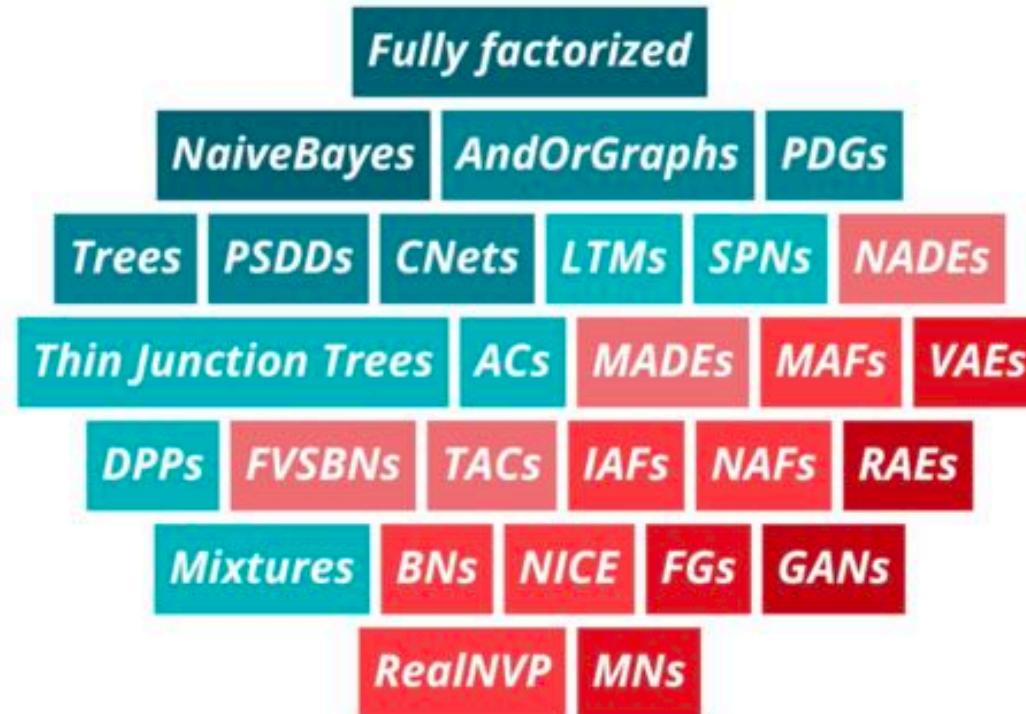




Tractable Probabilistic Models



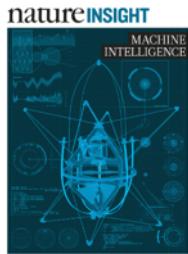
Thanks to Pedro Domingos for making
slides publically available
This is also based on joint work with
Alejandro Molina, Antonio Vergari,
Robert Peharz, Guy van den Broeck,
Karl Stelzner and many others. Thanks!

tractability is a spectrum





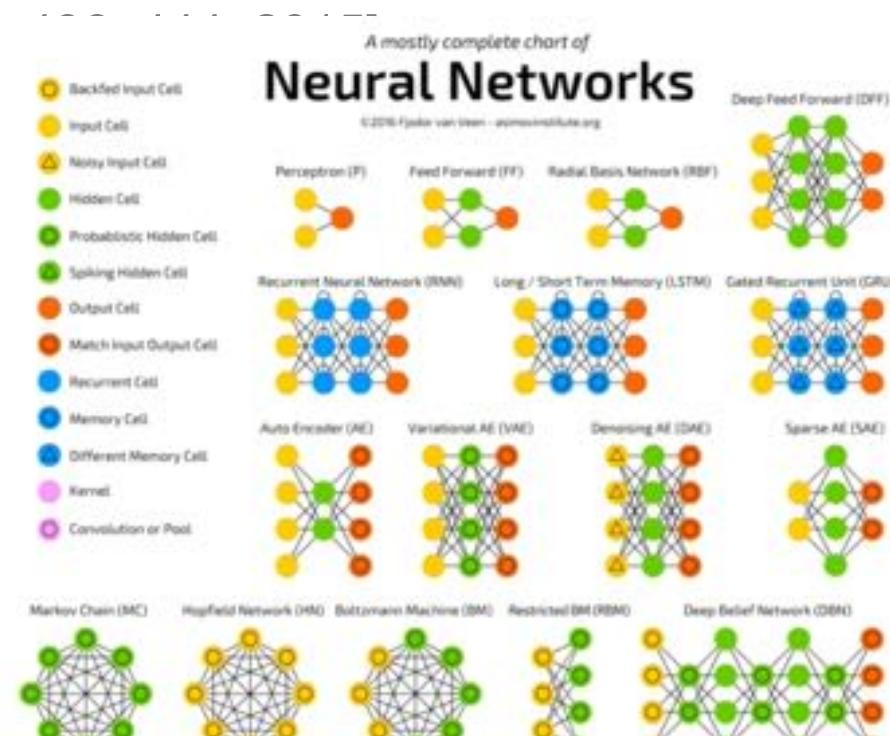
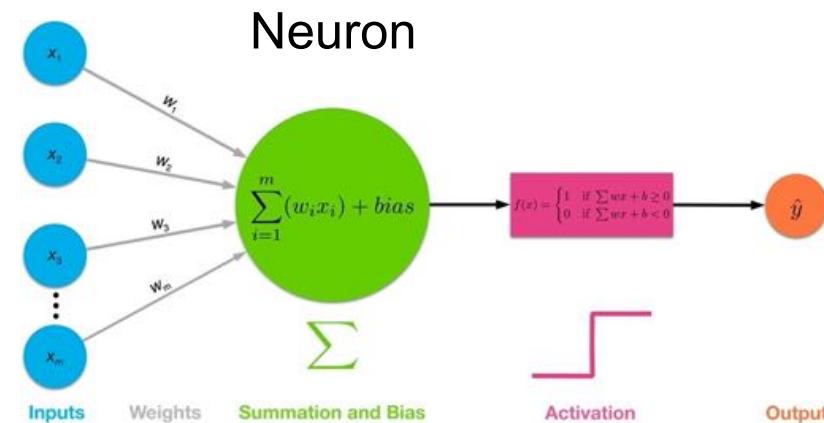
The Current Second Wave of AI



Deep Neural Networks

Potentially much more powerful than shallow architectures, represent computations

