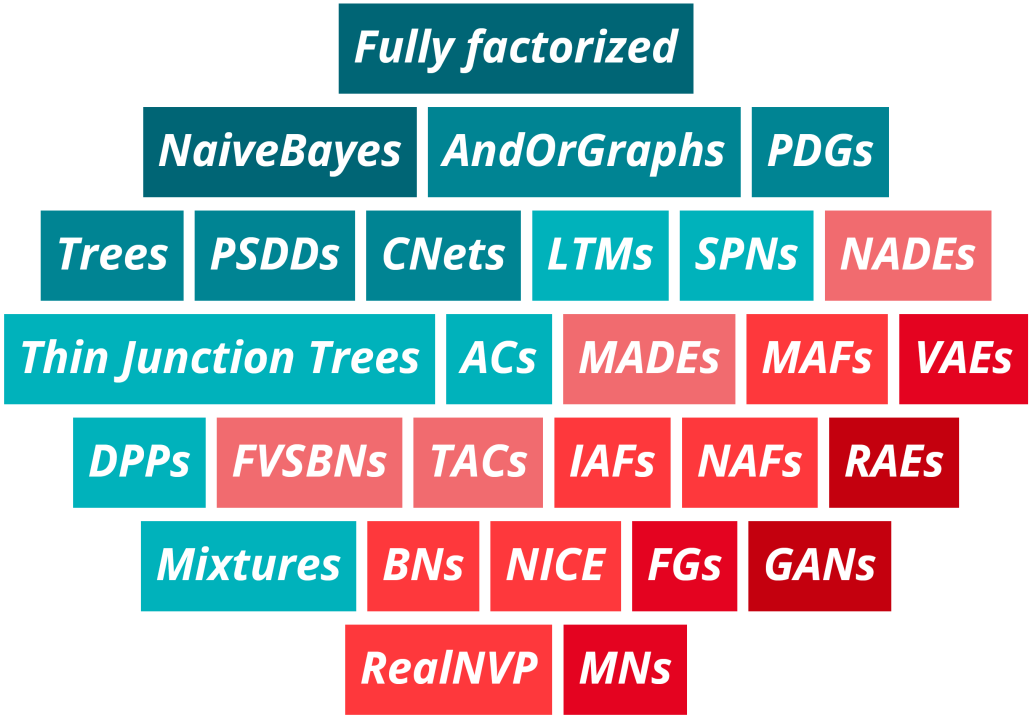
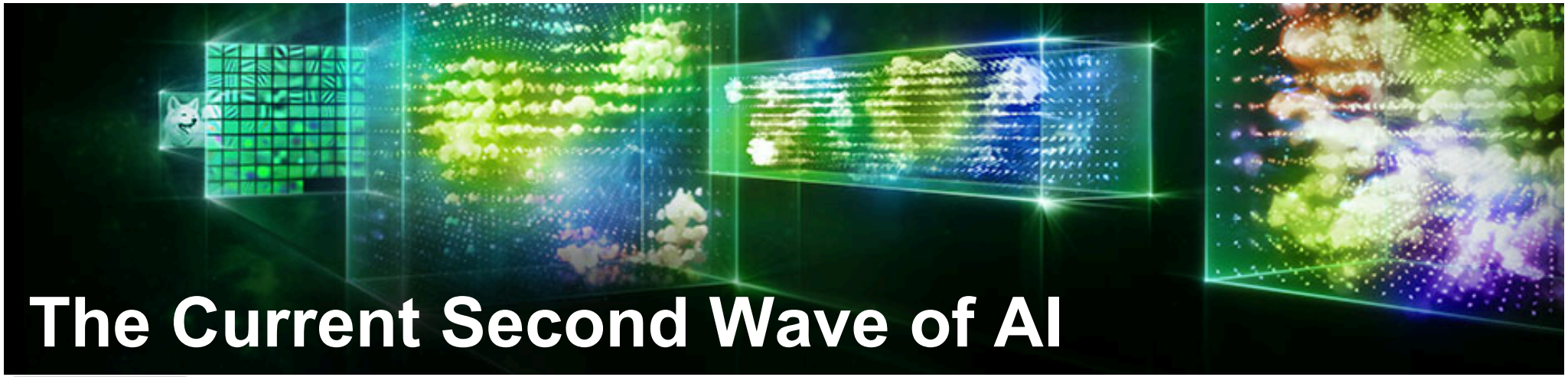
TECHNISCHE UNIVERSITÄT DARMSTADT

Tractable Probabilistic Models



Thanks to Pedro Domingos for making slides publically available
This is also based on joint work with Alejandro Molina, Antonio Vergari, Robert Peharz, Guy van den Broeck, Karl Stelzner and many others. Thanks!

tractability is a spectrum

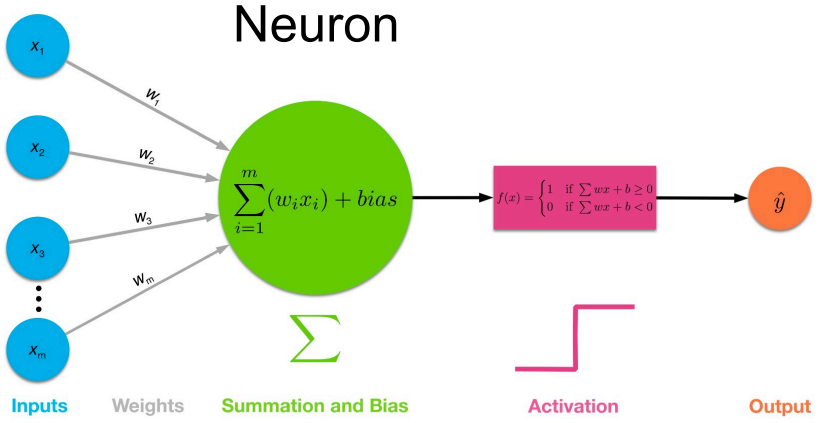


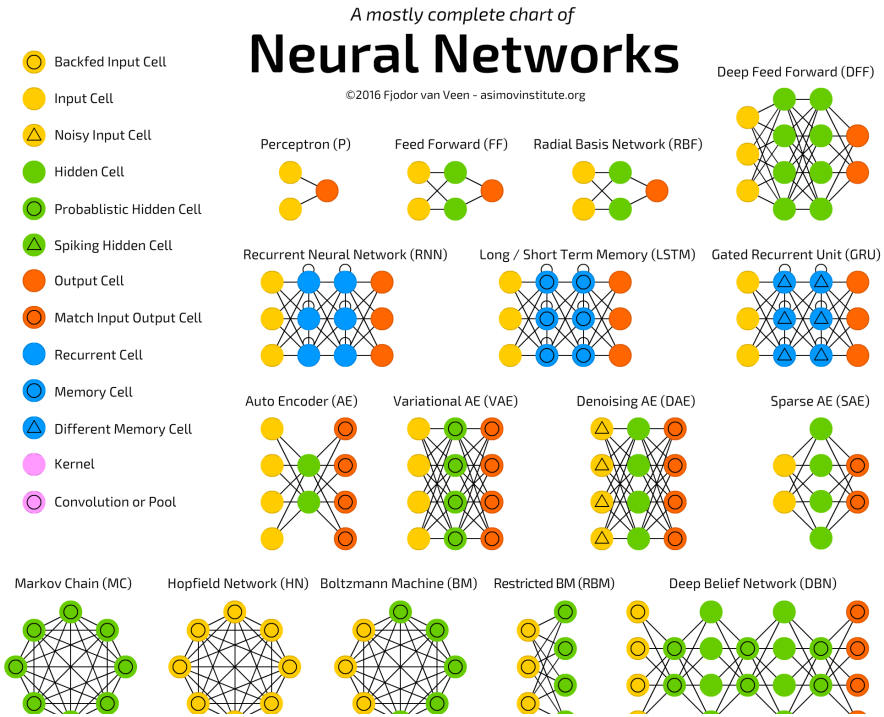
The Current Second Wave of AI

Deep Neural Networks

Potentially much more powerful than shallow architectures, represent computations

[LeCun, Bengio, Hinton Nature 521





Differentiable Programming

Deep Learning

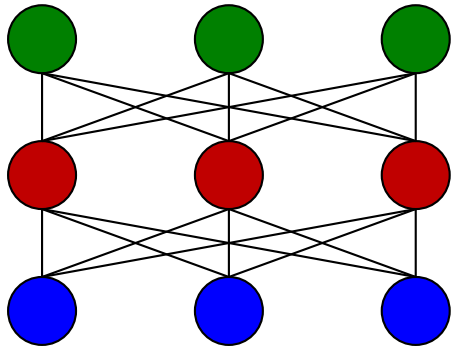
TECHNISCHE UNIVERSITÄT DARMSTADT

Stack many layers

E.g.: DBN [Hinton & Salakhutdinov,2006]

CDBN [Lee et al.,2009]

DBM [Salakhutdinov & Hinton,2010]



Potentially much more powerful than shallow architectures, represent computations [Bengio, 2009]

But …

- **Often no probabilistic semantics**
- **Learning requires extensive efforts**

DNNs often have no probabilistic semantics. They are not calibrated joint distributions.

$$P(Y|X) \neq P(Y,X)$$

MNIST

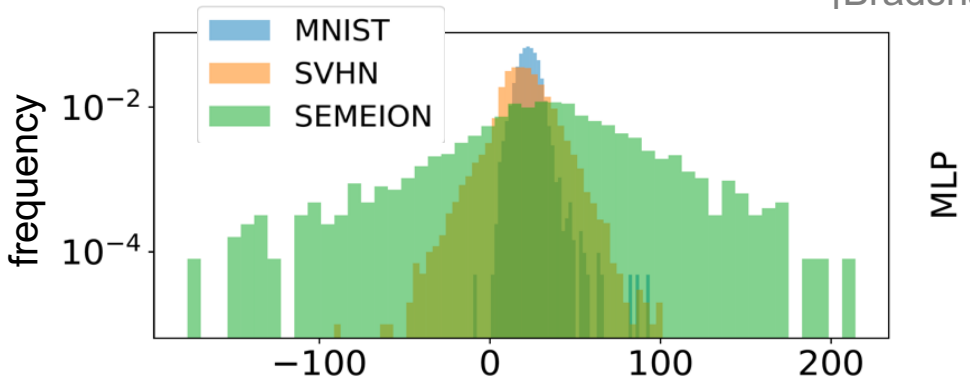
SVHN

SEMEION

Train & Evaluate

Transfer Testing

[Bradshaw et al. arXiv:1707.02476 2017]



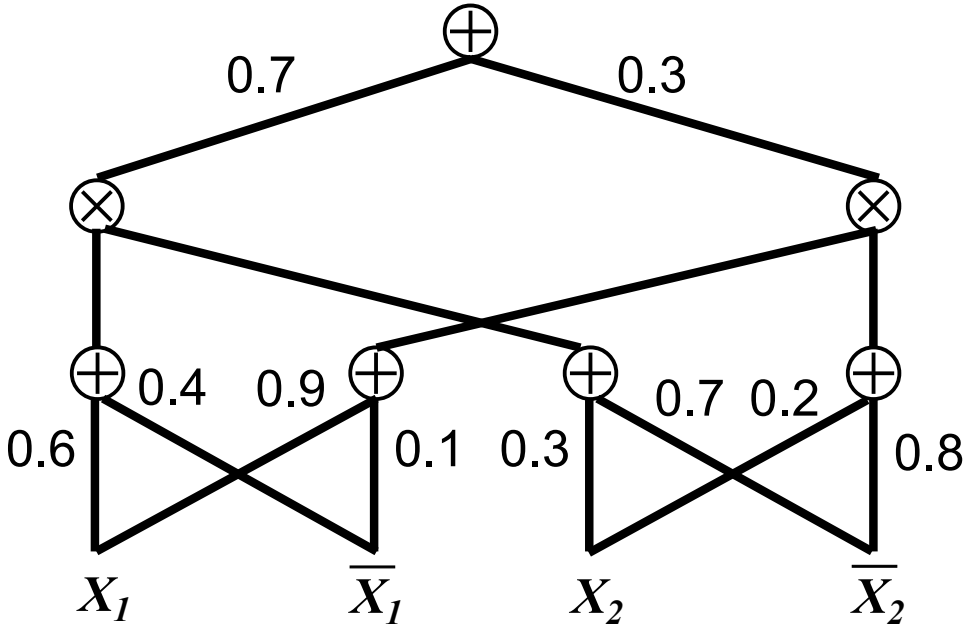
Many DNNs cannot distinguish the datasets

[Peharz, Vergari, Molina, Stelzner, Trapp, Kersting, Ghahramani UAI 2019]

