UCLA Computer Science

STAR AI RESEARCH LAB UCLA

Tractable Probabilistic Circuits

Guy Van den Broeck

appliedAI Seminar - Jan 18 2024

Outline

1. What are probabilistic circuits?

   tractable deep generative models

2. What are they useful for?

   controlling generative AI

3. What is the underlying theory?

   probability generating polynomials

Outline

1. **What are probabilistic circuits?**

   tractable deep generative models

2. What are they useful for?

   controlling generative AI

3. What is the underlying theory?

   probability generating polynomials

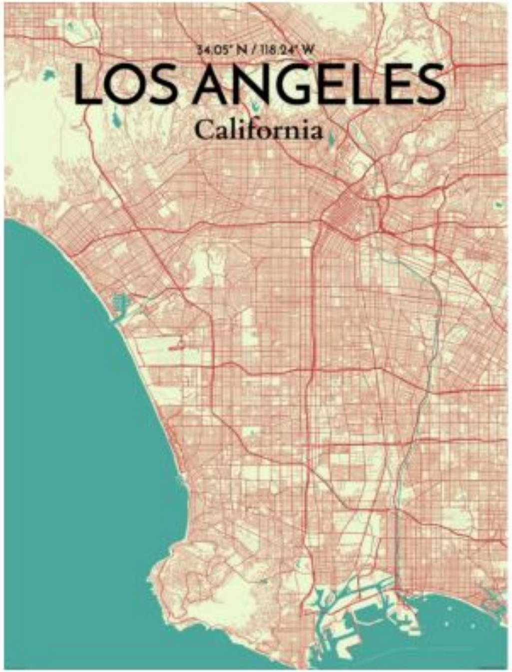
Why probabilistic inference?

$\mathbf{q}_1$: *What is the probability that today is a Monday and there is a traffic jam on Westwood Blvd.?*

$$\mathbf{X} = \{\mathsf{Day}, \mathsf{Time}, \mathsf{Jam}_{\mathsf{Str1}}, \mathsf{Jam}_{\mathsf{Str2}}, \ldots, \mathsf{Jam}_{\mathsf{StrN}}\}$$

$$\mathbf{q}_1(\mathbf{m}) = p_{\mathbf{m}}(\mathsf{Day} = \mathsf{Mon}, \mathsf{Jam}_{\mathsf{Wwood}} = 1)$$

$\Rightarrow$ ***marginals***



© fineartamerica.com

Tractable Probabilistic Inference

A class of queries $\mathcal{Q}$ *is tractable on a family of probabilistic models* $\mathcal{M}$
iff for any query $\mathbf{q} \in \mathcal{Q}$ *and model* $\mathbf{m} \in \mathcal{M}$
***exactly** computing* $\mathbf{q}(\mathbf{m})$ *runs in time* $O(\mathsf{poly}(|\mathbf{m}|))$.

$\Longrightarrow$ *often* poly *will in fact be **linear**!*

Complete evidence (EVI)

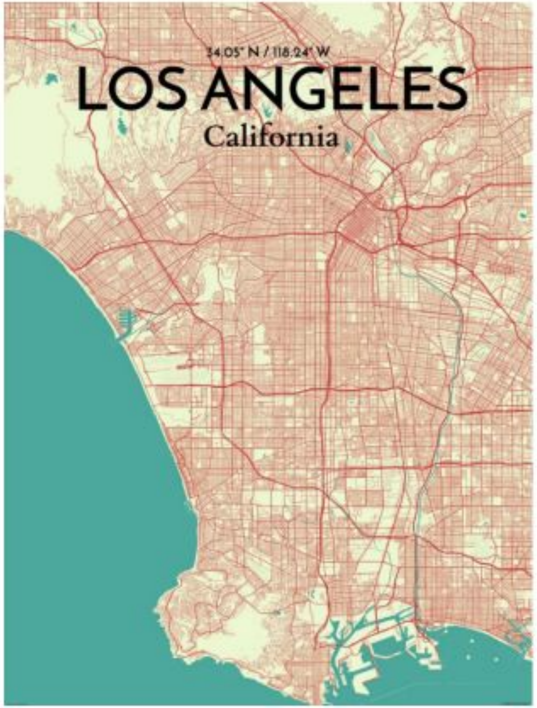
$\mathbf{q}_3$: *What is the probability that today is a Monday at 12.00 and there is a traffic jam only on Westwood Blvd.?*

$\mathbf{X} = \{\mathsf{Day}, \mathsf{Time}, \mathsf{Jam}_{\mathsf{Wwood}}, \mathsf{Jam}_{\mathsf{Str2}}, \ldots, \mathsf{Jam}_{\mathsf{StrN}}\}$

$\mathbf{q}_3(\mathbf{m}) = p_{\mathbf{m}}(\mathbf{X} = \{\mathsf{Mon}, 12.00, 1, 0, \ldots, 0\})$

...fundamental in ***maximum likelihood learning***

$$\theta_{\mathbf{m}}^{\mathsf{MLE}} = \text{argmax}_{\theta} \prod_{\mathbf{x}\in\mathcal{D}} p_{\mathbf{m}}(\mathbf{x};\theta)$$



~~Variational Autoencoders~~

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] - \mathbb{KL}(q_\phi(\mathbf{z} \mid \mathbf{x})||p(\mathbf{z}))$$

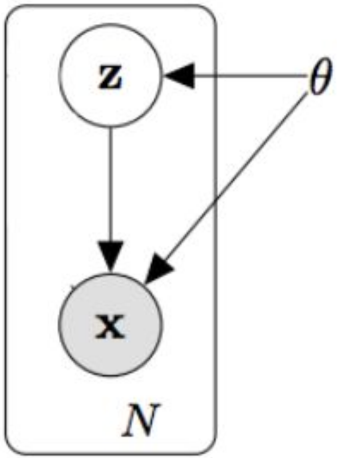
- an explicit likelihood model!
- ... but computing $\log p_\theta(\mathbf{x})$ is intractable

  $\Rightarrow$ *an infinite and uncountable mixture*

  $\Rightarrow$ *no tractable EVI*

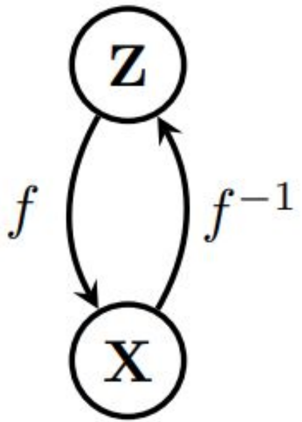
- we need to optimize the ELBO...

  $\Rightarrow$ *which is "tricky"*



Normalizing flows

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f^{-1}(\mathbf{x})) \left|\det\left(\frac{\delta f^{-1}}{\delta \mathbf{x}}\right)\right|$$



- an explicit likelihood!
  $\Rightarrow$ *tractable EVI queries!*
- many neural variants
  - RealNVP *(Dinh et al. 2016)*,
    MAF *(Papamakarios et al. 2017)*
  - MADE *(Germain et al. 2015)*,
    PixelRNN *(Oord et al. 2016)*

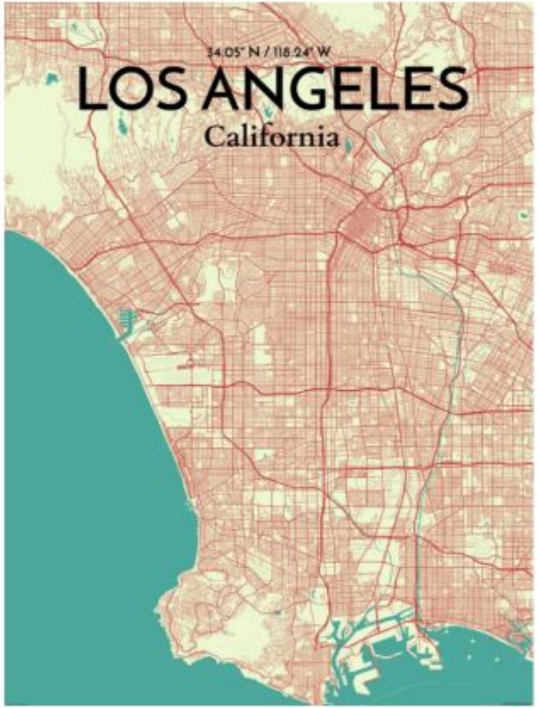
Marginal queries (MAR)

$\mathbf{q}_1$: *What is the probability that today is a Monday ~~at 12:00~~ and there is a traffic jam ~~only~~ on Westwood Blvd.?*

$$\mathbf{q}_1(\mathbf{m}) = p_{\mathbf{m}}(\text{Day} = \text{Mon}, \text{Jam}_{\text{Wwood}} = 1)$$

General: $p_{\mathbf{m}}(\mathbf{e}) = \int p_{\mathbf{m}}(\mathbf{e}, \mathbf{H})\, d\mathbf{H}$

where $\mathbf{E} \subset \mathbf{X}, \quad \mathbf{H} = \mathbf{X} \setminus \mathbf{E}$



© fineartamerica.com

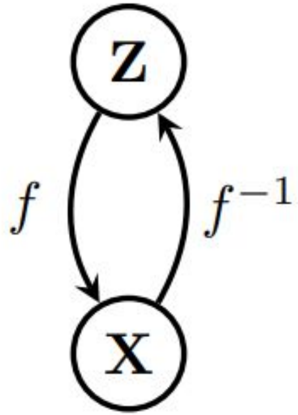
~~Normalizing flows~~

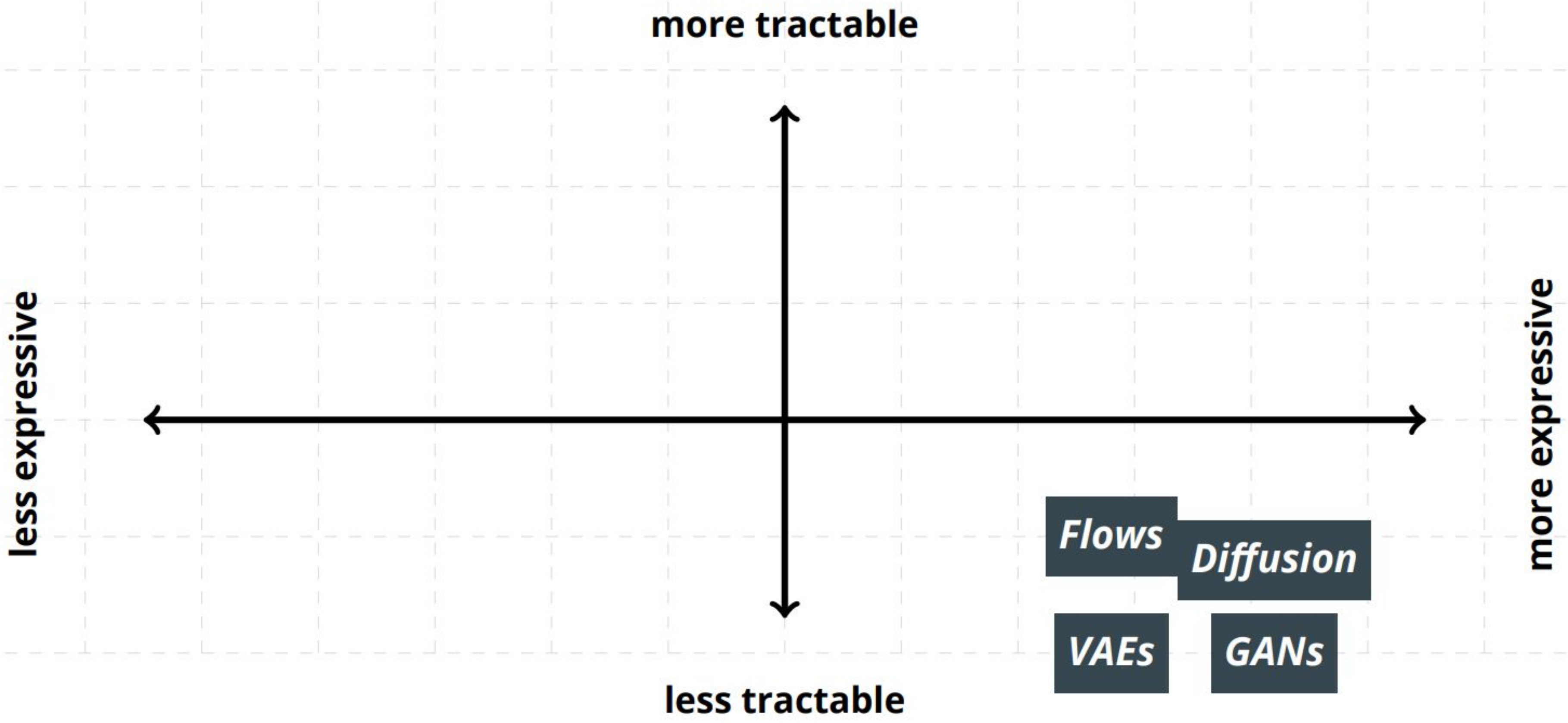
$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f^{-1}(\mathbf{x})) \left|\det\left(\frac{\delta f^{-1}}{\delta \mathbf{x}}\right)\right|$$

- an explicit likelihood!