[LeCun, Bengio, Hinton Nature 521



Differentiable Programming

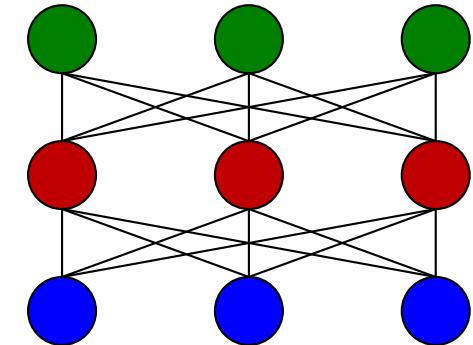
Deep Learning

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



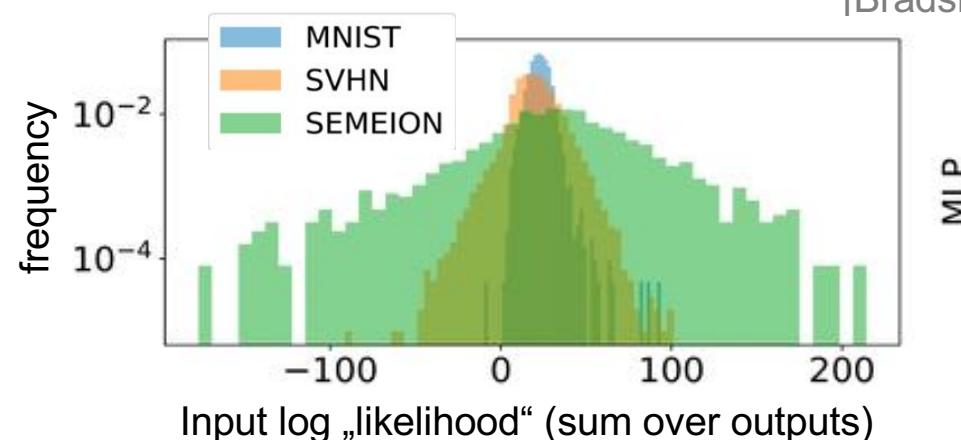
SVHN



SEMEION



Train & Evaluate



Transfer Testing

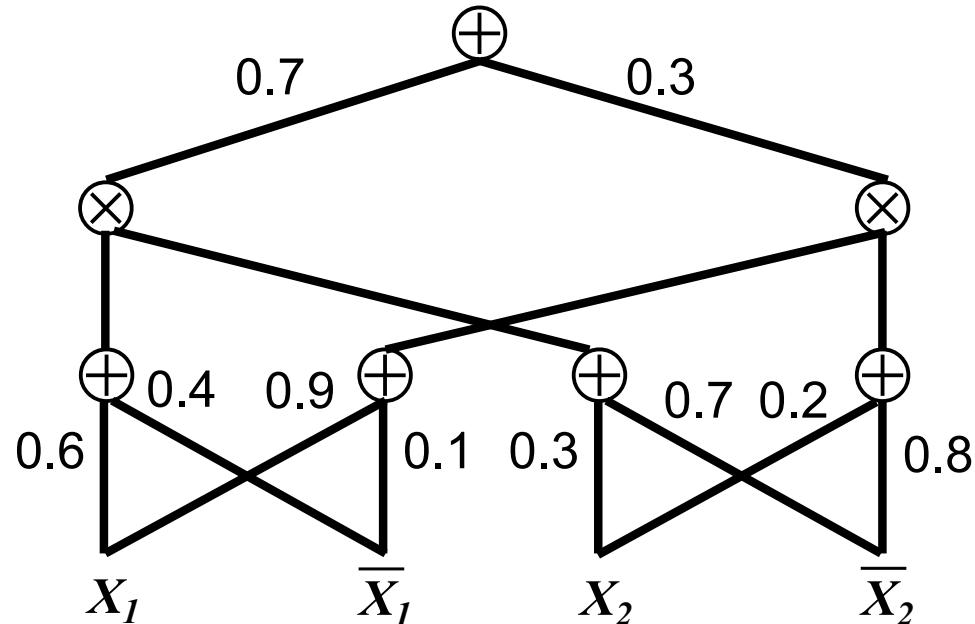
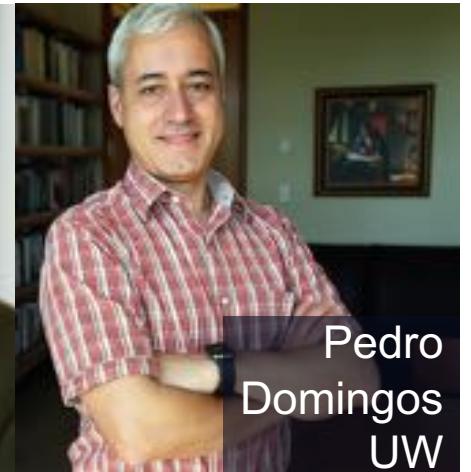
[Bradshaw et al. arXiv:1707.02476 2017]

Many DNNs cannot distinguish the datasets

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

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

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

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



Alternative Representation: Joint Distributions as (Deep) Networks

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

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



Shorthand for Indicators

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

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

Sum Out Variables

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

$$e: X_1 = 1$$

$$\begin{aligned} P(e) &= \mathbf{0.4} \cdot X_1 \cdot X_2 \\ &\quad + \mathbf{0.2} \cdot X_1 \cdot \bar{X}_2 \\ &\quad + 0.1 \cdot \bar{X}_1 \cdot X_2 \\ &\quad + 0.3 \cdot \bar{X}_1 \cdot \bar{X}_2 \end{aligned}$$

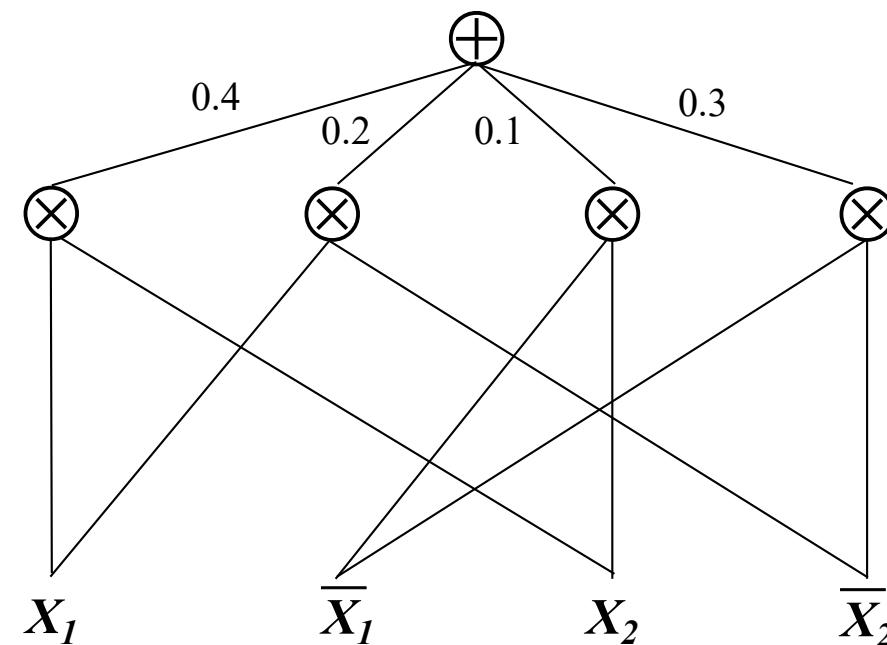
Set $X_1 = 1, \bar{X}_1 = 0, X_2 = 1, \bar{X}_2 = 1$

Easy: Set both indicators of X_2 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

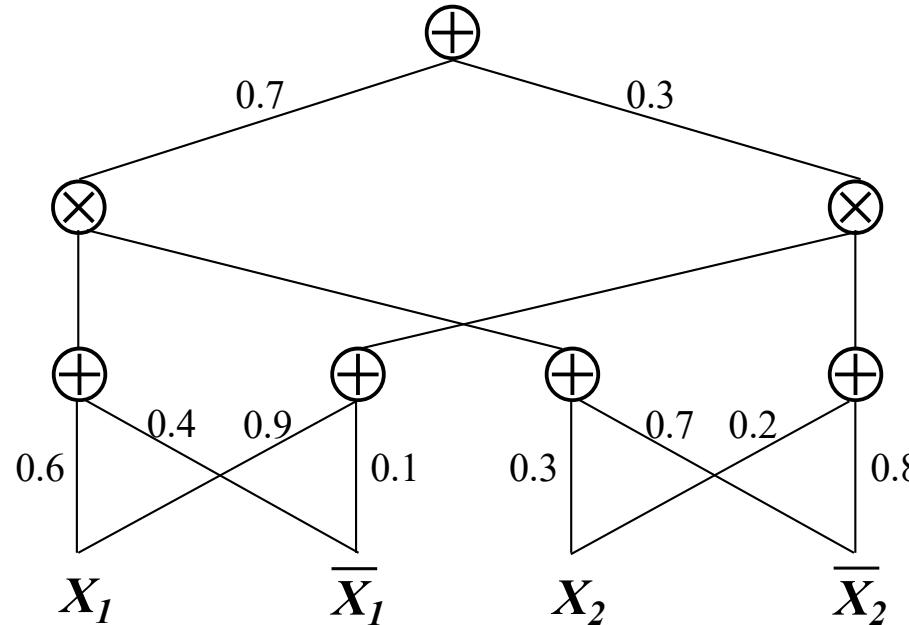


Sum-Product Networks* (SPNs)



[Poon, Domingos UAI 2011]

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

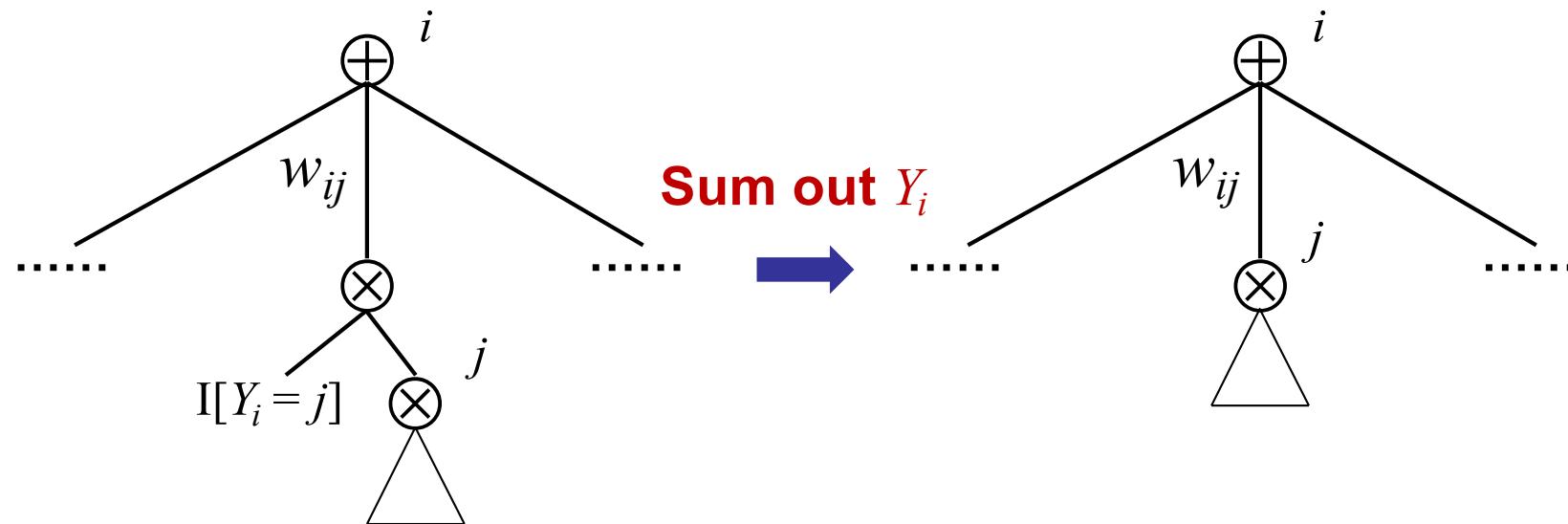


*SPNs are an instance of Arithmetic Circuits (ACs). ACs have been introduced into the AI literature more than 15 years ago as a tractable representation of probability distributions
[Darwiche CACM 48(4):608-647 2001]