UNIVERSITY OF CAMBRIDGE
Max Planck Institute for Intelligent Systems
TECHNISCHE UNIVERSITÄT DARMSTADT
UBER AI Labs
UNI GRAZ
Conference on Uncertainty in Artificial Intelligence
Tel Aviv, Israel
July 22 - 25, 2019
uai2019

Probabilistic Circuits

a deep probabilistic learning framework (here sum-product networks)



Computational graph (kind of TensorFlow graphs) that encodes how to compute probabilities

Inference is linear in size of network

Alternative Representation: Joint Distributions as Deep Networks

TECHNISCHE UNIVERSITÄT DARMSTADT

X_1	X_2	$P(X)$
1	1	0.4
1	0	0.2
0	1	0.1
0	0	0.3

$$P(X) = 0.4 \cdot \mathrm{I}[X_1{=}1] \cdot \mathrm{I}[X_2{=}1] + 0.2 \cdot \mathrm{I}[X_1{=}1] \cdot \mathrm{I}[X_2{=}0] + 0.1 \cdot \mathrm{I}[X_1{=}0] \cdot \mathrm{I}[X_2{=}1] + 0.3 \cdot \mathrm{I}[X_1{=}0] \cdot \mathrm{I}[X_2{=}0]$$

Alternative Representation: Joint Distributions as (Deep) Networks

TECHNISCHE UNIVERSITÄT DARMSTADT

X_1	X_2	$P(X)$
1	**1**	**0.4**
1	0	0.2
0	1	0.1
0	0	0.3

$$P(X) = \mathbf{0.4 \cdot I[X_1{=}1] \cdot I[X_2{=}1]} + 0.2 \cdot \mathrm{I}[X_1{=}1] \cdot \mathrm{I}[X_2{=}0] + 0.1 \cdot \mathrm{I}[X_1{=}0] \cdot \mathrm{I}[X_2{=}1] + 0.3 \cdot \mathrm{I}[X_1{=}0] \cdot \mathrm{I}[X_2{=}0]$$

Shorthand for Indicators

TECHNISCHE UNIVERSITÄT DARMSTADT

X_1	X_2	$P(X)$
1	1	0.4
1	0	0.2
0	1	0.1
0	0	0.3

$$
\begin{aligned}
P(X) = 0.4 \cdot X_1 \cdot X_2 \\
+ 0.2 \cdot X_1 \cdot \overline{X}_2 \\
+ 0.1 \cdot \overline{X}_1 \cdot X_2 \\
+ 0.3 \cdot \overline{X}_1 \cdot \overline{X}_2
\end{aligned}
$$

Sum Out Variables

TECHNISCHE UNIVERSITÄT DARMSTADT

X_1	X_2	$P(X)$
1	**1**	**0.4**
1	**0**	**0.2**
0	1	0.1
0	0	0.3

$$e: X_1 = 1$$

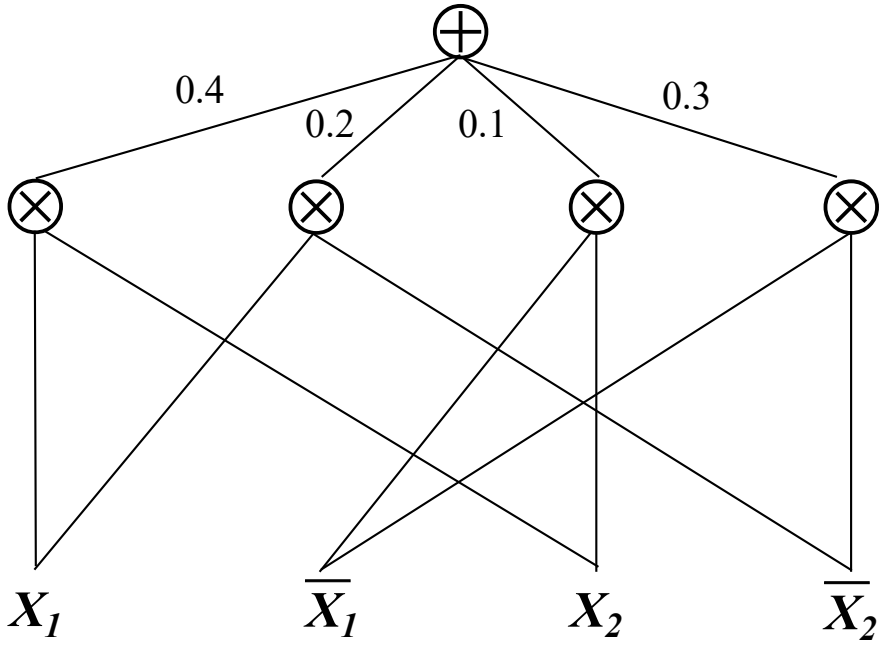
$$P(e) = \mathbf{0.4 \cdot X_1 \cdot X_2} + \mathbf{0.2 \cdot X_1 \cdot \overline{X}_2} + 0.1 \cdot \overline{X}_1 \cdot X_2 + 0.3 \cdot \overline{X}_1 \cdot \overline{X}_2$$

Set $X_1 = 1, \overline{X}_1 = 0, X_2 = 1, \overline{X}_2 = 1$

Easy: Set both indicators of X2 to 1

Idea: Deeper Network Representation of a Graphical Model that encodes how to compute probabilities

X_1	X_2	$P(X)$
1	1	0.4
1	0	0.2
0	1	0.1
0	0	0.3

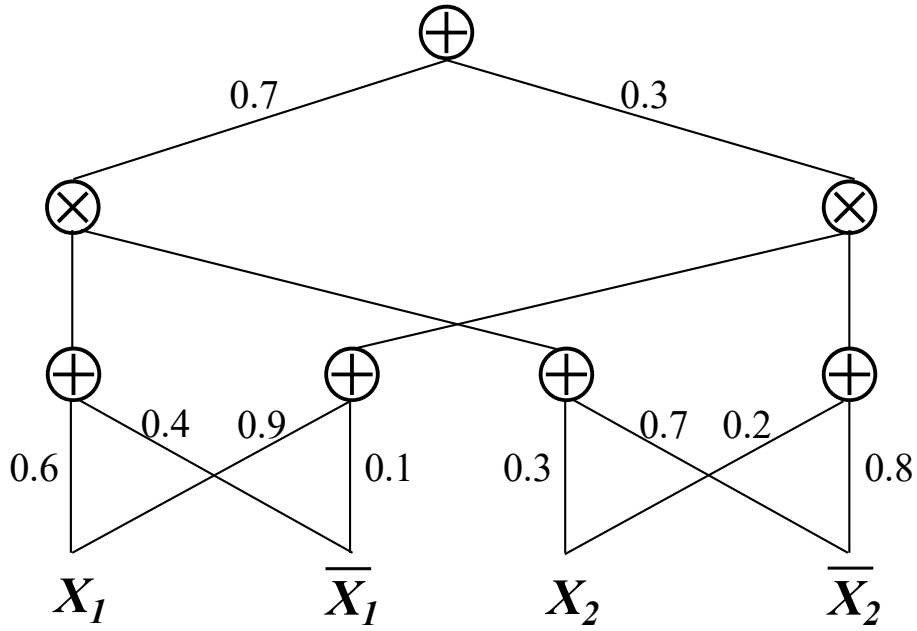


Sum-Product Networks* (SPNs)

[Poon, Domingos UAI 2011]

TECHNISCHE UNIVERSITÄT DARMSTADT

A SPN S is a rooted DAG where:
Nodes: Sum, product, input indicator
Weights on edges from sum to children



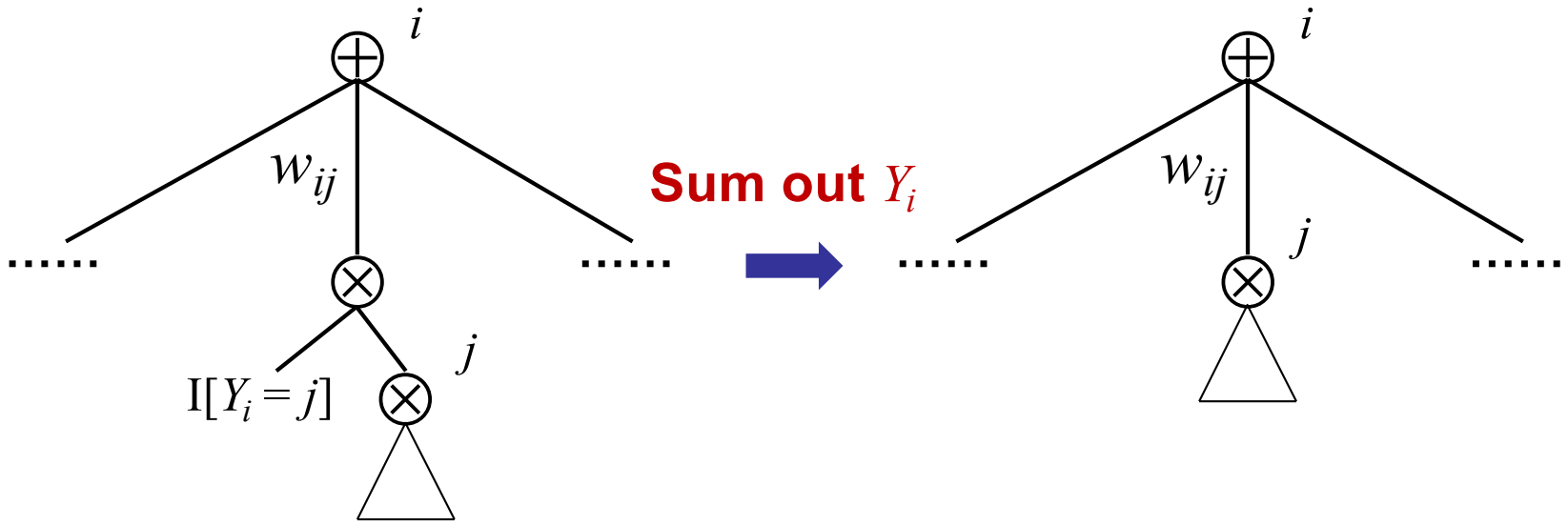
*SPNs are an instance of Arithmetic Circuits (ACs). ACs have been introduced into the AI literature more than15 years ago as a tractable representation of probability distributions

[Darwiche CACM 48(4):608-647 2001]

Semantics of Sums and Products

Product ~ Feature $\rightarrow$ Form feature hierarchy

Sum ~ Mixture (with hidden var. summed out)



Inference: Linear in Size of Network

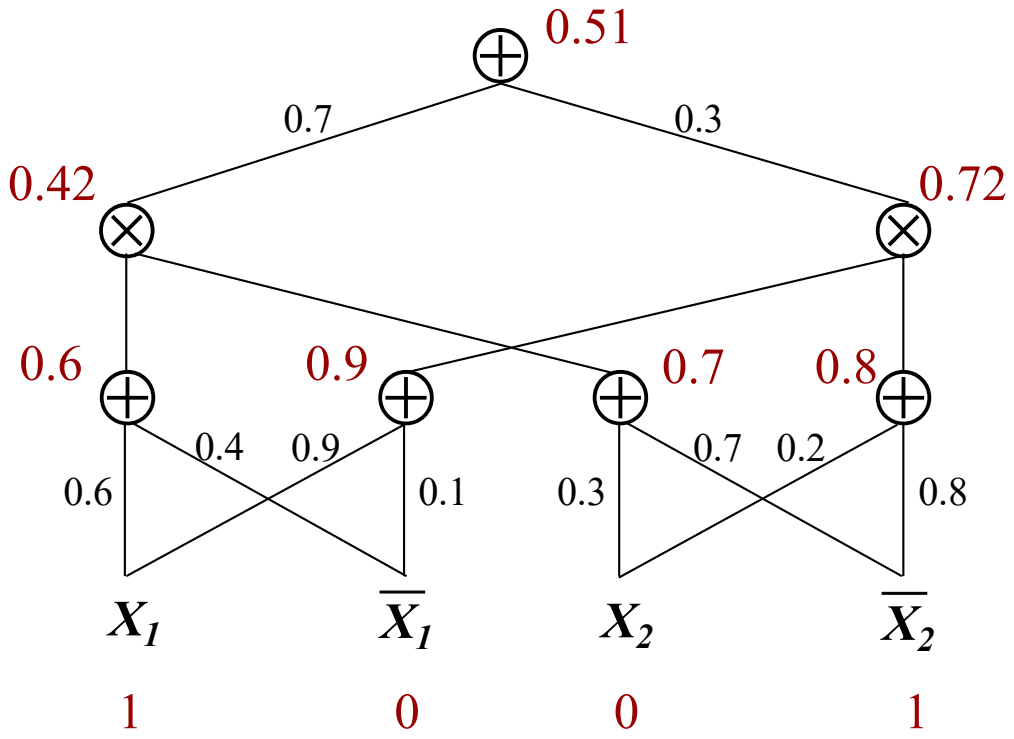
TECHNISCHE UNIVERSITÄT DARMSTADT

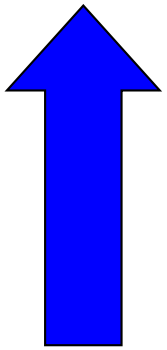
As long as weights sum to 1
at each sum node

