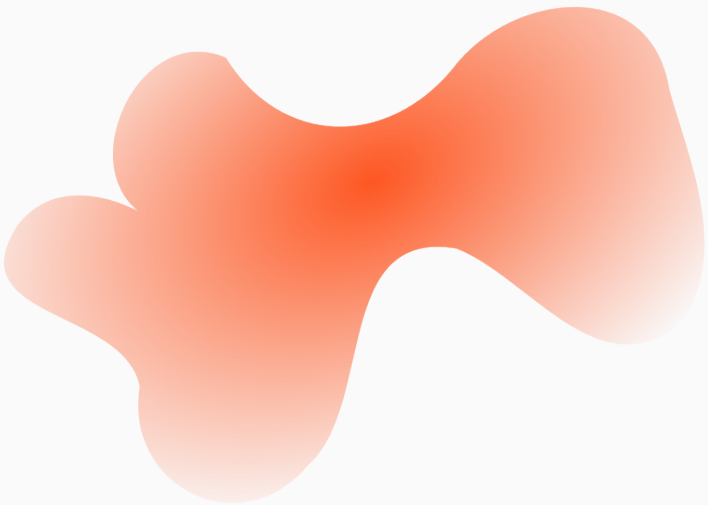
Probabilistic Circuits: Progress and Challenges

Robert Peharz
Graz University of Technology

Generative Models and Uncertainty Quantification
Copenhagen, 19th September 2024

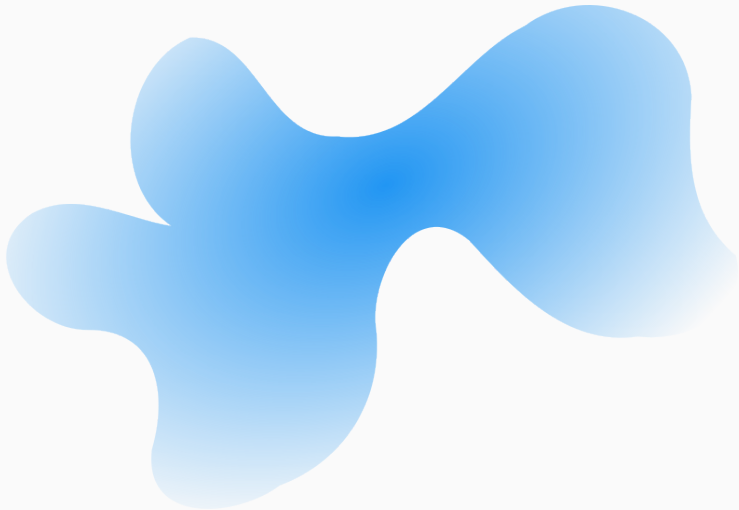
Probabilistic Modelling in a Nutshell

Model Distribution $\mathbb{P}_\theta$, p_θ



$\approx$

True Distribution $\mathbb{P}^*$, p^*

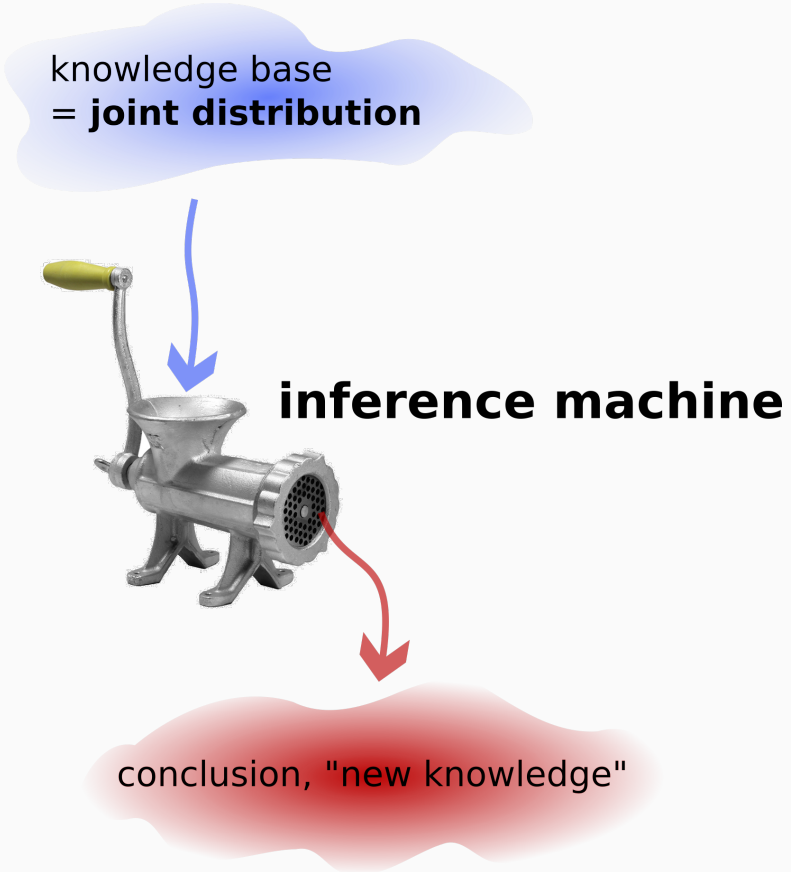


Why would we want to do that?

Generative Modelling

A cute kitty riding a unicycle while juggling with chainsaws

Reasoning under Uncertainty



What is probabilistic reasoning (inference machine)?

- **marginals** ("Ignore", "Account for Unknowns")

$$p(\boldsymbol{Y}) = \int_{\boldsymbol{Z}} p(\boldsymbol{Y}, \boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}$$

- **conditionals** ("Observe", "Inject Information")

$$p(\boldsymbol{Y} \mid \boldsymbol{z}) = \frac{p(\boldsymbol{Y}, \boldsymbol{z})}{p(\boldsymbol{z})} = \frac{p(\boldsymbol{Y}, \boldsymbol{z})}{\int_{\boldsymbol{Y}} p(\boldsymbol{y}, \boldsymbol{z}) \, \mathrm{d}\boldsymbol{y}}$$

- **sampling** $\boldsymbol{x} \sim p(\boldsymbol{X})$
- **compute density** $p(\boldsymbol{x})$
- **expectations** $\mathbb{E}_{\boldsymbol{X}}[f(\boldsymbol{x})]$
- **maximization** $\arg\max_{\boldsymbol{x}} p(\boldsymbol{x})$

The Problem

However, probabilistic inference is **hard** 

	GANs	VAEs	DBMs	Flows	ARMs
sampling	✓	✓	✓	✓	✓
density	✗	✗	✓	✓	✓
marginals	✗	✗	✗	✗	✗
condition	✗	✗	✗	✗	✗
moments	✗	✗	✗	✗	✗
max (MAP)	✗	✗	✗	✗	✗
$\mathbb{E}$	✗	✗	✗	✗	✗

The Problem

However, probabilistic inference is **hard**  . . . except for PCs!

	GANs	VAEs	DBMs	Flows	ARMs	PCs
sampling	✓	✓	✓	✓	✓	✓
density	✗	✗	✓	✓	✓	✓
marginals	✗	✗	✗	✗	✗	✓
condition	✗	✗	✗	✗	✗	✓
moments	✗	✗	✗	✗	✗	✓
max (MAP)	✗	✗	✗	✗	✗	✓ (✗)
$\mathbb{E}$	✗	✗	✗	✗	✗	✓ (✗)

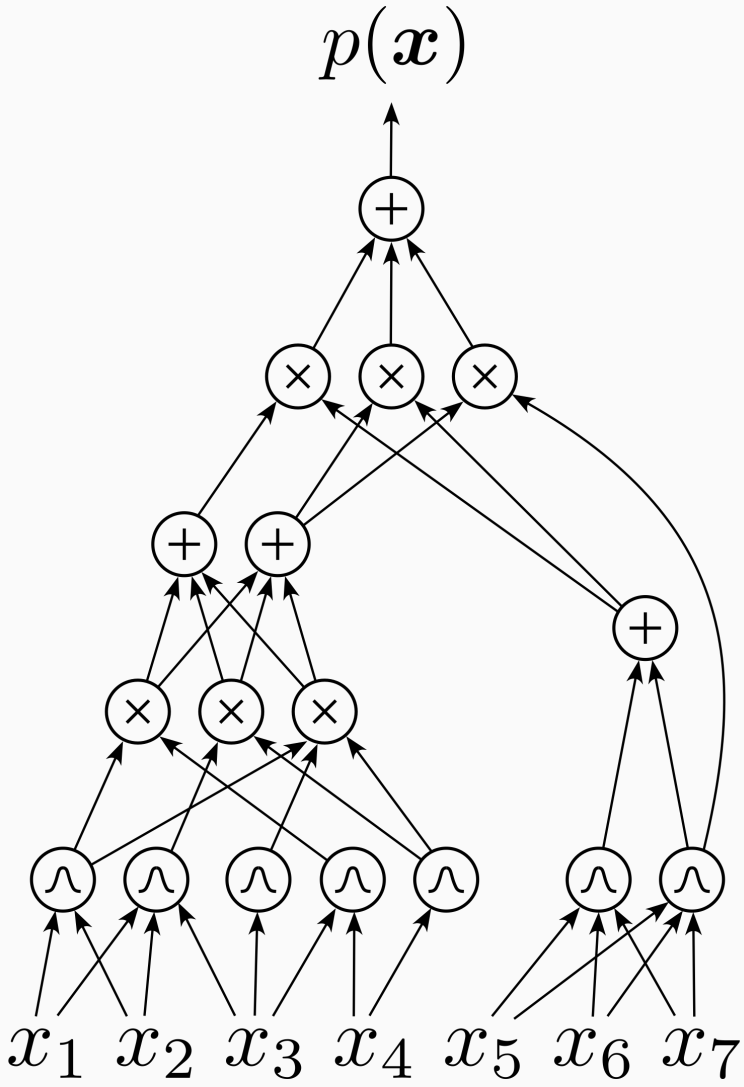
Probabilistic Circuits

A **probabilistic circuit (PC)** is a computational graph (neural network) containing three types of nodes:

