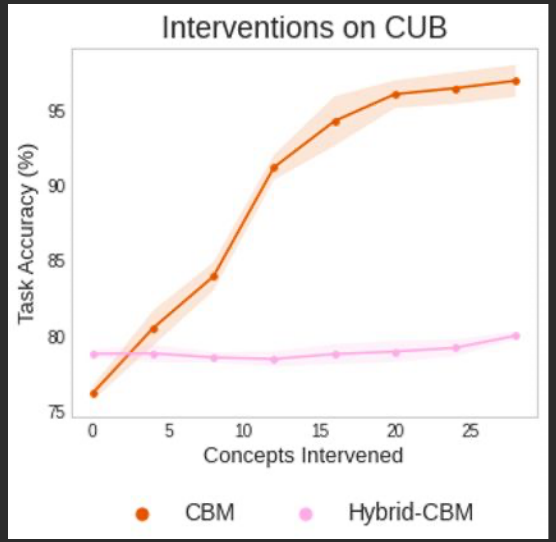
} side-channel models

} bottleneck models

[1] Debot et al; Interpretable Concept-Based Memory Reasoning, NeurIPS 2024

Solution attempt: unsupervised capacity

Problem: intervenability suffers when unsupervised capacity is introduced



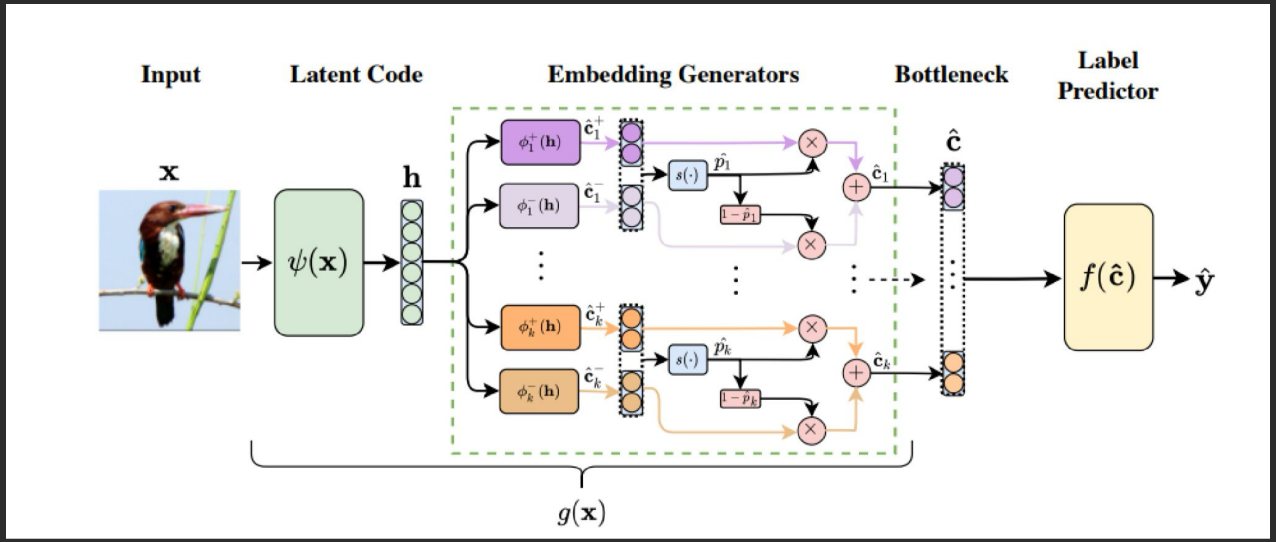
[1] Espinosa Zarlenga, Barbiero, et al. "Concept embedding models: Beyond the accuracy-explainability trade-off." NeurIPS (2022).

Solution idea: structured high-dimensional vectors

Idea: can we use structured extra capacity instead?

Overcoming incompleteness: concept embeddings

For each training concept, let's represent extra structured embedding representations of the concept



[1] Espinosa Zarlenga, Barbiero, et al. "Concept embedding models: Beyond the accuracy-explainability trade-off." NeurIPS (2022).

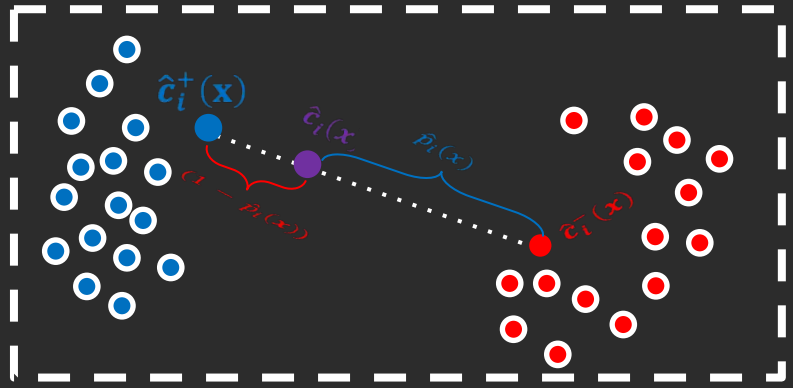
Learning dynamic concept embeddings

For example, we can decompose the concept representation $\hat{\mathbf{c}}_i$ as the mixture of two exogenous representations $\{\hat{\mathbf{c}}_i^+(\mathbf{x}), \hat{\mathbf{c}}_i^-(\mathbf{x})\}$:

$$\hat{\boldsymbol{c}}_i(\boldsymbol{x}) = \hat{p}_i(x)\,\hat{\boldsymbol{c}}_i^+(\boldsymbol{x}) + \big(1 - \hat{p}_i(x)\big)\,\hat{\boldsymbol{c}}_i^-(\boldsymbol{x})$$

"Positive" concept embeddings

"Negative" concept embeddings



[1] Espinosa Zarlenga, Barbiero, et al. "Concept embedding models: Beyond the accuracy-explainability trade-off." NeurIPS (2022).

Learning dynamic concept embeddings

Then, we can compute the probability of a concept being active by learning a simple function of the exogenous vectors:

$$P(C_i = 1|X = \mathbf{x}) = s(\mathbf{x}) := \theta_s^T \begin{bmatrix} \mathbf{c}_i^+(\mathbf{x}) \\ \hat{\mathbf{c}}_i^-(\mathbf{x}) \end{bmatrix} + \mathbf{b}_s$$

[1] Espinosa Zarlenga, Barbiero, et al. "Concept embedding models: Beyond the accuracy-explainability trade-off." NeurIPS (2022).

Intervening on high-dimensional embeddings

We can intervene on high-dimensional representations by changing how we mix the exogenous embeddings:

$$\hat{c}_i := \begin{cases} \hat{c}_i^+(z) \text{ if } c_i = 1 \\ \hat{c}_i^-(z) \text{ otherwise} \end{cases}$$

We can randomly do these interventions at training time to learn more useful representations!

[1] Espinosa Zarlenga, Barbiero, et al. "Concept embedding models: Beyond the accuracy-explainability trade-off." NeurIPS (2022).

Concept Embedding Models (CEMs)

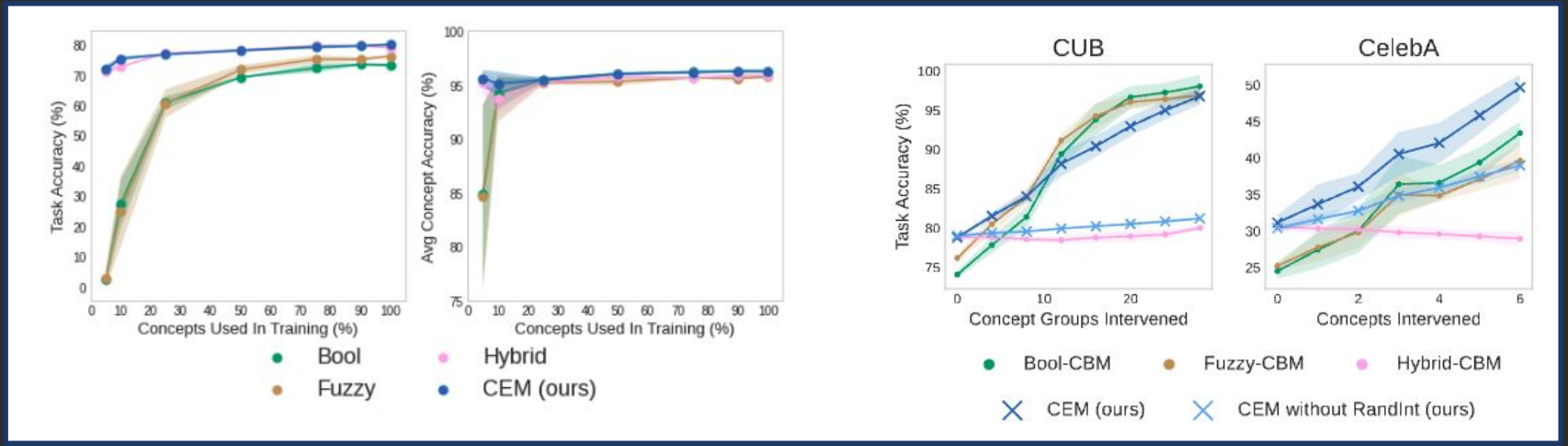
Assuming concept independence, this yields a Concept Embedding Model (CEM):

```python
# Build model using low-level layers
latent_encoder = torch.nn.Sequential(
    torch.nn.Linear(n_features, latent_dims),
    torch.nn.LeakyReLU(),
)
# Exogenous encoder: latent -> per-concept exogenous
exog_encoder = torch.nn.Sequential(
    torch.nn.Linear(latent_dims, exogenous_size*n_concepts),
    torch.nn.Unflatten(dim=1, unflattened_size=(n_concepts, exogenous_size)),
)
# Concept encoder: exogenous -> concepts
c_encoder = pyc.nn.Sequential(
    torch.nn.Linear(exogenous_size, 1),
    torch.nn.Flatten(),
    out_concepts=concept_annotations
)
# Predictor: concepts + exogenous -> tasks
y_predictor = MixConceptEmbeddingToConcept(
    in_concepts=concept_annotations,
    in_embeddings=exogenous_size,
    out_concepts=n_tasks,
)
model = ModuleDict({
    "latent_encoder": latent_encoder,
     "exog_encoder": exog_encoder,
     "concept_encoder": c_encoder,
     "task_predictor": y_predictor
})
```

[1] Espinosa Zarlenga, Barbiero, et al. "Concept embedding models: Beyond the accuracy-explainability trade-off." NeurIPS (2022).