Semantics of Sums and Products

Product ~ **Feature** → Form feature hierarchy

Sum ~ **Mixture** (with hidden var. summed out)



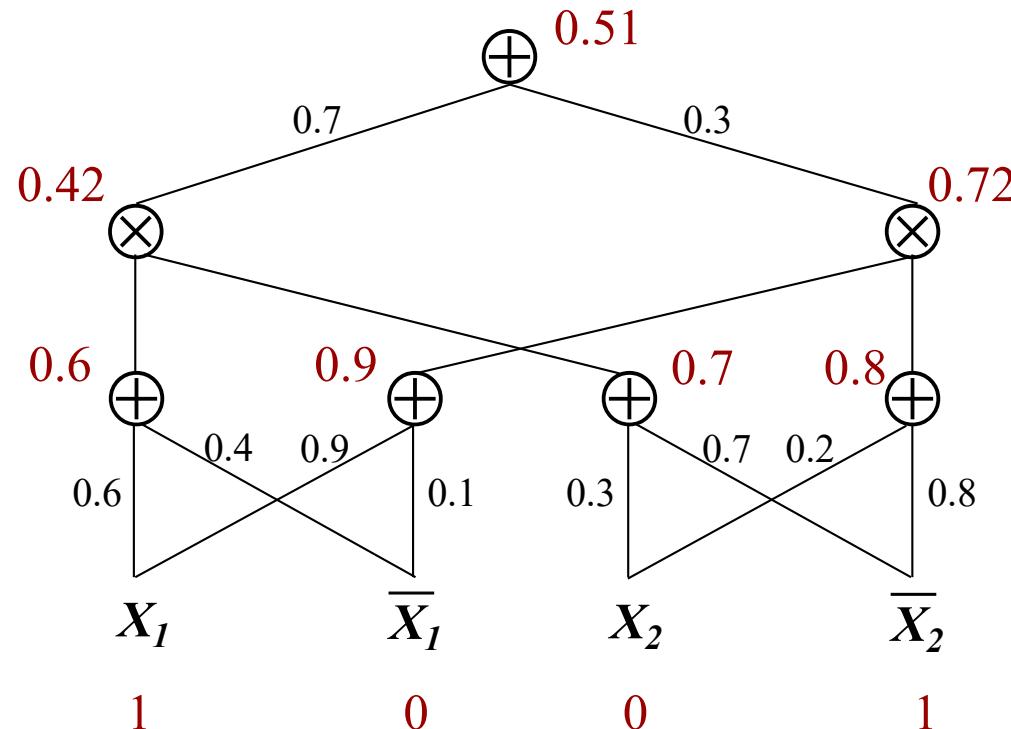
Inference: Linear in Size of Network

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
\bar{X}_1	0
X_2	0
\bar{X}_2	1

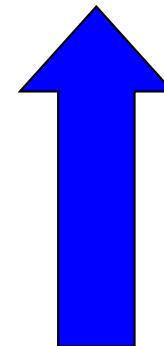
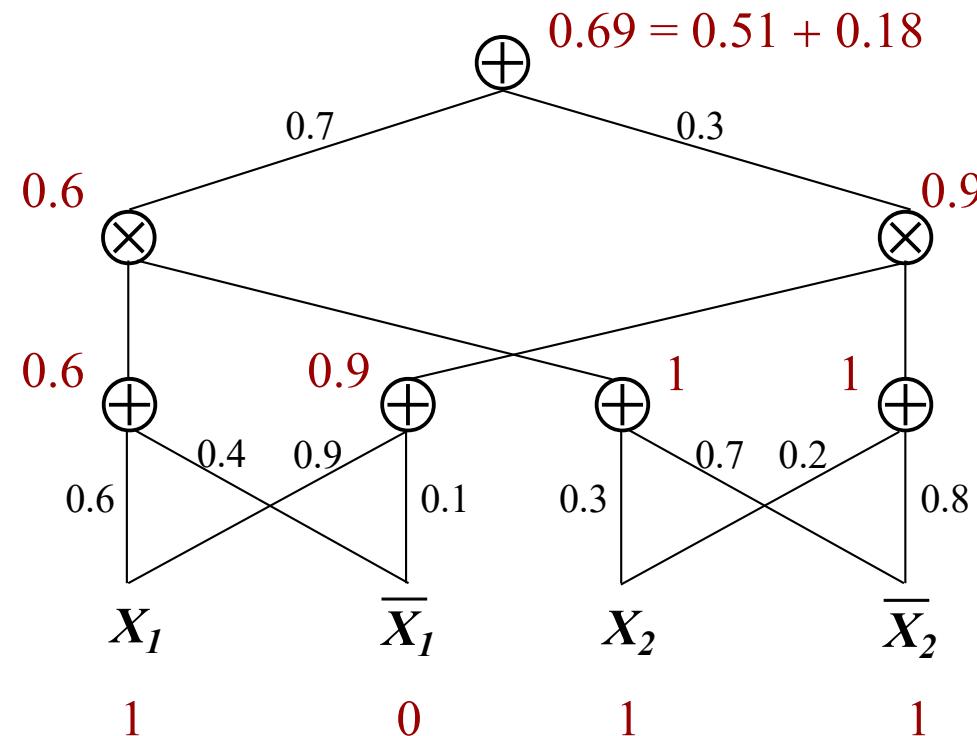