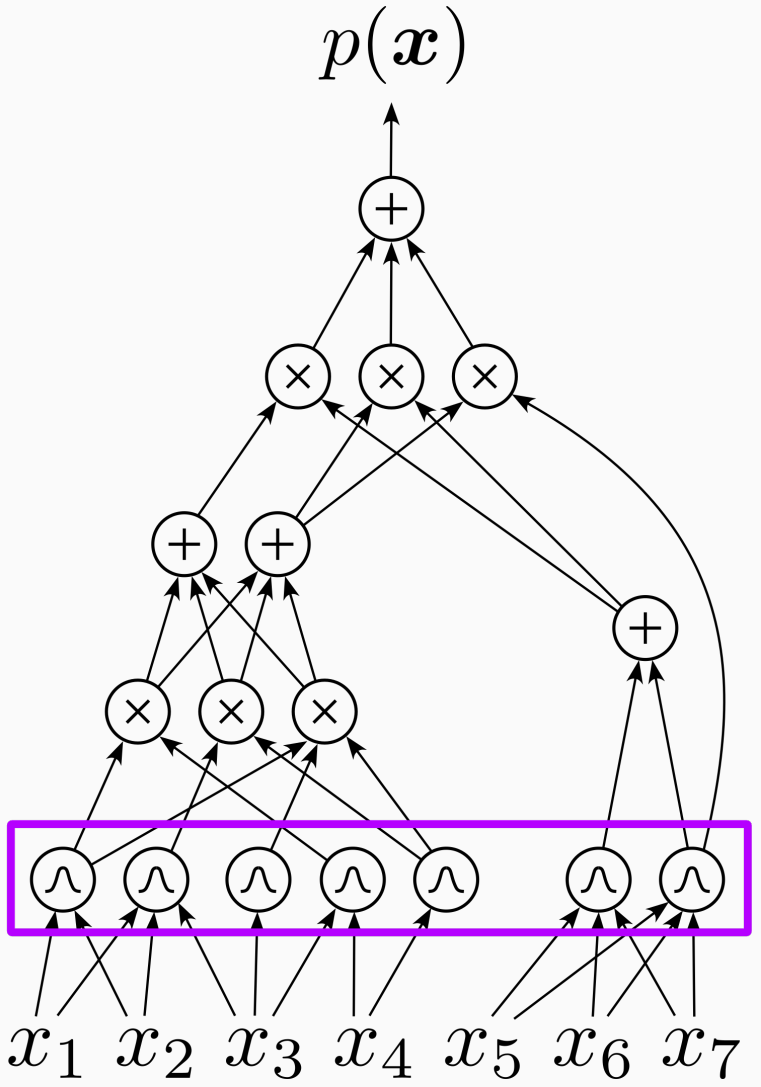
- **distributions**
- **products**
- **sums**

Input: values $\boldsymbol{x}$ for RVs $\boldsymbol{X}$ we want to model

Output: joint density $p(\boldsymbol{x})$ evaluated at $\boldsymbol{x}$ (perhaps unnormalized)



- nodes on the first layer are distribution nodes 
- they compute **some** probability density (or mass function) over their **scope** (=variables they get input from)
- every distribution node has its **own learnable parameters**
- hence two distribution nodes over the same scope will represent **different** distributions.



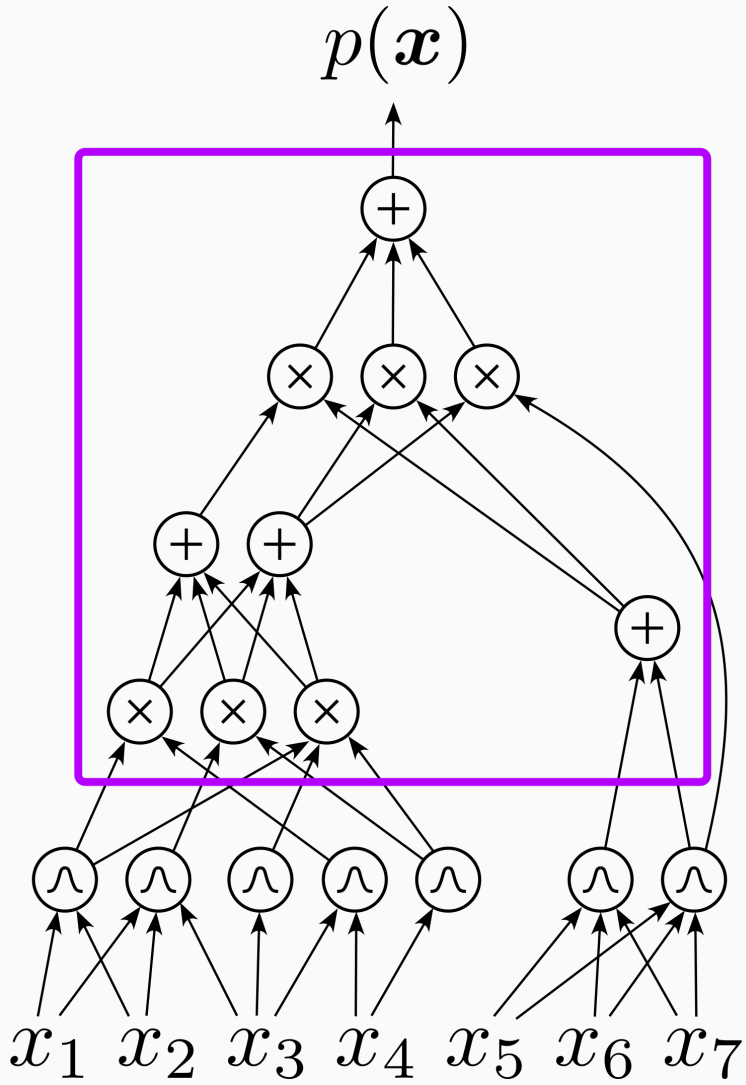
- all nodes on higher layers are sums or products
- a sum node $\oplus$ computes a weighted sum of its inputs:

$$\oplus := \sum_{k=1}^{K} w_k \, N_k$$

where $w_k \geq 0$ and $\sum_k w_k = 1$ are learnable parameters

- a product node $\otimes$ computes the product of its inputs

$$\otimes := \prod_{k=1}^{K} N_k$$



Structural Constraints

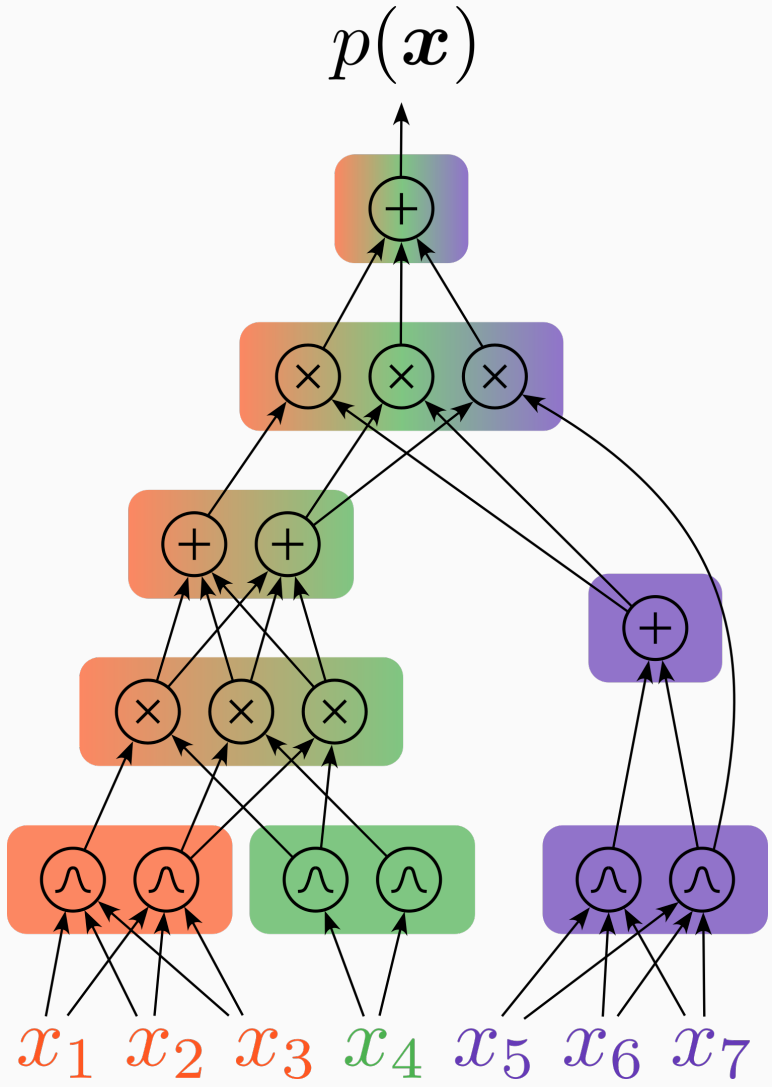
Every node in a PC has a **scope**, which is the set of variables it depends on (color-coded on the right).