CEMs in Incompleteness

Embeddings enable accurate and highly intervenable* models in incompleteness

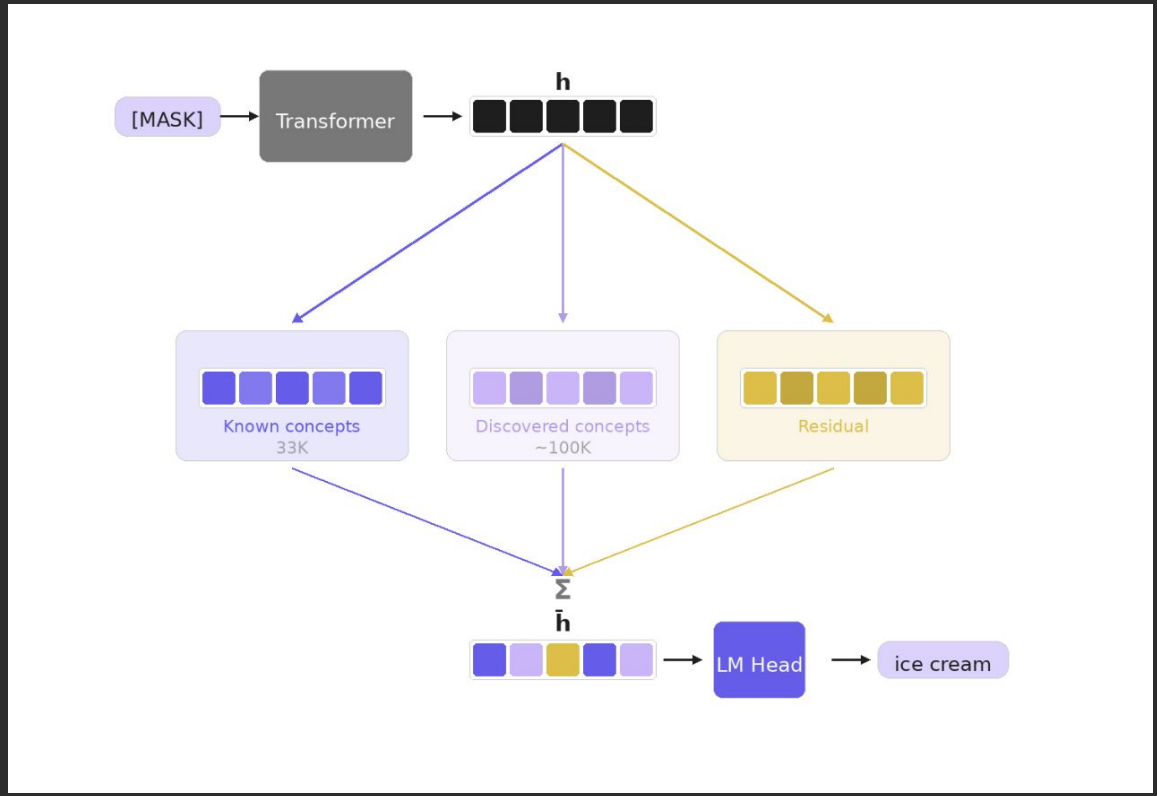


*This is particularly true when, during training, you randomly intervene** on a concept with probability $\mathbf{p_{int}}$.

** These sorts of training-time interventions are useful here only because by using embeddings, we can backpropagate gradients to the concept encoder even when a concept is intervened on (a CBM wouldn't).

[1] Espinosa Zarlenga, Barbiero, et al. "Concept embedding models: Beyond the accuracy-explainability trade-off." NeurIPS (2022).

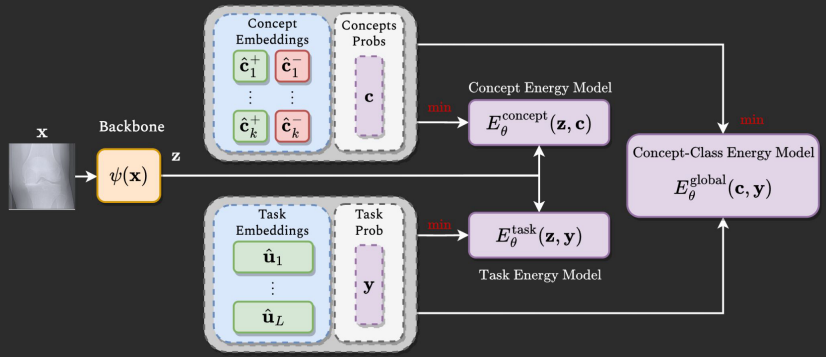
Multiple extra channels - Sterling 8B



[1] https://www.guidelabs.ai/post/steerling-8b-base-model-release/

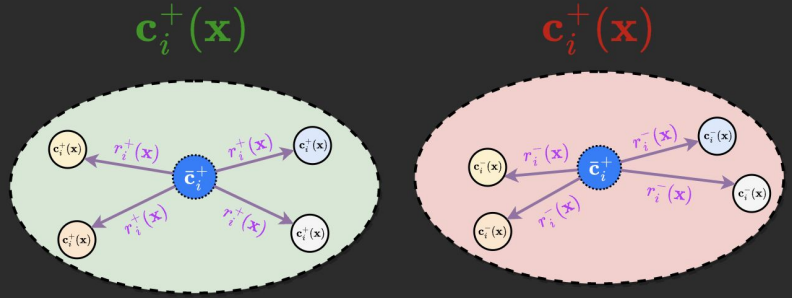
Further generalisations of concept embeddings

Energy-based CBMs



Concept embeddings can be exploited to compute arbitrary conditional distributions via energy models

MixCEMs



Concept embeddings can be learnt to support interventions in out-of-distribution test sets

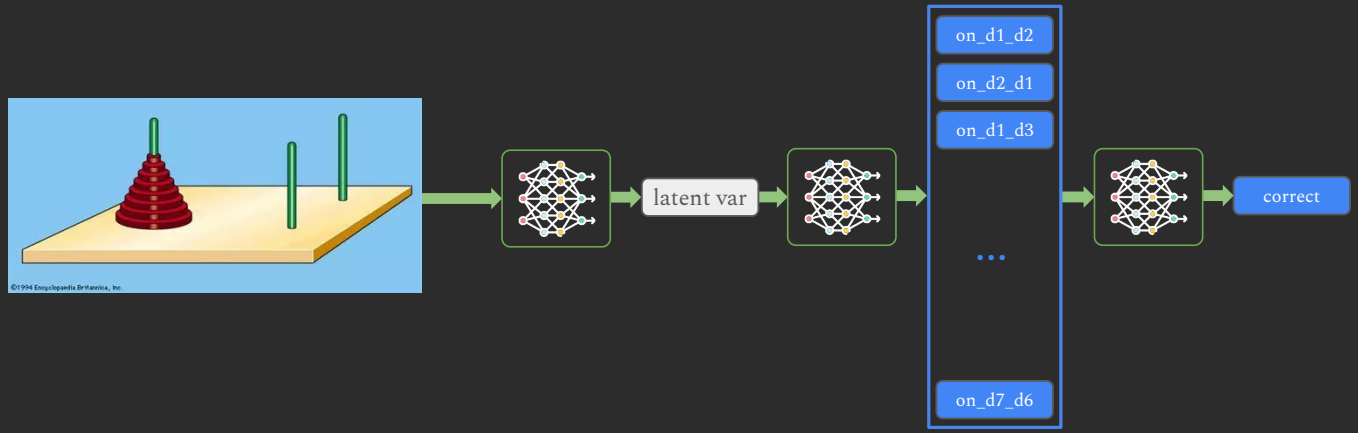
[1] Xu et al. "Energy-based concept bottleneck models: Unifying prediction, concept intervention, and probabilistic interpretations." ICLR (2024).
[2] Espinosa Zarlenga et al. "Avoiding Leakage Poisoning: Concept Interventions Under Distribution Shifts." ICML(2025).

