StARLinG LAB
THE UNIVERSITY OF TEXAS AT DALLAS

39th Conference on Uncertainty in Artificial Intelligence
Pittsburgh, PA, USA
31st July – 4th August, 2023
uai2023

TECHNISCHE UNIVERSITÄT DARMSTADT

Probabilistic Flow Circuits: Towards Unified Deep Models for Tractable Probabilistic Inference

Sahil Sidheekh

UTD

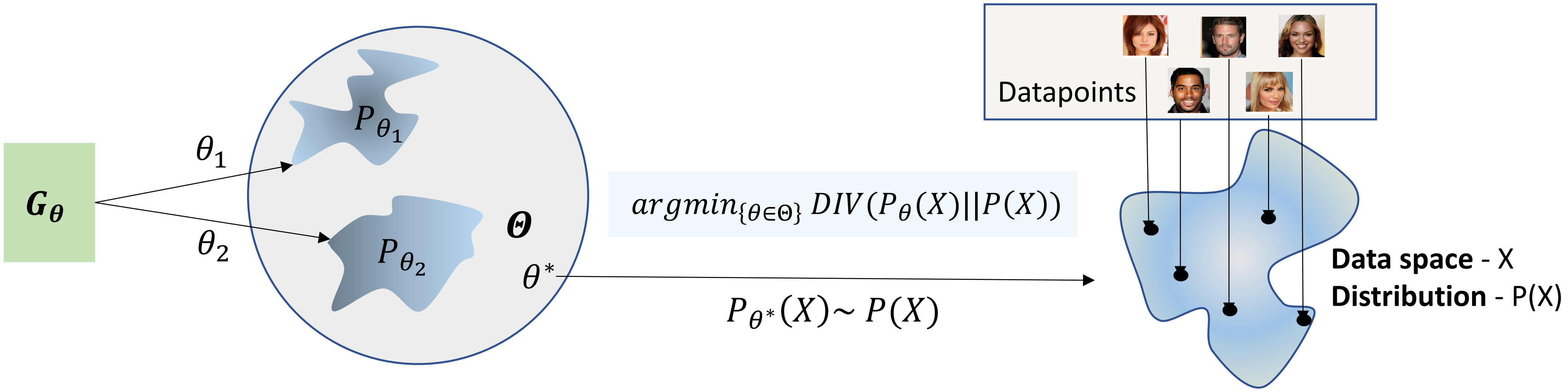
Kristian Kersting

Sriraam Natarajan

UTD

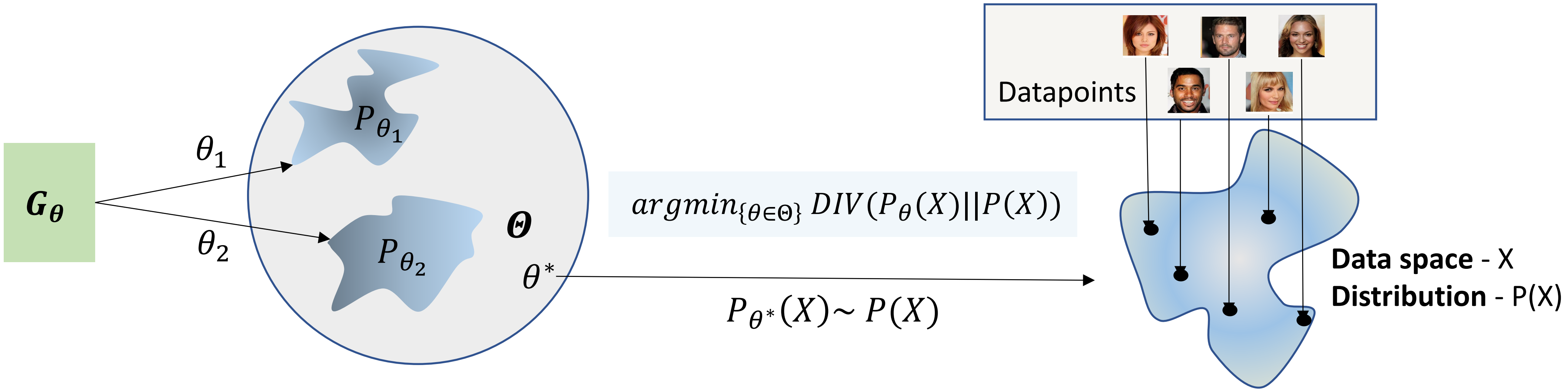
Motivation

Building Expressive and Tractable Generative Models



Motivation

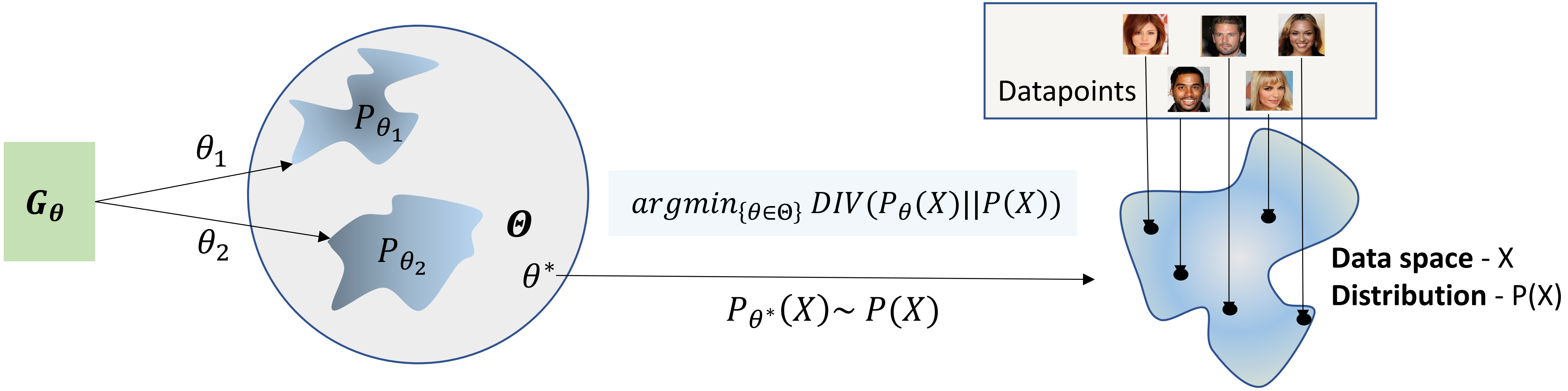
Building Expressive and Tractable Generative Models



Expressivity	Characterizes the complexity of functions that can be represented. A more expressive model can better approximate complex distributions in high-dimensional spaces

Motivation

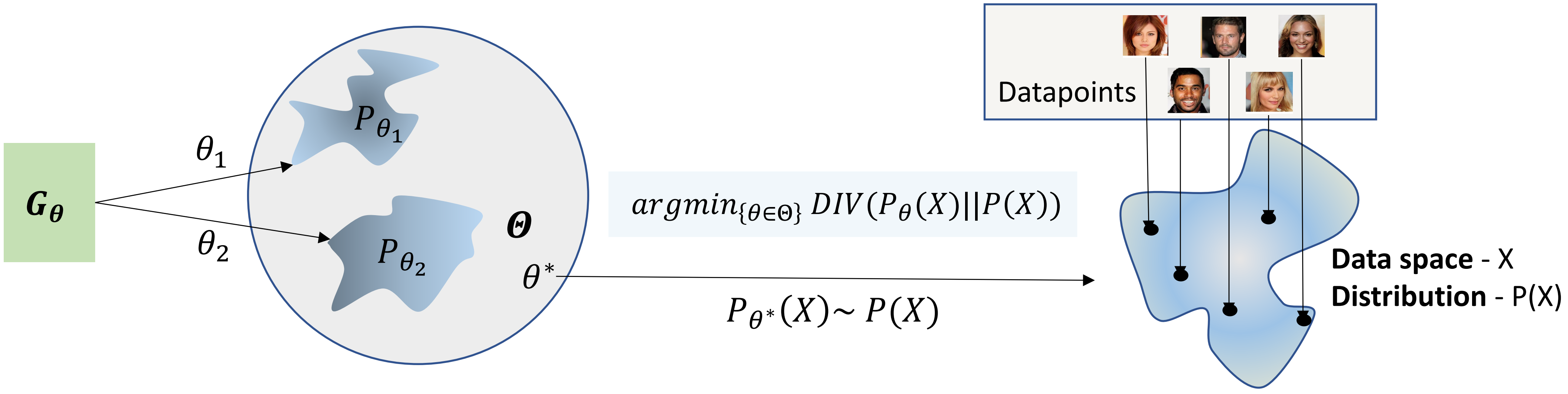
Building Expressive and Tractable Generative Models



Expressivity	Characterizes the complexity of functions that can be represented. A more expressive model can better approximate complex distributions in high-dimensional spaces
Tractability	Ability to answer probabilistic **inference** queries about the learned distribution in **polynomial** time. Can **reason** probabilistically about the learned distribution.

Motivation

Building Expressive and Tractable Generative Models



Using Normalizing Flows

Expressivity

Characterizes the complexity of functions that can be represented.

A more expressive model can better approximate complex distributions in high-dimensional spaces

Using Probabilistic Circuits

Tractability

Ability to answer probabilistic **inference** queries about the learned distribution in **polynomial** time.

Can **reason** probabilistically about the learned distribution.

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Computational graphs that recursively define distributions via 3 types of nodes

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Computational graphs that recursively define distributions via 3 types of nodes

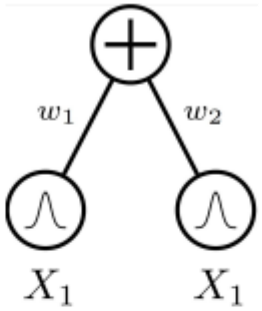
$p(X) = Normal(X)$

Leaf nodes
Simple univariate distributions

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Computational graphs that recursively define distributions via 3 types of nodes



$p(X) = Normal(X)$

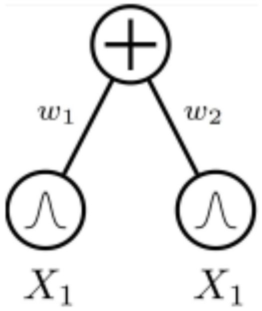
$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)$

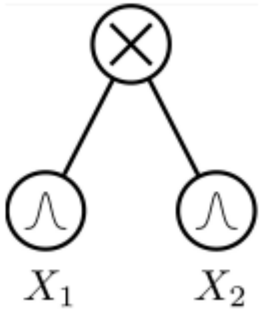
Leaf nodes	Sum nodes
Simple univariate distributions	Represents **mixtures**
	Adds expressivity

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Computational graphs that recursively define distributions via 3 types of nodes





$p(X) = Normal(X)$

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

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

Leaf nodes	Sum nodes	Product nodes
Simple univariate distributions	Represents **mixtures**	Represents **factorizations**
	Adds expressivity	Enables tractability

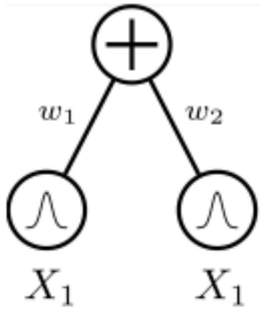
Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Computational graphs that recursively define distributions via 3 types of nodes

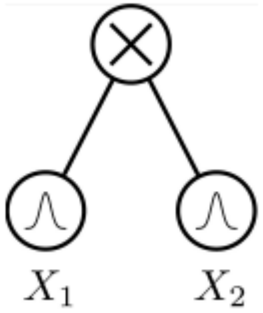
$p(X) = Normal(X)$

Leaf nodes
Simple univariate distributions



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

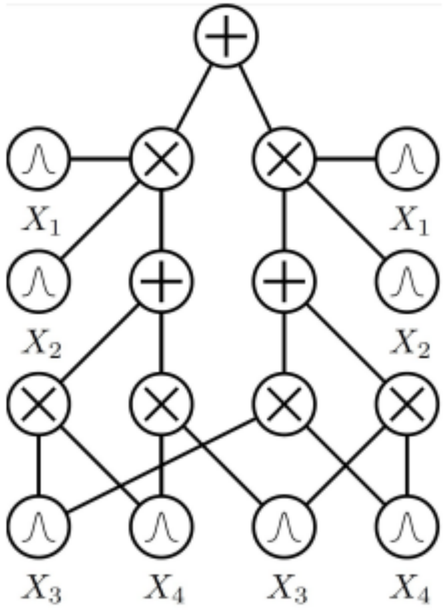
Sum nodes
Represents **mixtures**
Adds expressivity



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

Product nodes
Represents **factorizations**
Enables tractability

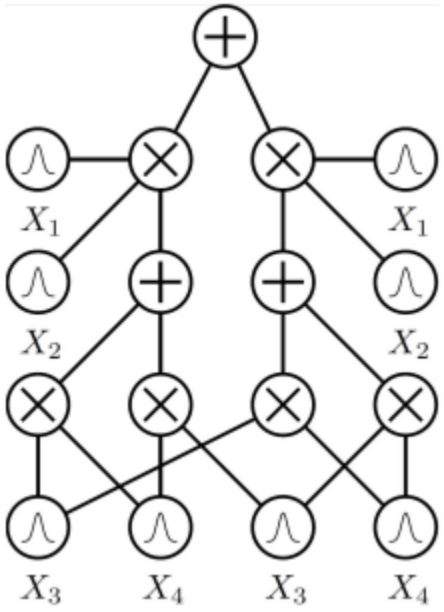
Stacked as a circuit



Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties

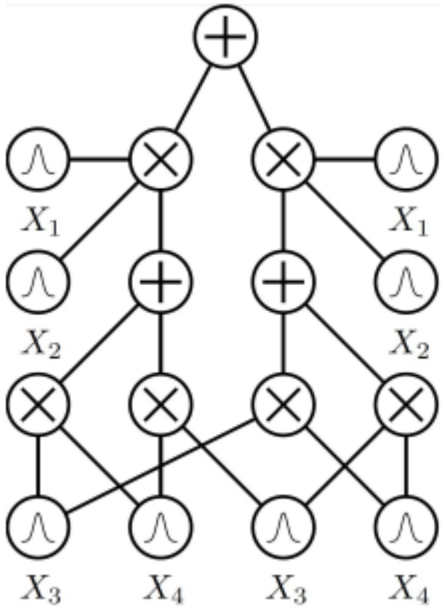


Inference Queries

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties



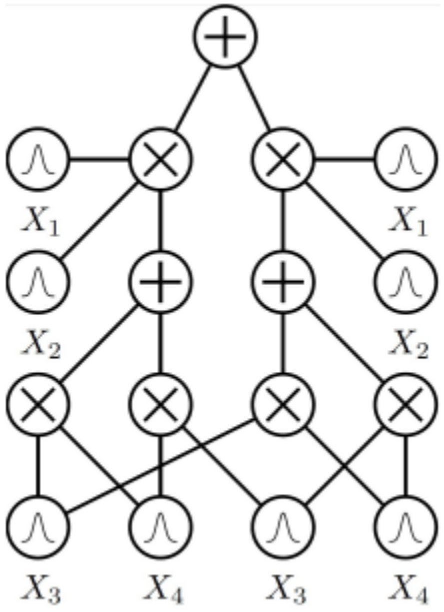
Inference Queries

Evidential Inference: $p(X_1, X_2, X_3, X_4)$

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties



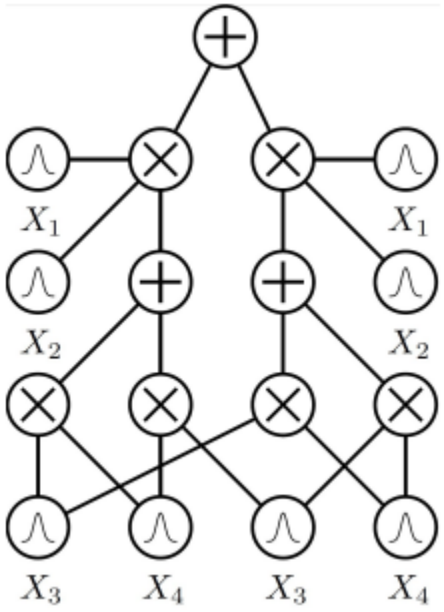
Inference Queries

Evidential Inference:	$p(X_1, X_2, X_3, X_4)$
Marginal Inference:	$p(X_1)$

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties



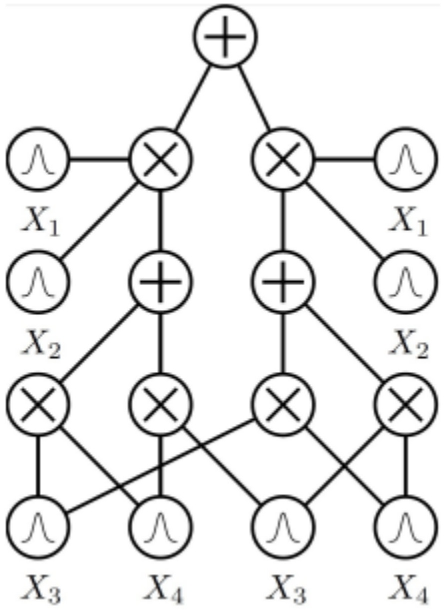
Inference Queries

Evidential Inference:	$p(X_1, X_2, X_3, X_4)$
Marginal Inference:	$p(X_1)$
Conditional Inference:	$p(X_1 \mid X_2, X_3, X_4)$