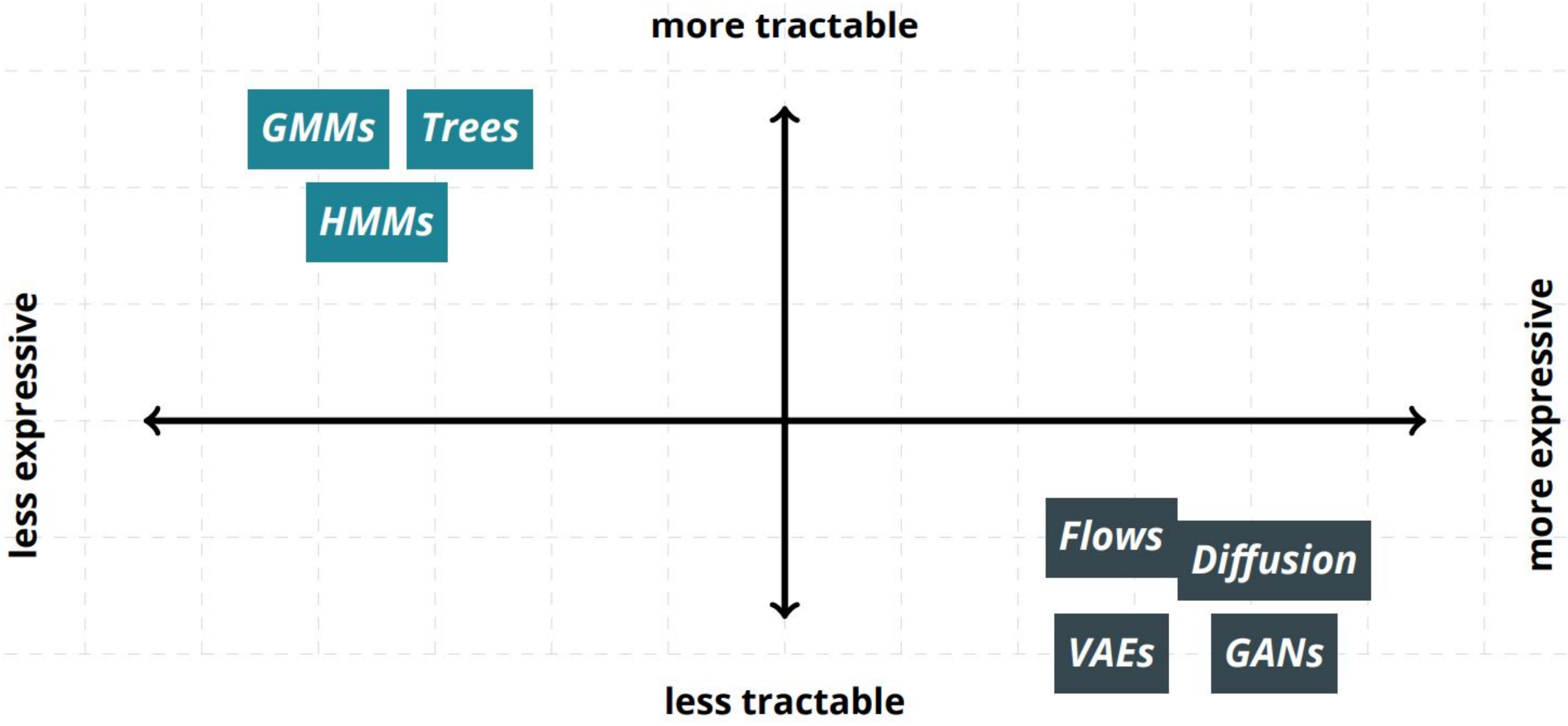
  $\Rightarrow$ *tractable EVI queries!*

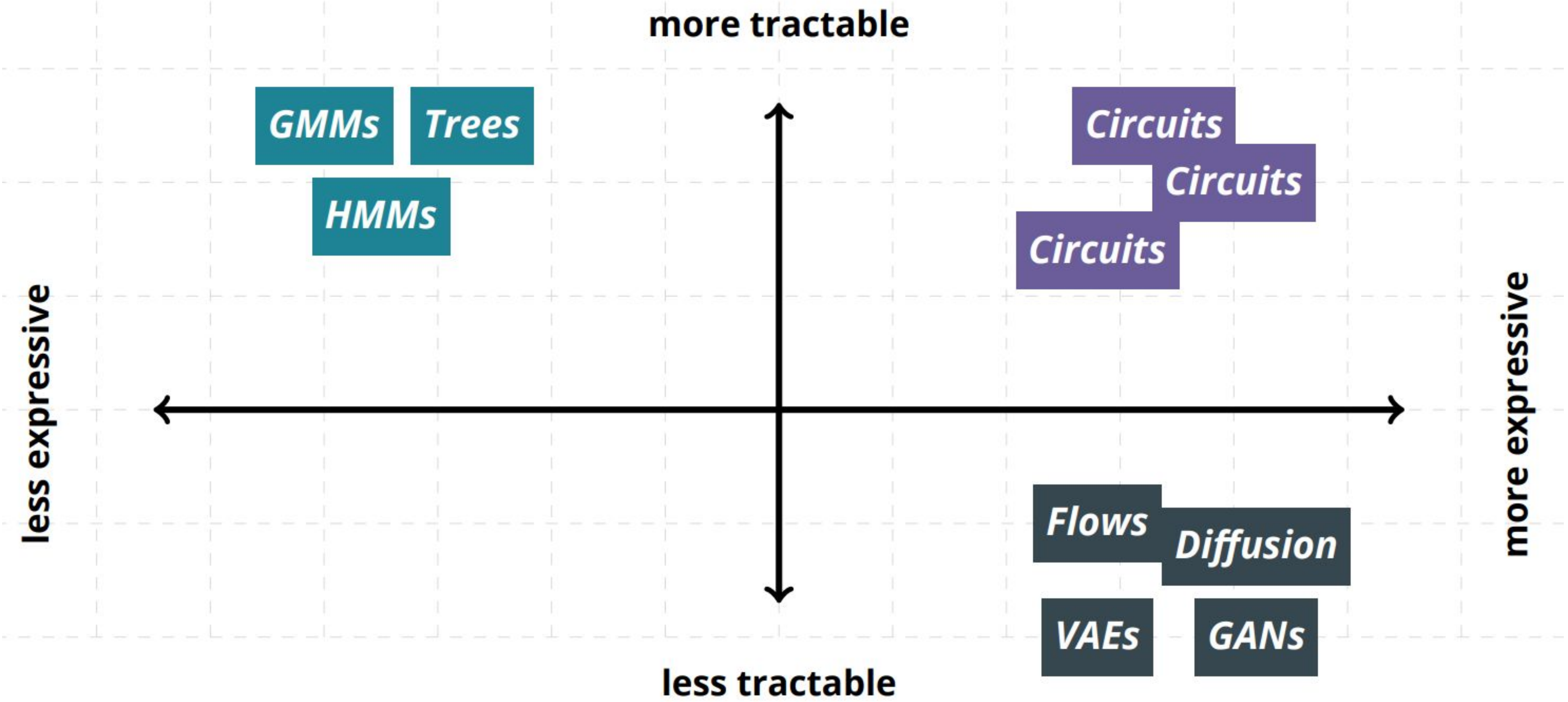
- ***MAR is generally intractable***: we cannot easily integrate over high-dimensional f

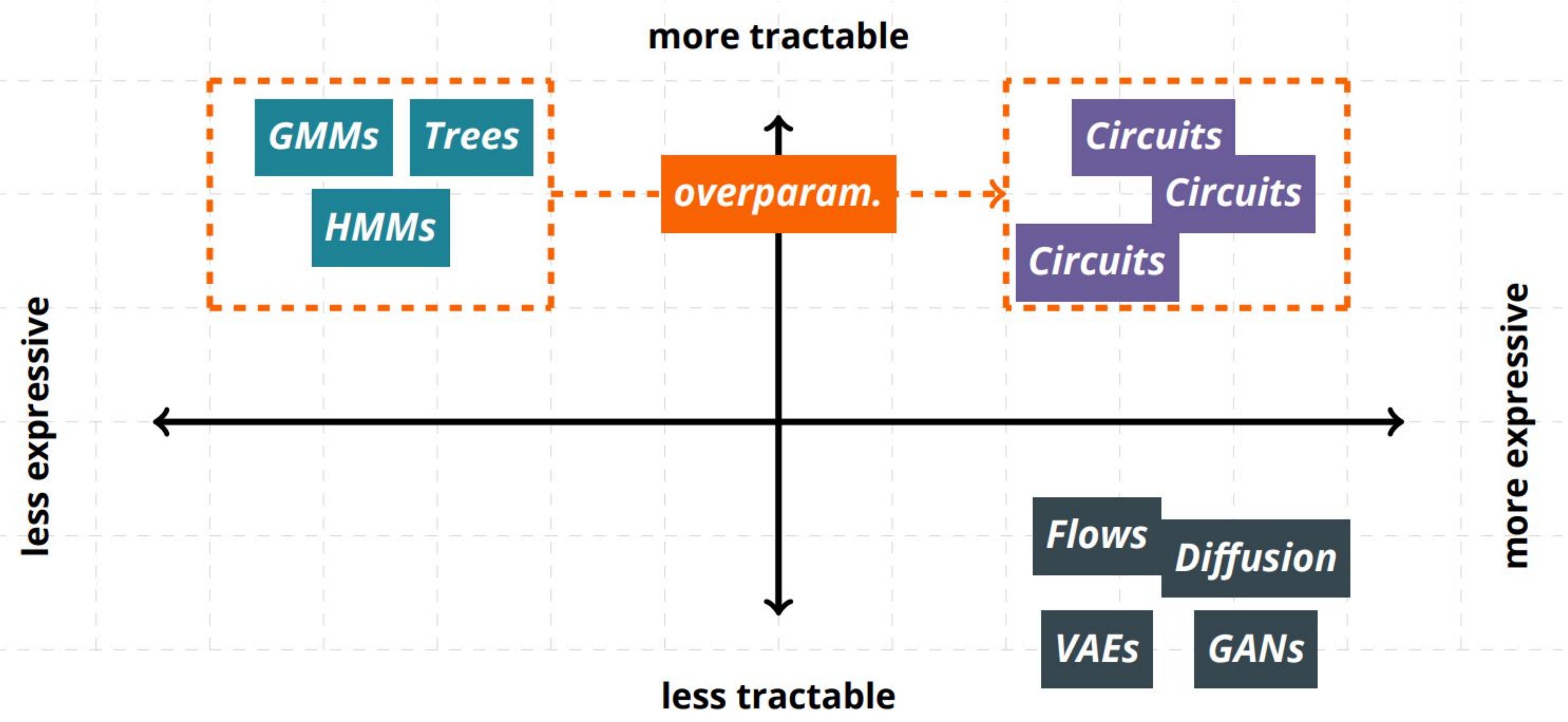


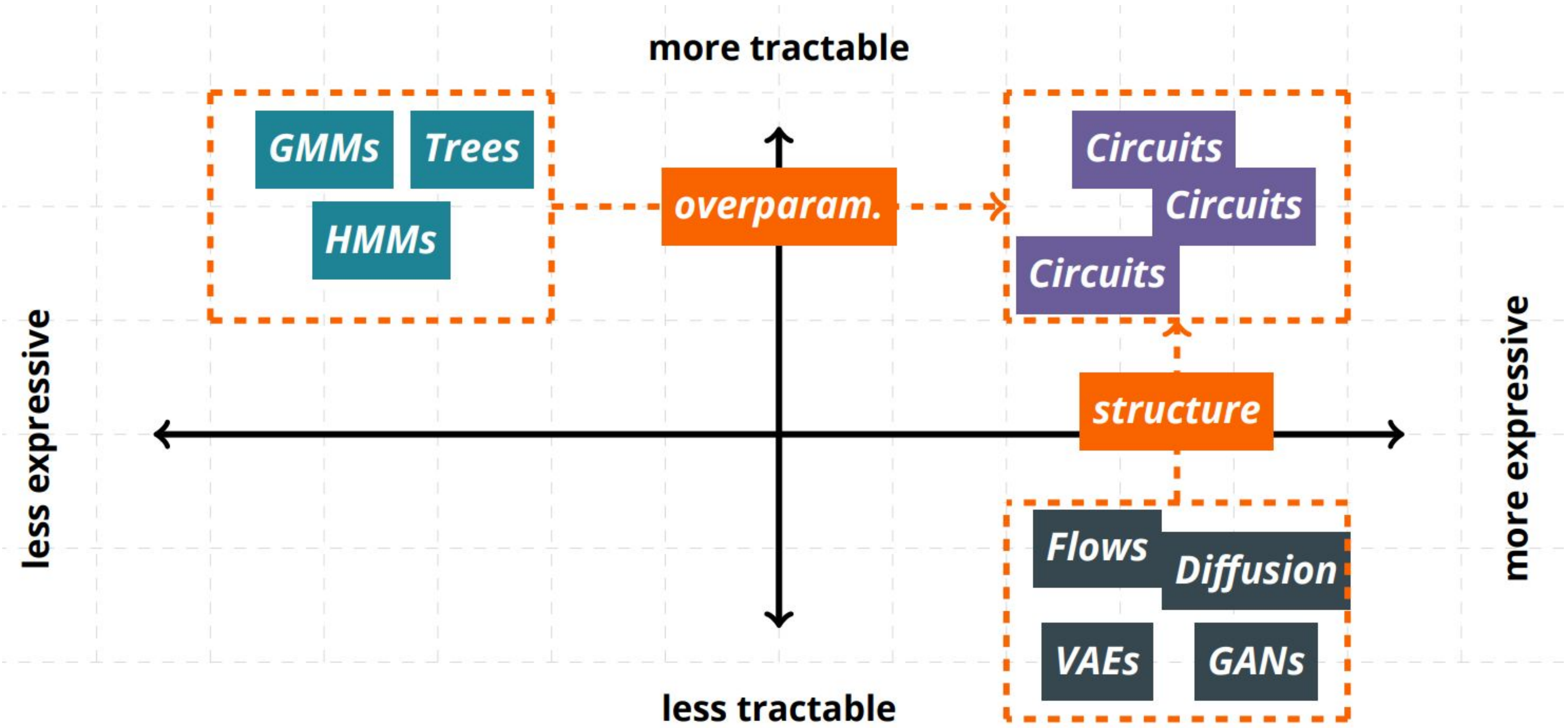


more tractable
less expressive
more expressive
Flows
Diffusion
VAEs
GANs
less tractable


more tractable
GMMs
Trees
HMMs
less expressive
more expressive
Flows
Diffusion
VAEs
GANs
less tractable

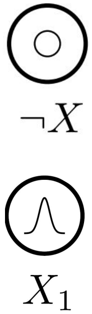

more tractable
GMMs
Trees
HMMs
Circuits
Circuits
Circuits
less expressive
more expressive
Flows
Diffusion
VAEs
GANs
less tractable


more tractable
GMMs
Trees
HMMs
overparam.
Circuits
Circuits
Circuits
less expressive
more expressive
Flows
Diffusion
VAEs
GANs
less tractable


more tractable
GMMs
Trees
HMMs
overparam.
Circuits
Circuits
Circuits
less expressive
structure
more expressive
Flows
Diffusion
VAEs
GANs
less tractable

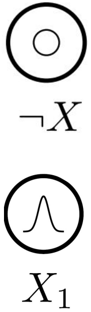
Probabilistic circuits

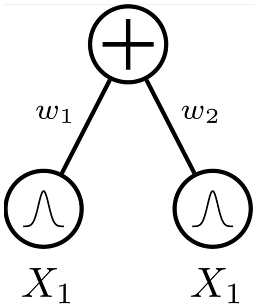
computational graphs that recursively define distributions