The Concept Bottleneck Model contract

1) Syntax:
   a) Vocabulary of symbols (predicates and entities)
   b) The arities of the predicates

Only flat concepts (propositional): no objects / entities

2) Semantics:
   a) what values do we allow?
   b) which values do we assign at each symbol?



The Concept Bottleneck Model contract

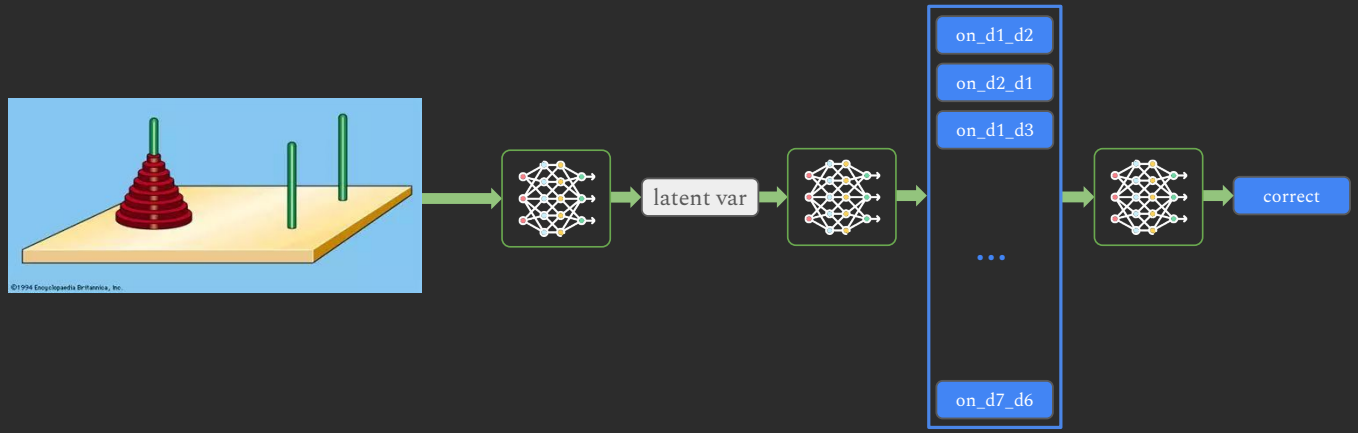
1) Syntax:
   a) Vocabulary of symbols (predicates and entities)
   b) The arities of the predicates

Only flat concepts (propositional): no objects / entities

2) Semantics:
   a) what values do we allow?
   b) which values do we assign at each symbol?

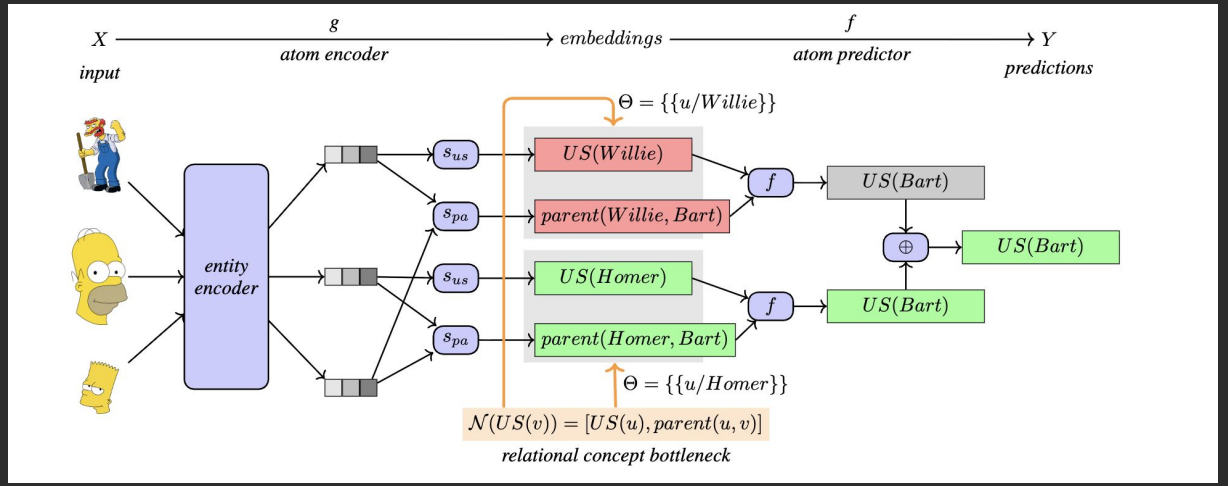
Failure:

Limited compositional generalization



Failure: limited compositional generalization
Repair: Relational vocabulary

- Concept-encoders become relational



[1] Barbiero et al; Relational Concept Bottleneck Models, NeurIPS 2024

Relational concept bottleneck

- Generalization to settings larger than the ones seen during training

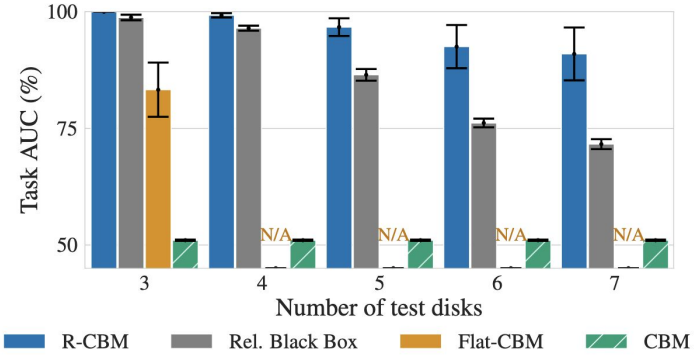


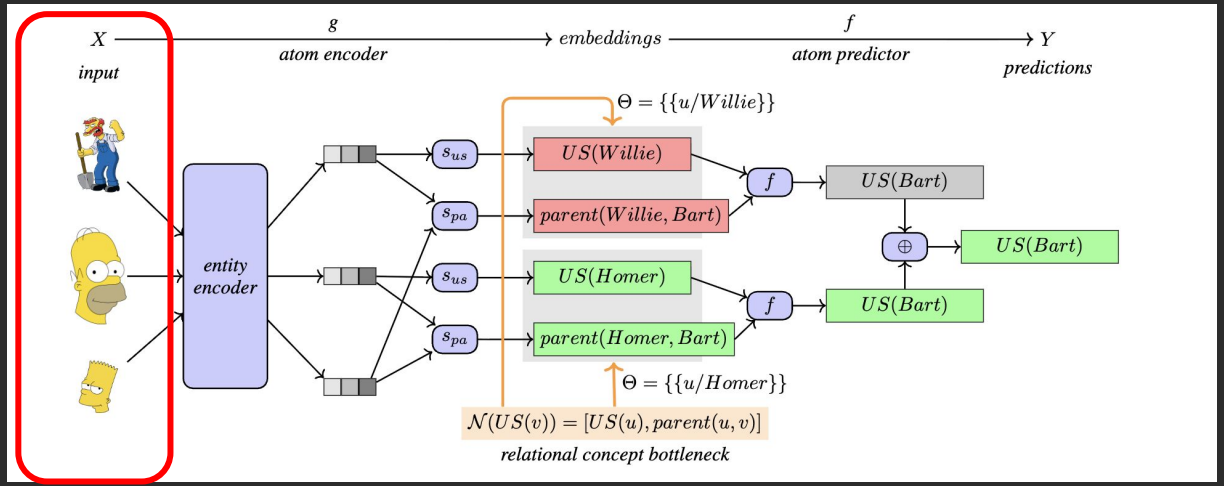
Figure 4: **Model generalization on Hanoi OOD on the number of disks.** Only R-CBMs are able to generalize effectively to settings larger than the ones they are trained on.

[1] Barbiero et al; Relational Concept Bottleneck Models, NeurIPS 2024

Relational Bottlenecks require structured inputs

Limitation:

- The input should already contain information about all the entities that exist in the given input

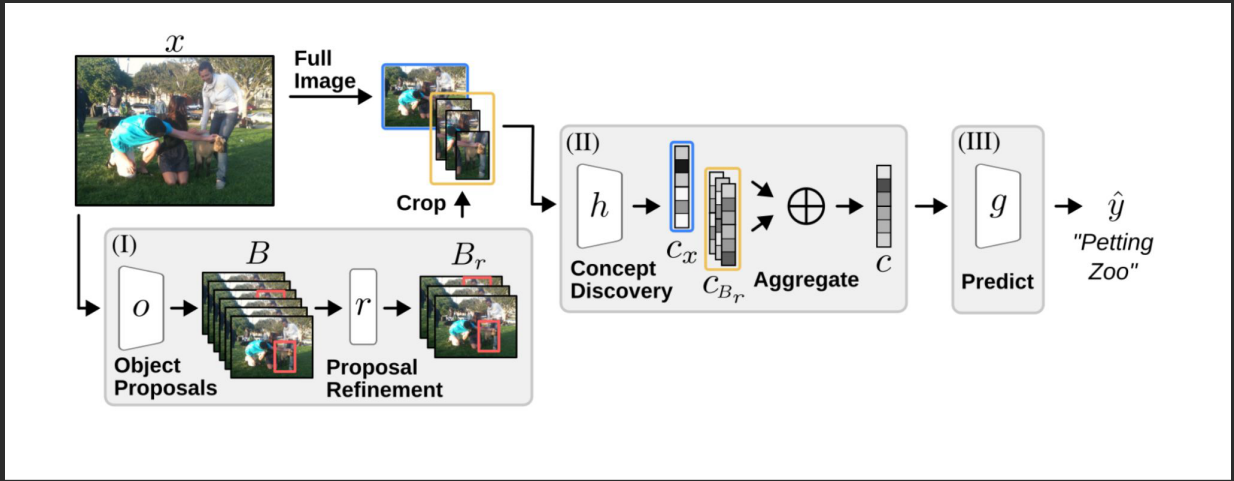


[1] Barbiero et al; Relational Concept Bottleneck Models, NeurIPS 2024

Object-centric bottlenecks automatically discover entities

Solution:

