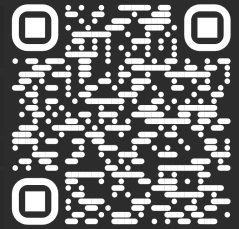
Foundations of Interpretable Deep Learning

Pietro Barbiero & Mateo Espinosa Zarlenga
pietro.barbiero@ibm.com mateo.espinosazarlenga@trinity.ox.ac.uk

In collaboration with: Alberto Termine, Mateja Jamnik, Giuseppe Marra



Who are we?

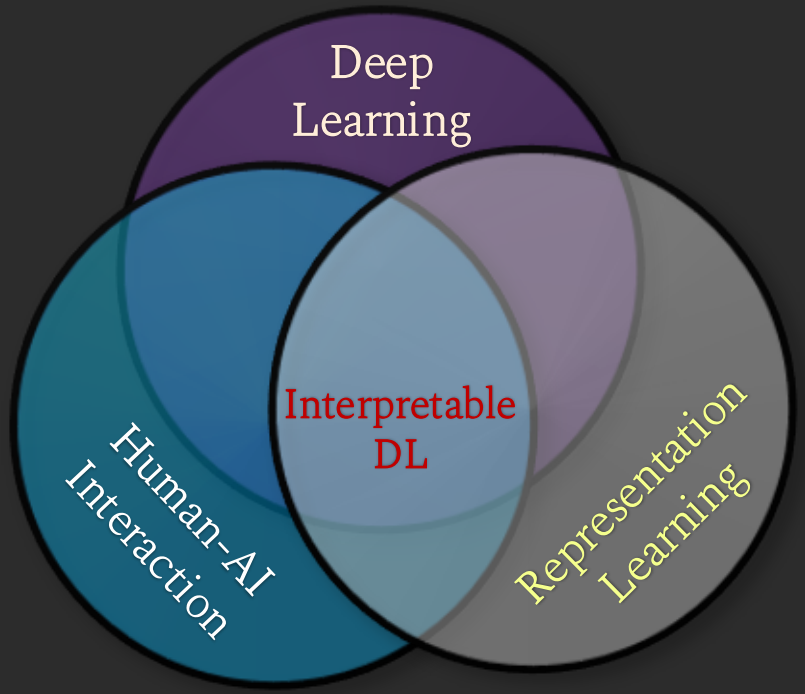
Pietro Barbiero
Swiss Postdoctoral Fellow
IBM Research (Switzerland)
pietro.barbiero@ibm.com

IBM Research

Mateo Espinosa Zarlenga
Junior Research Fellow
University of Oxford
mateo.espinosazarlenga@trinity.ox.ac.uk

UNIVERSITY OF OXFORD

What is this tutorial about?



Q: how we can build Deep Neural Networks (DNNs) that reason based on human-interpretable representations?

What is this tutorial about?

This tutorial aims to:

1. Convince you that interpretability can be thought of in terms of symmetries which induce a blueprint with well-defined components;

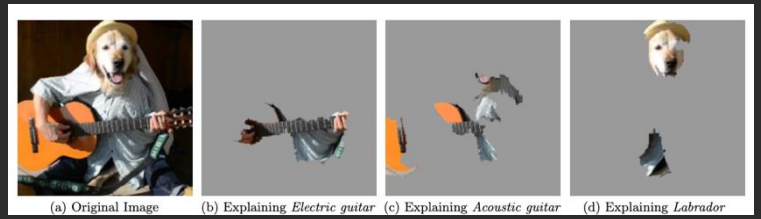
2. Provide a non-exhaustive but well-rounded overview of interpretable models using this blueprint as a basis;

3. Bring together a variety of resources (surveys, method papers, etc.) together with a useful library (PyC) to facilitate development of interpretable DNNs

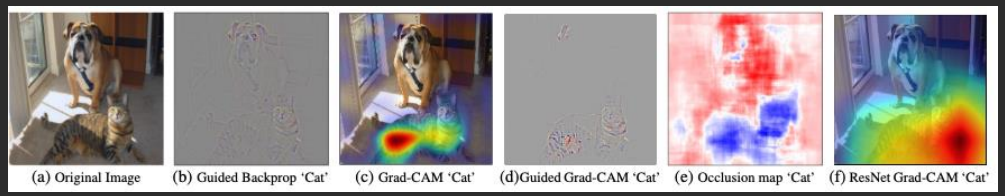
What is this tutorial NOT about?

We will not have time to dive deep into:

1. "Traditional" explainable AI (XAI) methodologies



Example of LIME (taken from [1])



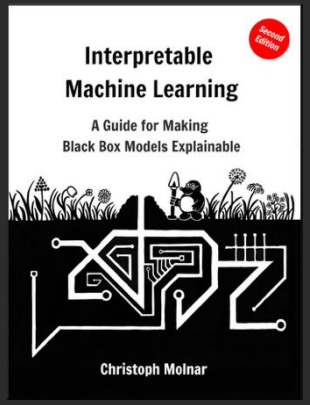
Example of GradCAM and other saliency methods (taken from [2])

[1] Ribeiro et al. "'Why should I trust you?' Explaining the predictions of any classifier." KDD (2016).
[2] Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." ICCV (2017).

What is this tutorial NOT about?

We will not have time to dive deep into:

1. "Traditional" explainable AI (XAI) methodologies



"Interpretable Machine Learning"
Christoph Molnar

Link to Book

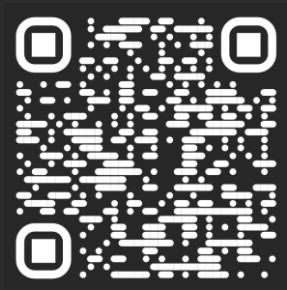
[1] Molnar. Interpretable machine learning. Lulu. com, 2020.

What is this tutorial NOT about?

We will not have time to dive deep into:

1. "Traditional" explainable AI (XAI) methodologies
2. Deep philosophical aspects of explaining models

"The Mythos of Interpretability"
Lipton (2018) [1]



"To Explain or to Predict?"
Shmueli (2018) [2]

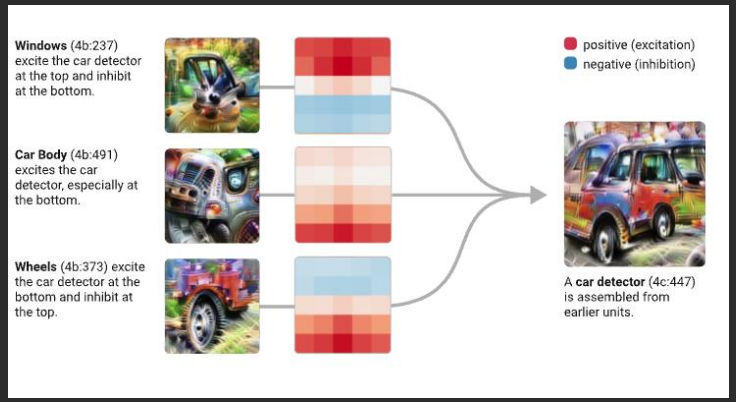
"Explanation Theory"
Bromberger (1992) [3]

[1] Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery," Queue (2018).
[2] Shmueli, "To Explain or to Predict?," Statist. Sci. 25 (3) 289 - 310, August 2010. https://doi.org/10.1214/10-STS330
[3] Bromberger, On what we know we don't know: Explanation, theory, linguistics, and how questions shape them. University of Chicago Press, 1992.

What is this tutorial NOT about?

We will not have time to dive deep into:

1. "Traditional" explainable AI (XAI) methodologies
2. Deep philosophical aspects of explaining models
3. Post-hoc interpretability such as Mechanistic Interpretability



Taken from [1]

[1] Olah et al. "Zoom in: An introduction to circuits." Distill 5.3 (2020): e00024-001.

What is this tutorial NOT about?

We will not have time to dive deep into:

1. "Traditional" explainable AI (XAI) methodologies
2. Deep philosophical aspects of explaining models
3. Post-hoc interpretability such as Mechanistic Interpretability

Distill Circuits Thread

Anthropic Circuits Thread

[1] Cammarata et al. "Thread: circuits." Distill 5.3 (2020): e24.
[2] Anthropic "Transformer Circuits Thread" found at https://transformer-circuits.pub/

Tutorial Outline

I. Introduction (~15 min)

II. Interpretability Symmetries (~40 min)

III. Interpretable Models & Inferences (~30 min)

Q&A + 30 min break

V. Designing Concept Representations

VI. Designing Task Predictors

VII. Human-AI Interaction

VIII. Conclusion

Final Q&A

Tutorial Outline

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences

8:30 am – ~10:10 am

Q&A + 30 min break

10:10 am – ~10:45 am

IV. Designing Concept Representations

V. Designing Label Predictors

VI. Human-AI Interaction

VII. Conclusion

Final Q&A

Tutorial Outline

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences

Q&A + 30 min break

IV. Designing Concept Representations

V. Designing Label Predictors

VI. Human-AI Interaction

VII. Conclusion

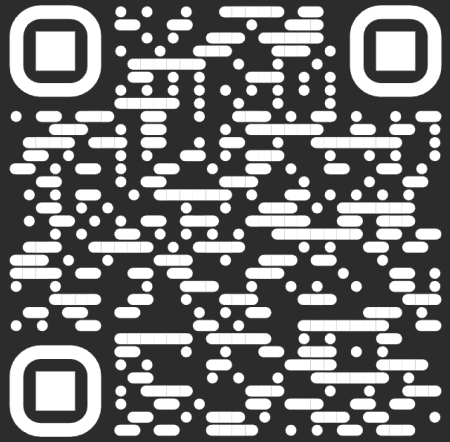
10:45 am – ~12:20 pm

Final Q&A

~12:20 pm – 12:20 pm

Tutorial Website and Materials

This tutorial's slides, schedule, and website are all available at our website:

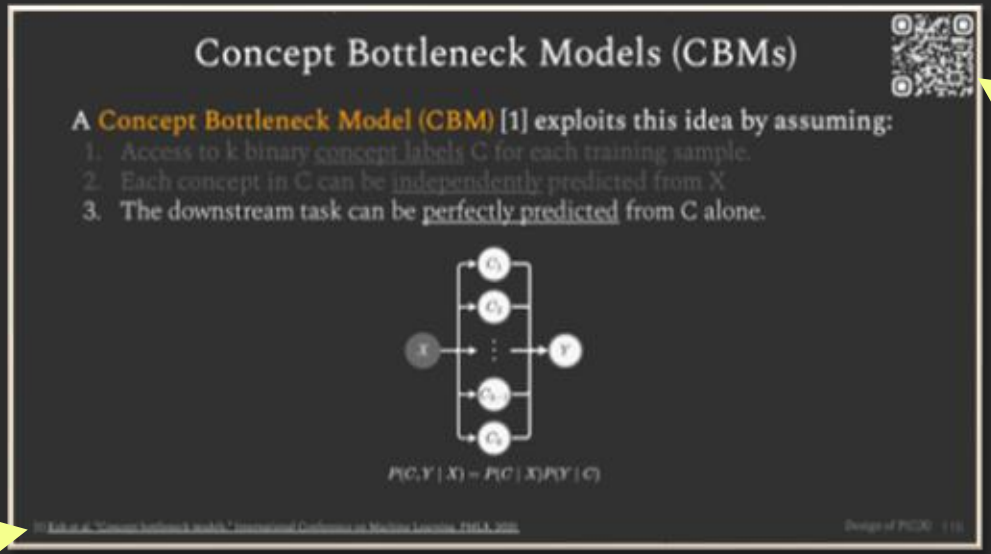


interpretabledeeplearning.github.io

Tutorial Website and Materials

Throughout the tutorial, watch for:

1. QR codes for relevant references and extra material



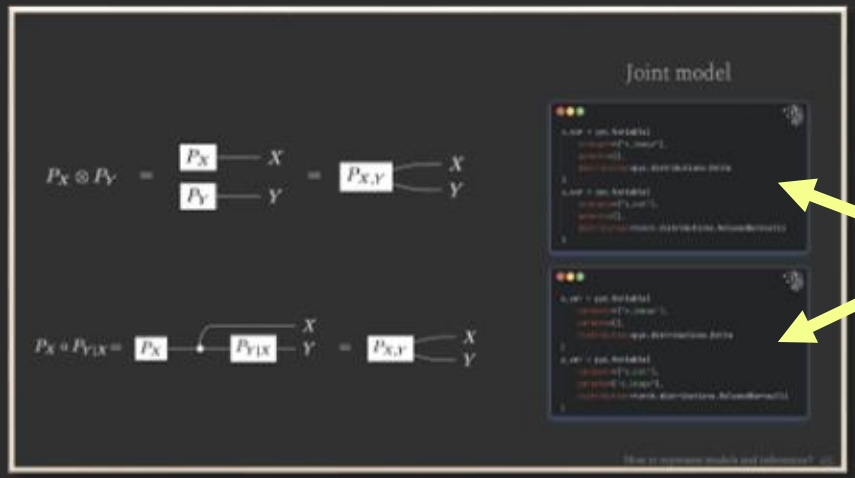
QR code to reference/extra material

Citation + Hyperlink (if you download slides)

Tutorial Website and Materials

Throughout the tutorial, watch for:

1. QR codes for relevant references and extra material
2. Code snippets showing example implementations in PyC



PyC Code Snippets

Tutorial Outline

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences

Q&A + 30 min break

IV. Designing Concept Representations

V. Designing Task Predictors

VI. Human-AI Interaction

VII. Conclusion

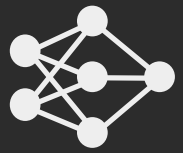
Final Q&A

Introduction to interpretability

Why do we need interpretability?

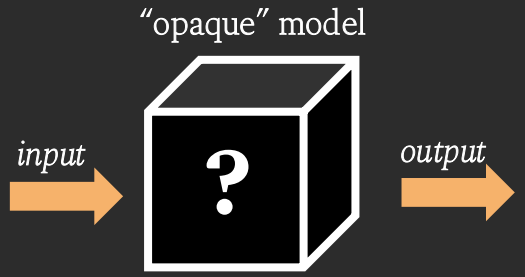
How is interpretability defined?

Why do we need interpretability in the first place?

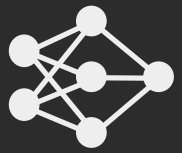


- Highly parametric
- Complicated inference steps
- Non-linearities
- Sensitive to initial states and update rules

≈

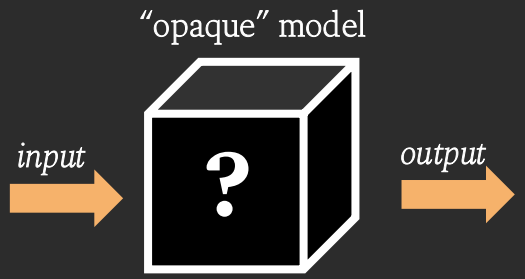


Why do we need interpretability in the first place?



- Highly parametric
- Complicated inference steps
- Non-linearities
- Sensitive to initial states and update rules

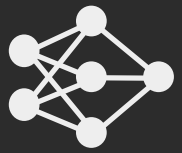
≈



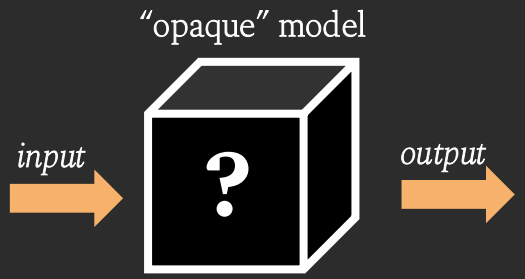
1. **Blindly using opaque models** can lead to all sorts of **problems**

[1] Kashmir Hill, "Wrongfully Accused by an Algorithm," The New York Times (2020).
[2] Rachel Goodman, "Why Amazon's Automated Hiring Tool Discriminated Against Women," ACLU (2018).
[3] Will Douglas Heaven, "Predictive policing algorithms are racist. They need to be dismantled," MIT Technology Review (2020).

Why do we need interpretability in the first place?



- Highly parametric
- Complicated inference steps
- Non-linearities
- Sensitive to initial states and update rules



1. **Blindly using opaque models** can lead to all sorts of **problems**
2. **Regulatory/legal** constraints:

General Data Protection Regulations (GDPR, 2016)

"The data subject shall have the right not to be subject to a **decision** based solely on **automated processing**, including profiling,…" (Art. 22)

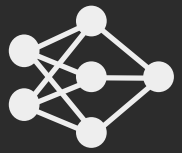
The data subject has the right to "**meaningful information** about the **logic** involved" in the decision. (Art. 13 and 15)

EU AI Act (2024):

"Any affected person subject to a **decision** which is taken by… a **high-risk AI system** … shall have the right to obtain from the deployer **clear and meaningful explanations** (Art. 86)

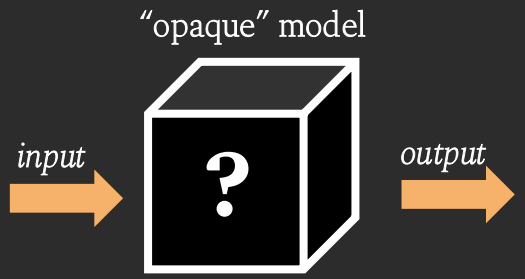
[1] GDPR, EU. "Automated individual decision-making, including profiling." (2022).
[2] Act, EU Artificial Intelligence. "The EU Artificial Intelligence Act." (2024).

Why do we need interpretability in the first place?



- Highly parametric
- Complicated inference steps
- Non-linearities
- Sensitive to initial states and update rules

≈



1. **Blindly using opaque models** can lead to all sorts of **problems**
2. **Regulatory/legal** constraints
3. **Human-AI interactions** are enabled by understanding how a model reasons

But what does interpretability really mean?

Welcome to the Wild West of interpretability terminology

Goebel et al. (2018)

Freiesleben et al. (2023)

Gilpin et al. (2018)

Vilone et al. (2021)

Confalonieri et al. (2020)

Rudin et al. (2019)

Barredo Arrieta et al. (2020)

But what does interpretability really mean?

Before we formalize matters, it is helpful to informally think of interpretability in informal terms (Gilpin et al. [1]):

- **Explainability (why):** the ability to answer questions of the form "why does this particular input lead to that particular output?"

- Interpretability (how): the ability to describe "the internals of a system in a way that is understandable to humans."

Under this view, interpretability implies explainability

[1] Gilpin et al. "Explaining explanations: An overview of interpretability of machine learning," DSAA (2018).

*"A method is interpretable **if a user can** correctly and efficiently **predict** the method's **results**"* [1]

*"Systems are interpretable **if their operations can be understood** by a human"* [2]

*"Interpretability is the degree to which an observer can **understand the cause of a decision**"* [3]

*"There is **no universal, mathematical definition** of interpretability, and there never will be"* [4]

Informal

Not actionable

[1] Kim et al. "Examples are not enough, learn to criticize! criticism for interpretability" NeurIPS (2016).
[2] Biran et al. "Explanation and Justification in Machine Learning: A Survey" IJCAI workshop (2017).
[3] Miller. "Explanation in artificial intelligence: Insights from the social sciences" Artificial Intelligence (2019).
[4] Murphy. "Probabilistic machine learning: Advanced topics" MIT press (2023).

Tutorial Outline

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences

Q&A + 30 min break

IV. Designing Concept Representations

V. Designing Task Predictors

VI. Human-AI Interaction

VII. Conclusion

Final Q&A

How to formalize interpretability in AI?

Interpretability Symmetries

Symmetry I
Inference equivariance

Symmetry II
Information invariance

Symmetry III
Concept invariance

Symmetry IV
Structural invariance

Interpretability Symmetries

Symmetry I
Inference equivariance

Symmetry II
Information invariance

Symmetry III
Concept invariance

Symmetry IV
Structural invariance

*"A method is interpretable **if a user can** correctly and efficiently **predict** the method's **results**"*

[1] Kim et al. "Examples are not enough, learn to criticize! criticism for interpretability" NeurIPS (2016).

object *features* *inference*

	X_1	X_2	$P_{Y\|X}(Y=1 \mid X_1=x_1, X_2=x_2)$
	0	1	1
	0	0	1
	1	0	0

Is the model $P_{Y|X}$ interpretable?

object *features* *inference*

	X_1	X_2	$P_{Y\mid X}(Y=1 \mid X_1=x_1, X_2=x_2)$
	0	1	1
	0	0	1
	1	0	0

$$(X_1, X_2) = (0,0) \xrightarrow{\text{unknown model } P_{Y\mid X}} Y = 1$$

object *features* *inference*

	X_1	X_2	$P_{Y\|X}(Y=1 \mid X_1=x_1, X_2=x_2)$
	0	1	1
	0	0	1
	1	0	0

$$(X_1, X_2) = (0,0) \xrightarrow{\text{unknown model } P_{Y|X}} Y = 1$$

"translate" ↓ ↓ "translate"

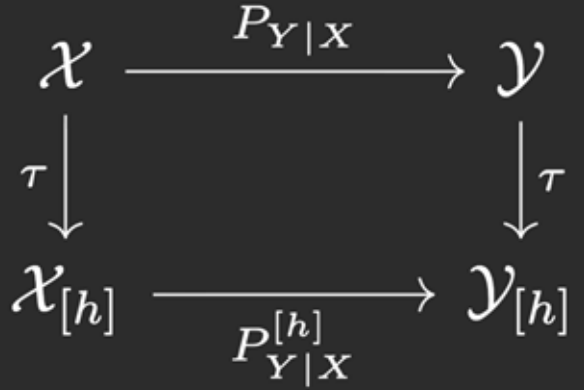
$$(X_{\texttt{one}}, X_{\texttt{red}}) = (\text{no}, \text{no}) \qquad Y_{\texttt{even}} = \text{yes}$$

object *features* *inference*

	X_1	X_2	$P_{Y\mid X}(Y=1 \mid X_1=x_1, X_2=x_2)$
	0	1	1
	0	0	1
	1	0	0

$$
\begin{array}{ccc}
(X_1, X_2) = (0,0) & \xrightarrow{\text{unknown model } P_{Y\mid X}} & Y = 1 \\
\Big\downarrow \text{“translate”} & & \Big\downarrow \text{“translate”} \\
(X_{\texttt{one}}, X_{\texttt{red}}) = (\text{no}, \text{no}) & \xrightarrow[\text{human model } P^{[h]}_{Y\mid X}]{} & Y_{\texttt{even}} = \text{yes}
\end{array}
$$

Symmetry 1. (**Inference Equivariance**) Inference $P_{Y|X}(Y \mid X = x)$ is *equivariant* w.r.t. a reference Hm under a translation $\tau : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}_{[h]} \times \mathcal{Y}_{[h]}$ if and only if it exists a model $P^{[h]}_{Y|X} : \mathcal{X}_{[h]} \to \mathbb{P}(Y)$ in Hm such that the following diagram commutes:



Why we need more symmetries?

- Naively verifying interpretability via inference equivariance is **intractable**
- Many sound translations might exist, but some translations are "**not sound**"
- Many models might exist, but some **may not satisfy desirable human properties**

Conclusion 1. A user's ability to predict a model's outputs is essential (but not enough) for the model $P_{Y|X}$ to be interpretable.

Interpretability Symmetries

Symmetry I
Inference equivariance

Symmetry II
Information invariance

Symmetry III
Concept invariance

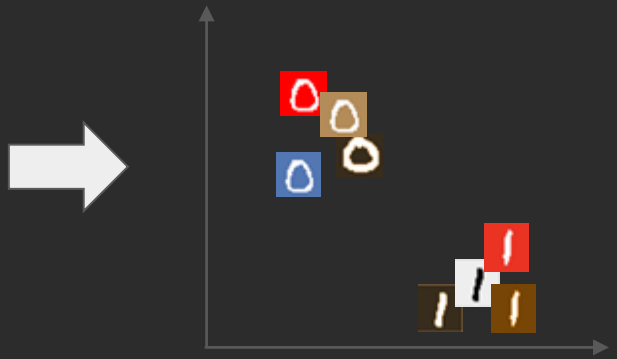
Symmetry IV
Structural invariance

"Inference equivariance is more tractable
if we compress the model without losing information"

X

X

Z

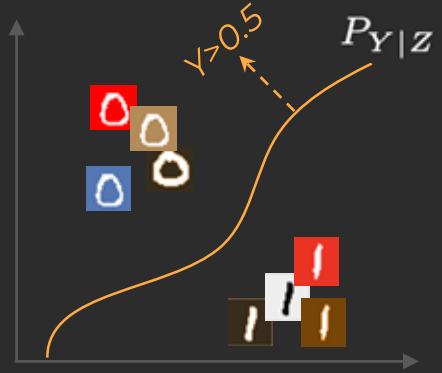


$$H(Z) \ll H(X)$$

compression

X

Z



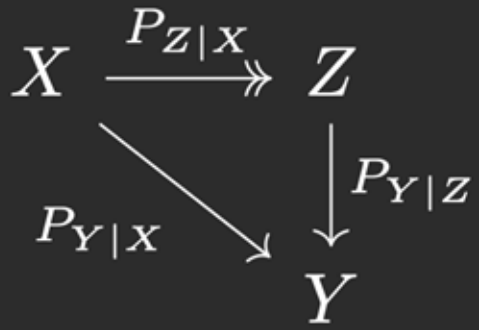
$$H(Z) \ll H(X)$$

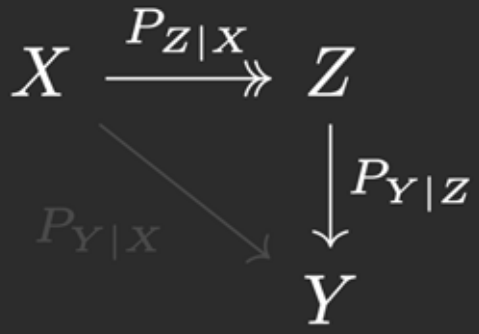
compression

$$I(Y;X) = (Y;Z)$$

preservation of information

Symmetry 2. **(Information Invariance)** Given a model $P_{Y|X}$, the mutual information $I(Y;X)$ is invariant under marginalisation $P_{Y|Z} \circ P_{Z|X}$ with $H(Z) \ll H(X)$ if the following diagram commutes:





$$I(Y;X \mid Z) = 0$$

conditional independence

X is irrelevant to understand inference on Y

Conclusion 2. Verifying inference equivariance for $P_{Y|Z}$ is equivalent but more tractable than for $P_{Y|X}$.

Why we need more symmetries?

- ✓ Naively verifying interpretability via inference equivariance is **intractable**
- ❑ Many sound translations might exist, but some translations are "**not sound**"
- ❑ Many models might exist, but some **may not satisfy desirable human properties**

Interpretability Symmetries

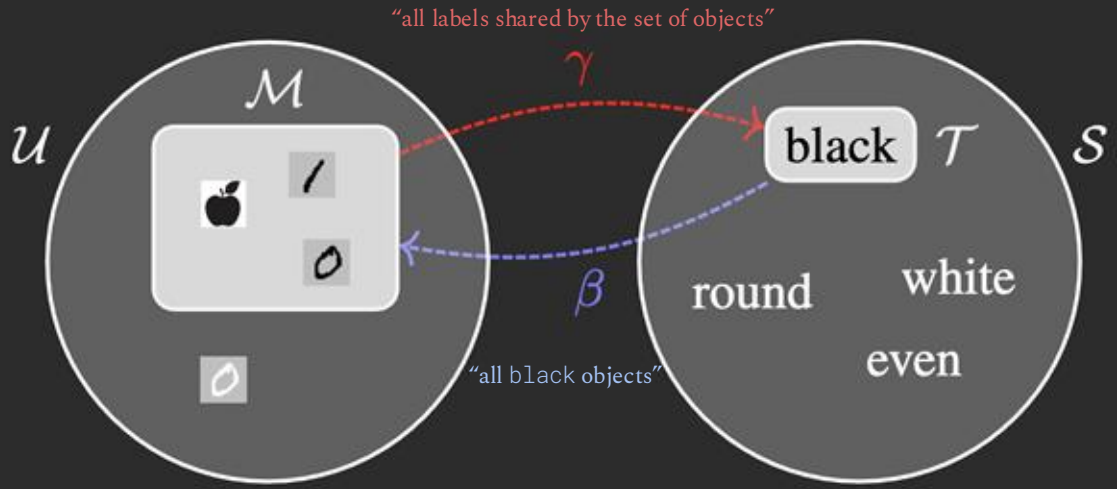
Symmetry I
Inference equivariance

Symmetry II
Information invariance

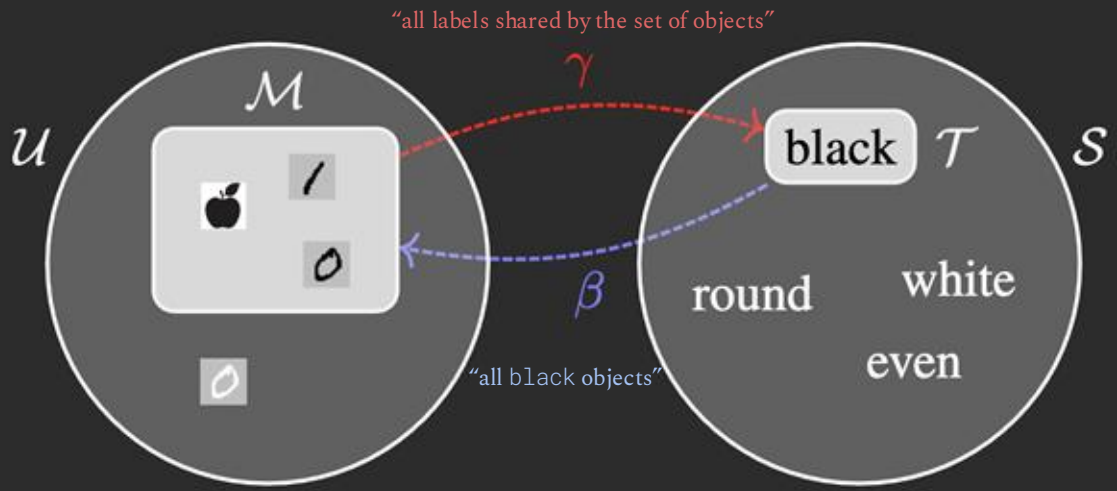
Symmetry III
Concept invariance

Symmetry IV
Structural invariance

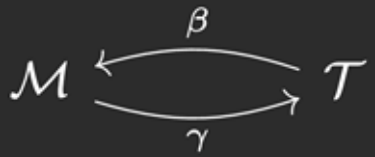
“Sound translations preserve concepts
*(*red *for the model has the same meaning of* red *for a human)”*



[1] Goguen "What is a concept?" International Conference on Conceptual Structures (2005).
[2] Ganter at al "Formal concept analysis" Springer (1999).

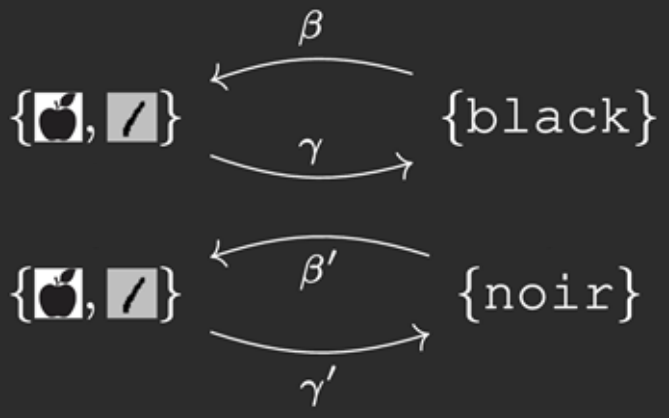


Concept — Set of objects and sentences satisfying "**closure**" condition:

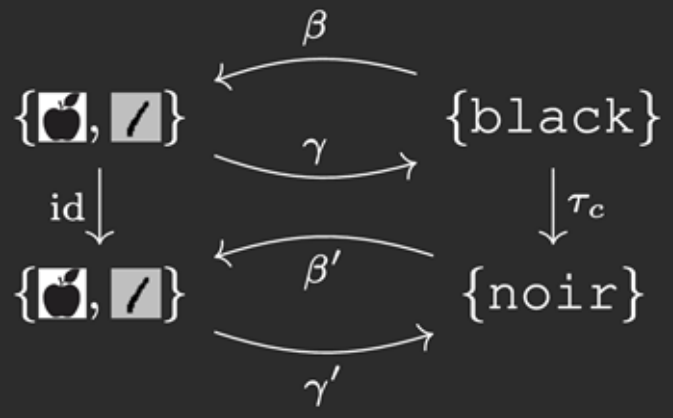


[1] Goguen "What is a concept?" International Conference on Conceptual Structures (2005).
[2] Ganter at al "Formal concept analysis" Springer (1999).

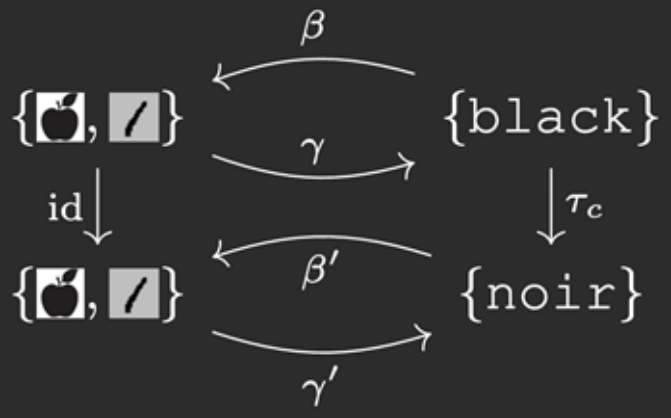
Sound translations preserve closure:



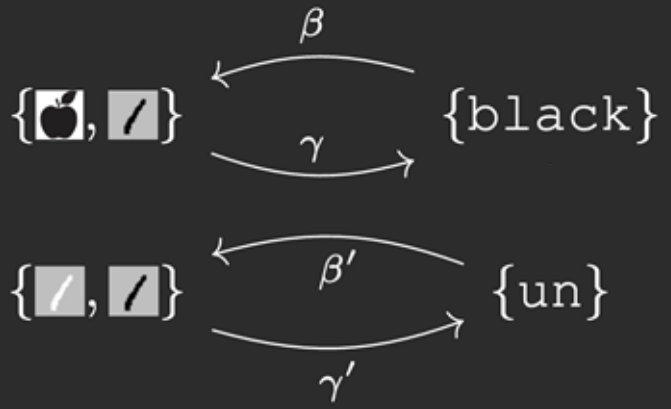
Sound translations preserve closure:



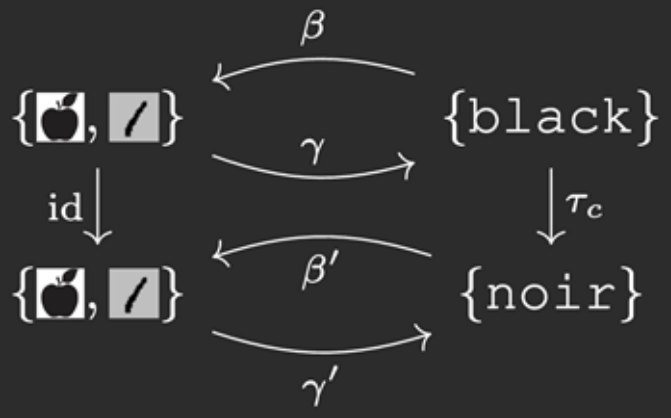
Sound translations preserve closure:



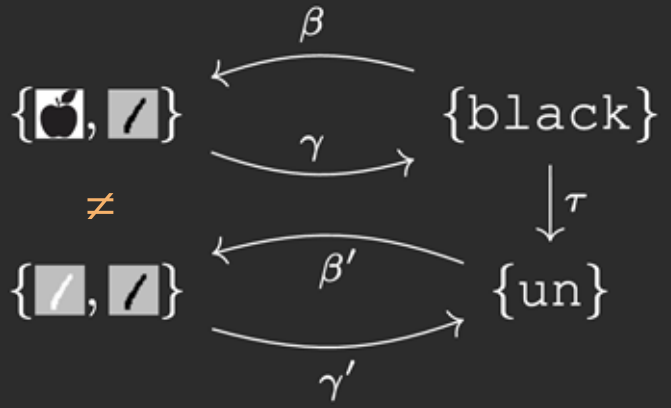
"**Unsound**" translations do not preserve closure:



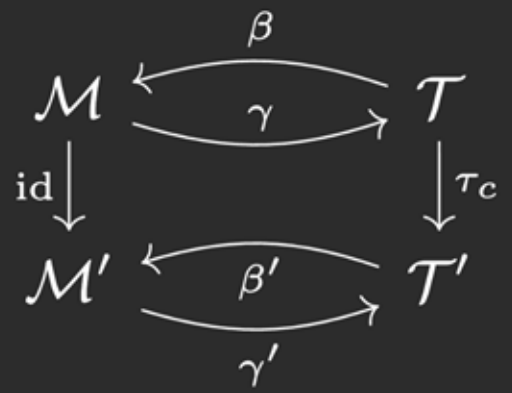
Sound translations preserve closure:

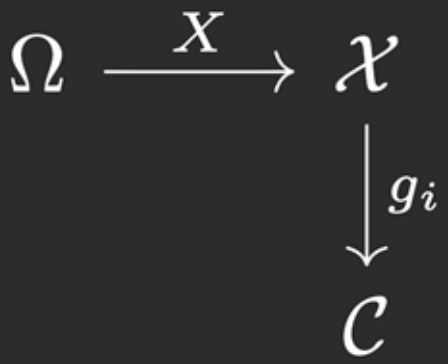


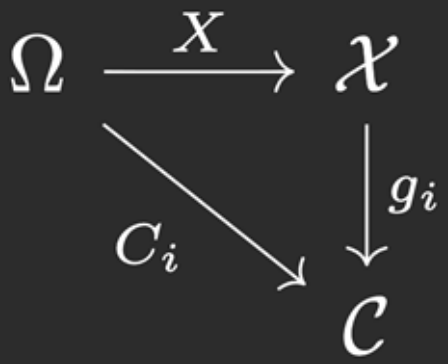
"**Unsound**" translations do not preserve closure:



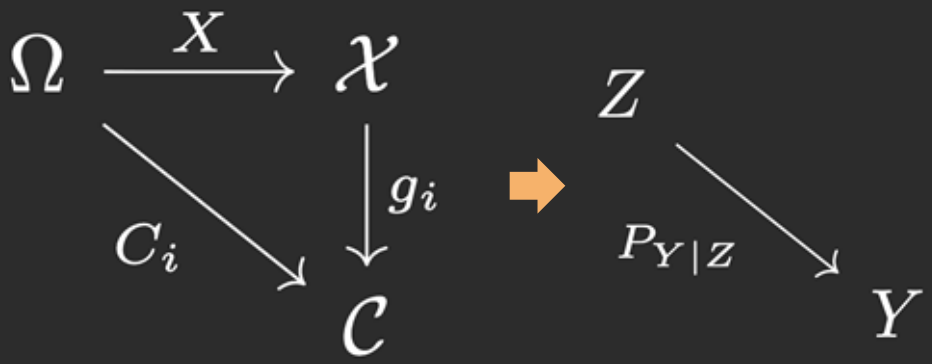
Symmetry 3. **(Concept Closure Invariance)** Concept closure is invariant under a sentence translation function $\tau_c : \mathcal{T} \to \mathcal{T}'$ if, for any pair of concepts $C = (\mathcal{T}, \mathcal{M})$ and $C' = (\mathcal{T}', \mathcal{M}')$, the following diagram commutes for all objects $\omega \in \mathcal{M}$:



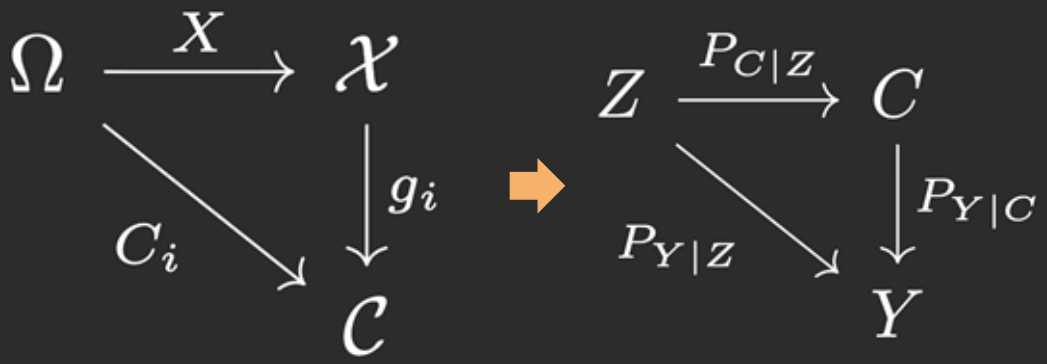

Ω
X
$\mathcal{X}$
g_i
$\mathcal{C}$



concept random variable

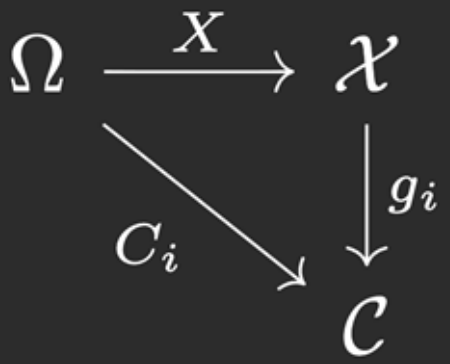


concept random variable

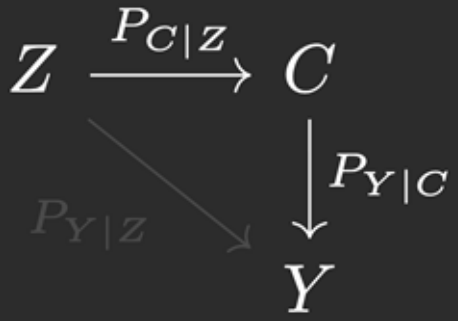


concept random variable

marginalisation



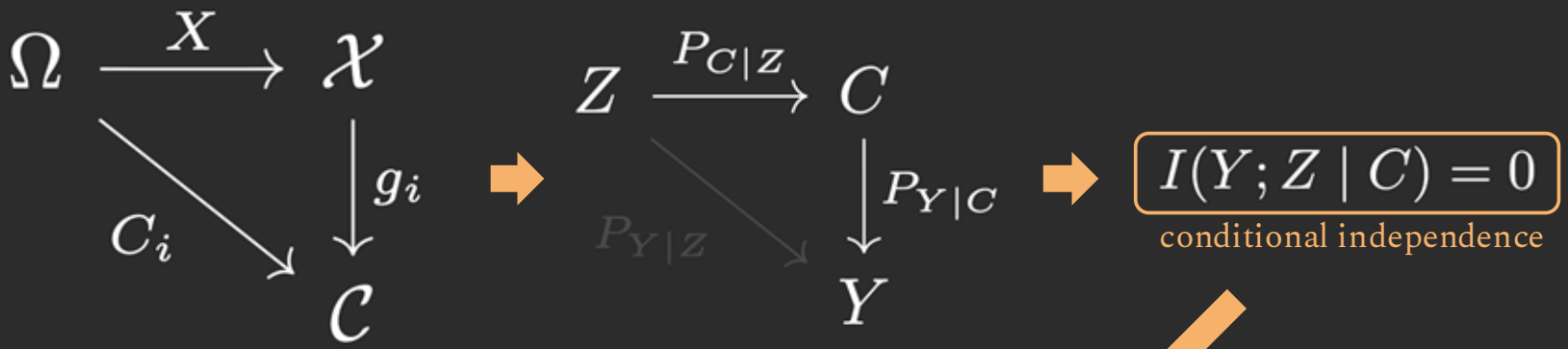
concept random variable



marginalisation

$$I(Y; Z \mid C) = 0$$

conditional independence



Z is irrelevant to understand inference on Y

Conclusion 3. Verifying inference equivariance for $P_{Y|C}$ is equivalent to that for $P_{Y|Z}$, but it supports the use of sound translations.