Probabilistic circuits

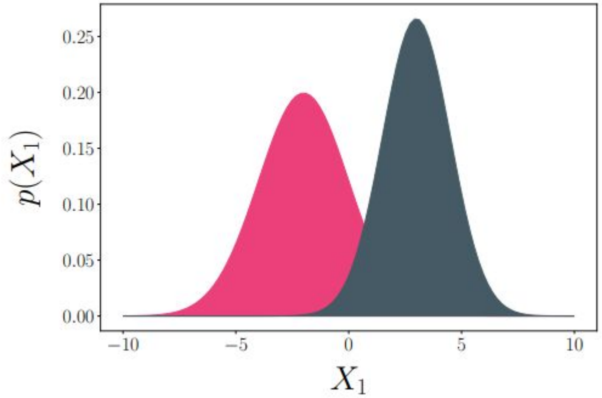
computational graphs that recursively define distributions





$$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)$$

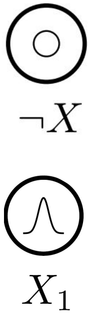
⇒ *mixtures*

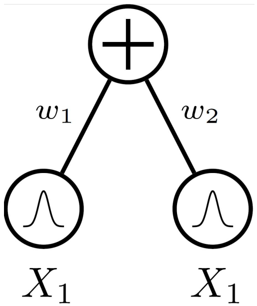


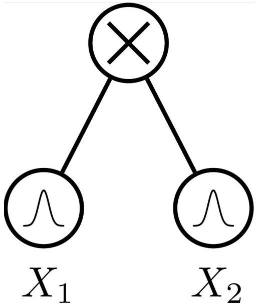
$$p(X) = p(Z = 1) \cdot p_1(X|Z = 1) + p(Z = 2) \cdot p_2(X|Z = 2)$$

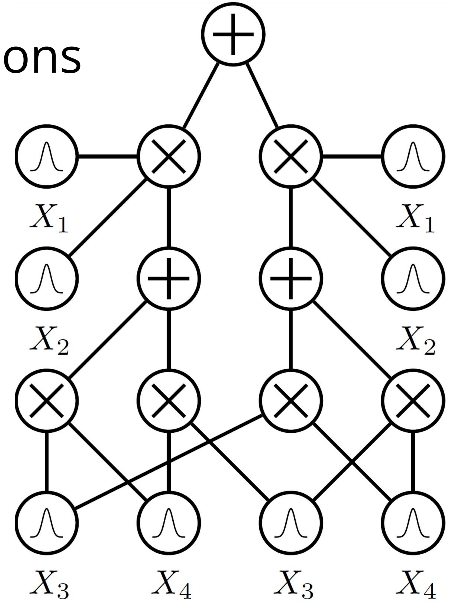
Probabilistic circuits

computational graphs that recursively define distributions









$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)$

$\Rightarrow$ *mixtures*

$p(X_1, X_2) = p(X_1) \cdot p(X_2)$

$\Rightarrow$ *factorizations*

Fully factorized

NaiveBayes *AndOrGraphs* *PDGs*

Trees *PSDDs* *CNets* *LTMs* *SPNs* *NADEs*

Thin Junction Trees *ACs* *MADEs* *MAFs* *VAEs*

DPPs *FVSBNs* *TACs* *IAFs* *NAFs* *RAEs*

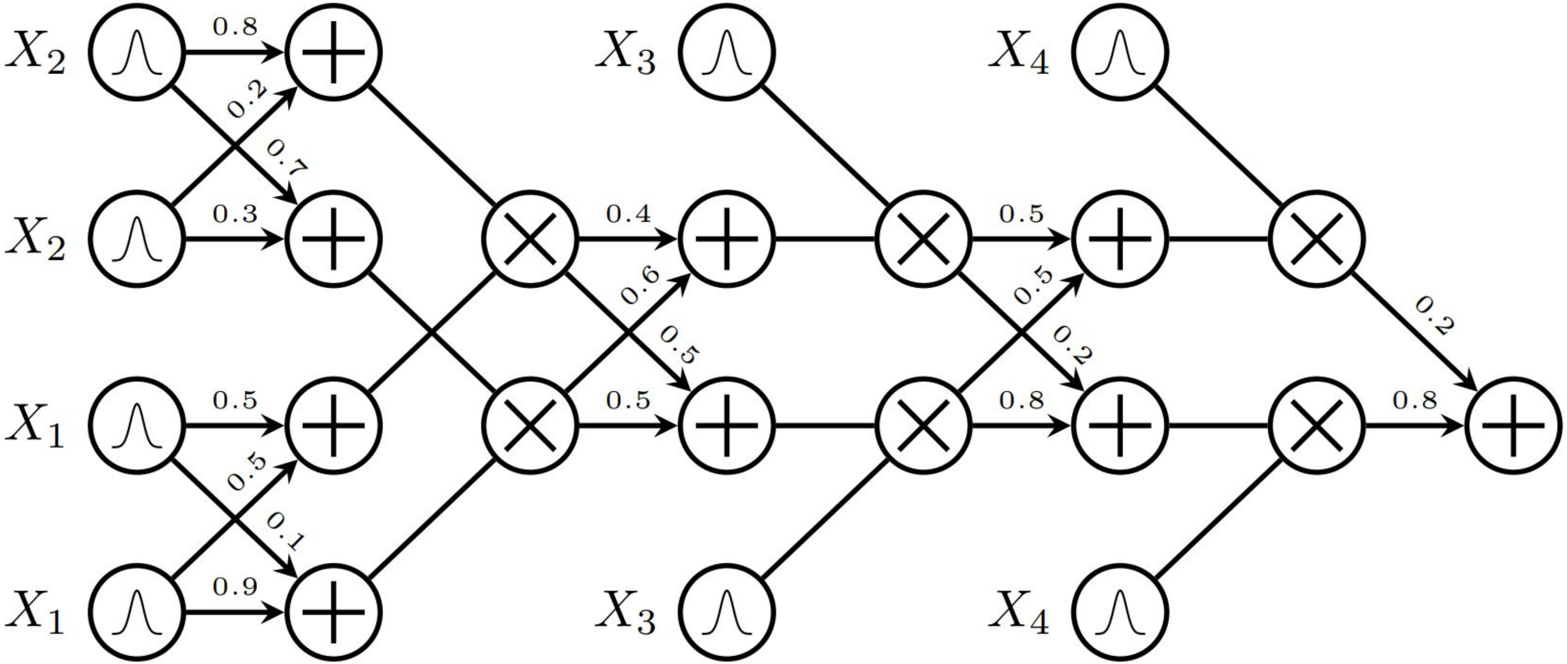
Mixtures *BNs* *NICE* *FGs* *GANs*

RealNVP *MNs*

a *unifying framework* for tractable models

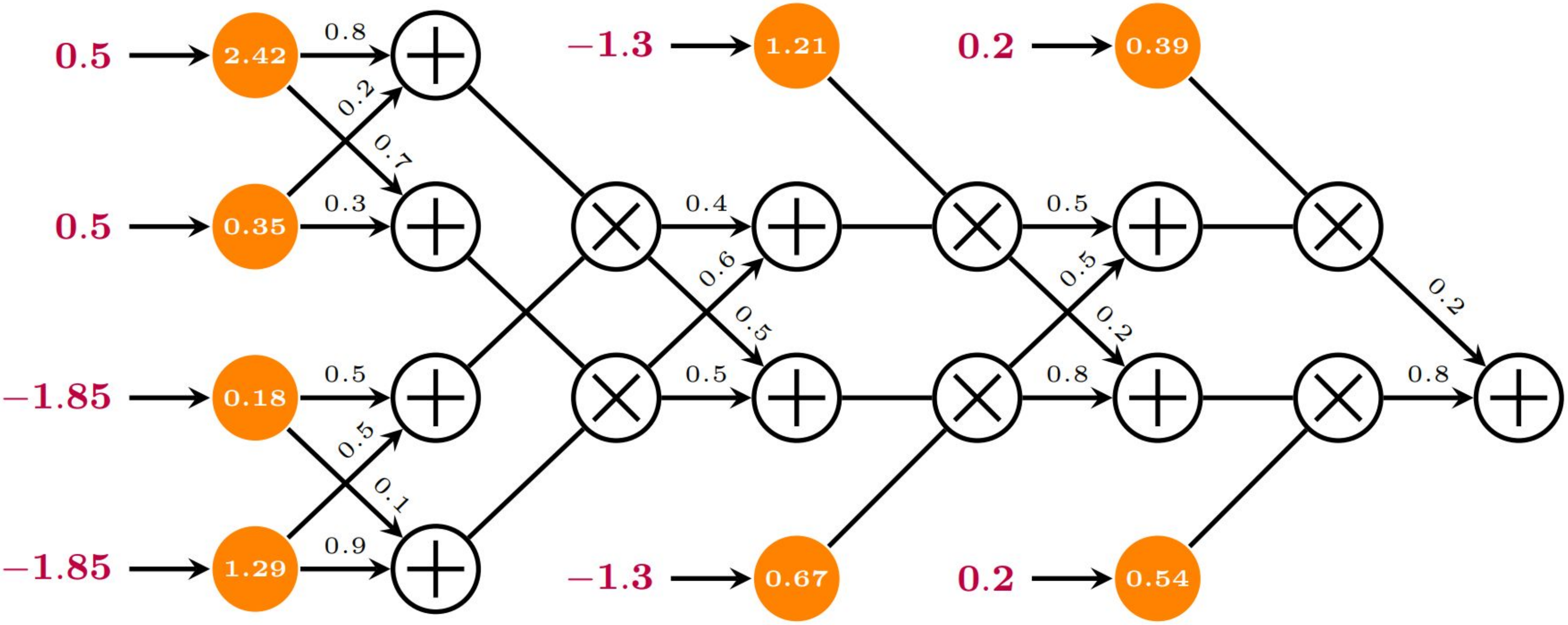
Likelihood

$$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$$



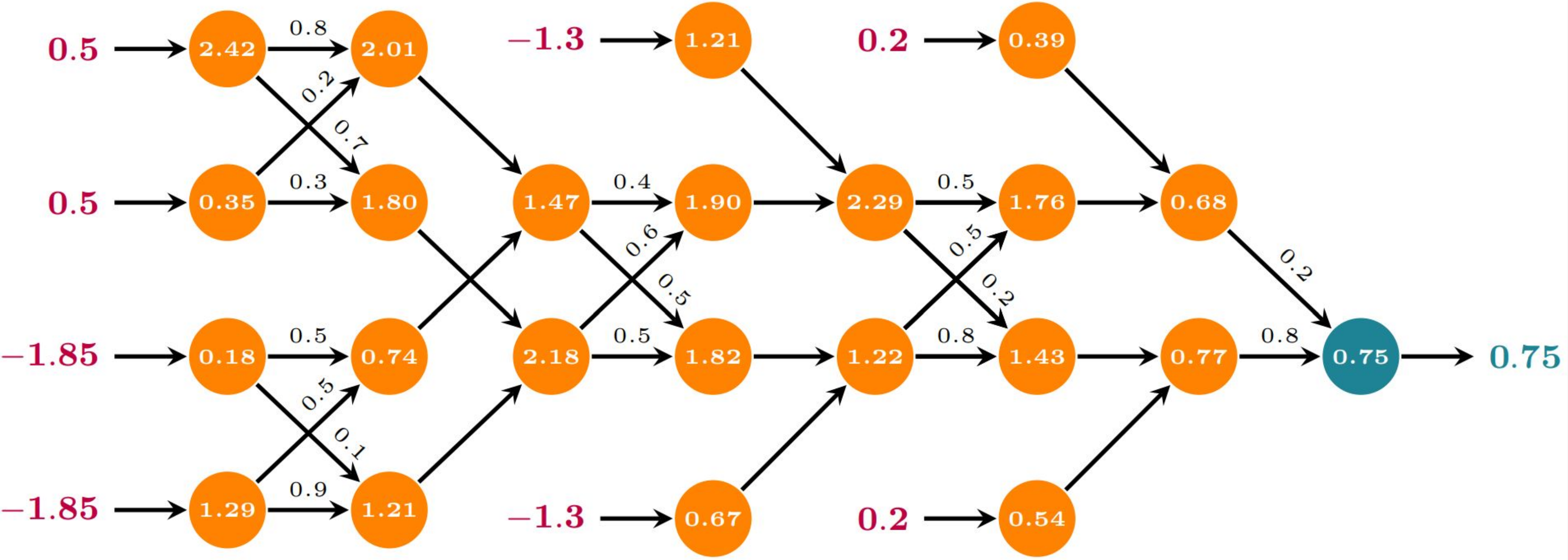
Likelihood

$$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$$



Likelihood

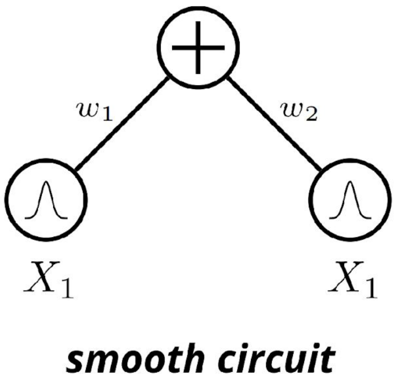
$$p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2)$$

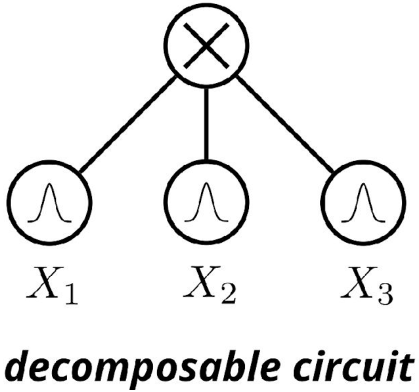


Tractable marginals

A sum node is ***smooth*** if its children depend on the same set of variables.

A product node is ***decomposable*** if its children depend on disjoint sets of variables.