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties



Inference Queries

Evidential Inference:	$p(X_1, X_2, X_3, X_4)$
Marginal Inference:	$p(X_1)$
Conditional Inference:	$p(X_1 \| X_2, X_3, X_4)$
MAP Inference:	$\text{argmax}_{X_1}\ p(X_1 \| X_2, X_3, X_4)$

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties

Inference Queries

- Evidential Inference
- Marginal Inference
- Conditional Inference
- MAP Inference

Structural Properties

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties

Inference Queries

- Evidential Inference
- Marginal Inference
- Conditional Inference
- MAP Inference

Structural Properties

Smoothness	Children of sum nodes have same scope

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties

Inference Queries

- Evidential Inference
- Marginal Inference
- Conditional Inference
- MAP Inference

Structural Properties

Smoothness	Children of sum nodes have same scope
Decomposability	Children of product nodes have disjoint scope

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties

Inference Queries

- Evidential Inference
- Marginal Inference
- Conditional Inference
- MAP Inference

Structural Properties

Smoothness	Children of sum nodes have same scope
Decomposability	Children of product nodes have disjoint scope
Determinism	Children of sum nodes have disjoint support
…	

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties

Inference Queries

- Evidential Inference
- Marginal Inference
- Conditional Inference
- MAP Inference

Structural Properties

Smoothness	Children of sum nodes have same scope
Decomposability	Children of product nodes have disjoint scope
Determinism	Children of sum nodes have disjoint support
...	

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties

Inference Queries

- Evidential Inference
- Marginal Inference
- Conditional Inference
- MAP Inference

Structural Properties

- **Smoothness** Children of sum nodes have same scope
- **Decomposability** Children of product nodes have disjoint scope
- **Determinism** Children of sum nodes have disjoint support
- ...

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties

Inference Queries

- Evidential Inference
- Marginal Inference
- Conditional Inference
- MAP Inference

Structural Properties

Smoothness	Children of sum nodes have same scope
Decomposability	Children of product nodes have disjoint scope
Determinism	Children of sum nodes have disjoint support
…	

Probabilistic Circuits

Hierarchical Mixtures of Simple Factorized Distributions

Tractability for probabilistic inference is achieved via structural properties

Inference Queries

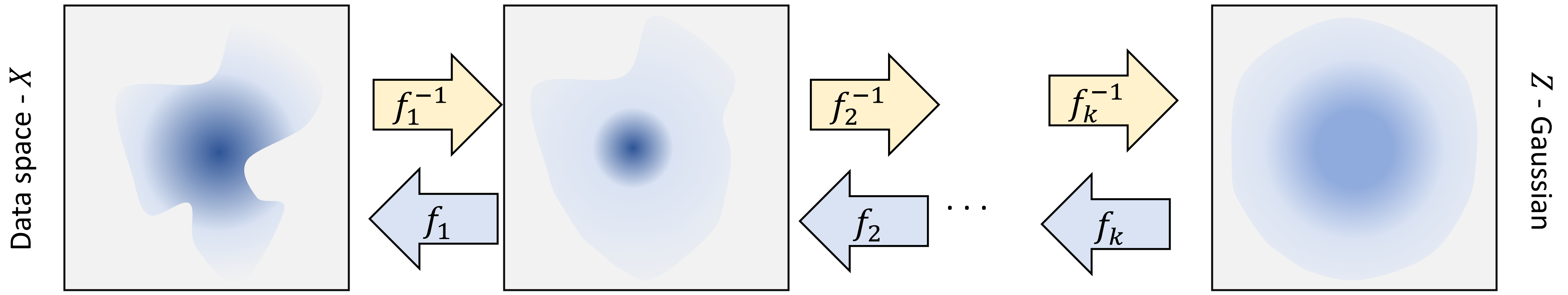
- Evidential Inference
- Marginal Inference
- Conditional Inference
- MAP Inference

Structural Properties

Smoothness	Children of sum nodes have same scope
Decomposability	Children of product nodes have disjoint scope
Determinism	Children of sum nodes have disjoint support
...	

Normalizing Flows

Model data distributions using Invertible transformations and the change of variables formula

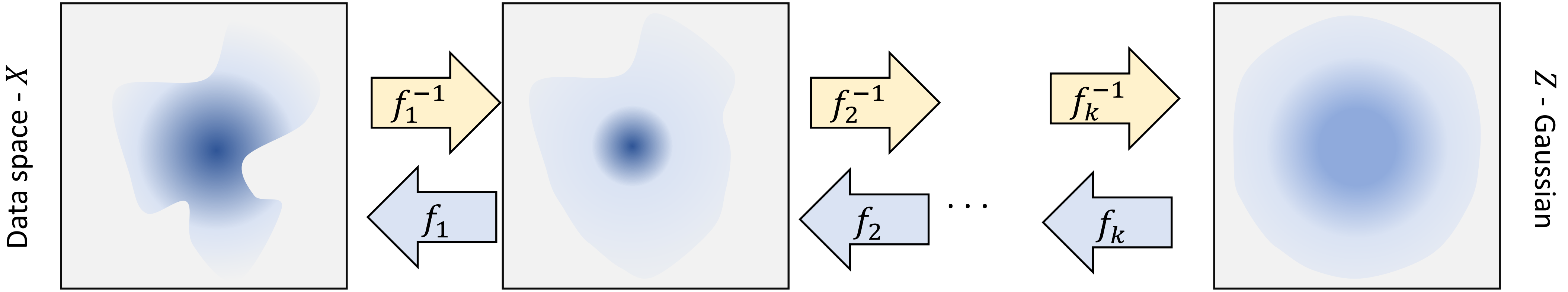


Normalizing Flows

Model data distributions using Invertible transformations and the change of variables formula

Change of Variables: Z and X be random variables which are related by a mapping $f : \mathbb{R}^n \to \mathbb{R}^n$ such that $X = f(Z)$ and $Z = f^{-1}(X)$. Then

$$p_X(\mathrm{x}) = p_Z(f^{-1}(\mathrm{x})) \left|\det\left(\frac{\partial f^{-1}(\mathrm{x})}{\partial \mathrm{x}}\right)\right|$$

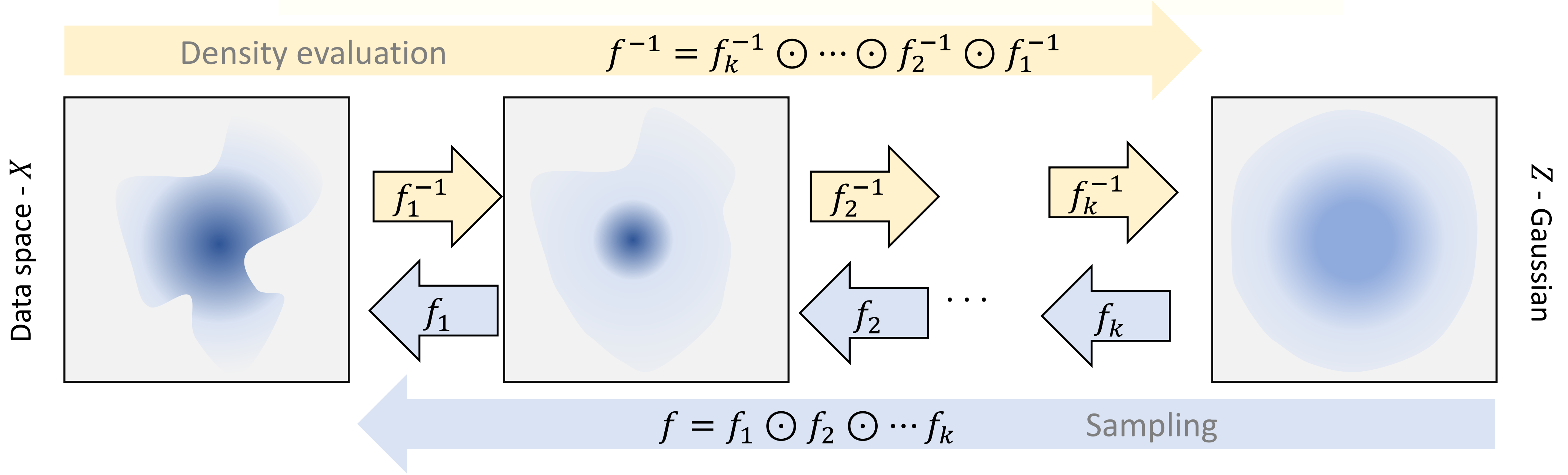


Normalizing Flows

Model data distributions using Invertible transformations and the change of variables formula

Change of Variables: Z and X be random variables which are related by a mapping $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ such that $X = f(Z)$ and $Z = f^{-1}(X)$. Then

$$p_X(\mathrm{x}) = p_Z(f^{-1}(\mathrm{x})) \left| \det \left(\frac{\partial f^{-1}(\mathrm{x})}{\partial \mathrm{x}} \right) \right|$$

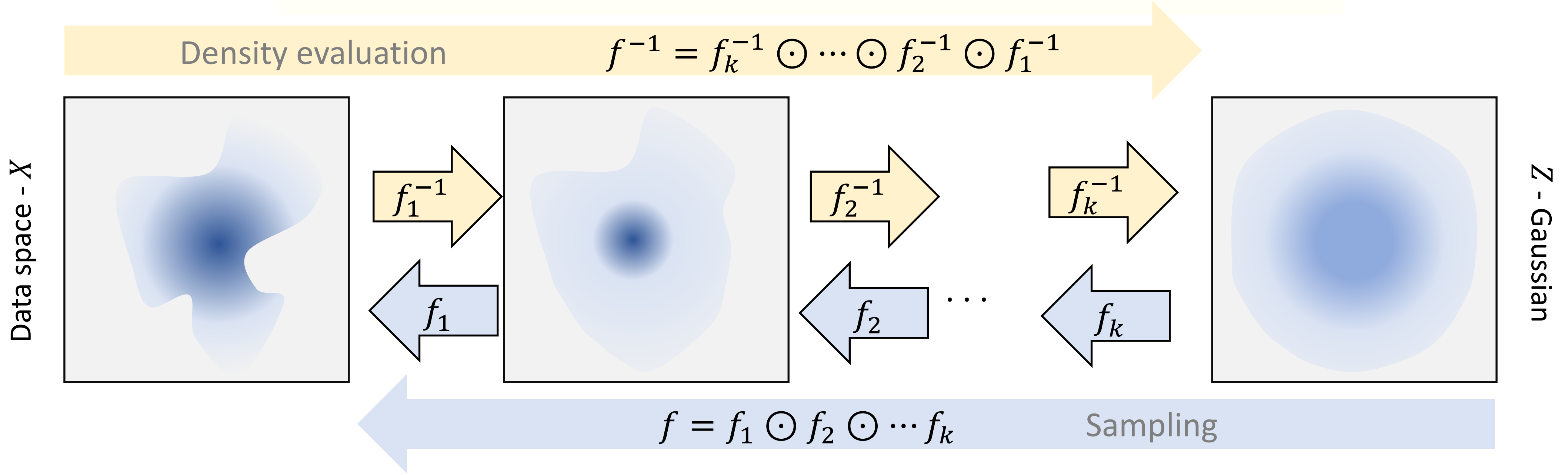


Normalizing Flows

Model data distributions using Invertible transformations and the change of variables formula

Change of Variables: Z and X be random variables which are related by a mapping $f : \mathbb{R}^n \to \mathbb{R}^n$ such that $X = f(Z)$ and $Z = f^{-1}(X)$. Then

$$p_X(\mathrm{x}) = p_Z(f^{-1}(\mathrm{x})) \left|\det\left(\frac{\partial f^{-1}(\mathrm{x})}{\partial \mathrm{x}}\right)\right|$$



Parameterize efficient invertible transformations using **neural** networks

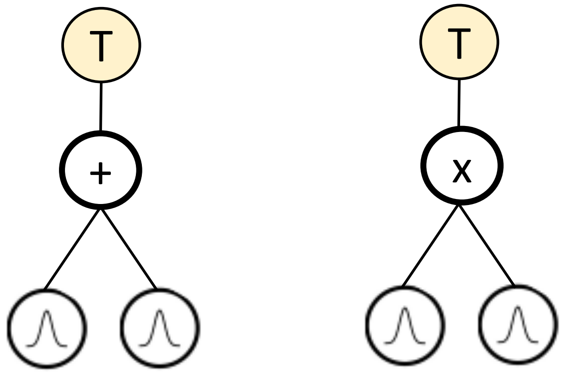
Supports **exact** density evaluation

More expressive than Probabilistic Circuits

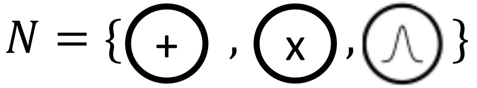
Integrating Flows with Probabilistic Circuits

Use the change of variables within PCs

Introduce new **transform nodes** in PCs



$$T(N(\boldsymbol{x})) = N(f(\boldsymbol{x}))|\det J_f|$$



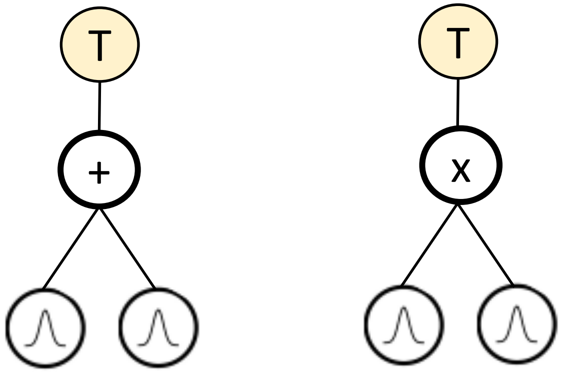
Transform nodes
Represents **normalizing** flows
Adds expressivity

Sum-Product Transform Network
(Pevny et. Al 2020)