$$P(X) = S(X)$$

X: $X_1 = 1, X_2 = 0$

X_1	1
$\overline{X}_1$	0
X_2	0
$\overline{X}_2$	1





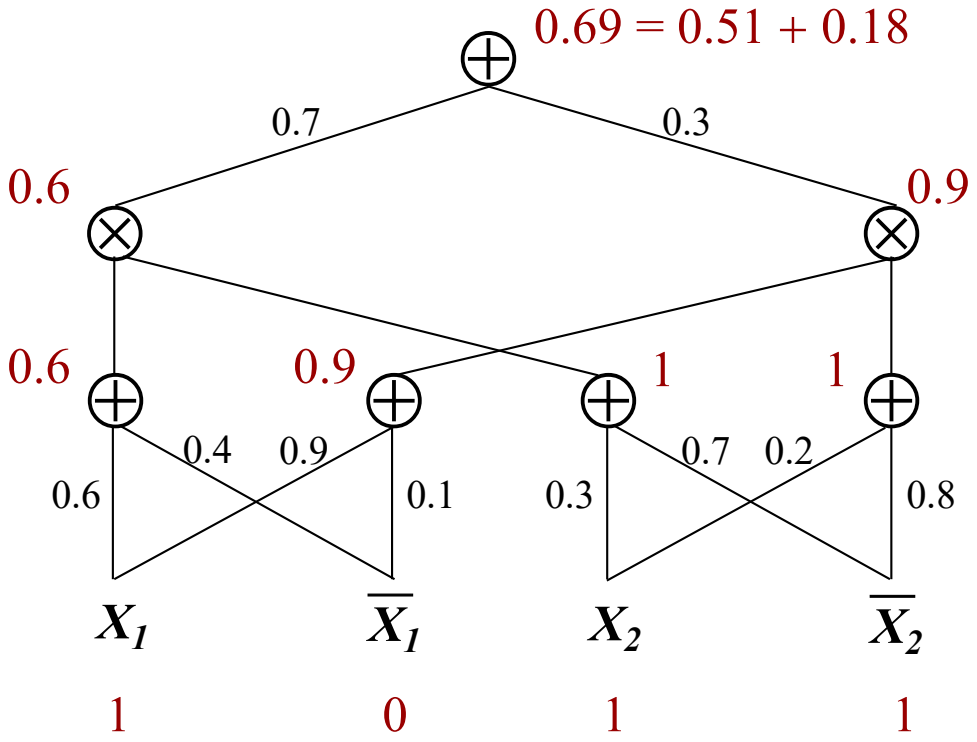
Inference: Linear in Size of Network

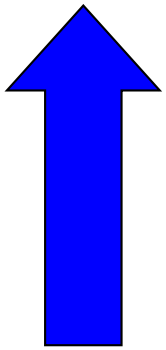
TECHNISCHE UNIVERSITÄT DARMSTADT

Marginal: $P(e) = S(e)$

e: $X_1 = 1$

X_1	1
$\overline{X}_1$	0
X_2	1
$\overline{X}_2$	1





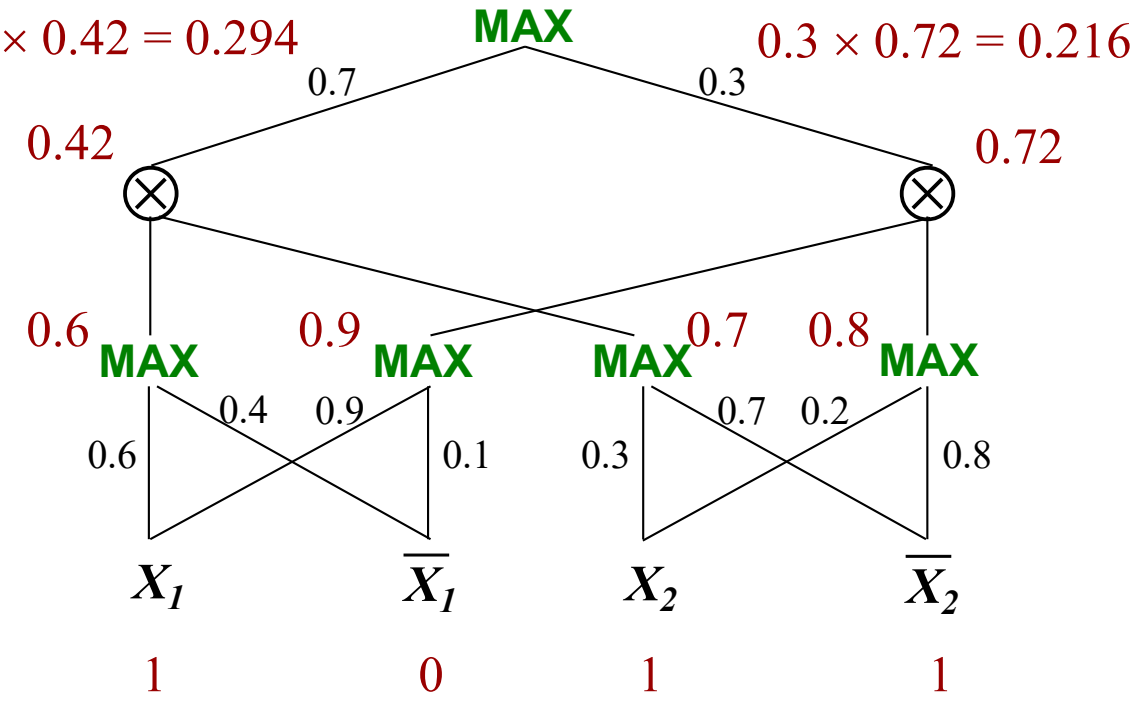
Inference: Linear in Size of Network

TECHNISCHE UNIVERSITÄT DARMSTADT

MAP: Replace sums with maxs

e: $X_1 = 1$

X_1	1
$\overline{X}_1$	0
X_2	1
$\overline{X}_2$	1



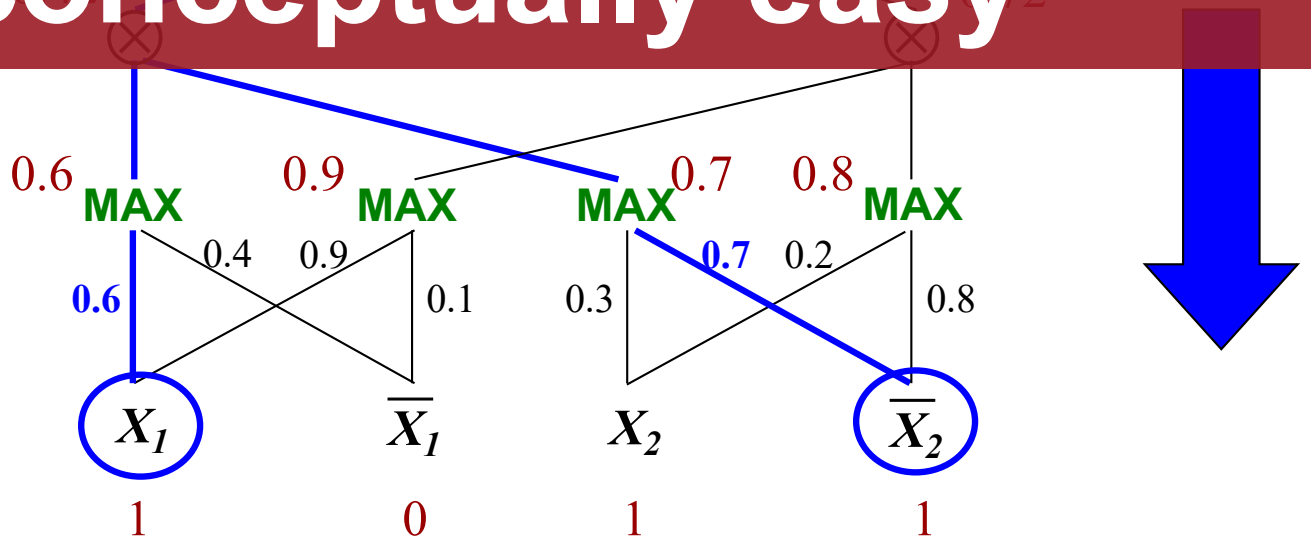
Inference: Linear in Size of Network

MAX: Pick child with highest value

MAP State: $X_1 = 1, X_2 = 0$

And also learning is conceptually easy

X_1	1
$\overline{X}_1$	0
X_2	1
$\overline{X}_2$	1



General Approach via Parameter Estimation assuming a fixed network (like in Deep Learning)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Start with a dense SPN
- Find the structure by (online) learning weights
  Zero weights signify absence of connections
- (Hard) EM beneficial to avoid gradient vanishing
  Each sum node is a mixture over children

In principle you can turn a given SPN into a TensorFlow computation graph and apply any known algorithm from there

Image Completion

TECHNISCHE UNIVERSITÄT DARMSTADT

Main evaluation: Caltech-101 [Fei-Fei et al., 2004]

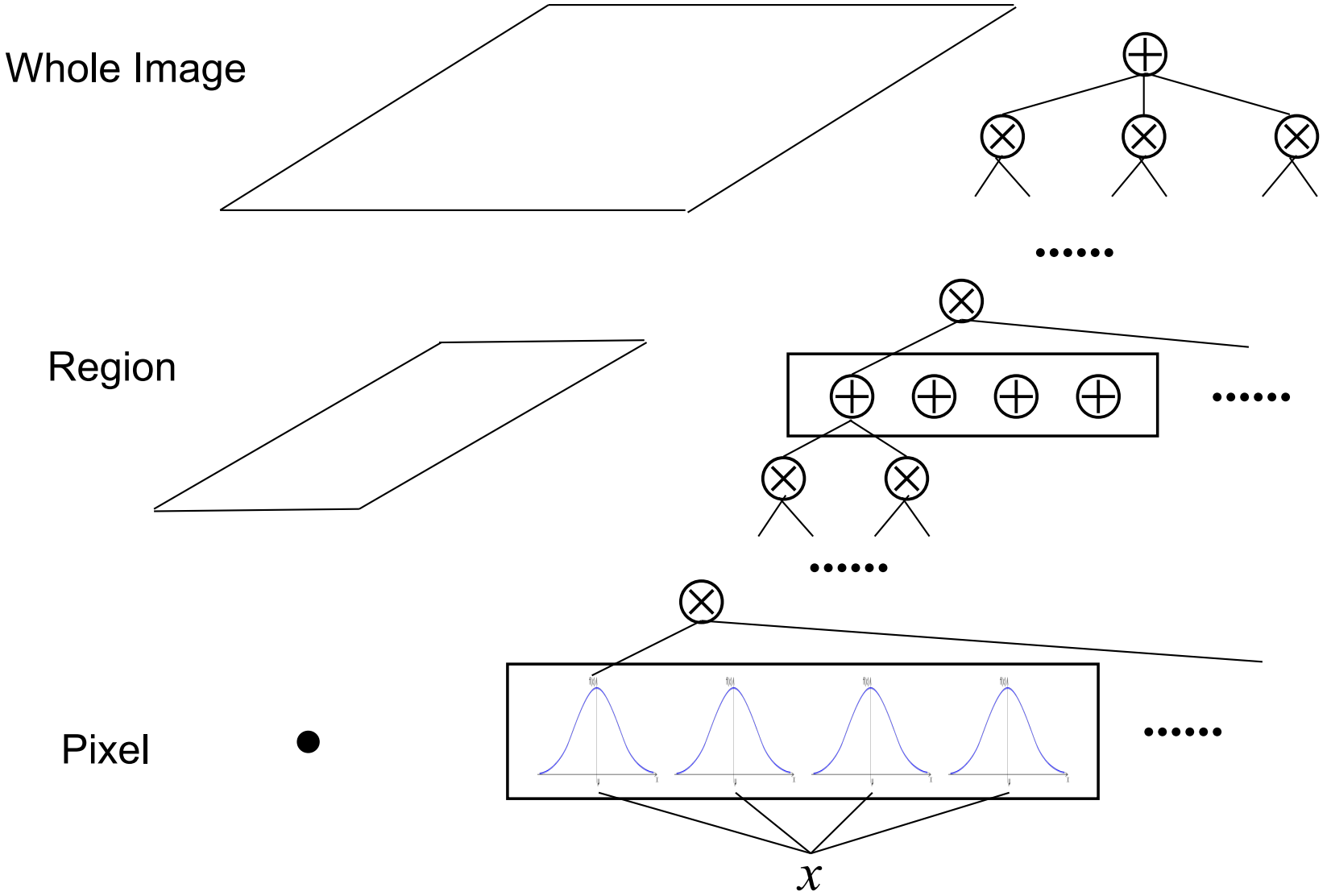
- 101 categories, e.g., faces, cars, elephants
- Each category: 30 – 800 images