Smoothness

A **sum node** is **smooth**, if all its inputs have the **same scope**. A PC is smooth, if all sum nodes in it are smooth.

Decomposability

A **product node** is **decomposable**, if all its inputs have **disjoint scope**. A PC is decomposable, if all product nodes in it are decomposable.



What do these constraints mean?

A **product node** P computes

$$P := \prod_{k=1}^{K} N_k$$

When the N_k's are distributions and the product is **decomposable**, this just means that the product node is a

factorized distribution

which represents (context-specific) **independence** between the N_k's.

A **sum node** S computes

$$S := \sum_{k=1}^{K} w_k \, N_k$$

where $w_k \geq 0$, $\sum_k w_k = 1$. When the N_k's are distributions and the sum is **smooth**, this just means that the sum node is a

mixture distribution

of the distributions represented by the N_k's.

Tractable Inference

Smoothness and decomposability lead to a natural interpretation of PCs as **hierarchical mixture models**.

Crucially, however, they unlock tractable inference in PCs, in particular **marginalization** and **conditioning**.

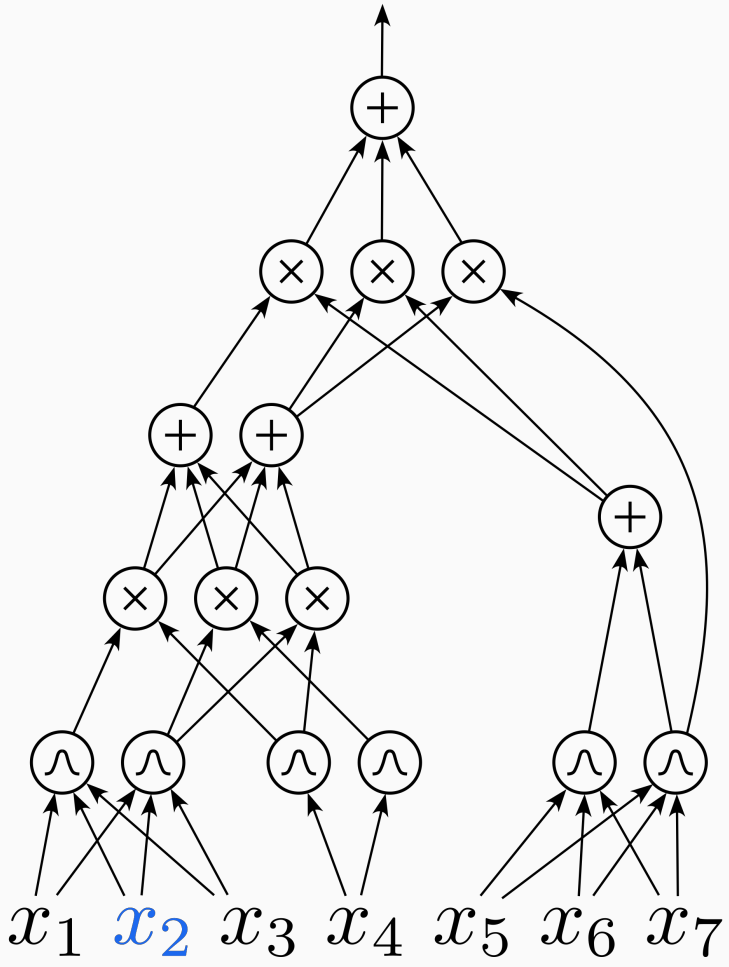
Key requirement: all input distributions allow marginalization and conditioning (e.g. Gaussians). A smooth and decomposable PC essentially inherits this property from its input distributions.

Marginal of PC

Example

- say we want to marginalize X_2

$$\int p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)\, \mathrm{d}x_2$$

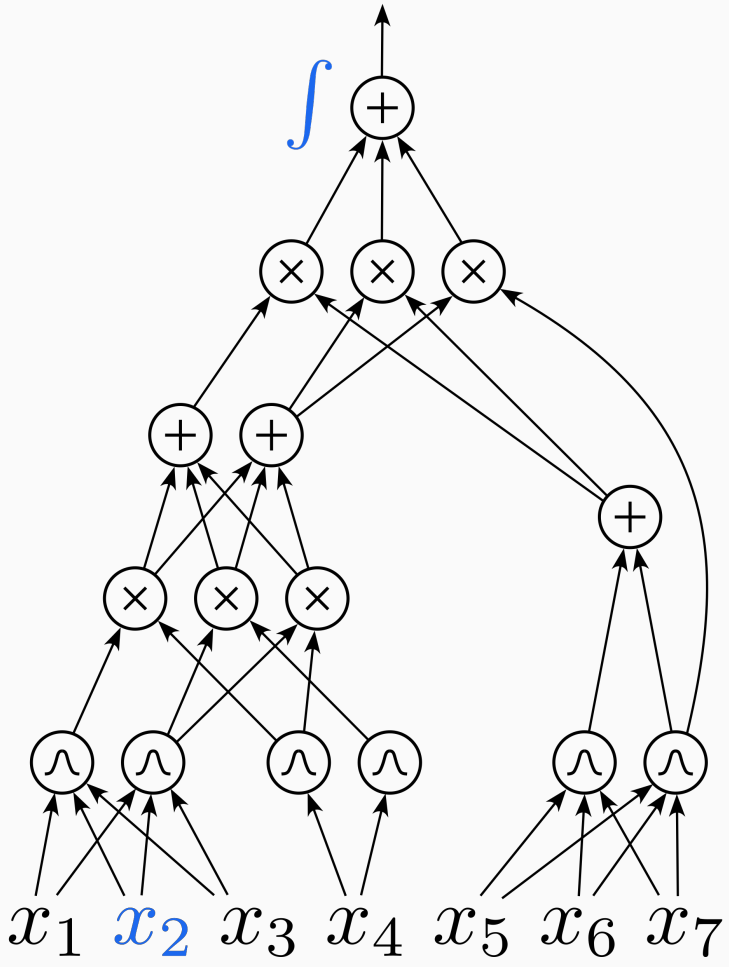


Marginal of PC

Example

- say we want to marginalize X_2
- the integral over x_2 is applied to the output node

$$\int p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)\, \mathrm{d}x_2$$



Marginal of PC

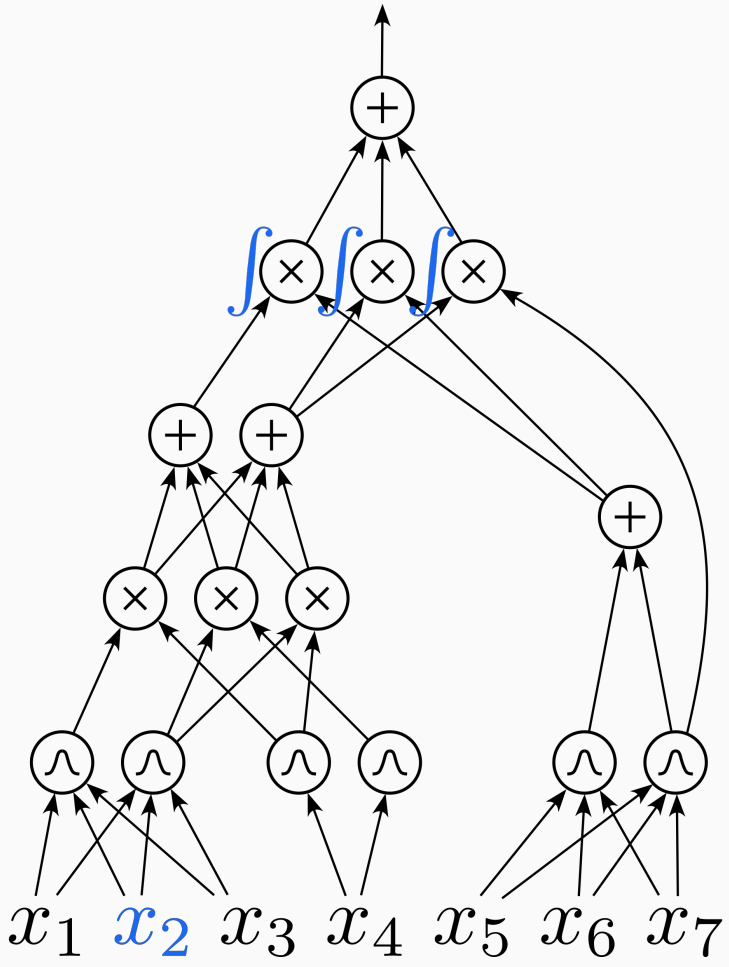
Example

- say we want to marginalize X_2
- the integral over x_2 is applied to the output node
- a sum node! Integrals and sums commute