Integrating Flows with Probabilistic Circuits

Use the change of variables within PCs

Introduce new **transform nodes** in PCs



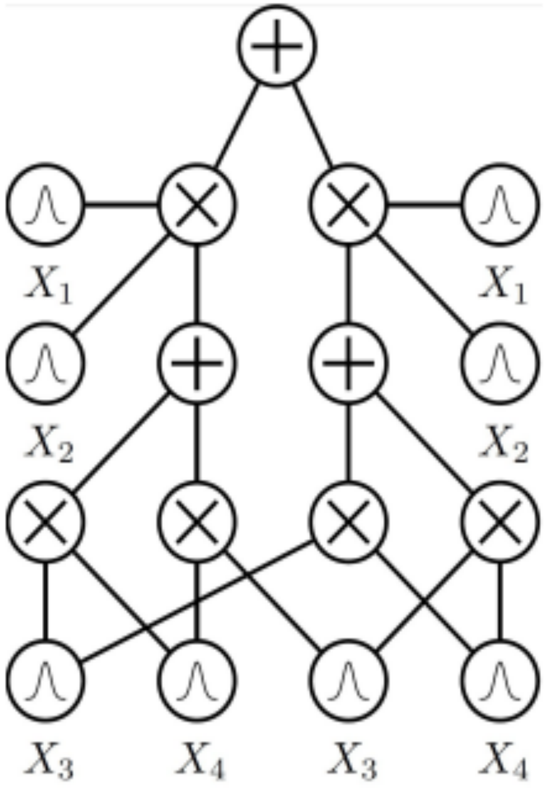
$$T(N(\boldsymbol{x})) = N(f(\boldsymbol{x}))|\det J_f|$$

$$N = \{+,\ \mathrm{x},\ \Lambda\}$$

Transform nodes
Represents **normalizing** flows
Adds expressivity

Sum-Product Transform Network
(Pevny et. Al 2020)

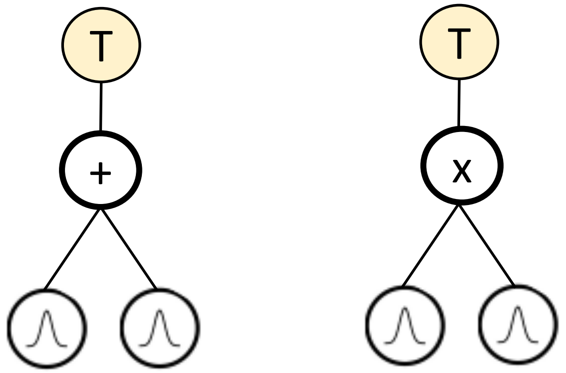
Probabilistic Circuit



Integrating Flows with Probabilistic Circuits

Use the change of variables within PCs

Introduce new **transform nodes** in PCs

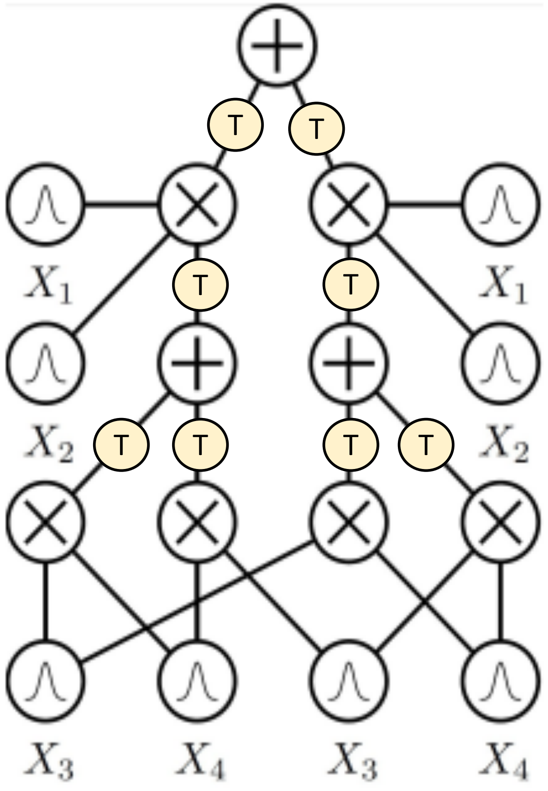


$$T(N(\boldsymbol{x})) = N(f(\boldsymbol{x}))|\det J_f|$$

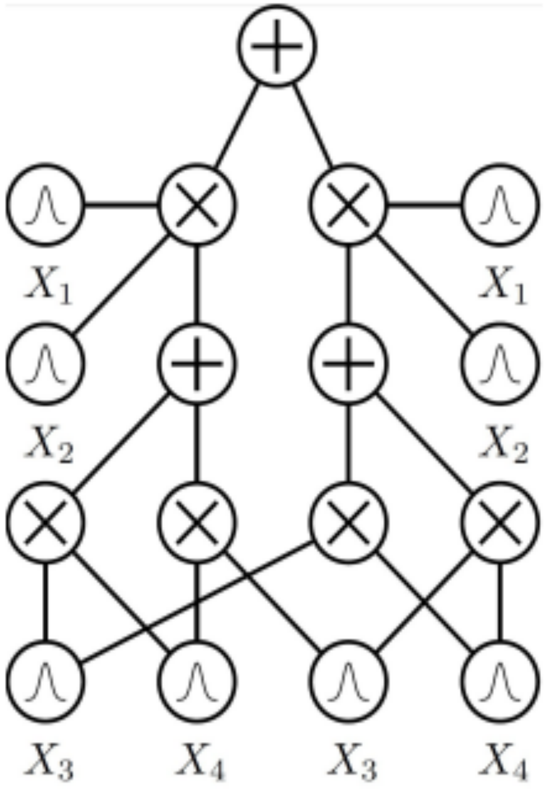
$$N = \{+, x, \Lambda\}$$

Transform nodes
Represents **normalizing** flows
Adds expressivity

Sum-Product Transform Network
(Pevny et. Al 2020)



Probabilistic Circuit

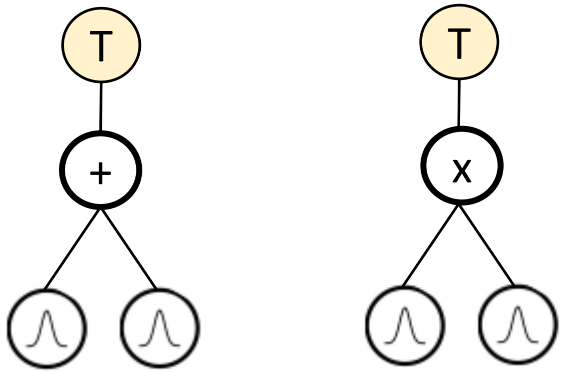


Place transform nodes arbitrarily in a PC

Integrating Flows with Probabilistic Circuits

Use the change of variables within PCs

Introduce new **transform nodes** in PCs

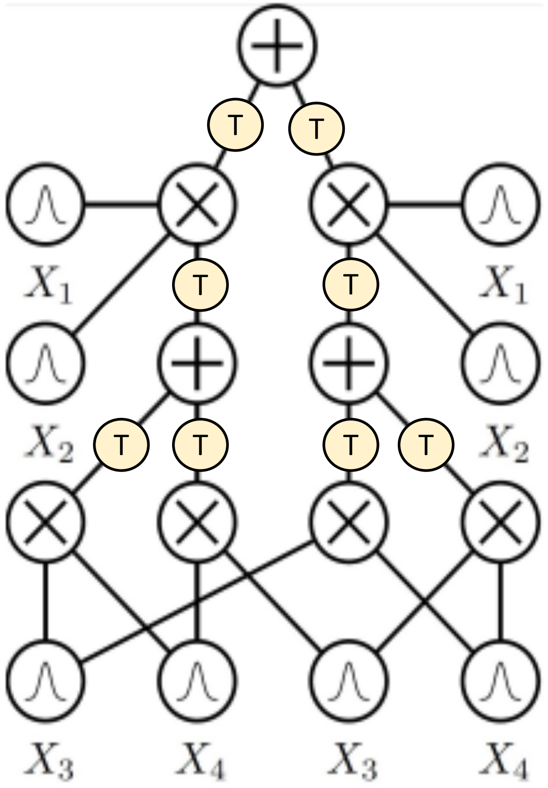


$$T(N(\boldsymbol{x})) = N(f(\boldsymbol{x}))|\det J_f|$$

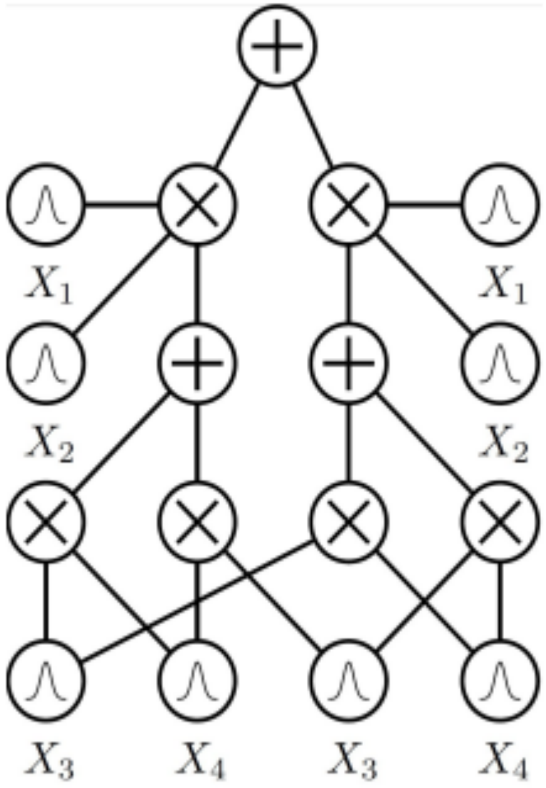
$$N = \{+, \text{x}, \}$$

Transform nodes
Represents **normalizing** flows
Adds expressivity

Sum-Product Transform Network
(Pevny et. Al 2020)



Probabilistic Circuit



Place transform nodes arbitrarily in a PC

Use invertible **affine** transformations

Integrating Flows with Probabilistic Circuits – Our Work

Understanding the Pathologies of Sum-Product Transform Networks

Placing **transform nodes** arbitrarily in a PC can **violate** its **decomposability** property
Inference becomes **Intractable** for Marginal, Conditional, and MAP

Integrating Flows with Probabilistic Circuits – Our Work

Understanding the Pathologies of Sum-Product Transform Networks

Placing **transform nodes** arbitrarily in a PC can **violate** its **decomposability** property
Inference becomes **Intractable** for Marginal, Conditional, and MAP

Probabilistic Circuits Smoothness and Decomposability allowed **pushing down** integrals to enable tractability

Integrating Flows with Probabilistic Circuits – Our Work

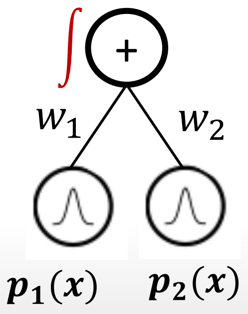
Understanding the Pathologies of Sum-Product Transform Networks

Placing **transform nodes** arbitrarily in a PC can **violate** its **decomposability** property
Inference becomes **Intractable** for Marginal, Conditional, and MAP

Probabilistic Circuits Smoothness and Decomposability allowed **pushing down** integrals to enable tractability

Smooth Sum Node

$$\int p(\boldsymbol{x})\, d\boldsymbol{x} = \int w_1\, p_1(\boldsymbol{x}) + w_2 p_2(\boldsymbol{x})\, d\boldsymbol{x}$$



Integrating Flows with Probabilistic Circuits – Our Work

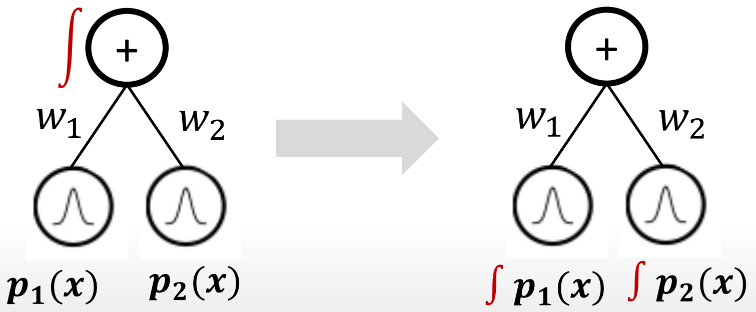
Understanding the Pathologies of Sum-Product Transform Networks

Placing **transform nodes** arbitrarily in a PC can **violate** its **decomposability** property
Inference becomes **Intractable** for Marginal, Conditional, and MAP

Probabilistic Circuits Smoothness and Decomposability allowed **pushing down** integrals to enable tractability

Smooth Sum Node

$$\int p(\boldsymbol{x})\, d\boldsymbol{x} = \int w_1\, p_1(\boldsymbol{x}) + w_2 p_2(\boldsymbol{x})\, d\boldsymbol{x}$$
$$= w_1 \int p_1(\boldsymbol{x}) d\boldsymbol{x} + w_2 \int p_2(\boldsymbol{x}) d\boldsymbol{x}$$



Integrating Flows with Probabilistic Circuits – Our Work

Understanding the Pathologies of Sum-Product Transform Networks

Placing **transform nodes** arbitrarily in a PC can **violate** its **decomposability** property

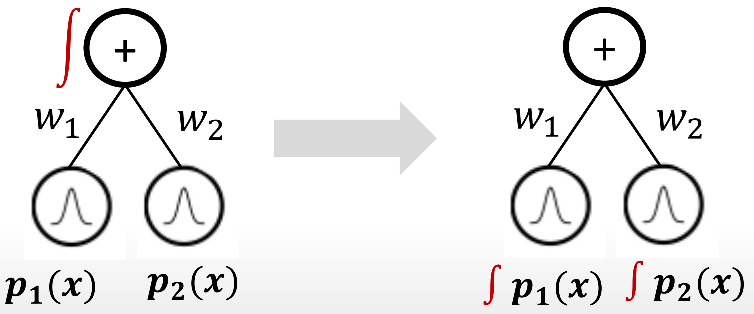
Inference becomes **Intractable** for Marginal, Conditional, and MAP

Probabilistic Circuits Smoothness and Decomposability allowed **pushing down** integrals to enable tractability

Smooth Sum Node

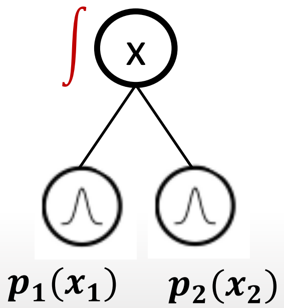
$$\int p(\boldsymbol{x})\, d\boldsymbol{x} = \int w_1 p_1(\boldsymbol{x}) + w_2 p_2(\boldsymbol{x})\, d\boldsymbol{x}$$

$$= w_1 \int p_1(\boldsymbol{x}) d\boldsymbol{x} + w_2 \int p_2(\boldsymbol{x}) d\boldsymbol{x}$$



Decomposable Product Node

$$\int p(\boldsymbol{x_1}, \boldsymbol{x_2})\, d\boldsymbol{x_1} d\boldsymbol{x_2} = \int p_1(\boldsymbol{x_1}) p_2(\boldsymbol{x_2}) d\boldsymbol{x_1} d\boldsymbol{x_2}$$



Integrating Flows with Probabilistic Circuits – Our Work

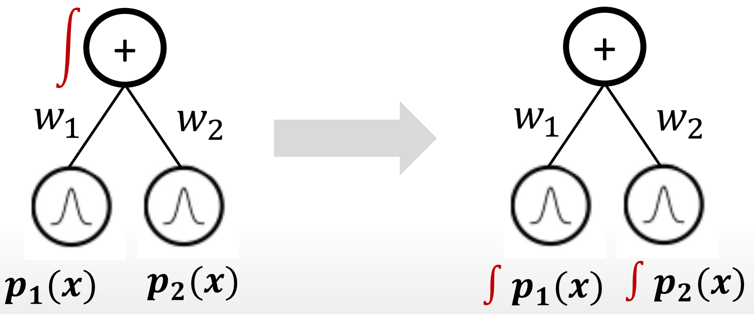
Understanding the Pathologies of Sum-Product Transform Networks

Placing **transform nodes** arbitrarily in a PC can **violate** its **decomposability** property
Inference becomes **Intractable** for Marginal, Conditional, and MAP

Probabilistic Circuits Smoothness and Decomposability allowed **pushing down** integrals to enable tractability

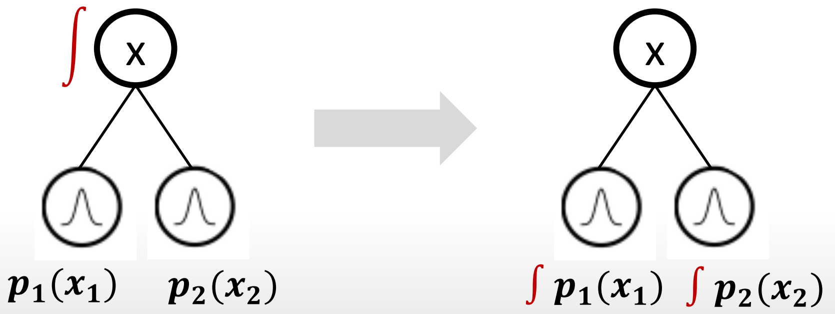
Smooth Sum Node

$$\int p(\boldsymbol{x})\, d\boldsymbol{x} = \int w_1 p_1(\boldsymbol{x}) + w_2 p_2(\boldsymbol{x})\, d\boldsymbol{x}$$
$$= w_1 \int p_1(\boldsymbol{x}) d\boldsymbol{x} + w_2 \int p_2(\boldsymbol{x}) d\boldsymbol{x}$$



Decomposable Product Node

$$\int p(\boldsymbol{x_1}, \boldsymbol{x_2})\, d\boldsymbol{x_1} d\boldsymbol{x_2} = \int p_1(\boldsymbol{x_1}) p_2(\boldsymbol{x_2}) d\boldsymbol{x_1} d\boldsymbol{x_2}$$
$$= \left(\int p_1(\boldsymbol{x_1}) d\boldsymbol{x_1}\right) . \left(\int p_2(\boldsymbol{x_2})\, d\boldsymbol{x_2}\right)$$



Integrating Flows with Probabilistic Circuits – Our Work

Understanding the Pathologies of Sum-Product Transform Networks

Sum Product Transform Networks	Can you still push down integrals over transform nodes ?