Why we need more symmetries?

- ✓ Naively verifying interpretability via inference equivariance is **intractable**
- ✓ Many sound translations might exist, but some translations are "**not sound**"
- ❑ Many models might exist, but some **may not satisfy desirable human properties**

Interpretability Symmetries

Symmetry I
Inference equivariance

Symmetry II
Information invariance

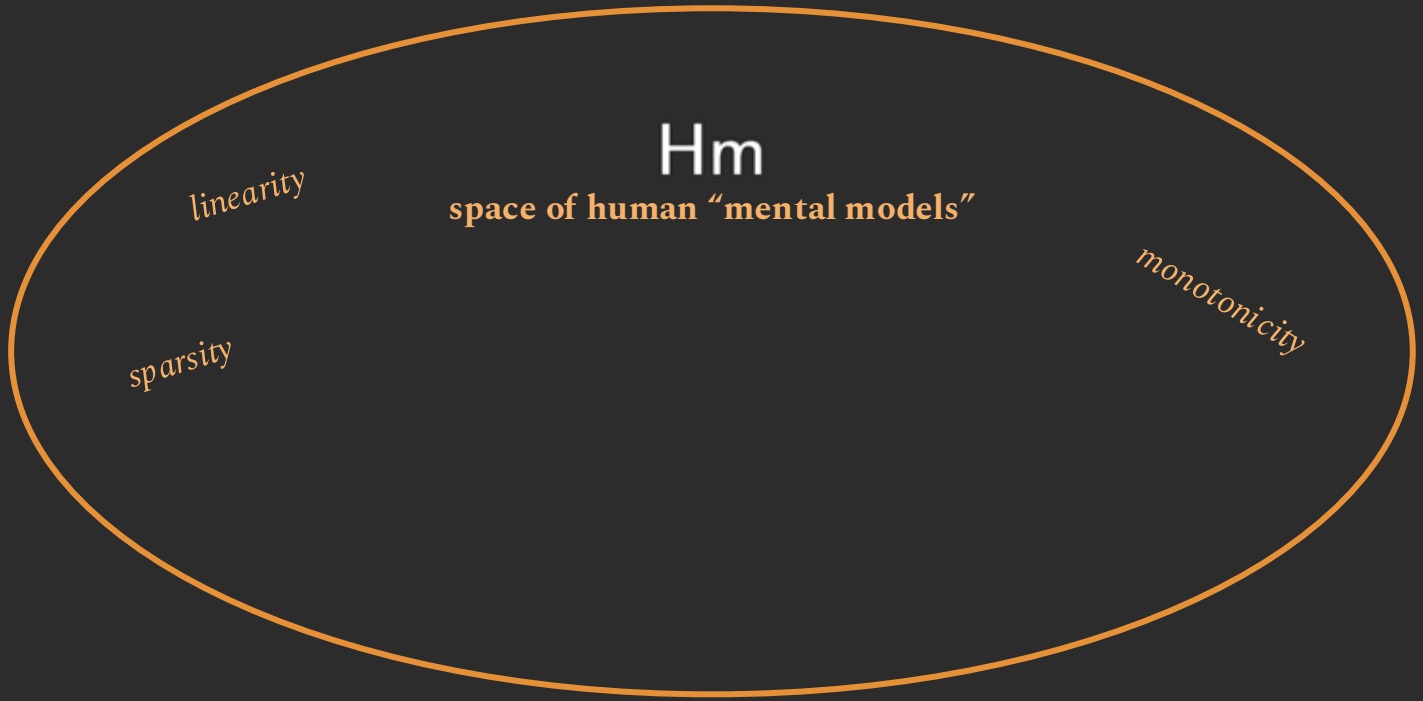
Symmetry III
Concept invariance

Symmetry IV
Structural invariance

"A model must be drawn from a hypothesis class the user can reason about"

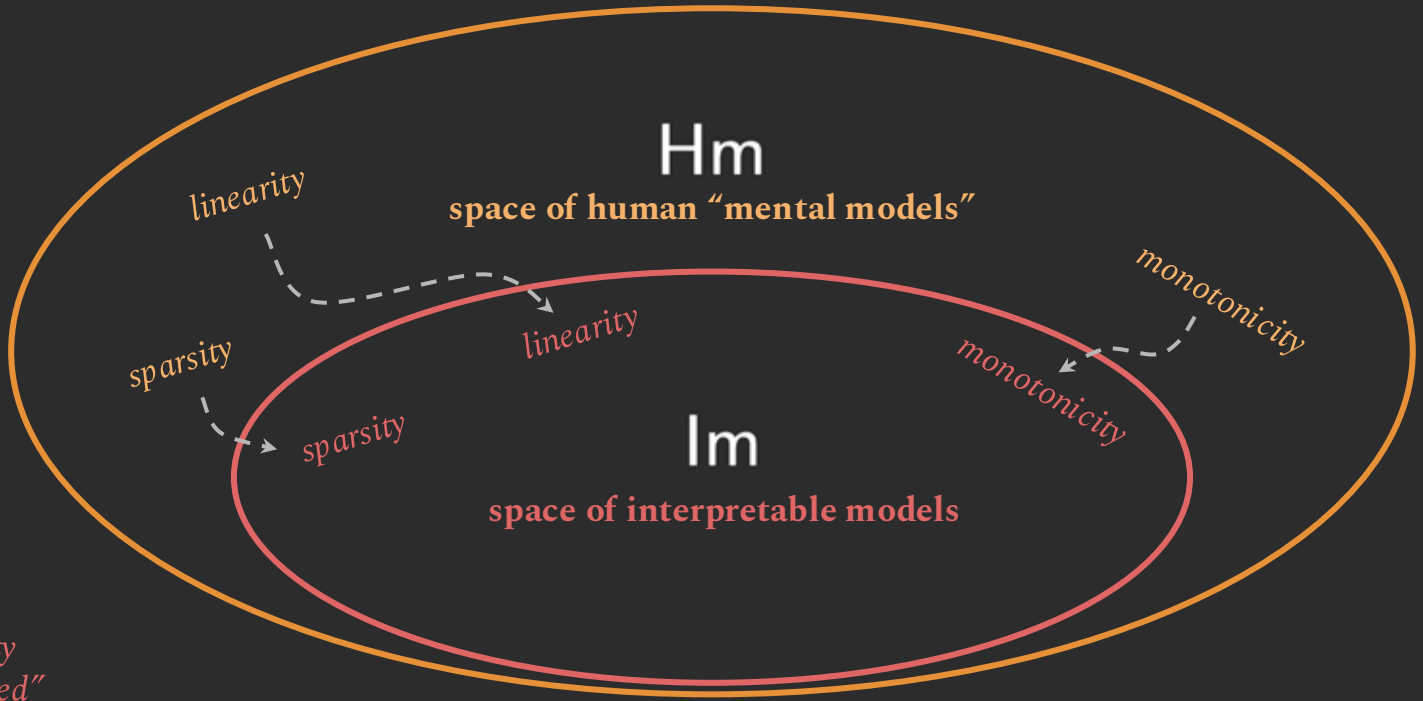
Stoch

hypothesis space of probabilistic models



Stoch

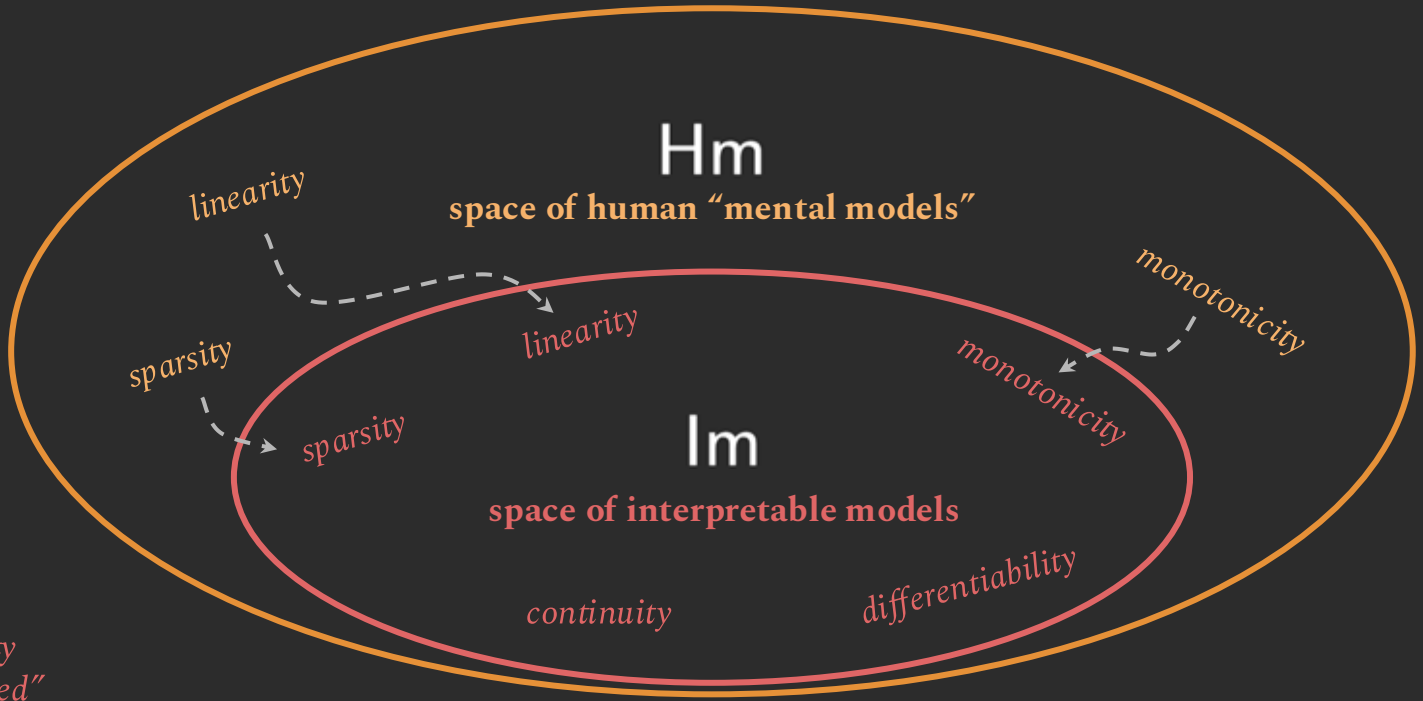
hypothesis space of probabilistic models



sparsity,
linearity,
monotonicity
are "preserved"

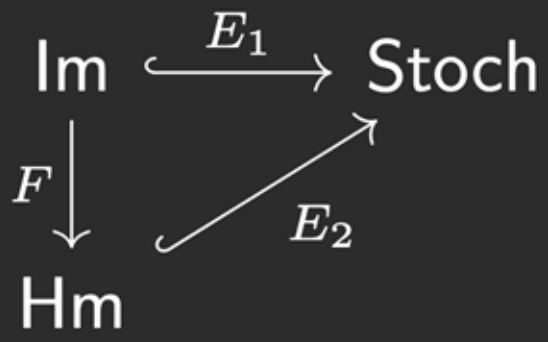
Stoch

hypothesis space of probabilistic models



sparsity, linearity, monotonicity are "preserved"

Symmetry 4. **(Structural Invariance)** The structural properties of models in Im are invariant under the functor $F : \mathsf{Im} \to \mathsf{Hm}$ if and only if there exist two injective functors $E_1 : \mathsf{Im} \to \mathsf{Stoch}$ and $E_2 : \mathsf{Hm} \to \mathsf{Stoch}$ such that the following diagram commutes:



Conclusion 4. If $P_{Y|C}$ is compatible with the hypothesis space of user's mental models, then the user can internally simulate the model to verify inference equivariance.

Why we need more symmetries?

- ✓ Naively verifying interpretability via inference equivariance is **intractable**
- ✓ Many sound translations might exist, but some translations are "**not sound**"
- ✓ Many models might exist, but some **may not satisfy desirable human properties**

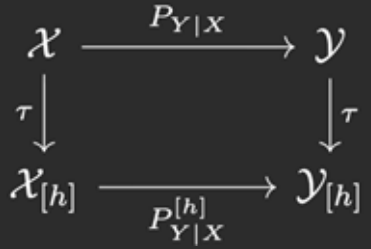
Interpretability Symmetries

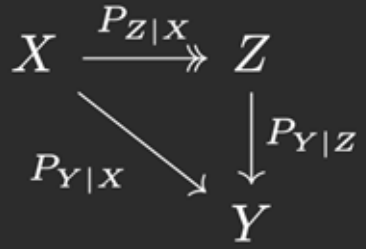
Symmetry I
Inference equivariance

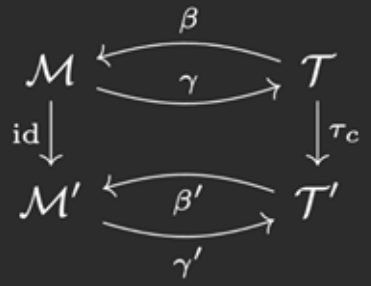
Symmetry II
Information invariance

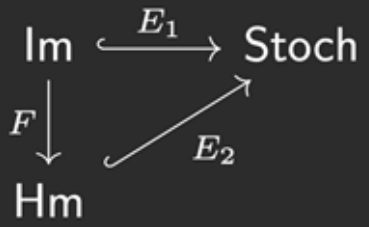
Symmetry III
Concept invariance

Symmetry IV
Structural invariance









Tutorial Outline

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences

Q&A + 30 min break

IV. Designing Concept Representations

V. Designing Task Predictors

VI. Human-AI Interaction

VII. Conclusion

Final Q&A

How to represent models and inferences?

Picturing probabilistic models and inferences

Probabilistic models

Probabilistic inference

Single-variable model

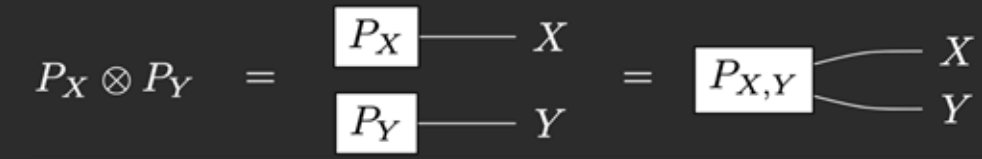
```python
import torch_concepts as pyc

x_var = pyc.Variable(
    concepts=["x_image"],
    parents=[],
    distribution=pyc.distributions.Delta
)
```

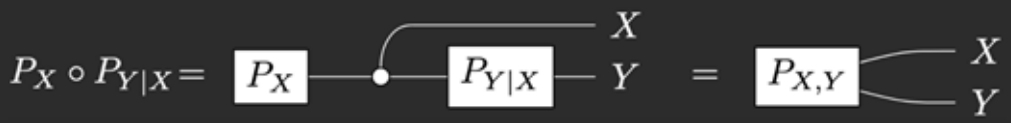
Joint model

$$P_X \otimes P_Y = \begin{array}{l} P_X \text{ — } X \\ P_Y \text{ — } Y \end{array} = P_{X,Y} \begin{array}{l} \text{— } X \\ \text{— } Y \end{array}$$

```python
x_var = pyc.Variable(
    concepts=["x_image"],
    parents=[],
    distribution=pyc.distributions.Delta
)
y_var = pyc.Variable(
    concepts=["y_cat"],
    parents=[],
    distribution=torch.distributions.RelaxedBernoulli
)
```

Joint model

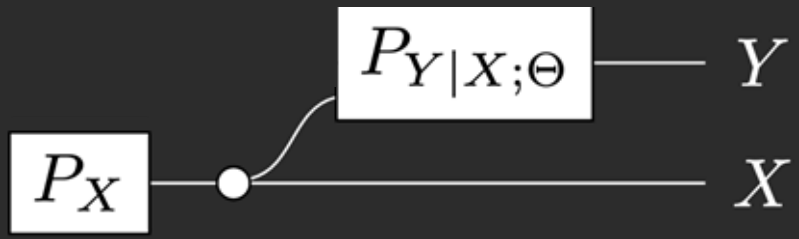





```
x_var = pyc.Variable(
    concepts=["x_image"],
    parents=[],
    distribution=pyc.distributions.Delta
)
y_var = pyc.Variable(
    concepts=["y_cat"],
    parents=[],
    distribution=torch.distributions.RelaxedBernoulli
)
```

```
x_var = pyc.Variable(
    concepts=["x_image"],
    parents=[],
    distribution=pyc.distributions.Delta
)
y_var = pyc.Variable(
    concepts=["y_cat"],
    parents=["x_image"],
    distribution=torch.distributions.RelaxedBernoulli
)
```

Parametric model

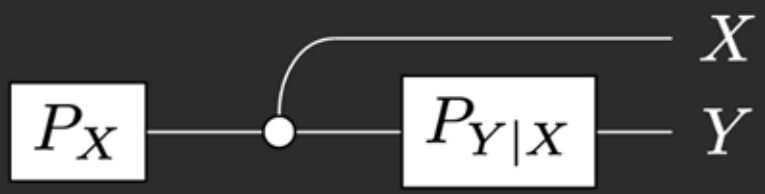



```python
x_var = pyc.Variable(
    concepts=["x_image"],
    parents=[],
    distribution=pyc.distributions.Delta
)
y_var = pyc.Variable(
    concepts=["y_cat"],
    parents=["x_image"],
    distribution=torch.distributions.RelaxedBernoulli
)
x_cpd = pyc.nn.ParametricCPD(
    concepts=["x_image"],
    parametrization=torch.nn.Identity()
)
y_cpd = pyc.nn.ParametricCPD(
    concepts=["y_cat"],
    parametrization=torch.nn.Linear(x_size, y_size)
)
parametric_model = pyc.nn.ProbabilisticModel(
    variables=[x_var, y_var],
    parametric_cpds=[x_cpd, y_cpd]
)
```

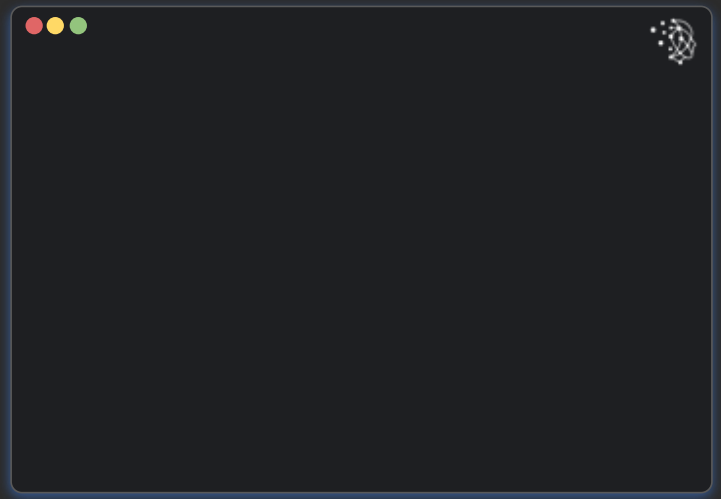
$$P_{Y|X=x}(y) = P_{Y|X}(y \mid x)$$

Deterministic inference

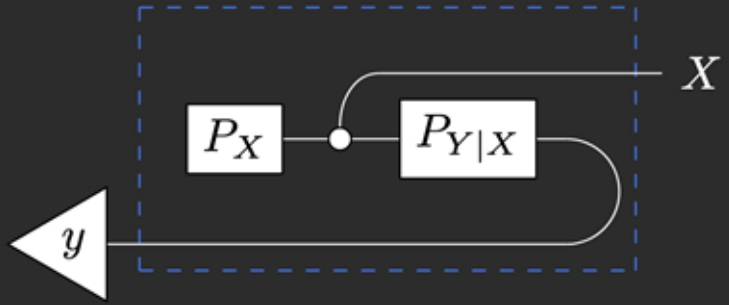
```python
parametric_model = pyc.nn.ProbabilisticModel(
    variables=[x_var, y_var],
    parametric_cpds=[x_cpd, y_cpd]
)
inference_engine = pyc.nn.DeterministicInference(
    probabilistic_model=parametric_model
)
y_pred = inference_engine.query(
    query_concepts=["y_cat"],
    evidence={"x_image": x}
)
```



Posterior inference



Posterior inference

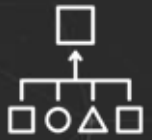


$$P_{X|Y=y}(x) = \frac{P_{Y|X}(y \mid x)P_X(x)}{\int P_{Y|X}(y \mid x)P_X(x)\mathrm{d}x}$$

```python
x_cpd = pyc.nn.ParametricCPD(
    concepts=["x_token"],
    parametrization=torch.nn.Identity()
)
y_cpd = pyc.nn.ParametricCPD(
    concepts=["y_cat"],
    parametrization=torch.nn.Linear(x_size, y_size)
)
parametric_model = pyc.nn.ProbabilisticModel(
    variables=[x_var, y_var],
    parametric_cpds=[x_cpd, y_cpd]
)
inference_engine = pyc.nn.PosteriorInference(
    probabilistic_model=parametric_model
)
y_pred = inference_engine.query(
    query_concepts=["x_token"],
    evidence={"x_token": x, "y_cat": y_cat},
    algorithm="belief_propagation"
)
```

How to represent interpretable models?

Interpretable models



Category of (open) interpretable models

Blueprint for interpretable models

Category of (open) interpretable models

Objects

C

Concept variables

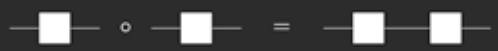
Processes

Concept-based process

Copy process

Discard process

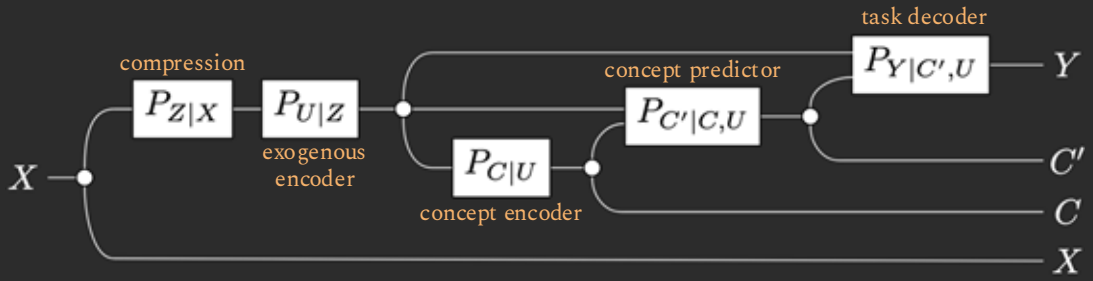
Composition rules



Sequential

Parallel

Blueprint for interpretable models




```python
# manifold reparametrization
z_var = pyc.Variable(
    concepts=["z_latent"], parents=[], distribution=Delta
)
z_cpd = pyc.nn.ParametricCPD(
    concepts=["z_latent"], parametrization=Linear(x_size, z_size)
)
# concept encoder
c_cat = pyc.EndogenousVariable(
    concepts=["c_cat"], parents=["z_latent"], distribution=RelaxedBernoulli
)
c_cat_cpd = pyc.nn.ParametricCPD(
    concepts=["c_cat"], parametrization=LinearZC(z_size, c_cat_size)
)
# concept predictor
c_animal = pyc.EndogenousVariable(
    concepts=["c_animal"], parents=["c_cat"], distribution=RelaxedBernoulli
)
c_animal_cpd = pyc.nn.ParametricCPD(
    concepts=["c_animal"], parametrization=LinearCC(c_cat_size, c_animal_size)
)
# task decoder
y_task = pyc.Variable(
    concepts=["y_task"], parents=["c_animal"], distribution=RelaxedOneHotCategorical
)
y_task_cpd = pyc.nn.ParametricCPD(
    concepts=["y_task"], parametrization=Linear(c_animal_size, y_size)
)
# probabilistic model
parametric_model = pyc.nn.ProbabilisticModel(
    variables=[z_var, c_cat, c_animal, y_task],
    parametric_cpds=[z_cpd, c_cat_cpd, c_animal_cpd, y_task_cpd]
)
```

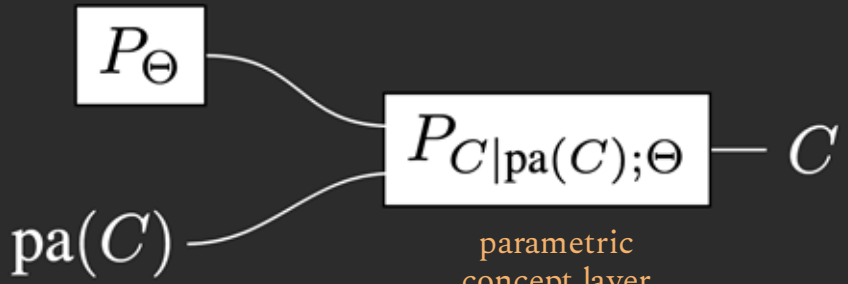
How to represent interpretable inference?

Inference with interpretable models

Alignment

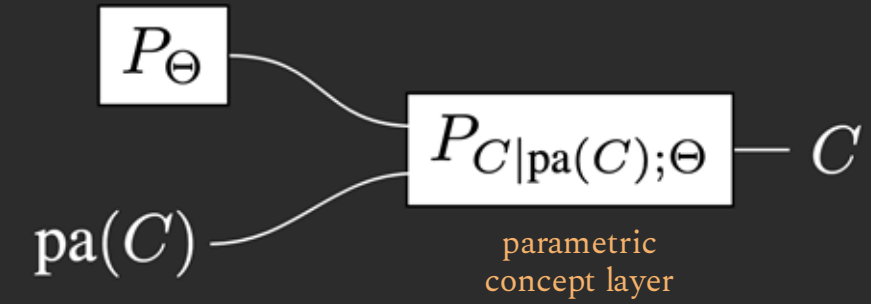
Interventional inference

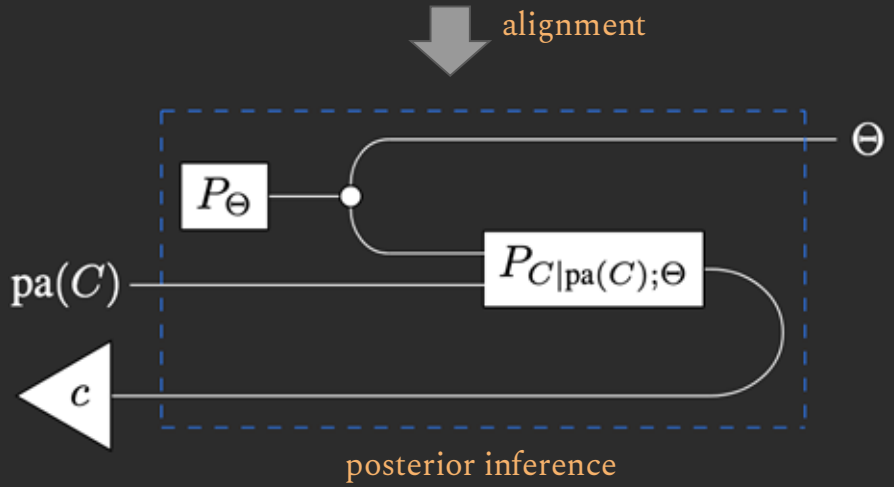
Counterfactual inference



Alignment

```python
# model & training setup
parametric_model = pyc.nn.ProbabilisticModel(
    variables=[z_var, c_cat, c_animal],
    parametric_cpds=[z_cpd, c_cat_cpd, c_animal_cpd]
)
```

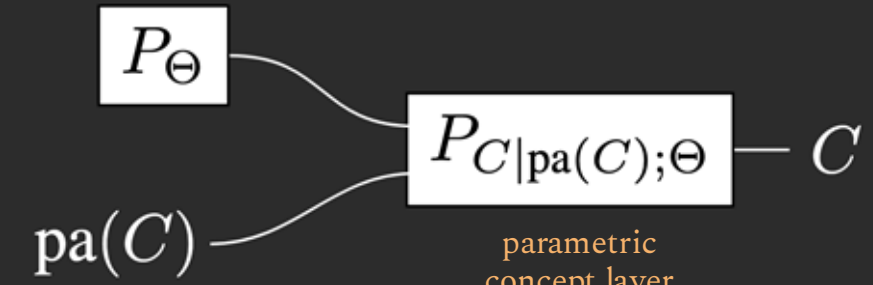


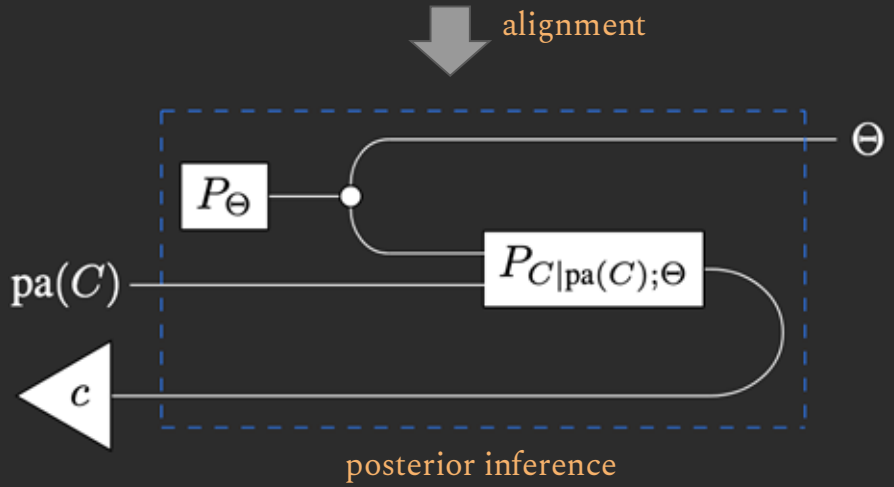


$$P_{\Theta|C=c,\mathrm{pa}(C)}(\theta) = \frac{P_{C|\mathrm{pa}(C);\Theta}(c|\mathrm{pa}(C);\theta)P_\Theta(\theta)}{\int_\Theta P_{C|\mathrm{pa}(C);\Theta}(c|\mathrm{pa}(C);\theta)P_\Theta(\theta)\mathrm{d}\theta}$$

Alignment

```
# model & training setup
parametric_model = pyc.nn.ProbabilisticModel(
    variables=[z_var, c_cat, c_animal],
    parametric_cpds=[z_cpd, c_cat_cpd, c_animal_cpd]
)
```





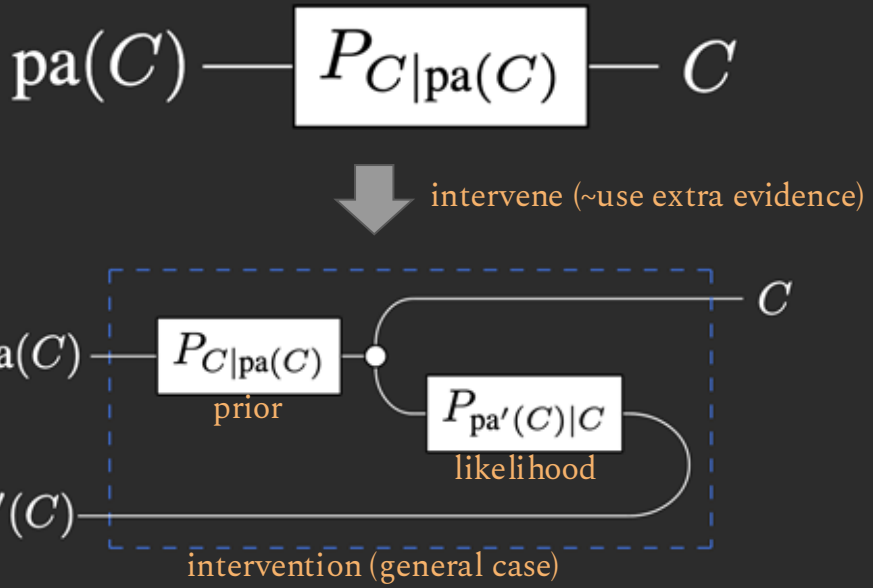
$$P_{\Theta|C=c,\mathrm{pa}(C)}(\theta) = \frac{P_{C|\mathrm{pa}(C);\Theta}(c|\mathrm{pa}(C);\theta)P_\Theta(\theta)}{\int_\Theta P_{C|\mathrm{pa}(C);\Theta}(c|\mathrm{pa}(C);\theta)P_\Theta(\theta)\mathrm{d}\theta}$$

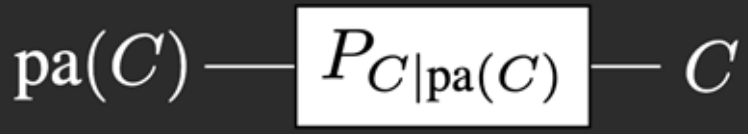
Alignment

```python
# model & training setup
parametric_model = pyc.nn.ProbabilisticModel(
    variables=[z_var, c_cat, c_animal],
    parametric_cpds=[z_cpd, c_cat_cpd, c_animal_cpd]
)
optimizer = torch.optim.AdamW(parametric_model.parameters(), lr=0.01)
loss_fn = torch.nn.BCELoss()
parametric_model.train()

# inference
inference_engine = pyc.nn.AncestralSamplingInference(
    probabilistic_model=parametric_model,
    temperature=0.1
)
c_animal_pred = inference_engine.query(
    query_concepts=["c_animal"],
    evidence={"z_latent": x}
)

# compute MLE and update parameters
optimizer.zero_grad()
concept_loss = loss_fn(c_animal_pred, c_animal_true)
concept_loss.backward()
optimizer.step()
```

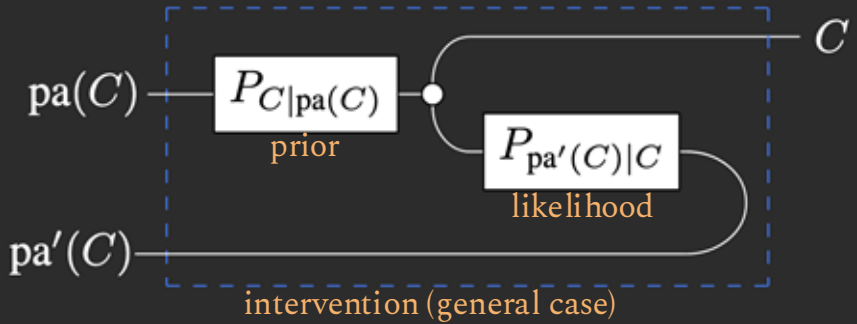
Interventional inference

Interventional inference



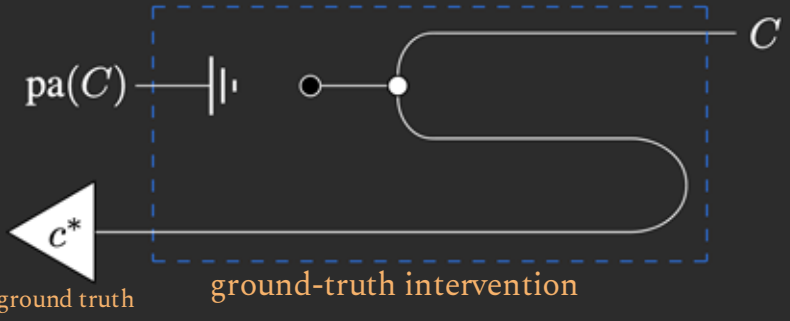


intervene (~use extra evidence)



intervention (general case)

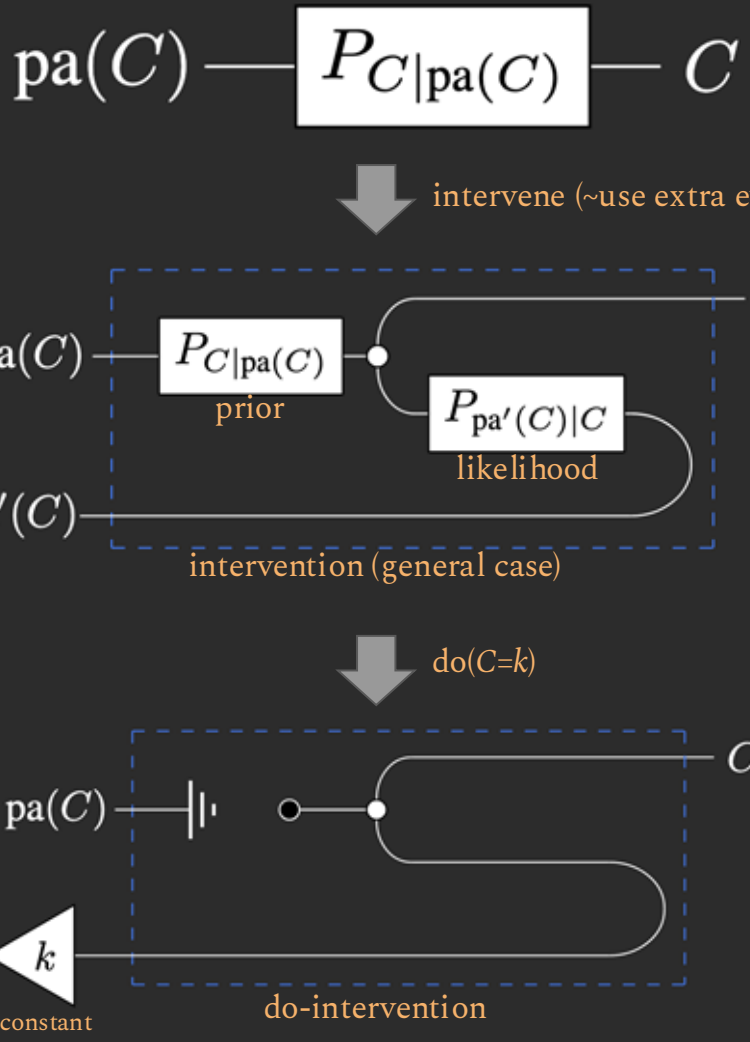
gt(C=c^*)



ground-truth intervention

Ground-truth intervention

```python
smoking_strategy = GroundTruthIntervention(
    model=parametric_model.parametric_cpds,
    ground_truth=smoking_true,
)
with intervention(
    policies=UniformPolicy(out_features=1),  # intervene on all samples
    strategies=smoking_strategy,
    target_concepts=["smoking"]
):
    intervened_results = inference_engine.query(
        query_concepts=["genotype", "smoking", "tar", "cancer"],
        evidence={'exogenous': x}
    )
    cancer_ground_truth_smoking = intervened_results[:, 3]
```