Darwiche and Marquis, “A Knowledge Compilation Map”, 2002

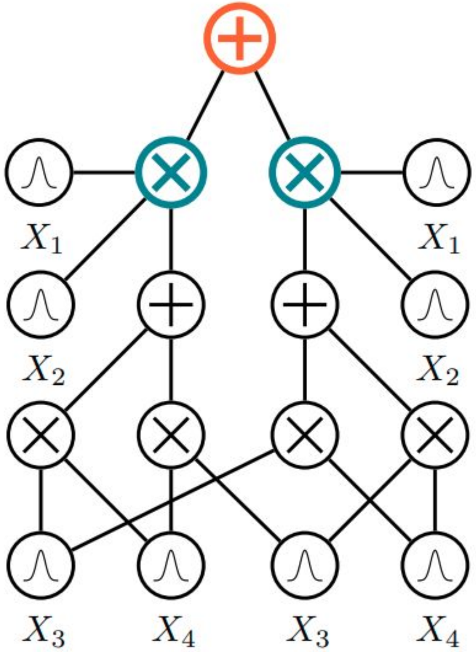
Smoothness + decomposability = tractable MAR

If $p(\mathbf{x}) = \sum_i w_i p_i(\mathbf{x})$, (***smoothness***):

$$\int p(\mathbf{x})d\mathbf{x} = \int \sum_i w_i p_i(\mathbf{x})d\mathbf{x} =$$

$$= \sum_i w_i \int p_i(\mathbf{x})d\mathbf{x}$$

$\Rightarrow$ *integrals are "pushed down" to children*



[Darwiche & Marquis JAIR 2001, Poon & Domingos UAI11]

Smoothness + decomposability = tractable MAR

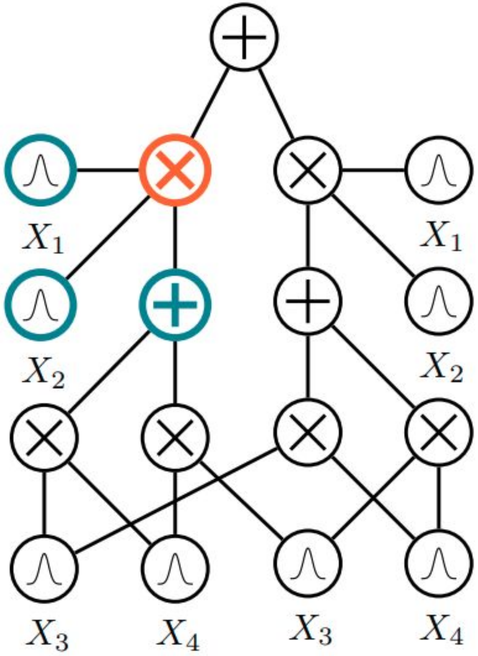
If $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{y})p(\mathbf{z})$, (***decomposability***):

$$\int\int\int p(\mathbf{x}, \mathbf{y}, \mathbf{z}) d\mathbf{x} d\mathbf{y} d\mathbf{z} =$$

$$= \int\int\int p(\mathbf{x})p(\mathbf{y})p(\mathbf{z}) d\mathbf{x} d\mathbf{y} d\mathbf{z} =$$

$$= \int p(\mathbf{x}) d\mathbf{x} \int p(\mathbf{y}) d\mathbf{y} \int p(\mathbf{z}) d\mathbf{z}$$

$\Rightarrow$ *integrals decompose into easier ones*



Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

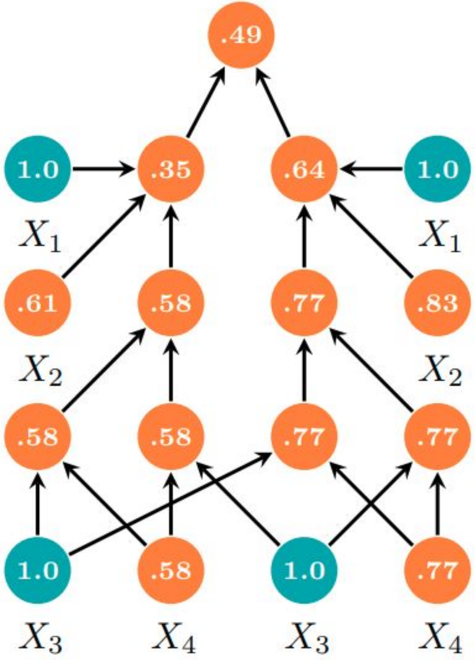
$\Rightarrow$ *linear in circuit size!*

E.g. to compute $p(x_2, x_4)$:

- leafs over X_1 and X_3 output $\boldsymbol{Z}_i = \int p(x_i)dx_i$

  $\Rightarrow$ *for normalized leaf distributions:* 1.0

- leafs over X_2 and X_4 output **EVI**
- feedforward evaluation (bottom-up)



Outline

1. What are probabilistic circuits?

   tractable deep generative models

2. **What are they useful for?**

   controlling generative AI

3. What is the underlying theory?

   probability generating polynomials

Cute, but these models cannot compete?

bpd	2008-2020
Tabular	😐
MNIST	😱
F-MNIST	😱
EMNIST-L	😱
CIFAR	😱
Imagenet32	😱
Imagenet64	😱

Cute, but these models cannot compete?

bpd	2008-2020	2020-2021
Tabular	😐	😊
MNIST	😱	😱 > 1.67
F-MNIST	😱	😱 > 4.29
EMNIST-L	😱	😱 > 2.73
CIFAR	😱	😱
Imagenet32	😱	😱
Imagenet64	😱	😱

General-purpose architecture

Cute, but these models cannot compete?

bpd	2008-2020	2020-2021	ICLR 22
Tabular	😐	😊	🍰
MNIST	😱	😱 > 1.67	1.20
F-MNIST	😱	😱 > 4.29	3.34
EMNIST-L	😱	😱 > 2.73	1.80
CIFAR	😱	😱	😱 > 5.50
Imagenet32	😱	😱	😱
Imagenet64	😱	😱	😱

General-purpose architecture

Custom GPU kernels

Cute, but these models cannot compete?

bpd	2008-2020	2020-2021	ICLR 22	NeurIPS 22
Tabular	😐	😊	🍰	🍰
MNIST	😱	😱 > 1.67	1.20	1.14
F-MNIST	😱	😱 > 4.29	3.34	3.27
EMNIST-L	😱	😱 > 2.73	1.80	1.58
CIFAR	😱	😱	😱 > 5.50	😱
Imagenet32	😱	😱	😱	😱
Imagenet64	😱	😱	😱	😱

General-purpose architecture

Custom GPU kernels

Pruning without losing likelihood

Cute, but these models cannot compete?

bpd	2008-2020	2020-2021	ICLR 22	NeurIPS 22
Tabular	😐	😊	🍰	🍰
MNIST	😱	😱 > 1.67	1.20	1.14
F-MNIST	😱	😱 > 4.29	3.34	3.27
EMNIST-L	😱	😱 > 2.73	1.80	1.58
CIFAR	😱	😱	😱 > 5.50	😱
Imagenet32	😱	😱	😱	😱
Imagenet64	😱	😱	😱	😱

	Discrete Flow	Hierarchical VAE	PixelVAE
MNIST	1.90	1.27	1.39
F-MNIST	3.47	3.28	3.66
EMNIST-L	1.95	1.84	2.26

Cute, but these models cannot compete?

bpd	2008-2020	2020-2021	ICLR 22	NeurIPS 22	ICLR 23
Tabular	😐	😊	🍰	🍰	🍰
MNIST	😱	😱 > 1.67	1.20	1.14	🍰
F-MNIST	😱	😱 > 4.29	3.34	3.27	🍰
EMNIST-L	😱	😱 > 2.73	1.80	1.58	🍰
CIFAR	😱	😱	😱 > 5.50	😱	4.38
Imagenet32	😱	😱	😱	😱	4.39
Imagenet64	😱	😱	😱	😱	4.12

General-purpose architecture

Custom GPU kernels

Pruning without losing likelihood

Latent Variable Distillation

Cute, but these models cannot compete?

bpd	2008-2020	2020-2021	ICLR 22	NeurIPS 22	ICLR 23	ICML 23
Tabular	😐	😊	🍰	🍰	🍰	🍰
MNIST	😱	😱 > 1.67	1.20	1.14	🍰	🍰
F-MNIST	😱	😱 > 4.29	3.34	3.27	🍰	🍰
EMNIST-L	😱	😱 > 2.73	1.80	1.58	🍰	🍰
CIFAR	😱	😱	😱 > 5.50	😱	4.38	3.87
Imagenet32	😱	😱	😱	😱	4.39	4.06
Imagenet64	😱	😱	😱	😱	4.12	3.80

	Flow	Hierarchical VAE	Diffusion
CIFAR	3.35	3.08	2.65
Imagenet32	4.09	3.96	3.72
Imagenet64	3.81	-	3.40

Generate a sentence using "frisbee", "caught" and "dog", following the given order.

Generate a sentence using "frisbee", "caught" and "dog", following the given order.

After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT

Generate a sentence using "frisbee", "caught" and "dog", following the given order.

After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT

That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.

Here's the correct sentence: The dog caught the frisbee in mid-air, showing off its amazing catching skills.

ChatGPT

Generate a sentence using "frisbee", "caught" and "dog", following the given order.

After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT

That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.

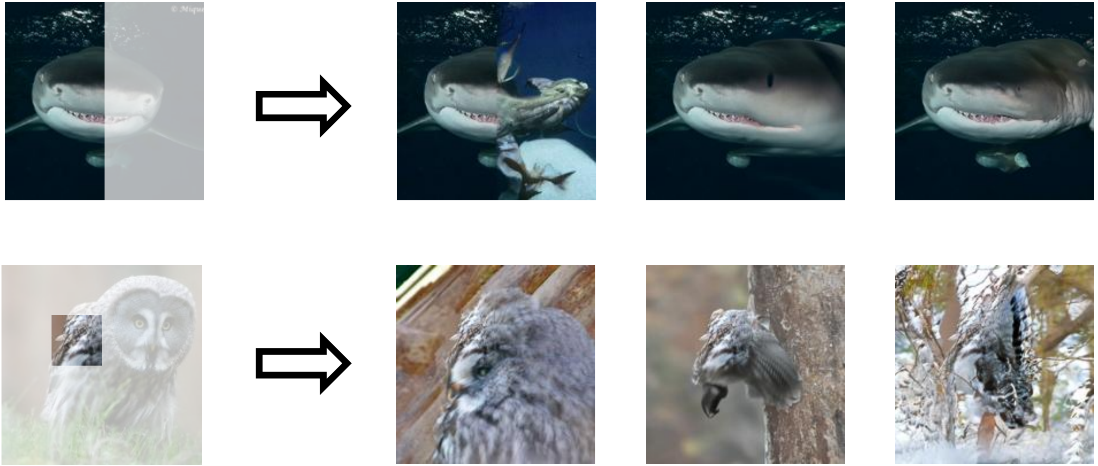
Here's the correct sentence: The dog caught the frisbee in mid-air, showing off its amazing catching skills.

ChatGPT

A frisbee is caught by a dog.
A pair of frisbee players are caught in a dog fight.

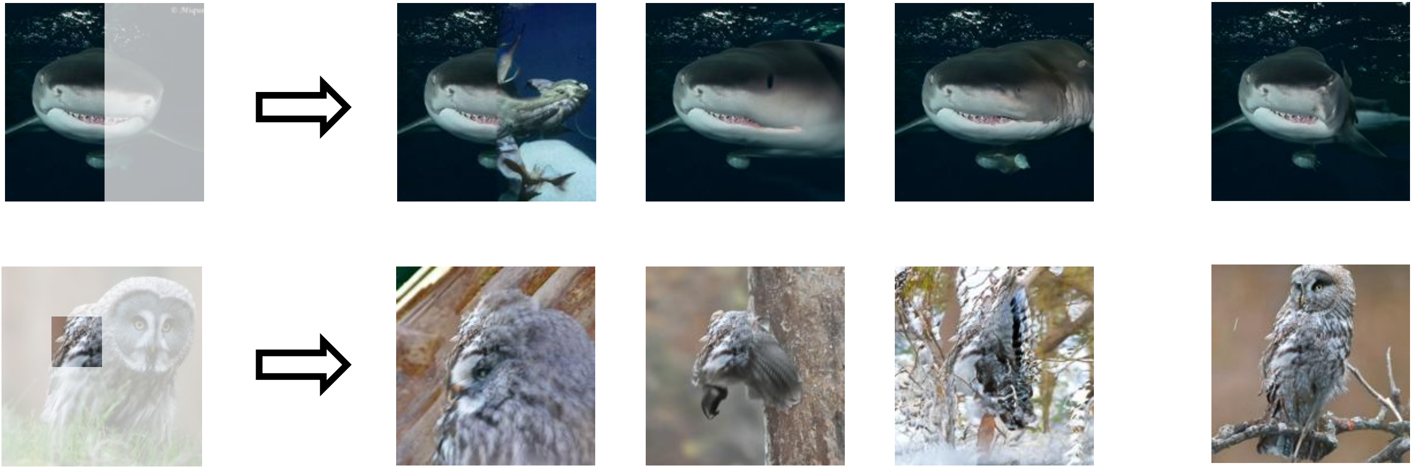
GeLaTo

Inpainting/constrained generation is still challenging

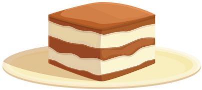


Diffusion models are good at fine-grained details, but not so good at global consistency of generated images.

Inpainting/constrained generation is still challenging



Tiramisu



What do we have?

Prefix: "The weather is"

Constraint α: text contains "winter"

Model only does $p(\text{next-token}|\text{prefix}) =$

cold	0.05
warm	0.10

Train some $q(.\,|\alpha)$ for a specific task distribution $\alpha \sim p_{\text{task}}$

(amortized inference, encoder, masked model, seq2seq, prompt tuning,...)

Train $q(\text{next-token}|\text{prefix}, \alpha)$

What do we need?

Prefix: “The weather is”

Constraint α: text contains “winter”

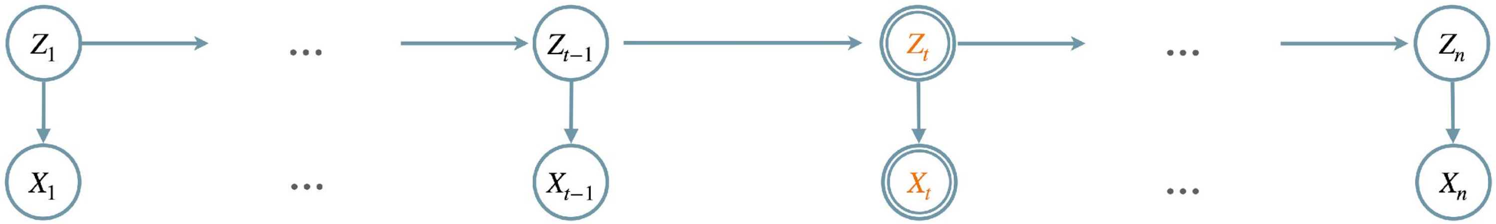
Generate from $p(\text{next-token}|\text{prefix}, \alpha) =$

cold	0.50
warm	0.01

$$\propto \sum_{\text{text}} p(\text{next-token}, \text{text}, \text{prefix}, \alpha)$$

Marginalization!