Integrating Flows with Probabilistic Circuits – Our Work

Understanding the Pathologies of Sum-Product Transform Networks

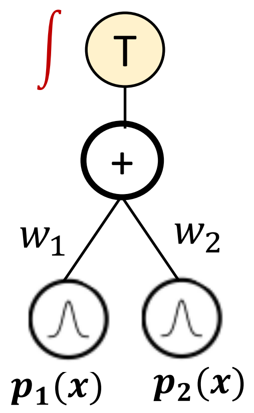
Sum Product Transform Networks Can you still push down integrals over transform nodes ?

Integrals on **Transform** Nodes **over Sum** Node

$$\int p(\boldsymbol{x})\, d\boldsymbol{x}$$

$$= \int T(+(\boldsymbol{x}))d\boldsymbol{x} = \int +(f(\boldsymbol{x}))|\det J_f|\ d\boldsymbol{x}$$

$$= \int [w_1 p_1(f(\boldsymbol{x})) + w_2 p_2(f(\boldsymbol{x}))]\, |\det J_f|\ d\boldsymbol{x}$$



Integrating Flows with Probabilistic Circuits – Our Work

Understanding the Pathologies of Sum-Product Transform Networks

Sum Product Transform Networks Can you still push down integrals over transform nodes ?

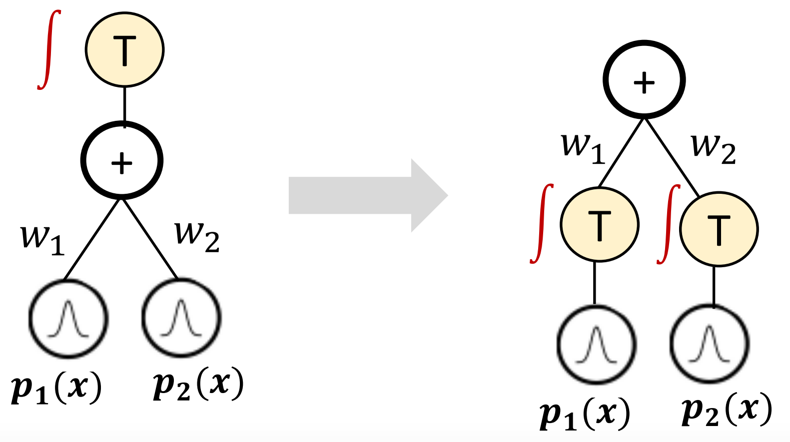
Integrals on **Transform** Nodes **over Sum** Node

$$\int p(\boldsymbol{x})\, d\boldsymbol{x}$$

$$= \int T(+(\boldsymbol{x}))d\boldsymbol{x} = \int +(f(\boldsymbol{x}))|\det J_f|\ d\boldsymbol{x}$$

$$= \int [w_1 p_1(f(\boldsymbol{x})) + w_2 p_2(f(\boldsymbol{x}))]\ |\det J_f|\ d\boldsymbol{x}$$

$$= w_1 \int p_1(f(\boldsymbol{x}))|\det J_f| d\boldsymbol{x}\ + w_2 \int p_2(f(\boldsymbol{x}))|\det J_f| d\boldsymbol{x}$$



Integrating Flows with Probabilistic Circuits – Our Work

Pathologies of Sum-Product Transform Networks

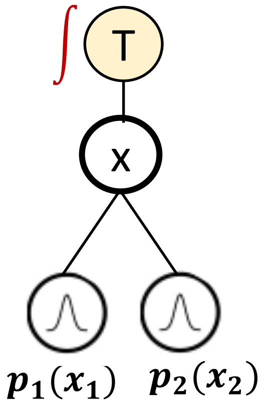
Integrals on Transform Nodes over Product Node

$$\int p(\boldsymbol{x_1},\boldsymbol{x_2})\, d\boldsymbol{x} \qquad (d\boldsymbol{x} = d\boldsymbol{x_1} d\boldsymbol{x_2})$$

$$= \int T(\times(\boldsymbol{x_1},\boldsymbol{x_2}))\, d\boldsymbol{x}$$

$$= \int \times(f(\boldsymbol{x_1},\boldsymbol{x_2}))|\det J_f|\ d\boldsymbol{x}$$

$$= \int p_1\big(f(\boldsymbol{x_1},\boldsymbol{x_2})\big)p_2(f(\boldsymbol{x_1},\boldsymbol{x_2}))|\det J_f|\ d\boldsymbol{x}$$



Integrating Flows with Probabilistic Circuits – Our Work

Pathologies of Sum-Product Transform Networks

Integrals on **Transform** Nodes **over Product** Node

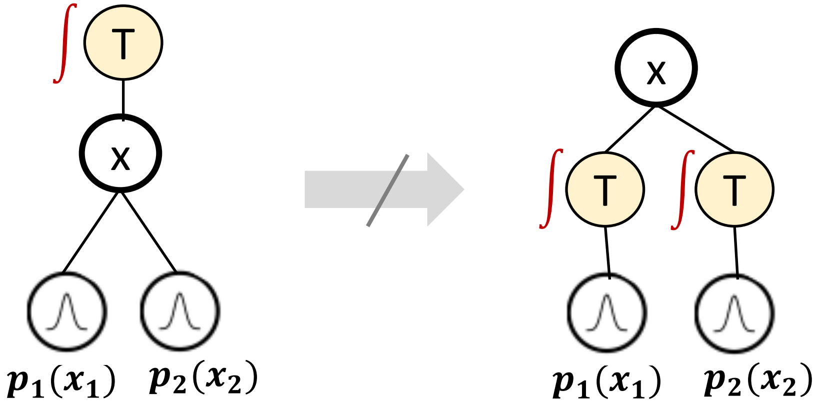
$$\int p(\boldsymbol{x_1}, \boldsymbol{x_2})\, d\boldsymbol{x} \qquad (d\boldsymbol{x} = d\boldsymbol{x_1} d\boldsymbol{x_2})$$

$$= \int T(\times(\boldsymbol{x_1}, \boldsymbol{x_2}))\, d\boldsymbol{x}$$

$$= \int \times(f(\boldsymbol{x_1}, \boldsymbol{x_2}))|\det J_f|\; d\boldsymbol{x}$$

$$= \int p_1\big(f(\boldsymbol{x_1}, \boldsymbol{x_2})\big) p_2(f(\boldsymbol{x_1}, \boldsymbol{x_2}))|\det J_f|\; d\boldsymbol{x}$$

$$\neq \int p_1\big(f(\boldsymbol{x_1})\big) p_2(f(\boldsymbol{x_2}))|\det J_f|\; d\boldsymbol{x}$$



Integrating Flows with Probabilistic Circuits – Our Work

Pathologies of Sum-Product Transform Networks

Integrals on **Transform** Nodes **over Product** Node

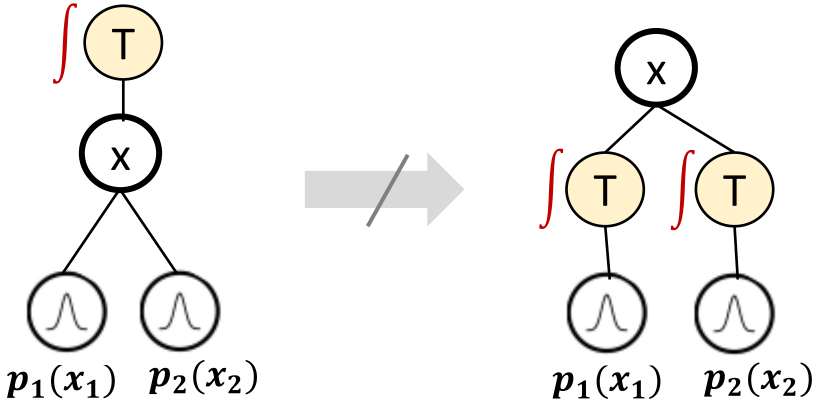
$$\int p(\boldsymbol{x_1}, \boldsymbol{x_2})\, d\boldsymbol{x} \qquad (d\boldsymbol{x} = d\boldsymbol{x_1} d\boldsymbol{x_2})$$

$$= \int T(\times(\boldsymbol{x_1}, \boldsymbol{x_2}))\, d\boldsymbol{x}$$

$$= \int \times(f(\boldsymbol{x_1}, \boldsymbol{x_2}))|\det J_f|\ d\boldsymbol{x}$$

$$= \int p_1(f(\boldsymbol{x_1}, \boldsymbol{x_2}))p_2(f(\boldsymbol{x_1}, \boldsymbol{x_2}))|\det J_f|\ d\boldsymbol{x}$$

$$\neq \int p_1(f(\boldsymbol{x_1}))p_2(f(\boldsymbol{x_2}))|\det J_f|\ d\boldsymbol{x}$$



Transform nodes causes scopes of children of product nodes to overlap

Cannot push down integrals on transform nodes over products → Intractable for marginal, conditional and MAP

Integrating Flows with Probabilistic Circuits – Our Work

Defining Structural Properties for Transform Nodes

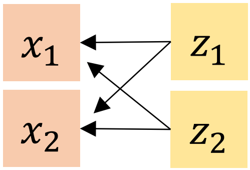
τ −Decomposability	When defined over a product node, f needs to transform the variables involved in the scope of its children **independently**

Integrating Flows with Probabilistic Circuits – Our Work

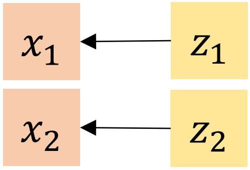
Defining Structural Properties for Transform Nodes

τ −Decomposability

When defined over a product node, f needs to transform the variables involved in the scope of its children **independently**



Not τ-Decomposable
$\boldsymbol{z} = f(\boldsymbol{x})$

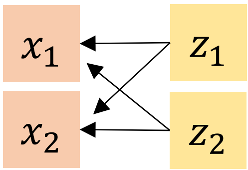


τ-Decomposable
$\boldsymbol{z} = [z_1, z_2] = [f_1(x_1), f_2(x_2)]$

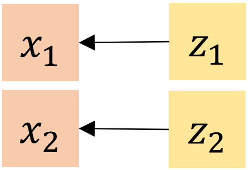
Integrating Flows with Probabilistic Circuits – Our Work

Defining Structural Properties for Transform Nodes

τ –Decomposability When defined over a product node, f needs to transform the variables involved in the scope of its children **independently**



Not τ-Decomposable
$\boldsymbol{z} = f(\boldsymbol{x})$



τ-Decomposable
$\boldsymbol{z} = [z_1, z_2] = [f_1(x_1), f_2(x_2)]$

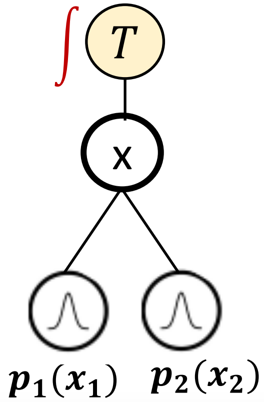
Integrals on τ-Decomposable Transform Nodes over Product Node

$$\int p(\boldsymbol{x_1}, \boldsymbol{x_2})\, d\boldsymbol{x} \qquad (d\boldsymbol{x} = d\boldsymbol{x_1} d\boldsymbol{x_2})$$

$$= \int T(\times(\boldsymbol{x_1}, \boldsymbol{x_2}))\, d\boldsymbol{x}$$

$$= \int \times(\boldsymbol{f}(\boldsymbol{x_1}, \boldsymbol{x_2}))|\det J_f|\ d\boldsymbol{x}$$

$$= \int p_1(f(\boldsymbol{x_1}, \boldsymbol{x_2}))p_2(f(\boldsymbol{x_1}, \boldsymbol{x_2}))|\det J_f|\ d\boldsymbol{x}$$

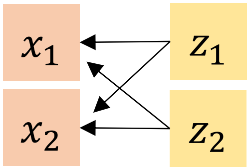


Integrating Flows with Probabilistic Circuits – Our Work

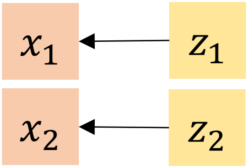
Defining Structural Properties for Transform Nodes

τ –Decomposability

When defined over a product node, f needs to transform the variables involved in the scope of its children **independently**



Not τ-Decomposable
$\boldsymbol{z} = f(\boldsymbol{x})$



τ-Decomposable
$\boldsymbol{z} = [z_1, z_2] = [f_1(x_1), f_2(x_2)]$

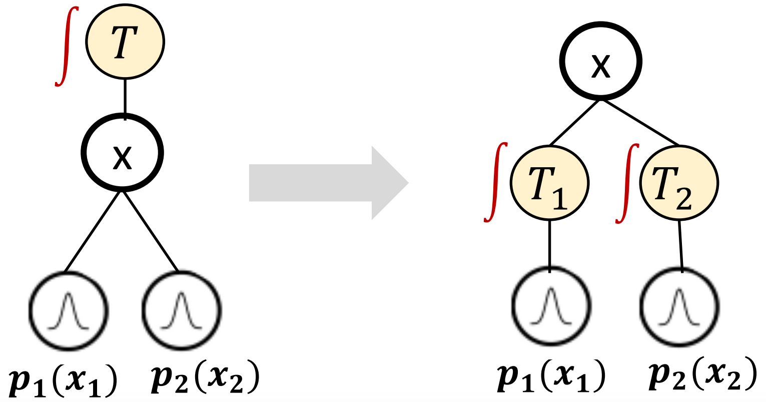
Integrals on $\boldsymbol{\tau}$-Decomposable Transform Nodes over Product Node

$$\int p(\boldsymbol{x_1}, \boldsymbol{x_2})\, d\boldsymbol{x} \qquad (d\boldsymbol{x} = d\boldsymbol{x_1} d\boldsymbol{x_2})$$

$$= \int T(\times(\boldsymbol{x_1}, \boldsymbol{x_2}))\, d\boldsymbol{x}$$

$$= \int \times(\boldsymbol{f}(\boldsymbol{x_1}, \boldsymbol{x_2}))|\det J_f|\ d\boldsymbol{x}$$

$$= \int p_1(f(\boldsymbol{x_1}, \boldsymbol{x_2}))p_2(f(\boldsymbol{x_1}, \boldsymbol{x_2}))|\det J_f|\ d\boldsymbol{x}$$



$$= \int p_1(f_1(\boldsymbol{x_1}))p_2(f_2(\boldsymbol{x_2}))|\det J_f| d\boldsymbol{x} = \left(\int p_1(f_1(\boldsymbol{x_1}))|\det J_{f_1}| d\boldsymbol{x_1}\right)\left(\int p_2(f_2(\boldsymbol{x_2}))|\det J_{f_2}| d\boldsymbol{x_2}\right)$$