$$\int \sum_k \dots \mathrm{d}x_2 = \sum_k \int \dots \mathrm{d}x_2$$

- hence we can push the integral to the inputs of the sum node

$$\int p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)\, \mathrm{d}x_2$$



Marginal of PC

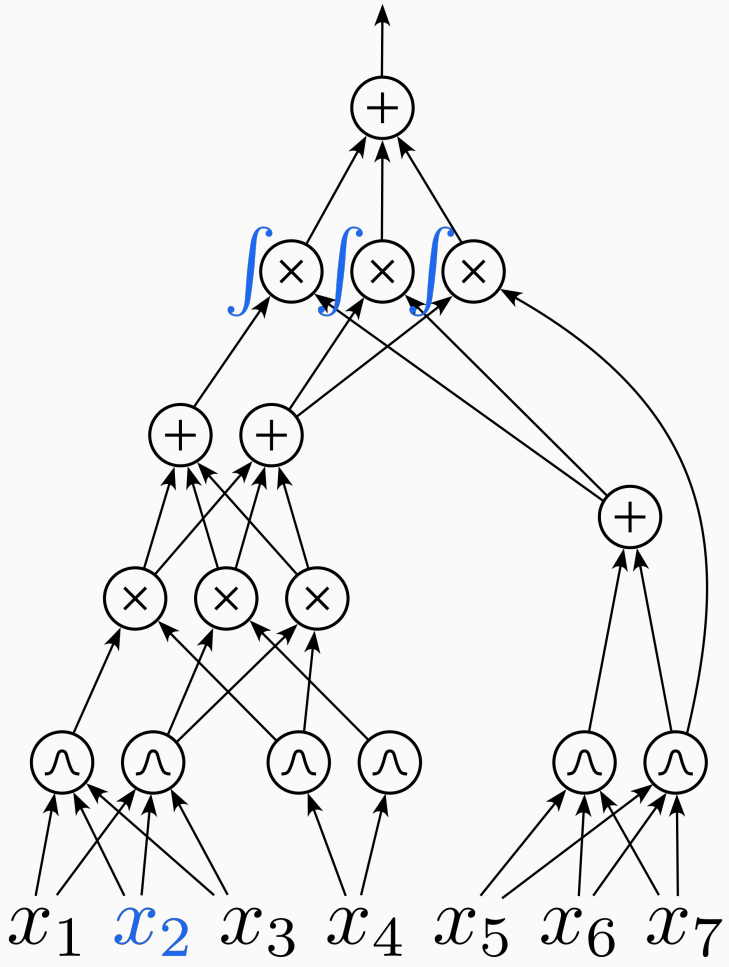
Example

- say we want to marginalize X_2
- the integral over x_2 is applied to the output node
- a sum node! Integrals and sums commute

$$\int \sum_k \dots \mathrm{d}x_2 = \sum_k \int \dots \mathrm{d}x_2$$

- hence we can push the integral to the inputs of the sum node
- now the problem is to compute the integrals of the nodes below (all products in this case)

$$\int p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)\, \mathrm{d}x_2$$



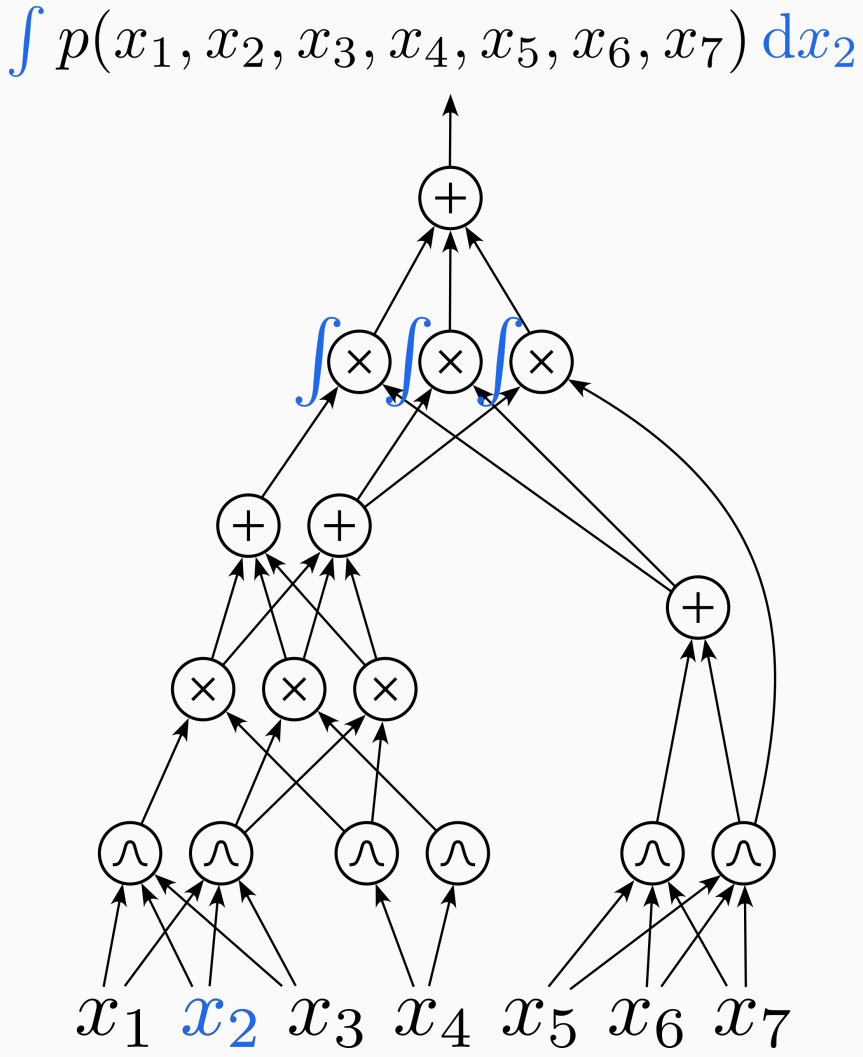
Marginal of PC

Example

- we assume **decomposability**, which means X_2 only appears in one input of each product!
- the other inputs are just a multiplicative constant for the integral over x_2:

$$\int c\, f(x_2, \dots)\, \mathrm{d}x_2 =$$

$$c \int f(x_2, \dots)\, \mathrm{d}x_2$$



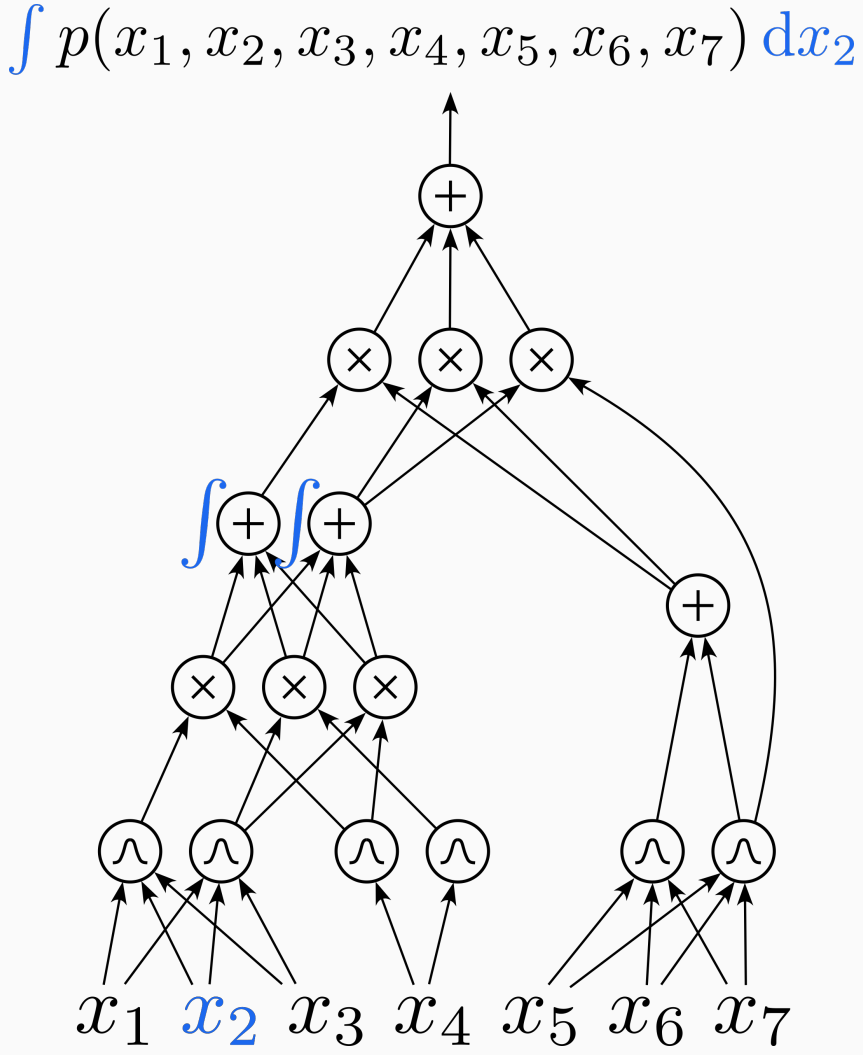
Marginal of PC

Example

- we assume **decomposability**, which means X_2 only appears in one input of each product!
- the other inputs are just a multiplicative constant for the integral over x_2:

$$\int c\, f(x_2, \dots)\, \mathrm{d}x_2 = c \int f(x_2, \dots)\, \mathrm{d}x_2$$