Step 1: Distill an HMM p_{hmm} that approximates p_{gpt}



1. HMM with 4096 hidden states and 50k emission tokens

2. Data sampled from GPT2-large (domain-adapted), minimizing KL(p_{gpt} // p_{HMM})

3. Leverages latent variable distillation for training PCs at scale [ICLR 23].
   (Cluster embeddings of examples to estimate latent Z_i)

Anji Liu, Honghua Zhang and Guy Van den Broeck. Scaling Up Probabilistic Circuits by Latent Variable Distillation, 2023.

CommonGen: a Challenging Benchmark

Given 3-5 keywords, generate a sentence using all keywords, in any order and any form of inflections. e.g.,

Input: snow drive car

Reference 1: A car drives down a snow covered road.

Reference 2: Two cars drove through the snow.

Constraint α in CNF: $(w_{1,1} \vee \ldots \vee w_{1,d1}) \wedge \ldots \wedge (w_{m,1} \vee \ldots \vee w_{m,dm})$

Each clause represents the inflections for one keyword.

Computing $p(\alpha \mid x_{1:t+1})$

For constraint α in CNF:

$$(w_{1,1} \lor \ldots \lor w_{1,d1}) \land \ldots \land (w_{m,1} \lor \ldots \lor w_{m,dm})$$

where each w_{ij} is a keyword (i.e. a **string of tokens**), representing that w_{ij} appears in the generated text.

e.g., α = ("swims" ∨ "like swimming") ∧ ("lake" ∨ "pool")

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. [Tractable Control for Autoregressive Language Generation](), 2023.

Computing $p(\alpha \mid x_{1:t+1})$

For constraint α in CNF:

$$(w_{1,1} \vee \ldots \vee w_{1,d1}) \wedge \ldots \wedge (w_{m,1} \vee \ldots \vee w_{m,dm})$$

where each w_{ij} is a keyword (i.e. a **string of tokens**),
representing that w_{ij} appears in the generated text.

e.g., α = ("swims" ∨ "like swimming") ∧ ("lake" ∨ "pool")

Efficient algorithm:
For m clauses and sequence length n, time-complexity for HMM generation is $O(2^{|m|}n)$

Trick: dynamic programming with clever preprocessing and local belief updates

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

GeLaTo Overview

Lexical Constraint α: sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} \mid \alpha, x_{1:t} =$ "the weather is"$)$

intractable → Pre-trained Language Model

efficient → Tractable Probabilistic Model

Minimize KL-divergence

x_{t+1}	$\Pr_{LM}(x_{t+1} \mid x_{1:t})$
cold	0.05
warm	0.10

x_{t+1}	$\Pr_{TPM}(\alpha \mid x_{t+1}, x_{1:t})$
cold	0.50
warm	0.01

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

GeLaTo Overview

Lexical Constraint α: sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} \mid \alpha, x_{1:t} =$ "the weather is"$)$

intractable → Pre-trained Language Model

efficient → Tractable Probabilistic Model

Minimize KL-divergence

x_{t+1}	$\Pr_{LM}(x_{t+1} \mid x_{1:t})$
cold	0.05
warm	0.10

x_{t+1}	$\Pr_{TPM}(\alpha \mid x_{t+1}, x_{1:t})$
cold	0.50
warm	0.01

x_{t+1}	$p(x_{t+1} \mid \alpha, x_{1:t})$
cold	0.025
warm	0.001

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Step 2: Control p_{gpt} via p_{hmm}

<u>*Unsupervised*</u>

Language model is not
fine-tuned/prompted to satisfy constraints

By Bayes rule:

$$p_{gpt}(x_{t+1} \mid x_{1:t}, \alpha) \propto p_{gpt}(\alpha \mid x_{1:t+1}) \cdot p_{gpt}(x_{t+1} \mid x_{1:t})$$

Assume $p_{hmm}(\alpha \mid x_{1:t+1}) \approx p_{gpt}(\alpha \mid x_{1:t+1})$, we generate from:

$$p(x_{t+1} \mid x_{1:t}, \alpha) \propto p_{hmm}(\alpha \mid x_{1:t+1}) \cdot p_{gpt}(x_{t+1} \mid x_{1:t})$$

Method	*Generation Quality*								*Constraint Satisfaction*			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
Unsupervised	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*
InsNet (Lu et al., 2022a)	-	-	18.7	-	-	-	-	-	**100.0**	-	**100.0**	-
NeuroLogic (Lu et al., 2021)	-	41.9	-	24.7	-	14.4	-	27.5	-	96.7	-	-
A*esque (Lu et al., 2022b)	-	**44.3**	-	28.6	-	15.6	-	29.6	-	97.1	-	-
NADO (Meng et al., 2022)	-	-	26.2	-	-	-	-	-	96.1	-	-	-
GeLaTo	**44.6**	44.1	**29.9**	**29.4**	**16.0**	**15.8**	**31.3**	**31.0**	**100.0**	**100.0**	**100.0**	**100.0**

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Step 2: Control p_{gpt} via p_{hmm}

Supervised

Language model is fine-tuned to perform constrained generation (e.g. seq2seq)

Empirically $p_{HMM}(\alpha \mid x_{1:t+1}) \approx p_{gpt}(\alpha \mid x_{1:t+1})$ does not hold well enough;

we view $p_{HMM}(x_{t+1} \mid x_{1:t}, \alpha)$ and $p_{gpt}(x_{t+1} \mid x_{1:t})$ as classifiers trained for the same task with different biases; thus we generate from their *weighted geometric mean*:

$$p(x_{t+1} \mid x_{1:t}, \alpha) \propto p_{hmm}(x_{t+1} \mid x_{1:t}, \alpha)^{w} \cdot p_{gpt}(x_{t+1} \mid x_{1:t})^{1-w}$$

Method	*Generation Quality*								*Constraint Satisfaction*			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
Supervised	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*
NeuroLogic (Lu et al., 2021)	-	42.8	-	26.7	-	14.7	-	30.5	-	97.7	-	93.9†
A*esque (Lu et al., 2022b)	-	43.6	-	28.2	-	15.2	-	30.8	-	97.8	-	97.9†
NADO (Meng et al., 2022)	44.4†	-	30.8	-	16.1†	-	**32.0**†	-	97.1	-	88.8†	-
GeLaTo	**46.0**	**45.6**	**34.1**	**32.9**	**16.7**	**16.8**	31.3	**31.9**	**100.0**	**100.0**	**100.0**	**100.0**

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Advantages of GeLaTo:

1. Constraint α is <u>guaranteed to be satisfied</u>:
   for any next-token x_{t+1} that would make α unsatisfiable, $p(x_{t+1} \mid x_{1:t}, \alpha) = 0$.

2. Training p_{hmm} <u>does not depend on α</u>,
   which is only imposed at inference (generation) time.

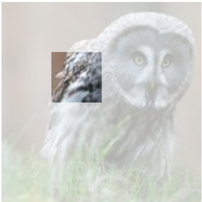
3. Can impose <u>additional tractable constraints</u>:
   - keywords follow a particular order
   - keywords appear at a particular position
   - keywords must not appear

Conclusion: you can control an intractable generative model using a tractable probabilistic circuit.

Inpainting/constrained generation is still challenging

Constrained posterior in diffusion models

Unconstrained denoising step: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \sum_{\tilde{\mathbf{x}}_0} q(\mathbf{x}_{t-1}|\tilde{\mathbf{x}}_0, \mathbf{x}_t) \cdot p_\theta(\tilde{\mathbf{x}}_0|\mathbf{x}_t)$

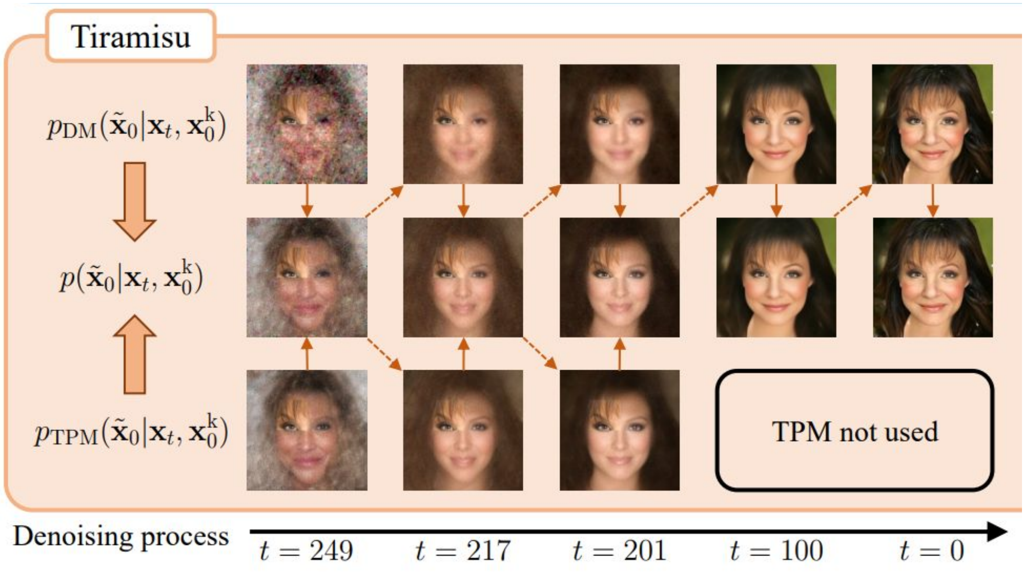


⇩ Constraint c on the generated image (e.g., inpainting)

Constrained denoising step: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, c) := \sum_{\tilde{\mathbf{x}}_0} q(\mathbf{x}_{t-1}|\tilde{\mathbf{x}}_0, \mathbf{x}_t) \cdot p_\theta(\tilde{\mathbf{x}}_0|\mathbf{x}_t, c)$

Computing or sampling from the constrained posterior $p_\theta(\tilde{\mathbf{x}}_0|\mathbf{x}_t, c)$ is **intractable** for diffusion models.

Denoising $p(\tilde{\boldsymbol{x}}_0|\boldsymbol{x}_t, \boldsymbol{x}_0^{\mathrm{k}}) \propto p_{\mathrm{DM}}(\tilde{\boldsymbol{x}}_0|\boldsymbol{x}_t, \boldsymbol{x}_0^{\mathrm{k}})^{\alpha} \cdot p_{\mathrm{TPM}}(\tilde{\boldsymbol{x}}_0|\boldsymbol{x}_t, \boldsymbol{x}_0^{\mathrm{k}})^{1-\alpha}$



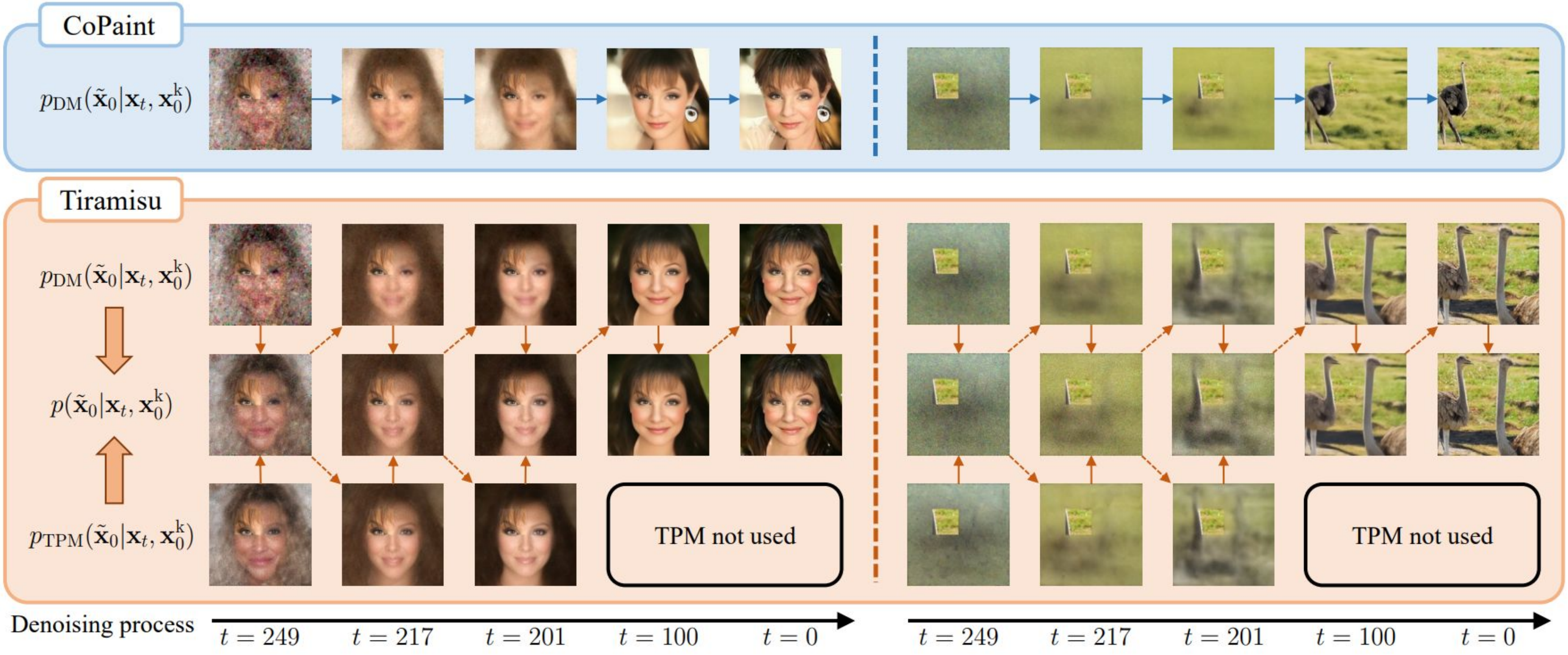
$p_{\mathrm{DM}}(\tilde{\mathbf{x}}_0|\mathbf{x}_t, c)$