Also, Olivetti [Samaria & Harter, 1994] (400 faces)

Each category: Last third for test

Test images: Unseen objects

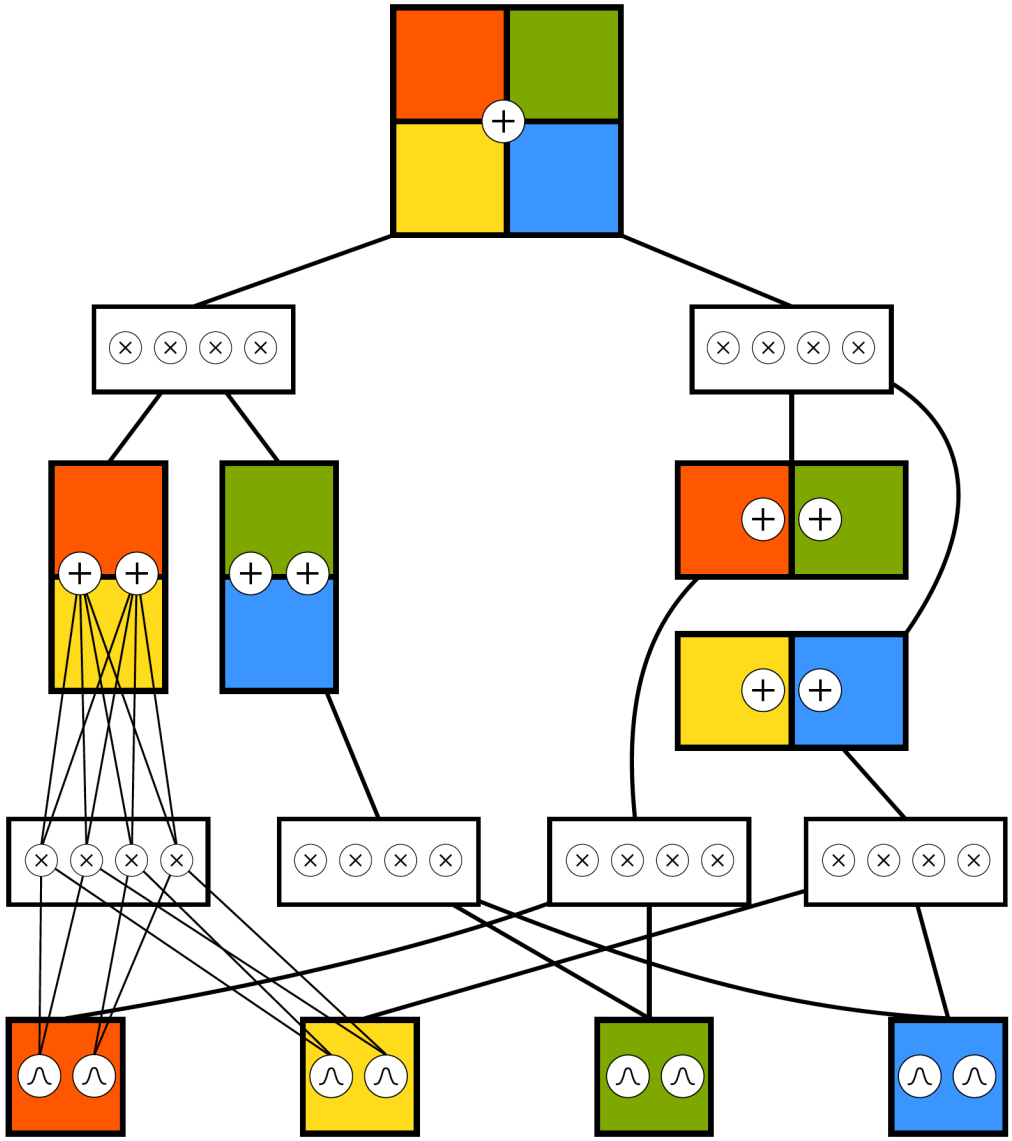
SPN Architecture



SPN Architecture

Whole Image

Regions



Caltech: Mean-Square Errors

TECHNISCHE UNIVERSITÄT DARMSTADT

	LEFT	BOTTOM
SPN	**3551**	**3270**
DBM	9043	9792
DBN	4778	4492
PCA	4234	4465
Nearest Neighbor	4887	5505

SPN vs. DBM / DBN

TECHNISCHE UNIVERSITÄT DARMSTADT

SPN is order of magnitude faster

	SPN	DBM / DBN
Learning	2-3 hours	Days
Inference	< 1 second	Minutes or hours

No elaborate preprocessing, tuning

Reduced errors by 30-60%

Learned up to 46 layers

Example Completions

TECHNISCHE UNIVERSITÄT DARMSTADT

Original

SPN

DBM

DBN

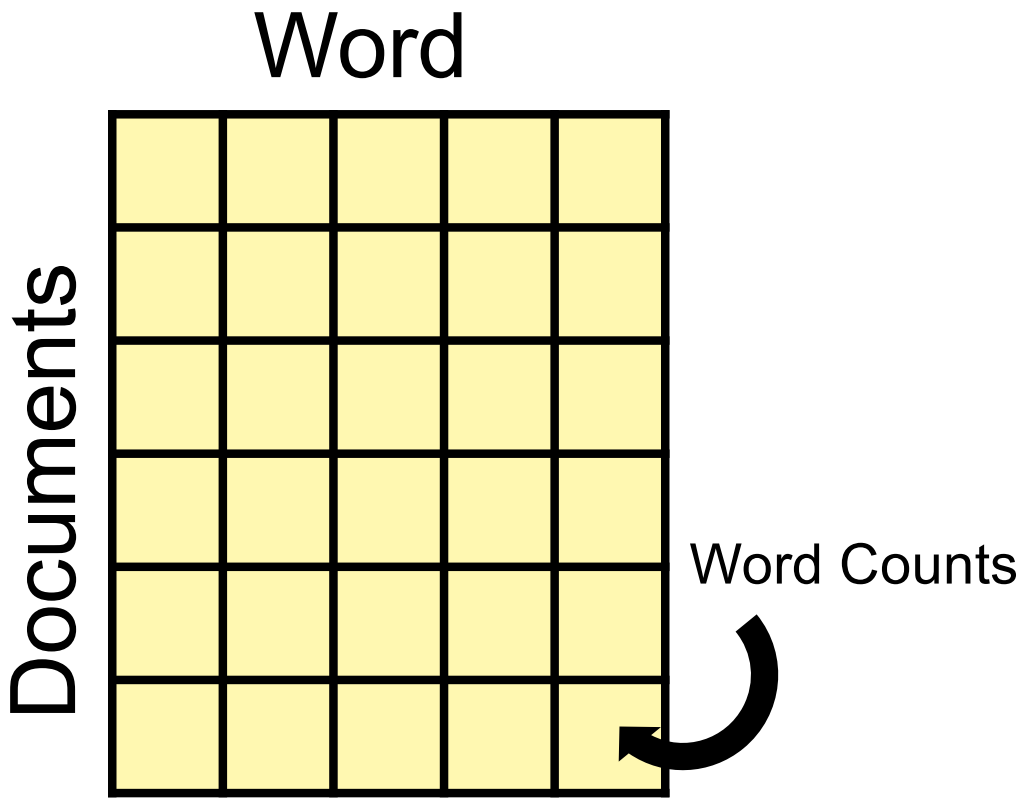
PCA

Nearest Neighbor

Or we learn directly (Tree-)SPNs

TECHNISCHE UNIVERSITÄT DARMSTADT

Testing independence of
Poisson random variables



[Poon, Domingos UAI'11; Molina, Natarajan, Kersting AAAI´17]

Or we learn directly (Tree-)SPNs

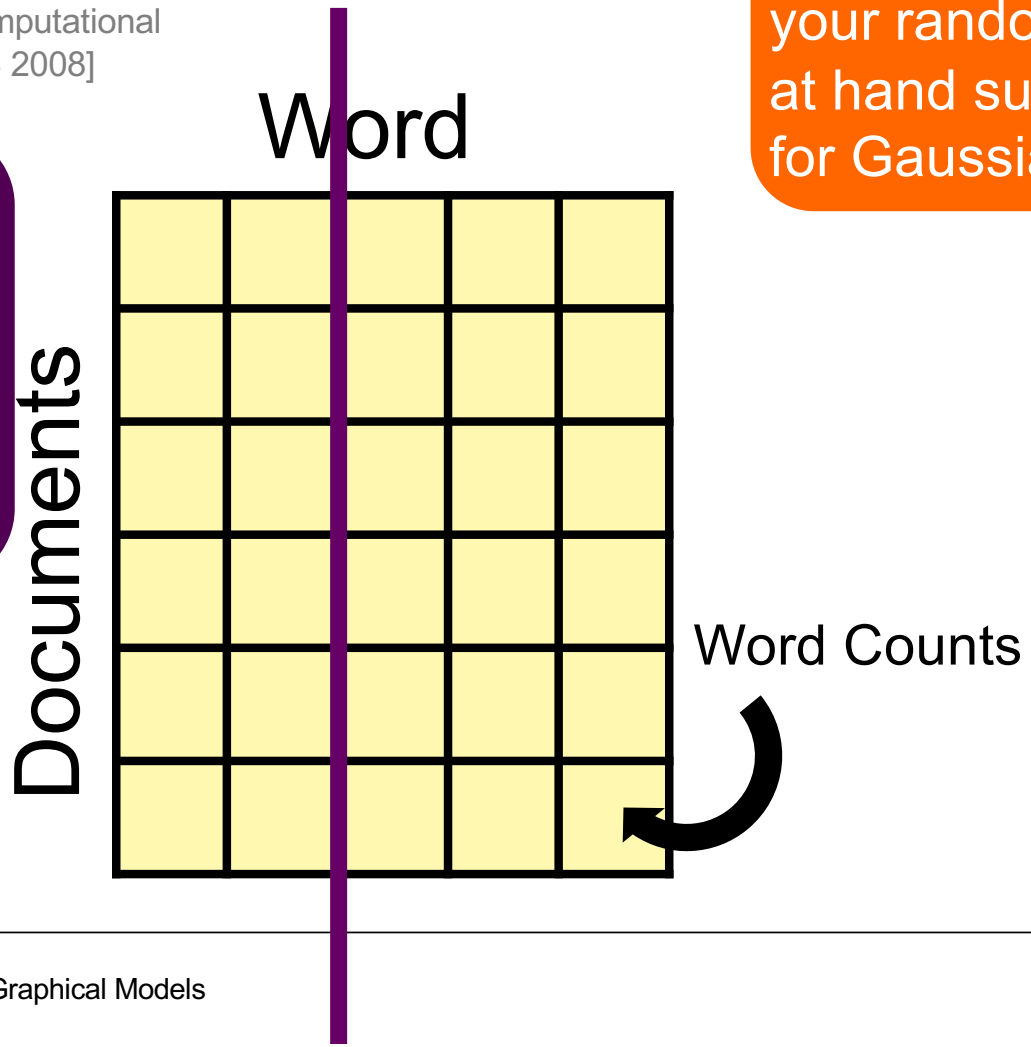
TECHNISCHE UNIVERSITÄT DARMSTADT

Testing independence of Poisson random variables

[Zeileis, Hothorn, Hornik Journal of Computational And Graphical Statistics 17(2):492–514 2008]

In general use the independency test for your random variables at hand such as g-test for Gaussians

Learn Poisson model trees for P(x|V-x) and P(y|V-y). Check whether X resp. Y is significant in P(y|V-x) resp. P(x|V-y)

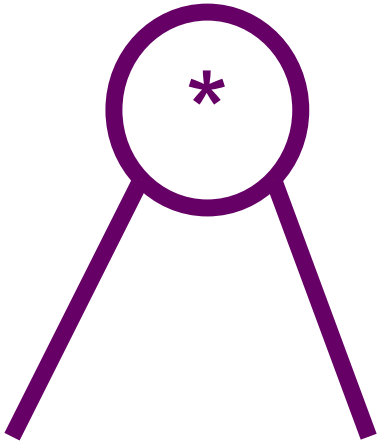


[Poon, Domingos UAI'11; Molina, Natarajan, Kersting AAAI'17]