- Expand relational bottlenecks with an entity discovery model:
  - Mask-RCNN, SAM

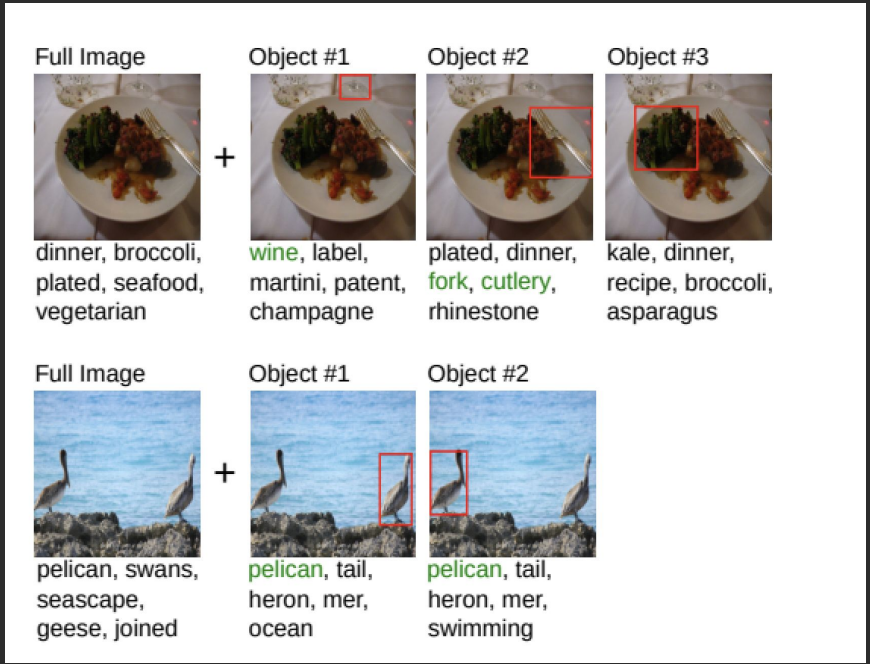


[1] Steinmann et al, Object-Centric Concept-Bottlenecks, NeurIPS 2025

Object-centric bottlenecks automatically discover entities

Solution:

- Expand relational bottlenecks with an entity discovery model

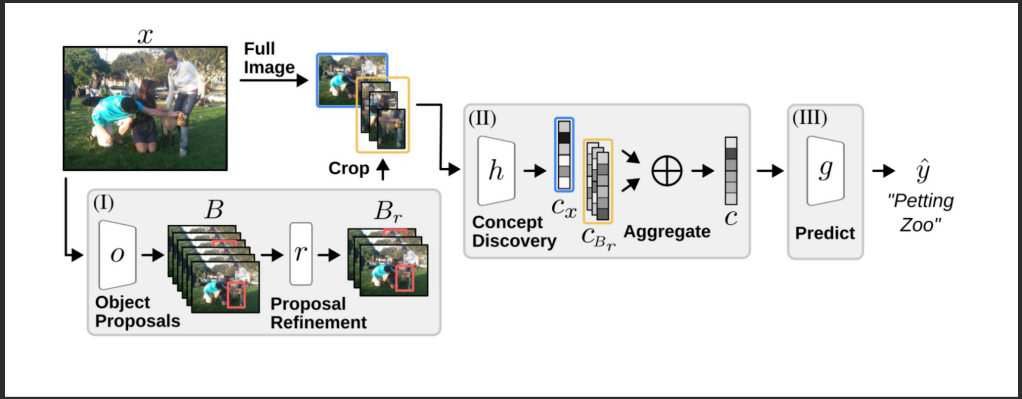


[1] Steinmann et al, Object-Centric Concept-Bottlenecks, NeurIPS 2025

Object-centric bottlenecks are based on pretrained encoders

Limitation:

- the encoder is frozen and pretrained
- what if it is not ideal for our downstream task?
  - e.g. a model trained on segmenting internet images may not be good at segmenting cancerogenic cells

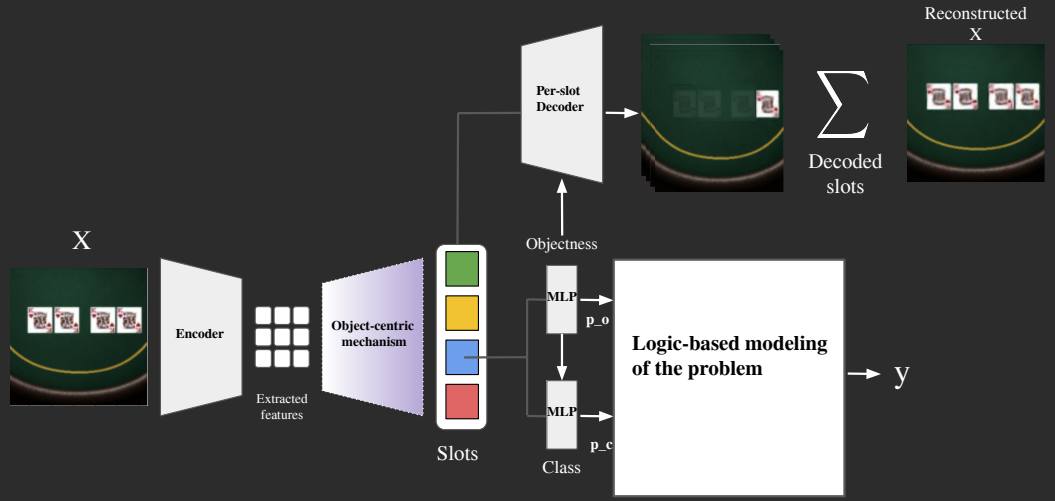


[1] Steinmann et al, Object-Centric Concept-Bottlenecks, NeurIPS 2025

Object-centric differentiable interfaces

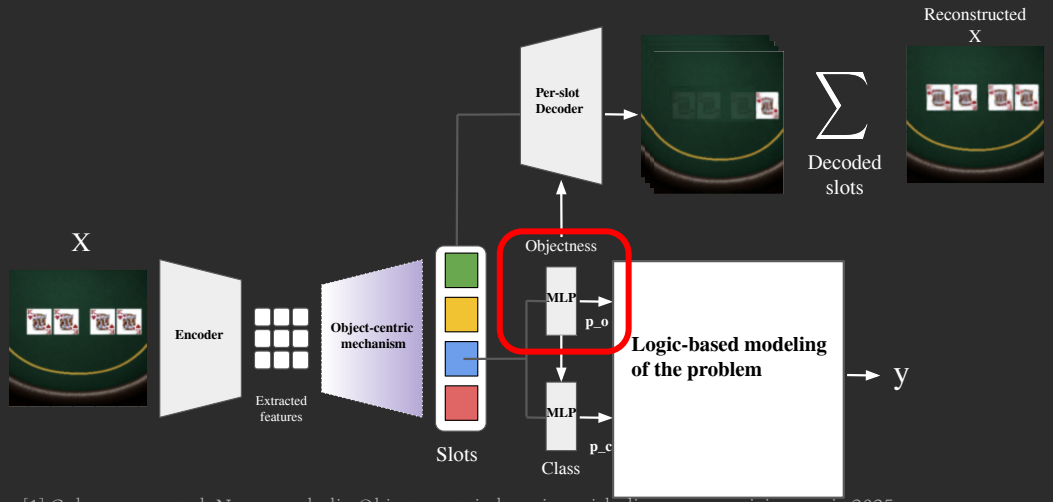
Solutions:

1) Train the object detector with explicit supervision
2) Design a differentiable interface that can "distantly" use the signal of your task to finetune your detector



[1] Colamonaco et al, Neurosymbolic Object-centric learning with distant supervision, arxiv 2025.

Object-centric differentiable interfaces



Solution:

- new "entity concepts"
  - we want to reason not only on concepts applied to entities
  - but on the very existence of entities
-

[1] Colamonaco et al, Neurosymbolic Object-centric learning with distant supervision, arxiv 2025.