From the diffusion model:
Good at generating vivid details

$p_{\mathrm{TPM}}(\tilde{\mathbf{x}}_0|\mathbf{x}_t, c)$

From the probabilistic circuit:
Exact samples – better global coherence

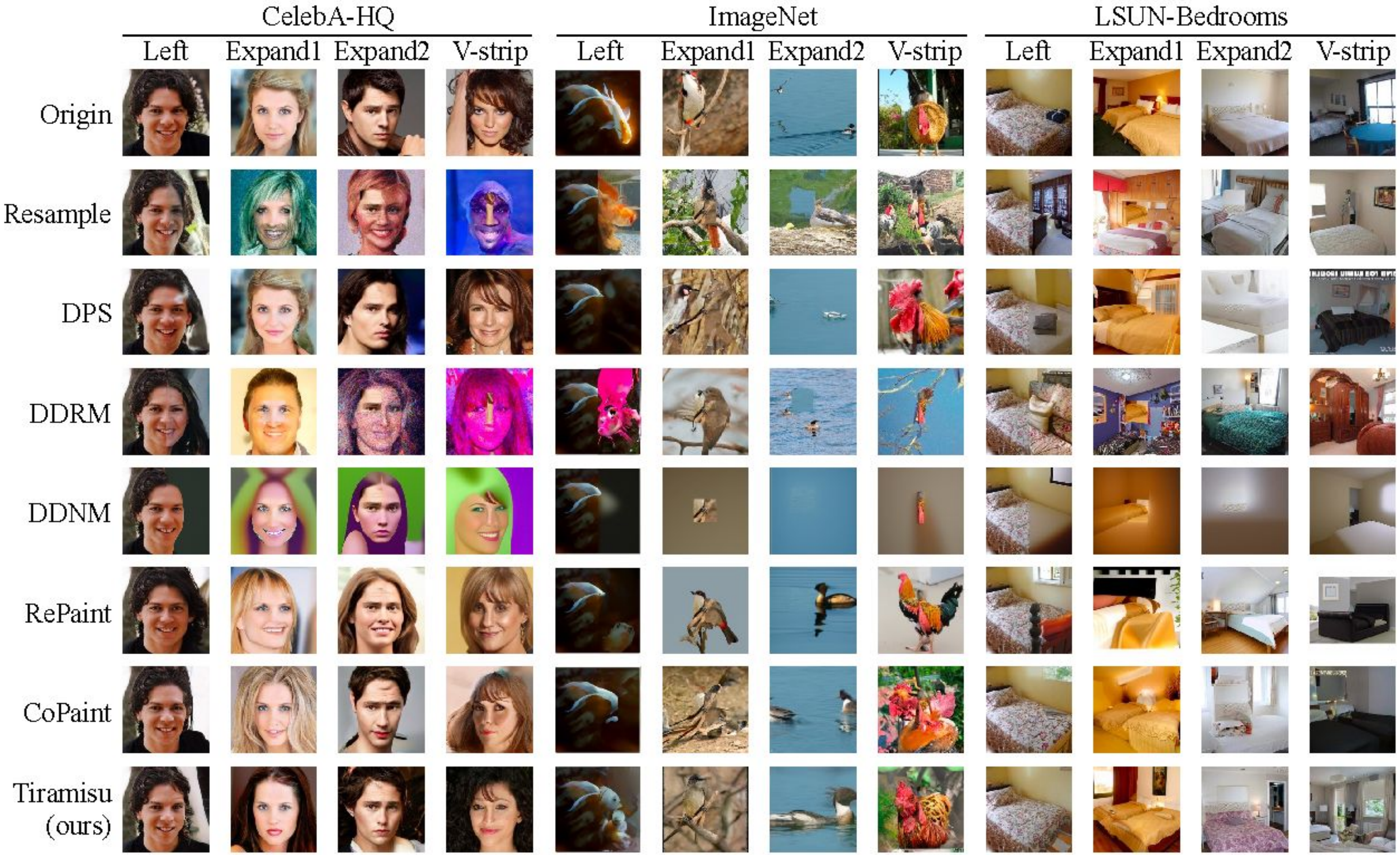
Controlling the denoiser with a probabilistic circuit



High-resolution image benchmarks

Tasks		Algorithms						
Dataset	Mask	Tiramisu (ours)	CoPaint	RePaint	DDNM	DDRM	DPS	Resampling
CelebA-HQ	Left	0.189	**0.185**	0.195	0.254	0.275	0.201	0.257
	Top	0.187	**0.182**	0.187	0.248	0.267	0.187	0.251
	Expand1	**0.454**	0.468	0.504	0.597	0.682	0.466	0.613
	Expand2	0.442	0.455	0.480	0.585	0.686	**0.434**	0.601
	V-strip	**0.487**	0.502	0.517	0.625	0.724	0.535	0.647
	H-strip	**0.484**	0.488	0.517	0.626	0.731	0.492	0.639
ImageNet	Left	**0.286**	0.289	0.296	0.410	0.369	0.327	0.369
	Top	**0.308**	0.312	0.336	0.427	0.373	0.343	0.368
	Expand1	**0.616**	0.623	0.691	0.786	0.726	0.621	0.711
	Expand2	**0.597**	0.607	0.692	0.799	0.724	0.618	0.721
	V-strip	0.646	0.654	0.741	0.851	0.761	**0.637**	0.759
	H-strip	0.657	0.660	0.744	0.851	0.753	**0.647**	0.774
LSUN-Bedroom	Left	**0.285**	0.287	0.314	0.345	0.366	0.314	0.367
	Top	**0.310**	0.323	0.347	0.376	0.368	0.355	0.372
	Expand1	**0.615**	0.637	0.676	0.716	0.695	0.641	0.699
	Expand2	**0.635**	0.641	0.666	0.720	0.691	0.638	0.690
	V-strip	**0.672**	0.676	0.711	0.760	0.721	0.674	0.725
	H-strip	0.679	0.686	0.722	0.766	0.726	**0.674**	0.724
Average		**0.474**	0.481	0.518	0.596	0.591	0.489	0.571

Qualitative results on high-resolution image datasets



Outline

1. What are probabilistic circuits?

   tractable deep generative models

2. What are they useful for?

   controlling generative AI

3. **What is the underlying theory?**

   probability generating polynomials

Probabilistic circuits seem awfully general.

Are all tractable probabilistic models probabilistic circuits?

Enter: Determinantal Point Processes (DPPs)

DPPs are models where probabilities are specified by (sub)determinants

$$L = \begin{bmatrix} 1 & 0.9 & 0.8 & 0 \\ 0.9 & 0.97 & 0.96 & 0 \\ 0.8 & 0.96 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

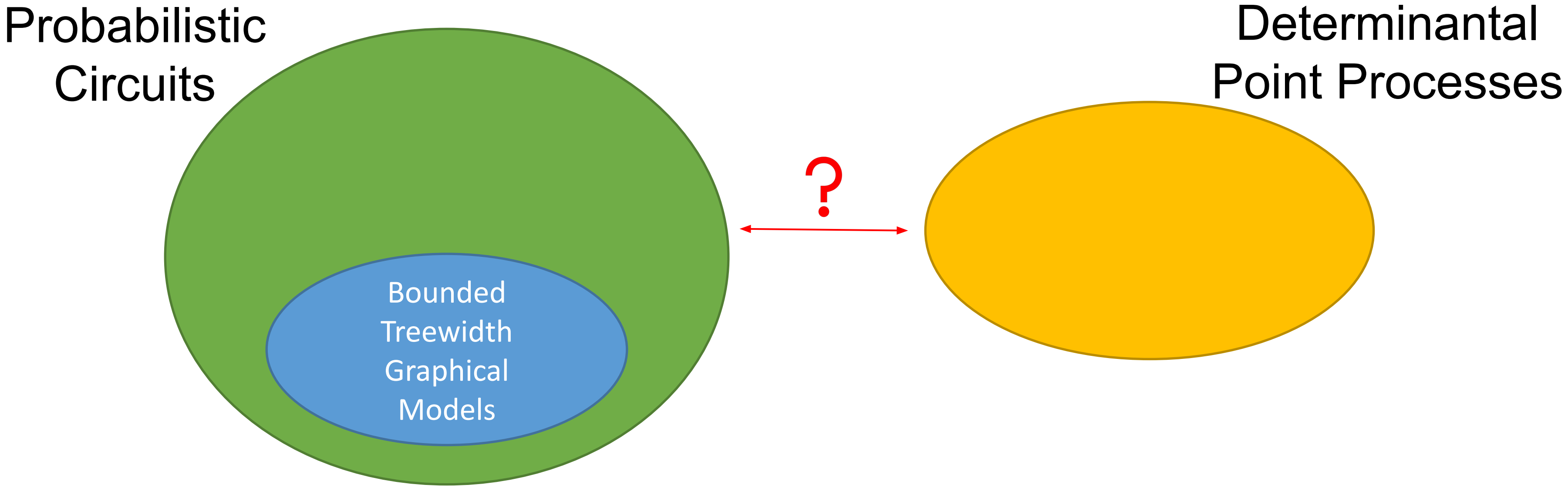
Global Negative Dependence

Diversity in recommendation systems

Tractable likelihoods and marginals

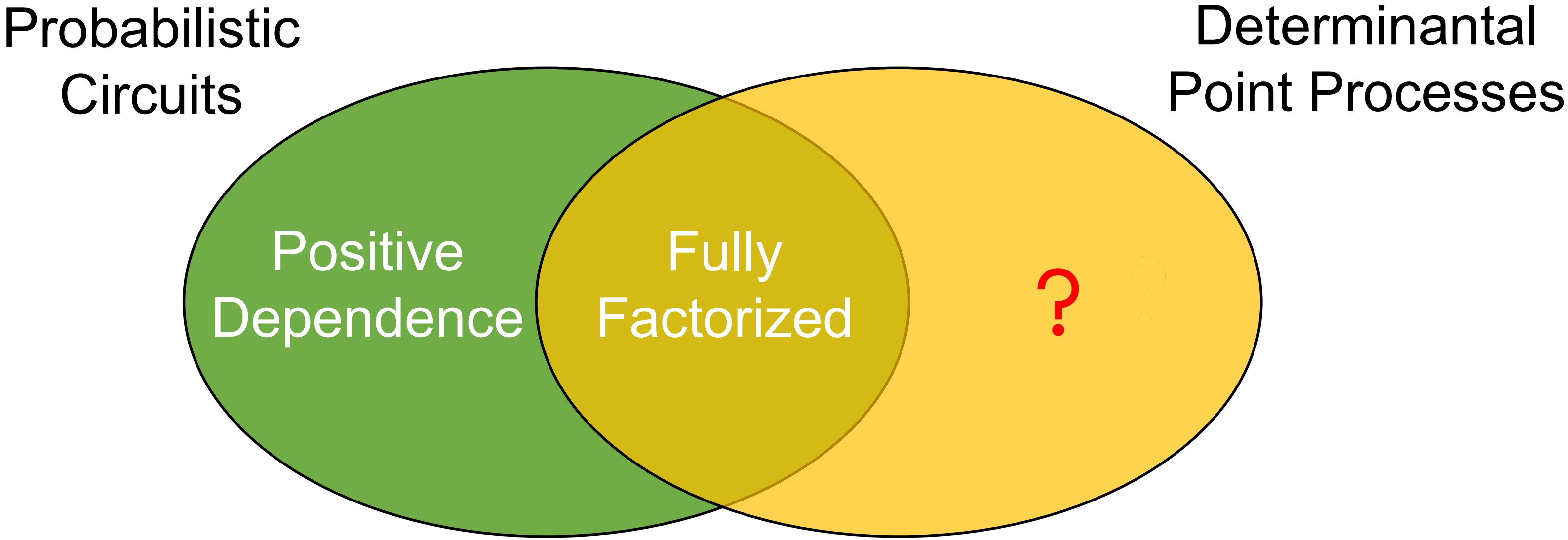
$$\Pr_L(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 0) = \frac{1}{\det(L + I)} \det(L_{\{1,2\}})$$

Are all tractable probabilistic models probabilistic circuits?



Honghua Zhang, Steven Holtzen and Guy Van den Broeck. On the Relationship Between Probabilistic Circuits and Determinantal Point Processes, *UAI*, 2020.

Are all tractable probabilistic models probabilistic circuits?

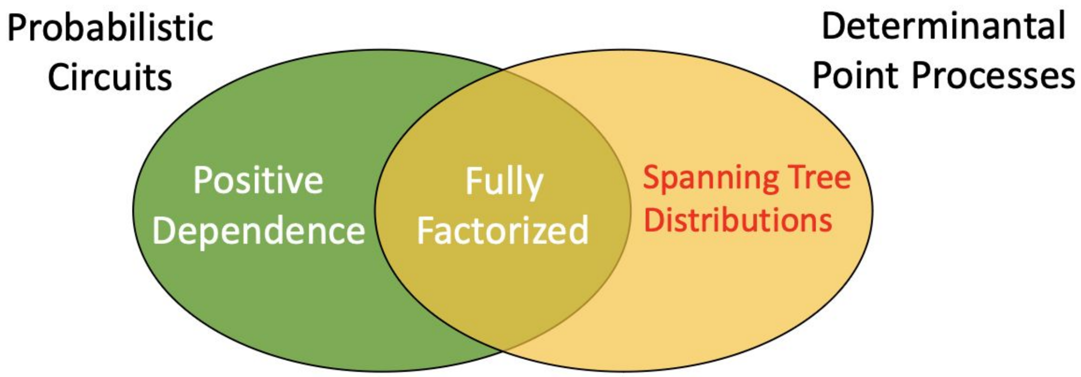


Honghua Zhang, Steven Holtzen and Guy Van den Broeck. On the Relationship Between Probabilistic Circuits and Determinantal Point Processes, *UAI*, 2020.