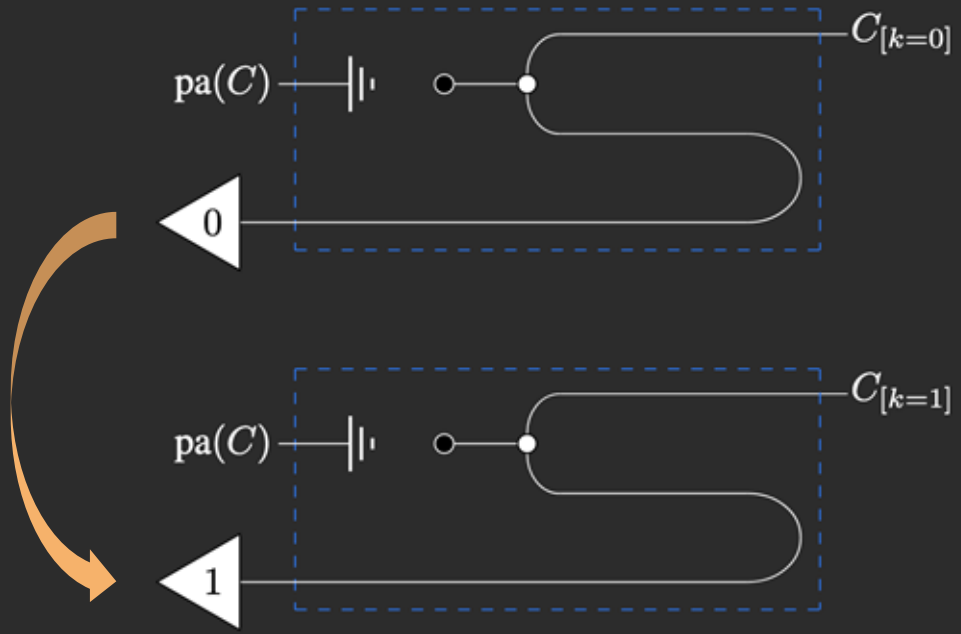


Do-intervention

k=0

```python
# Intervention 1: Force smoking to 0 (prevent smoking)
smoking_strategy_0 = DoIntervention(
    model=parametric_model.parametric_cpds,
    constants=0.0
)
with intervention(
    policies=UniformPolicy(out_features=1),   # intervene on all samples
    strategies=smoking_strategy_0,
    target_concepts=["smoking"]
):
    intervened_results_0 = inference_engine.query(
        query_concepts=["genotype", "smoking", "tar", "cancer"],
        evidence={'exogenous': x}
    )
    cancer_do_smoking_0 = intervened_results_0[:, 3]
```

Causal effect estimation



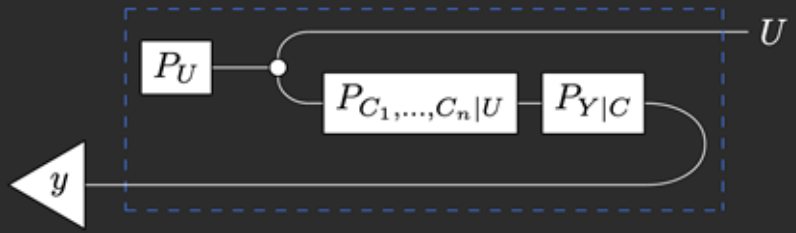
k=0

k=1

```python
# Intervention 1: Force smoking to 0 (prevent smoking)
smoking_strategy_0 = DoIntervention(
    model=parametric_model.parametric_cpds,
    constants=0.0
)
with intervention(
    policies=UniformPolicy(out_features=1),
    strategies=smoking_strategy_0,
    target_concepts=["smoking"]
):
    intervened_results_0 = inference_engine.query(
        query_concepts=["genotype", "smoking", "tar", "cancer"],
        evidence={'exogenous': x}
    )
    cancer_do_smoking_0 = intervened_results_0[:, 3]

# Intervention 2: Force smoking to 1 (promote smoking)
smoking_strategy_1 = DoIntervention(
    model=parametric_model.parametric_cpds,
    constants=1.0
)
with intervention(
        policies=UniformPolicy(out_features=1),
        strategies=smoking_strategy_1,
        target_concepts=["smoking"]
):
    intervened_results_1 = inference_engine.query(
        query_concepts=["genotype", "smoking", "tar", "cancer"],
        evidence={'exogenous': x}
    )
    cancer_do_smoking_1 = intervened_results_1[:, 3]

# causal effect estimation
ace_cancer_do_smoking = cace_score(cancer_do_smoking_0, cancer_do_smoking_1)
```

Counterfactual inference

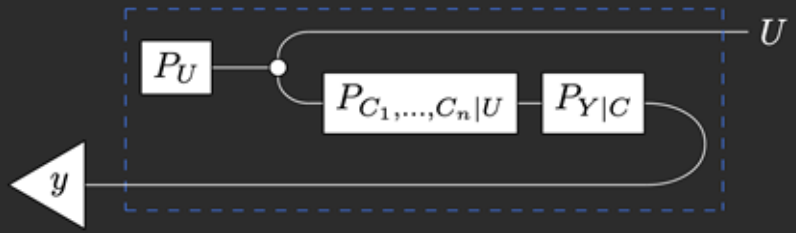
Step 1) Abduction (posterior on exogenous given evidence)



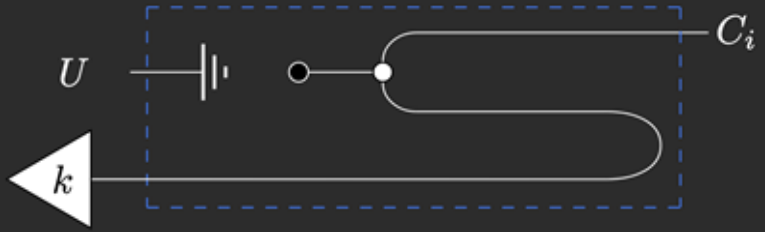

```python
# Step 1: Posterior inference to get exogenous variables given cancer outcome
inference_engine = pyc.nn.PosteriorInference(
    probabilistic_model=parametric_model
)
exogenous_post = inference_engine.query(
    query_concepts=["exogenous"],
    evidence={"y_cancer": y_cancer},
    algorithm="belief_propagation"
)
```

Counterfactual inference

Step 1) Abduction (posterior on exogenous given evidence)



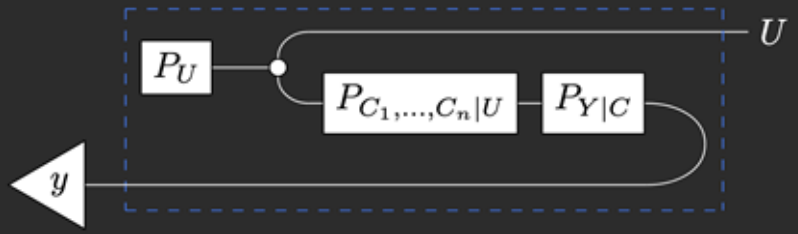
Step 2) Action (intervene on i-th concept)



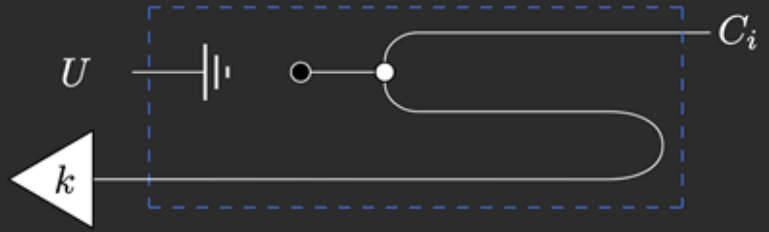

```python
# Step 1: Posterior inference to get exogenous variables given cancer outcome
inference_engine = pyc.nn.PosteriorInference(
    probabilistic_model=parametric_model
)
exogenous_post = inference_engine.query(
    query_concepts=["exogenous"],
    evidence={"y_cancer": y_cancer},
    algorithm="belief_propagation"
)

# Step 2: Force smoking to 1
smoking_strategy_1 = DoIntervention(
    model=parametric_model.parametric_cpds,
    constants=1.0
)
inference_engine = pyc.nn.AncestralSamplingInference(
    probabilistic_model=parametric_model
)
with intervention(
    policies=UniformPolicy(out_features=1),
    strategies=smoking_strategy_1,
    target_concepts=["smoking"]
):
    intervened_results_1 = inference_engine.query(
        query_concepts=["genotype", "smoking", "tar", "cancer"],
        evidence={'exogenous': exogenous_post}
    )
    cancer_do_smoking_1 = intervened_results_1[:, 3]
```

Counterfactual inference

Step 1) Abduction (posterior on exogenous given evidence)



Step 2) Action (intervene on i-th concept)



Step 3) Prediction (infer the value of Y in intervened model)

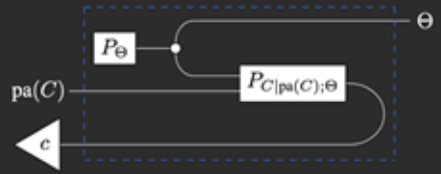
```python
# Step 1: Posterior inference to get exogenous variables given cancer outcome
inference_engine = pyc.nn.PosteriorInference(
    probabilistic_model=parametric_model
)
exogenous_post = inference_engine.query(
    query_concepts=["exogenous"],
    evidence={"y_cancer": y_cancer},
    algorithm="belief_propagation"
)

# Step 2: Force smoking to 1
smoking_strategy_1 = DoIntervention(
    model=parametric_model.parametric_cpds,
    constants=1.0
)
inference_engine = pyc.nn.AncestralSamplingInference(
    probabilistic_model=parametric_model
)
with intervention(
    policies=UniformPolicy(out_features=1),
    strategies=smoking_strategy_1,
    target_concepts=["smoking"]
):
    intervened_results_1 = inference_engine.query(
        query_concepts=["genotype", "smoking", "tar", "cancer"],
        evidence={'exogenous': exogenous_post}
    )
    cancer_do_smoking_1 = intervened_results_1[:, 3]

    # Step 3: Query cancer outcome under the intervention
    y_cancer_counterfactual = inference_engine.query(
        query_concepts=["y_cancer"],
        evidence={"cancer": cancer_do_smoking_1,
                  "exogenous": exogenous_post}
    )
```

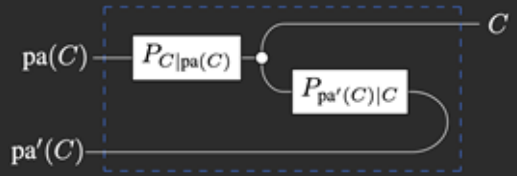
Inference with interpretable models

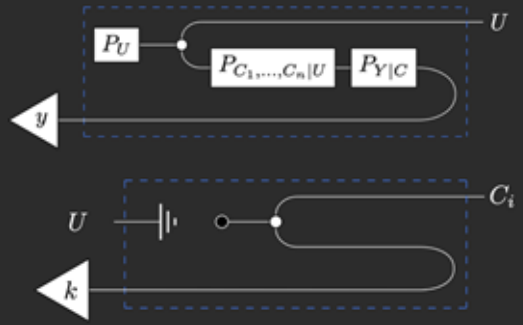
Alignment

Interventional inference

Counterfactual inference





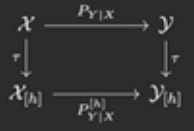


Conclusion 5. Alignment, interventional, and counterfactual inference are all forms of Bayesian inversion.

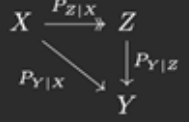
First Q&A

Interpretability Symmetries

Symmetry I
Inference equivariance

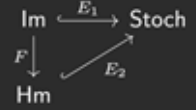


Symmetry II
Information invariance

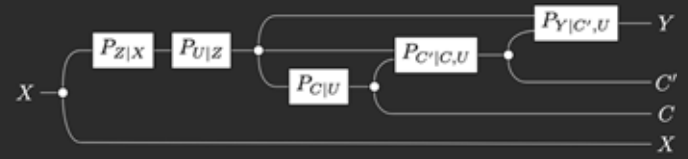


Symmetry III
Concept invariance

Symmetry IV
Structural invariance



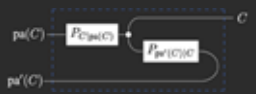
Blueprint for interpretable models



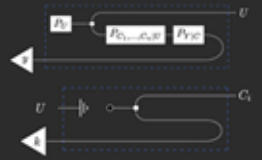
Inference with interpretable models

Alignment

Interventional inference

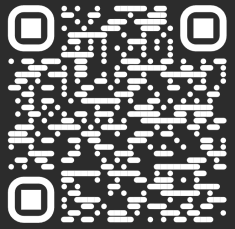


Counterfactual inference

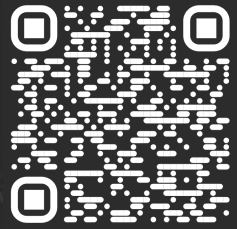


First Q&A + Break Time

See you back at 10:45 am!



interpretabledeeplearning.github.io



Part 2 of Tutorial:
Interpretable Model Instantiations

Our Story So Far

Our Story So Far

I. Introduction

Why do we need interpretability?

How is interpretability defined?

Our Story So Far

I. Introduction

II. Interpretability Symmetries

Symmetry I
Inference equivariance

Symmetry II
Information invariance

Symmetry III
Concept invariance

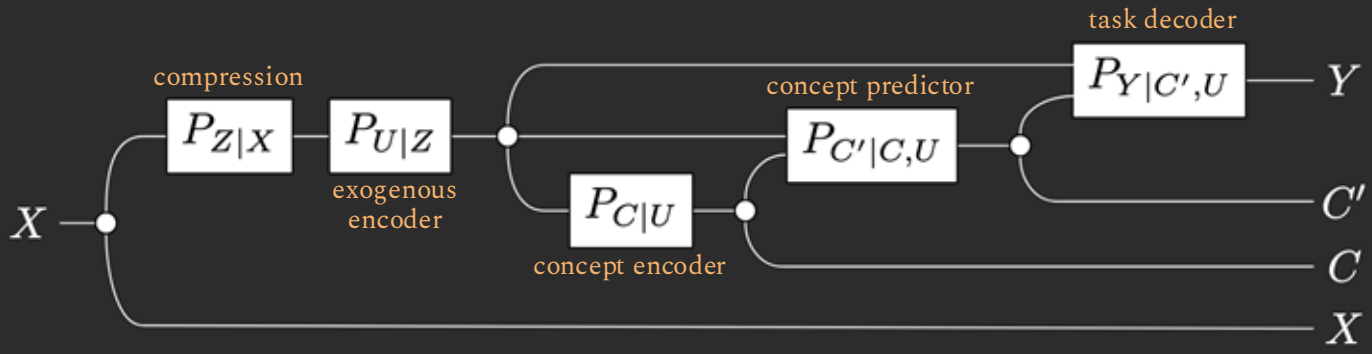
Symmetry IV
Structural invariance

Our Story So Far

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences



What's in line for the second half?

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences

Q&A + 30 min break

IV. Instantiating Concept Encoders

V. Instantiating Label Predictors

VI. Human-AI Interaction

VII. Conclusion

Final Q&A

} Instantiations of interpretable models

What do we (measurably) want from an interpretable model?

Desideratum #1: Task Fidelity

$$P\Big(Y = \texttt{Zebra} \mid X = \Big) = 1 \qquad P\Big(Y = \texttt{Mole} \mid X = \Big) = 1$$

The model should be able to accurately predict the downstream task P(Y | X)

[1] Espinosa Zarlenga et al. "Avoiding Leakage Poisoning: Concept Interventions Under Distribution Shifts," (2025).

Desideratum #2: Explanation Fidelity

$P\left(C'_{\text{stripes}} = \texttt{True} \mid X = \right) = 1$ 

$P\left(C'_{\text{small}} = \texttt{True} \mid X = \right) = 0$ 

$\vdots \quad \vdots \quad \vdots$

$P\left(C'_{\text{white}} = \texttt{True} \mid X = \right) = 1$ 

$P\left(C'_{\text{brown}} = \texttt{True} \mid X = \right) = 0$ 

$P\left(C'_{\text{stripes}} = \texttt{True} \mid X = \right) = 0$ 

$P\left(C'_{\text{small}} = \texttt{True} \mid X = \right) = 1$ 

$\vdots \quad \vdots \quad \vdots$

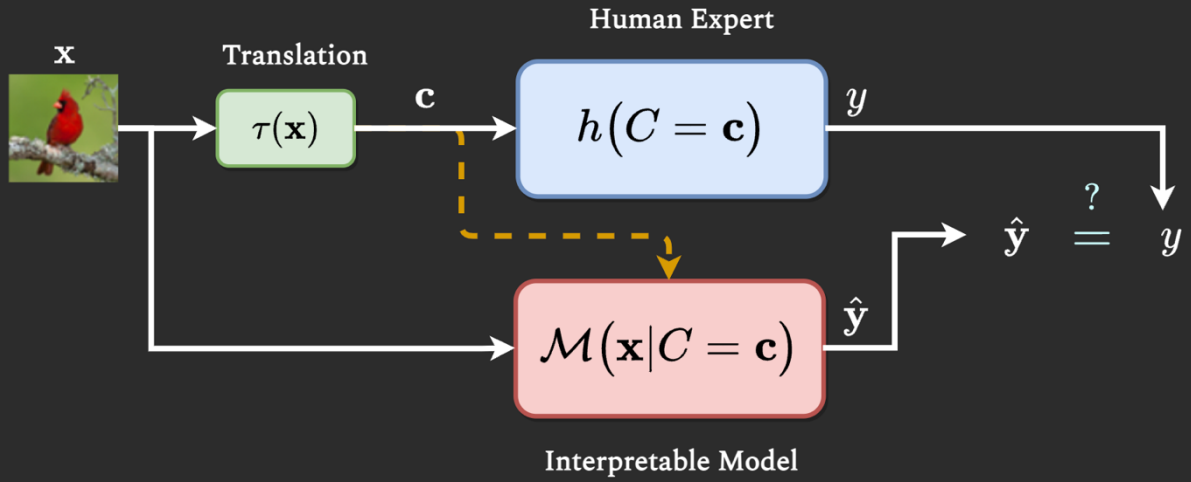
$P\left(C'_{\text{white}} = \texttt{True} \mid X = \right) = 0$ 

$P\left(C'_{\text{brown}} = \texttt{True} \mid X = \right) = 1$ 

The model should be able to accurately concepts P(C | X)

[1] Espinosa Zarlenga et al. "Avoiding Leakage Poisoning: Concept Interventions Under Distribution Shifts," (2025).

Desideratum #3: Intervenability

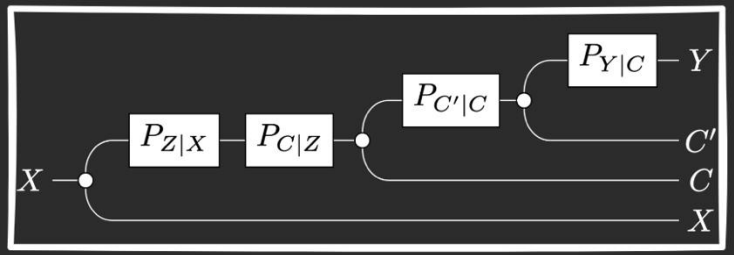


If we intervene on model M with concepts **c**, does it output the same task label an expert would given c? (i.e., does it satisfy *inference equivariance*?)

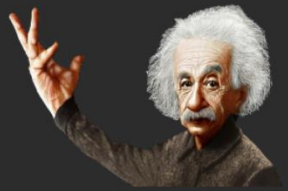
[1] Laguna et al. "Beyond concept bottleneck models: How to make black boxes intervenable?." NeurIPS (2024).
[2] Espinosa Zarlenga et al. "Avoiding Leakage Poisoning: Concept Interventions Under Distribution Shifts." ICML(2025).

Desideratum #3: Intervenability

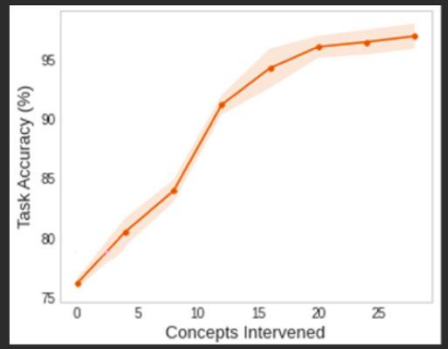
Interpretable Model



Expert Concept Labels



Improved Performance

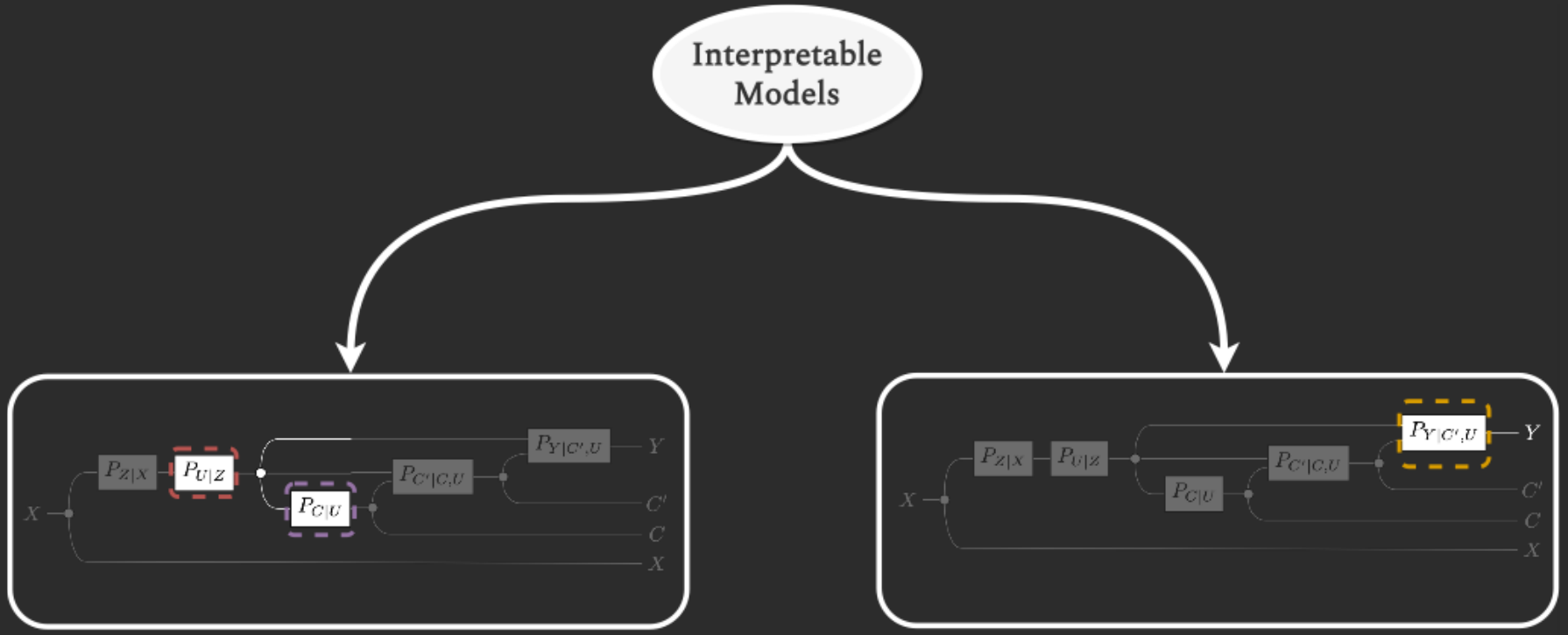


How to evaluate this? Task fidelity should be an increasing function of the number of interventions (i.e., interventions should have the intended effect)

[1] Laguna et al. "Beyond concept bottleneck models: How to make black boxes intervenable?." NeurIPS (2024).
[2] Espinosa Zarlenga et al. "Avoiding Leakage Poisoning: Concept Interventions Under Distribution Shifts." ICML(2025).

How can we instantiate our blueprint to achieve all three desiderata?

Design of interpretable models



Tutorial Outline

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences

Q&A + 30 min break

IV. Designing Concept Representations

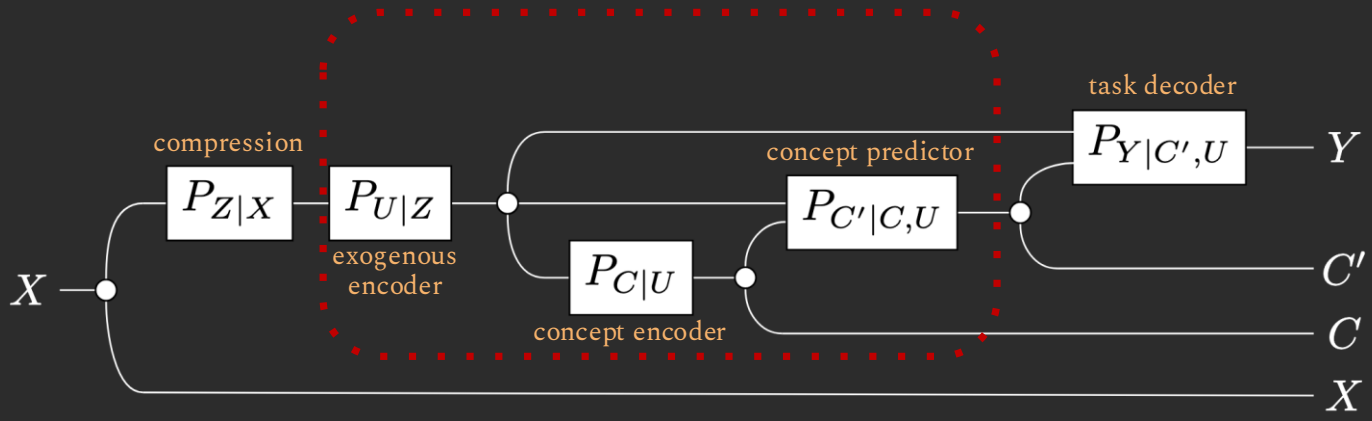
V. Designing Task Predictors

VI. Human-AI Interaction

VII. Conclusion

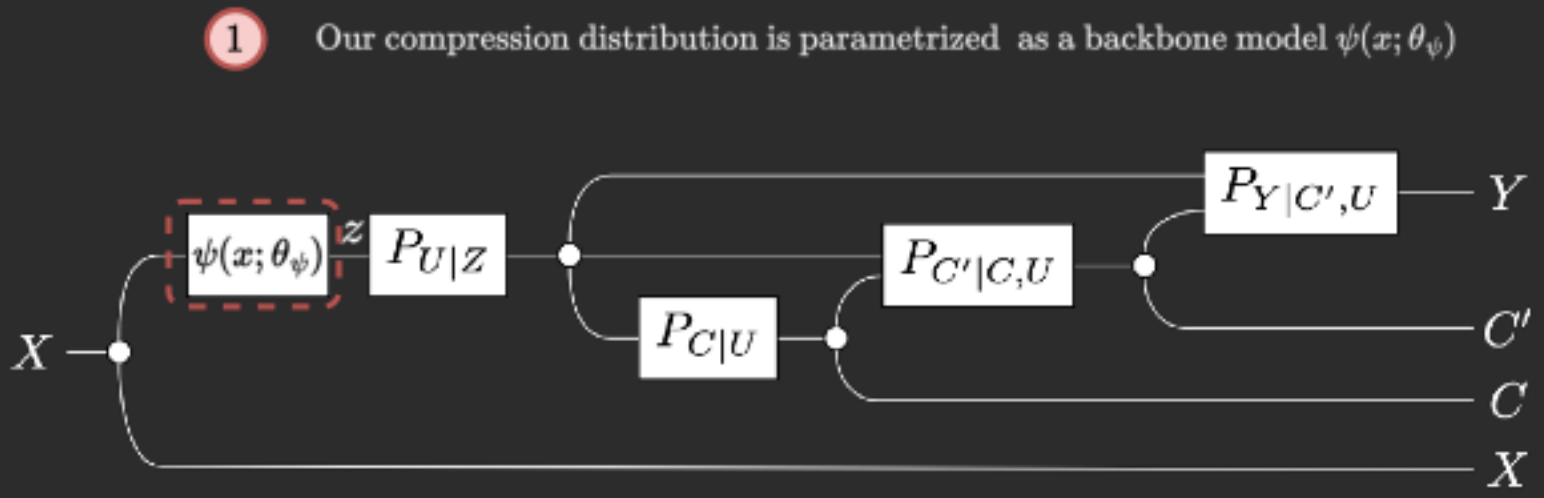
Final Q&A

Designing concept representations



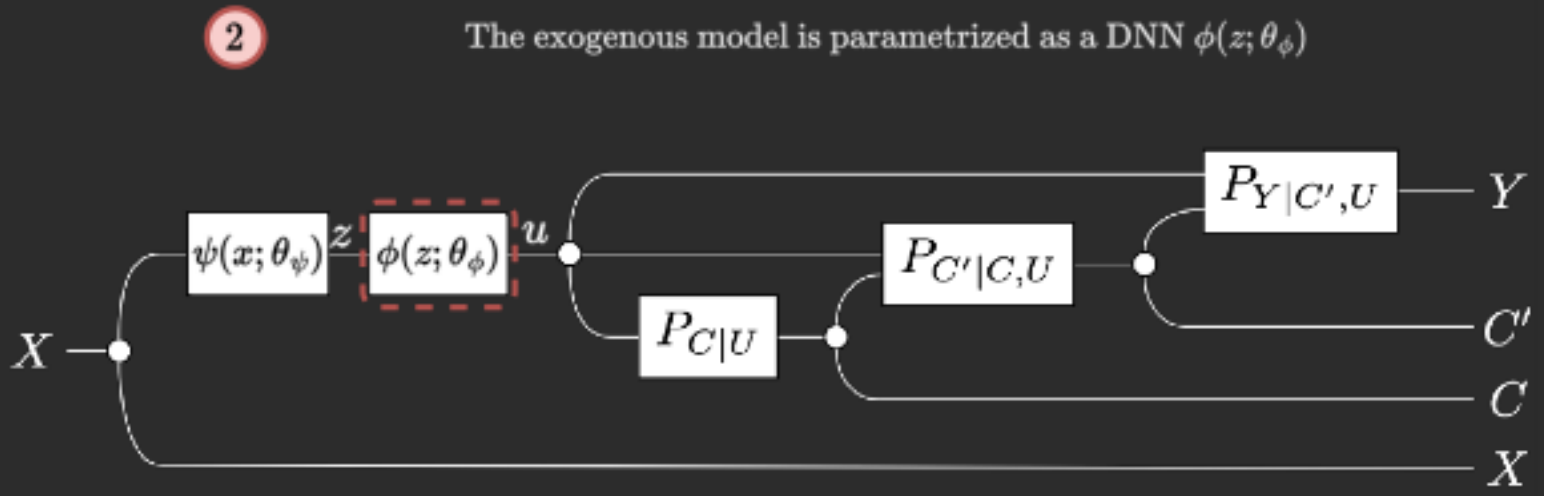
How should P(C|X) represent concepts?

Instantiating P(C|X): Assumptions



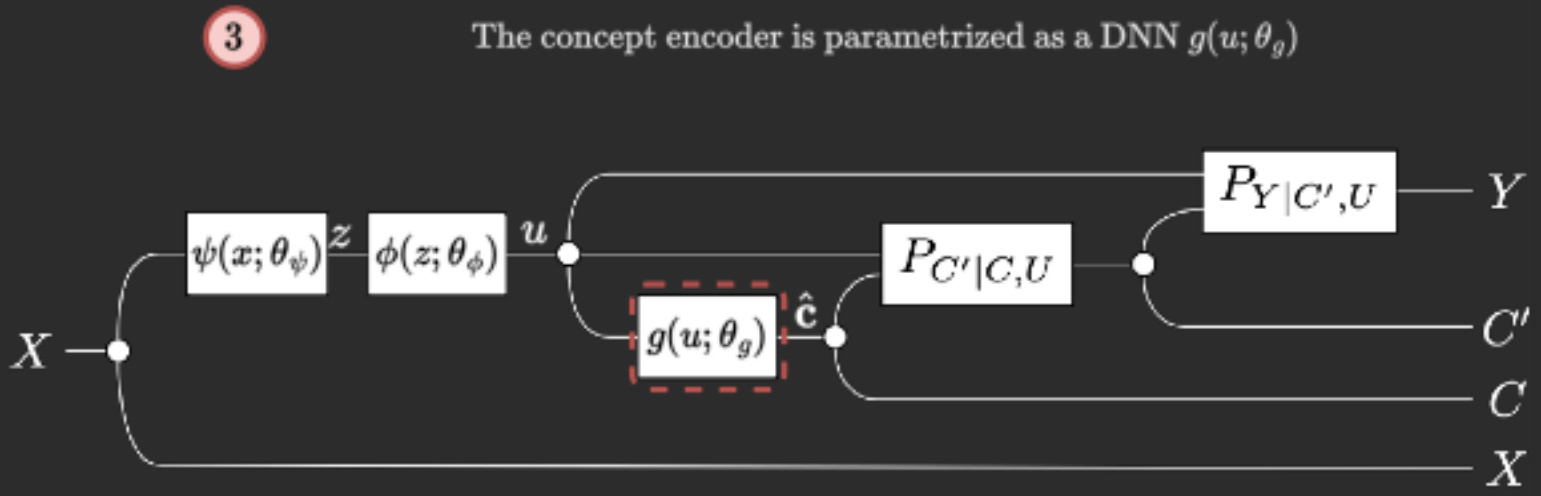
This could be, say, a pre-trained model

Instantiating P(C|X): Assumptions



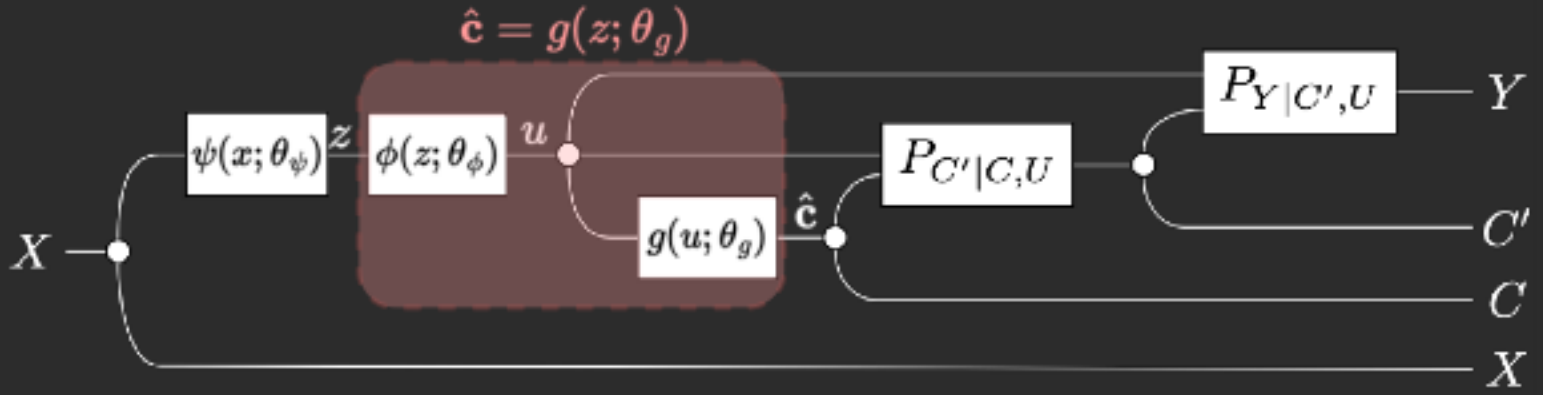
This will capture all information exogenous to our concepts

Instantiating P(C|X): Assumptions



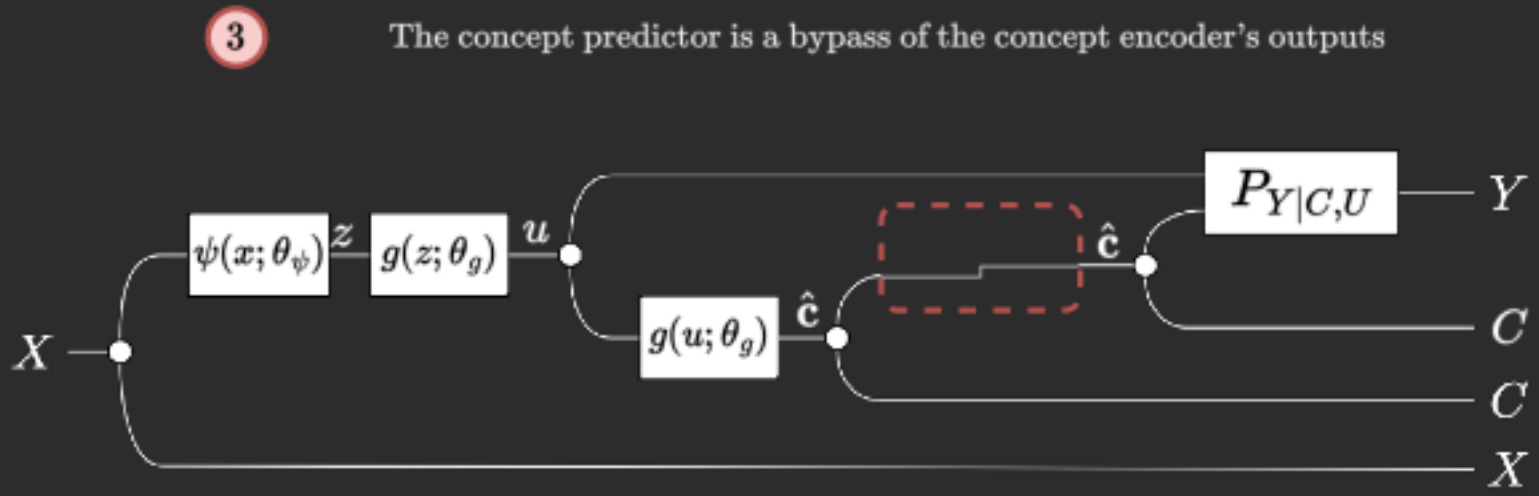
This encoder will produce our **concept representations**

Instantiating P(C|X): Assumptions



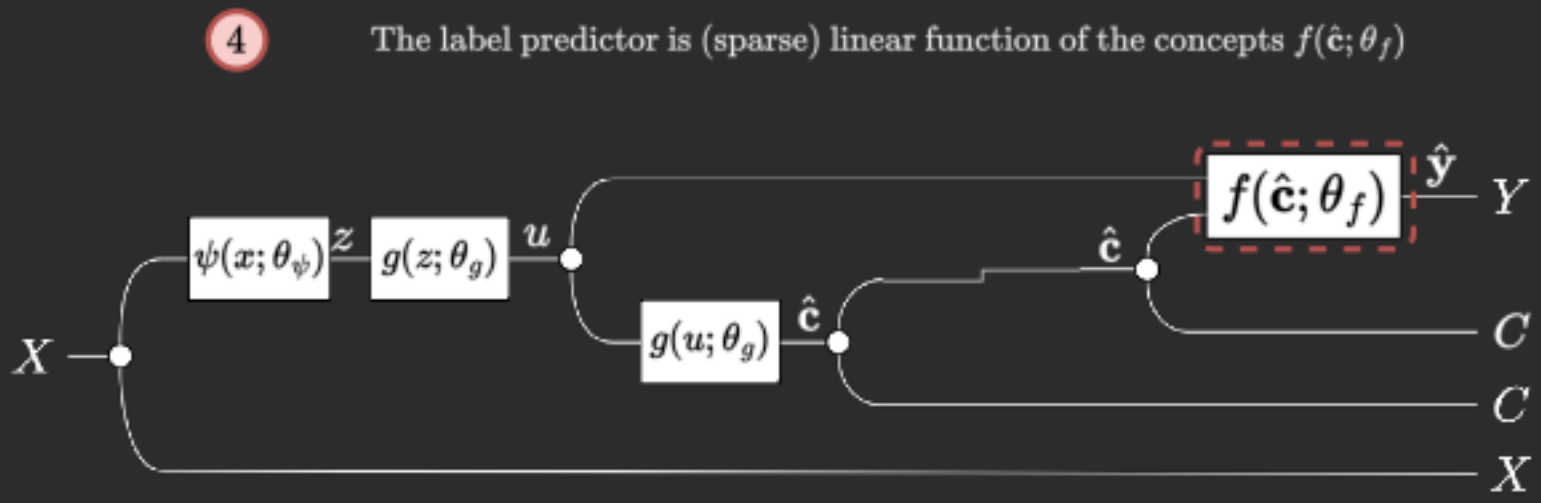
When we don't exploit exogenous information, we will fuse the exogenous model and the concept predictor

Instantiating P(C|X): Assumptions

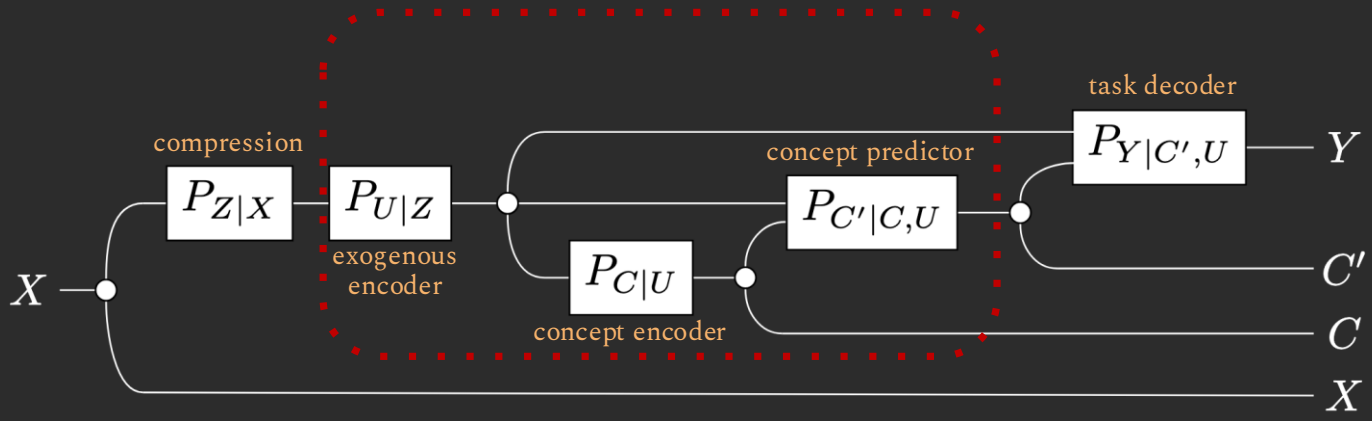


We will consider simple concept predictors where C′ = C

Instantiating P(C|X): Assumptions



For now, we will use a simple interpretable label predictor

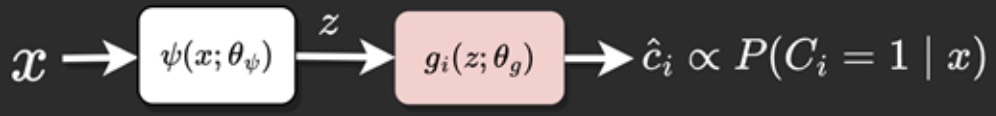


Given a representation z of a sample x, how can we design useful/powerful exogenous and concept encoders?