Object-centric differentiable interfaces

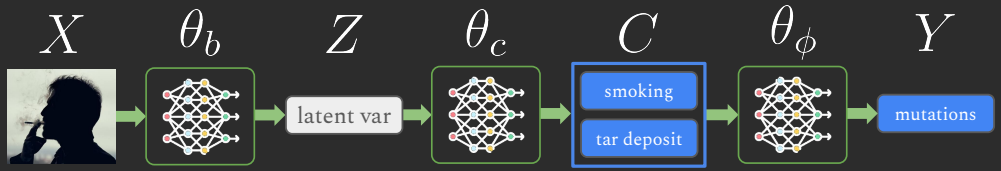
Solution:

- new “entity concepts”
  - we want to reason not only on concepts applied to entities
  - but on the very existence of entities

MM-A	Task Acc.		Concept Acc.		Extrapolation	
	Test	OOD	Test	OOD	4 digits	5 digits
CNN	79.90±0.17	3.43±0.15	-	-	22.16±0.90	13.16±0.96
SA	**98.90**±0.34	13.30±3.14	-	-	36.23±4.44	9.76±6.11
MESH	98.86±0.47	18.26±1.33	-	-	37.50±1.15	12.00±0.51
CoSA	93.00±1.92	36.2±15.10	-	-	52.20±14.01	25.06±12.82
NeSy	90.70±4.02	35.76±6.24	55.43±26.40	23.83±3.82	-	-
Ours	94.26±2.00	**90.00**±3.01	**85.16**±2.45	**65.46**±5.70	**69.73**±10.74	**44.06**±3.85

[1] Colamonaco et al, Neurosymbolic Object-centric learning with distant supervision, arxiv 2025.

Object centric concept alignment



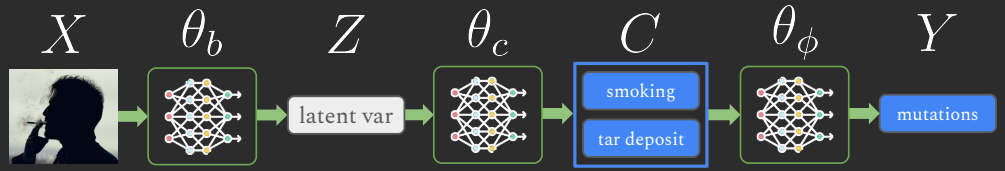
Why mutations?

"Mutations" because of "tar deposit"

Why tar deposit?

All CBMs stands on the assumption that we are aligned with concepts

Object centric concept alignment

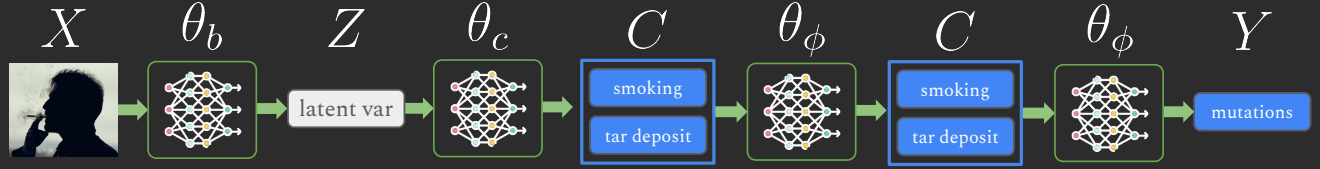


Why mutations?

"Mutations" because of "tar deposit"

Why tar deposit?

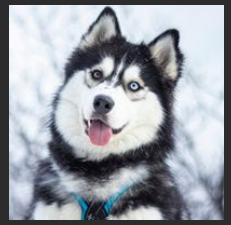
All CBMs stands on the assumption that we are aligned with concepts

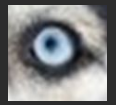


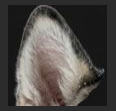
Prototype-based concept prediction

Concept Alignment Table

	$p(X_i \mid S_i)$	$p(C_{i,\text{almond eyes}} \mid S_i)$	$p(C_{i,\text{triangular ear}} \mid S_i)$
$S_i = 1$		0.9	0.2
$S_i = 2$		0.0	0.8
$S_i = 3$		0.0	0.0







$p(S_1 \mid X_1 =$  $) = \begin{matrix}1\\0\\0\end{matrix}$ → 0.9 0.2

$p(S_2 \mid X_2 =$  $) = \begin{matrix}0\\1\\0\end{matrix}$ → 0.0 0.8

Image part extraction

Prototype selection

Concept prediction

[1] Colamonaco et al. "Prototype-Grounded Concept Models for Verifiable Concept Alignment." ICML 2026

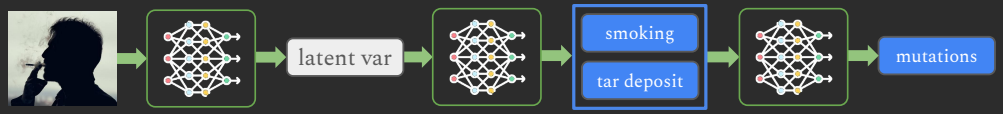
5′ Break

The Concept Bottleneck Model contract

1) Syntax:
   a) Vocabulary of symbols (predicates and entities)
   b) The arities of the predicates
2) Semantics:
   a) what values do we allow?

      Unclear

   b) which values do we assign at each symbol?



Semantics in neurosymbolic AI

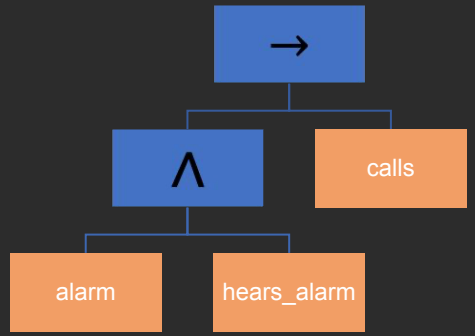
- Given a concept (like happy(homer)):

- semantics ask questions connected to:
  - is it True/False?
  - how much happy is this?
  - how likely is that he is happy?
  - are there factors of happiness? e.g. [*pleasure=2.7, satisfaction=11.7, purpose=0.1, ...*]
    - an embedding?

- We can make a parallel with logic

Boolean Logic

$$alarm \wedge hears\_alarm \rightarrow calls$$



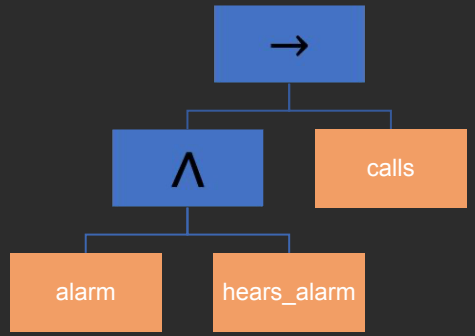
values are in {T,F} (or {0,1})

p	q	p→q
T	T	T
T	F	F
F	T	T
F	F	T

p	q	p∧q
T	T	T
T	F	F
F	T	F
F	F	F

Fuzzy Logic

$$alarm \wedge hears\_alarm \rightarrow calls$$



values are in [0,1]

	Product	Łukasiewicz	Gödel
$x \wedge y$	$x \cdot y$	$\max(0, x + y - 1)$	$\min(x, y)$
$x \vee y$	$x + y - x \cdot y$	$\min(1, x + y)$	$\max(x, y)$
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \Rightarrow y \ \ (x > y)$	y/x	$\min(1, 1 - x + y)$	y

Probabilistic Logic Semantics

- Two layer semantics

- Defines a **probability distribution** **over** **boolean interpretations**

$$p(x_1 = True, x_2 = False, ..., x_n = True) = 0.6$$

Probabilistic Logic Semantics
Independent Choice Semantics

e.g. in ProbLog:

B	E	H	p(B,E,H)
F	F	F	0.342
F	F	T	0.513
F	T	F	0.018
F	T	T	0.027
T	F	F	0.038
T	F	T	0.057
T	T	F	0.002
T	T	T	0.003

0.1 :: burglary. (B)

0.05 ::earthquake. (E)

0.6 ::hears_alarm(john). (H)

0.1 x 0.05 x (1- 0.6)

Probabilistic Logic Semantics
Markov Logic

1.5 : calls(Mary) <- hears_alarm(Mary), alarm

2.0 : alarm <- earthquake

0.5 : alarm <- burglary

Weight formula

1 if α is True otherwise 0

$$p(\ell_B(x_1), \ldots, \ell_B(x_n)) = \frac{1}{Z} exp(\sum_{\alpha} w_\alpha \ell_B(\alpha))$$

Probabilistic Logic Semantics
Markov Logic

1.5 : calls(Mary) <- hears_alarm(Mary), alarm

2.0 : alarm <- earthquake

0.5 : alarm <- burglary

B	E	A	H	C	p
T	F	T	T	T	0.05
T	F	T	T	F	0.01
…	…	…	…	…	…

prop. to exp(**1.5** + **2.0** + **0.5**)
prop. to exp(**0.0** + **2.0** + **0.5**)

The “neural” semantics

In concept based models and neurosymbolic models, one perspective on the role of neural networks is:

neural networks are conditional deep parameterizations of formal languages semantics

Symbol: sunny

Scalar semantics: label(sunny) = 0.7
Neural semantics: label(sunny |  ; parameters)

The Concept Bottleneck Model contract

2) Semantics:
   a) what values do we allow?