Inference: Linear in Size of Network

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

$$e: X_1 = 1$$

X_1	1
\bar{X}_1	0
X_2	1
\bar{X}_2	1



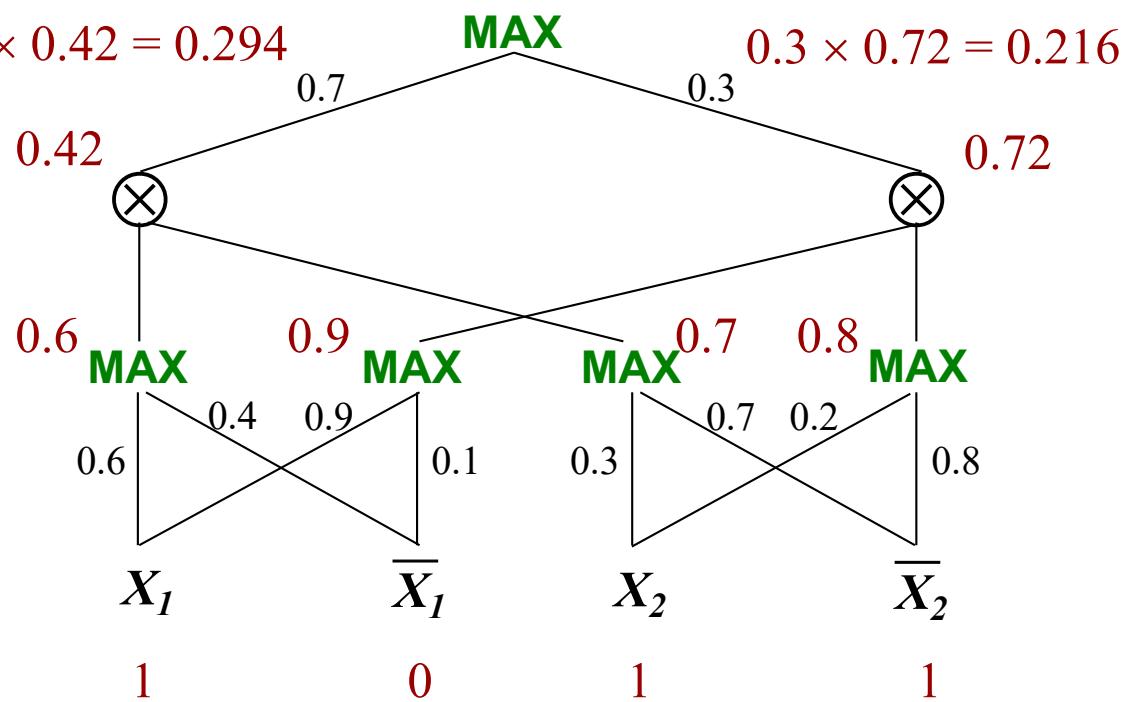
Inference: Linear in Size of Network

MAP: Replace sums with maxs

$$e: X_1 = 1$$

$$0.7 \times 0.42 = 0.294$$

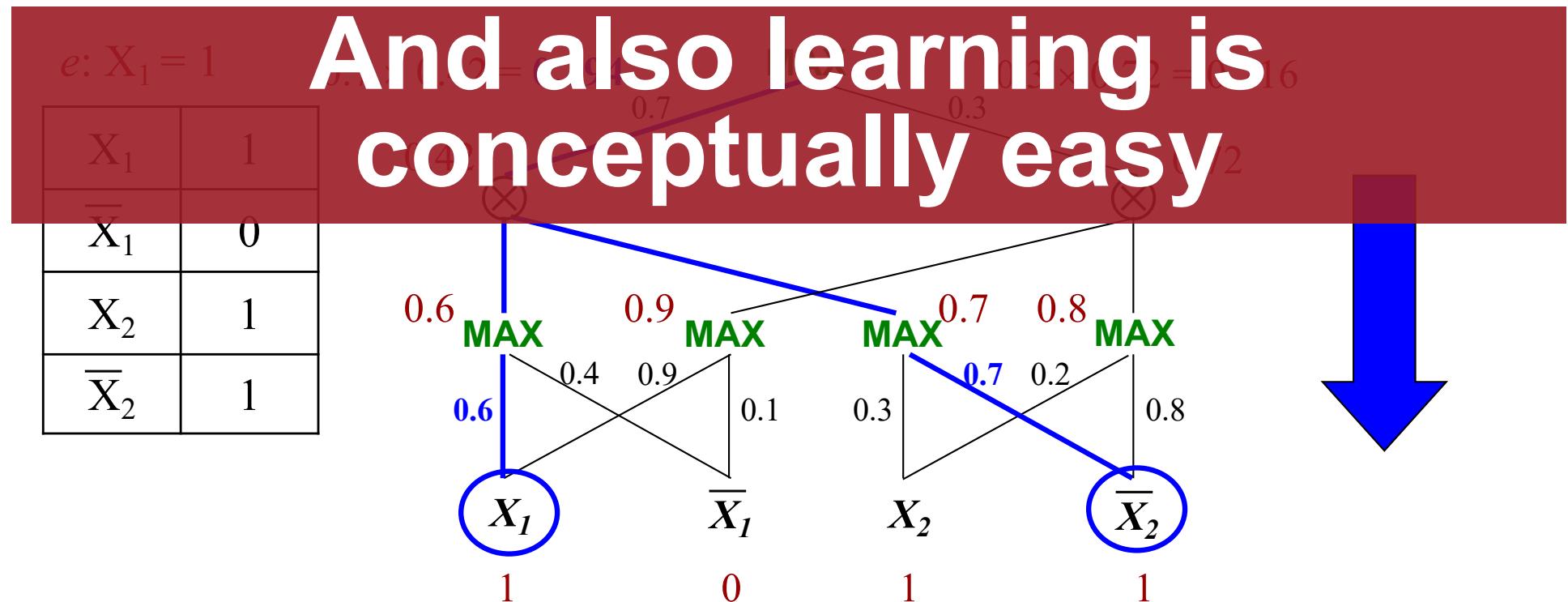
X_1	1
\bar{X}_1	0
X_2	1
\bar{X}_2	1



Inference: Linear in Size of Network

MAX: Pick child with highest value

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



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

- 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

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

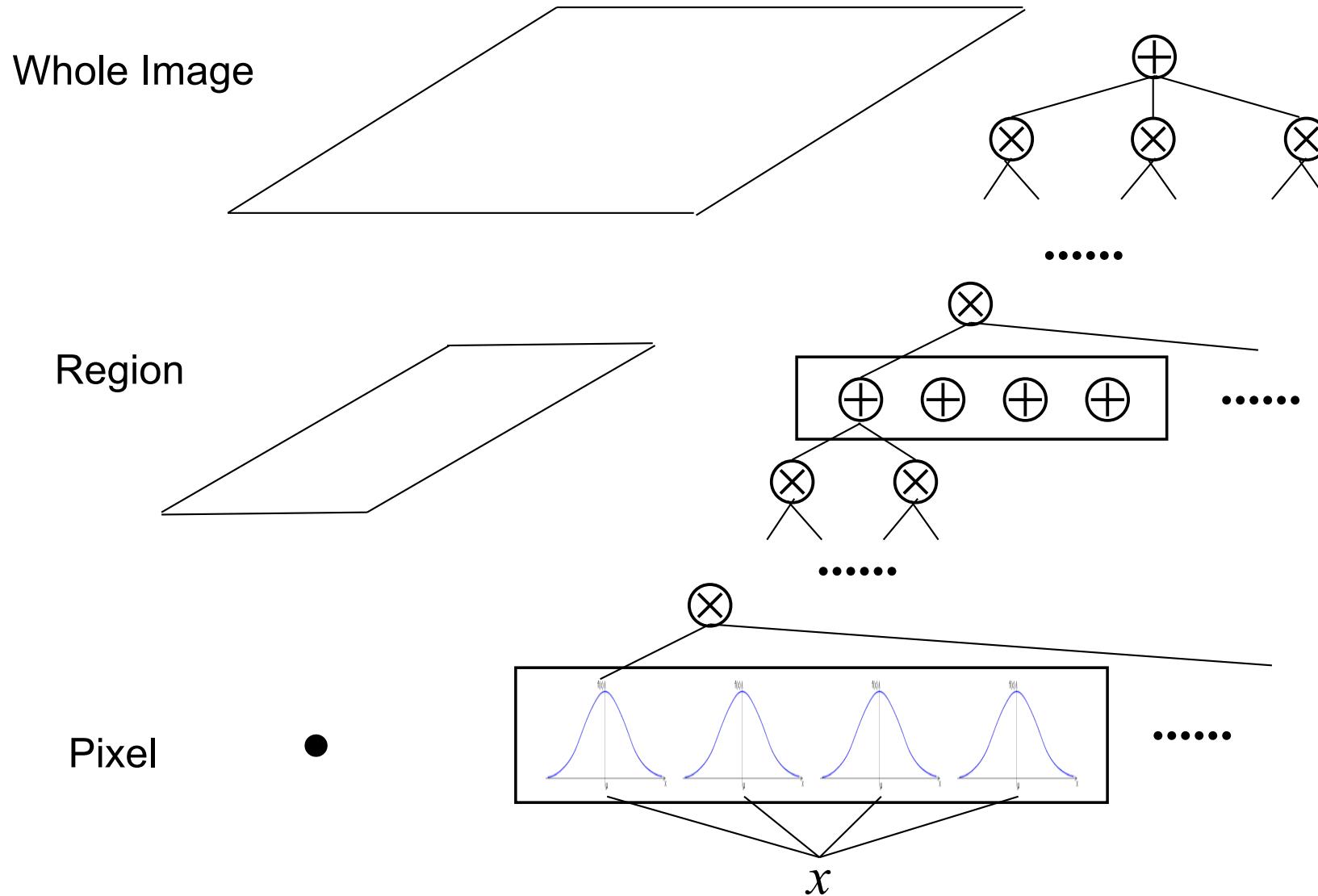
- 101 categories, e.g., faces, cars, elephants
- Each category: 30 – 800 images