Bounded scalar concept representations

Simplest approach:

Assume no exogenous information (i.e., $\phi(z) = z$) and represent each concept as a bounded scalar

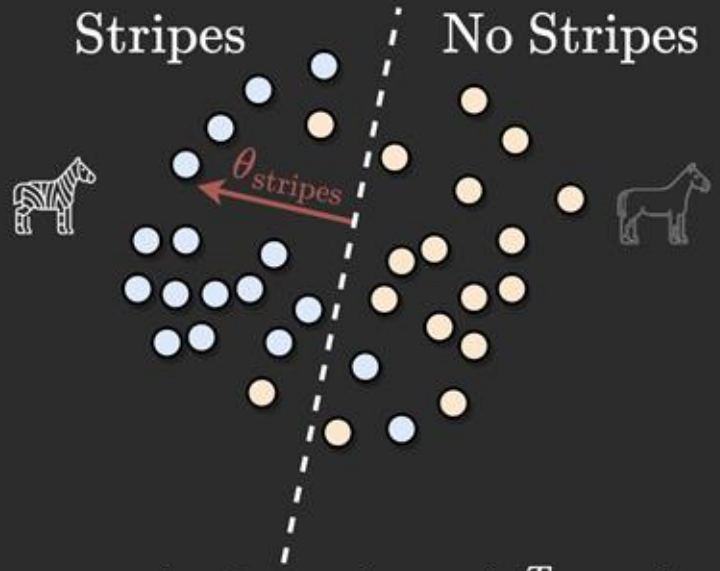


Where $\hat{c}_i \in [C_{\min}, C_{\max}]$ (e.g., $\hat{c}_i \in [0, 1]$)

```python
# Dimensionalities:
n = x_train.shape[1]  # Num of features
k = len(concepts)     # Num of concepts
z_dim = 32            # size of latent space


# Example Backbone: input features -> latent representtion
phi_backbone = ParametricCPD(
    "input",
    parametrization=torch.nn.Sequential(
        torch.nn.Linear(n, z_dim),
        torch.nn.LeakyReLU()
    )
)

# Example Concept Encoder: latent representation -> concept probabilities
c_encoder = ParametricCPD(
    concepts,
    parametrization=LinearZC(
        in_features=z_dim,
        out_features=k,
    )
)
```

Bounded scalar concept representations

A simple way of building these representations are linear probes:



$$\hat{\mathbf{c}}_{\text{stripes}} = g_{\text{stripes}}(\mathbf{z}; \theta_{\text{stripes}}) = \sigma(\theta_{\text{stripes}}^T \mathbf{z}) \in [0, 1]$$

Concept Bottleneck Models (CBMs)

A Concept Bottleneck Model (CBM) [1] exploits this idea by assuming:

1. Access to k binary concept labels C for each training sample.

$$\left(\mathbf{x}, \begin{bmatrix} c_{\text{sclerosis}} \\ c_{\text{spurs}} \\ \vdots \\ c_{\text{osteophytes}} \\ c_{\text{narrow\_joint}} \end{bmatrix}, y \right) \sim \mathcal{D}$$

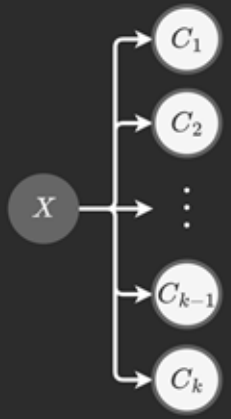
$y = 2$

[1] Koh et al. "Concept bottleneck models." International Conference on Machine Learning. PMLR, 2020.

Concept Bottleneck Models (CBMs)

A Concept Bottleneck Model (CBM) [1] exploits this idea by assuming:

1. Access to k binary concept labels C for each training sample.
2. Each concept in C can be independently predicted from X



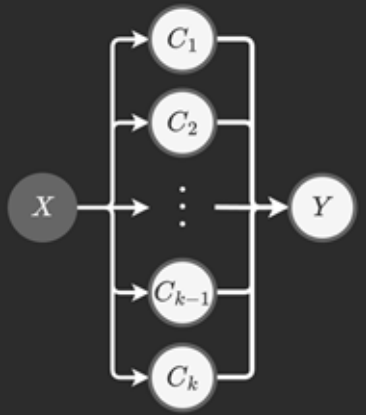
$$P(C \mid X) = \Pi_{i=1}^{k} P(C_i \mid X)$$

[1] Koh et al. "Concept bottleneck models." International Conference on Machine Learning. PMLR, 2020.

Concept Bottleneck Models (CBMs)

A Concept Bottleneck Model (CBM) [1] exploits this idea by assuming:

1. Access to k binary concept labels C for each training sample.
2. Each concept in C can be independently predicted from X
3. The downstream task can be perfectly predicted from C alone.

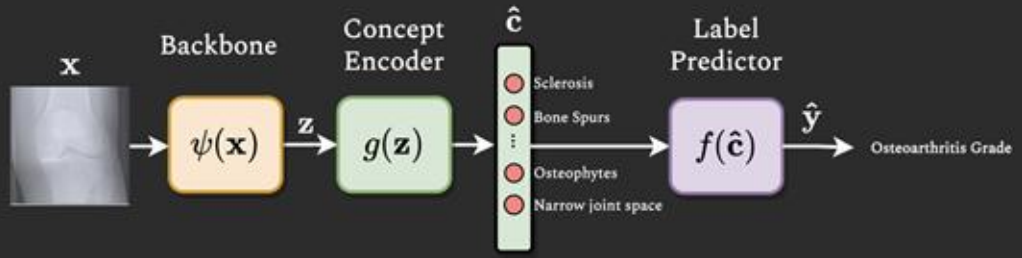


$$P(C, Y \mid X) = P(C \mid X) P(Y \mid C)$$

[1] Koh et al. "Concept bottleneck models." International Conference on Machine Learning. PMLR, 2020.

Implementing a CBM

In practice, a CBM looks like a composition of three back-to-back DNNs:

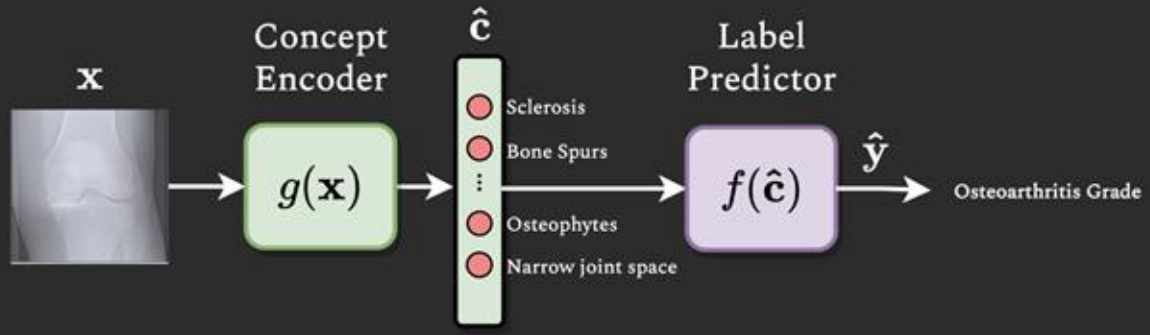


1. A backbone model mapping samples to a manifold
2. A concept encoder predicting concept probabilities
3. A label predictor predicting task labels

```python
from torch_concepts.nn import (
    LinearZC,
    LinearCC,
    BipartiteModel,
    LazyConstructor,
)

# Defining quantities:
n = x_train.shape[1]                # Num of features
k = len(concepts)                   # Num of concepts
z_dim = 32                          # size of latent space
task_names = ['positive', 'negative'] # Names of downstream taks labels

# Backbone: input features -> latent representtion
phi_backbone = torch.nn.Sequential(
    torch.nn.Linear(n, z_dim),
    torch.nn.LeakyReLU()
)

# Concept encoder: linear layer (latent -> concept preds)
concept_encoder = LazyConstructor(LinearZC)

# Label predictor: linear layer (concept preds -> task preds)
label_predictor = LazyConstructor(LinearCC)

# Put them together into a single CBM which connects all concepts to all
# task outputs (and it is hence a bipartite model)
cbm = BipartiteModel(
    encoder=concept_encoder,
    predictor=label_predictor,
    task_names=task_names,
    input_size=z_dim,
    annotations=concepts,
)
```

Implementing a CBM

Usually, we train the backbone and the concept encoder as a fused model:



Generally, unless illustrative, we assume the backbone is fused into the concept encoder

How to train your CBM

Given a training set $\mathcal{D} = \{(\mathbf{x}^{(j)}, \mathbf{c}^{(j)}, y^{(i)})\}_{j=1}^{N}$, we can train a CBM in one of three ways:

1. **Independently**: train $g(\mathbf{x})$ and $f(\mathbf{c})$ separately.

$$\begin{cases} \underset{\theta_g}{\operatorname{argmin}}\, \mathbb{E}_{(\mathbf{x},\mathbf{c},y)\sim\mathcal{D}}\Big[\underbrace{\operatorname{BCE}\big(g(\mathbf{x};\theta_g), \mathbf{c}\big)}_{\text{Concept Alignment Loss}} \Big] \\ \underset{\theta_f}{\operatorname{argmin}}\, \mathbb{E}_{(\mathbf{x},\mathbf{c},y)\sim\mathcal{D}}\Big[\underbrace{\operatorname{CE}\big(f(\mathbf{c};\theta_f), y\big)}_{\text{Task Loss}} \Big] \end{cases}$$

```python
# Setup training
task_loss_fn = torch.nn.CrossEntropyLoss()
concept_loss_fn = torch.nn.BCEWithLogitsLoss()

# STEP #1: Train concept encoder
concept_encoder.train()
optimizer = torch.optim.AdamW(
    (phi_backbone.parameters() + concept_encoder.parameters()),
    lr=0.01,
)
for epoch in range(n_epochs):
    optimizer.zero_grad()

    # Compute embedding
    latent = phi_backbone(x_train)
    # And the concepts
    c_pred = concept_encoder(latent)

    # Compute the concept loss
    concept_loss = concept_loss_fn(c_pred, c_train)

    # Backward pass
    concept_loss.backward()
    optimizer.step()

#  STEP #2: Train label predictor with GROUND-TRUTH concepts
optimizer = torch.optim.AdamW(
    label_predictor.parameters(),
    lr=0.01,
)
label_predictor.train()
for epoch in range(n_epochs):
    optimizer.zero_grad()
    y_pred = label_predictor(c_ctrain)

    # Compute the task loss
    task_loss = task_loss_fn(y_pred, y_train)

    # Backward pass
    task_loss.backward()
    optimizer.step()
```

How to train your CBM

Given a training set $\mathcal{D} = \{(\mathbf{x}^{(j)}, \mathbf{c}^{(j)}, y^{(i)})\}_{j=1}^{N}$, we can train a CBM in one of three ways:

2. **Sequentially**: train $g(\mathbf{x})$ and then train $f(\mathbf{c})$ from the outputs of $g(\mathbf{x})$:

$$\underset{\theta_g}{\operatorname{argmin}}\, \mathbb{E}_{(\mathbf{x},\mathbf{c},y)\sim\mathcal{D}}\Big[\mathrm{BCE}\big(g(\mathbf{x};\theta_g), \mathbf{c}\big)\Big]$$

$$\downarrow$$

Freeze $g(\mathbf{x})$

$$\downarrow$$

$$\underset{\theta_f}{\operatorname{argmin}}\, \mathbb{E}_{(\mathbf{x},\mathbf{c},y)\sim\mathcal{D}}\Big[\mathrm{CE}\big(f(g(\mathbf{x};\theta_g);\theta_f), y\big)\Big]$$

```python
# STEP #1: Train concept encoder (as in independent training)
concept_encoder.train()
optimizer = torch.optim.AdamW(
    (phi_backbone.parameters() + concept_encoder.parameters()),
    lr=0.01,
)
for epoch in range(n_epochs):
    optimizer.zero_grad()

    # Compute embedding
    latent = phi_backbone(x_train)
    # And the concepts
    c_pred = concept_encoder(latent)

    # Compute the concept loss
    concept_loss = concept_loss_fn(c_pred, c_train)

    # Backward pass
    concept_loss.backward()
    optimizer.step()

# STEP #2: Train label predictor using the outputs of the concept encoder!
label_predictor.train()
optimizer = torch.optim.AdamW(label_predictor.parameters(), lr=0.01)
concept_encoder.eval()
with torch.no_grad():
    c_pred_train = concept_encoder(phi_backbone(x_train))

label_predictor.train()
for epoch in range(n_epochs):
    optimizer.zero_grad()
    y_pred = label_predictor(c_pred_train)

    # Compute the task loss
    task_loss = task_loss_fn(y_pred, c_train)

    # Backward pass
    task_loss.backward()
    optimizer.step()
```

How to train your CBM

Given a training set $\mathcal{D} = \{(\mathbf{x}^{(j)}, \mathbf{c}^{(j)}, y^{(i)})\}_{j=1}^{N}$, we can train a CBM in one of three ways:

3. **Jointly**: train $g(\mathbf{x})$ and $f(\mathbf{c})$ at the same time:

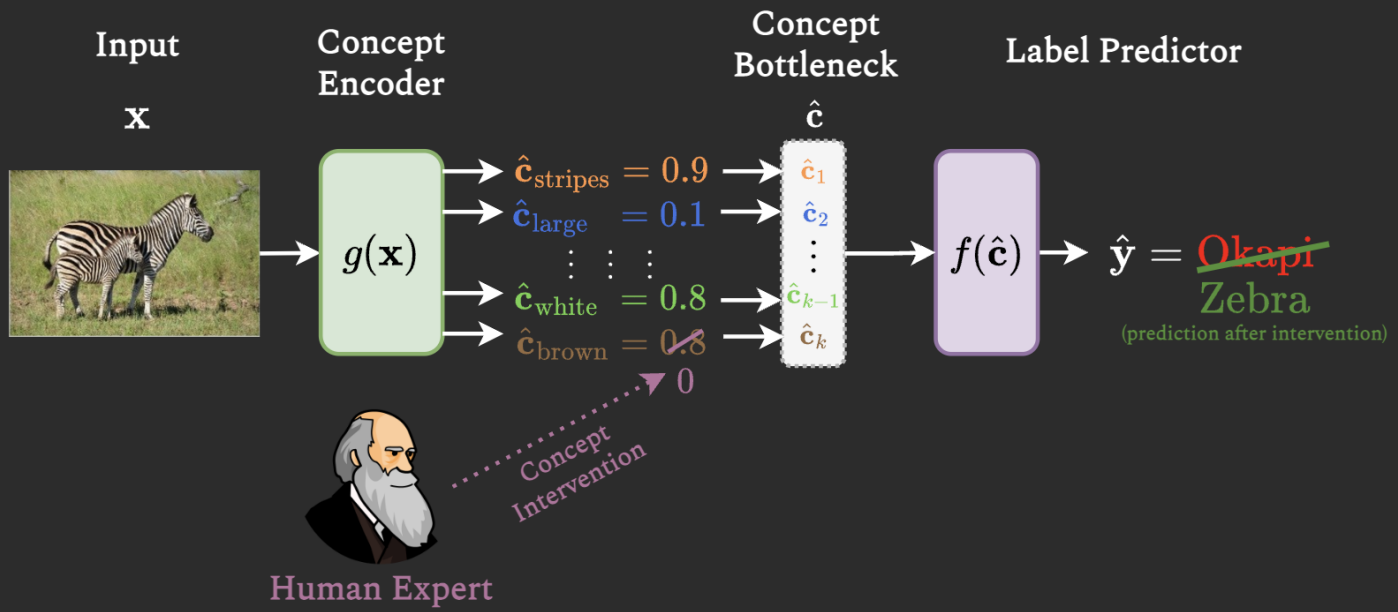
$$\underset{\theta_f, \theta_g}{\operatorname{argmin}} \mathbb{E}_{(\mathbf{x},\mathbf{c},y)\sim\mathcal{D}} \Bigg[\underbrace{\mathrm{CE}\Big(f(g(\mathbf{x};\theta_g);\theta_f), y\Big)}_{\text{Task Loss}} + \lambda \underbrace{\mathrm{BCE}\Big(g(\mathbf{x};\theta_g), \mathbf{c}\Big)}_{\text{Concept Alignment Loss}} \Bigg]$$

Controls how much we value concept alignment vs task alignment

```python
# Initialize the inference engine with the BipartiteModel's ProbabilisticModel
engine = DeterministicInference(cbm.probabilistic_model)

# Define the query (what we want to infer). In this case it is all
# concepts and all downstream task labels
model_outputs = concepts + task_names

# Setup training
optimizer = torch.optim.AdamW(cbm.parameters(), lr=0.01)
loss_fn = torch.nn.BCEWithLogitsLoss()
cbm.train()

# Training loop
for epoch in range(n_epochs):
    optimizer.zero_grad()

    # Compute embedding
    latent = phi_backbone(x_train)

    # Inference: query the ProbabilisticModel with embedding as evidence
    cy_pred = engine.query(model_outputs, evidence={'latent': latent})

    # Split predictions: first columns are concepts, remaining are task
    c_pred = cy_pred[:, :len(concepts)]
    y_pred = cy_pred[:, len(concepts):]

    # Compute losses
    concept_loss = loss_fn(c_pred, c_train)
    task_loss = loss_fn(y_pred, y_train)
    # Joint Training: optimize a linear combination of both losses
    loss = concept_loss + concept_reg * task_loss

    # Backward pass
    loss.backward()
    optimizer.step()
```

How to intervene on bounded scalar representations?

Notice bounded scalar representations easily support concept interventions:



Challenges of traditional CBMs

Although simple, learning bounded scalar concept representations like those seen in “vanilla” CBMs comes with some significant challenges:

Computational constraints

Challenges of traditional CBMs

Although simple, learning bounded scalar concept representations like those seen in "vanilla" CBMs comes with some significant challenges:

Computational constraints

$I(X;Y) \gg I(C;Y)$

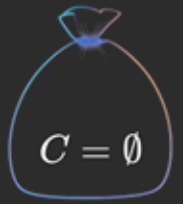
Concept incompleteness

Challenges of traditional CBMs

Although simple, learning bounded scalar concept representations like those seen in "vanilla" CBMs comes with some significant challenges:

Computational constraints

Concept incompleteness



Concept unavailability

Challenges of traditional CBMs

We will now introduce different concept representations we can use to address each of these limitations

Computational constraints

↓

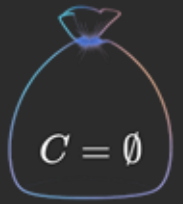
Unbounded scalar representations

$I(X;Y) \gg I(C;Y)$

Concept incompleteness

↓

Dynamic embeddings representations



Concept unavailability

↓

Multimodal similarity representations

A taster of concept representations

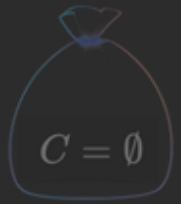
Computational constraints

Unbounded scalar representations

Concept incompleteness

Dynamic embeddings representations

Concept unavailability



Multimodal similarity representations

Computational constraints

Representation Problem #1: Computational Constraints

We want to avoid training a large concept encoder from scratch

Computational constraints

Representation Problem #1: Computational Constraints

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

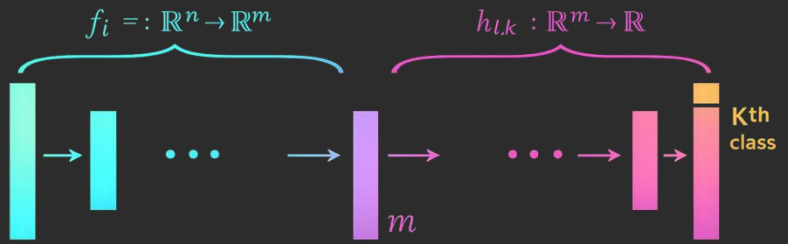
1. Do not have the necessary compute to train a concept encoder from scratch

Computational constraints

Representation Problem #1: Computational Constraints

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)



Computational constraints

Representation Problem #1: Computational Constraints

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

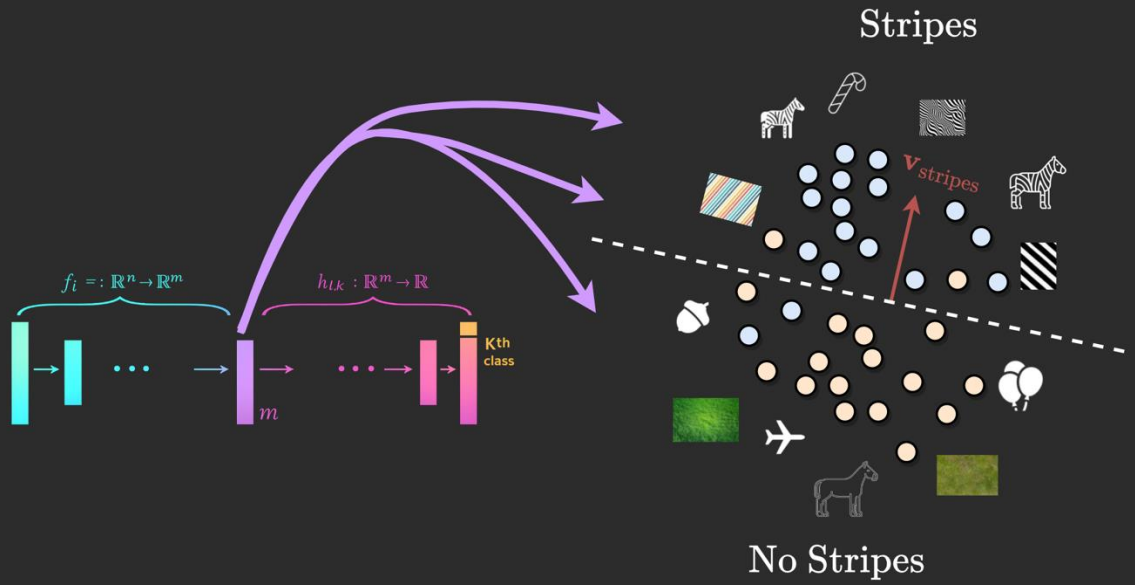
1. Do not have the necessary compute to train a concept encoder from scratch
2. Have at our disposal an already powerful but opaque DNN for P(Y|X)
3. Have sparse concept labels (e.g., sets of representative concept samples)

$X_{stripes}$

X_{grass}

Solution idea: 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:

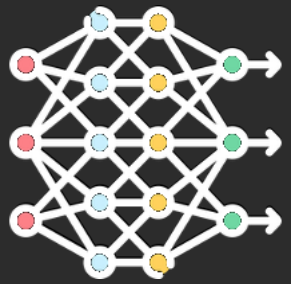


[1] Kim et al. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)," ICML (2018).

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

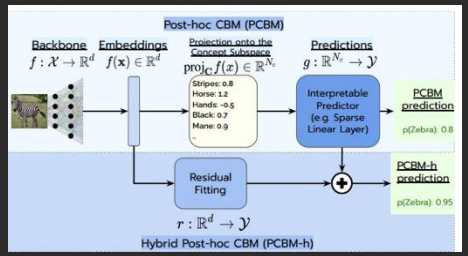


+

Concept Activation Vector Bank

=

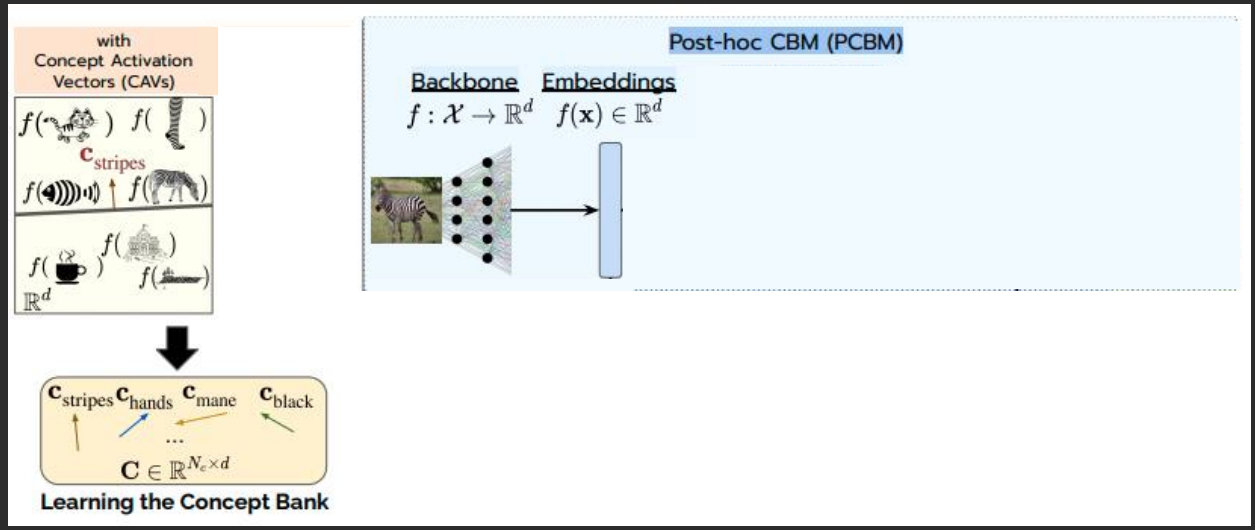
Post-hoc CBMs!



[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:

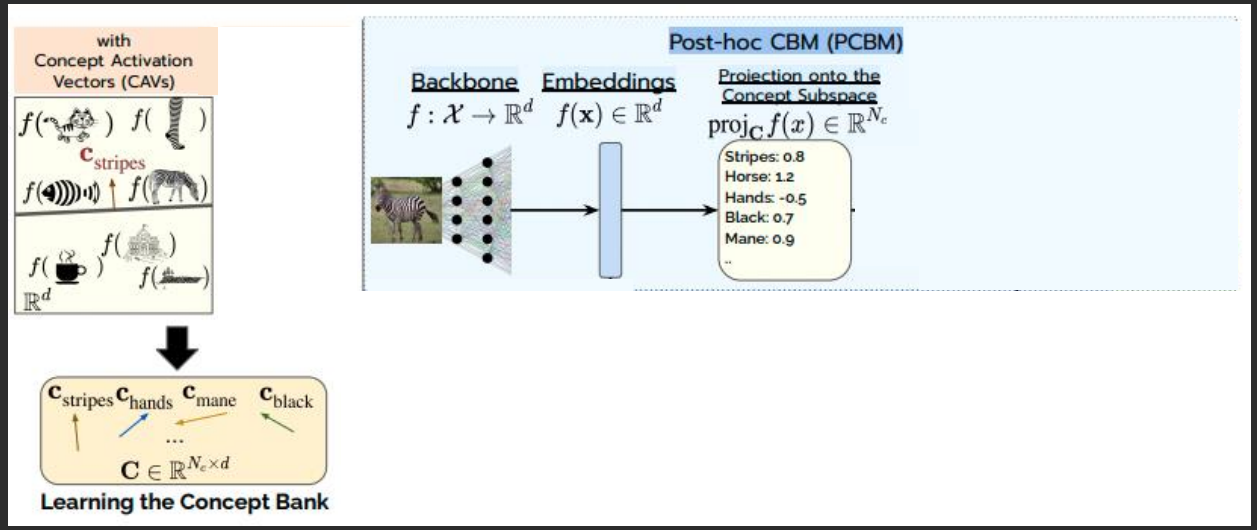


Step 1: learn a Concept Activation Vector (CAV) that separates the DNN's latent space between samples with a concept vs samples without a concept

[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:

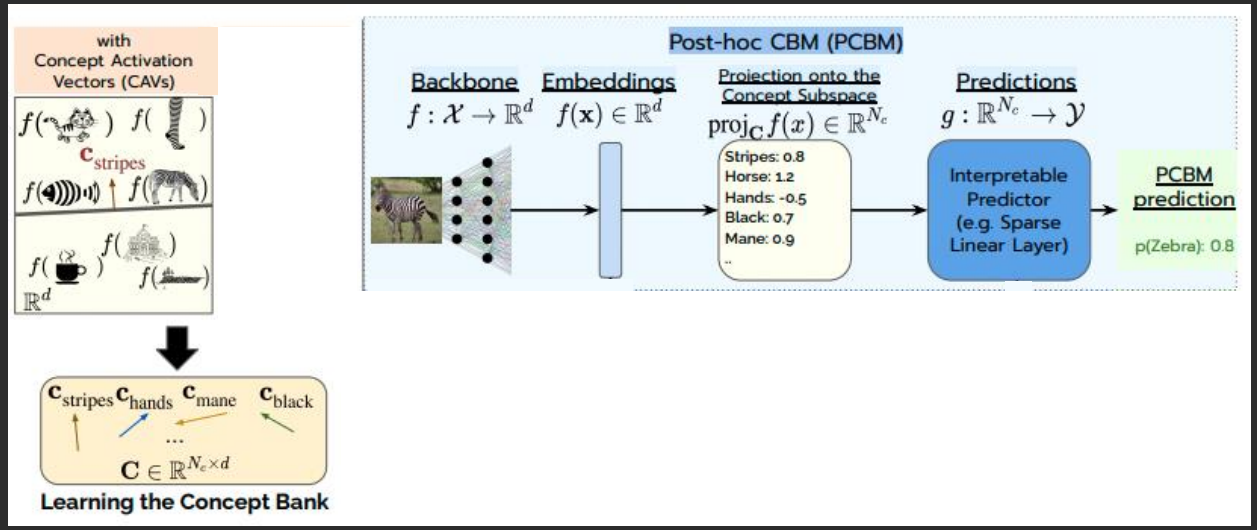


Step 2: project all training samples to the concept activation space using the CAVs

[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:



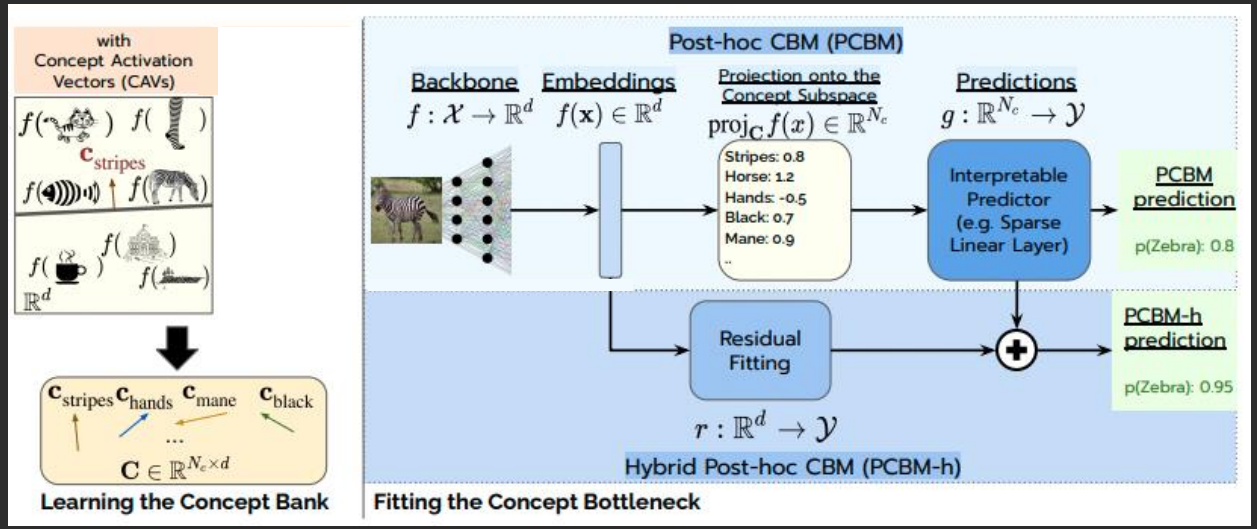
Make the final prediction with an interpretable predictor

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).

Intervening on unbounded concept scalars

We can intervene on unbounded concept scalar representation based on their empirical distribution:

$$\hat{\mathbf{c}}_i := \begin{cases} \hat{\mathbf{c}}_i^{[95\%]} & \text{if } c_i = 1 \\ \hat{\mathbf{c}}_i^{[5\%]} & \text{if } c_i = 0 \end{cases}$$

[1] Koh et al. "Concept bottleneck models." International Conference on Machine Learning. PMLR, 2020.

Intervening on unbounded concept scalars

We can intervene on unbounded concept scalar representation based on their empirical distribution:

$$\hat{\mathbf{c}}_i := \begin{cases} \hat{\mathbf{c}}_i^{[95\%]} & \text{if } c_i = 1 \\ \hat{\mathbf{c}}_i^{[5\%]} & \text{if } c_i = 0 \end{cases}$$

Limitation: interventions are less effective than in bounded scalar representations

[1] Koh et al. "Concept bottleneck models." International Conference on Machine Learning. PMLR, 2020.

A taster of concept representations

Computational constraints

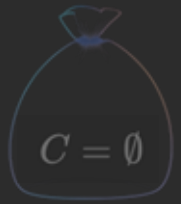
Unbounded scalar representations

Concept incompleteness

$I(X;Y) \gg I(C;Y)$

Dynamic embeddings representations

Concept unavailability

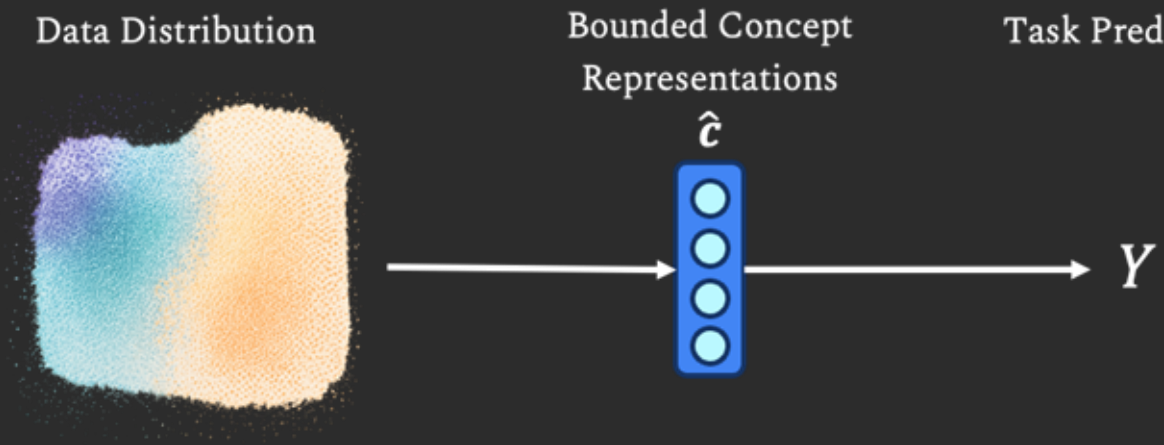


Multimodal similarity representations

Concept incompleteness

Representation Problem #2: Concept Incompleteness

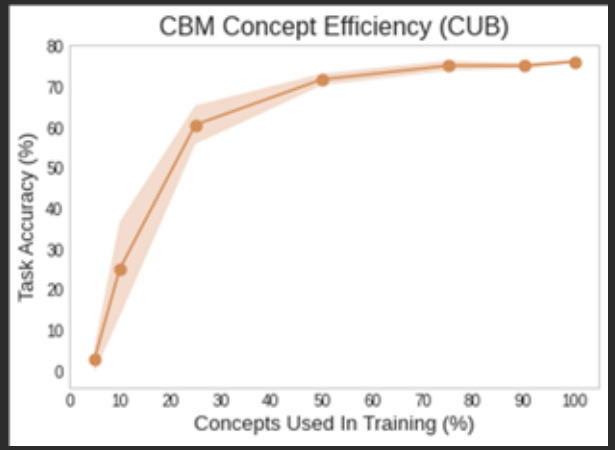
Although simple, scalar concept representations lead to an information bottleneck in the model when the concept set is incomplete

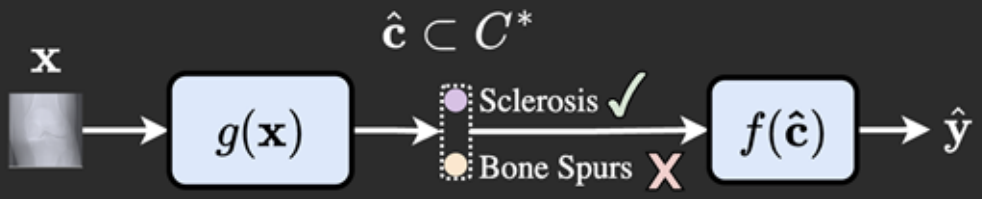


Concept incompleteness

Representation Problem #2: Concept Incompleteness

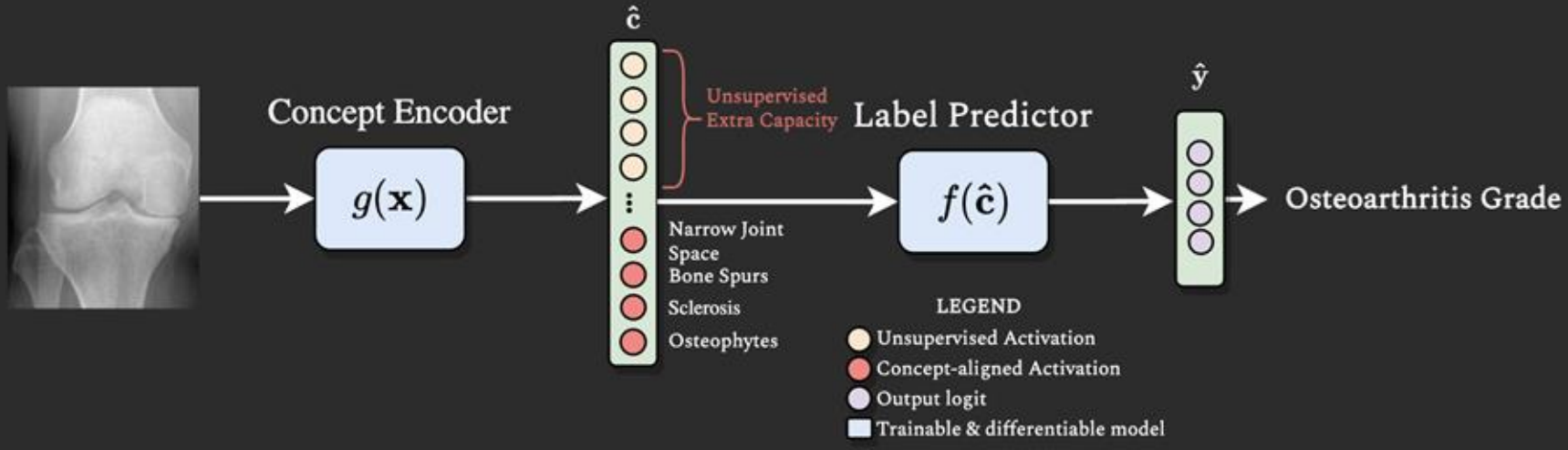
Although simple, scalar concept representations lead to an information bottleneck in the model when the concept set is incomplete





Solution attempt: unsupervised capacity

Idea: what if we extend the bottleneck with unsupervised capacity?



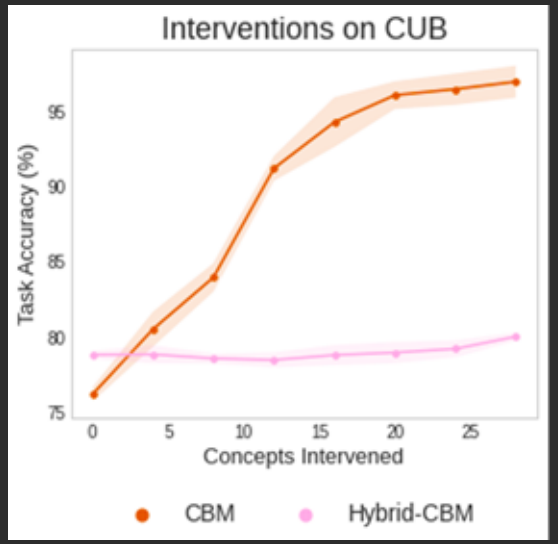
A **"Hybrid" CBM** adds extra unsupervised capacity to the CBM's bottleneck

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

Solution attempt: unsupervised capacity

Idea: what if we extend the bottleneck with 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: supervised high-dimensional vectors

Idea: can we use supervised extra capacity instead?