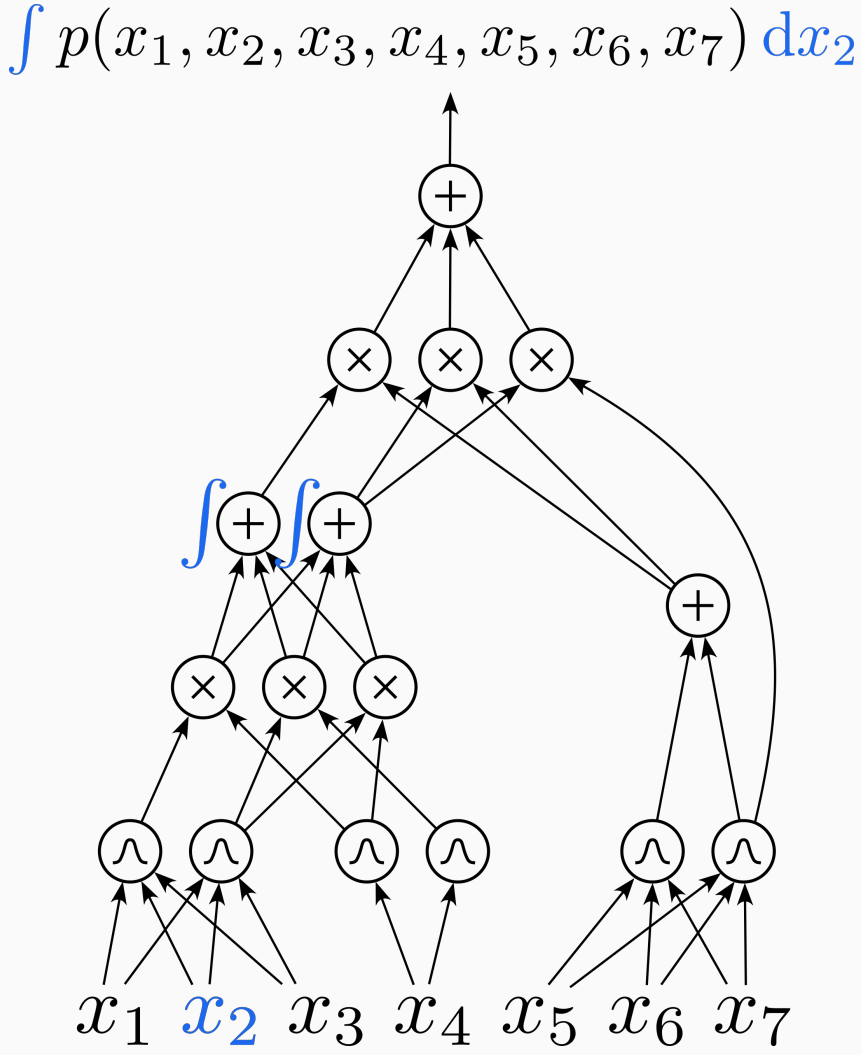
- hence, we can push the integral over each product, to the respective node containing X_2