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

Each category: Last third for test

Test images: Unseen objects

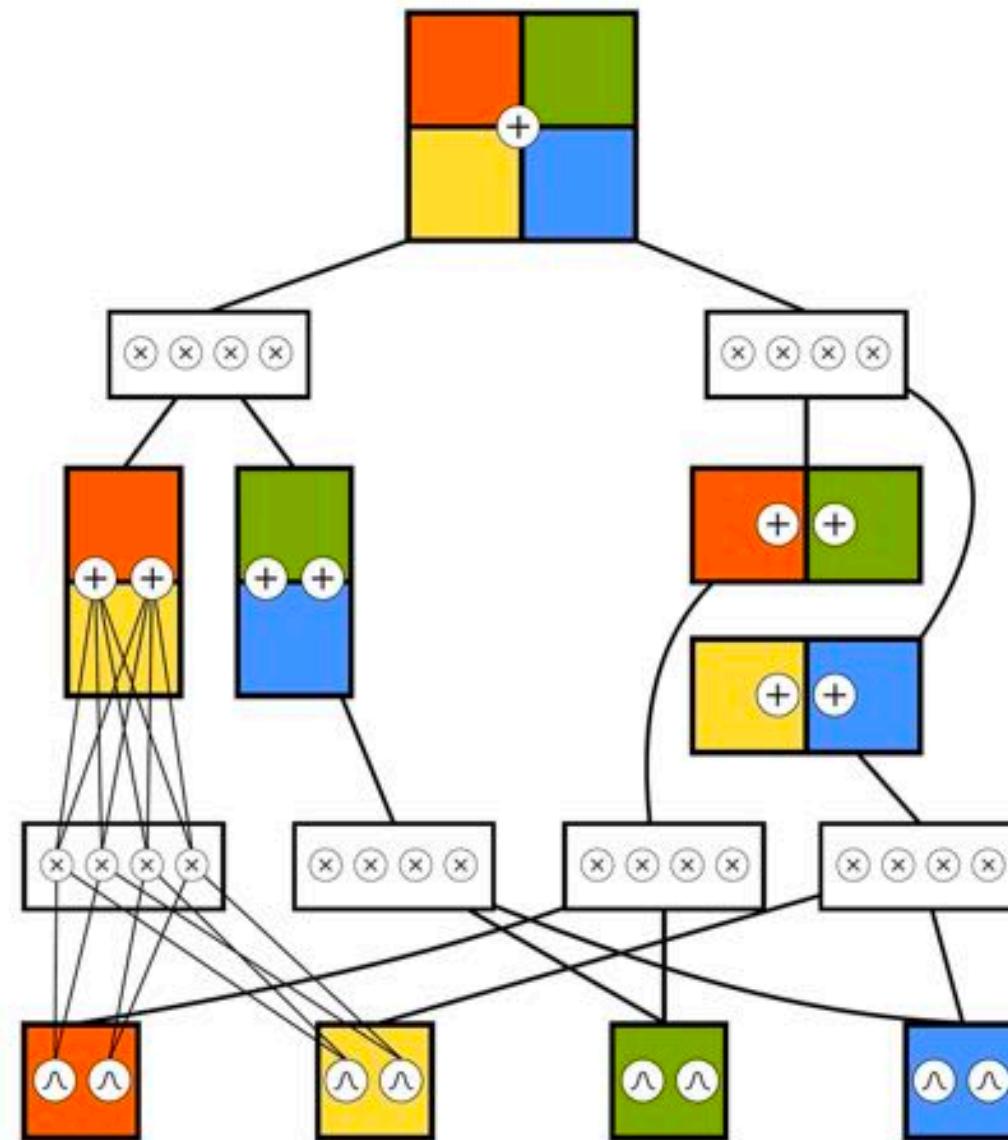
SPN Architecture



SPN Architecture

Whole Image

Regions



Caltech: Mean-Square Errors

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



SPN vs. DBM / DBN

SPN is order of magnitude faster

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

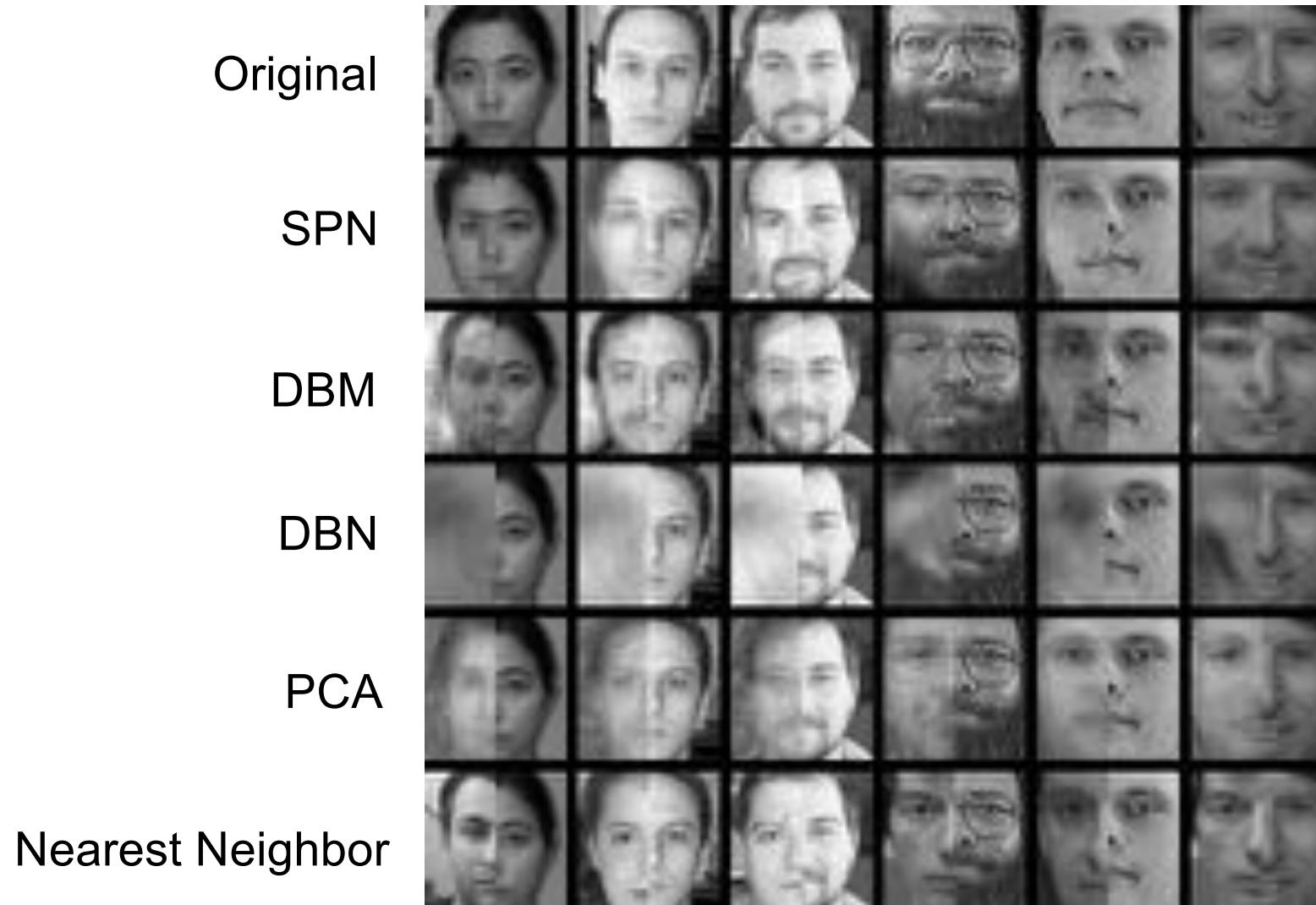
No elaborate preprocessing, tuning

Reduced errors by 30-60%

Learned up to 46 layers