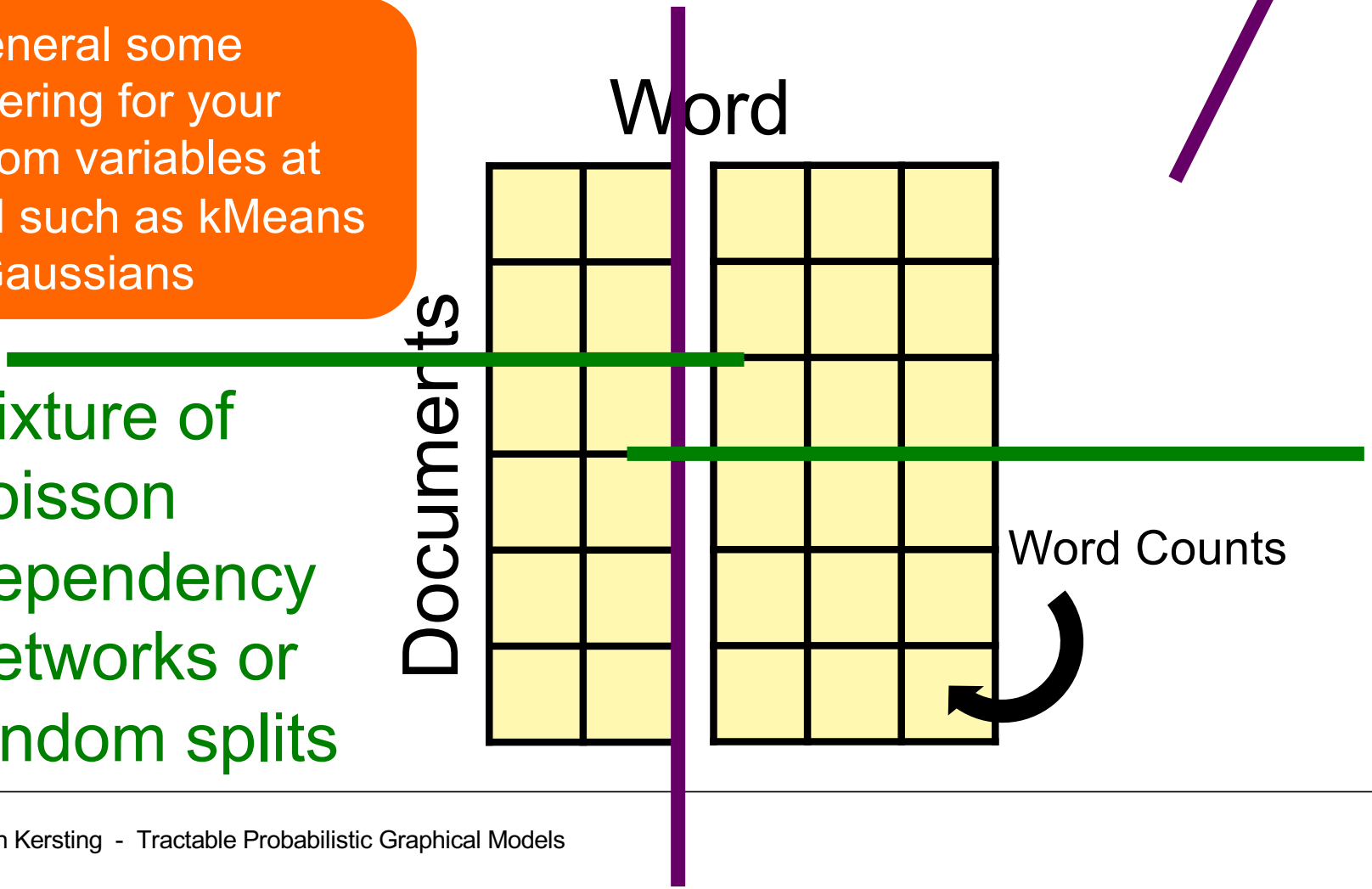
Or we learn directly (Tree-)SPNs

TECHNISCHE UNIVERSITÄT DARMSTADT

Testing independence of Poisson random variables



In general some clustering for your random variables at hand such as kMeans for Gaussians



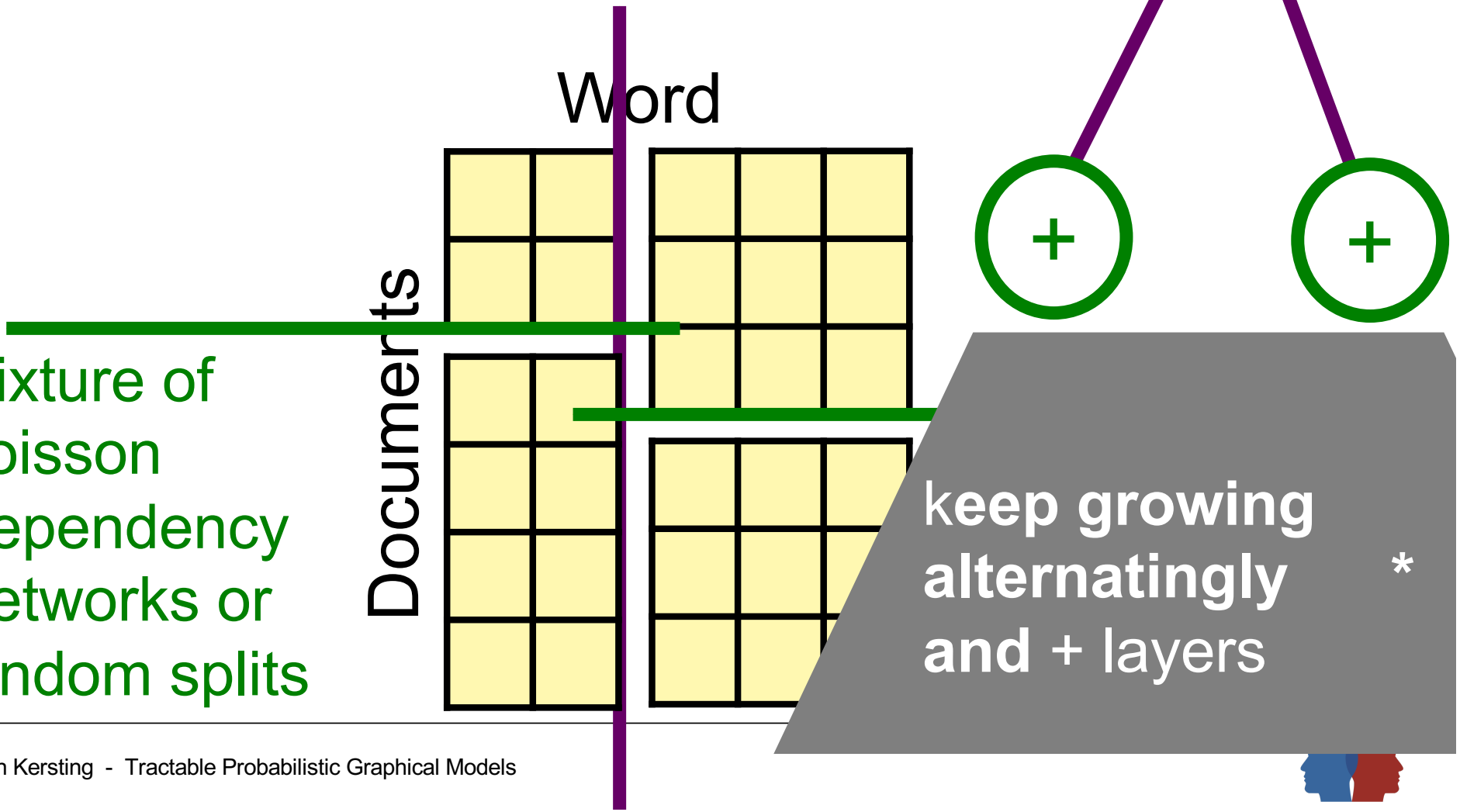
Mixture of Poisson Dependency Networks or random splits

[Poon, Domingos UAI'11; Molina, Natarajan, Kersting AAAI`17]

Or we learn directly (Tree-)SPNs

TECHNISCHE UNIVERSITÄT DARMSTADT

Testing independence of Poisson random variables



[Poon, Domingos UAI'11; Molina, Natarajan, Kersting AAAI'17; Vergari, Peharz, Di Mauro, Molina, Kersting, Esposito AAAI '18; Molina, Vergari, Di Mauro, Esposito, Natarajan, Kersting AAAI '18, Peharz et al. UAI 2019, Stelzner, Peharz, Kersting iCML 2019]

SPFlow: An Easy and Extensible Library for Sum-Product Networks

[Molina, Vergari, Stelzner, Peharz, Subramani, Poupart, Di Mauro, Kersting arXiv:1901.03704, 2019]

TECHNISCHE UNIVERSITÄT DARMSTADT · UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO · UNIVERSITY OF WATERLOO · Max Planck Institute for Intelligent Systems · UNIVERSITY OF CAMBRIDGE · VECTOR INSTITUTE

CAML · MADESI · DFG · Federal Ministry of Education and Research

195 commits · 2 branches · 0 releases · 6 contrib...

Branch: master · New pull request · Create new file · Upload files · Find file · Clone or download

https://github.com/SPFlow/SPFlow

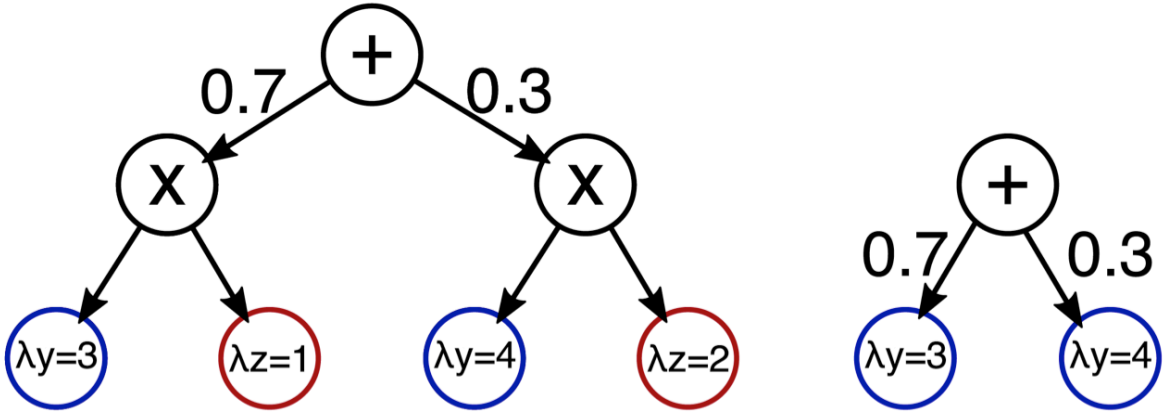
```
from spn.structure.leaves.parametric.Parametric import Categorical

from spn.structure.Base import Sum, Product

from spn.structure.base import assign_ids, rebuild_scopes_bottom_up


p0 = Product(children=[Categorical(p=[0.3, 0.7], scope=1), Categorical(p=[0.4, 0.6], scope=2)])
p1 = Product(children=[Categorical(p=[0.5, 0.5], scope=1), Categorical(p=[0.6, 0.4], scope=2)])
s1 = Sum(weights=[0.3, 0.7], children=[p0, p1])
p2 = Product(children=[Categorical(p=[0.2, 0.8], scope=0), s1])
p3 = Product(children=[Categorical(p=[0.2, 0.8], scope=0), Categorical(p=[0.3, 0.7], scope=1)])
p4 = Product(children=[p3, Categorical(p=[0.4, 0.6], scope=2)])
spn = Sum(weights=[0.4, 0.6], children=[p2, p4])

assign_ids(spn)
rebuild_scopes_bottom_up(spn)

return spn
```

Domain Specific Language, Inference, EM, and Model Selection as well as Compilation of SPNs into TF and PyTorch and also into flat, library-free code even suitable for running on devices: C/C++,GPU, FPGA

SPFlow, an open-source Python library providing a simple interface to inference, learning and manipulation routines for deep and tractable probabilistic models called Sum-Product Networks (SPNs). The library allows one to quickly create SPNs both from data and through a domain specific language (DSL). It efficiently implements several probabilistic inference routines like computing marginals, conditionals and (approximate) most probable explanations (MPEs) along with sampling