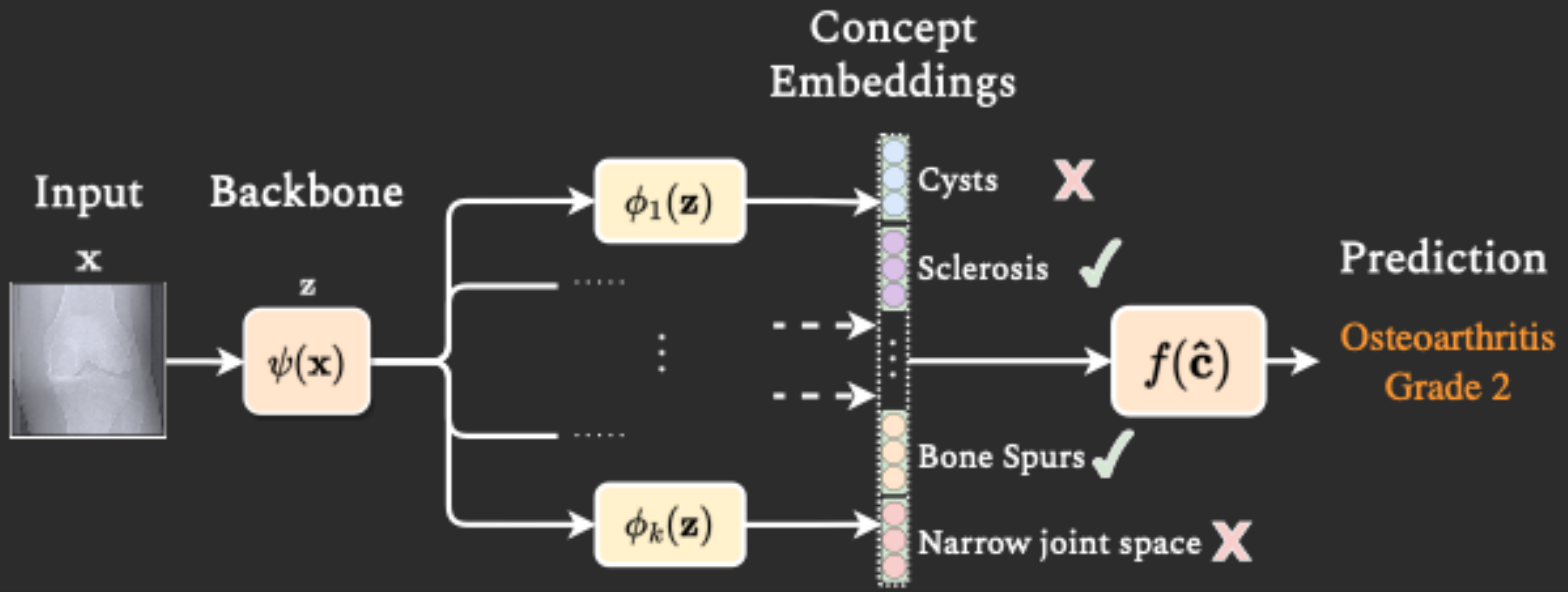
Solution idea: supervised high-dimensional vectors

Idea: can we use supervised extra capacity instead?

Solution: use a non-trivial exogenous model!

Overcoming incompleteness: concept embeddings

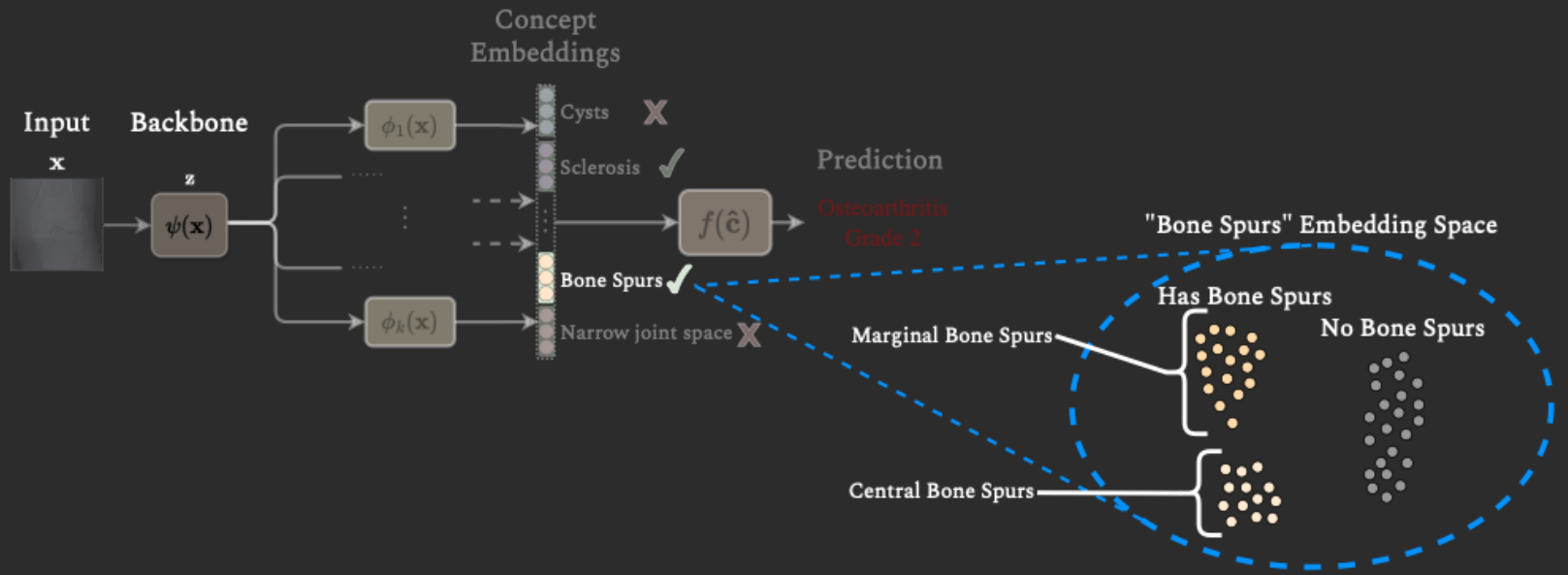
For each training concept, let's learn an exogenous model that allows us to construct supervised embedding representations of the concept



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

Overcoming incompleteness: concept embeddings

For each training concept, let's learn an exogenous model that allows us to construct supervised embedding representations of the concept



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

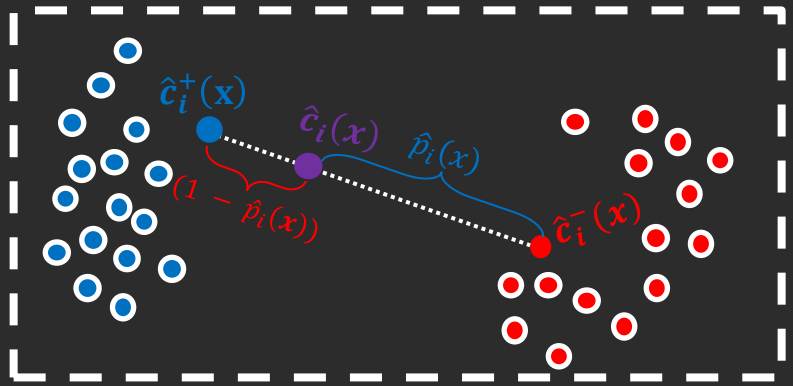
Learning dynamic concept embeddings

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:

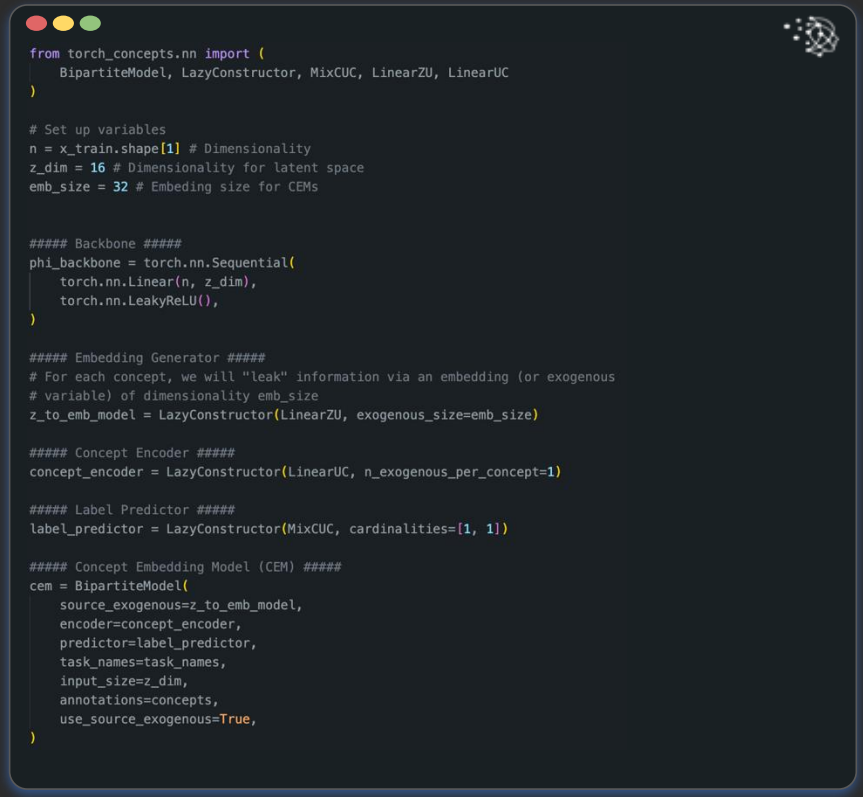
$$\hat{c}_i := \begin{cases} \hat{c}_i^+(z) \text{ if } c_i = 1 \\ \hat{c}_i^-, (z) \text{ otherwise} \end{cases}$$

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):

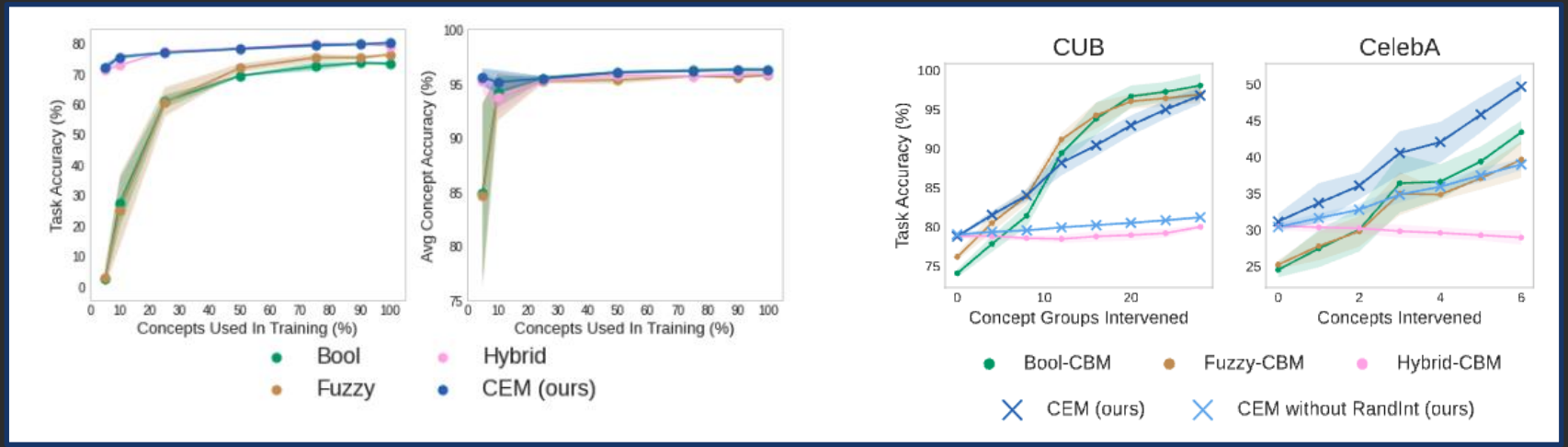


```python
from torch_concepts.nn import (
    BipartiteModel, LazyConstructor, MixCUC, LinearZU, LinearUC
)

# Set up variables
n = x_train.shape[1] # Dimensionality
z_dim = 16 # Dimensionality for latent space
emb_size = 32 # Embeding size for CEMs


##### Backbone #####
phi_backbone = torch.nn.Sequential(
    torch.nn.Linear(n, z_dim),
    torch.nn.LeakyReLU(),
)

##### Embedding Generator #####
# For each concept, we will "leak" information via an embedding (or exogenous
# variable) of dimensionality emb_size
z_to_emb_model = LazyConstructor(LinearZU, exogenous_size=emb_size)

##### Concept Encoder #####
concept_encoder = LazyConstructor(LinearUC, n_exogenous_per_concept=1)

##### Label Predictor #####
label_predictor = LazyConstructor(MixCUC, cardinalities=[1, 1])

##### Concept Embedding Model (CEM) #####
cem = BipartiteModel(
    source_exogenous=z_to_emb_model,
    encoder=concept_encoder,
    predictor=label_predictor,
    task_names=task_names,
    input_size=z_dim,
    annotations=concepts,
    use_source_exogenous=True,
)
```

[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



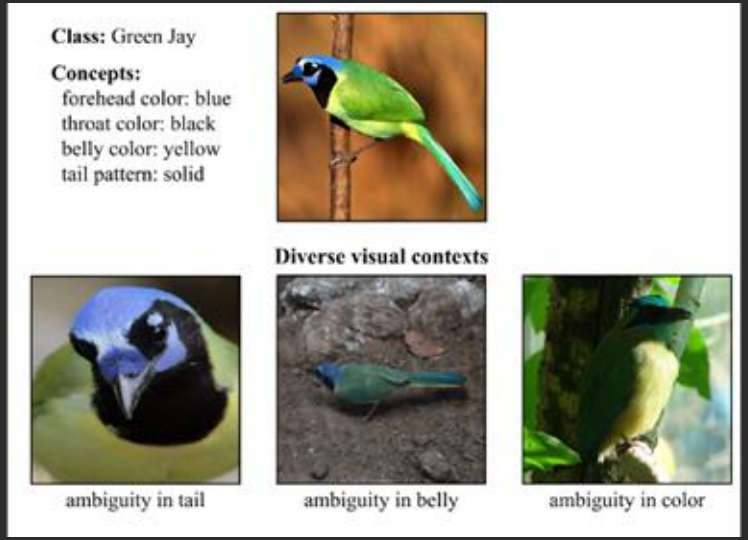
*This is particularly true when, during training, you randomly intervene** on a concept with probability p_{int}.

** 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).

Using embeddings for uncertainty estimation

Traditional CBMs are not great at providing calibrated concept uncertainties

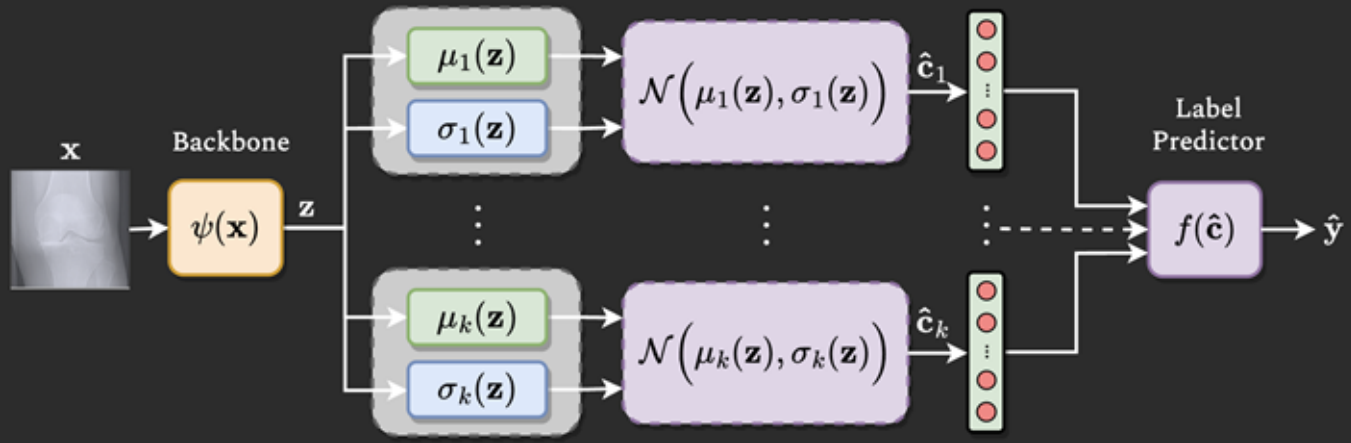


This is because of cross-entropy-based training doesn't lead to calibrated probabilities

[1] Kim et al. "Probabilistic Concept Bottleneck Models." ICML 2023).

Using embeddings for uncertainty estimation

Probabilistic concept embeddings provide better uncertainty estimates:



We encourage distinct absent/present representations via contrastive scoring function:

$$P(C_i = 1 \mid \hat{\mathbf{c}}_i) = s(\hat{\mathbf{c}}_i) := \sigma\left(a\left(\|\hat{\mathbf{c}}_i - \mathbf{c}_i^-\|_2 - \|\hat{\mathbf{c}}_i - \mathbf{c}_i^+\|_2\right)\right)$$

[1] Kim et al. "Probabilistic Concept Bottleneck Models." ICML 2023).

Using embeddings for uncertainty estimation

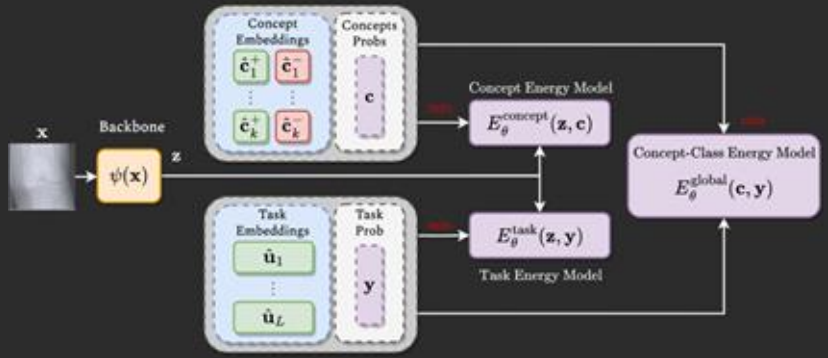
Here, we can compute uncertainty estimates by looking at the **variances**:

$$\text{Uncertainty}(c_i) \propto GM\big(\sigma_i(\mathbf{z})\big) = \sqrt[m]{\sigma_i(\mathbf{z})_1 \sigma_i(\mathbf{z})_2 \cdots \sigma_i(\mathbf{z})_m}$$

[1] Kim et al. "Probabilistic Concept Bottleneck Models." ICML 2023).

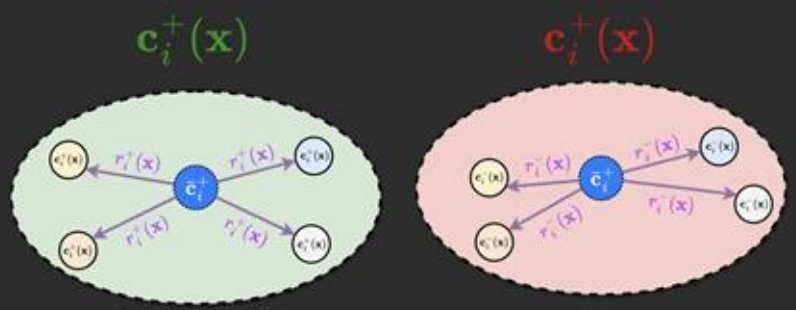
Further generalisations of concept embeddings

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).

A taster of concept representations

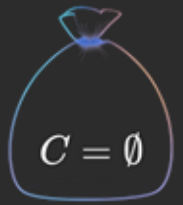
Computational constraints

Unbounded scalar representations

Concept incompleteness

Dynamic embeddings representations

Concept unavailability

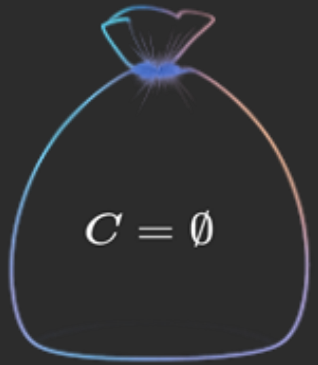


Multimodal similarity representations

Concept unavailability

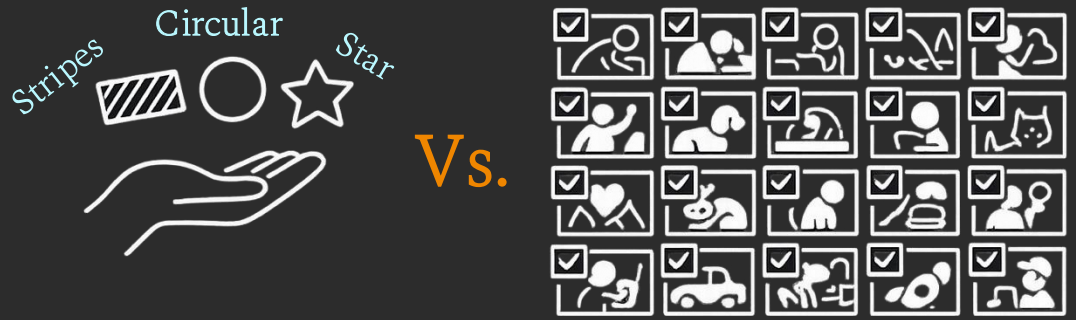
Representation Problem #3: Concept Unavailability

Assuming the access to concept annotations (even sparse) is highly unrealistic!



Solution idea: textual concept descriptions

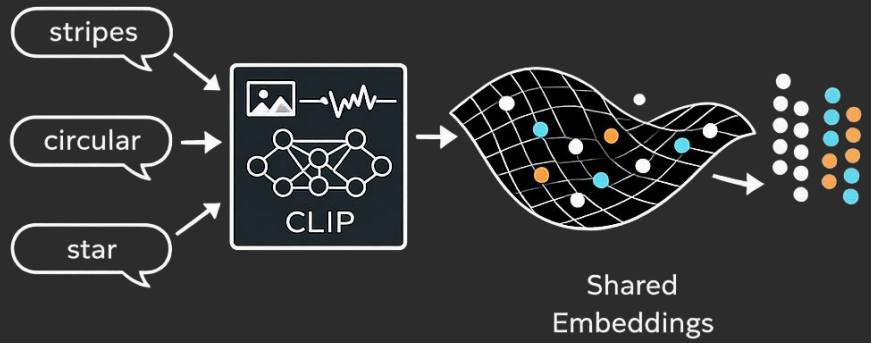
It is much *easier* to **describe the concepts** that are useful for a task than to annotate them on a large dataset



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

Exploiting multimodal representations

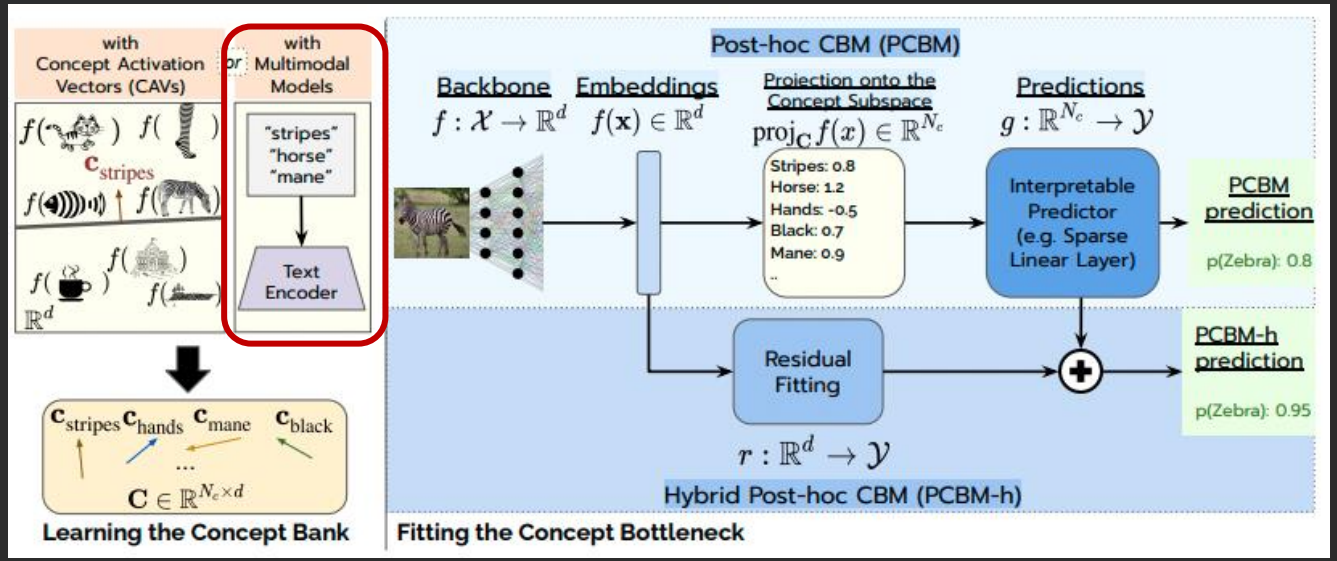
Unbounded similarity scores can be generated from textual descriptions of the concepts if we have some multimodal embedding model (e.g., CLIP)



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

Exploiting multimodal representations

Note we can use textual descriptions of the concepts to train Post-hoc CBMs:



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

Automating concept discovery

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

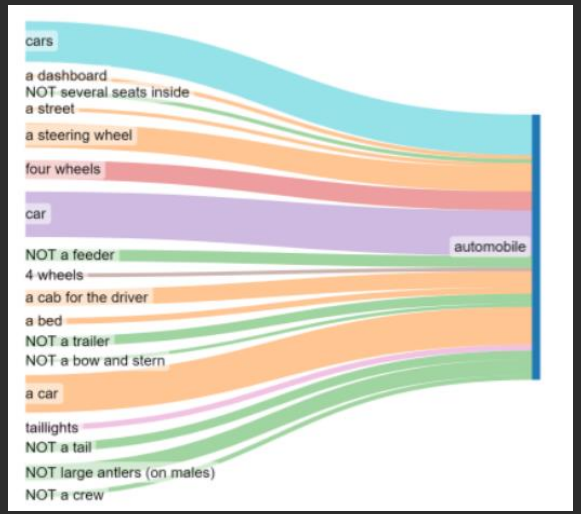
"List the most important features for recognizing something as a {class}:"

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

Automating concept discovery

However, for this to work, we need two extra bits:

1. Concept filtering: LLMs generate non-visual, redundant, and ambiguous concepts



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

Automating concept discovery

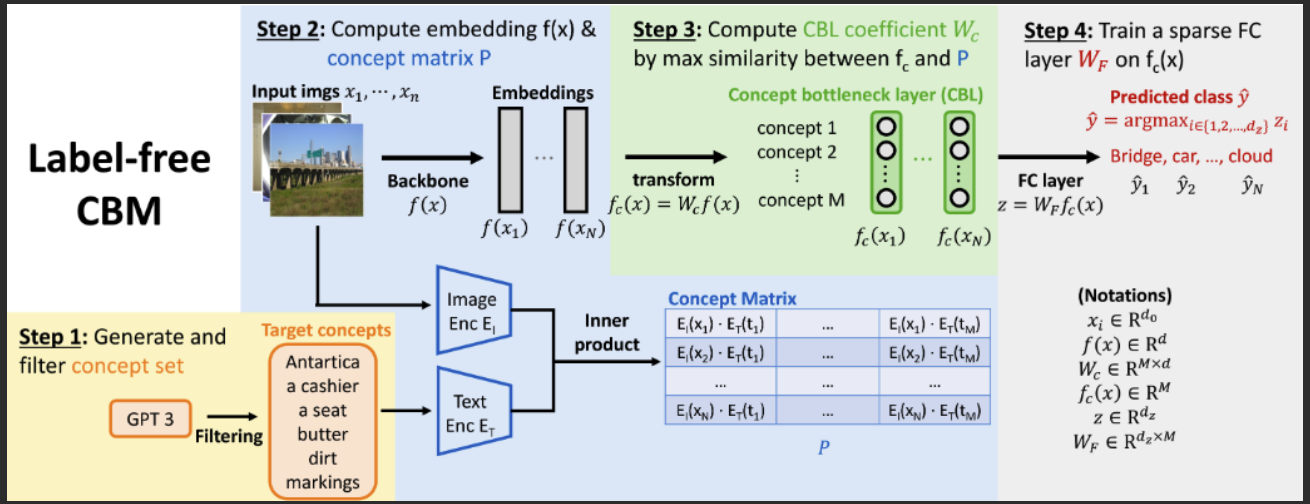
However, for this to work, we need two extra bits:

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

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

Multimodal concept encoders

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).

Filtering the concept bank

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

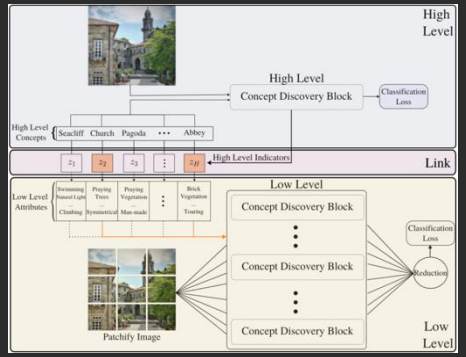
How good is the concept at discriminating only a handful of task labels?

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).

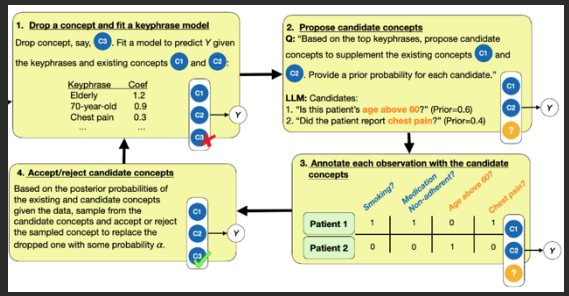
Recent directions in multimodal concept encoders

Coarse-to-Fine CBMs



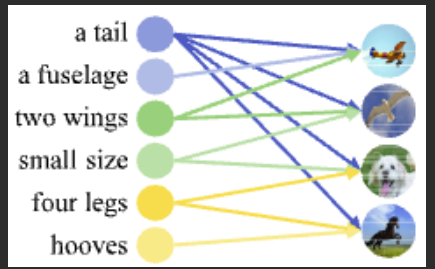
Discover a two-level concept hierarchy that can be then encoded in the model (Panousis et al., 2024) [2]

LLMs as Priors



Use an LLM as a prior over potential concept phrases to explore an effectively infinite concept space (Feng et al., 2025) [1]

Partially shared CBM



Ensure semantic visual grounding and remove concept redundancies (Zhao et al., 2025) [3]

[1] Feng et al. "Bayesian concept bottleneck models with LLM priors." NeurIPS (2025).
[2] Panousis et al. "Coarse-to-fine concept bottleneck models." NeurIPS (2024).
[3] Zhao et al. "Partially Shared Concept Bottleneck Models." AAAI (2026).

Tutorial Outline

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences

Q&A + 30 min break

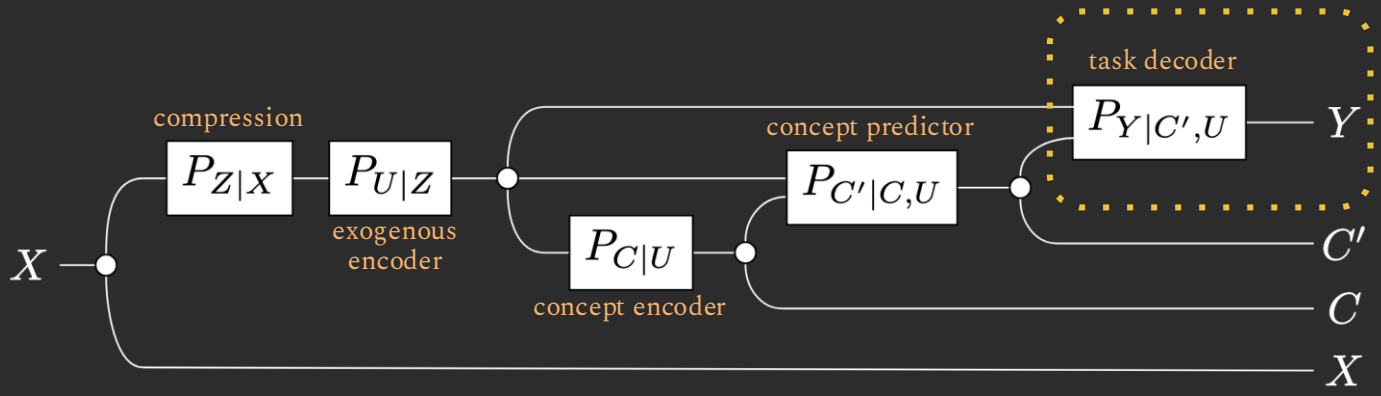
IV. Designing Concept Representations

V. Designing Task Predictors

VI. Human-AI Interaction

VII. Conclusion

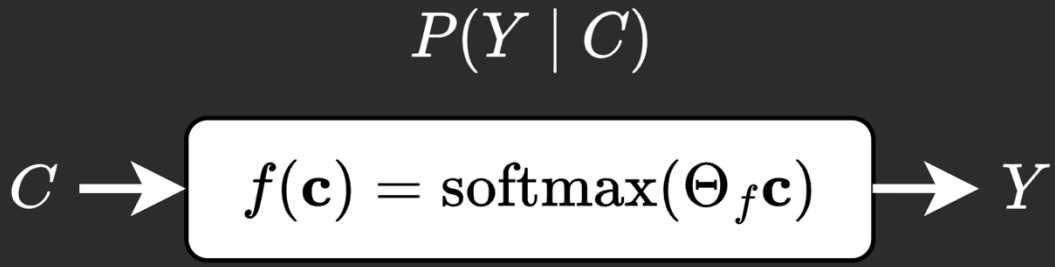
Final Q&A



How should we predict task labels P(Y|C, U)?

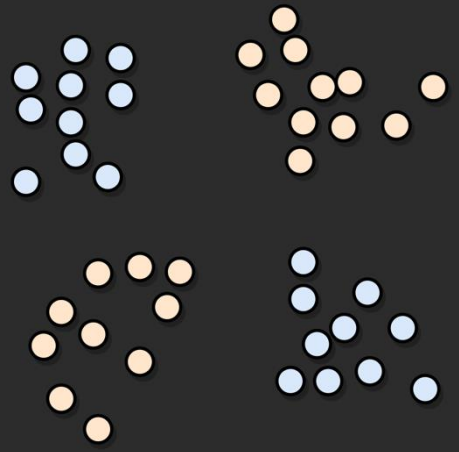
Instantiating the task predictor P(Y|C)

So far, we have assumed that the task predictor $P(Y \mid C, U)$ is a linear layer.



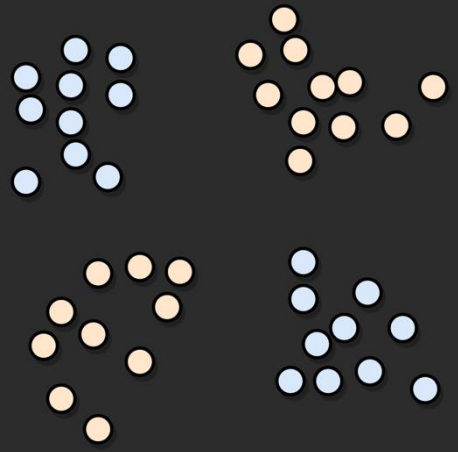
Limitations of linear predictors

Linear models, however, are not very expressive and may struggle to learn complex non-linear interactions between concepts and labels.



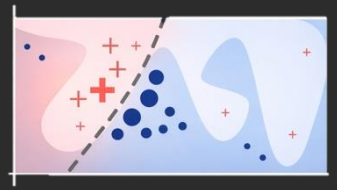
Limitations of linear predictors

Linear models, however, are not very expressive and may struggle to learn complex non-linear interactions between concepts and labels.



We will now consider different ways of parameterising $P(Y \mid C, U)$

A taster of label predictors

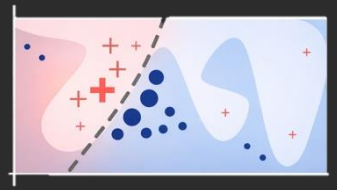


Locally-interpretable Predictors

Neuro-Symbolic Predictors

Causal Predictors

A taster of label predictors



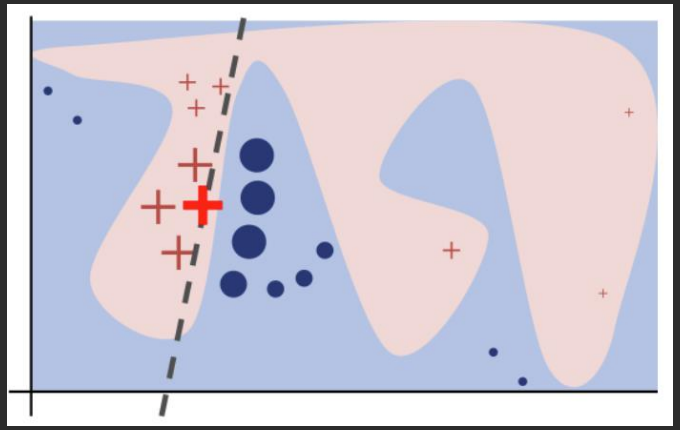
Locally-interpretable Predictors

Neuro-Symbolic Predictors

Causal Predictors

Locally-interpretable task predictors

As a first step, we can construct a nonlinear task predictor $f(\hat{c}) = P(Y \mid \hat{c})$ that behaves "linear-like" in the local neighborhood of $\hat{c}$



[1] Alvarez Melis et al. "Towards robust interpretability with self-explaining neural networks," NeurIPS (2018).
[2] Image adapted from Ribeiro et al. ""Why should Itrust you?" Explaining the predictions of any classifier," KDD (2016).

Locally-interpretable task predictors

To achieve local-linearity, we want:

1. [Expressiveness] The relevance weights $\boldsymbol{\theta}$ used to predict a downstream task from the concept representations $\hat{c}$ must adapt depending on the exogenous variables $u = \phi(\psi(x))$:

$$\text{Traditional Linear Model Output:} \quad f(\hat{c}; \theta) = \theta^T \hat{c}$$
$$\text{“Linear-ish DNN” Model output:} \quad f(\hat{c}; \theta(u)) = \theta(u)^T \hat{c}$$

where $\theta: \mathcal{U} \rightarrow \Theta$ is parameterised as a learnable DNN

[1] Alvarez Melis et al. "Towards robust interpretability with self-explaining neural networks," NeurIPS (2018).

Locally-interpretable task predictors

To achieve local-linearity, we want:

2. [Local Linearity] The model should behave, at least in the neighborhood of a sample's concept space, as a linear classifier. In other words, we want:

$$\nabla_{g(u)} f\left(\hat{\boldsymbol{c}}\,; \theta(\boldsymbol{u})\right) \approx \theta(\boldsymbol{u})$$

That is, we want samples with similar concept representations to have similar importance weights (even if they have very different features x)

[1] Alvarez Melis et al. "Towards robust interpretability with self-explaining neural networks," NeurIPS (2018).

Locally-interpretable task predictors

To achieve local-linearity, we want:

2. [Local Linearity] The model should behave, at least in the neighborhood of a sample's concept space, as a linear classifier. In other words, we want:

$$\nabla_{g(u)} f\left(\hat{\boldsymbol{c}}\,; \theta(\boldsymbol{u})\right) \approx \theta(\boldsymbol{u})$$

This can be enforced by minimizing a loss that implicitly leads to this condition holding:

$$\mathcal{L}_{reg}(u) = \left\| \nabla_u f(\hat{c}) \; - \theta(u)^T \, J_{\boldsymbol{u}}^{g(u)}(u) \right\|$$

[1] Alvarez Melis et al. "Towards robust interpretability with self-explaining neural networks," NeurIPS (2018).