Unclear

Learning: often with Binary Cross Entropy
- Implicit assumption: Independent Choice (Bernoulli concepts)

Task Prediction:
- Scores are passed to the task predictor: ~ Fuzzy Semantics

Intervention-time:
- True/False interventions: Boolean Semantics

Concept-Embedding Models:
- ?!??!!!?

Neural task predictors break concept semantics

- Concept-based models often ignore the semantics of the concepts used for task prediction.

- For example, consider a probabilistic concept, p(tail)= 0.7

- score 0.7 -> input to a neural network
- the model combines it with other concept scores in ways that no longer respect probability theory.

- Result = the original semantic meaning of the concept is lost.

Semantic unclarity leads to leakage

Leakage: the concept representation carries information beyond the intended human concept.

- Humans give a certain meaning

- The machine gives (possible) that meaning + extra info
  - not a clean bottleneck
  - hidden information

Types of leakage

tail = 0.7

At least 4 types of leakage:

- Task leakage
- Input (distribution) leakage
- Concept leakage (entanglement)
- OOD leakage

Types of leakage

tail = 0.7

At least 4 types of leakage:

- Task leakage
- Input (distribution) leakage
- Concept leakage (entanglement)
- OOD leakage

tail = 0.7 vs tail = 0.8

tail = 0.7 means:

- tail is True
- whiskers is True

-> cat is True

tail = 0.8 means:

- tail is True
- whiskers is False

-> dog is True

[1] Havasi et al, Addressing Leakage in Concept Bottleneck Models, NeurIPS 2022.

Types of leakage

tail = 0.7

At least 4 types of leakage:

- Task leakage
- Input (distribution) leakage
- Concept leakage (entanglement)
- OOD leakage

tail = 0.7 vs tail = 0.8

Causes:

- incomplete concept set
  - thus continuous degree as extra capacity
- inexpressive task predictor
  - task needs a “non-linear′ feature of concepts
  - use of continuous degree as the non linear feature

[1] Havasi et al, Addressing Leakage in Concept Bottleneck Models, NeurIPS 2022.

Types of leakage

tail = 0.7

At least 4 types of leakage:

- Task leakage
- Input (distribution) leakage
- Concept leakage (entanglement)
- OOD leakage

tail = 0.7

- Even when trained separately (no pressure from the task)
- concepts can encode other unrelated concepts that may be useful for an unknown task
  - (aka sequential training)

[1] Manipei et al, Promises and Pitfalls of Black-Box Concept Learning Models, ICML 2021

Types of leakage

tail = 0.7

At least 4 types of leakage:

- Task leakage
- Input (distribution) leakage
- Concept leakage (entanglement)
- OOD leakage

tail = 0.7

- Concepts can contain information of other (known) concepts
  - intervening on one is not enough

[1] Manipei et al, Promises and Pitfalls of Black-Box Concept Learning Models, ICML 2021

Types of leakage

tail = 0.7

At least 4 types of leakage:

- Task leakage
- Input (distribution) leakage
- Concept leakage (entanglement)
- OOD leakage

- Even if no leakage in-distribution
- OOD inputs can predicts misaligned concept scores

[1] Marconato et al, GlanceNets, NeurIPS 2022

Why NeSy pays more attention to semantics?

- Without proper semantics specification, concepts cannot be consumed by predictors defined within certain formal frameworks
  - probabilistic logics
  - fuzzy logics
  - programming languages

- Most of the semantic attention in CBMs come from a “pressure” from the task predictor

Spoiler: Probabilistic concepts for Probabilistic Tasks

Probabilistic Graphical Models (e.g. Counterfactual CBMs)

1. What? Predict concept and class labels:

(a) **Sample from latent concept posterior:** $z \sim \mathcal{N}(\phi_\mu(x), \phi_\sigma(x))$

(b) **Sample to predict concept and class labels:** $\hat{y} \sim \text{Cat}(f(\hat{c}))$, $\hat{c} \sim \text{Ber}(\phi_z(z))$

Probabilistic Logic Predictors (e.g. DeepProbLog, CMR)

The probability $P(w_F)$ of such a possible world w_F is given by the product of the probabilities of the truth values of the probabilistic facts, $P(w_F) = \prod_{f_i \in F} p_i \prod_{f_i \in \mathcal{F} \setminus F} (1 - p_i)$. For instance,

but also..

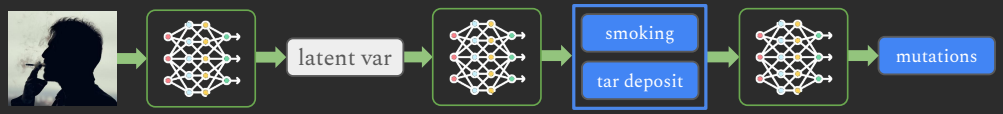
Fuzzy Logic Decoders (e.g. DCR)

concept-based models represent concepts using their truth degree, that is, $\hat{c}_1, \ldots, \hat{c}_k \in [0, 1]$.

The Concept Bottleneck Model contract

1) Syntax:
   a) Vocabulary of symbols (predicates and entities)
   b) The arities of the predicates
2) Semantics:
   a) what values do we allow?
   b) which values do we assign at each symbol?

Fully Supervised



The Concept Bottleneck Model contract

1) Syntax:
   a) Vocabulary of symbols (predicates and entities)
   b) The arities of the predicates
2) Semantics:
   a) what values do we allow?
   b) which values do we assign at each symbol?

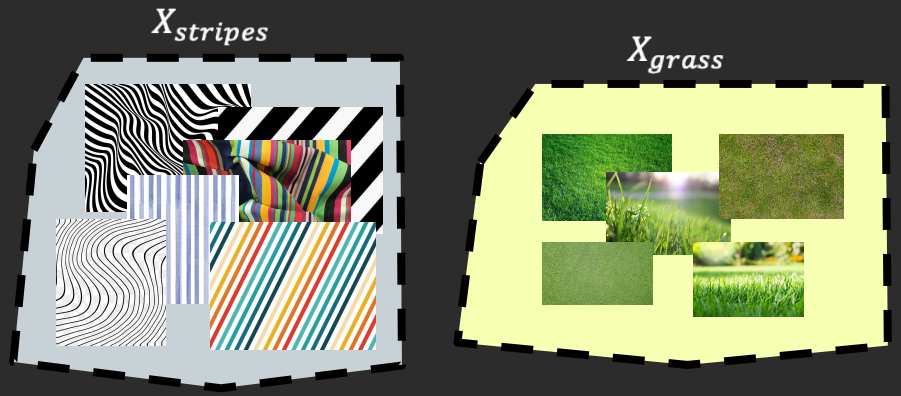
Fully Supervised

Failure: huge labelling and training effort

Failure: Huge training effort
Repair: Concept-activation vectors

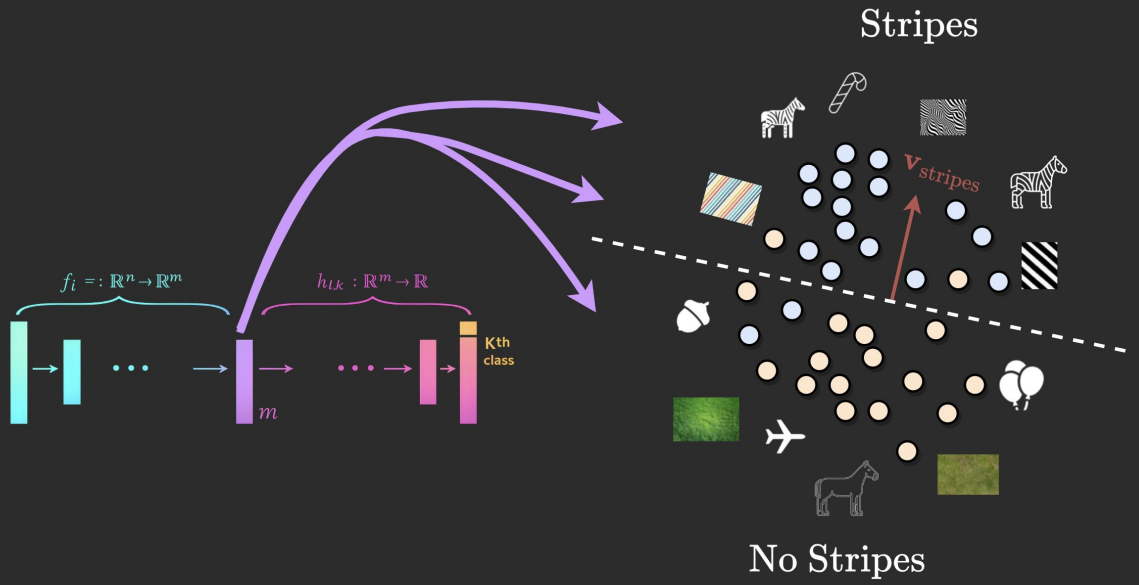
We want to avoid training a large concept encoder from scratch when we:

1. Do not have the necessary compute to train a concept encoder from scratch
2. Have at our disposal an already powerful but opaque DNN for P(Y|X)
3. Have sparse concept labels (e.g., sets of representative concept samples)



Concept activation vectors

Given concept sets, we can learn directions in the DNN's latent space, called **Concept Activation Vectors (CAVs)**, that "point" towards a concept:

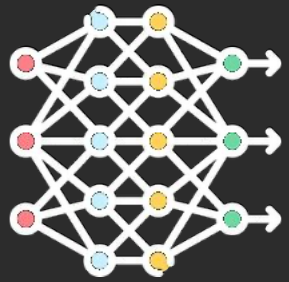


[1] Kim et al. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)." ICML (2018).