Inference on Devices

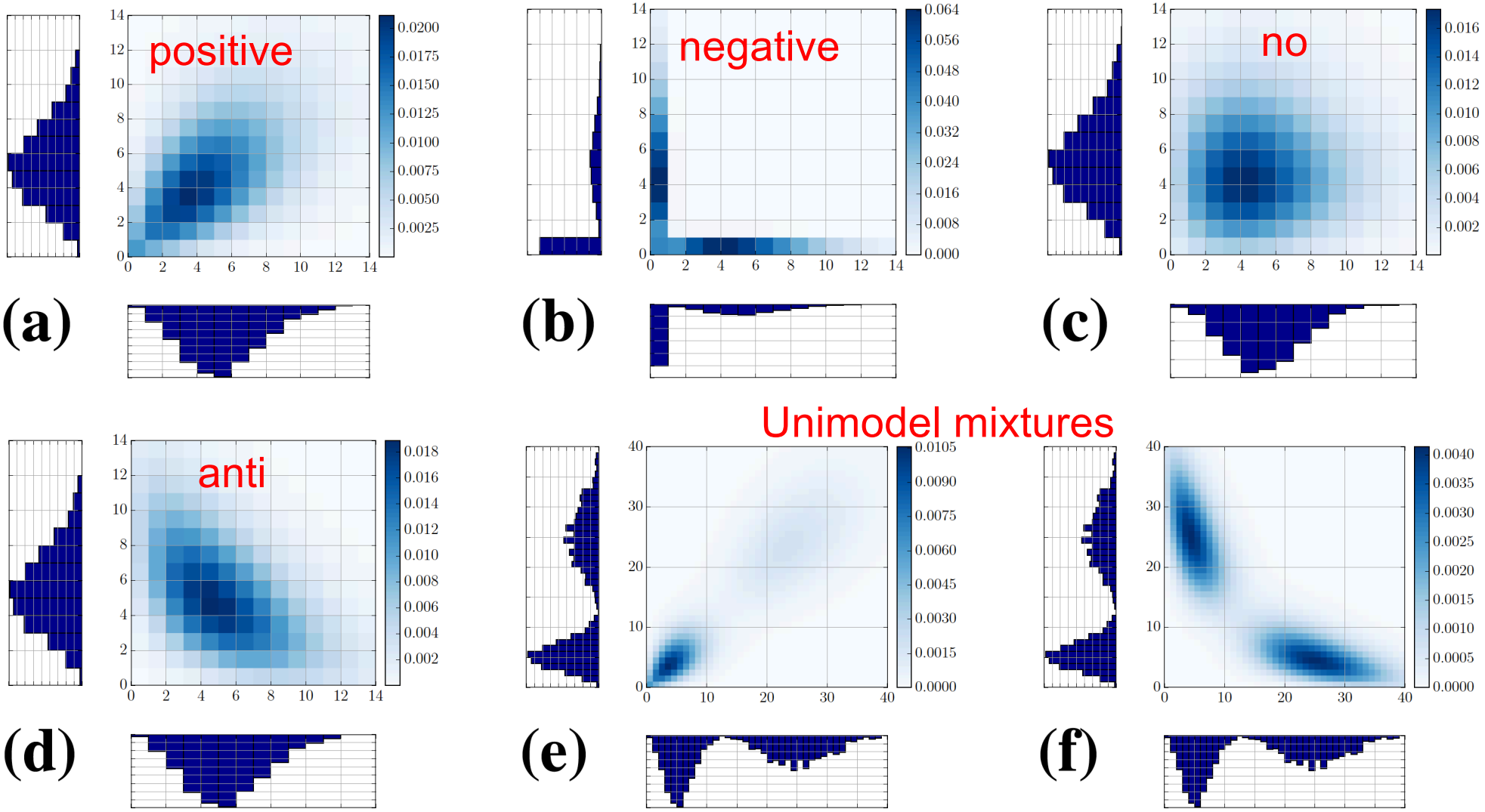
TECHNISCHE UNIVERSITÄT DARMSTADT



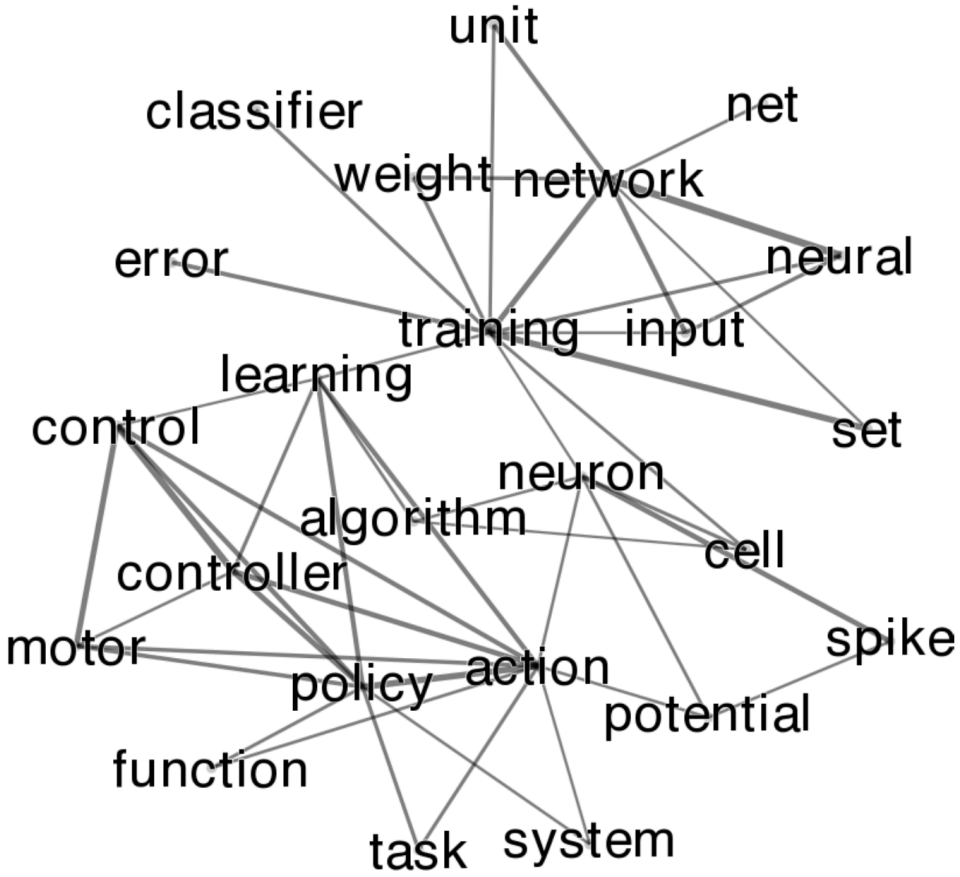
Ultimately, they may even be compiled into simple, flat, library-free code suitable for embedding in real-time applications and devices

SPNs encode polynomials over univariate distributions to build multivariate distributions. They can be fed into symbolic evaluation methods for inference

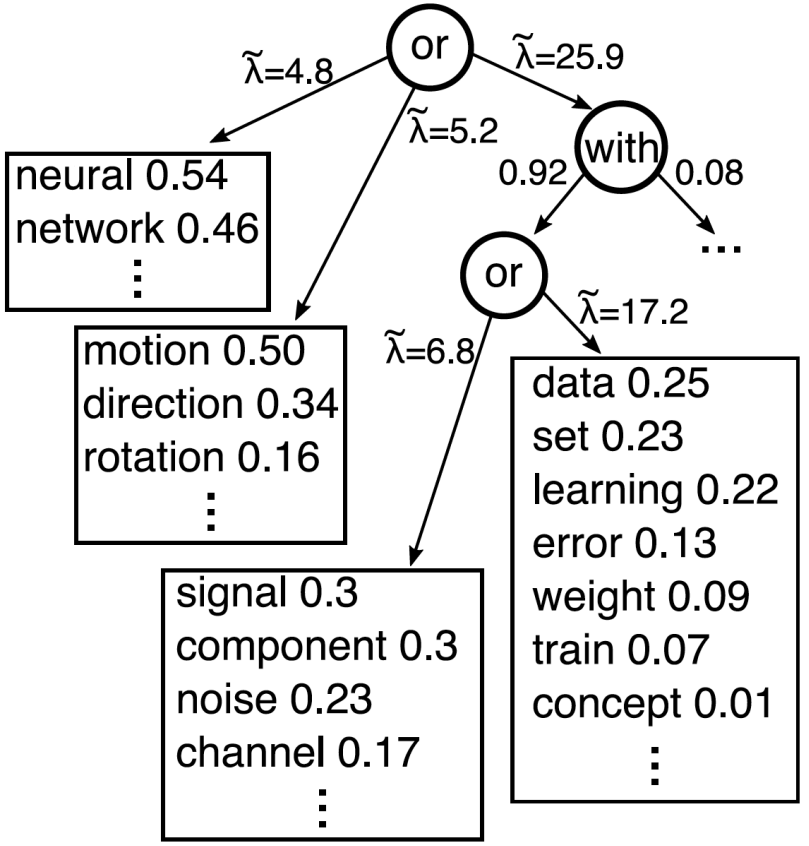
Flexible Dependencies even for challenging distributions, here multivariate Poisson distribution



Multiple views on you data due to efficient inference



Mutual Information
(NIPS corpus)



Poisson Multinomial SPN
= hierachical topic model

Poisson SPNs provide topics:
From Topic Models to Poisson Mean Fields

[Bishop, Fienberg, Holland. Discrete Multivariate Analysis: Theory and Practice. Springer, 2007]

Topic-word distributions of LDA can be viewed as a product of independent Poissons

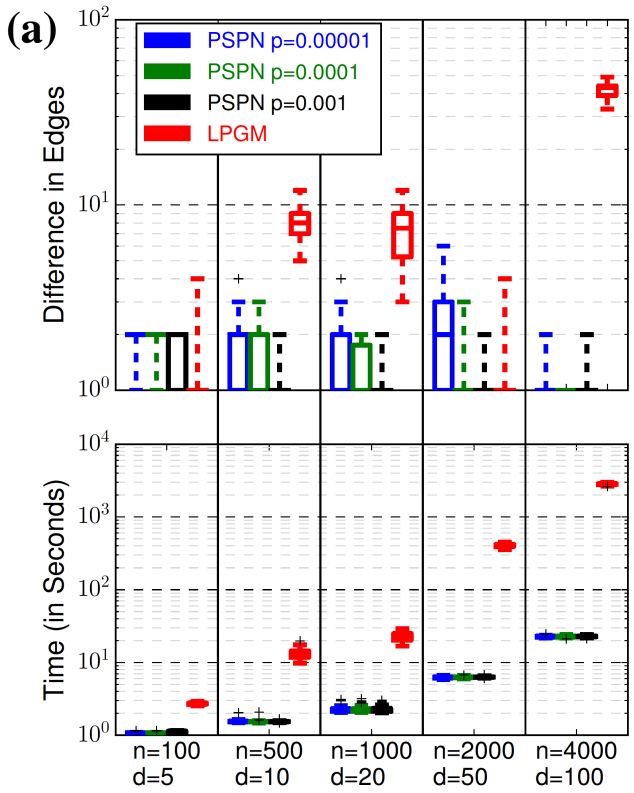
$$\begin{aligned}
&\Pr_{\text{Poiss}}\left(\tilde{x}\,|\,\tilde{\lambda}\right)\Pr_{\text{Mult}}\left(\mathbf{x}\,|\,\theta=(\lambda_1,\cdots,\lambda_p)\,/\tilde{\lambda},\mathsf{N}=\tilde{x}\right)\\
&\quad=\frac{e^{-\tilde{\lambda}}}{\tilde{x}!}\tilde{\lambda}^{\tilde{x}}\frac{\tilde{x}!}{\prod_{s=1}^{p}x_s!}\prod_{s=1}^{p}\left(\frac{\lambda_s}{\tilde{\lambda}}\right)^{x_s}\\
&\quad=\frac{\tilde{x}!}{\tilde{x}!}\frac{e^{-\tilde{\lambda}}}{\prod_{s=1}^{p}x_s!}\prod_{s=1}^{p}\left(\frac{\tilde{\lambda}\lambda_s}{\tilde{\lambda}}\right)^{x_s}\\
&\quad=\Pr_{\text{Ind. Poiss}}(\mathbf{x}\,|\,\lambda_1,\cdots,\lambda_p)=\prod_{s=1}^{p}\frac{e^{-\lambda_s}}{x_s!}\lambda_s^{x_s}
\end{aligned}$$