Problems with locally-linear predictors

Unfortunately, locally-linear predictors cannot:

1. Correctly model features with strong (even local) higher order relationships

Problems with locally-linear predictors

Unfortunately, locally-linear predictors cannot:

1. Correctly model features with strong (even local) higher order relationships
2. Provide insights about global behavior of the predictor

Problems with locally-linear predictors

Unfortunately, locally-linear predictors cannot:

1. Correctly model features with strong (even local) higher order relationships
2. Provide insights about global behavior of the predictor
3. Be trained very easily (unstable/difficult to train due to second-order gradients)

A taster of label predictors

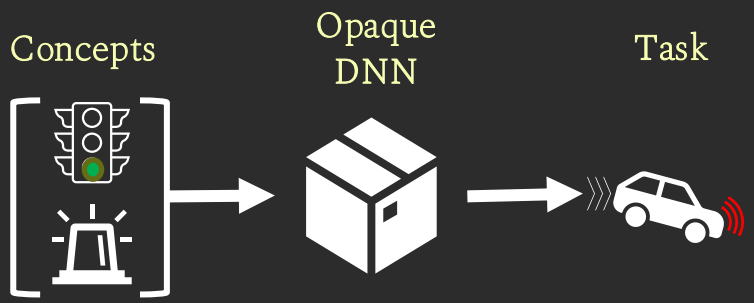
Locally-interpretable Predictors

Neuro-Symbolic Predictors

Causal Predictors

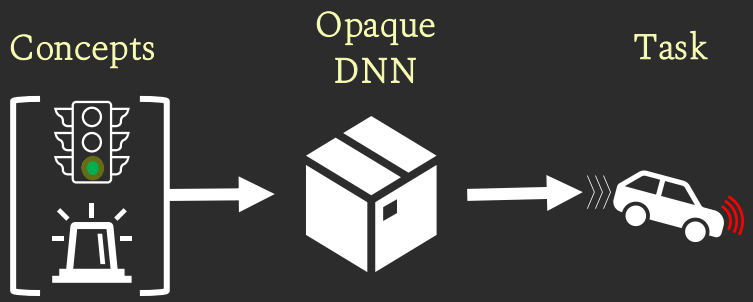
Going beyond linear predictors

Imagine we have all the time in the world to intervene on the inputs of a DNN predicting task labels from the set of predicted concepts:



Going beyond linear predictors

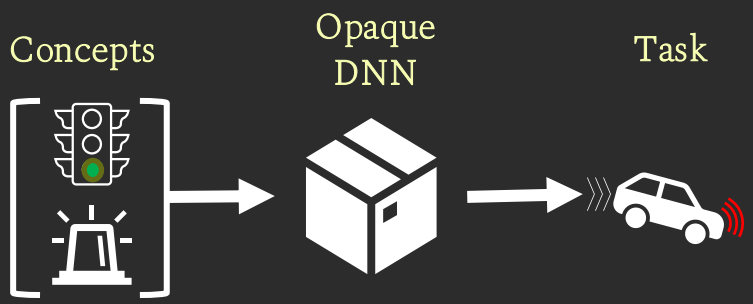
Imagine we have all the time in the world to intervene on the inputs of a DNN predicting task labels from the set of predicted concepts:

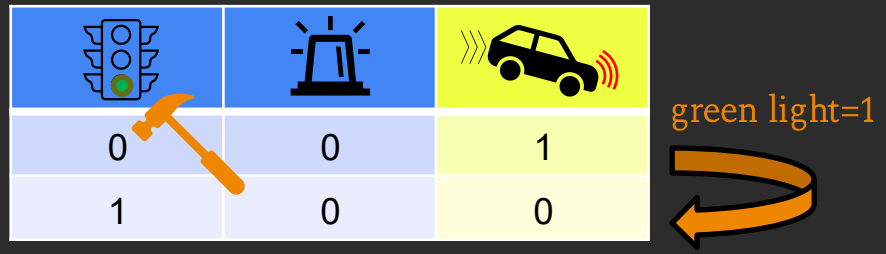


0	0	1

Going beyond linear predictors

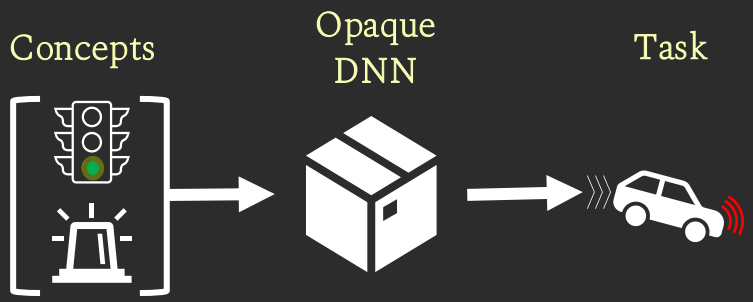
Imagine we have all the time in the world to intervene on the inputs of a DNN predicting task labels from the set of predicted concepts:

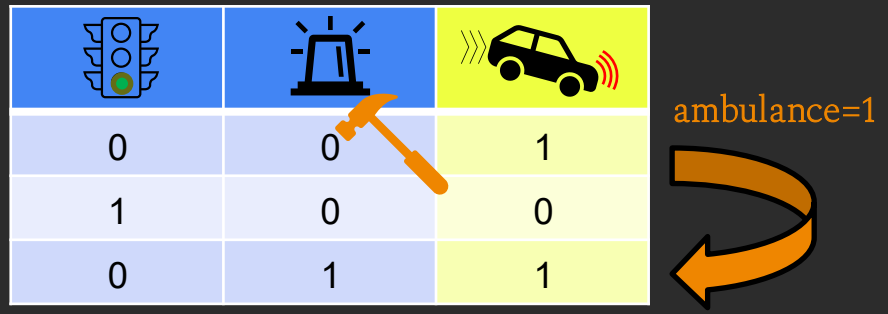




Going beyond linear predictors

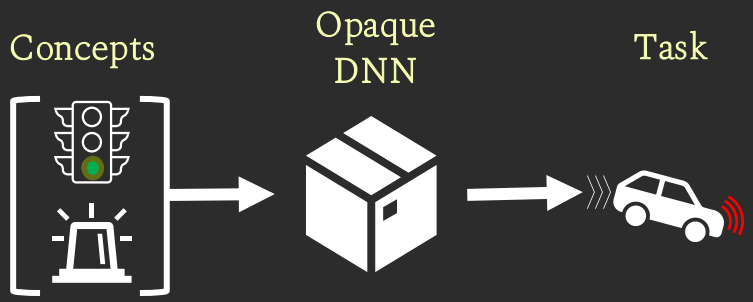
Imagine we have all the time in the world to intervene on the inputs of a DNN predicting task labels from the set of predicted concepts:





Going beyond linear predictors

Imagine we have all the time in the world to intervene on the inputs of a DNN predicting task labels from the set of predicted concepts:

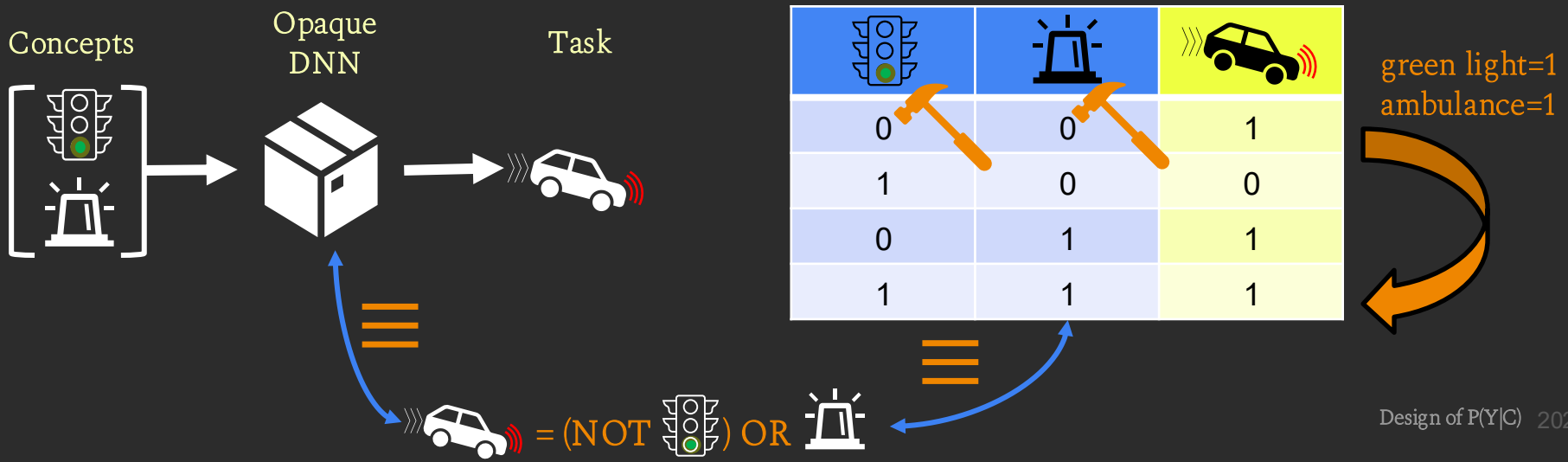


0	0	1
1	0	0
0	1	1
1	1	1

green light=1
ambulance=1

Going beyond linear predictors

Imagine we have all the time in the world to intervene on the inputs of a DNN predicting task labels from the set of predicted concepts:



Going beyond linear predictors

Idea: Can we use this approach to construct a truth table/logic rule mapping concepts to downstream tasks?

This would allow us to capture non-linear relationships between concepts using logic rules we can understand!

Going beyond linear predictors

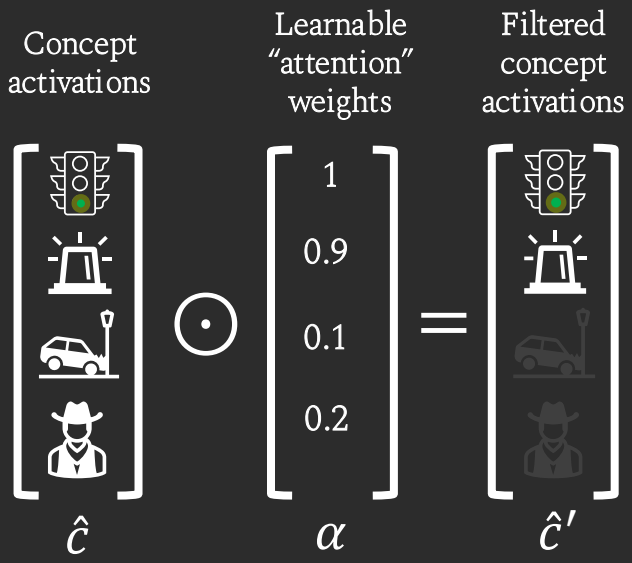
Idea: Can we use this approach to construct a truth table/logic rule mapping concepts to downstream tasks?

Problem: # of interventions required for this is exponential on the # of concepts!

Efficiently constructing logic-based predictors

One approach for efficiently construct a logic-based predictor is:

Step 1: Filter concept activations using learnable *attention weights* α

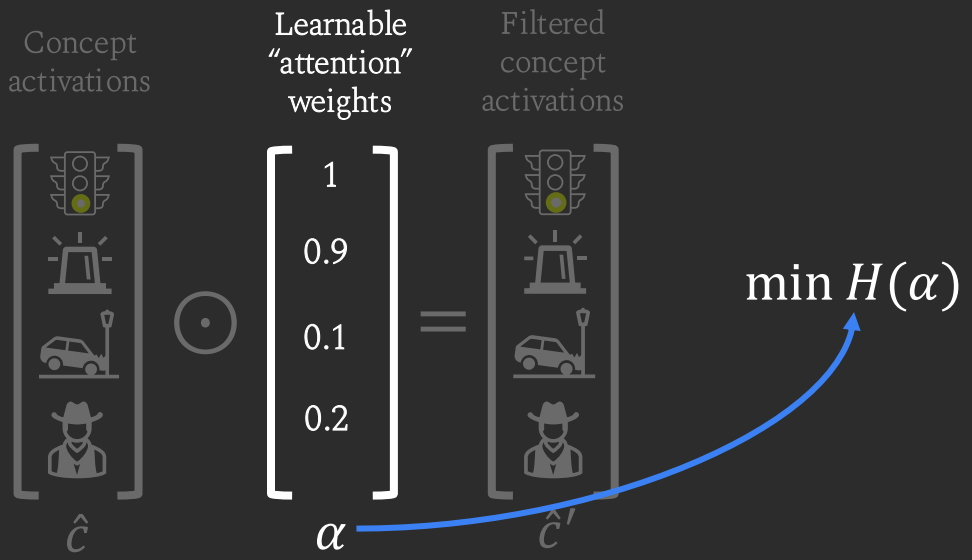


[1] Barbiero et al. "Entropy-based logic explanations of neural networks." AAAI (2022).

Efficiently constructing logic-based predictors

One approach for efficiently construct a logic-based predictor is:

Step 2: Select a small number of concepts by minimizing the entropy of the attention weights α.

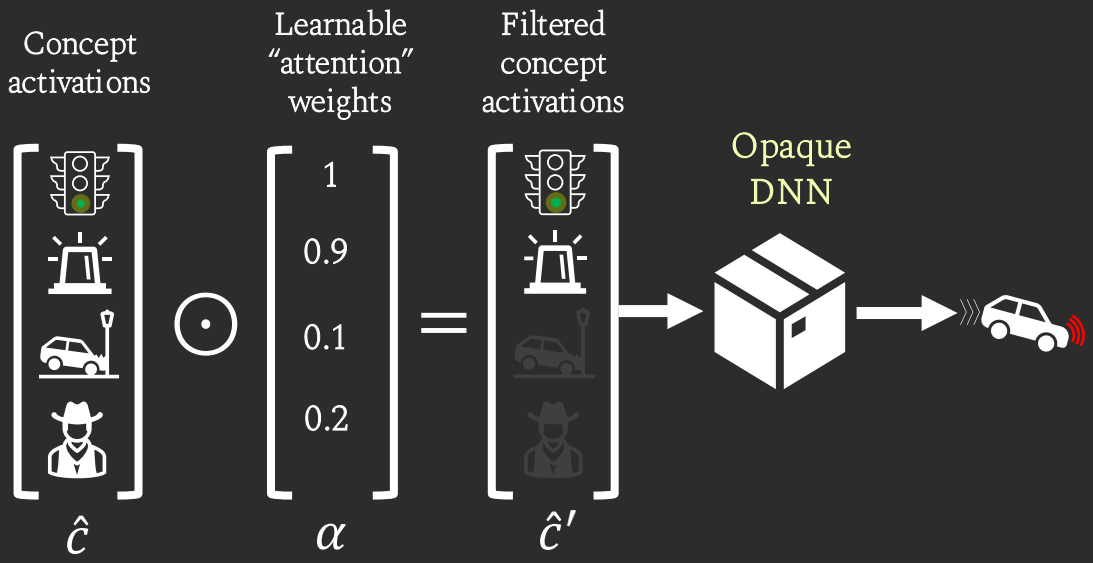


[1] Barbiero et al. "Entropy-based logic explanations of neural networks." AAAI (2022).

Efficiently constructing logic-based predictors

One approach for efficiently construct a logic-based predictor is:

Step 3: Solve task with the (small) set of selected concepts

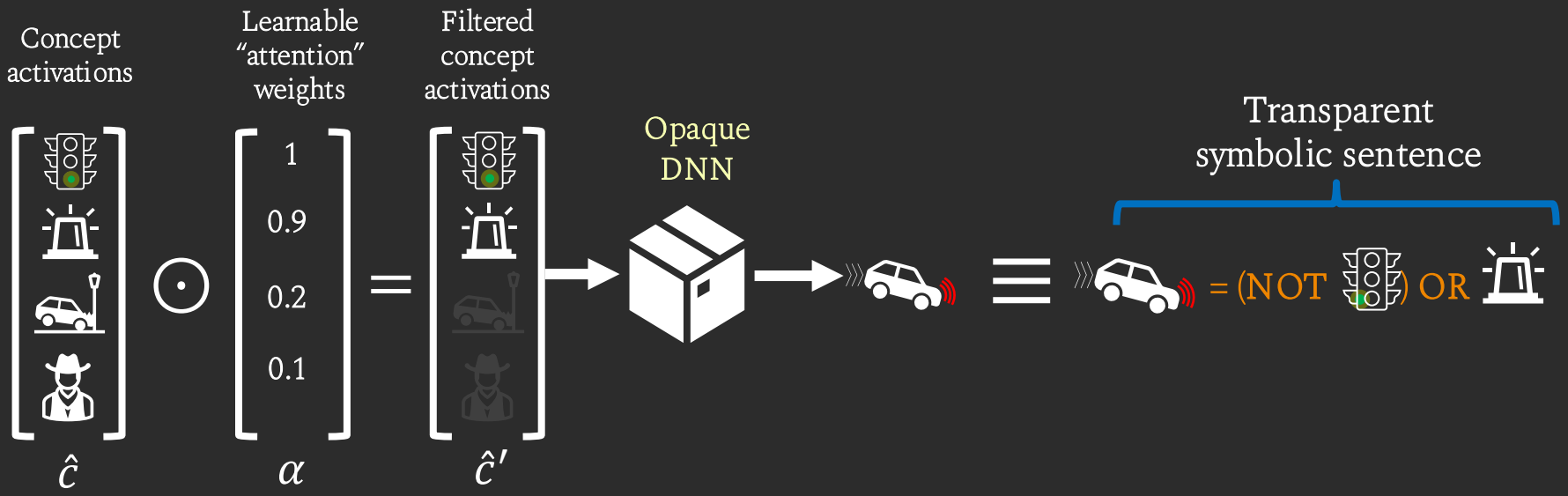


[1] Barbiero et al. "Entropy-based logic explanations of neural networks." AAAI (2022).

Efficiently constructing logic-based predictors

One approach for efficiently construct a logic-based predictor is:

Step 4: derive an explanation in DNF from the (empirical and much smaller) truth table

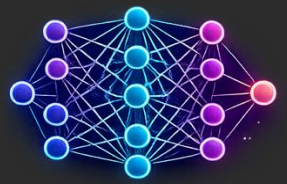


[1] Barbiero et al. "Entropy-based logic explanations of neural networks." AAAI (2022).

Neuro-symbolic task predictors

This logic predictor is an instance of a Neuro-Symbolic (NeSy) predictor

Deep Neural Networks



+

Logic/Reasoning

=

Neuro-Symbolic Predictors

Neuro-symbolic task predictors

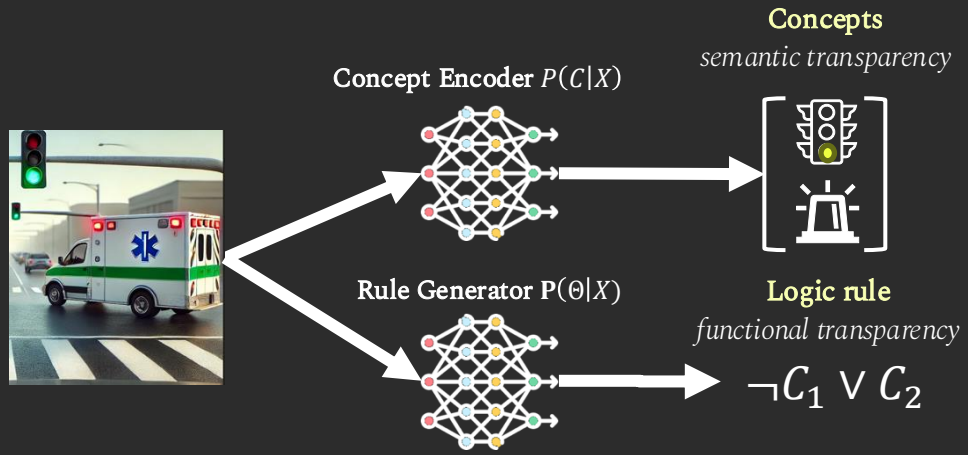
Generally, we can think of constructing a NeSy predictor as follows:

Neuro-symbolic task predictors

ICML 23 NeurIPS 24

Generally, we can think of constructing a NeSy predictor as follows:

Step 1: A DNN generates both concept activations & rule parameters (neural generation)



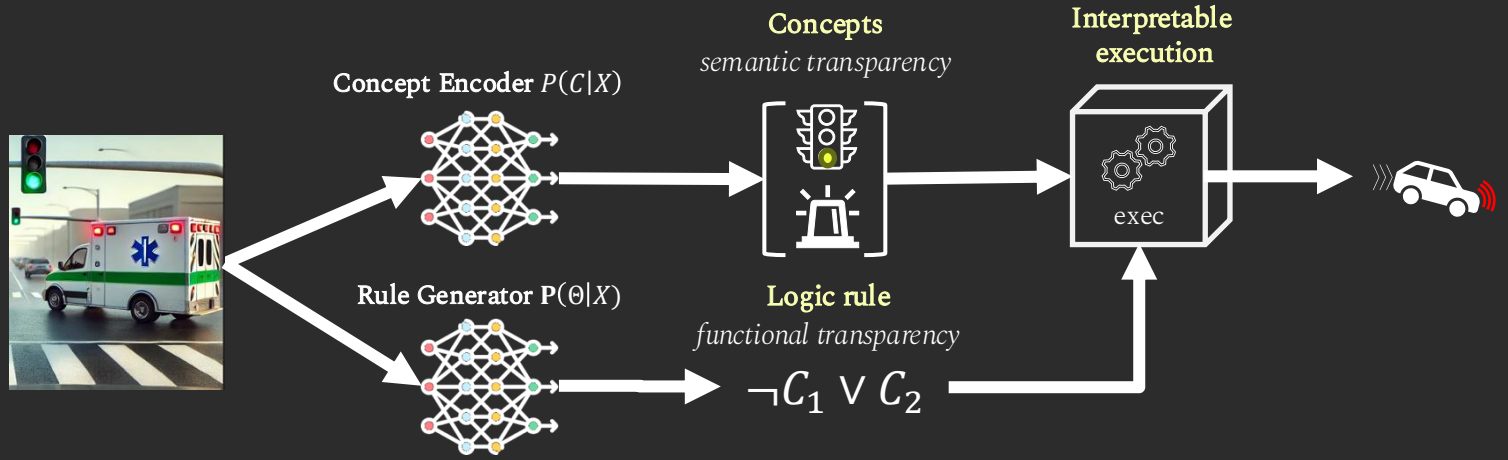
[1] Barbiero et al. "Interpretable neural-symbolic concept reasoning." International Conference on Machine Learning. PMLR, 2023.
[2] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Neuro-symbolic task predictors

ICML 23 NeurIPS 24

Generally, we can think of constructing a NeSy predictor as follows:

Step 1: A DNN generates both concept activations & rule parameters (neural generation)
Step 2: A symbolic engine executes rules using predicted concepts (interpretable execution)

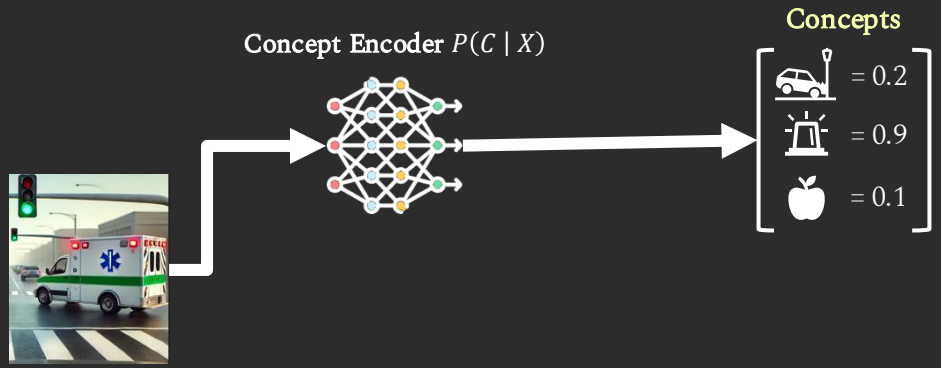


[1] Barbiero et al. "Interpretable neural-symbolic concept reasoning." International Conference on Machine Learning. PMLR, 2023.
[2] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Memory-based NeSy predictors

For example, we can build a concept-based memory reasoner (CMR) by:

Step 1: Predict a set of bounded scalar concept representations P(C | X) with a DNN

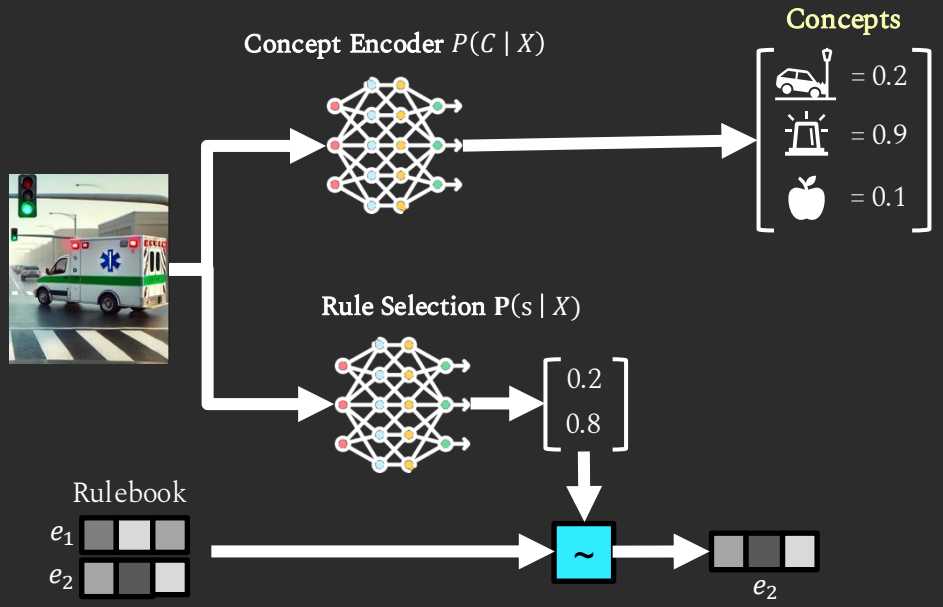


[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Memory-based NeSy predictors

For example, we can build a concept-based memory reasoner (CMR) by:

Step 2: Select a rule "embedding" from a learnable "rulebook" using a DNN

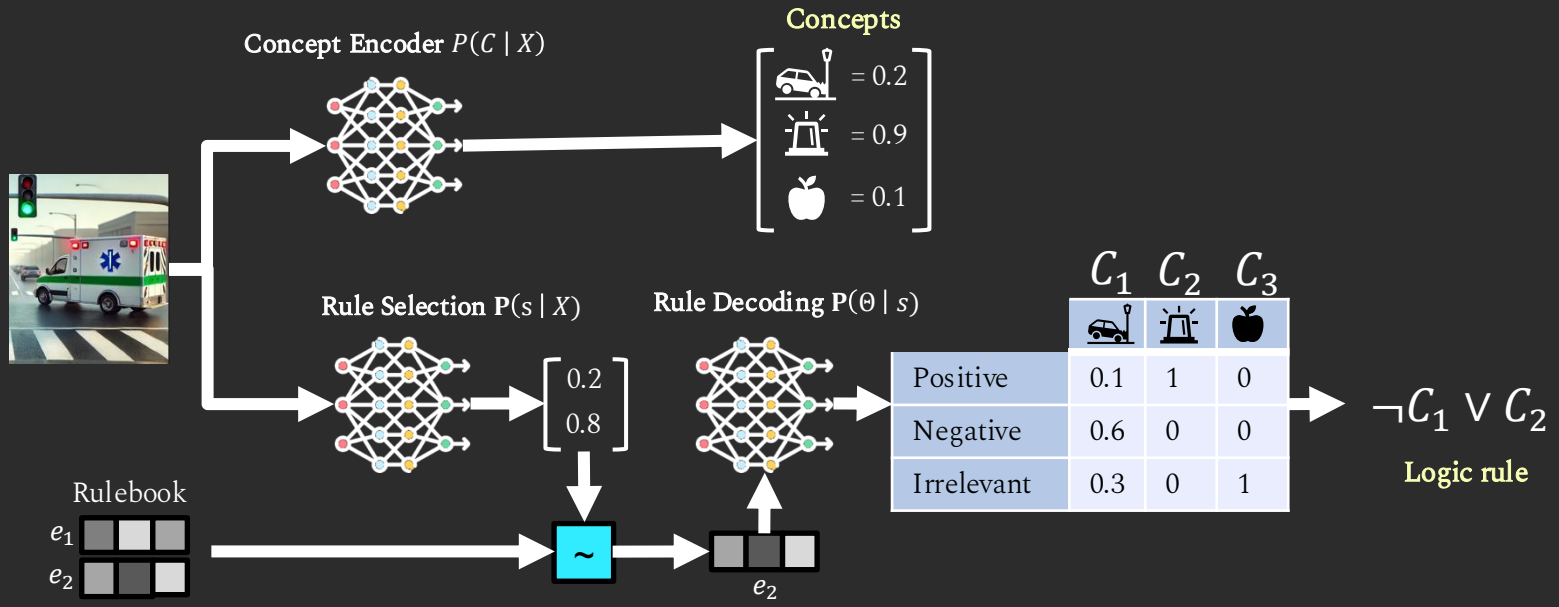


[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Memory-based NeSy predictors

For example, we can build a concept-based memory reasoner (CMR) by:

Step 3: Use a learnable model to decode the rule embedding into three states per concept

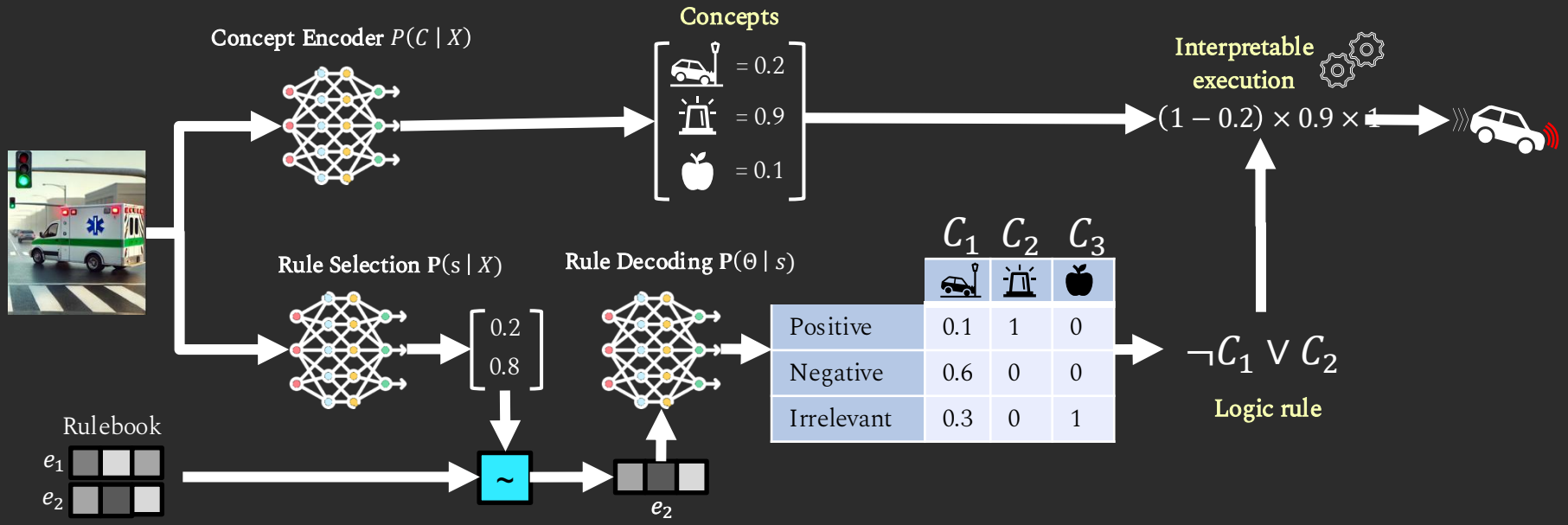


[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Memory-based NeSy predictors

For example, we can build a concept-based memory reasoner (CMR) by:

Step 4: Execute the rule combining concept states and concept activations



[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Memory-based NeSy predictors

PyC has built-in **memory layers** which can be used to easily implement CMR-like models:

```
##### Backbone #####
phi_backbone = torch.nn.Sequential(
    torch.nn.Linear(n, z_dim),
    torch.nn.LeakyReLU(),
)

##### Concept Encoder #####
concept_encoder = LinearZC(
    in_features=z_dim,
    out_features=c_annotations.shape[1],
)

##### Selector #####
selector = SelectorZU(
    in_features=z_dim,
    memory_size=memory_size,
    exogenous_size=z_dim,
    out_features=n_tasks,
)

##### Label predictor #####
y_predictor = HyperLinearCUC(
    in_features_endogenous=k,  # number of concepts
    in_features_exogenous=z_dim,
    embedding_size=z_dim,
)

##### CMR #####
cmr = torch.nn.Sequential(
    phi_backbone,
    selector,
    concept_encoder,
    y_predictor,
)
```

Properties of CMR

These steps enable CMR to:

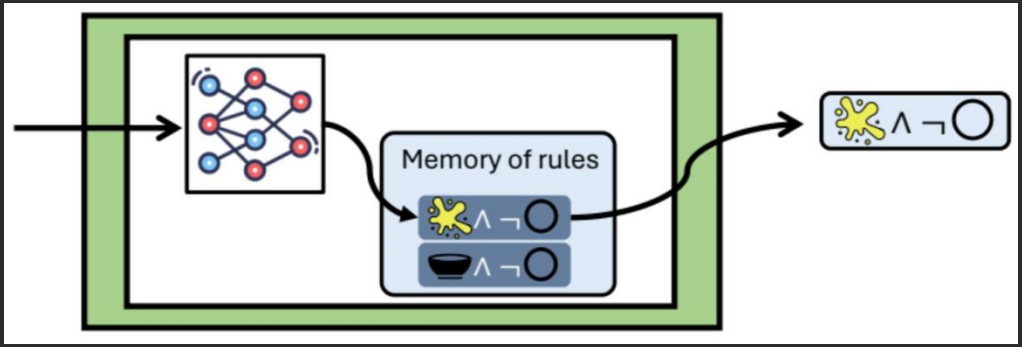
1. Become a universal approximator akin to opaque DNNs (Theorem 4.1)

[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Properties of CMR

These steps enable CMR to:

1. Become a universal approximator akin to opaque DNNs (Theorem 4.1)
2. Provide, by design, both local and global interpretability



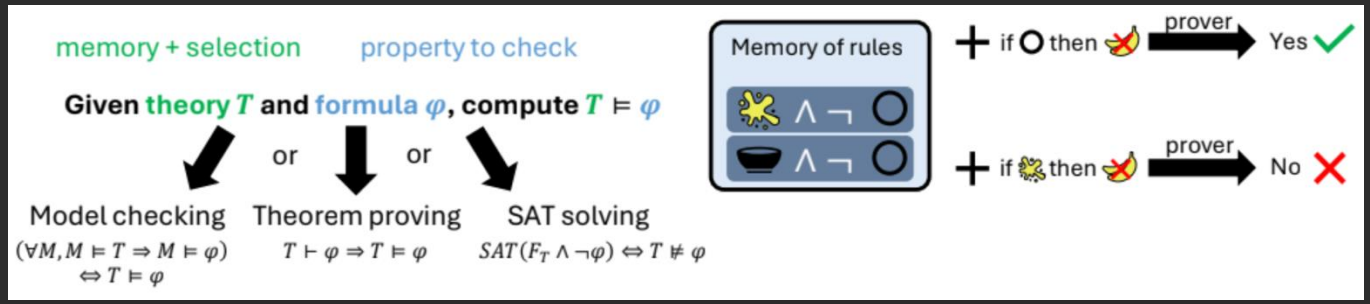
Inference mechanisms can only be selected from a finite set of transparent rules!

[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Properties of CMR

These steps enable CMR to:

1. Become a universal approximator akin to opaque DNNs (Theorem 4.1)
2. Provide, by design, both local and global interpretability
3. Allow formal verification of desirable properties



"Does a property hold no matter which rule is selected?"

[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Further neuro-symbolic task predictors

Other neuro-symbolic task predictors include:

Deep Concept Reasoner (DCR)

DeepProbLog

Neural Algorithmic Reasoning (NAR)

[1] Manhaeve et al. "Deepproblog: Neural probabilistic logic programming." NeurIPS (2018).
[2] Barbiero et al. "Interpretable neural-symbolic concept reasoning." ICML (2023).
[3] Veličković et al. "Neural algorithmic reasoning." Patterns (2021).

A taster of label predictors

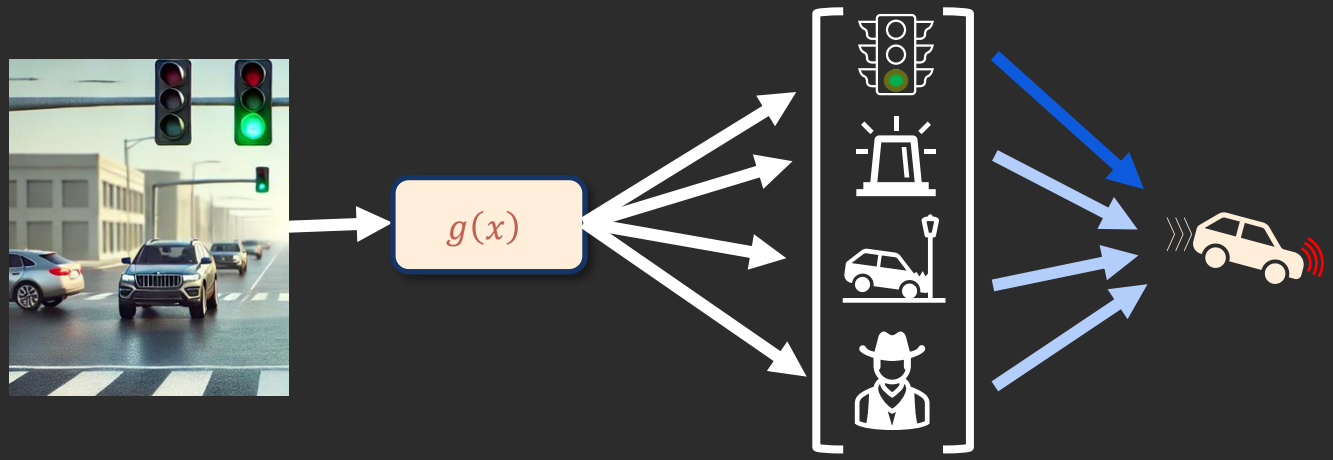
Locally-interpretable Predictors

Neuro-Symbolic Predictors

Causal Predictors

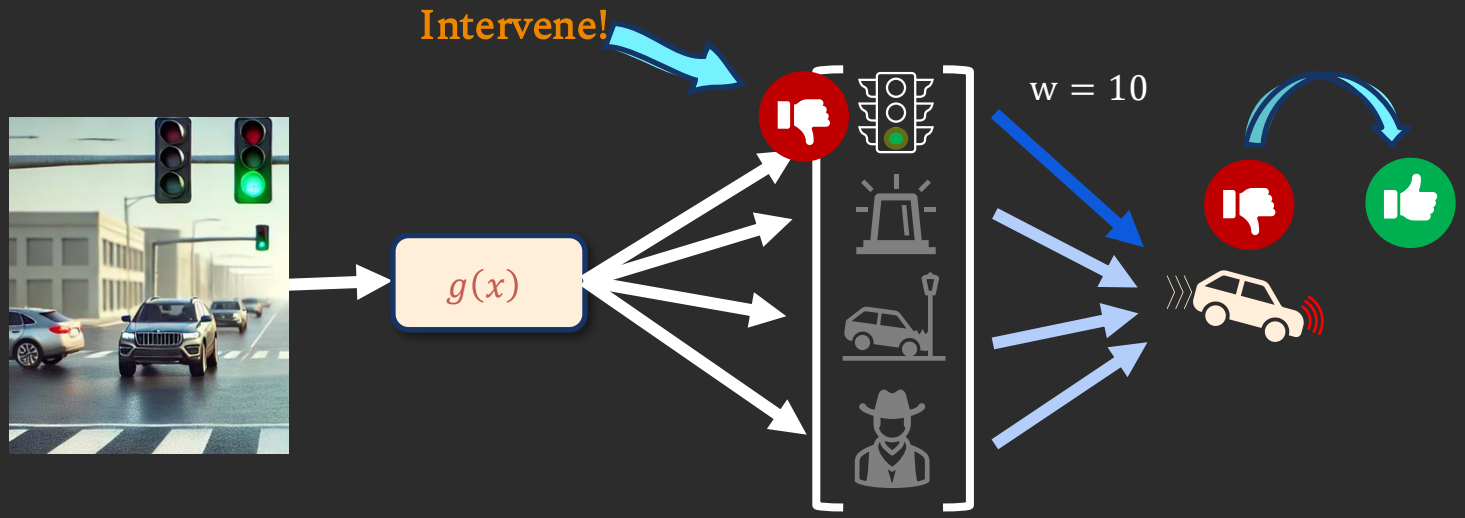
Causal task predictors

Let's consider interventions on a CBM-like model:



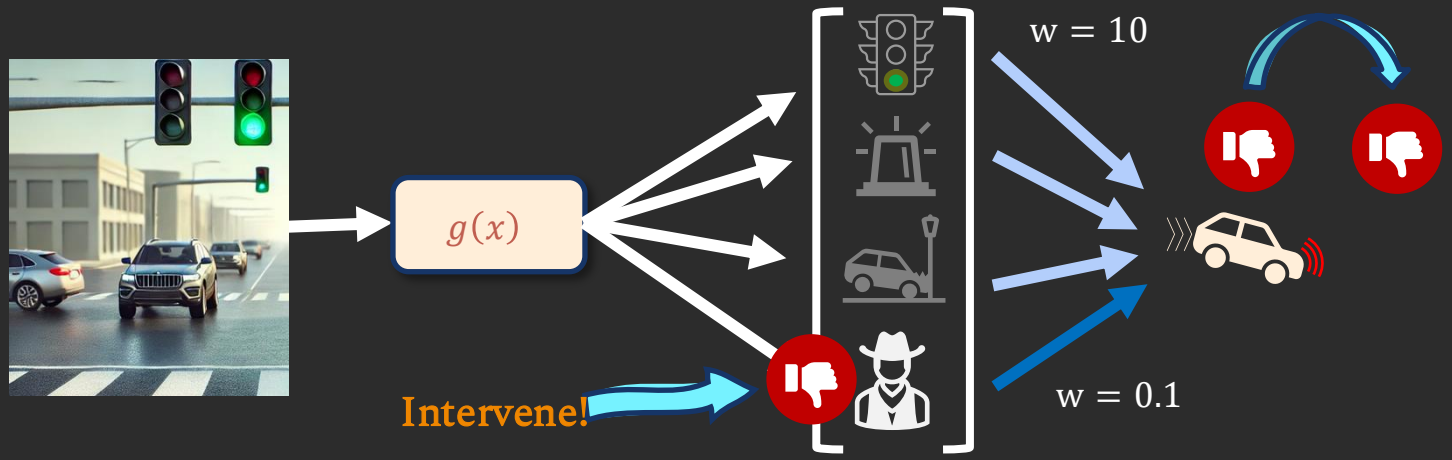
Causal task predictors

Sometimes intervening on a mispredicted helps…



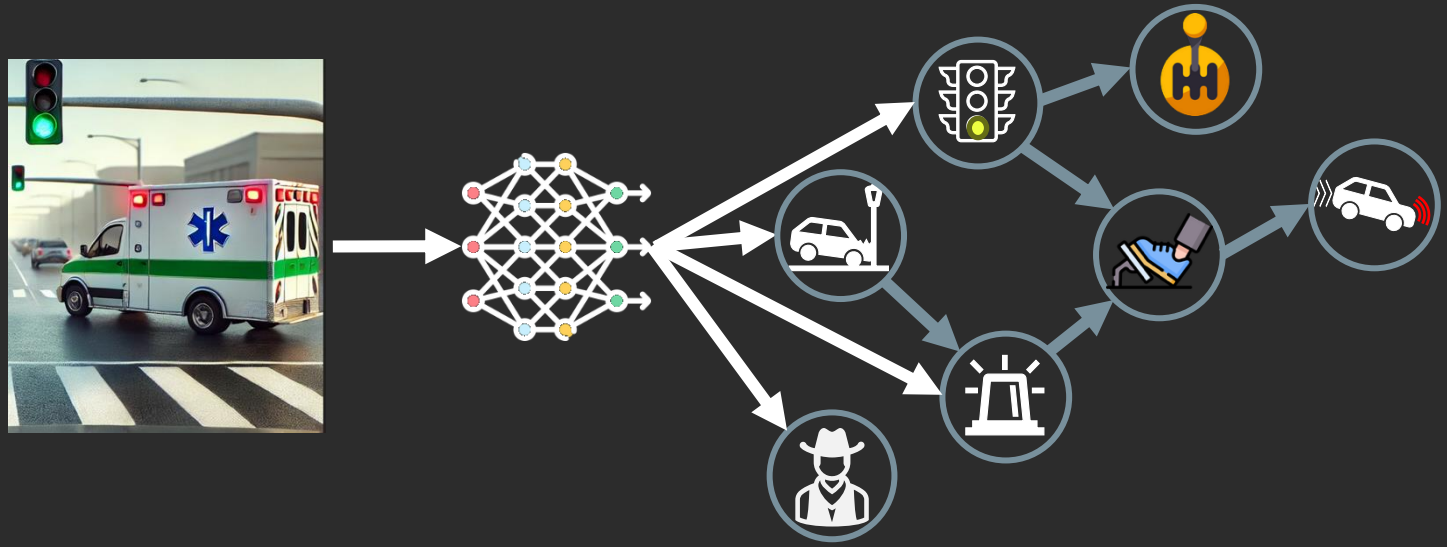
Causal task predictors

Sometimes intervening on a mispredicted concepts helps… but sometimes it doesn't!