Representing concepts as similarity scores

A Post-hoc CBM (PCBM) uses a pretrained DNN to construct concept representations based on similarity scores from projections onto CAVs:

A pre-trained DNN

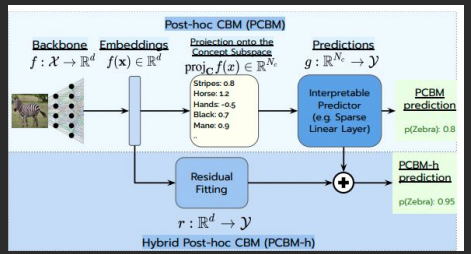


+

Concept Activation Vector Bank

=

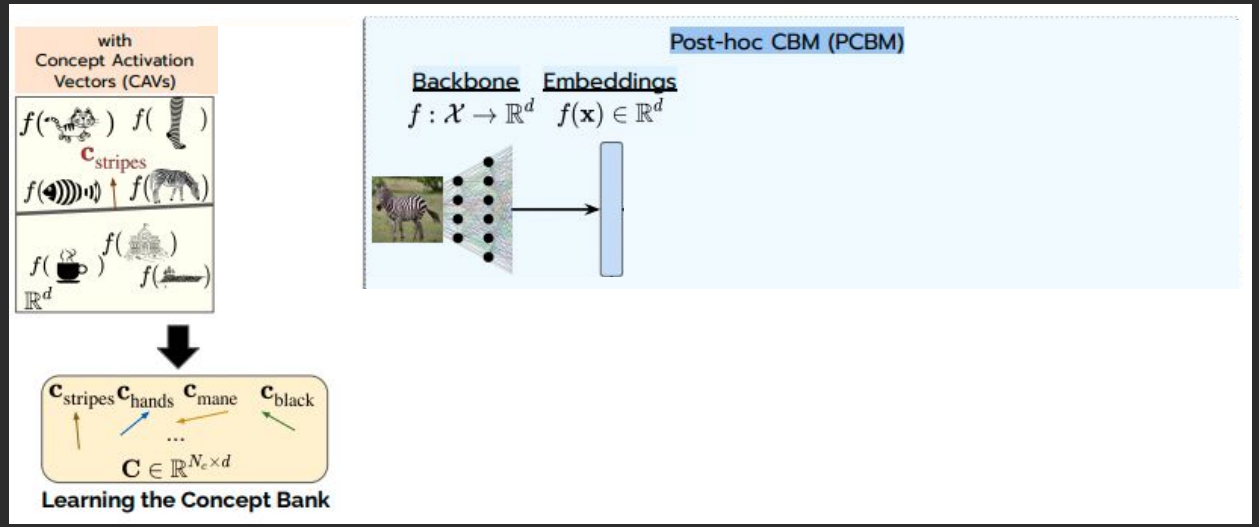
Post-hoc CBMs!



[1] Yuksekgonul et al. "Post-hoc concept bottleneck models." ICLR (2023).

Representing concepts as similarity scores

Learn a CBM that maps concept similarity scores to task labels as follows:

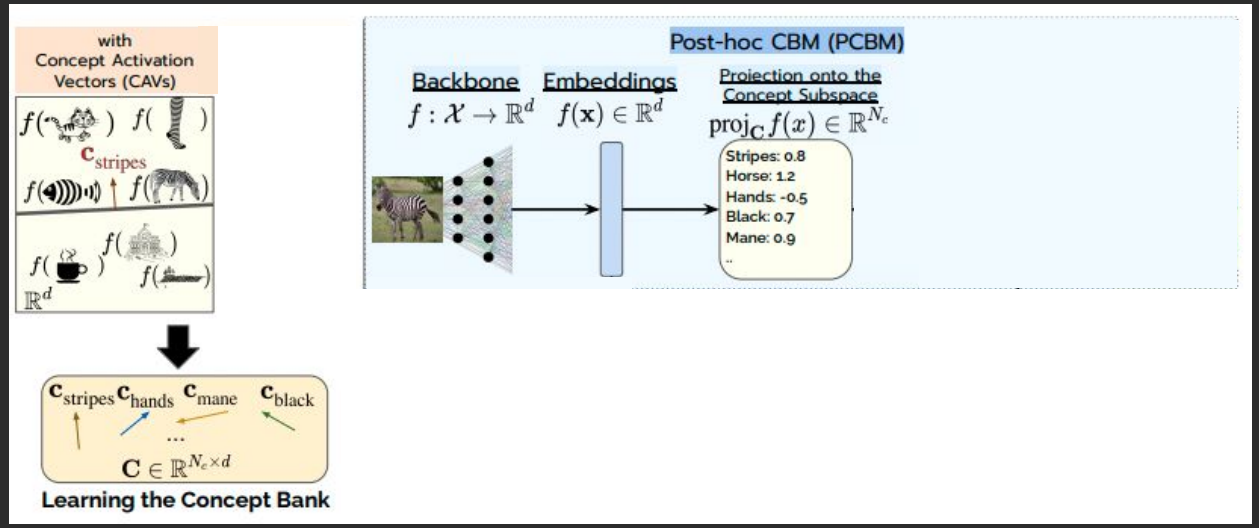


Step 1: learn a Concept Activation Vector (CAV) that separates the DNN's latent space between samples with a concept vs samples without a concept

[1] Yuksekgonul et al. "Post-hoc concept bottleneck models." ICLR (2023).

Representing concepts as similarity scores

Learn a CBM that maps concept similarity scores to task labels as follows:

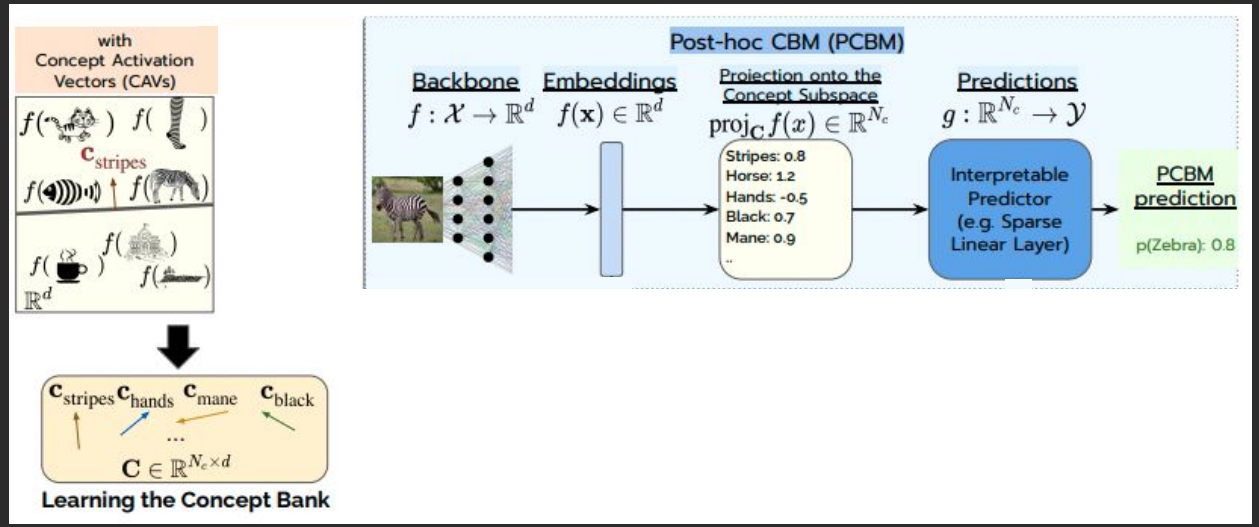


Step 2: project all training samples to the concept activation space using the CAVs

[1] Yuksekgonul et al. "Post-hoc concept bottleneck models." ICLR (2023).