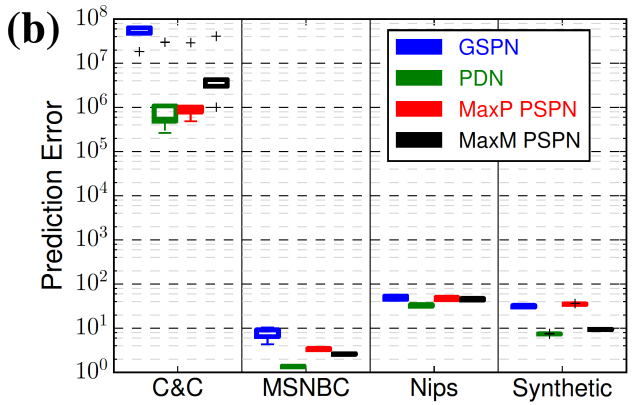
Competitive Predictive Performance

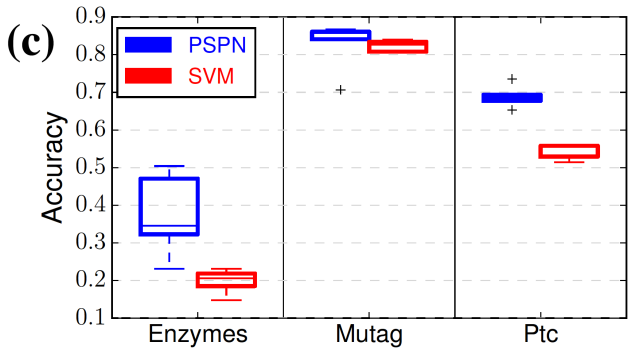
TECHNISCHE UNIVERSITÄT DARMSTADT

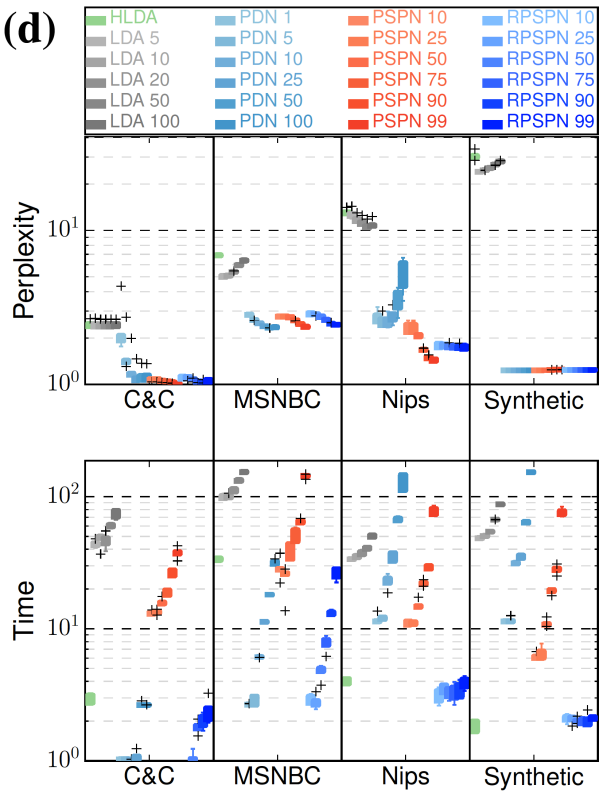
Prediction error

LDA









Network discovery

Deep Graph Kernels (Weisfeiler Lehman)

Random sum-product networks

TECHNISCHE UNIVERSITÄT DARMSTADT

UNIVERSITY OF CAMBRIDGE

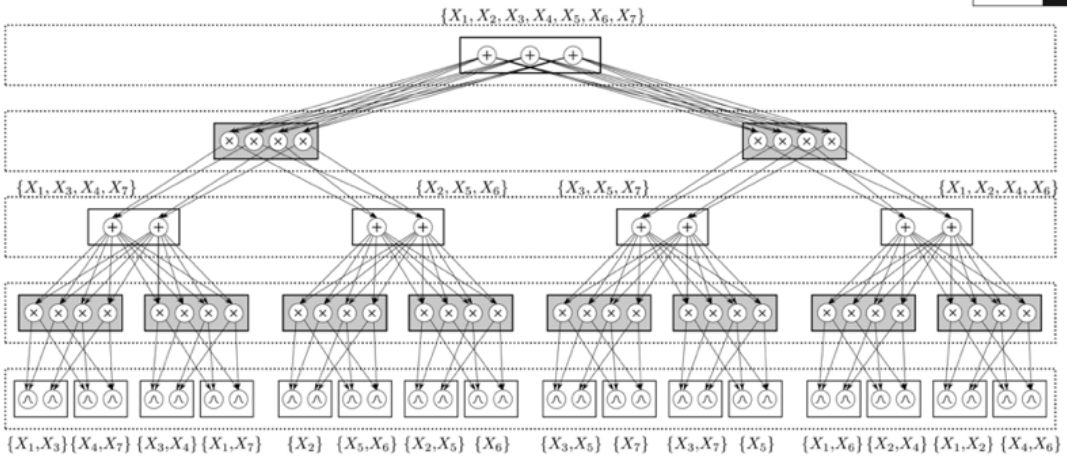
Max Planck Institute for Intelligent Systems

UBER AI Labs

TECHNISCHE UNIVERSITÄT DARMSTADT

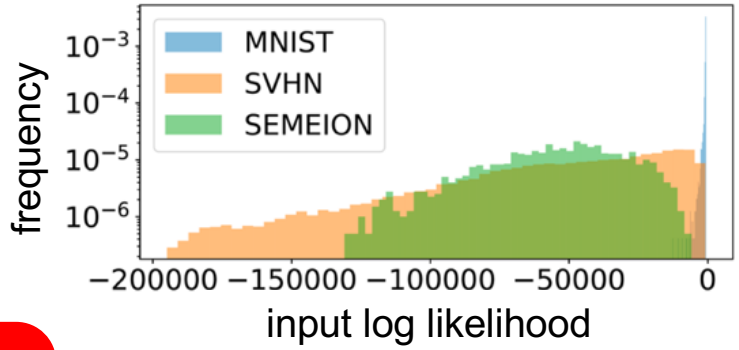
UNI GRAZ

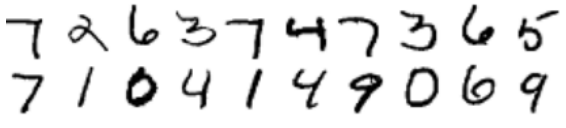
Conference on Uncertainty in Artificial Intelligence
Tel Aviv, Israel
July 22 - 25, 2019
uai2019



Similar to Random Forests, build a random SPN structure. This can be done in an informed way or completely at random

		RAT-SPN	MLP	vMLP
Accuracy	MNIST	98.19 (8.5M)	98.32 (2.64M)	98.09 (5.28M)
	F-MNIST	89.52 (0.65M)	90.81 (9.28M)	89.81 (1.07M)
	20-NG	47.8 (0.37M)	49.05 (0.31M)	48.81 (0.16M)
Cross-Entropy	MNIST	0.0852 (17M)	0.0874 (0.82M)	0.0974 (0.22M)
	F-MNIST	0.3525 (0.65M)	0.2965 (0.82M)	0.325 (0.29M)
	20-NG	1.6954 (1.63M)	1.6180 (0.22M)	1.6263 (0.22M)





outliers
prototypes

outliers
prototypes

SPNs can have similar predictive performances as (simple) DNNs

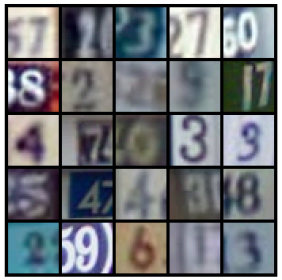
SPNs can distinguish the datasets

SPNs know when they do not know by design

Einsum Networks

[Peharz, Lang, Vergari, Stelzner, Molina, Trapp, Van den Broeck, Kersting, Ghahramani ICML 2020]

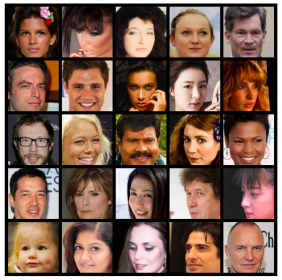
ICML | 2020
Thirty-seventh International Conference on Machine Learning



(a) Real SVHN images.

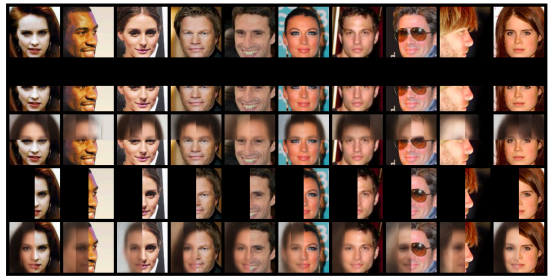
(b) EiNet SVHN samples.

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



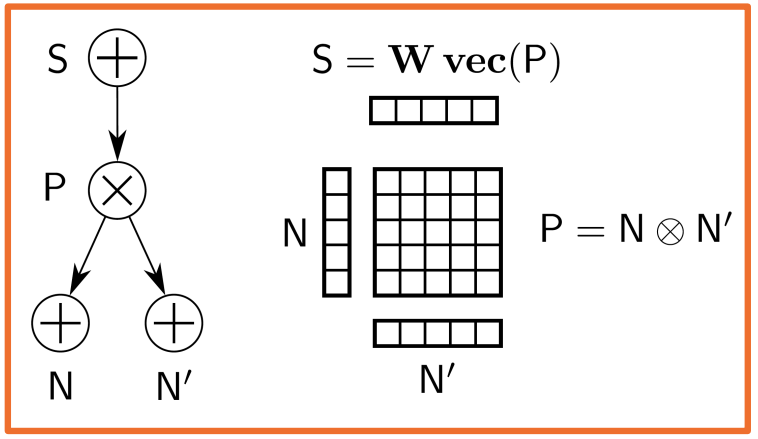
(d) Real Celeba samples.

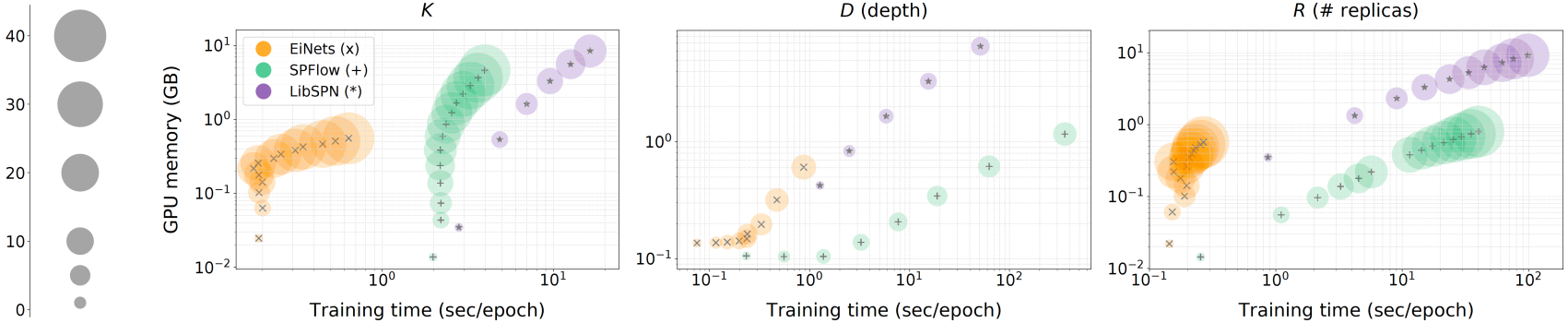
(e) EiNet Celeba samples.