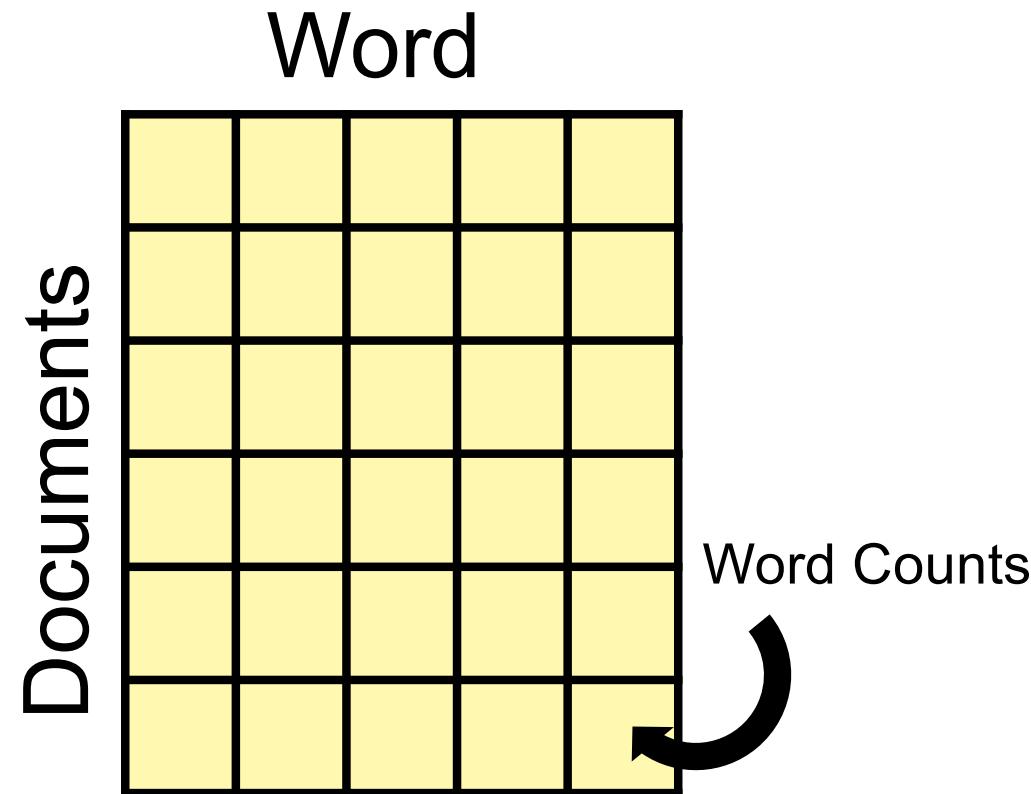


Example Completions



Or we learn directly (Tree-)SPNs

Testing independence of
Poisson random variables

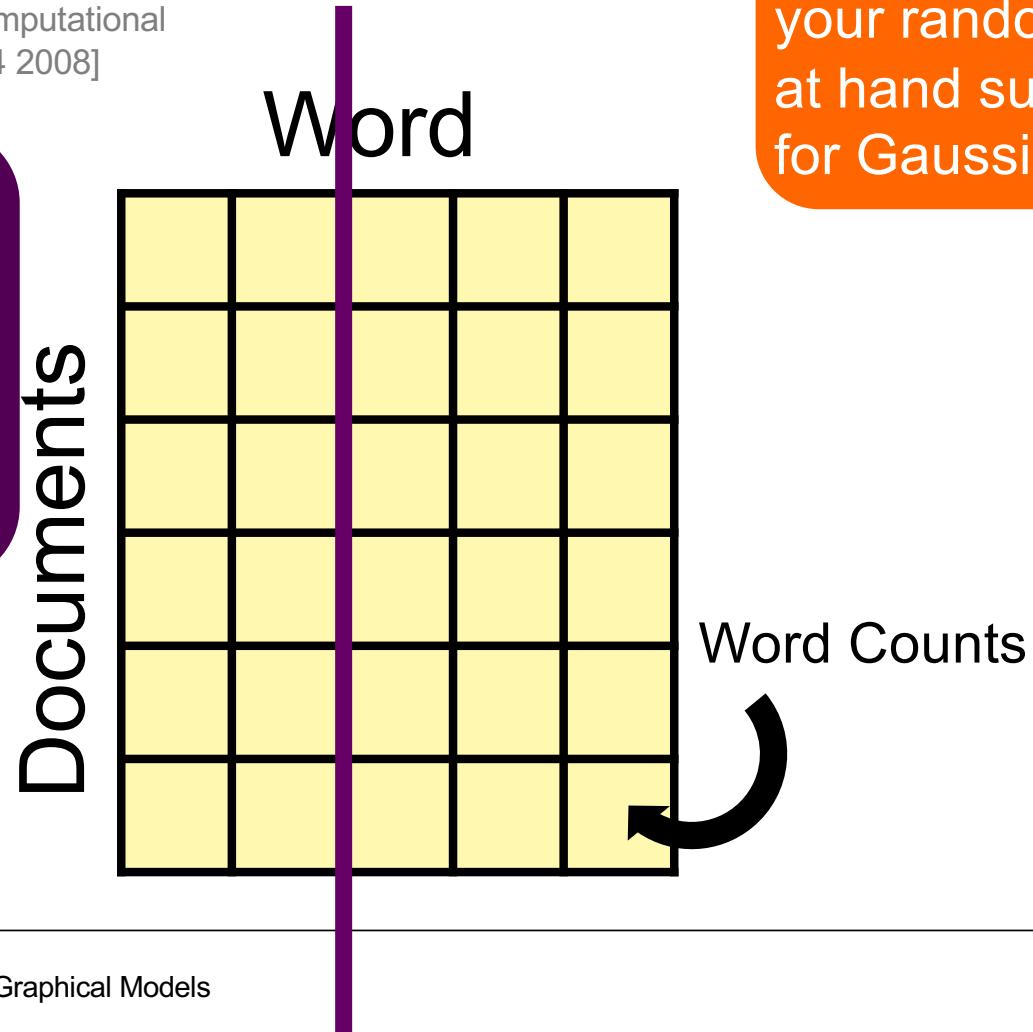


Or we learn directly (Tree-)SPNs

Testing independence of Poisson random variables

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

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. $P(x|V-y)$



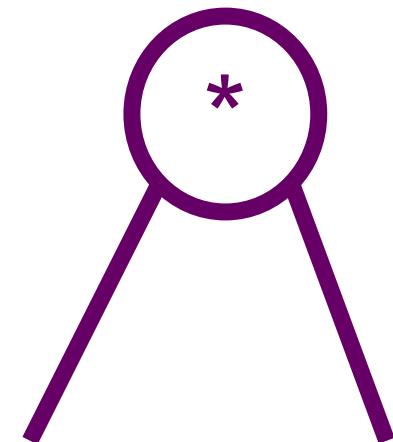
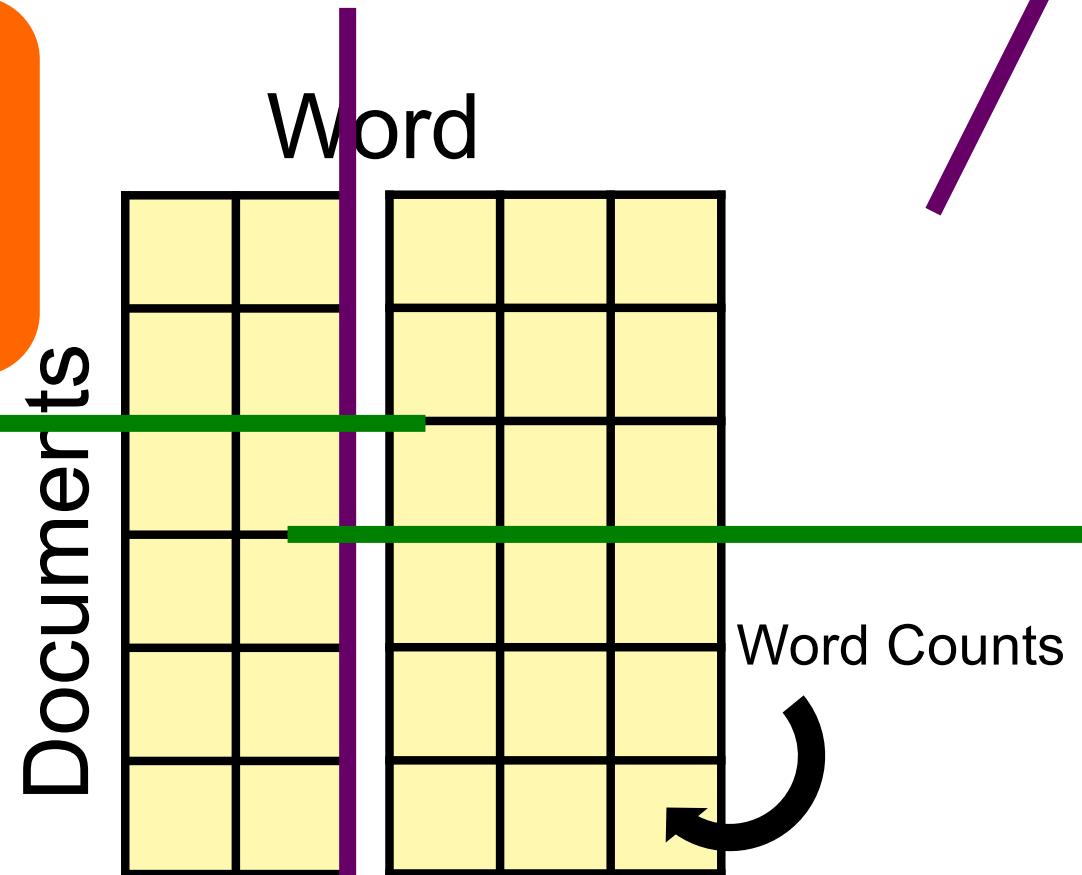
In general use the independency test for your random variables at hand such as g-test for Gaussians