Integrating Flows with Probabilistic Circuits – Our Work

Defining Structural Properties for Transform Nodes

τ −Decomposability is a necessary condition for tractability

A sum-product-transform network is decomposable only if all of its transform nodes are τ −decomposable

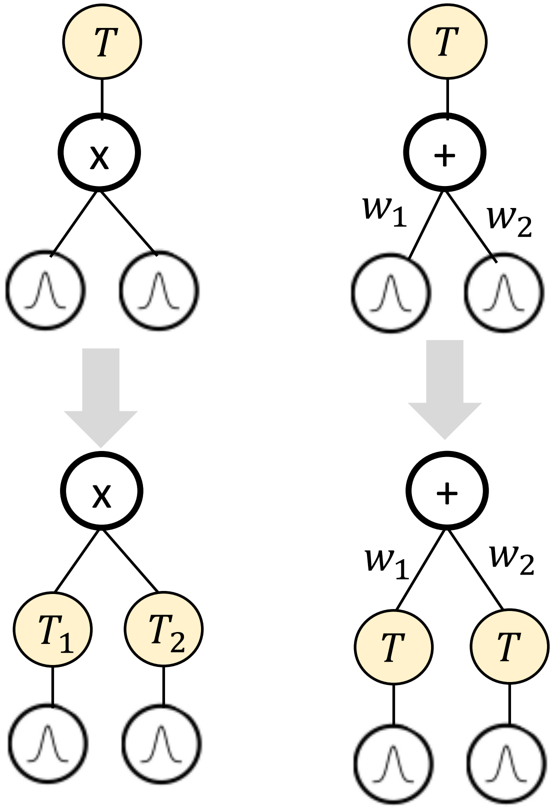
Integrating Flows with Probabilistic Circuits – Our Work

Defining Structural Properties for Transform Nodes

τ –Decomposability is a necessary condition for tractability

A sum-product-transform network is decomposable only if all of its transform nodes are τ –decomposable

Implications of τ –decomposability



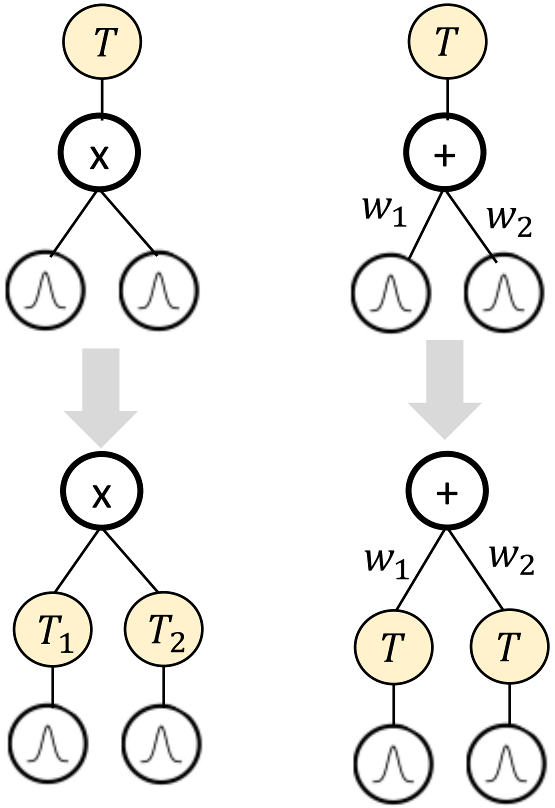
Integrating Flows with Probabilistic Circuits – Our Work

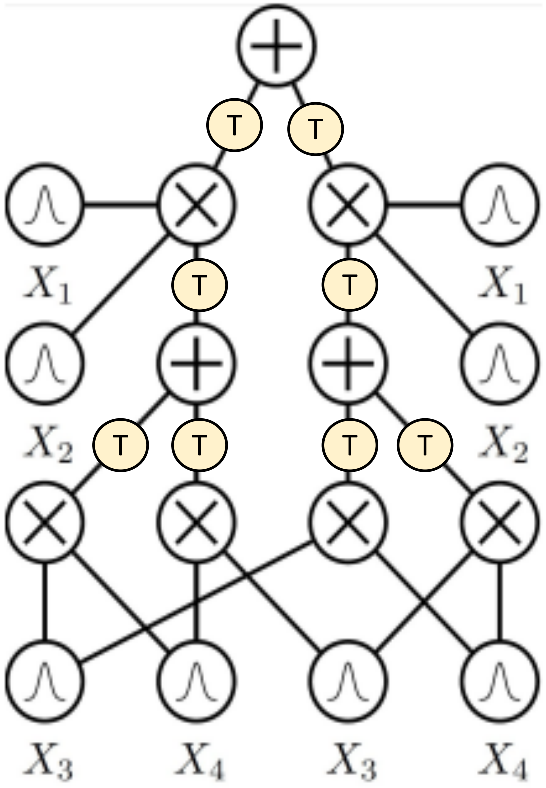
Defining Structural Properties for Transform Nodes

τ –Decomposability is a necessary condition for tractability

A sum-product-transform network is decomposable only if all of its transform nodes are τ –decomposable

Implications of τ –decomposability





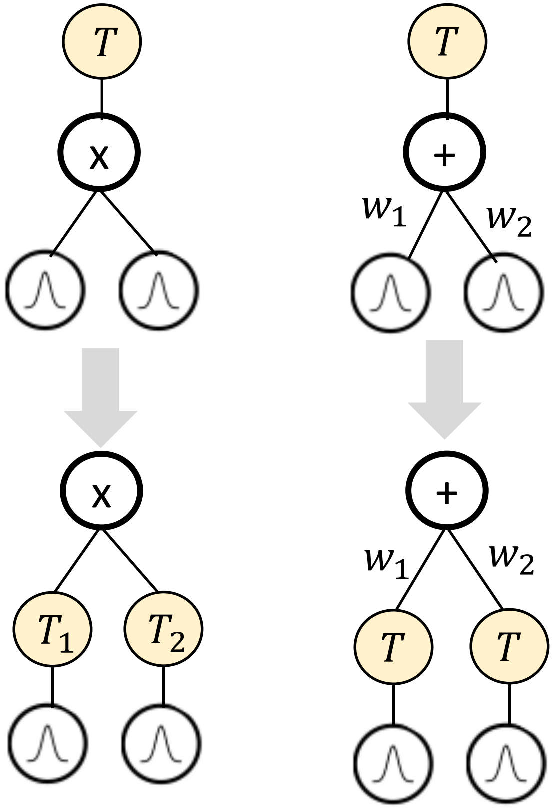
Integrating Flows with Probabilistic Circuits – Our Work

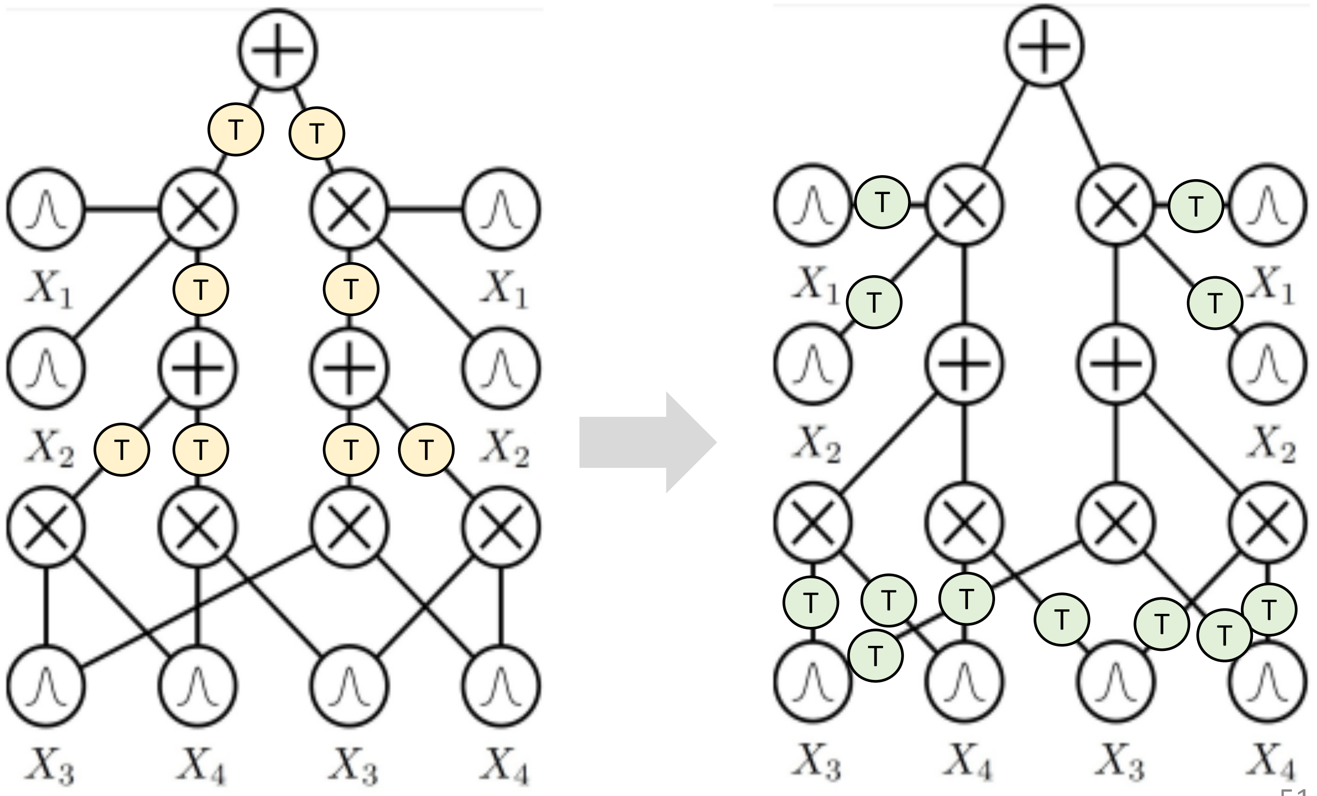
Defining Structural Properties for Transform Nodes

τ –Decomposability is a necessary condition for tractability

A sum-product-transform network is decomposable only if all of its transform nodes are τ –decomposable

Implications of τ –decomposability





Integrating Flows with Probabilistic Circuits – Our Work

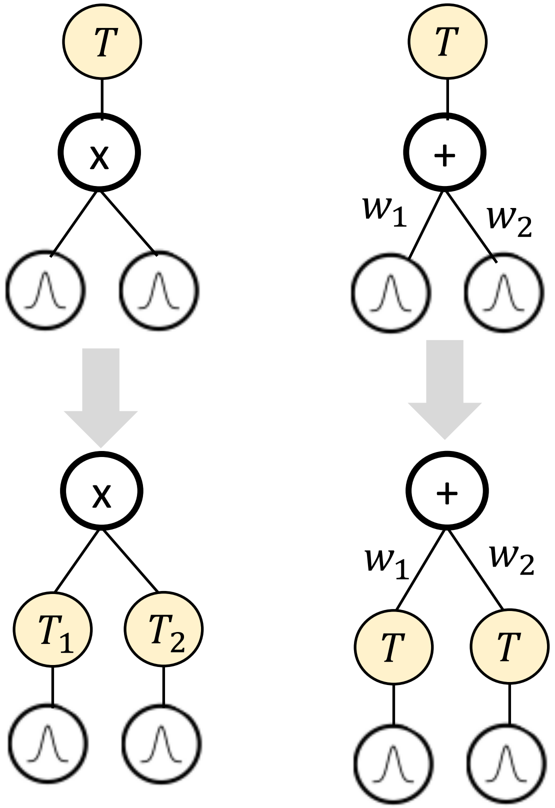
Defining Structural Properties for Transform Nodes

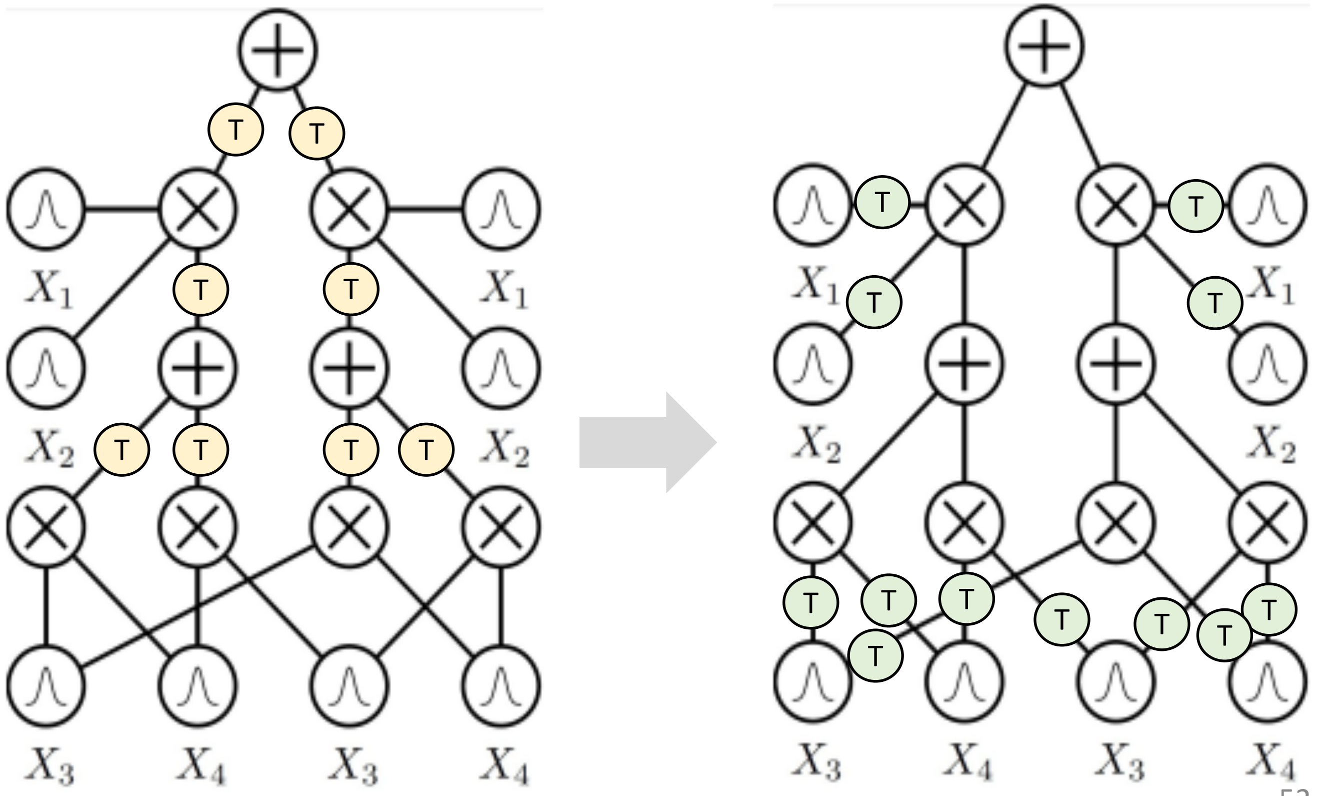
τ –Decomposability is a necessary condition for tractability