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

Compilation of PCs into efficient to evaluate computational graphs





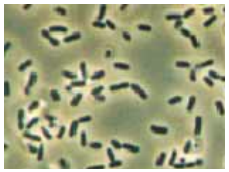
Unsupervised scene understanding

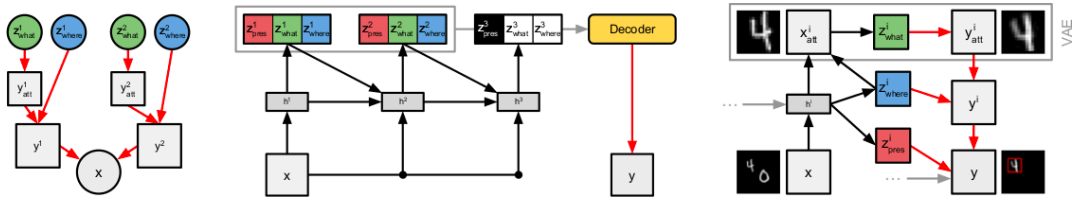
[Stelzner, Peharz, Kersting ICML 2019, Best Paper Award at TPM@ICML2019] https://github.com/stelzner/supair

UNIVERSITY OF CAMBRIDGE | TECHNISCHE UNIVERSITÄT DARMSTADT

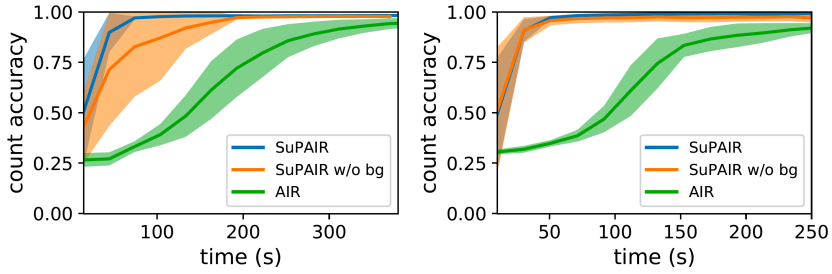
ICML | 2019
Thirty-sixth International Conference on Machine Learning

Consider e.g. unsupervised scene understanding using a generative model implemented in a neural fashion

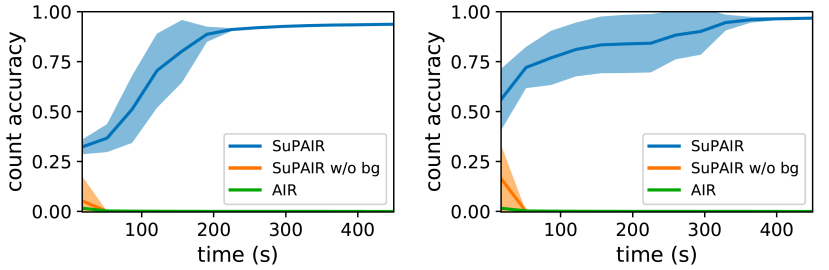




[Attend-Infer-Repeat (AIR) model, Hinton et al. NIPS 2016]



(a) MNIST (b) Sprites

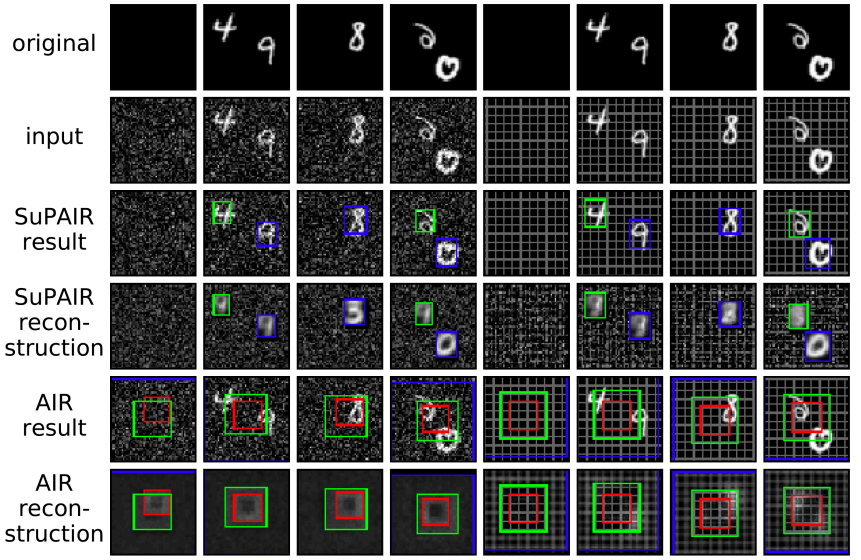


(c) Noisy MNIST (d) Grid MNIST

Replace VAE by SPN as object model

- infinite mixture model
- intractable density
- intractable posterior

- "large" but finite mixture model
- tractable density
- tractable marginals [Peharz et al., 2015]
- tractable posterior [Vergari et al., 2017]

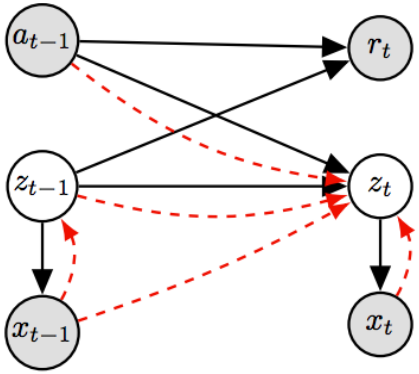


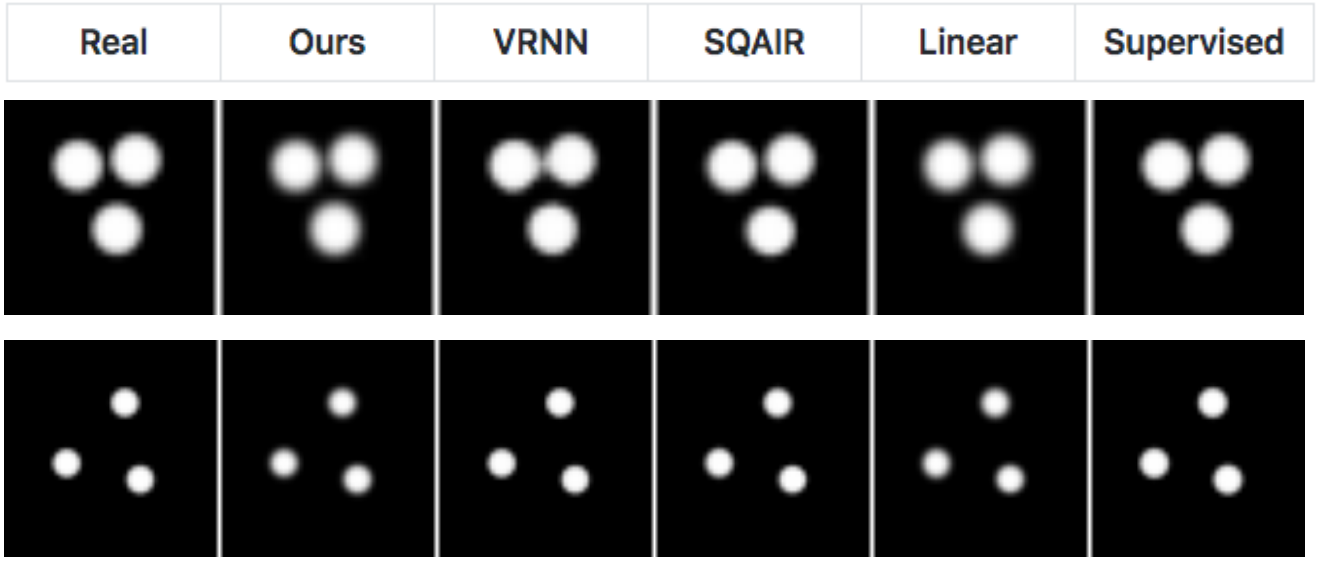
Unsupervised physics learning

[Kossen, Stelzner, Hussing, Voelcker, Kersting ICLR 2020]

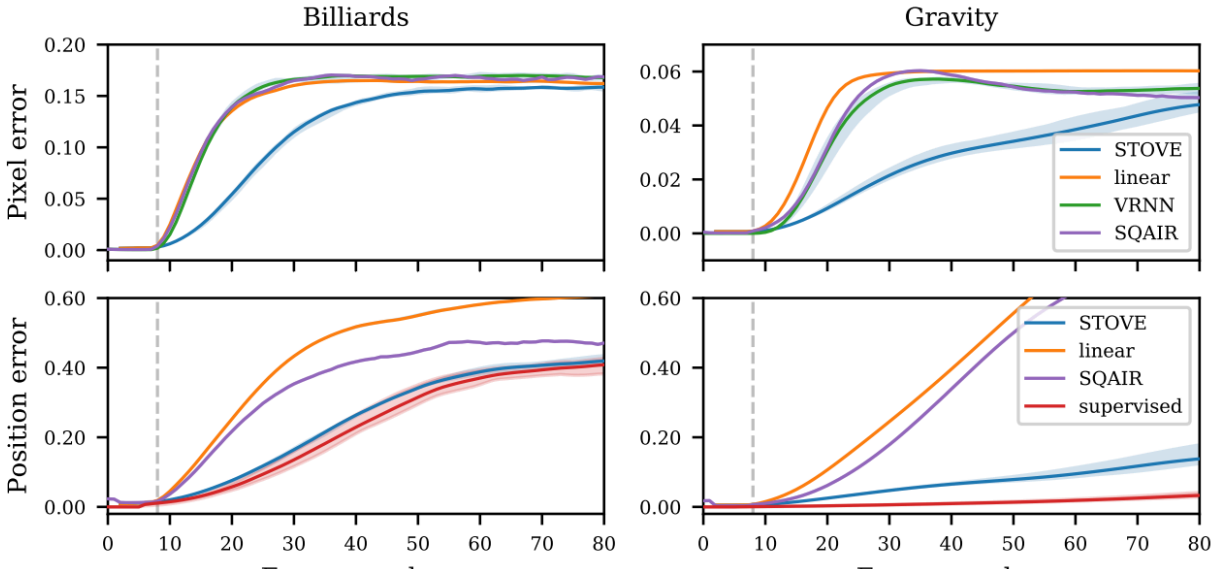
ICLR | 2020
Eighth International Conference on Learning Representations

TECHNISCHE UNIVERSITÄT DARMSTADT

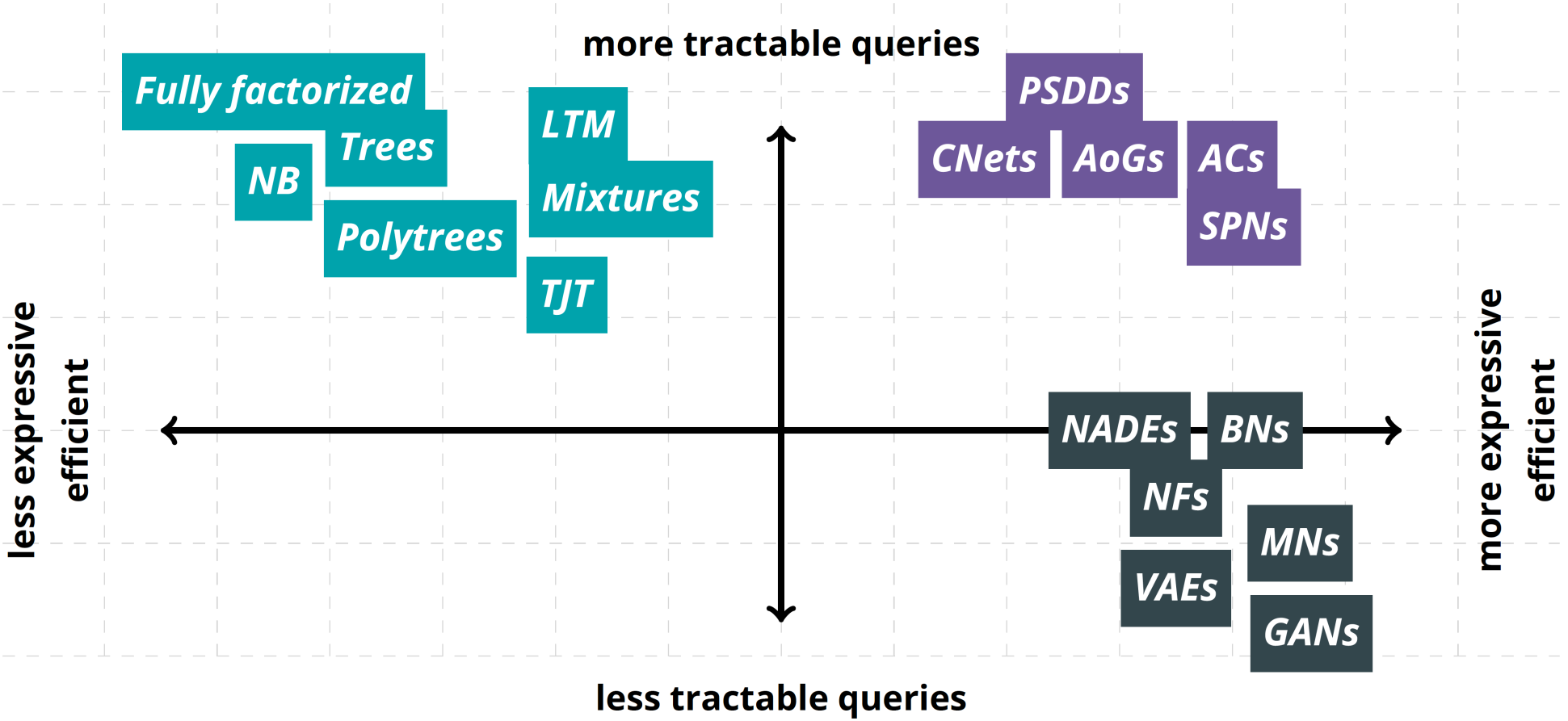




putting structure and tractable inference into deep models



Sum-Product Networks are an example of tractable probabilistic models. There are many others, and on typically speaks of probabilistic cirxuits



What have we learnt about SPNs?

TECHNISCHE UNIVERSITÄT DARMSTADT

Sum-product networks (SPNs)

- DAG of sums and products
- They are instances of Arithmetic Circuits (ACs)
- Compactly represent partition function
- Learn many layers of hidden variables

Efficient marginal inference

Easy learning

Can outperform well-known alternatives