Marginal of PC

Example

- these are sum nodes again

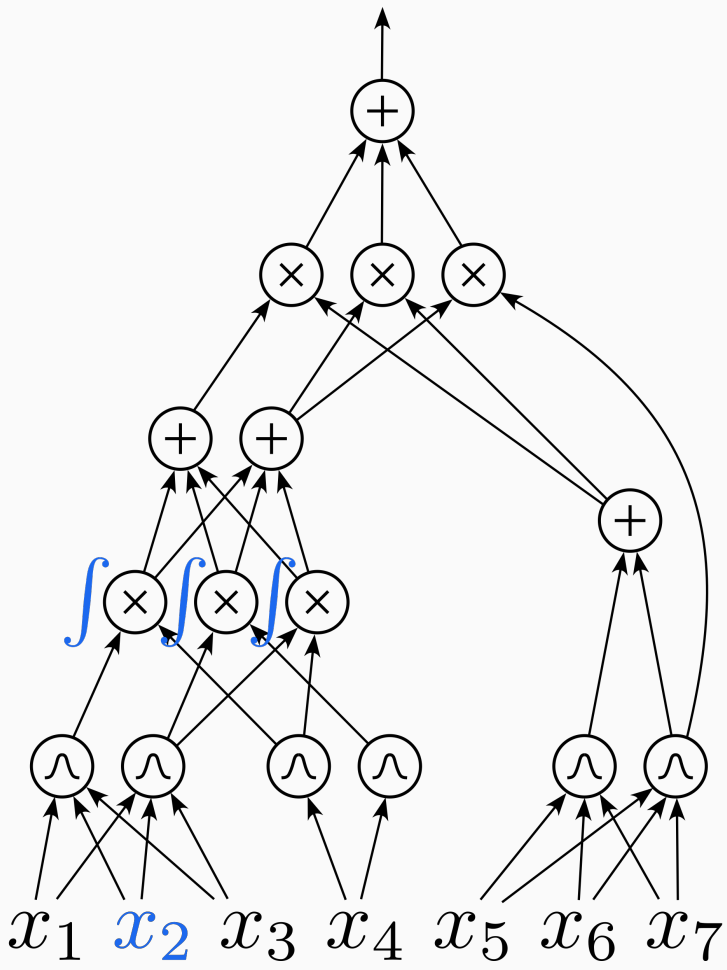


Marginal of PC

Example

- these are sum nodes again
- so we can push them down again

$$\int p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)\, \mathrm{d}x_2$$

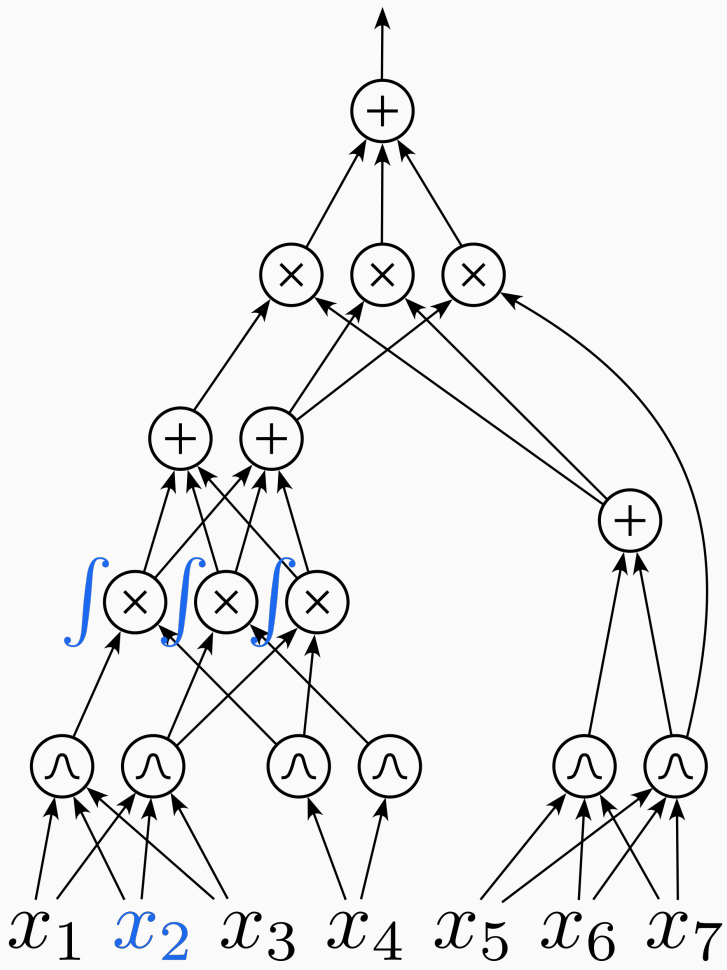


Marginal of PC

Example

- decomposable products again

$$\int p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)\, \mathrm{d}x_2$$

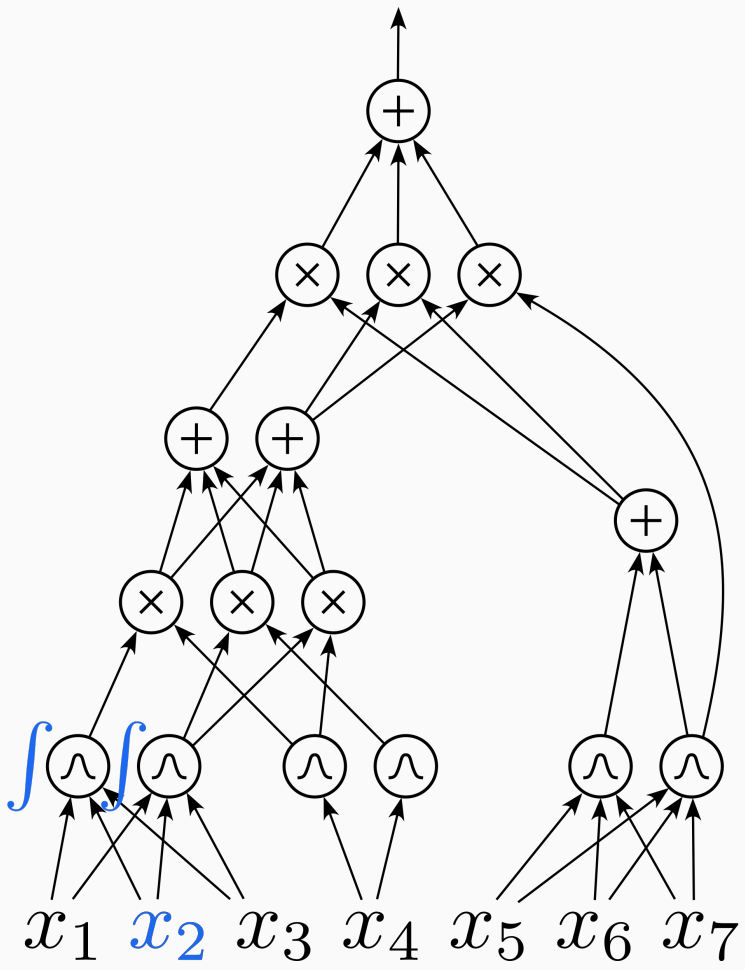


Marginal of PC

Example

- decomposable products again
- so we can push them down again

$$\int p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)\, \mathrm{d}x_2$$

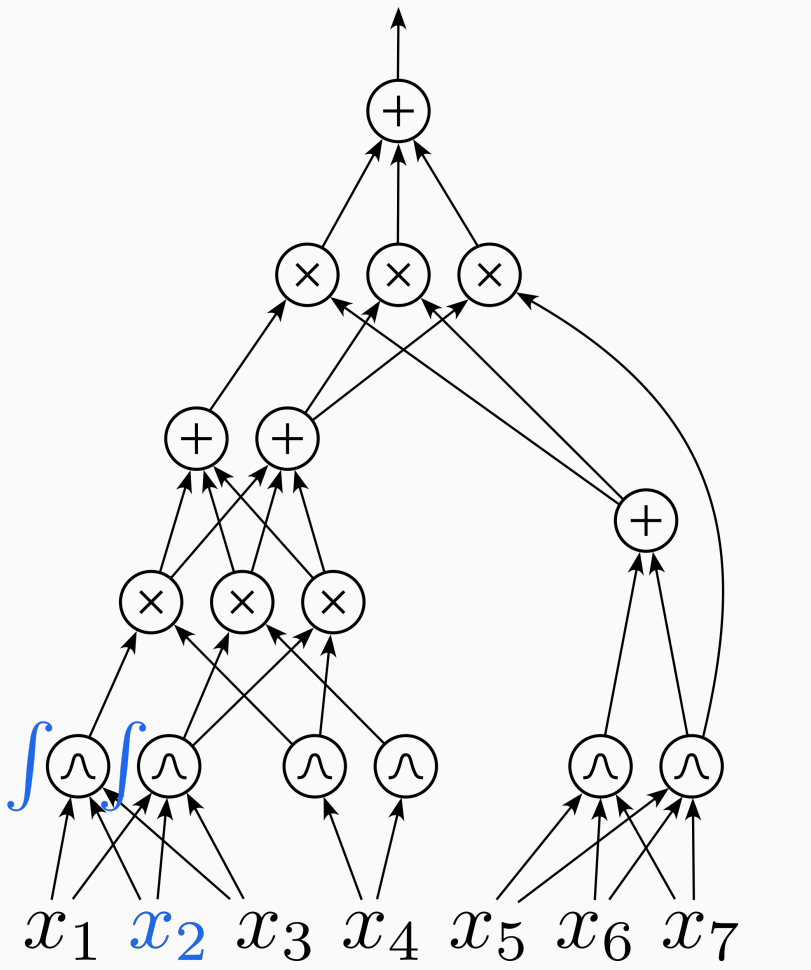


Marginal of PC

Example

- now, the integrals are at input distributions
- but we assumed that they allow tractable marginals!

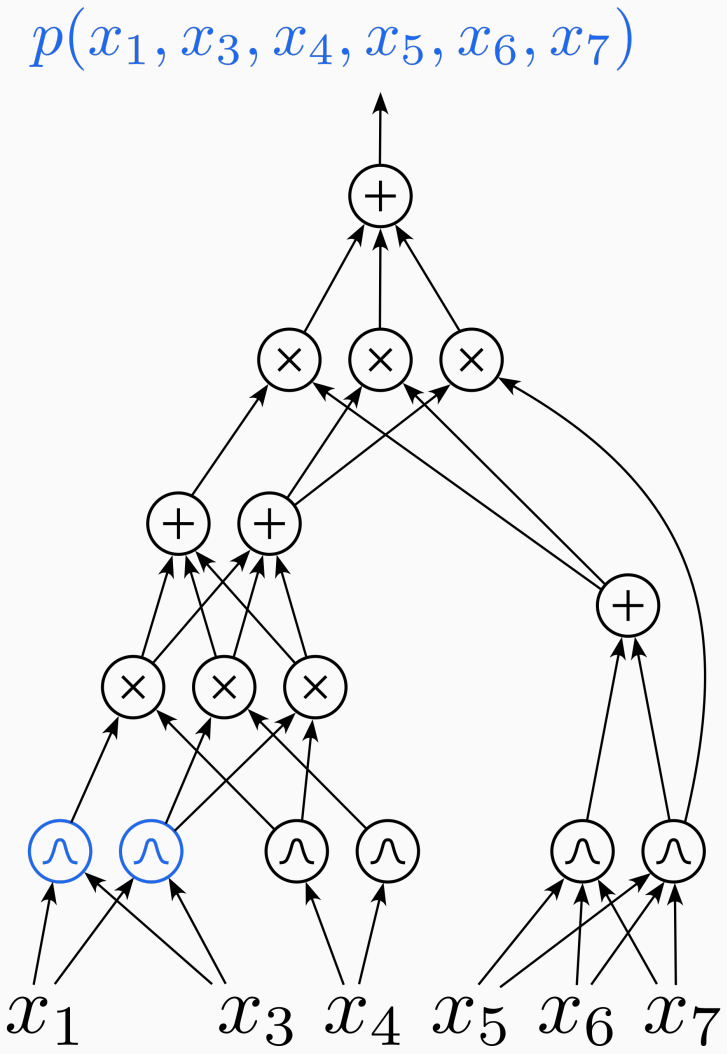
$$\int p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)\, \mathrm{d}x_2$$



Marginal of PC

Example

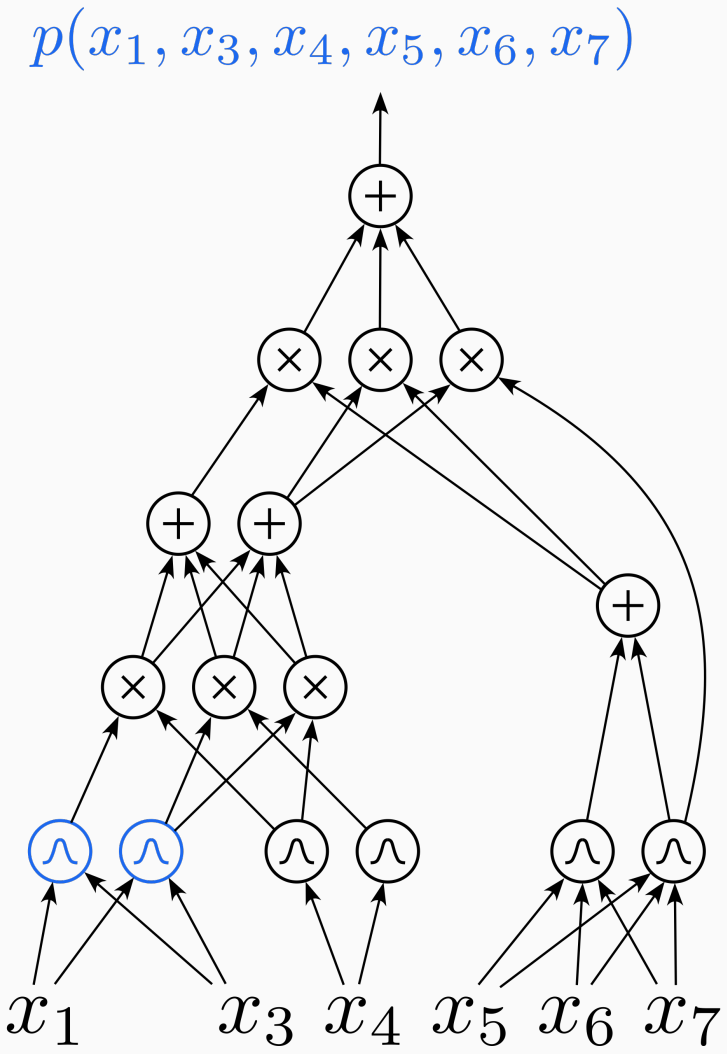
- now, the integrals are at input distributions
- but we assumed that they allow tractable marginals!
- thus, just replace the input distributions containing X_2 with the marginals where X_2 is marginalized out
- this yields a new PC, representing the desired marginal $p(X_1, X_3, X_4, X_5, X_6, X_7)$