Causal task predictors

Causal predictors explain how concepts affect other concepts and task!



Causal opacity

However, for this it is important to understand make a distinction between:

1. Causal reliability
2. Causal opacity

Causal opacity

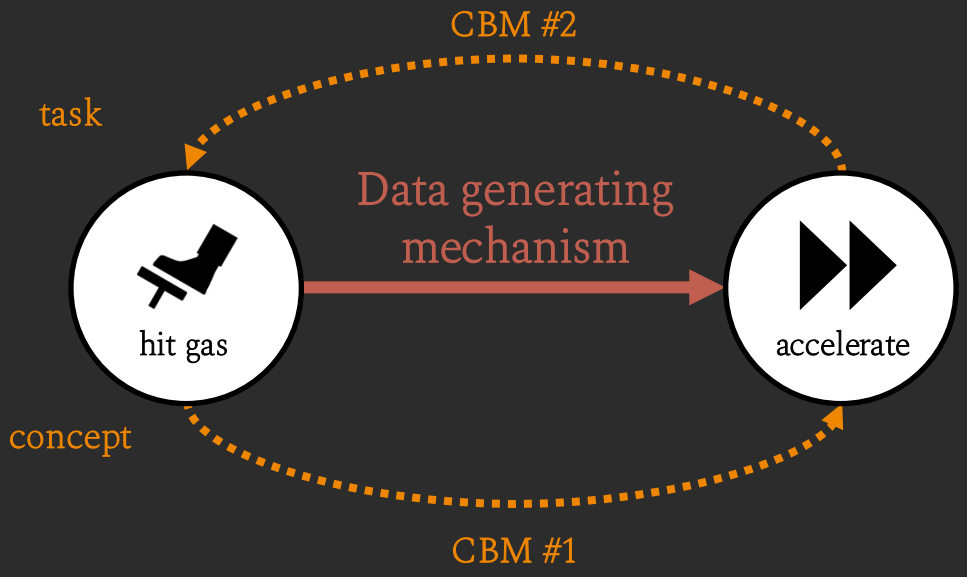
However, for this it is important to understand make a distinction between:

1. Causal reliability: discover causal mechanisms of the data generating process

Causal opacity

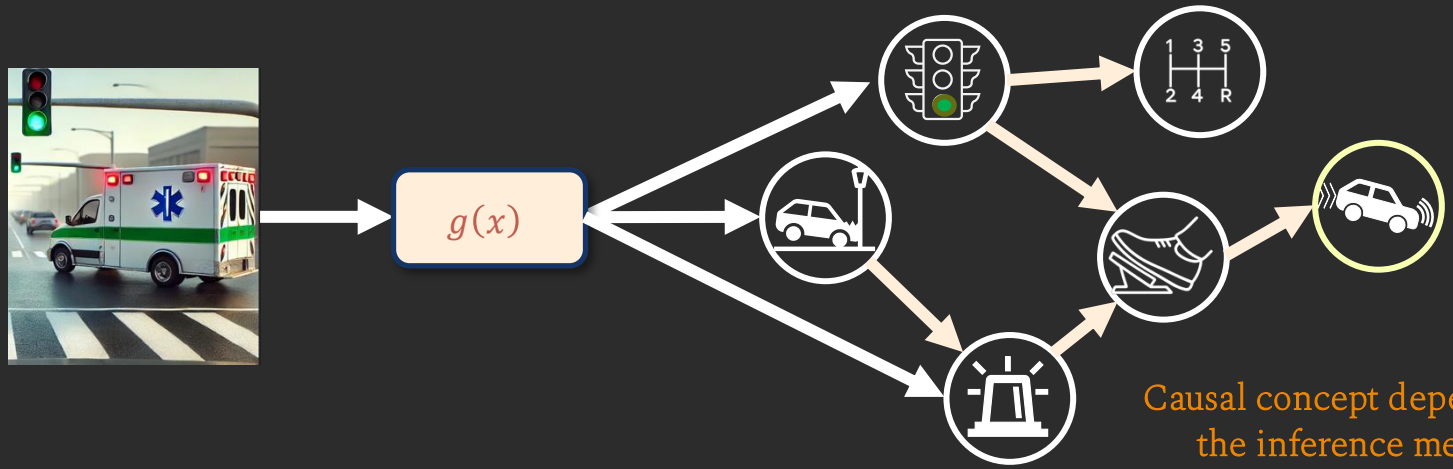
However, for this it is important to understand make a distinction between:

1. Causal reliability: discover causal mechanisms of the data generating process
2. Causal opacity: discover causal mechanism of a model's inference process



Concept graph models

We can build a causal label predictor that explicitly uses a concept graph:



[1] Dominici et al. "Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning." ICLR 2025.
[2] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR 2024.

Concept graph models

The concept graph itself can be:

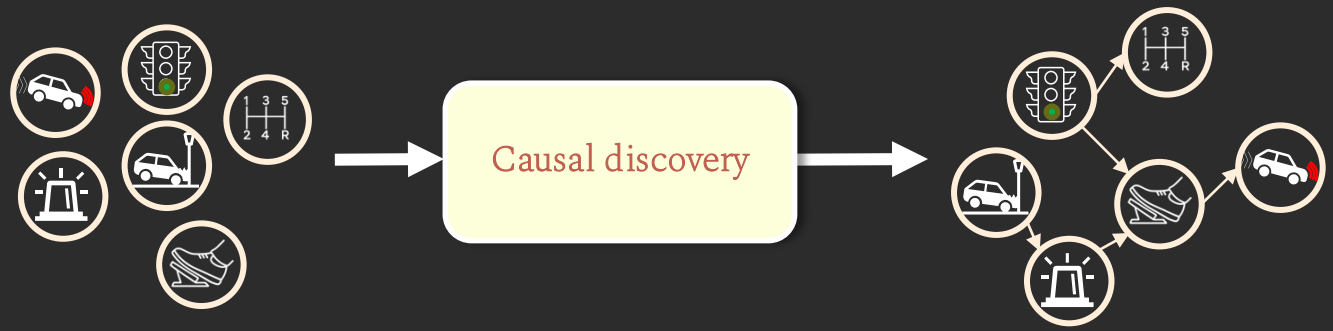
1. Given as a prior

[1] Dominici et al. "Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning." ICLR 2025.
[2] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR 2024.

Concept graph models

The concept graph itself can be:

1. Given as a prior
2. Learnt from data via causal discovery methods

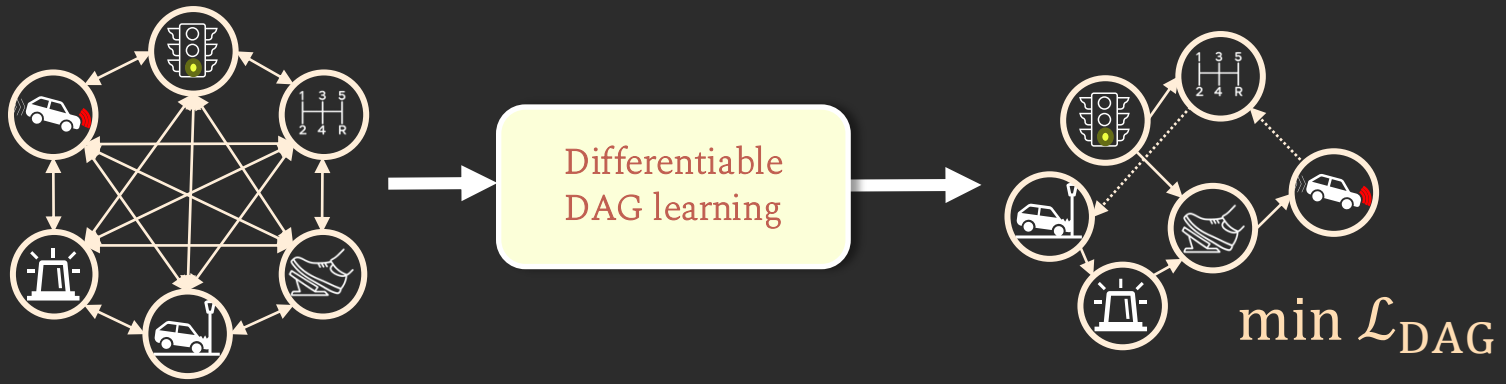


[1] Dominici et al. "Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning," ICLR 2025.
[2] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation," CLeaR 2024.

Concept graph models

The concept graph itself can be:

1. Given as a prior
2. Learnt from data via causal discovery methods
3. Learnt end-to-end using differentiable DAG learning



[1] Dominici et al. "Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning." ICLR 2025.
[2] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR 2024.

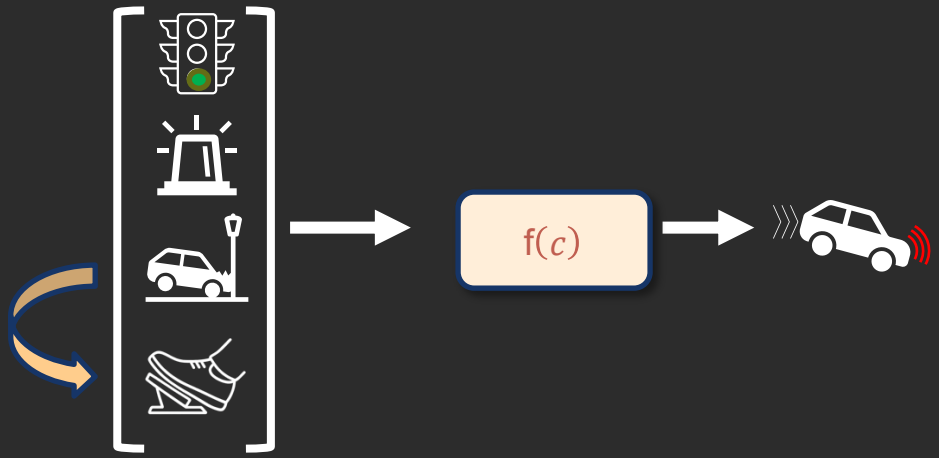
Concept graph models

Causal concept task allow us to get rid of two unrealistic assumptions:

Concept graph models

Causal concept task allow us to get rid of two unrealistic assumptions:

1. **Concept Independence**
Traditional concept predictors assume all concepts are **independent** (e.g., intervening on "*car crash*" does not affect "braking")

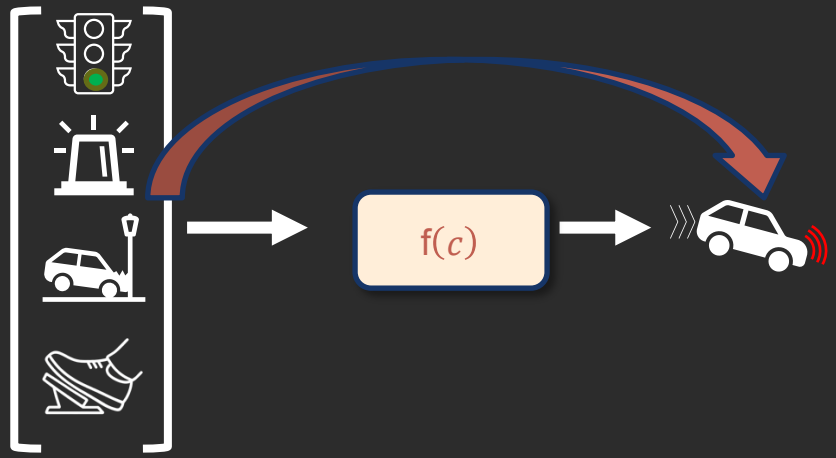


Concept graph models

Causal concept task allow us to get rid of two unrealistic assumptions:

1. **Concept Independence**
Traditional concept predictors assume all concepts are **independent** (e.g., intervening on "*car crash*" does not affect "braking")

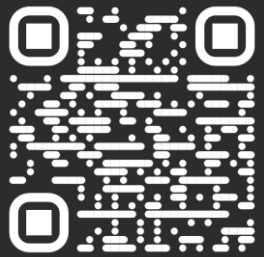
2. **Direct Causal Relationships**
Traditional task predictors assume concepts are **direct** causes of the task



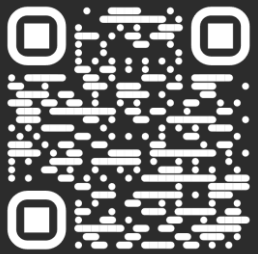
Further causal task predictors

Recent causal task predictors include:

DiConstruct



Causally Interpretable Models



[1] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR (2024).
[2] Hwang et al. "From black-box to causal-box: Towards building more interpretable models." NeurIPS (2025).

Tutorial Outline

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences

Q&A + 30 min break

IV. Designing Concept Representations

V. Designing Task Predictors

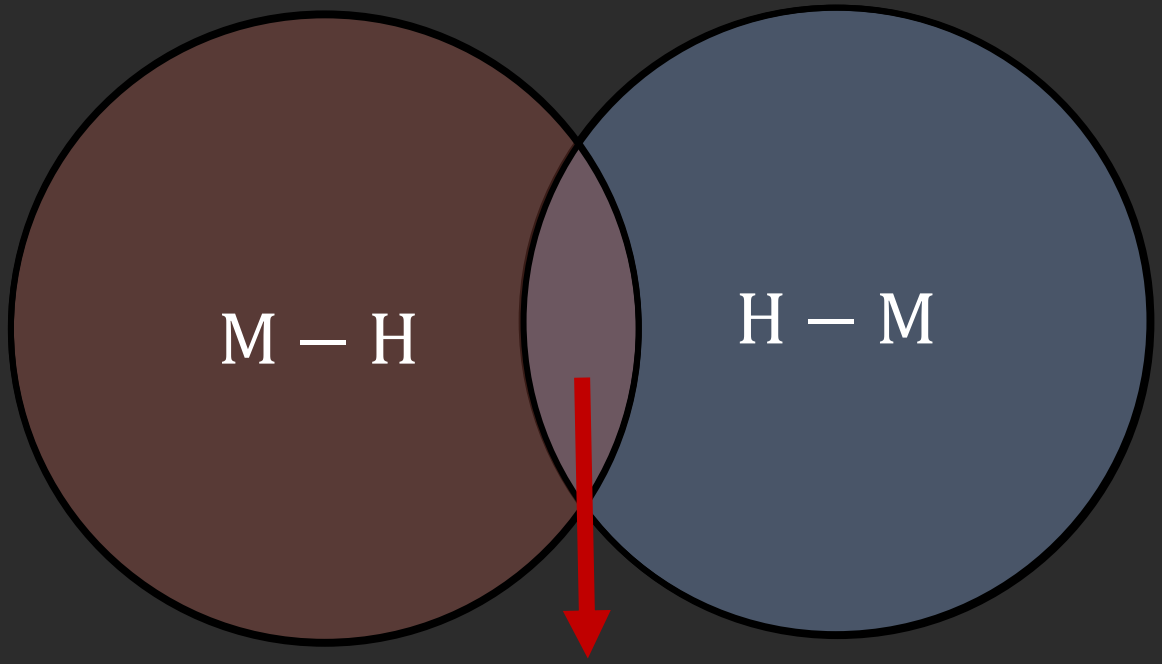
VI. Human-AI Interaction

VII. Conclusion

Final Q&A

Thinking of concepts as a shared language

M = Machine Representational Space H = Human Representational Space



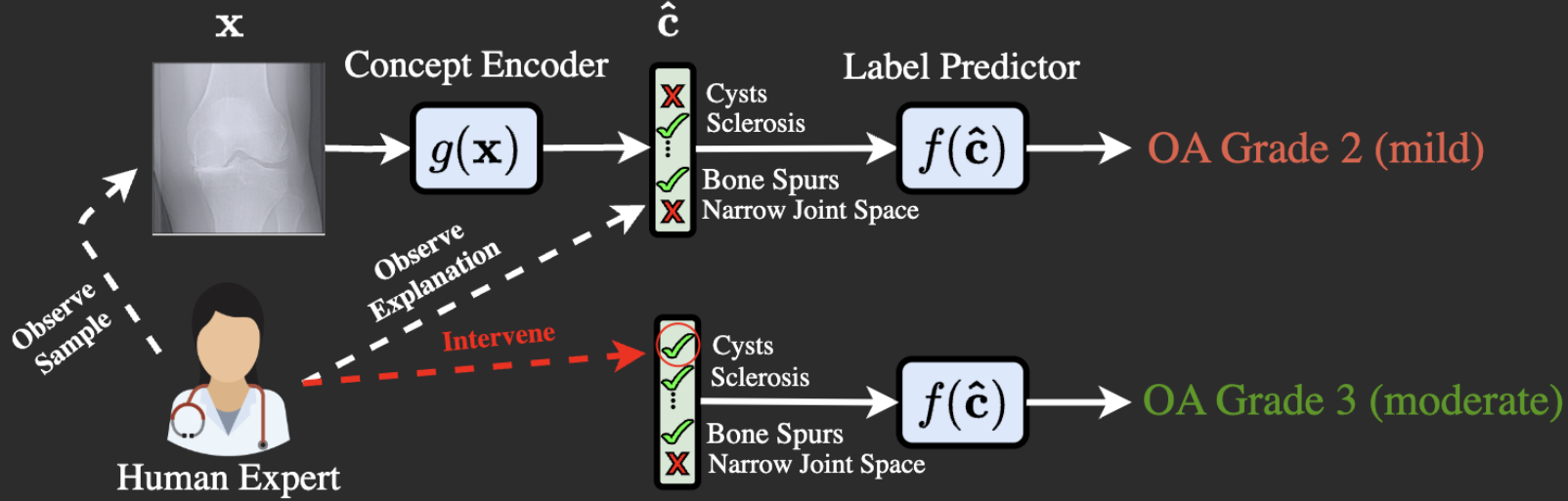
M ∩ H = Human-like Concepts

[1] **Heavily i**nspired by Schut et al. "Bridging the human-ai knowledge gap: Concept discovery and transfer in alphazero." arXiv (2023).

The power of concept interventions

This means that concept interventions can be used to:

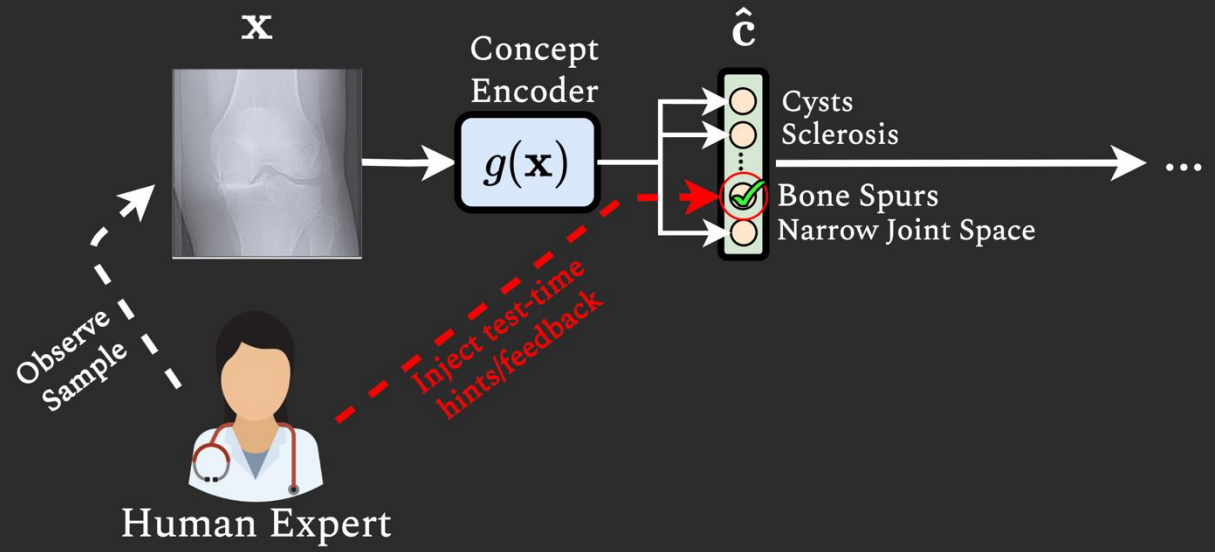
1. Correct mispredicted concepts at test-time



The power of concept interventions

This means that concept interventions can be used to:

1. Correct mispredicted concepts at test-time
2. Inject knowledge that cannot be easily expressed in the feature space



The illusion of concept equipotency

When intervening, it is important to realise that some concepts are:

1. Less informative than others (e.g., redundant w.r.t. other concepts)
2. Less certain than others (e.g., due to occlusions or inherent difficulties)

To identify a Raven from a Crow, "*tail shape*" is more informative than "*wing color*"

Concept "Belly Color" is partially occluded

The illusion of concept equipotency

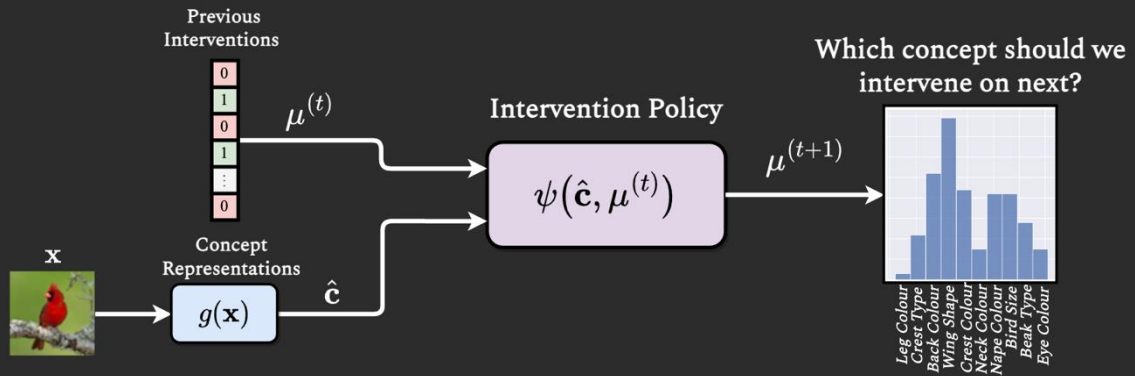
When intervening, it is important to realise that some concepts are:

1. Less informative than others (e.g., redundant w.r.t. other concepts)
2. Less certain than others (e.g., due to occlusions or inherent difficulties)

Hence, an intervention's effectiveness **depends on the intervened concept!**

Selecting meaningful concepts: intervention policies

Intervention policies select which concept to intervene on next by assigning each concept c_i an importance score s_i :



Given x and concept predictions ĉ, what concept should I intervene on next to minimize my model's task uncertainty?

A sampler of intervention policies

Policies should balance (1) concept uncertainty , and (2) concept importance

[1] Shin et al. "A Closer Look at the Intervention Procedure of Concept Bottleneck Models." ICML 2023.

A sampler of intervention policies

Policies should balance (1) concept uncertainty , and (2) concept importance

Uncertainty of concept prediction

Select the concept c_i with the highest predicted entropy $s_i = \mathcal{H}(\hat{c}_i)$

[1] Shin et al. "A Closer Look at the Intervention Procedure of Concept Bottleneck Models." ICML 2023.

A sampler of intervention policies

Policies should balance (1) concept uncertainty, and (2) concept importance

Uncertainty of concept prediction

Select the concept c_i with the highest predicted entropy $s_i = \mathcal{H}(\hat{c}_i)$

Contribution of concept on target prediction

Select the concept c_i with the highest contribution on target prediction $s_i = \sum_{j=1}^{L} \left| \hat{c}_i \frac{\partial f_j(x)}{\partial \hat{c}_i} \right|$.

[1] Shin et al. "A Closer Look at the Intervention Procedure of Concept Bottleneck Models." ICML 2023.

A sampler of intervention policies

Policies should balance (1) concept uncertainty, and (2) concept importance

Uncertainty of concept prediction

Select the concept c_i with the highest predicted entropy $s_i = \mathcal{H}(\hat{c}_i)$

Contribution of concept on target prediction

Select the concept c_i with the highest contribution on target prediction $s_i = \sum_{j=1}^{L} \left| \hat{c}_i \frac{\partial f_j(x)}{\partial \hat{c}_i} \right|$.

Expected change in target prediction (ECTP)

Select the concept c_i with the highest expected change in the target predictive distribution:

$$s_i = \mathbb{E}_{v \sim \text{Bern}(\hat{c}_i)} \left[D_{KL}\left(\hat{y}_{c_i:=v} || \hat{y} \right) \right]$$

[1] Shin et al. "A Closer Look at the Intervention Procedure of Concept Bottleneck Models." ICML 2023.

A sampler of intervention policies

Policies should balance (1) concept uncertainty, and (2) concept importance

Uncertainty of concept prediction

Select the concept c_i with the highest predicted entropy $s_i = \mathcal{H}(\hat{c}_i)$

Contribution of concept on target prediction

Select the concept c_i with the highest contribution on target prediction $s_i = \sum_{j=1}^{L} \left| \hat{c}_i \frac{\partial f_j(x)}{\partial \hat{c}_i} \right|$.

Expected change in target prediction (ECTP)

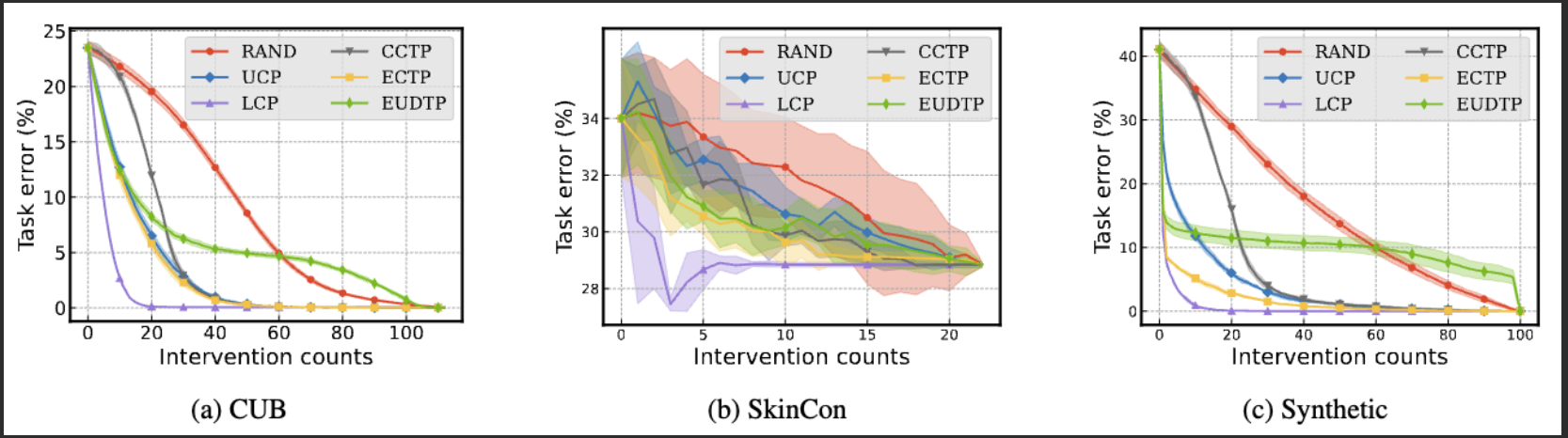
Select the concept c_i with the highest expected change in the target predictive distribution:

$$s_i = (1 - \hat{c}_i) D_{KL}\left(\hat{y}_{\hat{c}_i=0} || \hat{y}\right) + \hat{c}_i D_{KL}\left(\hat{y}_{\hat{c}_i=1} || \hat{y}\right)$$

[1] Shin et al. "A Closer Look at the Intervention Procedure of Concept Bottleneck Models." ICML 2023.

A sampler of intervention policies

This leads to significantly different intervention curves:



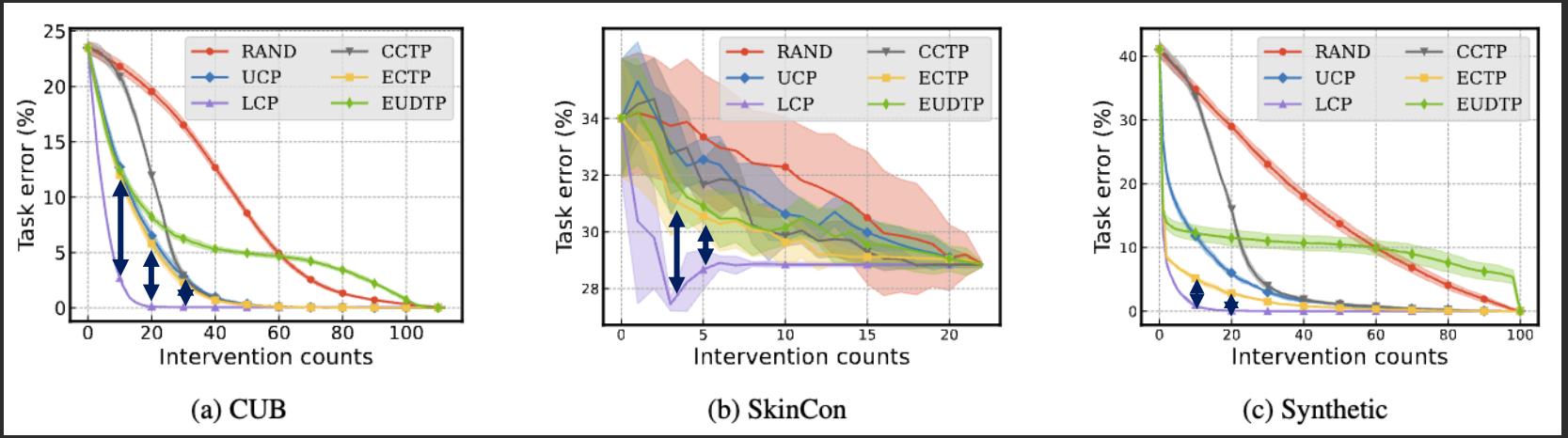
Best performing non-oracle policy is Expected change in target prediction (ECTP).
Even Uncertainty of concept prediction (UCP) significantly outperforms random concept selection (RAND)

(**Intuition**: one should select the concept leading to the highest expected change in the task's distribution)

[1] Shin et al. "A Closer Look at the Intervention Procedure of Concept Bottleneck Models." ICML 2023.

A sampler of intervention policies

This leads to significantly different intervention curves:



We still observe a significant gap between the best policy and an optimal greedy policy (LCP)

[1] Shin et al. "A Closer Look at the Intervention Procedure of Concept Bottleneck Models." ICML 2023.

Combining policies: CooP

It may be possible to shorten this gap by learning a weighting between the concept uncertainty and the expected intervention effect

Cooperative Prediction Intervention Policy (CooP)

$$s_i = \alpha \mathcal{H}(\hat{c}_i) + \beta \left| E_{v \sim p(c_i \mid x)} \left[\hat{y}_{\hat{c}_i = v} - \hat{y} \right] \right| + \gamma q_i$$

[1] Chauhan et al. "Interactive concept bottleneck models." AAAI (2023).

Combining policies: CooP

It may be possible to shorten this gap by learning a weighting between the concept uncertainty and the expected intervention effect

Cooperative Prediction Intervention Policy (CooP)

$$s_i = \alpha \mathcal{H}(\hat{c}_i) + \beta \left| \mathrm{E}_{v \sim p(c_i \mid x)} \left[\hat{y}_{\hat{c}_i = v} - \hat{y} \right] \right| + \gamma q_i$$

Uncertainty of concept prediction

Cost of the intervention

Expected change to the task prediction if we were to intervene on c_i based on c_i's current prediction

[1] Chauhan et al. "Interactive concept bottleneck models." AAAI (2023).

Learning policies: insights

We notice two useful insights that may help us learn better policies:

1. Access to optimal interventions: during training, we can compute the optimal intervention if we have access to ground truth concept labels

$$c_*(\mathbf{x}, \mu, \mathbf{c}, y) = \arg\max_{1 \leq i \leq k} f\Big(g(\mathbf{x} \mid C_\mu = \mathbf{c}_\mu, C_i = \mathbf{c}_i)\Big)_y$$

(Translation: Attempt every intervention and select that one maximises the ground truth label's confidence)

[1] Espinosa Zarlenga et al. "Learning to receive help: Intervention-aware concept embedding models." NeurIPS (2023)

Learning policies: insights

We notice two useful insights that may help us learn better policies:

1. Access to optimal interventions
2. Interventions are differentiable: when using embeddings, we can write down concept representations that are differentiable w.r.t. the intervention mask

Original Embedding Construction

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

Intervened Embedding Construction

$$\hat{\boldsymbol{c}}_i(\boldsymbol{x}) := \Big(\mu_i c_i + (1 - \mu_i)\hat{p}_i(x)\Big)\,\hat{\boldsymbol{c}}_i^+(\boldsymbol{x}) + \Big(1 - \Big(\mu_i c_i + (1 - \mu_i)\hat{p}_i(x)\Big)\Big)\,\hat{\boldsymbol{c}}_i^-(\boldsymbol{x})$$

Whether we intervene on the i-th concept (can be relaxed to be in [0,1])

[1] Espinosa Zarlenga et al. "Learning to receive help: Intervention-aware concept embedding models." NeurIPS (2023)

Learning policies: insights

We notice two useful insights that may help us learn better policies:

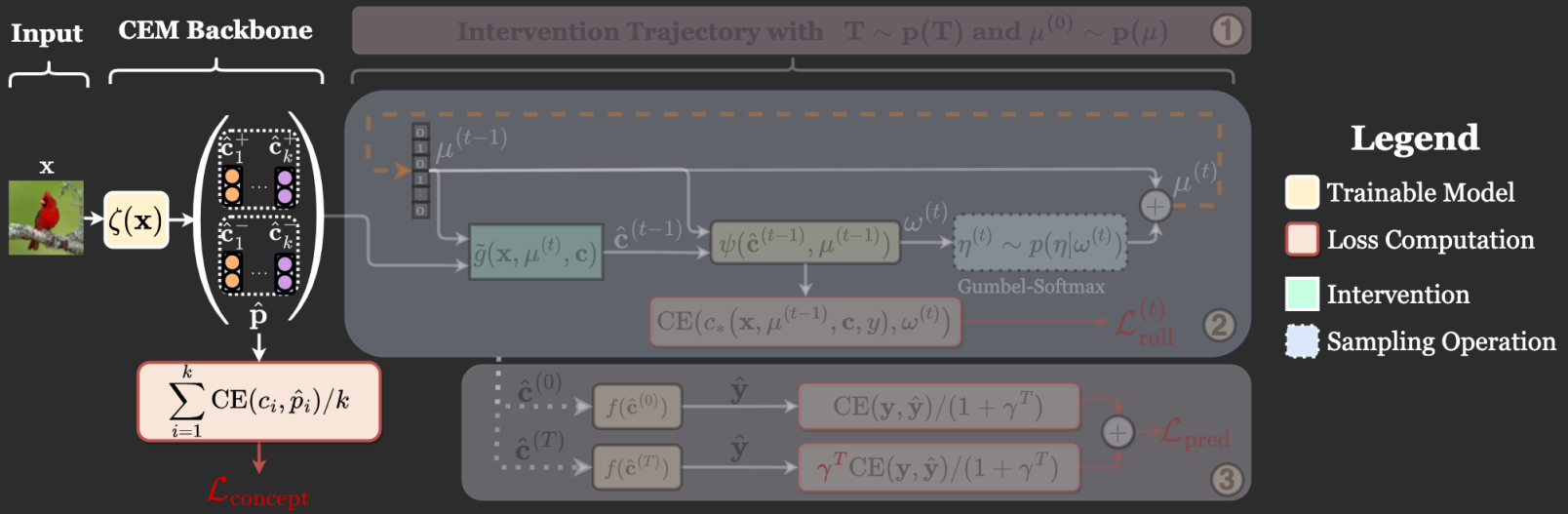
1. Access to optimal interventions
2. Interventions are differentiable

Together these two mean we can learn an intervention policy via gradient descent!