Or we learn directly (Tree-)SPNs

Testing independence of
Poisson random variables

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

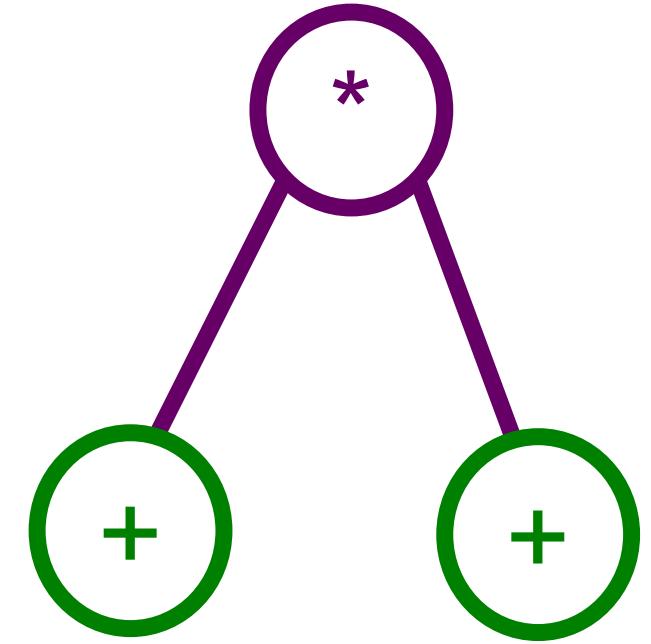
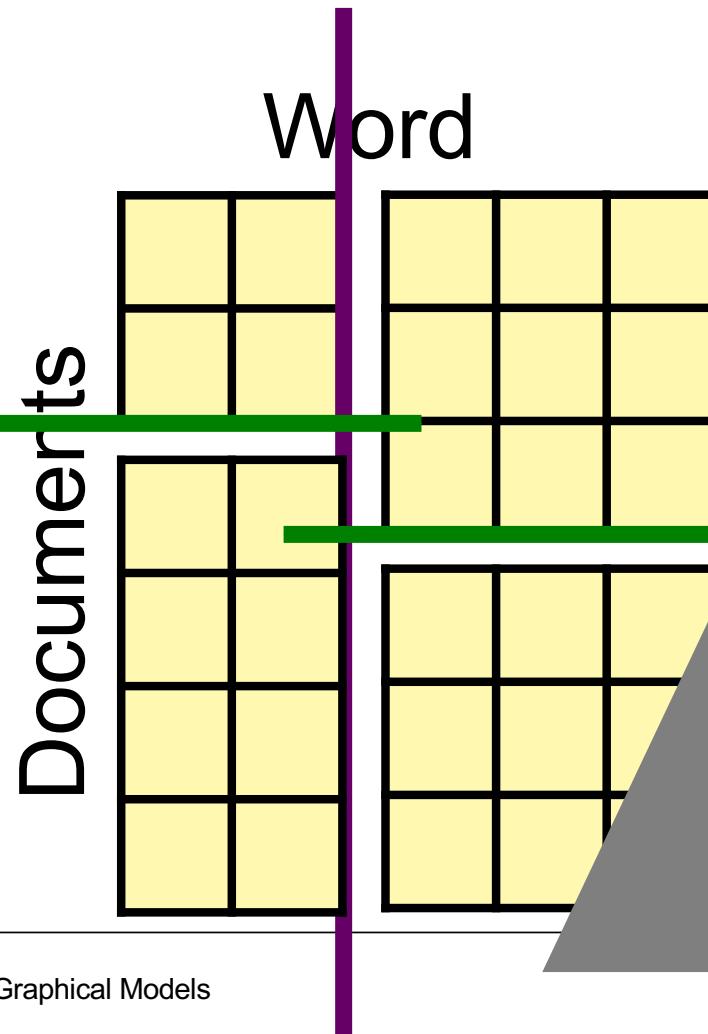
Mixture of Poisson Dependency Networks or random splits



Or we learn directly (Tree-)SPNs

Testing independence of
Poisson random variables

Mixture of
Poisson
Dependency
Networks or
random splits



keep growing
alternatingly
and + layers

[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]

FL⁺ SPFlow: An Easy and Extensible Library ⊗W for Sum-Product Networks



UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO



Max Planck Institute for
Intelligent Systems



UNIVERSITY OF
CAMBRIDGE



VECTOR
INSTITUTE

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



Federal Ministry
of Education
and Research

195 commits

2 branches

0 releases

13.6 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 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])

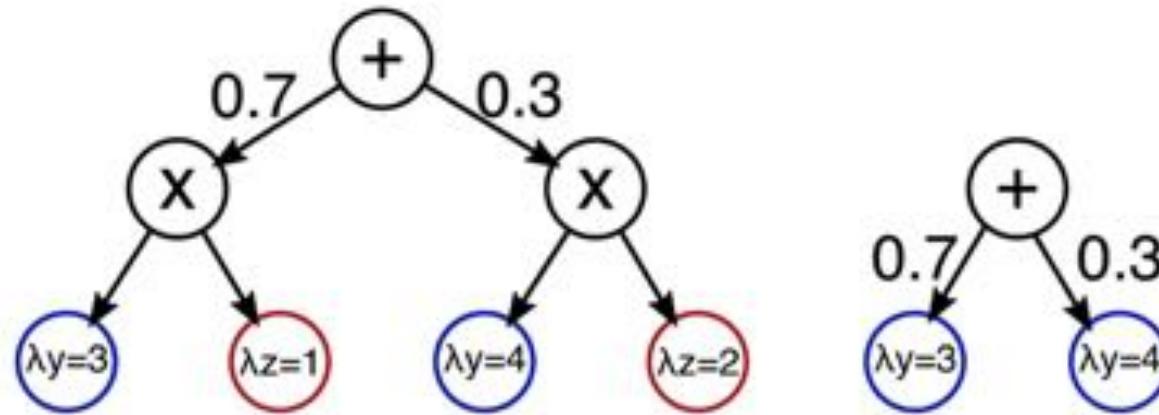
assign_ids(spn)
rebuild_scopes_bottom_up(spn)