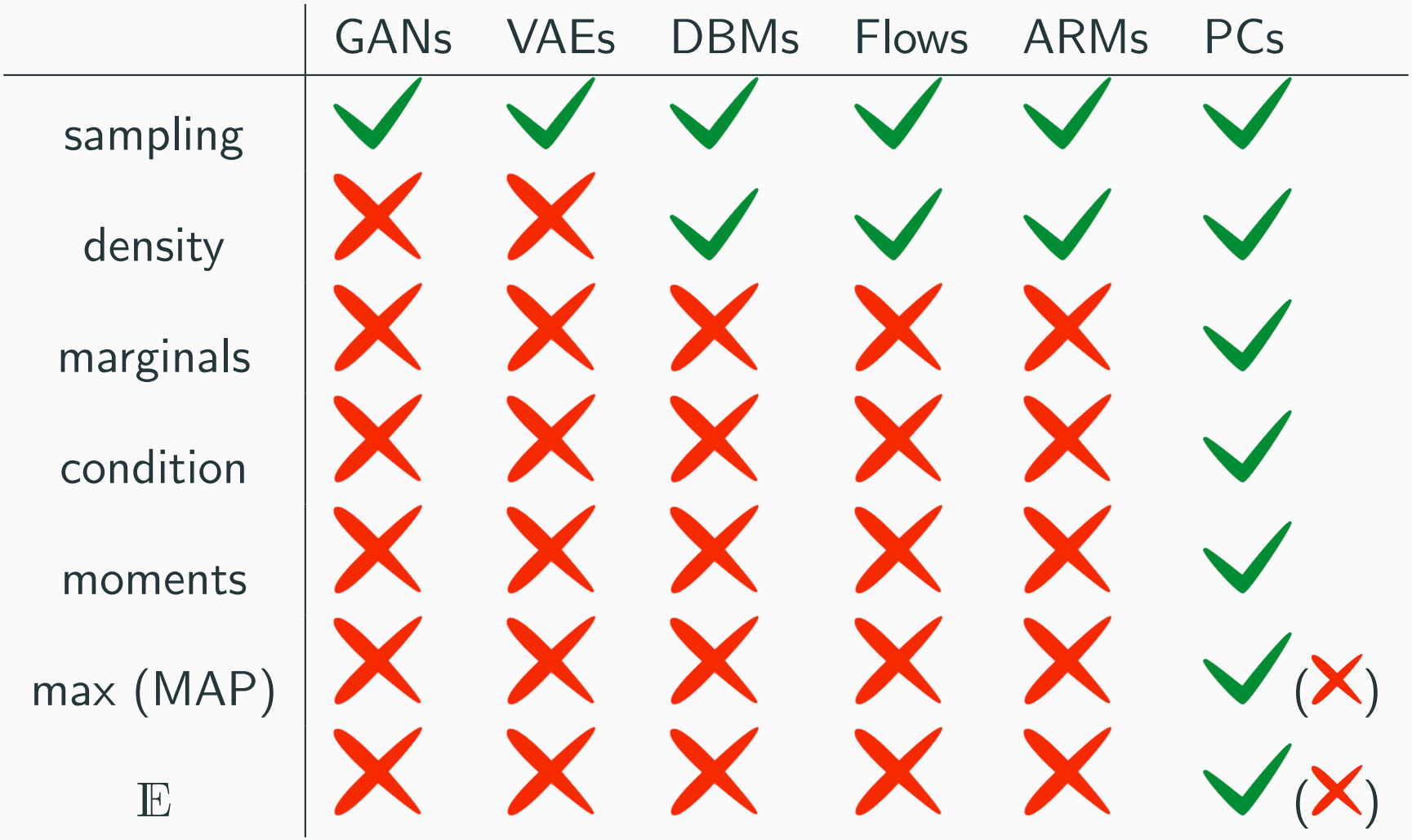
Marginal of PC

Example

We showed marginalization for just one variable, but by repeating the argument, we can marginalize arbitrarily many RVs.



Inference in Probabilistic Circuits



Progress and Challenges

Progress: PCs at Speed and Scale

$100 - 200X$ slower than MLPs

Random Sum-Product Networks:
A Simple and Effective Approach to Probabilistic Deep Learning

---

$50X$ slower than MLPs

Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits

---

$5X$ slower than MLPs

Scaling Tractable Probabilistic Circuits: A Systems Perspective

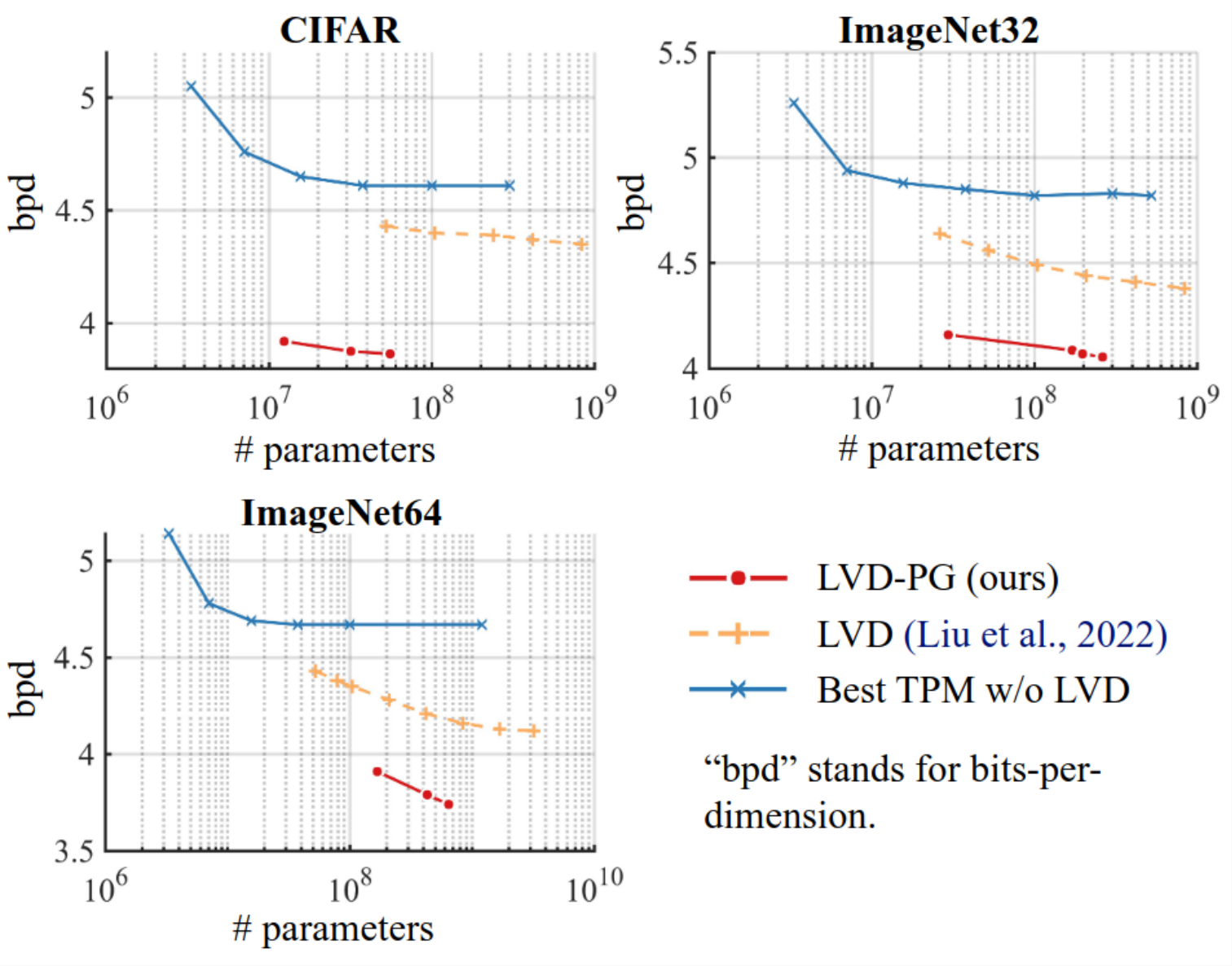
---

en par or faster than MLPs

Challenge: Expressiveness-vs-Tractability

- constraints are the price to pay for tractability
- constraints, however, limit the expressive power of an architecture (example: deep vs. shallow networks)
- hence, there is a **expressiveness-vs-tractability trade-off**
- for some classes of PCs, this trade-off is theoretically well understood (exponential separation)
- But, we can scale! This will surely save us, right?