[1] Espinosa Zarlenga et al. "Learning to receive help: Intervention-aware concept embedding models." NeurIPS (2023)

Learning policies: training-time demonstrations

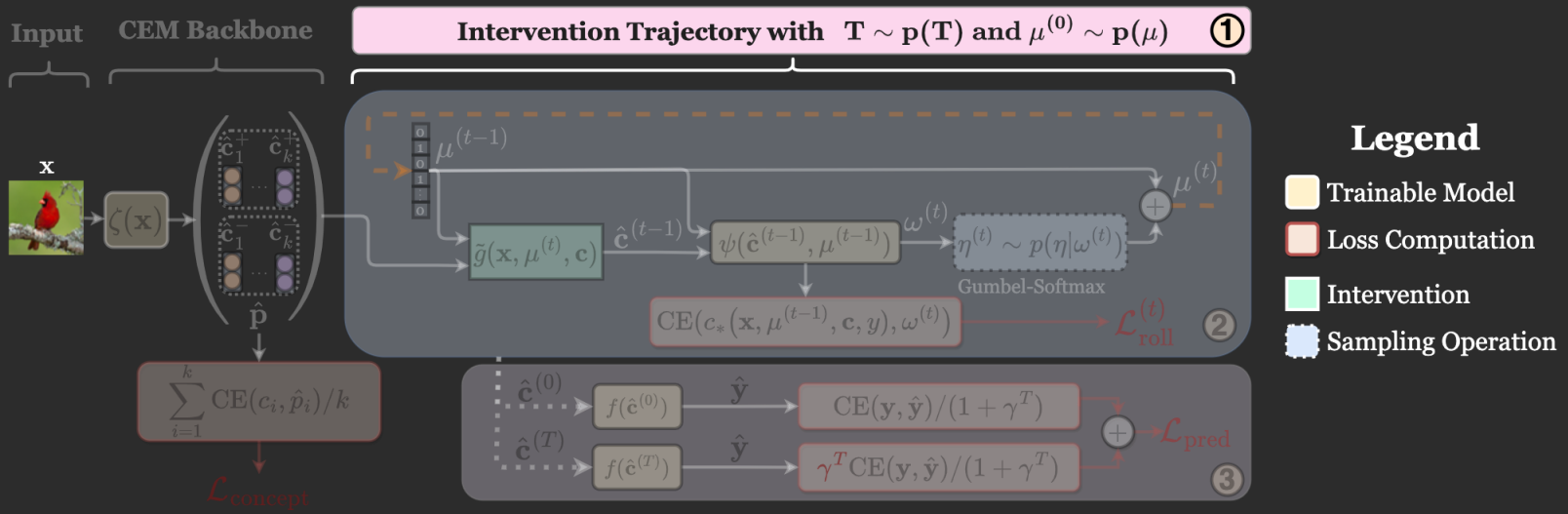
One way to achieve this is by introducing **training-time interventions**:



Step 0: Construct a positive and negative embedding representations for each concept

Learning policies: training-time demonstrations

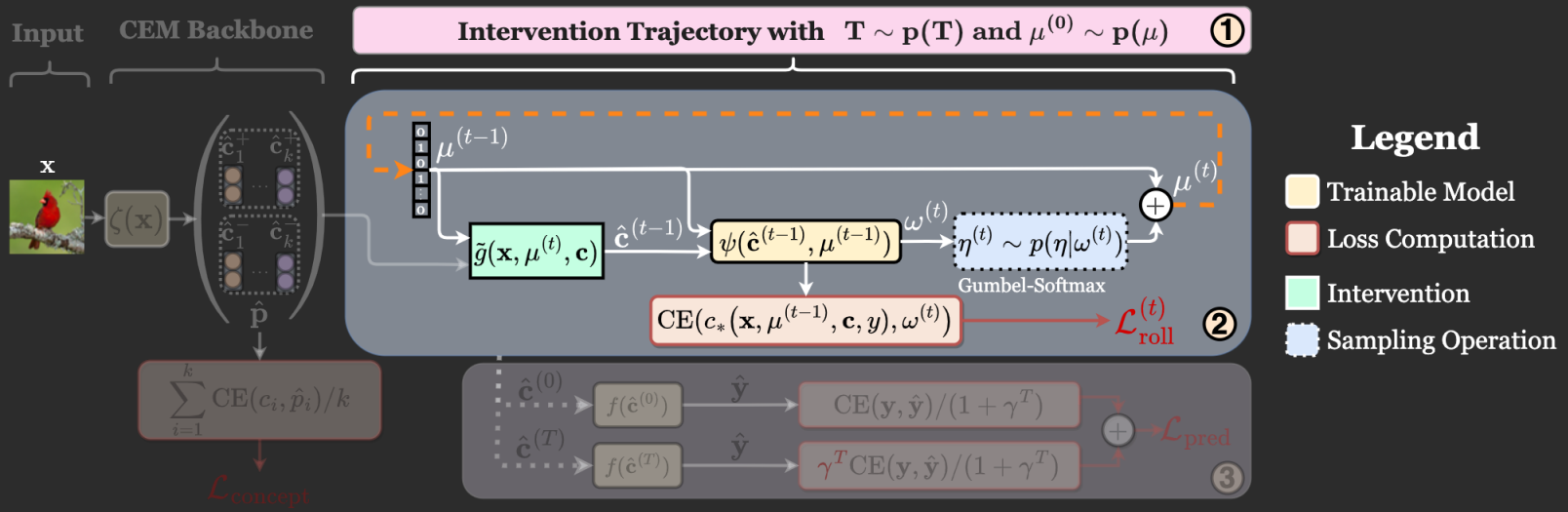
One way to achieve this is by introducing **training-time interventions**:



Step 1: Randomly select a subset of concepts which we will initially intervene on and a number of interventions T we will perform in this training step

Learning policies: training-time demonstrations

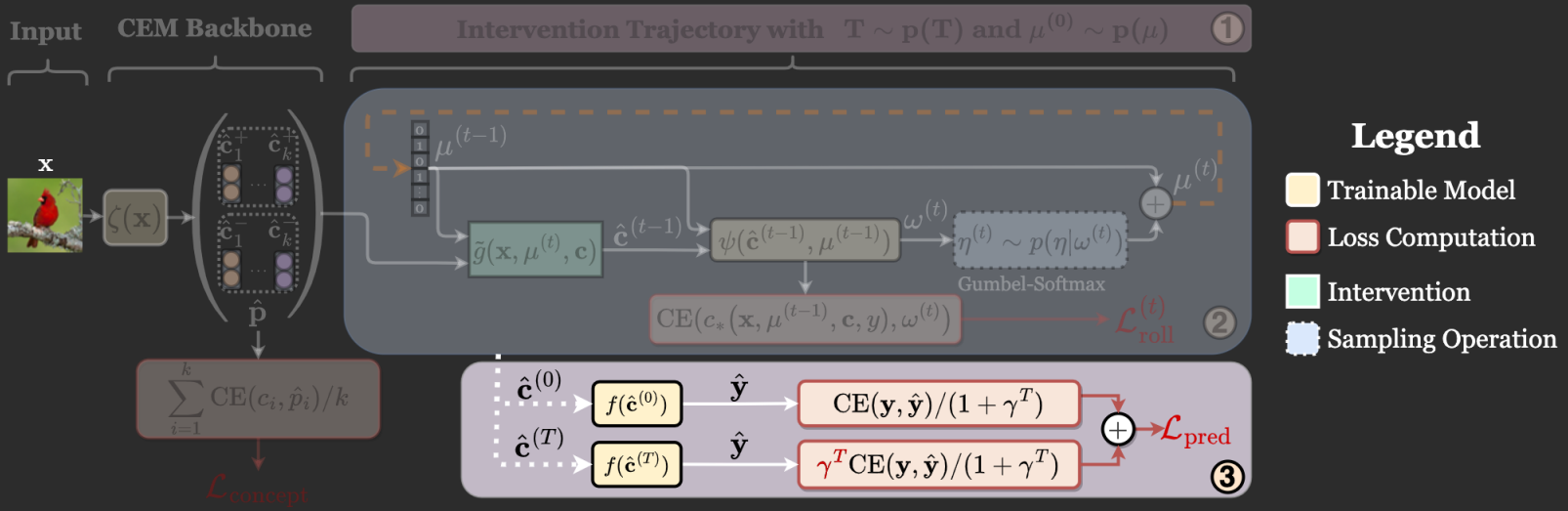
One way to achieve this is by introducing **training-time interventions**:



Step 2: Recursively sample a trajectory of T interventions from this set using a learnable intervention policy. We train this policy to align to the "oracle" optimal policy.

Learning policies: training-time demonstrations

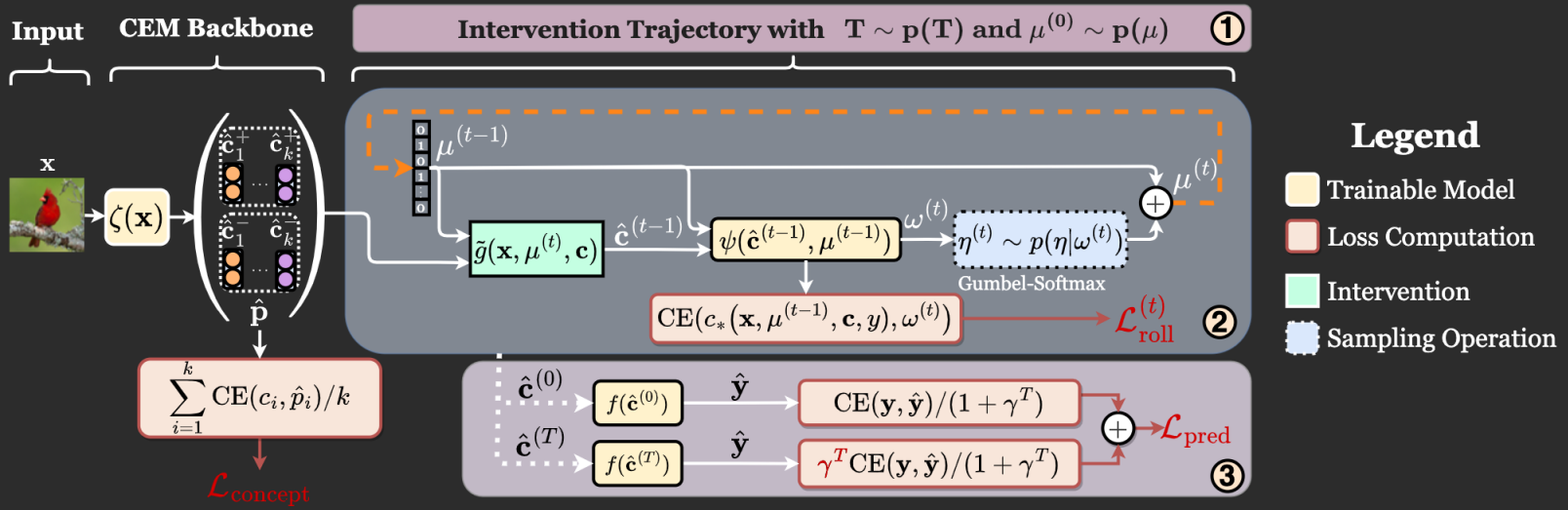
One way to achieve this is by introducing **training-time interventions**:



Step 3: Penalise the model more heavily for mispredicting the task label at the end of the intervention trajectory vs at the start of the trajectory

Intervention-aware interpretable models

All-together, this gives you an **intervention-aware** interpretable model!

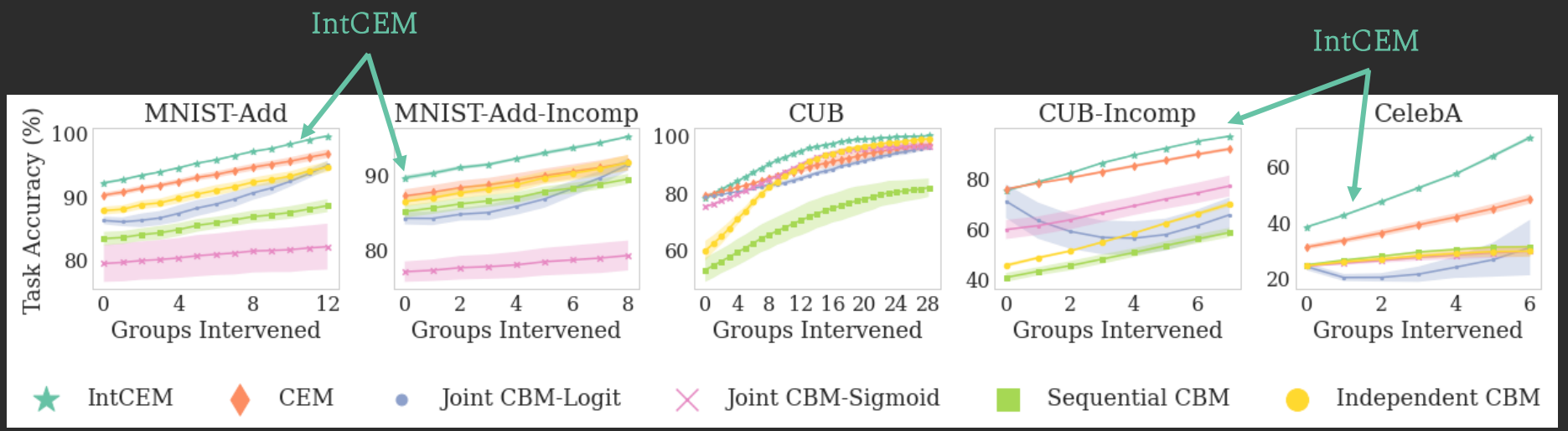


[1] Espinosa Zarlenga et al. "Learning to receive help: Intervention-aware concept embedding models." NeurIPS (2023)

Intervention-aware interpretable models

Intervention-aware models are:

1. Generally better at receiving any interventions

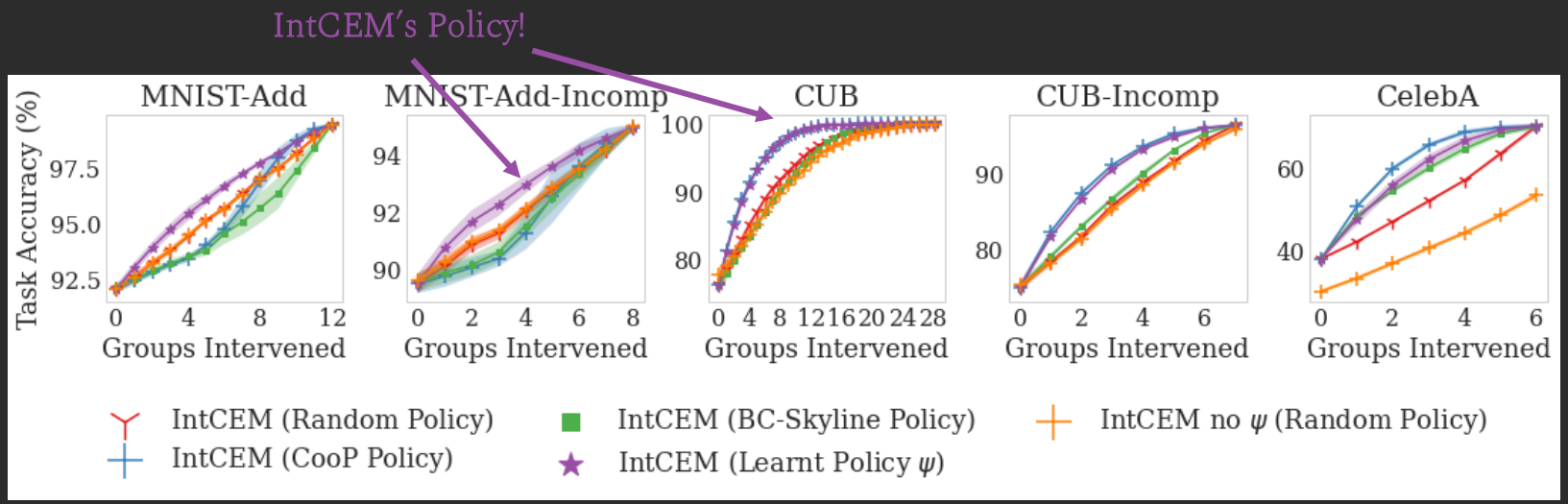


[1] Espinosa Zarlenga et al. "Learning to receive help: Intervention-aware concept embedding models." NeurIPS (2023)

Intervention-aware interpretable models

Intervention-aware models are:

1. Generally better at receiving any interventions
2. Capable of learning an efficient and effective intervention policy



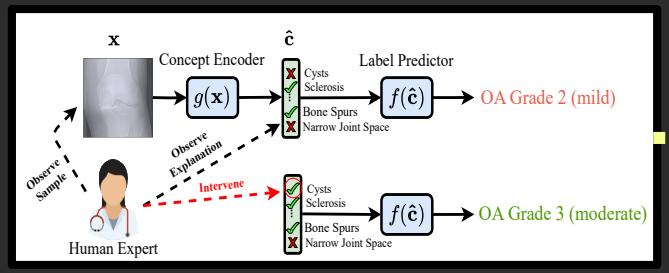
[1] Espinosa Zarlenga et al. "Learning to receive help: Intervention-aware concept embedding models." NeurIPS (2023)

Recent directions in concept interventions

When intervening, we assume that concept interventions are:

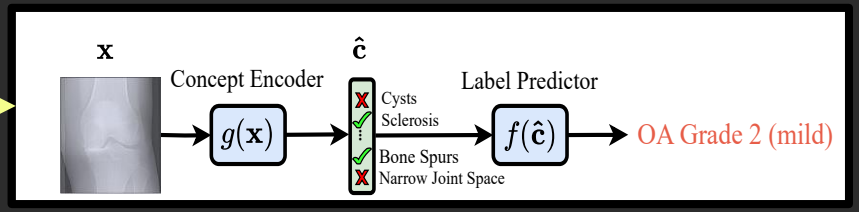
1. Transient: after an intervention is made, it is forgotten

If an intervention is made on a sample x



Some time later...

The same mistake if x is seen again

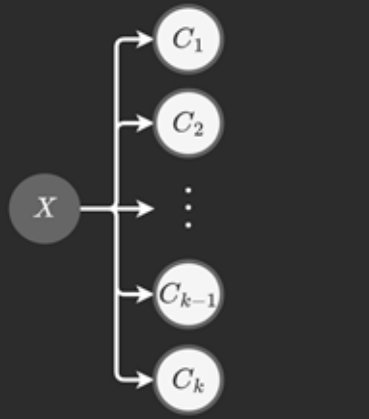


[1] Steinmann et al. "Learning to Intervene on Concept Bottlenecks." ICML (2024).

Recent directions in concept interventions

When intervening, we assume that concept interventions are:

1. Transient: after an intervention is made, it is forgotten
2. Independent: intervening on concept c_i will not affect other concepts' values



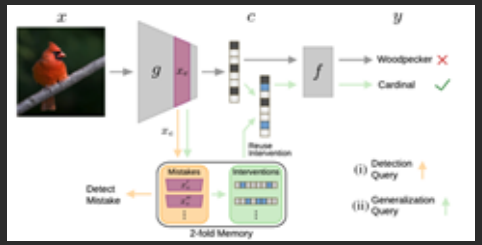
$$P(C \mid X) = \Pi_{i=1}^{k} P(C_i \mid X)$$

[1] Vandenhirtz, Laguna et al. "Stochastic Concept Bottleneck Models." NeurIPS (2024).
[2] Havasi et al. "Addressing leakage in concept bottleneck models." NeurIPS (2022).

Recent directions in concept interventions

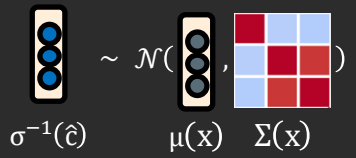
These constraints can be relaxed via clever modelling:

Concept Bottleneck Memory Models



Addresses: Transient nature of a concept intervention
Approach: Introduce a learnable memory module that keeps previously seen interventions and re-applies them

Stochastic Concept Bottleneck Models



Addresses: the assumption that concepts are independent
Approach: Model the predicted concept logits as a normal distribution with a (learnable) non-diagonal covariance.

[1] Steinmann et al. "Learning to Intervene on Concept Bottlenecks." ICML (2024).
[2] Vandenhirtz, Laguna et al. "Stochastic Concept Bottleneck Models." NeurIPS (2024).

Tutorial Outline

I. Introduction

II. Interpretability Symmetries

III. Interpretable Models & Inferences

Q&A + 30 min break

IV. Designing Concept Representations

V. Designing Task Predictors

VI. Human-AI Interaction

VII. Conclusion

Final Q&A

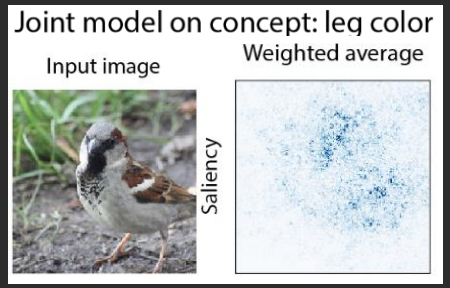
Challenges and open questions

There are still significant challenges and open questions in interpretable DL, particularly on making these models **more "real-world-friendly"**.

Open question: leakage

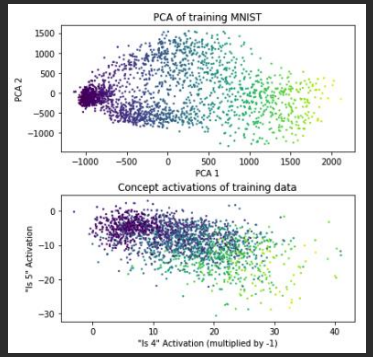
There is evidence that concept encoders are prone to encoding concept-irrelevant information in their representations (i.e., *information leakage*)

Attending spurious features



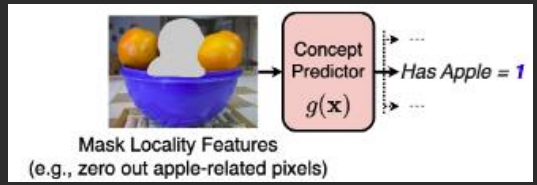
Saliency maps seem to suggest concepts are not properly attended (Margeloiu et al., 2021)

Leaking unwanted information



CBMs may have incentives to encode the entire data representation in the concepts' soft predictions (Mahinpei et al., 2021)

Failing to capture concept locality



CBMs may fail to capture a concept's locality (e.g., physical location) even if it is only found on a fixed feature subset (Raman et al., 2025)

[1] Margeloiu et al. "Do concept bottleneck models learn as intended?." ICLR Workshop on Responsible AI (2021).
[2] Mahinpei et al. "Promises and pitfalls of black-box concept learning models." ICML Workshop on Theoretic Foundation, Criticism, and Application of XAI (2021).
[3] Raman et al. "Do Concept Bottleneck Models Respect Localities?." TMLR (2025).

Open question: leakage

There is evidence that concept encoders are prone to encoding concept-irrelevant information in their representations (i.e., *information leakage*)

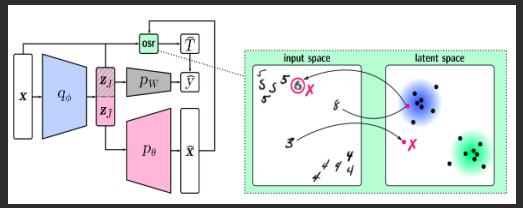
However, it is still unclear how leakage affects model interpretability and how to entirely mitigate it…

Open question: leakage

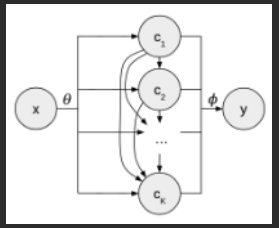
There is evidence that concept encoders are prone to encoding concept-irrelevant information in their representations (i.e., *information leakage*)

However, it is still unclear how leakage affects model interpretability and how to entirely mitigate it… although some research has made some progress here

GlanceNets



Autoregressive CBMs

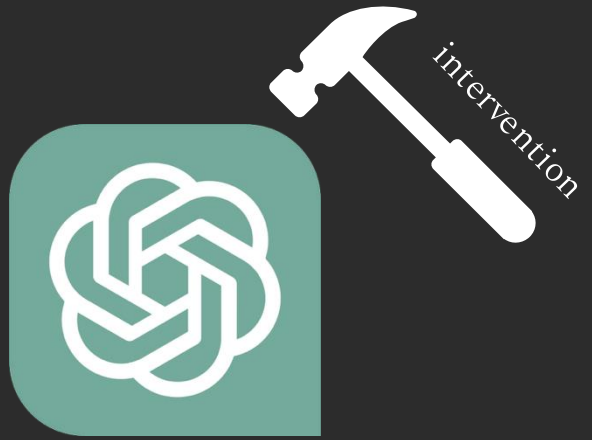


[1] Marconato et al. "Glancenets: Interpretable, leak-proof concept-based models." NeurIPS (2022).
[2] Havasi et al. "Addressing leakage in concept bottleneck models." NeurIPS (2022).

Open questions: label-free approaches

Label-free models open the door for **practical interpretable models**, yet they are currently not as reliable as supervised concept-based models

1. How can we effectively intervene in label-free settings?



Open questions: label-free approaches

Label-free models open the door for **practical interpretable models**, yet they are currently not as reliable as supervised concept-based models

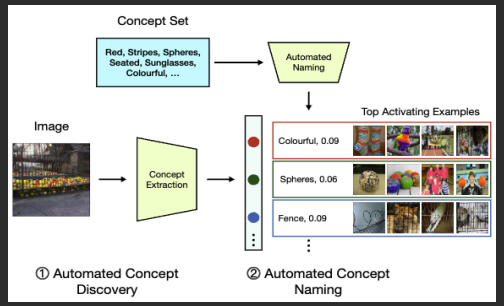
2. How can we learn label-free models without specialized foundation models?

Open questions: label-free approaches

Label-free models open the door for **practical interpretable models**, yet they are currently not as reliable as supervised concept-based models

2. How can we learn label-free models without specialized foundation models?

Discover-then-Name



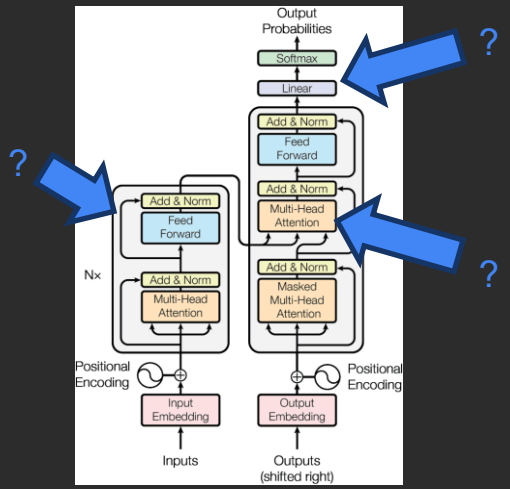
Discover *detectable* concepts using a SAE and then name them via an LLM
→ Could be extended to specialized domains where partial prototype naming is done by experts

[1] Rao et al. "Discover-then-name: Task-agnostic concept bottlenecks via automated concept discovery." ECCV (2024).

Open questions: scalability

Concept-based interpretable models have yet to be properly integrated to **large scale and multimodal models**. This raises a lot of important questions:

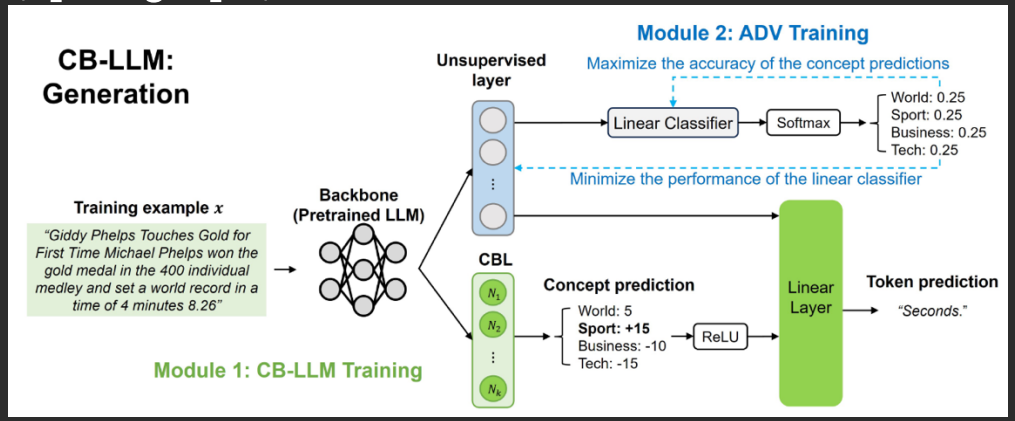
1. Where should we look for/place concepts in LLMs? Should this be done at a token, sentence, paragraph, or text-level?



Open questions: scalability

Concept-based interpretable models have yet to be properly integrated to **large scale and multimodal models**. This raises a lot of important questions:

1. Where should we look for/place concepts in LLMs? Should this be done at a token, sentence, paragraph, or text-level?



[1] Sun et al. "Concept bottleneck large language models." ICLR (2025).

Challenges and open questions: scalability

Concept-based interpretable models have yet to be properly integrated to **large scale and multimodal models**. This raises a lot of important questions:

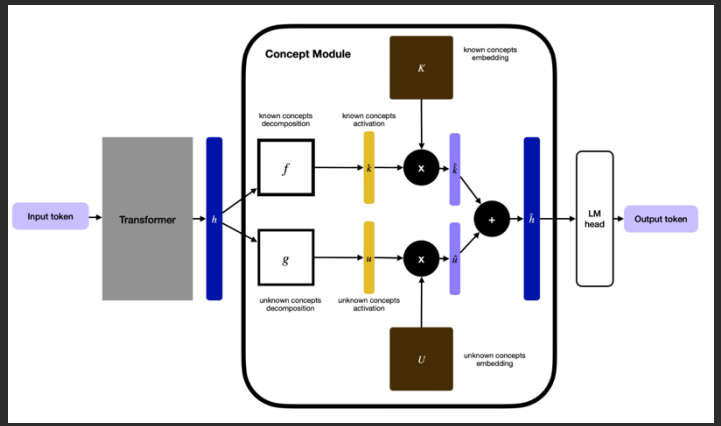
2. Can we even scale concept-based models to current large model standards?

Challenges and open questions: scalability

Concept-based interpretable models have yet to be properly integrated to **large scale and multimodal models**. This raises a lot of important questions:

2. Can we even scale concept-based models to current large model standards?

Interpretable LLMs (GuideLabs)



[1] Ismail et al. "Scaling Interpretable Language Models to 8 Billion Parameters" Accessed at: https://www.guidelabs.ai/post/scaling-interpretable-models-8b/

Challenges and open questions: scalability

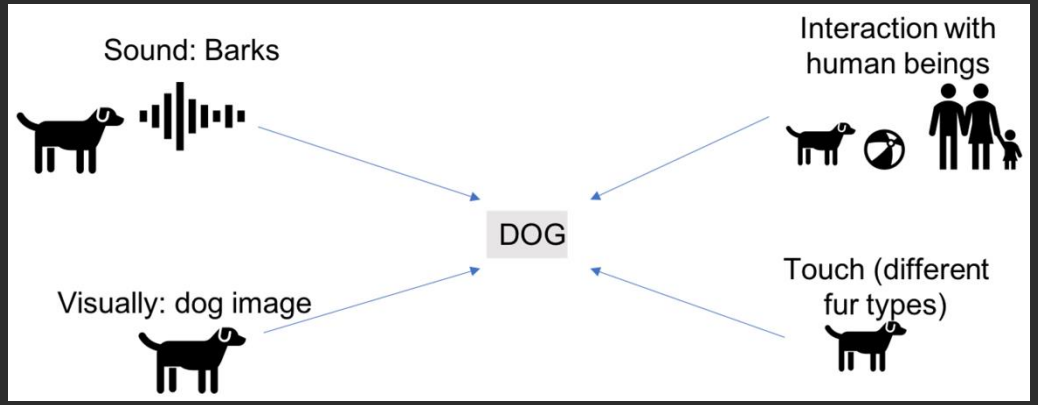
Concept-based interpretable models have yet to be properly integrated to **large scale and multimodal models**. This raises a lot of important questions:

3. What does a "concept" really mean in textual inputs? What about in other modalities such as genomics, finance, and law?

Challenges and open questions: scalability

Concept-based interpretable models have yet to be properly integrated to **large scale and multimodal models**. This raises a lot of important questions:

4. How do multi-modal concepts look like and how can we even intervene across multiple modalities?



Conclusion: Concepts and their role in interpretability

Today we:

1. Argued that interpretability can be thought as a form of inference equivariance

Conclusion: Concepts and their role in interpretability

Today we:

1. Argued that interpretability can be thought as a form of inference equivariance
2. Showed that equivariance is insufficient for interpretability unless we compress our features into a set of concepts and enforce a set of symmetries

Conclusion: Concepts and their role in interpretability

Today we:

1. Argued that interpretability can be thought as a form of inference equivariance
2. Showed that equivariance is insufficient for interpretability unless we compress our features into a set of concepts and enforce a set of symmetries
3. Proposed a blueprint for designing interpretable models formed by an exogenous encoder, a concept encoder, a concept predictor, and a label predictor

Conclusion: Concepts and their role in interpretability

Today we:

1. Argued that interpretability can be thought as a form of inference equivariance
2. Showed that equivariance is insufficient for interpretability unless we compress our features into a set of concepts and enforce a set of symmetries
3. Proposed a blueprint for designing interpretable models formed by an exogenous encoder, a concept encoder, a concept predictor, and a label predictor
4. Discussed several instantiations of this blueprint across the literature

Further resources: Website and library

1. Tutorial website (extended bibliography and slides):

interpretabledeeplearning.github.io

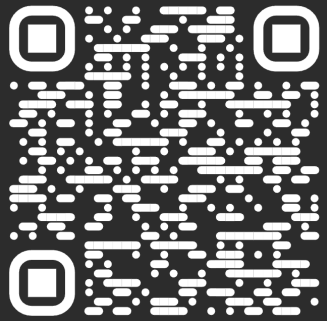
2. PyTorch Concepts (PyC), tutorials and library for concept-based interpretability:

pyc-team.github.io/pyc-book/intro.html

Thank you, grazie, gracias for your time!

Contact: pietro.barbiero@ibm.com and mateo.espinosazarlenga@trinity.ox.uk

Final Q&A



Appendix: A Deeper Dive of Alignment

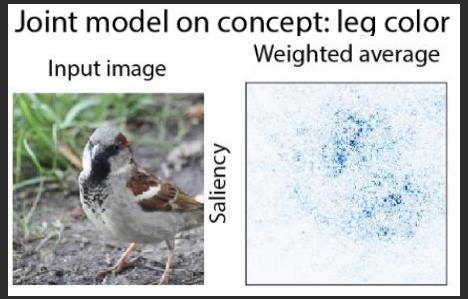
The difficulty of alignment: leakage

Cross-entropy-based concept alignment may lead to unwanted information leakage

The difficulty of alignment: leakage

Cross-entropy-based concept alignment may lead to unwanted information leakage

Attending spurious features



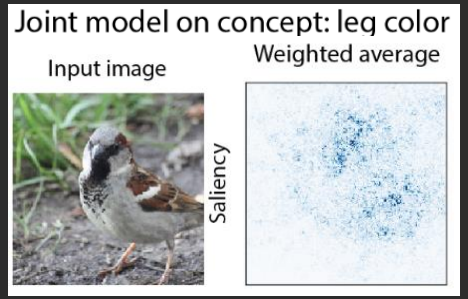
Saliency maps seem to suggest concepts are not properly attended (Margeloiu et al., 2021) [1]

[1] Margeloiu et al. "Do concept bottleneck models learn as intended?." ICLR Workshop on Responsible AI (2021).

The difficulty of alignment: leakage

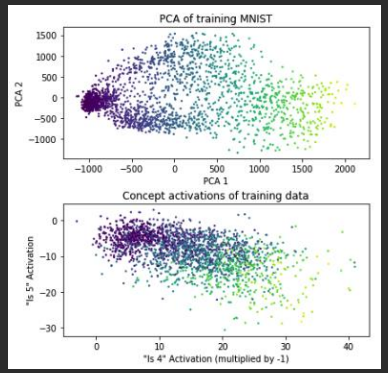
Cross-entropy-based concept alignment may lead to unwanted information leakage

Attending spurious features



Saliency maps seem to suggest concepts are not properly attended (Margeloiu et al., 2021) [1]

Leaking unwanted information



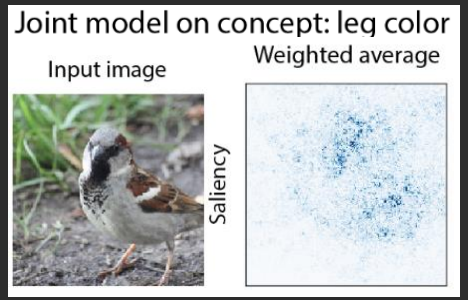
CBMs may have incentives to encode the entire data representation in the concepts' soft predictions (Mahinpei et al., 2021) [2]

[1] Margeloiu et al. "Do concept bottleneck models learn as intended?." ICLR Workshop on Responsible AI (2021).
[2] Mahinpei et al. "Promises and pitfalls of black-box concept learning models." ICML Workshop on Theoretic Foundation, Criticism, and Application of XAI (2021).

The difficulty of alignment: leakage

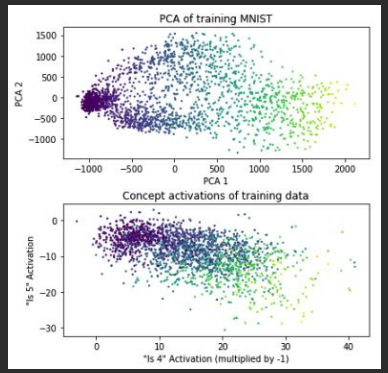
Cross-entropy-based concept alignment may lead to unwanted information leakage

Attending spurious features



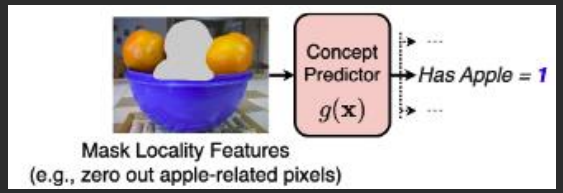
Saliency maps seem to suggest concepts are not properly attended (Margeloiu et al., 2021) [1]

Leaking unwanted information



CBMs may have incentives to encode the entire data representation in the concepts' soft predictions (Mahinpei et al., 2021) [2]

Failing to capture concept locality



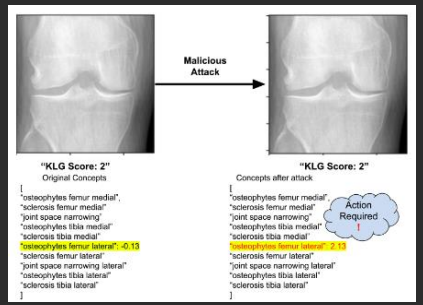
CBMs may fail to capture a concept's locality (e.g., physical location) even if it is only found on a fixed feature subset (Raman et al., 2025) [3]

[1] Margeloiu et al. "Do concept bottleneck models learn as intended?." ICLR Workshop on Responsible AI (2021).
[2] Mahinpei et al. "Promises and pitfalls of black-box concept learning models." ICML Workshop on Theoretic Foundation, Criticism, and Application of XAI (2021).
[3] Raman et al. "Do Concept Bottleneck Models Respect Localities?." TMLR (2025).

The difficulty of alignment: leakage

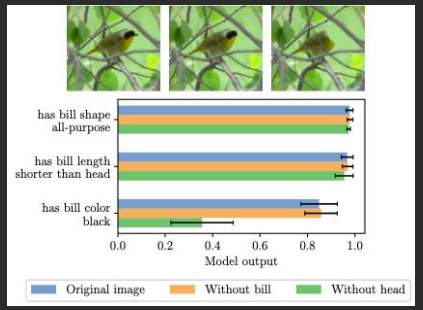
Many more works have dived deeper into these issues!

Adversarial attacks/defences



Concept predictions can be changed without affecting the final prediction (Sinha et al., 2023) [1]

Studying concept correlations



Simple changes to the loss, like loss weighting, can help avoid CBMs exploiting unwanted correlations (Heidenmann et al., 2023) [2]

Formalisms and metrics for leakage

Metrics for unwanted leakage [3]

Benchmark for reasoning robustness [4]

Formalisation of leakage [5]

Several works proposed ways to formalise or measure concept leakage [3, 4, 5]

[1] Sinha et al. "Understanding and enhancing robustness of concept-based models." AAAI (2023).
[2] Heidemann et al. "Concept correlation and its effects on concept-based models." WACV (2023).
[3] Espinosa Zarlenga, Barbiero, Shams et al. "Towards robust metrics for concept representation evaluation." AAAI (2023).
[4] Bortolotti, Marconato, et al. "A Neuro-Symbolic Benchmark Suite for Concept Quality and Reasoning Shortcuts." NeurIPS (2024).
[5] Marconato et al. "Interpretability is in the mind of the beholder: A causal framework for human-interpretable representation learning." Entropy (2023).

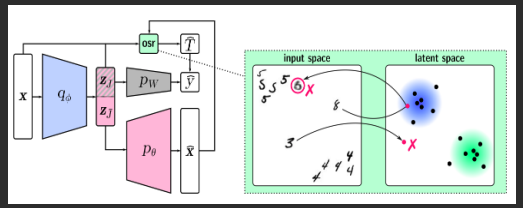
Rethinking traditional concept alignment

Several works have introduced different alignment mechanisms to mitigate leakage:

Rethinking traditional concept alignment

Several works have introduced different alignment mechanisms to mitigate leakage:

GlanceNets



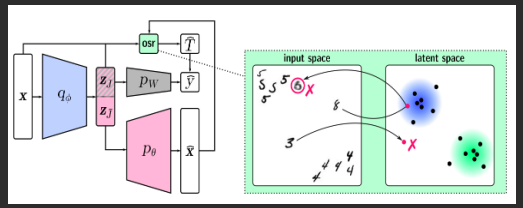
[Main Idea] Frame alignment as a disentanglement learning problem between known concepts and "residual" concepts

[1] Marconato et al. "Glancenets: Interpretable, leak-proof concept-based models." NeurIPS (2022).

Rethinking traditional concept alignment

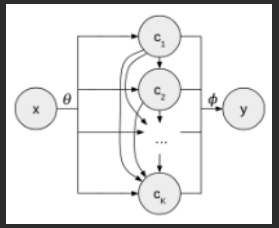
Several works have introduced different alignment mechanisms to mitigate leakage:

GlanceNets



[Main Idea] Frame alignment as a disentanglement learning problem between known concepts and "residual" concepts

Autoregressive CBMs



[Main Idea] Predict concepts in an auto-regressive fashion to explicitly capture cross-concept relationships

[1] Marconato et al. "Glancenets: Interpretable, leak-proof concept-based models." NeurIPS (2022).
[2] Havasi et al. "Addressing leakage in concept bottleneck models." NeurIPS (2022).