A sum-product-transform network is decomposable only if all of its transform nodes are τ –decomposable

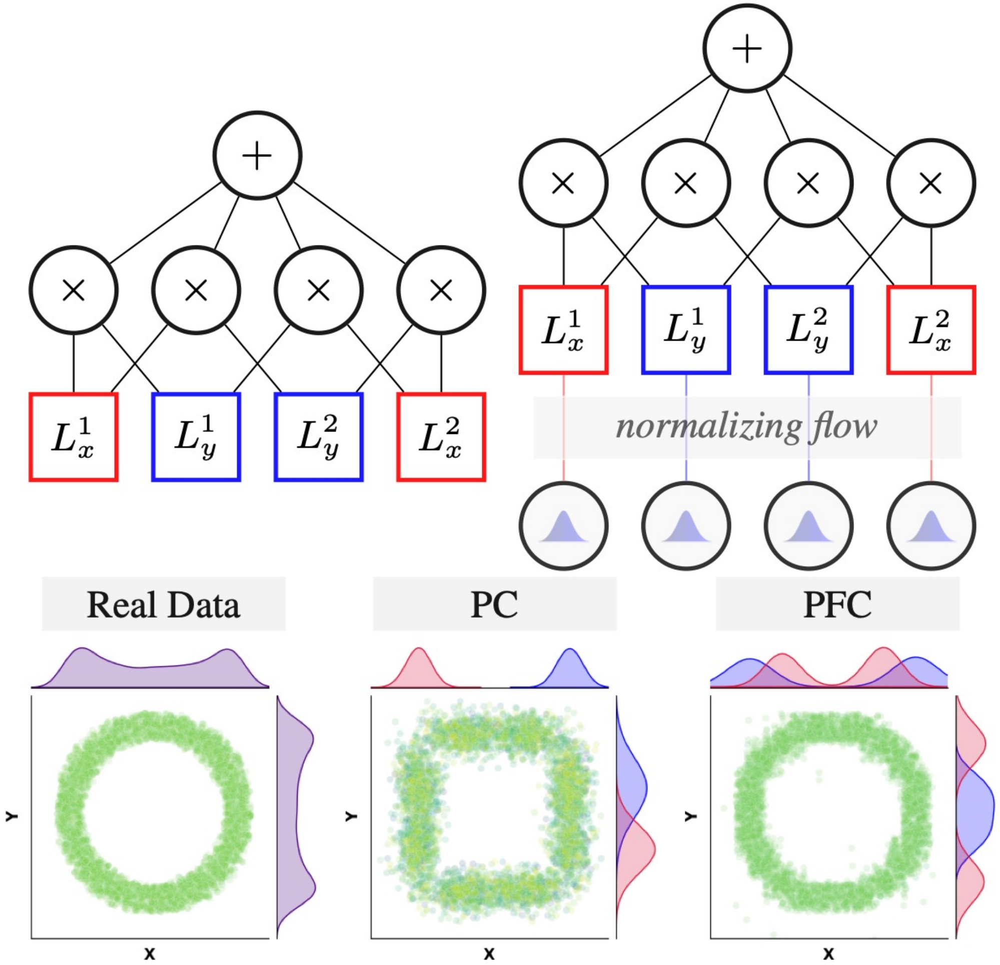
Implications of τ –decomposability

Reduces to having normalizing flows at the leaves!





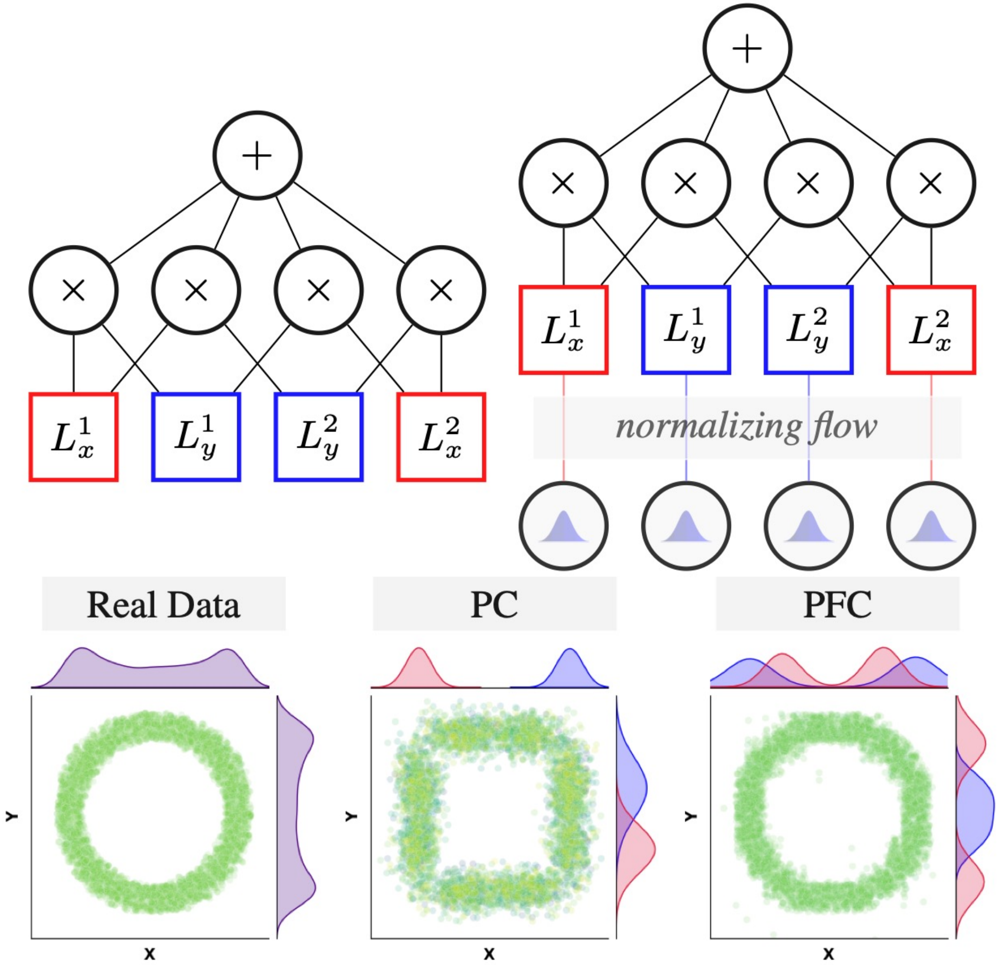
Probabilistic Flows Circuits



Has added Expressivity
Can model arbitrarily complex distributions at the leaves

Retains Tractability
As it encodes the same factorizations of the PC

Probabilistic Flows Circuits

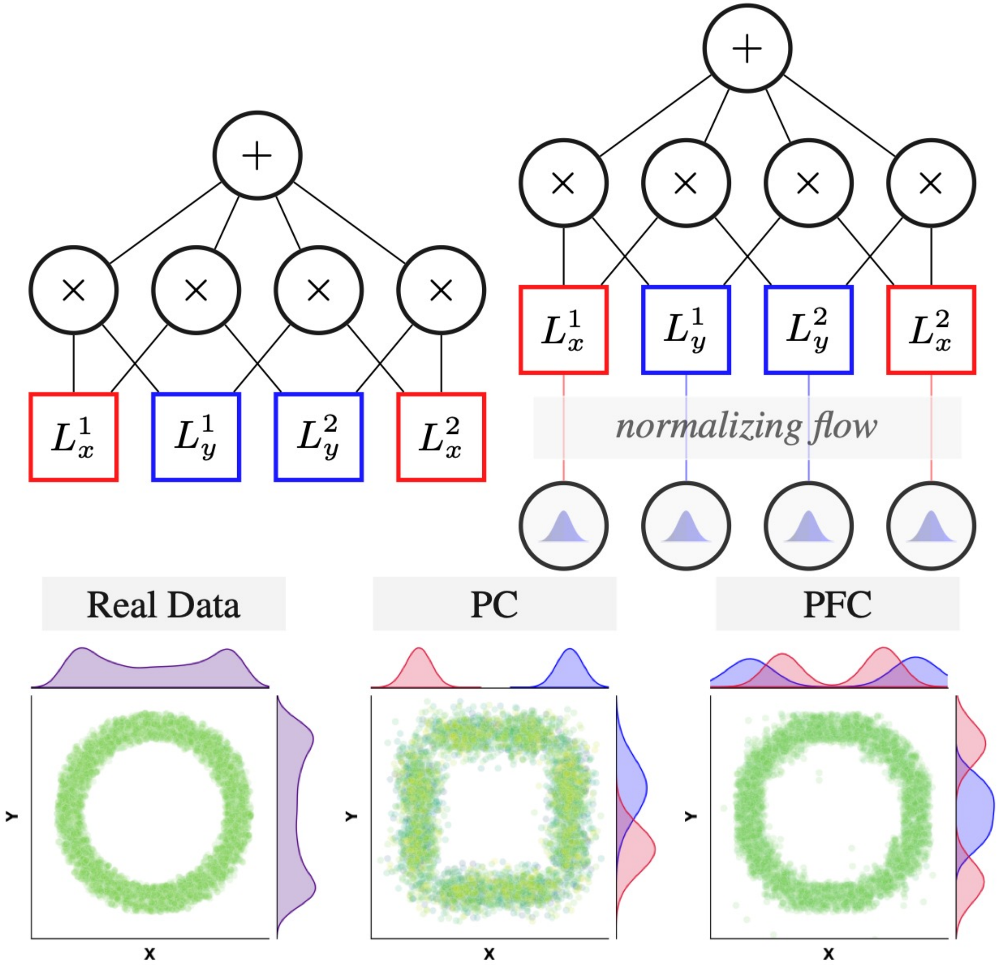


Has added Expressivity
Can model arbitrarily complex distributions at the leaves

Retains Tractability
As it encodes the same factorizations of the PC

What transformations to use ?

Probabilistic Flows Circuits



Has added Expressivity
Can model arbitrarily complex distributions at the leaves

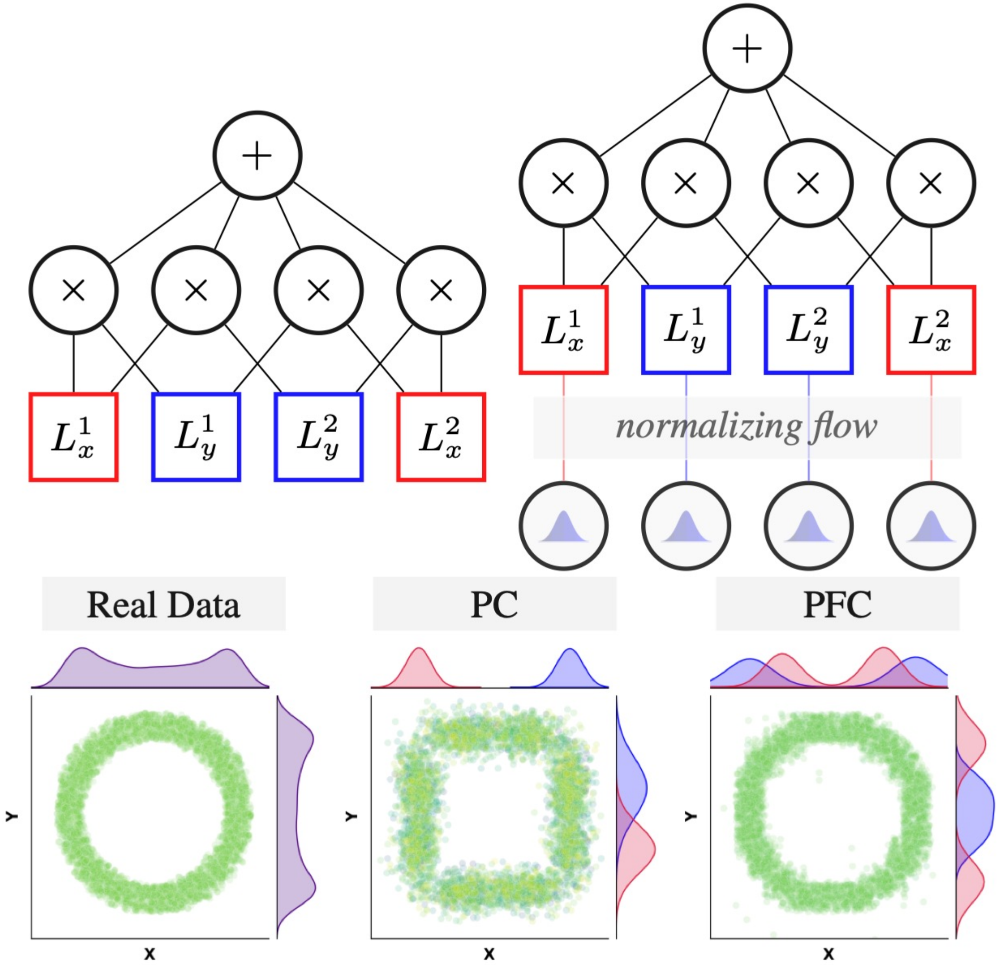
Retains Tractability
As it encodes the same factorizations of the PC

What transformations to use ?

Affine (Pevny et. Al) ?

Affine-transformed Gaussian leaf is still a Gaussian!

Probabilistic Flows Circuits



Has added Expressivity
Can model arbitrarily complex distributions at the leaves

Retains Tractability
As it encodes the same factorizations of the PC

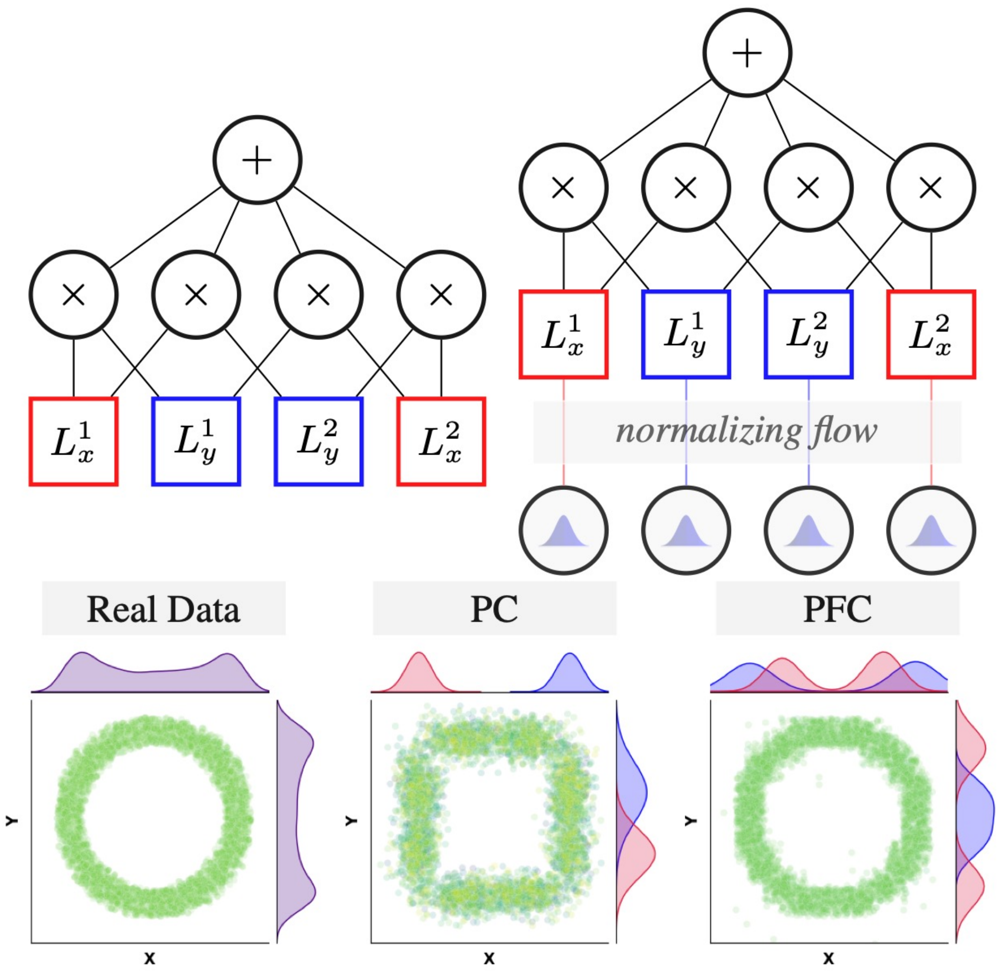
What transformations to use ?