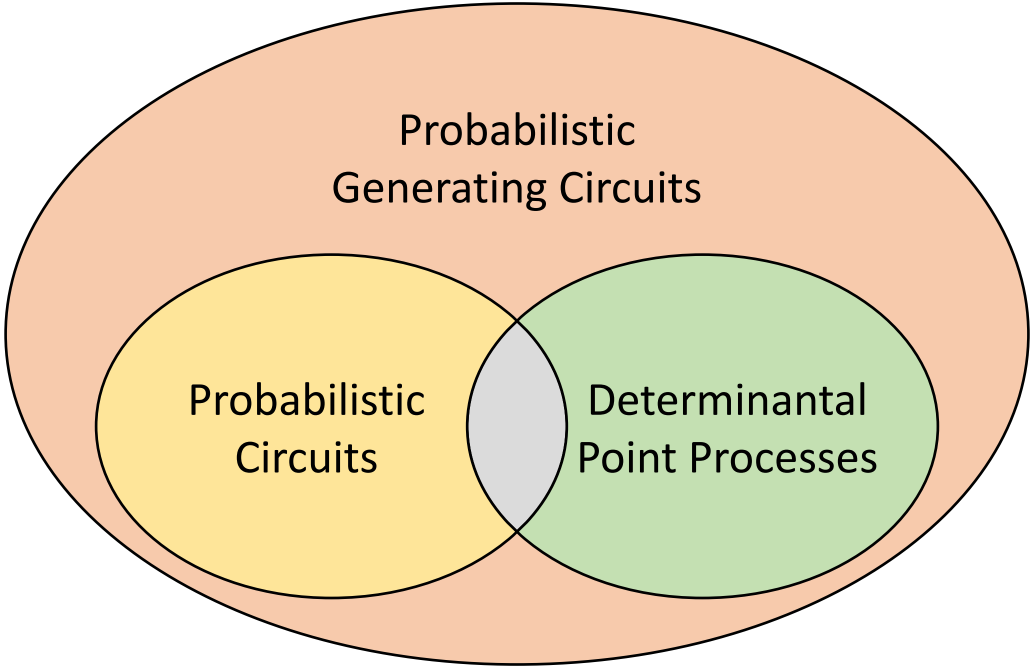
A separation between PCs and DPPs

Theorem (Martens and Medabalimi, 2014). *Let P_n be the uniform distribution over spanning trees on K_n. For $n \geq 20$, the size of any smooth and decomposable PC that represents P_n is at least $2^{n/30240}$.*

Theorem (Snell, 1995). *The uniform distribution over spanning trees on the complete graph K_n is a DPP over $\binom{n}{2}$ edges.*



Probabilistic Generating Circuits



A Tractable Unifying Framework for PCs and DPPs

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Probability Generating Functions

X_1	X_2	X_3	$\Pr_\beta$
0	0	0	0.02
0	0	1	0.08
0	1	0	0.12
0	1	1	0.48
1	0	0	0.02
1	0	1	0.08
1	1	0	0.04
1	1	1	0.16

$$g_\beta = 0.16 z_1 z_2 z_3 + 0.04 z_1 z_2 + 0.08 z_1 z_3 + 0.02 z_1 + 0.48 z_2 z_3 + 0.12 z_2 + 0.08 z_3 + 0.02.$$

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Probability Generating Functions

X_1	X_2	X_3	$\Pr_\beta$
0	0	0	0.02
0	0	1	0.08
0	1	0	0.12
0	1	1	0.48
1	0	0	0.02
1	0	1	0.08
1	1	0	0.04
1	1	1	0.16

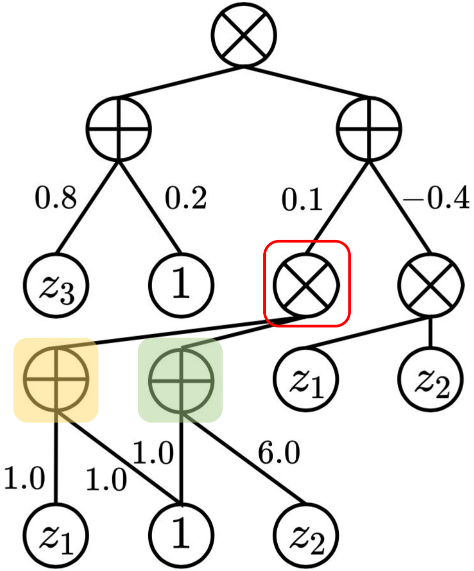
$$g_\beta = 0.16 z_1 z_2 z_3 + 0.04 z_1 z_2 + 0.08 z_1 z_3 + 0.02 z_1 + 0.48 z_2 z_3 + 0.12 z_2 + 0.08 z_3 + 0.02.$$

$$g_\beta = (0.1(z_1+1)(6z_2+1) - 0.4 z_1 z_2)(0.8 z_3 + 0.2)$$

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Probabilistic Generating Circuits (PGCs)

$$g_\beta = (0.1(z_1 + 1)(6z_2 + 1) - 0.4z_1z_2)(0.8z_3 + 0.2)$$

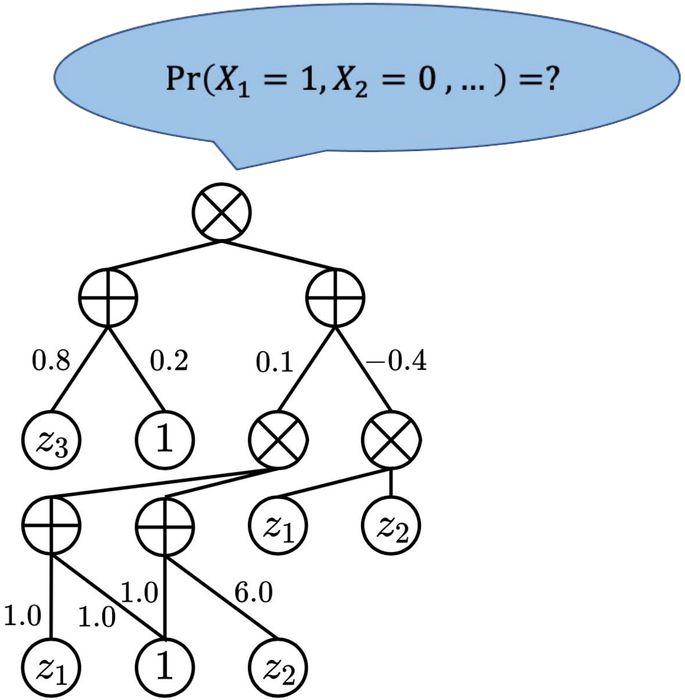


1. Sum nodes ⊕ with weighted edges to children.
2. Product nodes ⊗ with unweighted edges to children.
3. Leaf nodes: z_i or constant.

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

PGCs Support Tractable **Likelihoods**

How to extract the right monomial's coefficient?



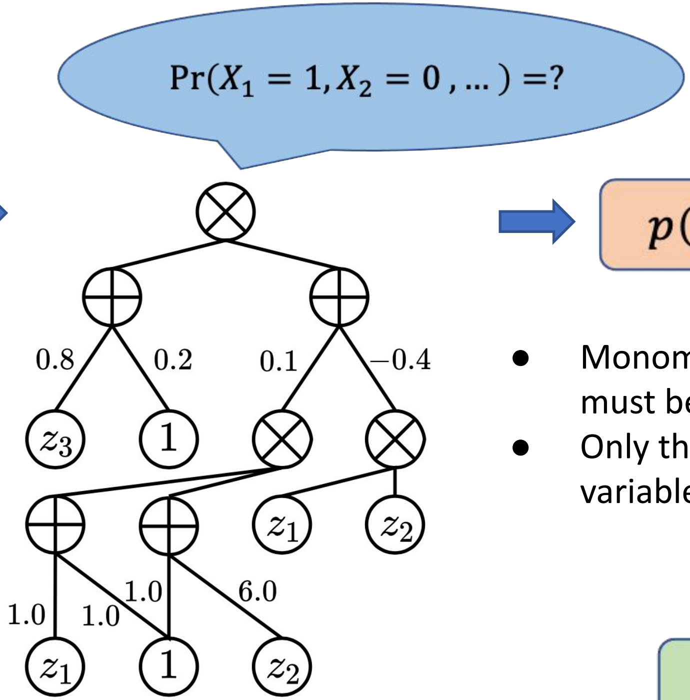
Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

PGCs Support Tractable **Likelihoods**

How to extract the right monomial's coefficient?

Purely symbolic

$$z_i = \begin{cases} t & \text{if } X_i = 1 \\ 0 & \text{if } X_i = 0 \end{cases}$$



complexity

O(circuit size x degree)

$$p(t) = \alpha_k t^k + \cdots + \alpha_1 t$$

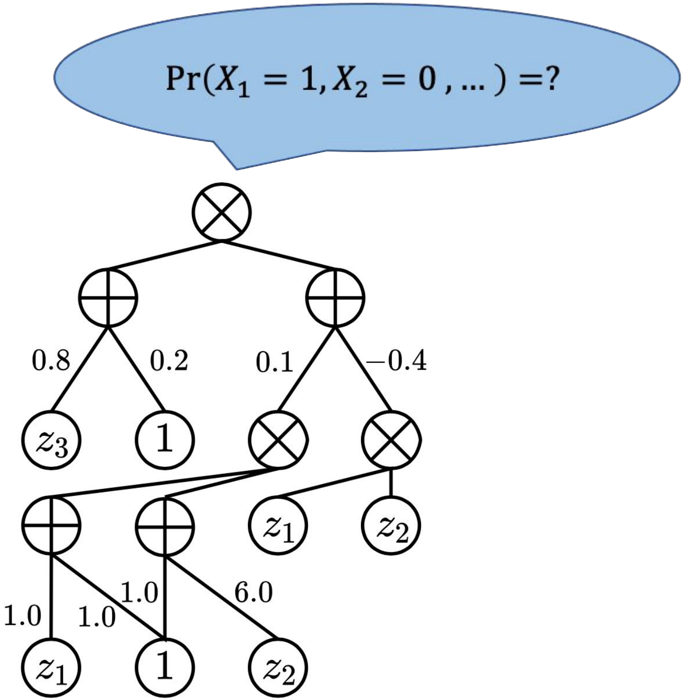
- Monomials setting to true variables that must be false are 0-ed out
- Only the monomial that sets all required variables to true has max degree.

α_k **gives the answer**

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021 & Harviainen et al., UAI 2023

PGCs Support Tractable **Marginals**

How to sum the right monomial's coefficients?



Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

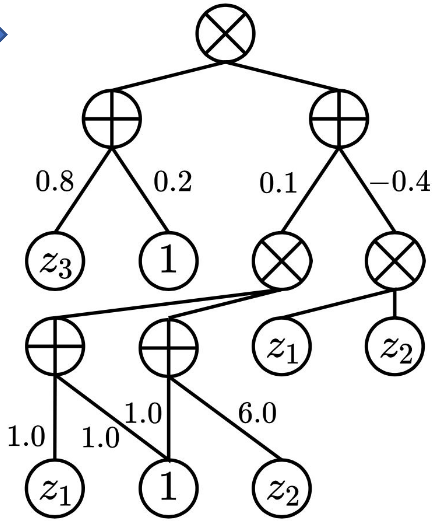
PGCs Support Tractable **Marginals**

How to sum the right monomial's coefficients?

Purely symbolic

$$z_i = \begin{cases} t & \text{if } X_i = 1 \\ 0 & \text{if } X_i = 0 \\ 1 & \text{if } X_i = ? \end{cases}$$

$\Pr(X_1 = 1, X_2 = 0, \dots) = ?$



$$p(t) = \alpha_k t^k + \cdots + \alpha_1 t$$

- Monomials setting to true variables that must be false are 0-ed out
- Other monomials contribute to result.
- Only monomials that set all required variables to true have max degree.

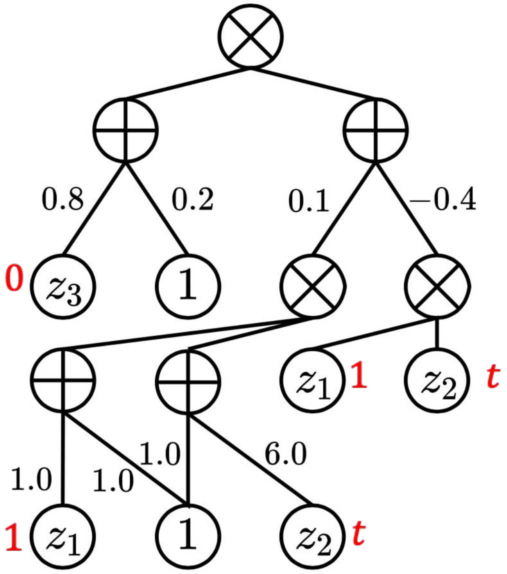
α_k gives the answer

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Example

$$z_i = \begin{cases} t & \text{if } X_i = 1 \\ 0 & \text{if } X_i = 0 \\ 1 & \text{if } X_i = ? \end{cases}$$

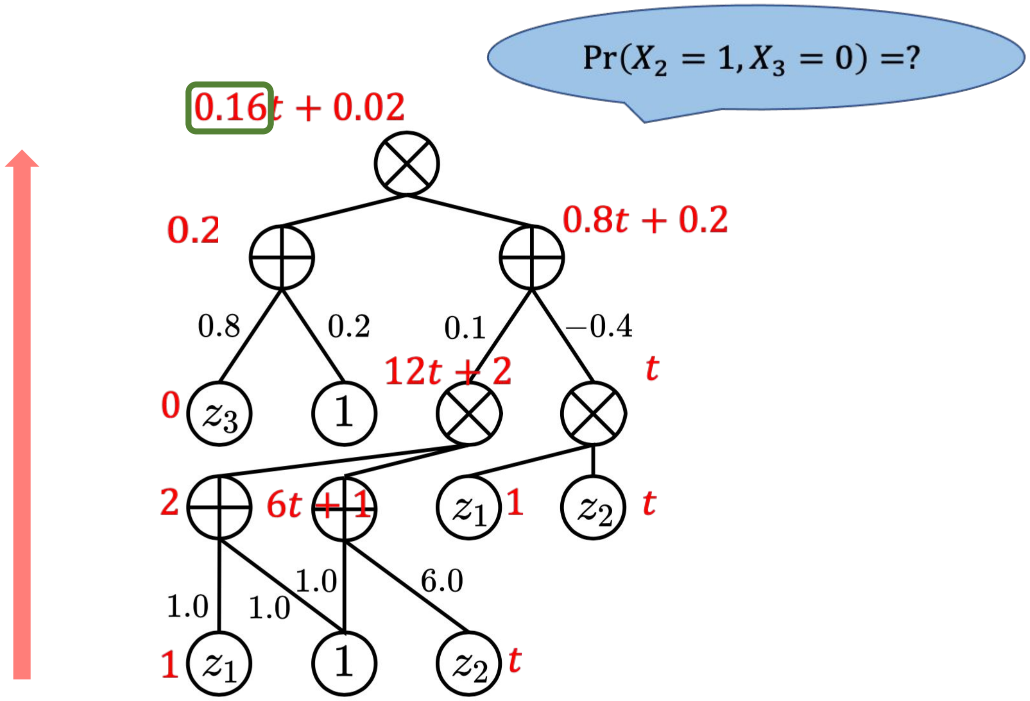
$\Pr(X_2 = 1, X_3 = 0) =?$



Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Example

$$z_i = \begin{cases} t & \text{if } X_i = 1 \\ 0 & \text{if } X_i = 0 \\ 1 & \text{if } X_i = ? \end{cases}$$