Representing concepts as similarity scores

Learn a CBM that maps concept similarity scores to task labels as follows:



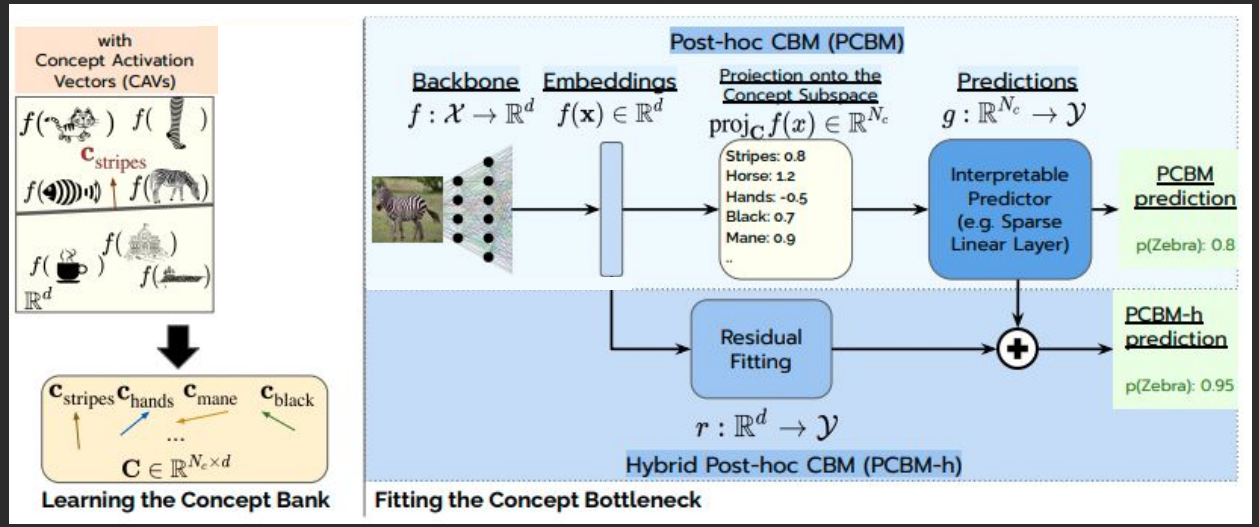
Make the final prediction with an interpretable predictor

Step 3: learn an interpretable predictor from the concept scores to the task labels

[1] Yuksekgonul et al. "Post-hoc concept bottleneck models." ICLR (2023).

Representing concepts as similarity scores

Learn a CBM that maps concept similarity scores to task labels as follows:



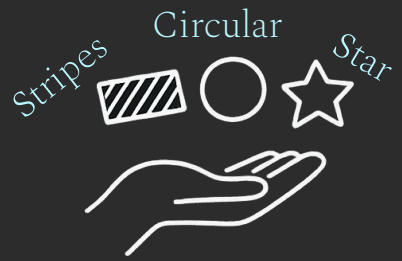
Step 4 (optional): fit a residual model if the new model is underperforming

[1] Yuksekgonul et al. "Post-hoc concept bottleneck models." ICLR (2023).

Failure: No available labels

Failure: No available labels
Repair: multimodal concept encoders

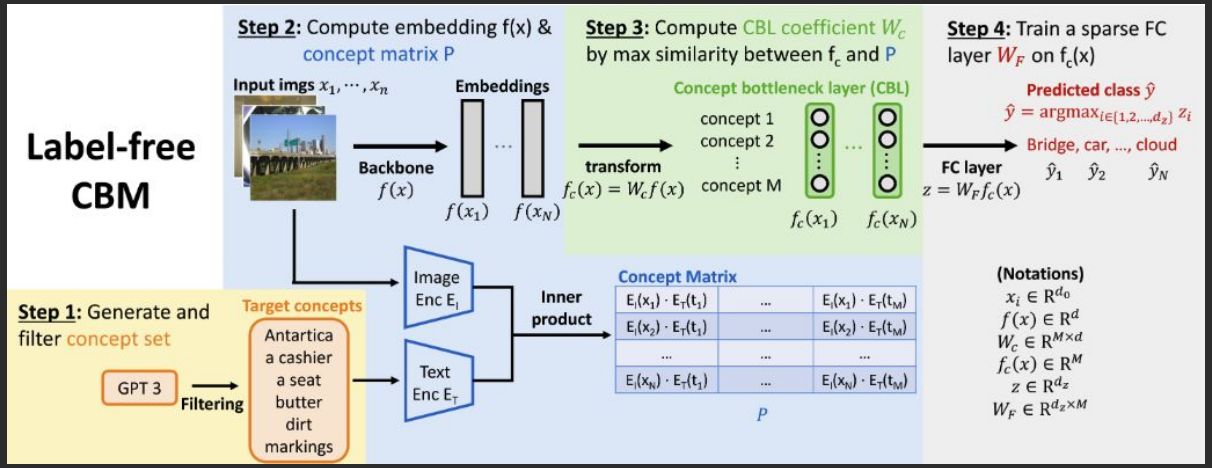
It is much *easier* to **describe (in text) the concepts** that are useful for a task than to annotate them on a large dataset



Vs.

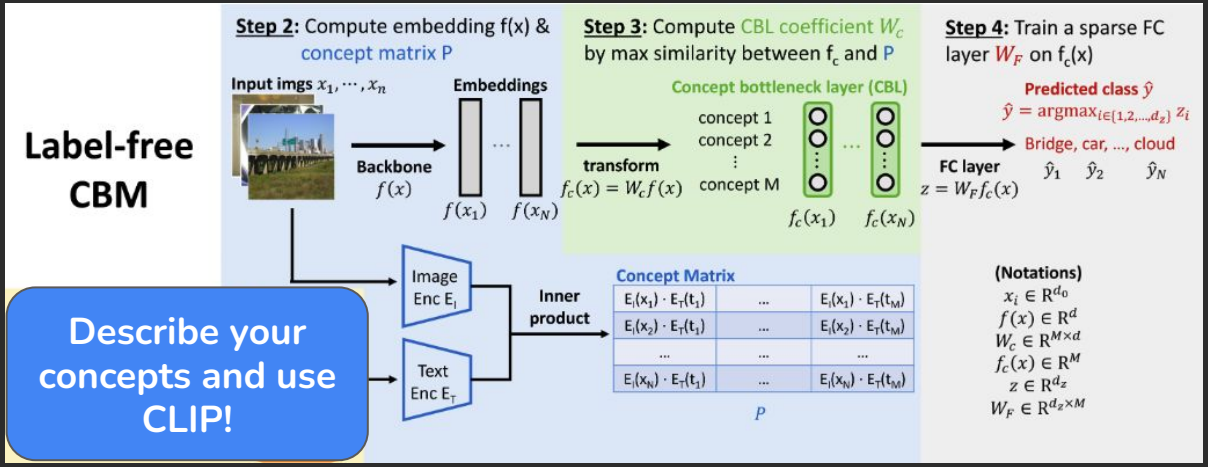
Multimodal concept encoders

It is much *easier* to **describe the concepts** that are useful for a task than to annotate them on a large dataset



Multimodal concept encoders

It is much *easier* to **describe the concepts** that are useful for a task than to annotate them on a large dataset



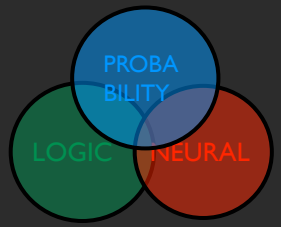
Failure: Few available labels
Repair: Constrain the concepts logically

Train concept encoders using a signal coming from a semantic based regularization:

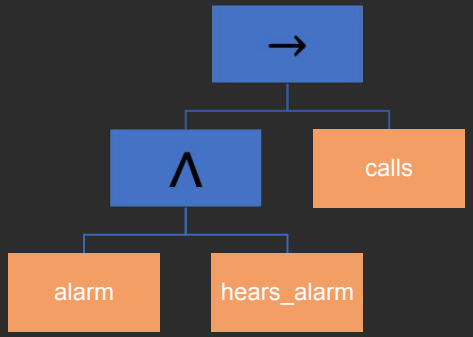
- unsupervised loss functions encoding logic

$$\text{Ski} \rightarrow \text{Snow} \vee \text{Mountain} \vee \text{Helmet}$$

$$\text{Dog} \wedge \text{Person} \rightarrow \text{Leash} \vee \text{Sofa} \vee \text{Ball}$$



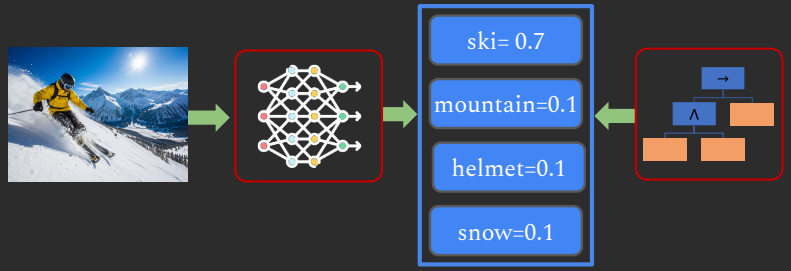
Constrain the concepts logically

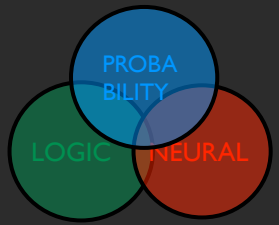


	Product	Łukasiewicz	Gödel
$x \wedge y$	$x \cdot y$	$\max(0, x + y - 1)$	$\min(x, y)$
$x \vee y$	$x + y - x \cdot y$	$\min(1, x + y)$	$\max(x, y)$
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \Rightarrow y \ \ (x > y)$	y/x	$\min(1, 1 - x + y)$	y

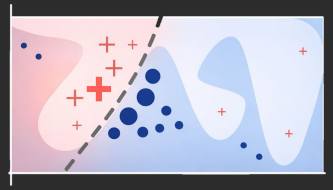
values are in [0,1]

Ski → Snow ∨ Mountain ∨ Helmet

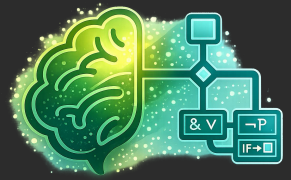




What's next?



Locally-interpretable Predictors



Neuro-Symbolic Predictors

Causal Predictors