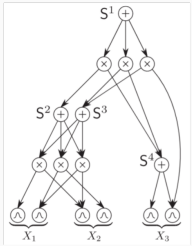
Challenge: Scale is not all we need

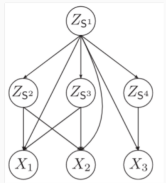


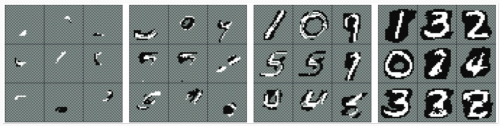
Liu et al., 2023

Progress: The many connections of PCs

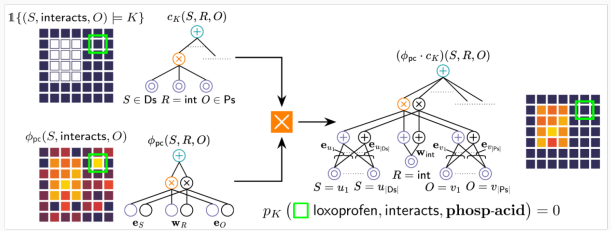
PCs as Latent Variable Models



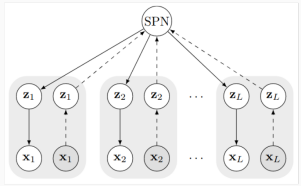


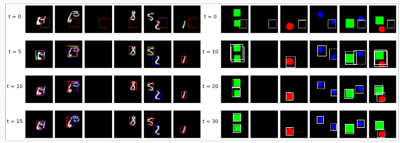


PCs x Knowledge Graphs

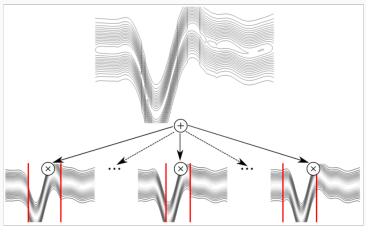


PCs x VAEs

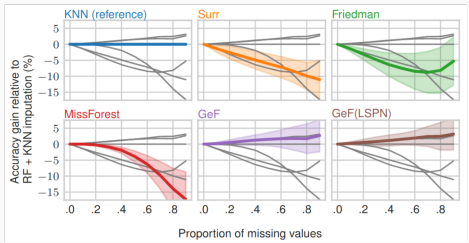




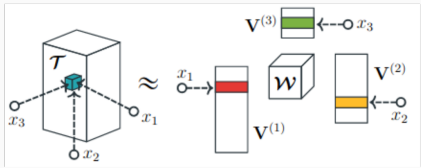
PCs x Gaussian Processes

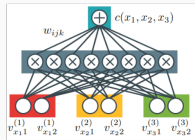


PCs x Decision Trees



PCs x Tensor Decompositions

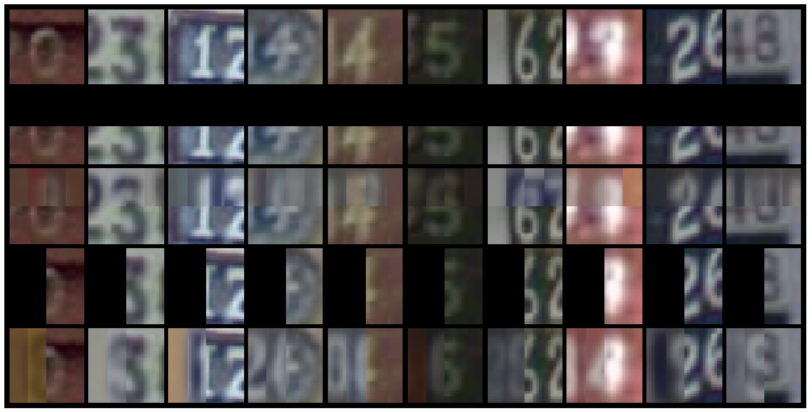




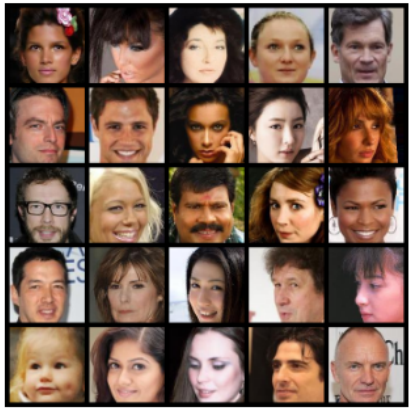
Challenge: Sample Quality (But, Exact Inpainting)

(a) Real SVHN images.

(b) EiNet SVHN samples.



(c) Real images (top), covered images, and EiNet reconstructions



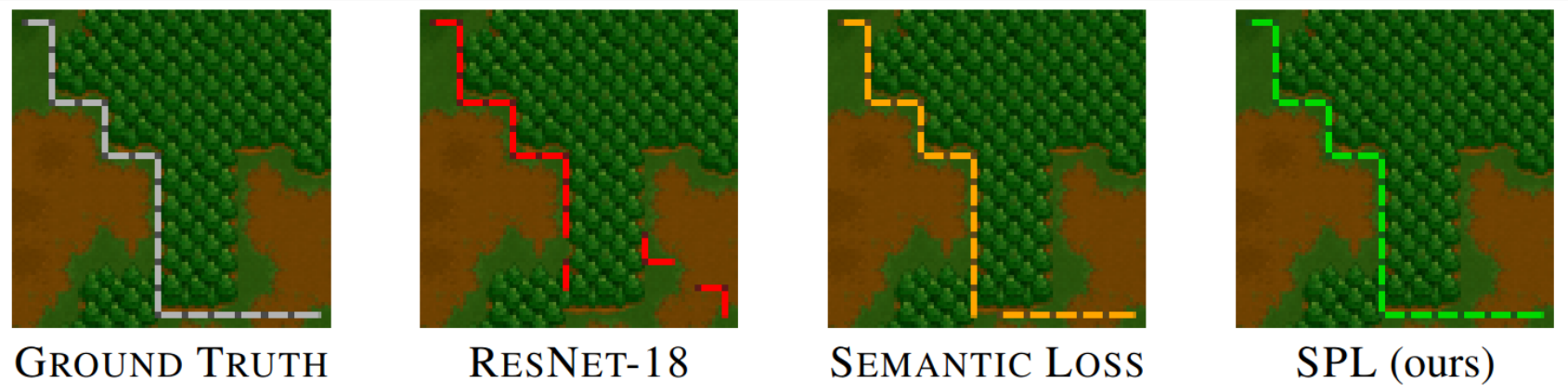
(d) Real CelebA samples.

(e) EiNet CelebA samples.

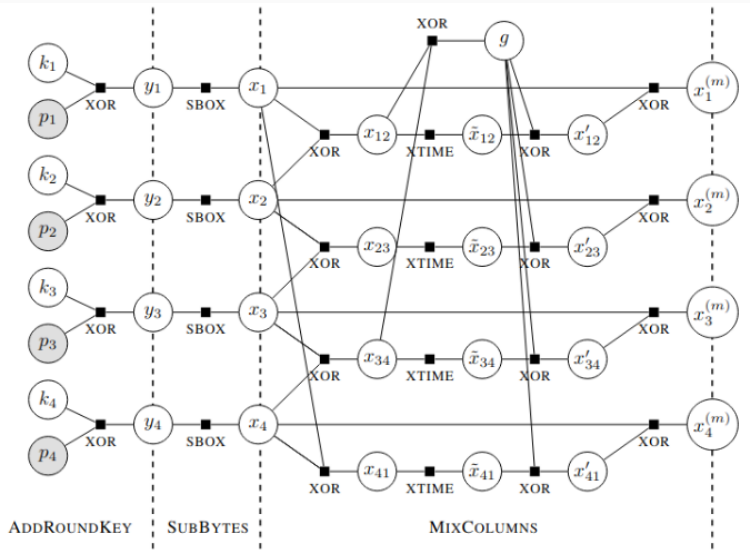
(f) Real images (top), covered images, and EiNet reconstructions

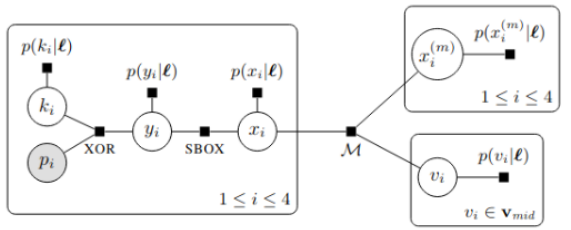
Peharz et al., 2020

Progress: Neurosymbolic ML



Ahmed et al., 2022





Inference Method	Top-1 Success Rate		
	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-8}$	$\varepsilon = 0$
Baseline	0.10%	0.10%	0.10%
SASCA (3 iters)	0.52%	0.52%	0.52%
SASCA (50 iters)	32.34%	33.81%	33.76%
SASCA (100 iters)	34.25%	36.04%	35.84%
ExSASCA + MAR	57.36%	67.37%	67.37%
ExSASCA + MPE	**57.51%**	**67.60%**	**67.61%**

Wedenig et al., 2024

References

Peharz et al., On the latent variable interpretation in sum-product networks, TPAMI 2017.
Peharz et al., Einsum networks: Fast and scalable learning of tractable probabilistic circuits, ICML 2020.
Vergari et al., Sum-product autoencoding: Encoding and decoding representations using sum-product networks, AAAI 2018.
Butz et al., Sum-product network decompilation, PGM 2020.
Loconte et al., How to turn your knowledge graph embeddings into generative models, NeurIPS 2023.
Stelzner et al., Faster attend-infer-repeat with tractable probabilistic models, ICML 2018.
Tan and Peharz, Hierarchical decompositional mixtures of variational autoencoders, ICML 2018.
Trapp et al., Deep structured mixtures of gaussian processes, AISTATS 2020.
Correia et al., Continuous mixtures of tractable probabilistic models, AAAI, 2023.
Gala et al., Probabilistic integral circuits, AISTATS 2024.
Correia et al., Joints in random forests, NeurIPS 2020.
Khosravi et al., Handling Missing Data in Decision Trees: A Probabilistic Approach, 2020
Ventola et al., Relational Decision Trees as Probabilistic Circuits, Springer 2021.
Loconte et al., What is the Relationship between Tensor Factorizations and Circuits (and How Can We Exploit it)?, 2024
Ahmed et al., Semantic Probabilistic Layers for Neuro-Symbolic Learning, NeurIPS 2022.
Wedenig et al., Exact Soft Analytical Side-Channel Attacks using Tractable Circuits, ICML 2024.
Liu et al., Understanding the distillation process from deep generative models to tractable probabilistic circuits, ICML 2023.
Liu et al., Scaling Tractable Probabilistic Circuits: A Systems Perspective, ICML 2024.