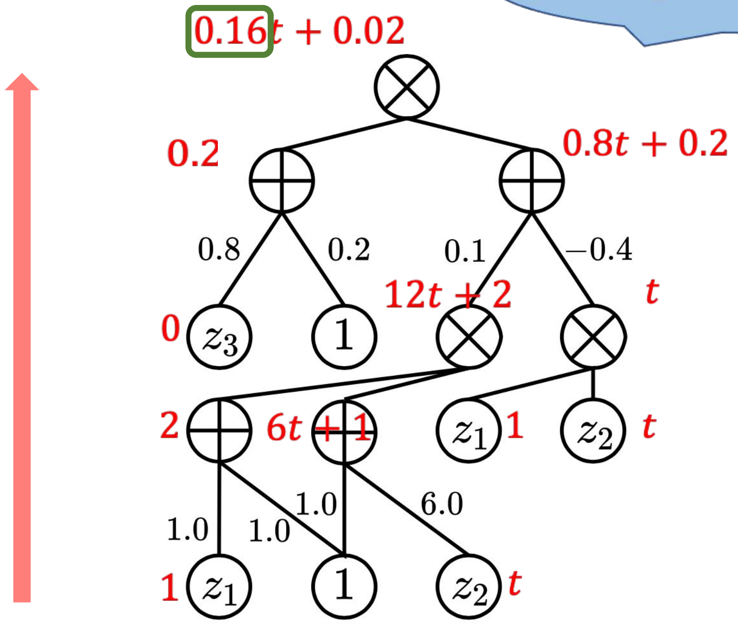


Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Example

$$z_i = \begin{cases} t & \text{if } X_i = 1 \\ 0 & \text{if } X_i = 0 \\ 1 & \text{if } X_i = ? \end{cases}$$

$\Pr(X_2 = 1, X_3 = 0) = ?$



X_1	X_2	X_3	$\Pr_\beta$
0	0	0	0.02
0	0	1	0.08
0	1	0	0.12
0	1	1	0.48
1	0	0	0.02
1	0	1	0.08
1	1	0	0.04
1	1	1	0.16

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Probabilistic circuits are probabilistic generating circuits

PCs represents probability mass functions:

$$m_\beta = 0.16X_1X_2X_3 + 0.04X_1X_2\overline{X_3} + 0.08X_1\overline{X_2}X_3 + 0.02X_1\overline{X_2}\overline{X_3}$$
$$+ 0.48\overline{X_1}X_2X_3 + 0.12\overline{X_1}X_2\overline{X_3} + 0.08\overline{X_1}\overline{X_2}X_3 + 0.02\overline{X_1}\overline{X_2}\overline{X_3}$$

PGCs represent probability generating functions:

$$g_\beta = 0.16z_1z_2z_3 + 0.04z_1z_2 + 0.08z_1z_3 + 0.02z_1$$
$$+ 0.48z_2z_3 + 0.12z_2 + 0.08z_1z_3 + 0.02$$

Given a smooth & decomposable PC, by setting $\overline{X_i}$ to 1, and X_i to z_i,
we obtain an equivalent PGC

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

DPPs are probabilistic generating circuits

The generating polynomial for a DPP with kernel L is given by:

$$g_L = \frac{1}{\det(L+I)} \det(I + L\mathrm{diag}(z_1, \ldots, z_n)).$$

We need it as a sum of products to obtain a Probabilistic Generating Circuit

Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

DPPs are probabilistic generating circuits

The generating polynomial for a DPP with kernel L is given by:

$$g_L = \frac{1}{\det(L+I)} \det(I + L\text{diag}(z_1, \dots, z_n)).$$

Constant

We need it as a sum of products to obtain a Probabilistic Generating Circuit

DPPs are probabilistic generating circuits

The generating polynomial for a DPP with kernel L is given by:

$$g_L = \frac{1}{\det(L+I)} \det(I + L\mathrm{diag}(z_1, \ldots, z_n)).$$

Constant

Division-free determinant algorithm
(Samuelson-Berkowitz algorithm)

g_L can be represented as a PGC of size $O(n^4)$

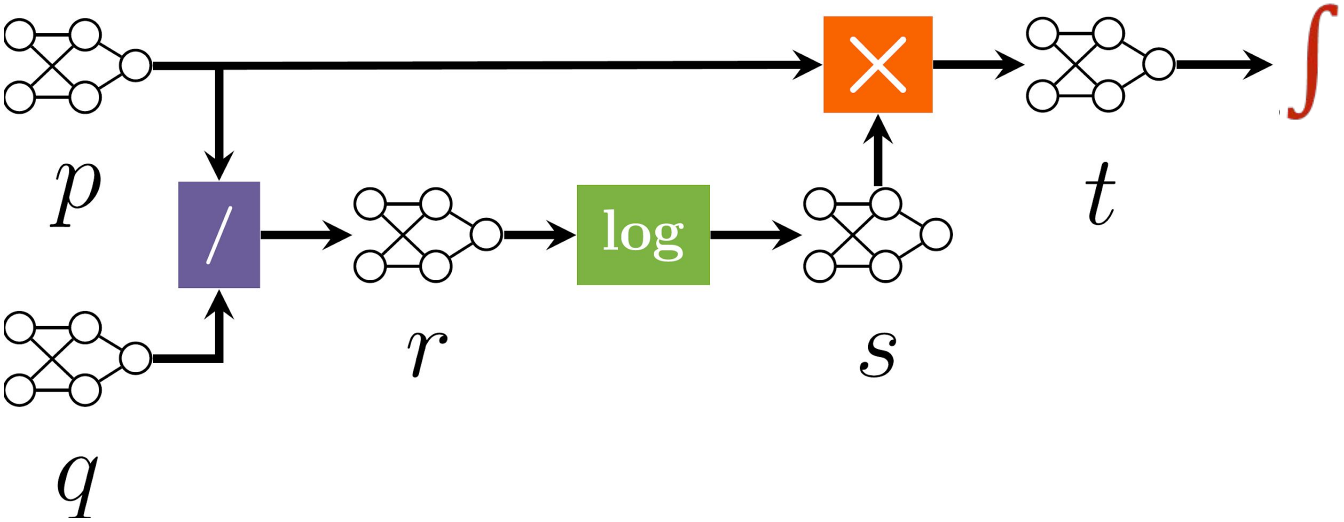
Honghua Zhang, Brendan Juba and Guy Van den Broeck. Probabilistic Generating Circuits, *ICML*, 2021.

Probabilistic **generating** circuits seem awfully general.

*Are all tractable probabilistic models probabilistic **generating** circuits?*

Queries as pipelines: KLD

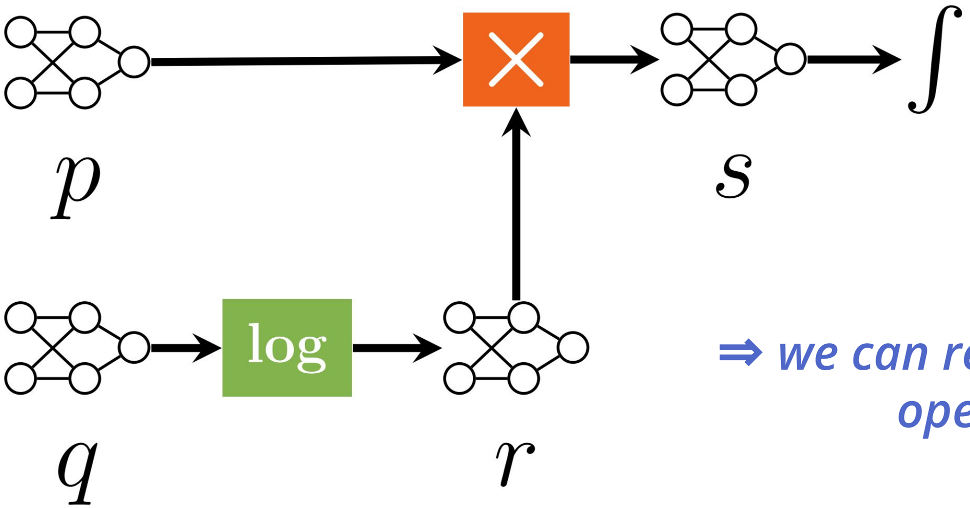
$$\mathbb{KLD}(p \parallel q) = \int p(\boldsymbol{x}) \times \log((p(\boldsymbol{x})/q(\boldsymbol{x}))d\boldsymbol{X}$$



Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference, *NeurIPS*, 2021.

Queries as pipelines: Cross Entropy

$$H(p,q) = \int p(\boldsymbol{x}) \times \log(q(\boldsymbol{x}))d\boldsymbol{X}$$

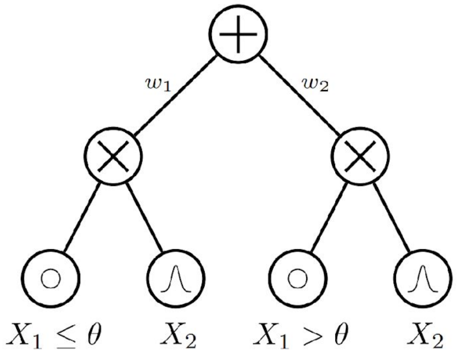


⇒ *we can reuse the operations!*

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference, *NeurIPS*, 2021.

Determinism

A sum node is ***deterministic*** if only one of its children outputs non-zero for any input



deterministic circuit

⇒ *allows **tractable MAP** inference*

$$argmax_{\boldsymbol{x}}\, p(\boldsymbol{x})$$

Darwiche and Marquis, "A Knowledge Compilation Map", 2002

Operation		Tractability	
		Input conditions	Output conditions
LOG	$\log(p)$	Sm, Dec, Det	Sm, Dec

smooth, decomposable, deterministic

smooth, decomposable

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference, *NeurIPS*, 2021.

Tractable circuit operations

Operation		Tractability: Input properties	Tractability: Output properties	Hardness
SUM	$\theta_1 p + \theta_2 q$	(+Cmp)	(+SD)	NP-hard for Det output
PRODUCT	$p \cdot q$	Cmp (+Det, +SD)	Dec (+Det, +SD)	#P-hard w/o Cmp
POWER	$p^n, n \in \mathbb{N}$	SD (+Det)	SD (+Det)	#P-hard w/o SD
	$p^\alpha, \alpha \in \mathbb{R}$	Sm, Dec, Det (+SD)	Sm, Dec, Det (+SD)	#P-hard w/o Det
QUOTIENT	p/q	Cmp; q Det (+p Det,+SD)	Dec (+Det,+SD)	#P-hard w/o Det
LOG	$\log(p)$	Sm, Dec, Det	Sm, Dec	#P-hard w/o Det
EXP	$\exp(p)$	linear	SD	#P-hard

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference, *NeurIPS*, 2021.

Inference by tractable operations

systematically derive tractable inference algorithm of complex queries

	Query	Tract. Conditions	Hardness
CROSS ENTROPY	$-\int p(\boldsymbol{x}) \log q(\boldsymbol{x})\, \mathrm{d}\mathbf{X}$	Cmp, q Det	#P-hard w/o Det
SHANNON ENTROPY	$-\sum p(\boldsymbol{x}) \log p(\boldsymbol{x})$	Sm, Dec, Det	coNP-hard w/o Det
RÉNYI ENTROPY	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x})\, d\mathbf{X}, \alpha \in \mathbb{N}$	SD	#P-hard w/o SD
	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x})\, d\mathbf{X}, \alpha \in \mathbb{R}_{+}$	Sm, Dec, Det	#P-hard w/o Det
MUTUAL INFORMATION	$\int p(\boldsymbol{x},\boldsymbol{y}) \log(p(\boldsymbol{x},\boldsymbol{y})/(p(\boldsymbol{x})p(\boldsymbol{y})))$	Sm, SD, Det*	coNP-hard w/o SD
KULLBACK-LEIBLER DIV.	$\int p(\boldsymbol{x}) \log(p(\boldsymbol{x})/q(\boldsymbol{x})) d\mathbf{X}$	Cmp, Det	#P-hard w/o Det
RÉNYI'S ALPHA DIV.	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x}) q^{1-\alpha}(\boldsymbol{x})\, d\mathbf{X}, \alpha \in \mathbb{N}$	Cmp, q Det	#P-hard w/o Det
	$(1-\alpha)^{-1} \log \int p^{\alpha}(\boldsymbol{x}) q^{1-\alpha}(\boldsymbol{x})\, d\mathbf{X}, \alpha \in \mathbb{R}$	Cmp, Det	#P-hard w/o Det
ITAKURA-SAITO DIV.	$\int [p(\boldsymbol{x})/q(\boldsymbol{x}) - \log(p(\boldsymbol{x})/q(\boldsymbol{x})) - 1]\, d\,\mathbf{X}$	Cmp, Det	#P-hard w/o Det
CAUCHY-SCHWARZ DIV.	$-\log \frac{\int p(\boldsymbol{x}) q(\boldsymbol{x}) d\mathbf{X}}{\sqrt{\int p^2(\boldsymbol{x}) d\mathbf{X} \int q^2(\boldsymbol{x}) d\mathbf{X}}}$	Cmp	#P-hard w/o Cmp
SQUARED LOSS	$\int (p(\boldsymbol{x}) - q(\boldsymbol{x}))^2 d\,\mathbf{X}$	Cmp	#P-hard w/o Cmp

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference, *NeurIPS*, 2021.

Conclusions

1. What are probabilistic circuits?

   tractable deep generative models

2. What are they useful for?

   controlling generative AI

3. What is the underlying theory?

   probability generating polynomials

Thanks

This was the work of many wonderful students/postdocs/collaborators!

References: http://starai.cs.ucla.edu/publications/