Affine (Pevny et. Al) ?

Affine-transformed Gaussian leaf is still a Gaussian!

What other properties do we need to consider when designing PFCs?

Probabilistic Flows Circuits



Has added Expressivity
Can model arbitrarily complex distributions at the leaves

Retains Tractability
As it encodes the same factorizations of the PC

What transformations to use ?

Affine (Pevny et. Al) ?

Affine-transformed Gaussian leaf is still a Gaussian!

What other properties do we need to consider when designing PFCs?

MAP requires the ability to compute the **modes** of leaf distributions

PC
Unimodal Leaf
Easy to compute mode

PFC
Multimodal Leaf
Difficult to compute mode

Probabilistic Flows Circuits

Designing expressive transformations using linear rational splines

Spline-based flows are among SOTA

Splines - piecewise functions

Divide the data space into K bins and fit a polynomial function f_k within each

Probabilistic Flows Circuits

Designing expressive transformations using linear rational splines

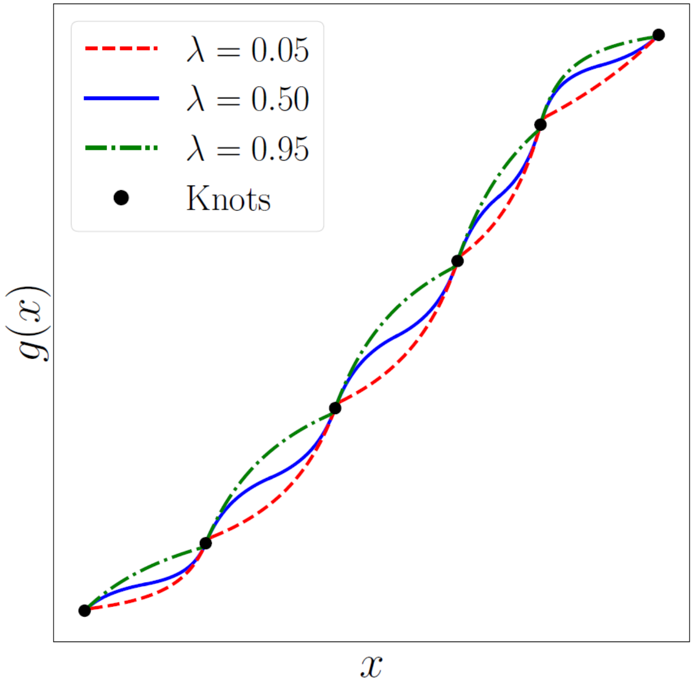
Spline-based flows are among SOTA

Splines - piecewise functions

Divide the data space into K bins and fit a polynomial function f_k within each

Linear Rational Splines (LRS)

Use monotone linear rational functions of the form $f(x) = \frac{ax+b}{cx+d}$



Probabilistic Flows Circuits

Designing expressive transformations using linear rational splines

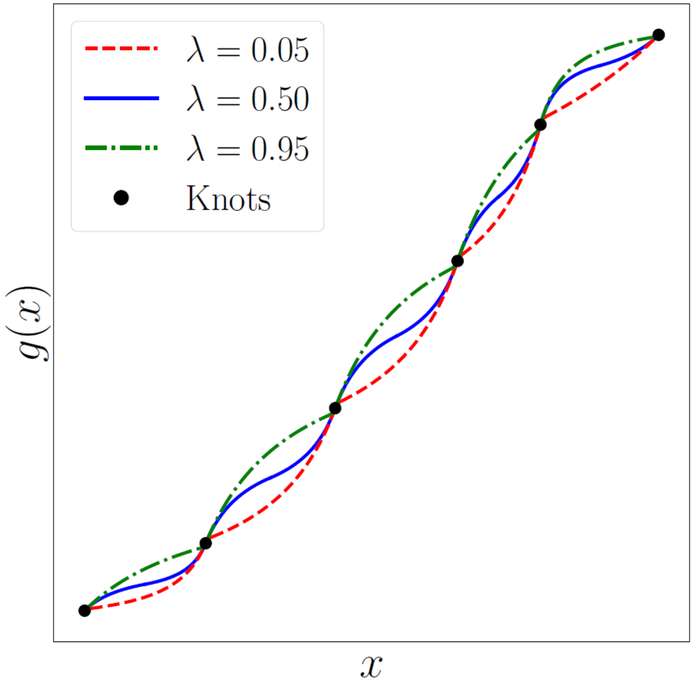
Spline-based flows are among SOTA

Splines - piecewise functions

Divide the data space into K bins and fit a polynomial function f_k within each

Linear Rational Splines (LRS)

Use monotone linear rational functions of the form $f(x) = \frac{ax+b}{cx+d}$



A PFC with LRS transformations is provably tractable for

Probabilistic Flows Circuits

Designing expressive transformations using linear rational splines

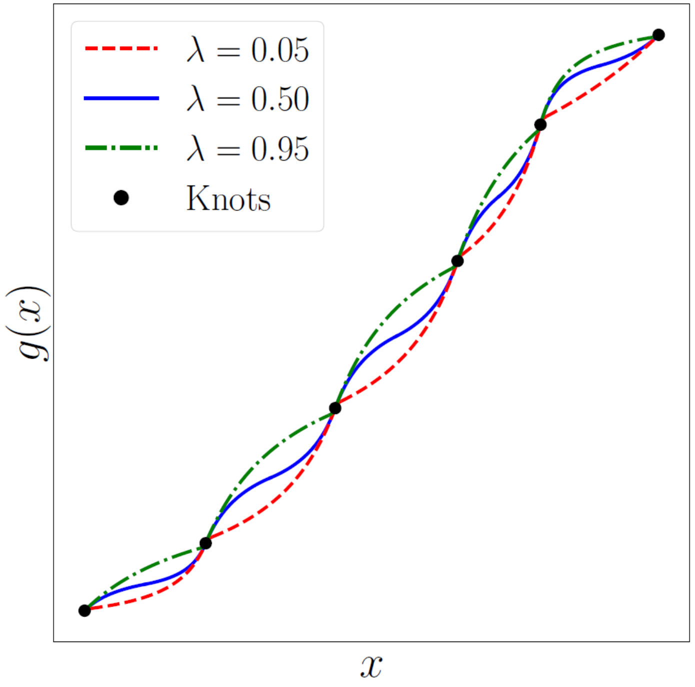
Spline-based flows are among SOTA

Splines - piecewise functions

Divide the data space into K bins and fit a polynomial function f_k within each

Linear Rational Splines (LRS)

Use monotone linear rational functions of the form $f(x) = \frac{ax+b}{cx+d}$



A PFC with LRS transformations is provably tractable for

Evidential

smooth

Probabilistic Flows Circuits

Designing expressive transformations using linear rational splines

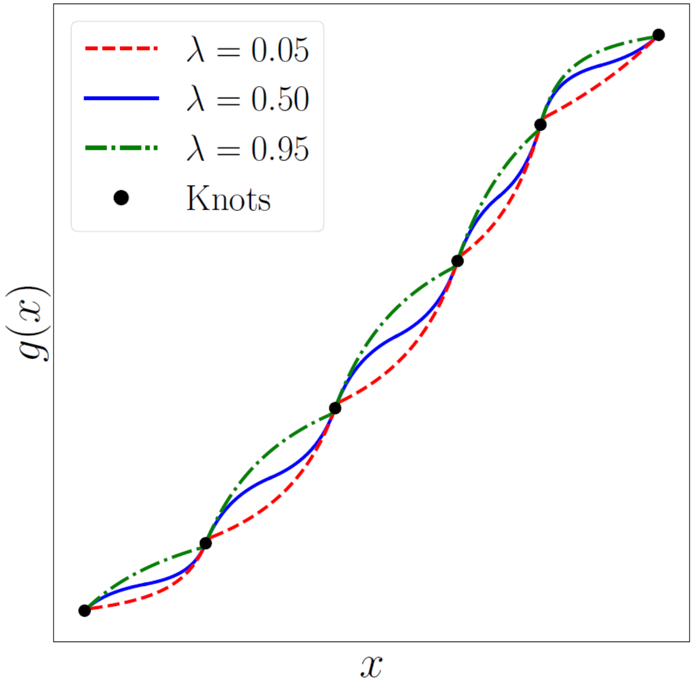
Spline-based flows are among SOTA

Splines - piecewise functions

Divide the data space into K bins and fit a polynomial function f_k within each

Linear Rational Splines (LRS)

Use monotone linear rational functions of the form $f(x) = \frac{ax+b}{cx+d}$



A PFC with LRS transformations is provably tractable for

- Evidential
  - smooth
- Marginal
  - smooth
  - decomposable

Probabilistic Flows Circuits

Designing expressive transformations using linear rational splines

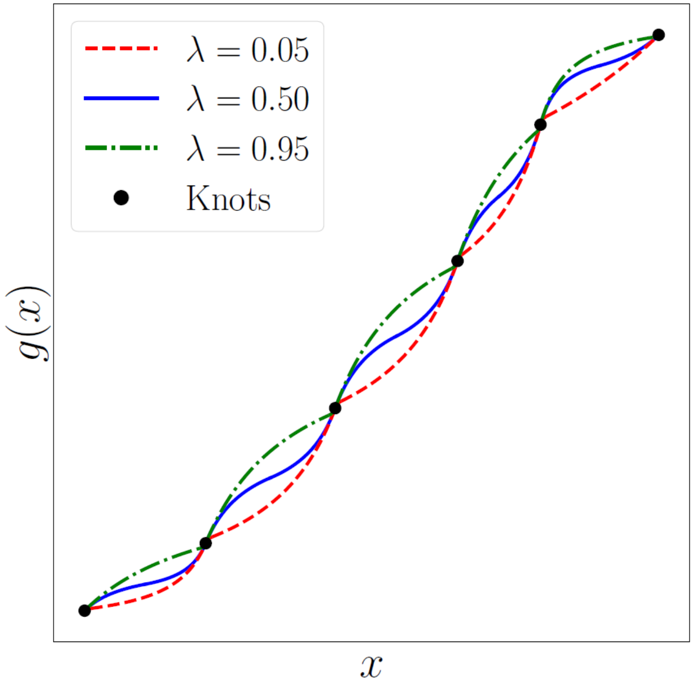
Spline-based flows are among SOTA

Splines - piecewise functions

Divide the data space into K bins and fit a polynomial function f_k within each

Linear Rational Splines (LRS)

Use monotone linear rational functions of the form $f(x) = \frac{ax+b}{cx+d}$



A PFC with LRS transformations is provably tractable for

- Evidential
  - smooth
- Marginal
  - smooth
  - decomposable
- Conditional
  - smooth
  - decomposable

Probabilistic Flows Circuits

Designing expressive transformations using linear rational splines

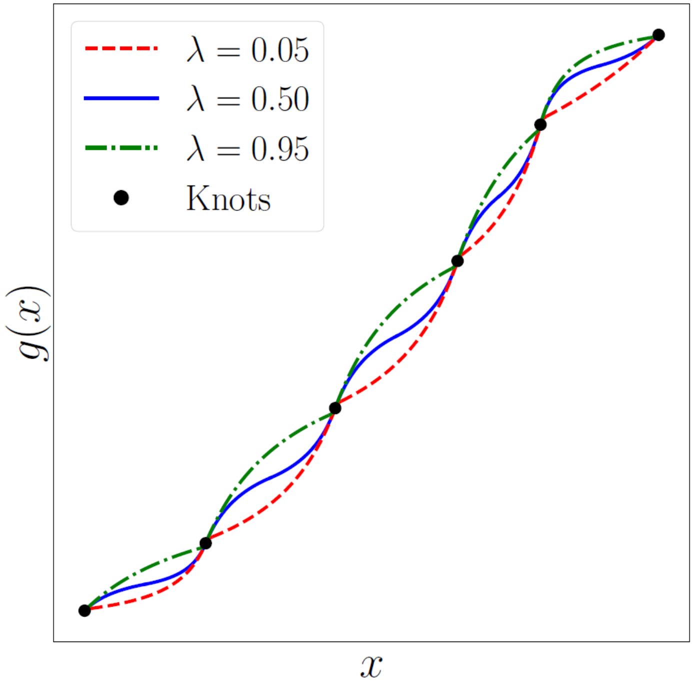
Spline-based flows are among SOTA

Splines - piecewise functions

Divide the data space into K bins and fit a polynomial function f_k within each

Linear Rational Splines (LRS)

Use monotone linear rational functions of the form $f(x) = \frac{ax+b}{cx+d}$



A PFC with LRS transformations is provably tractable for

- Evidential
  - smooth
- Marginal
  - smooth
  - decomposable
- Conditional
  - smooth
  - decomposable
- MAP
  - decomposable
  - deterministic
  - Student’s t base

Probabilistic Flows Circuits

Designing expressive transformations using linear rational splines

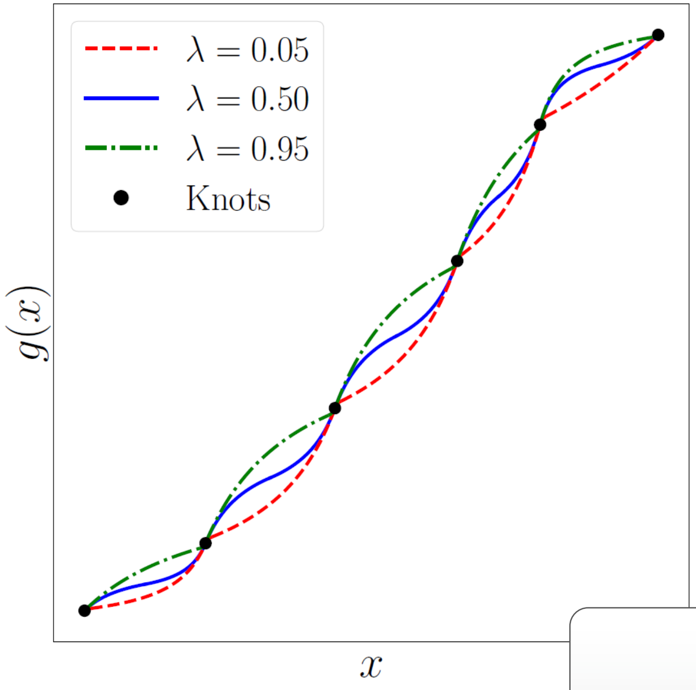
Spline-based flows are among SOTA

Splines - piecewise functions

Divide the data space into K bins and fit a polynomial function f_k within each

Linear Rational Splines (LRS)

Use monotone linear rational functions of the form $f(x) = \frac{ax+b}{cx+d}$



A PFC with LRS transformations is provably tractable for

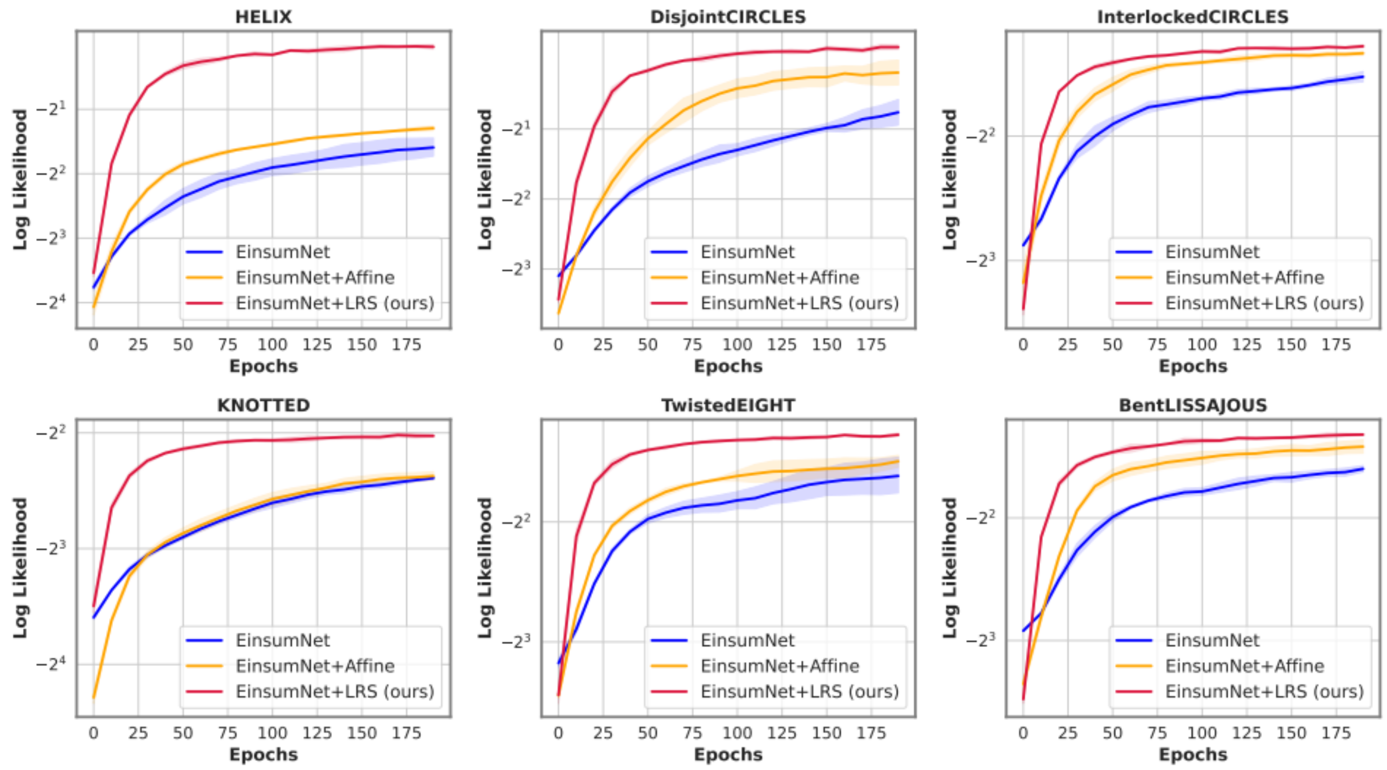
- Evidential
  - smooth
- Marginal
  - smooth
  - decomposable
- Conditional
  - smooth
  - decomposable
- MAP
  - decomposable
  - deterministic
  - Student's t base

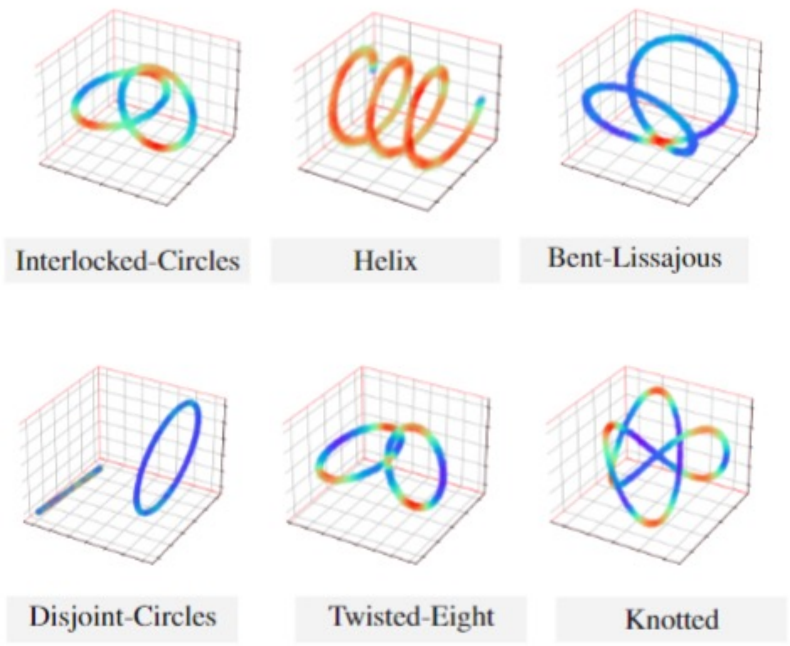
Other flow transformations that (provably) achieves the same ?

Probabilistic Flows Circuits

Experimental Results

Has added expressivity and learning efficiency over a PC - Better models the data

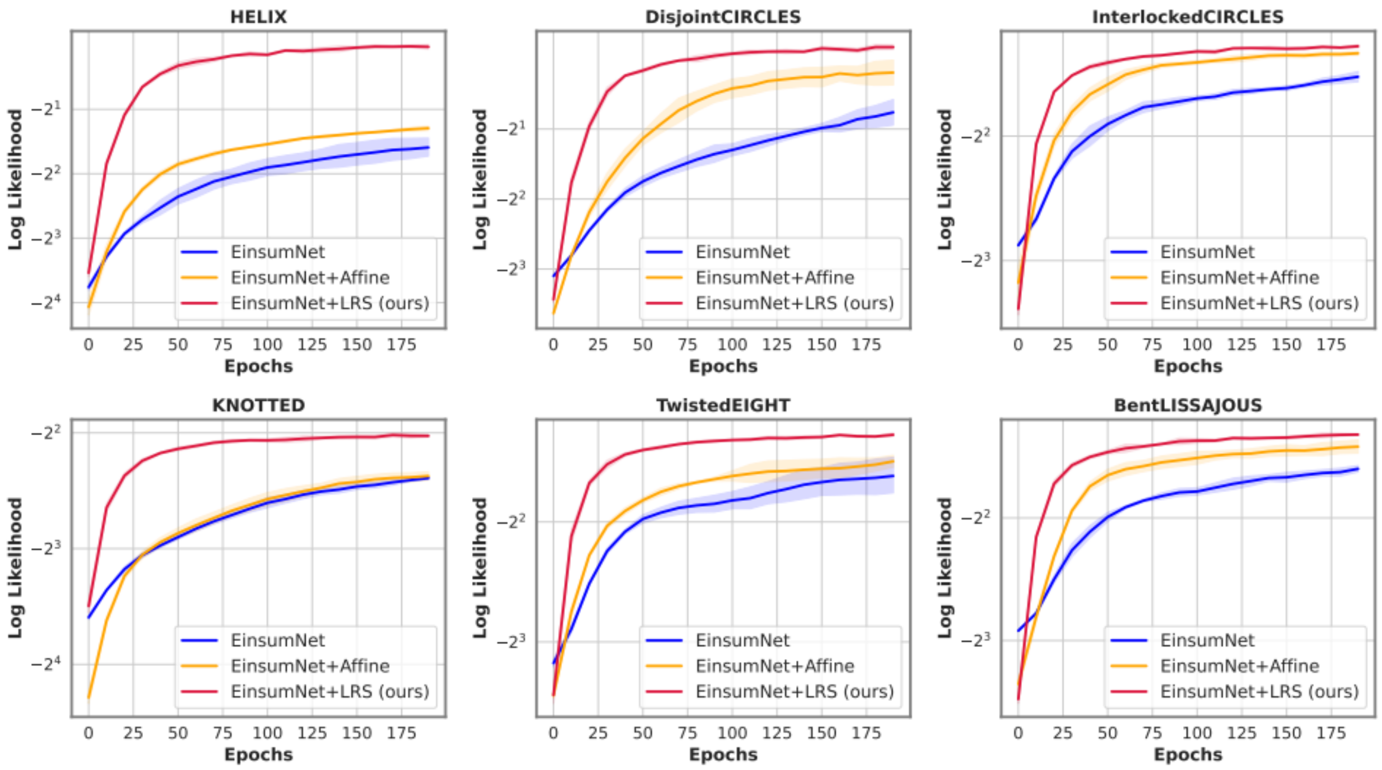


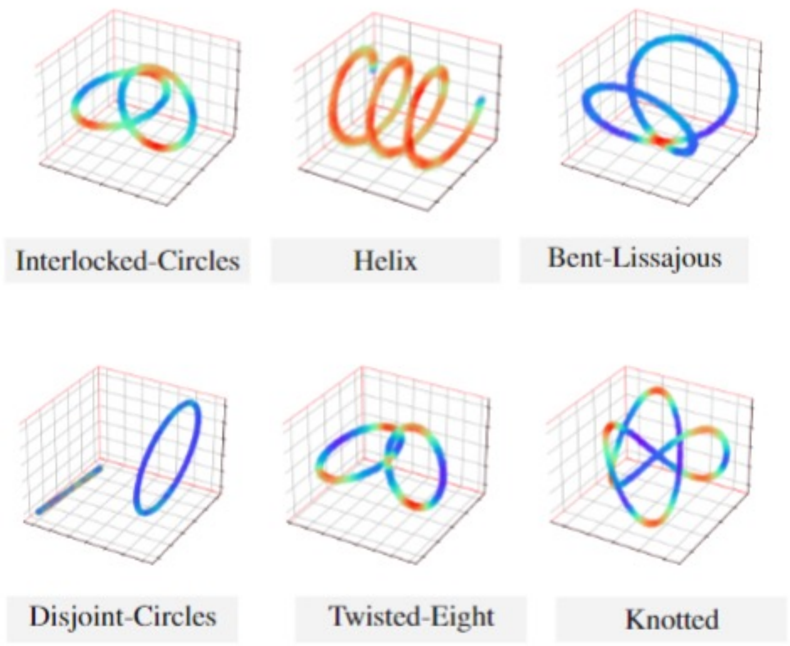


Probabilistic Flows Circuits

Experimental Results

Has added expressivity and learning efficiency over a PC - Better models the data





	POWER	GAS	MINIBOONE	HEPMASS	MNIST	Fashion-MNIST
MADE	-3.08 ± 0.03	3.56 ± 0.04	-15.59 ± 0.50	-20.98 ± 0.02	-1380.8 ± 4.8	-
Real NVP	-0.02 ± 0.01	4.78 ± 1.80	-13.55 ± 0.49	-19.62 ± 0.02	-1323.2 ± 6.6	-
MAF	0.14 ± 0.01	9.07 ± 0.02	-11.75 ± 0.44	-17.70 ± 0.02	-1300.5 ± 1.7	-
EinsumNet	0.20 ± 0.01	3.57 ± 0.08	-35.93 ± 0.06	-22.79 ± 0.05	-1015.1 ± 0.9	649.1 ± 0.3
Einsum+LRS	0.36 ± 0.01	4.79 ± 0.04	-34.21 ± 0.01	-22.46 ± 0.01	-959.4 ± 1.4	655.6 ± 0.6

Probabilistic Flows Circuits

Experimental Results

Retains Tractability – which can be exploited for downstream tasks

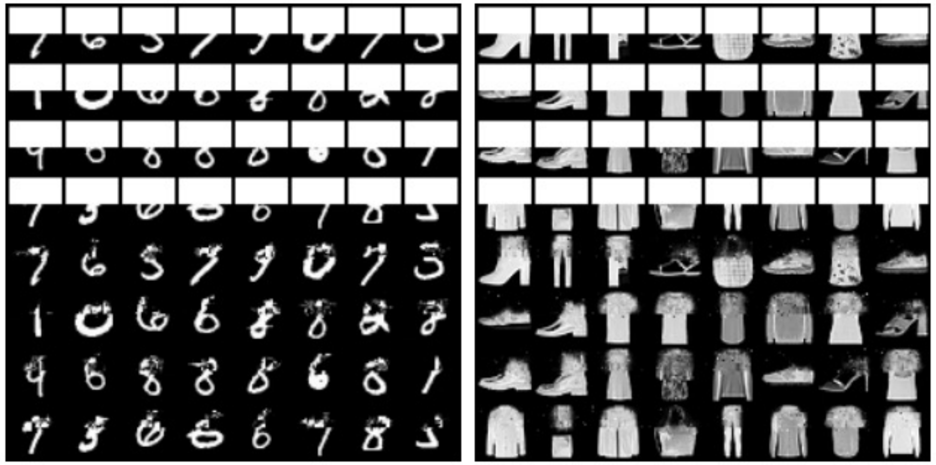
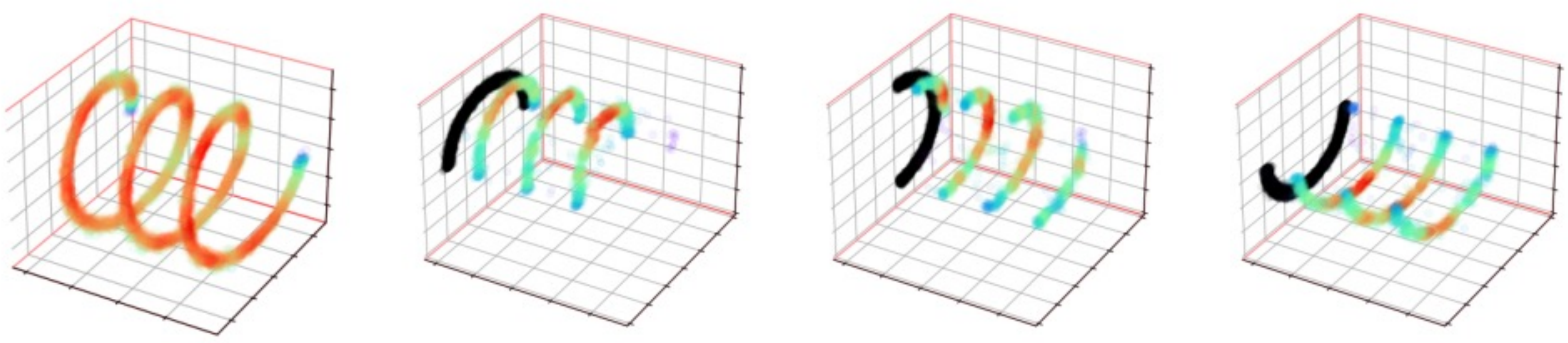


Image Inpainting

Can fill in missing data by sampling from conditional distributions



Can perform **controlled generation**

References

[1.] Choi, Y., Antonio Vergari, and Guy Van den Broeck. "Probabilistic circuits: A unifying framework for tractable probabilistic models." 2020.

[2.] Pevný, Tomáš, et al. "Sum-product-transform networks: Exploiting symmetries using invertible transformations." PGM 2020.

[3.] Peharz, Robert, et al. "Einsum networks: Fast and scalable learning of tractable probabilistic circuits." ICML 2020.

[4.] Dolatabadi, Hadi Mohaghegh, Sarah Erfani, and Christopher Leckie. "Invertible generative modeling using linear rational splines." AISTATS 2020.

[5.] Papamakarios, George, et al. "Normalizing flows for probabilistic modeling and inference." JMLR 2021.

Thank You! Questions?

https://starling.utdallas.edu

@starling_lab
@Sriraam_utd