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

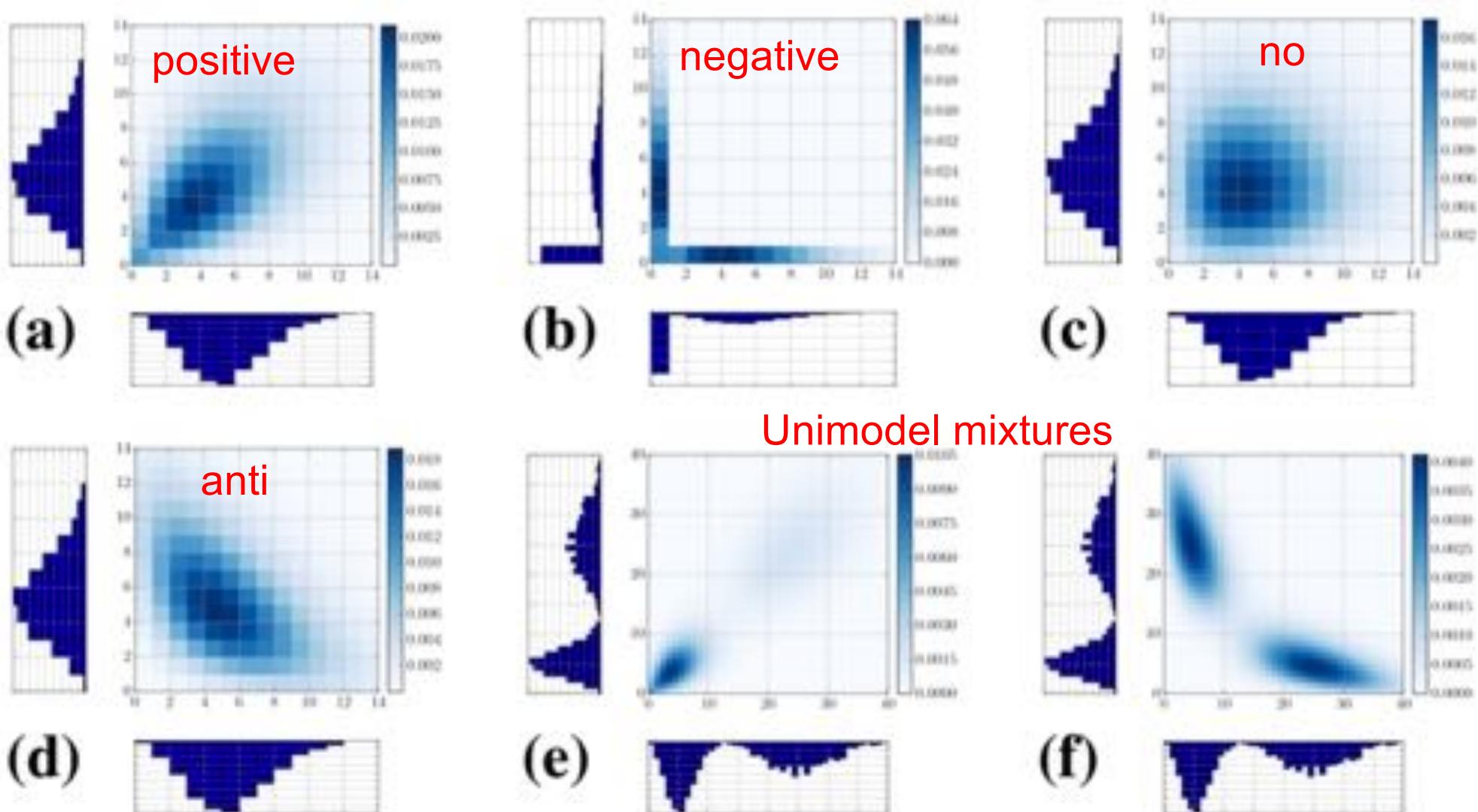
Inference on Devices



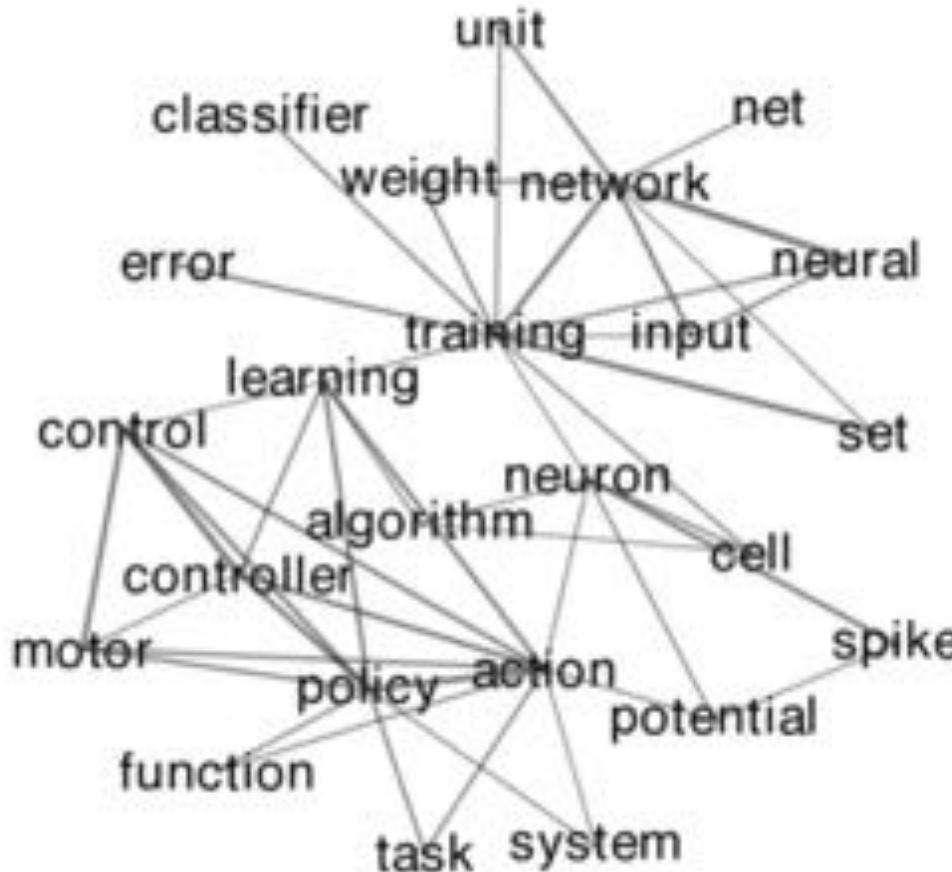
Ultimately, they may even be compiled into simple, flat, library-free code suitable for embedding in real-time applications and devices

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

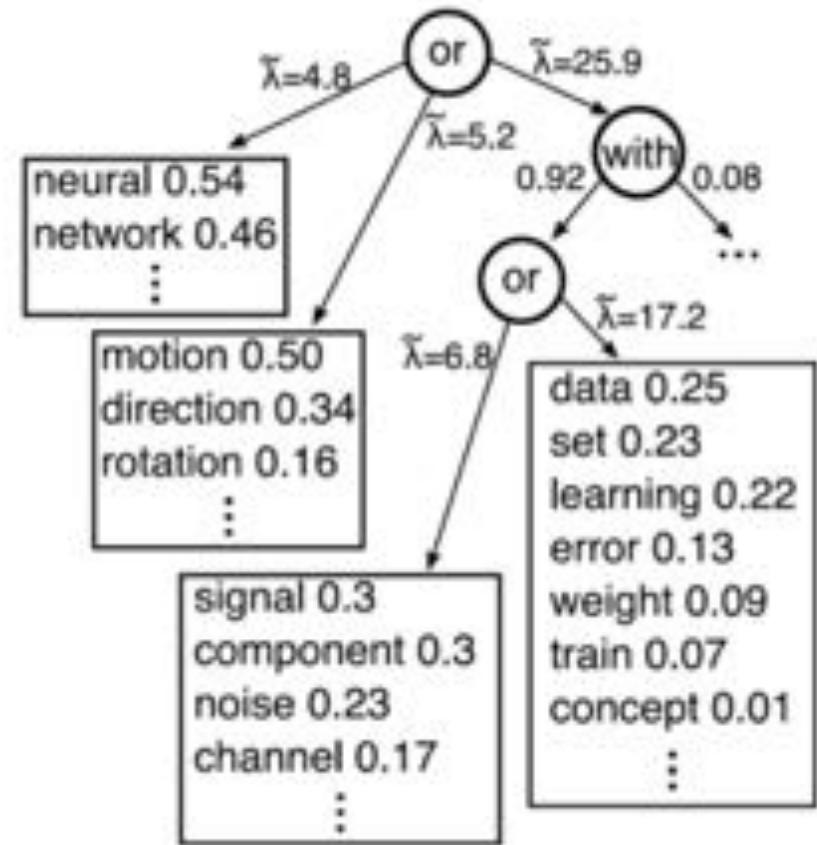
Flexible Dependencies even for challenging distributions, here multivariate Poisson distribution



Multiple views on your 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

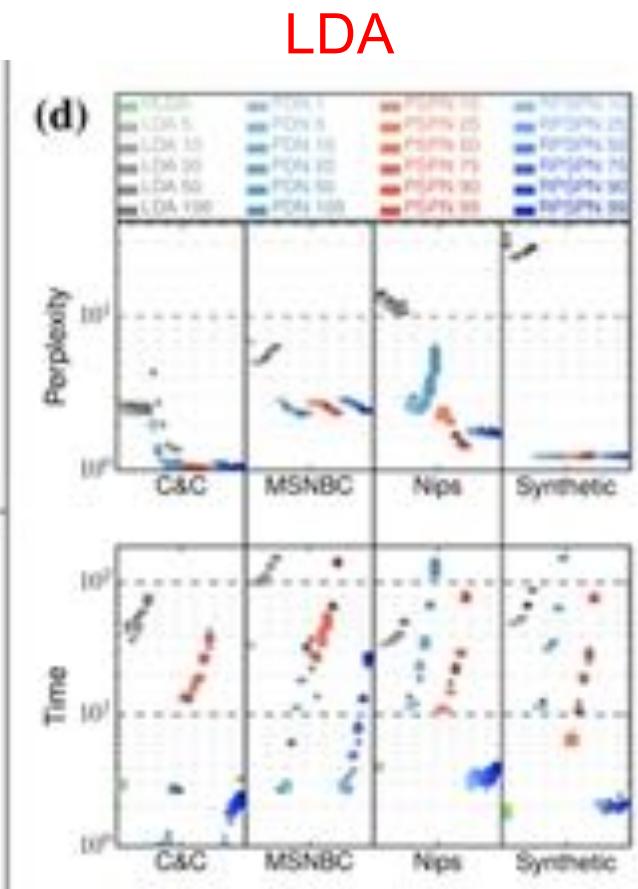
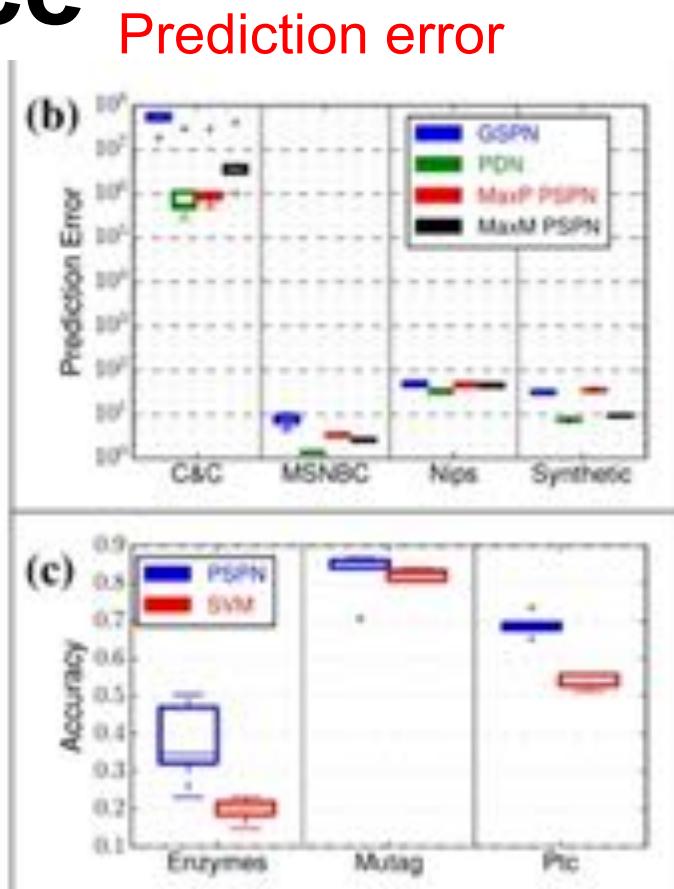
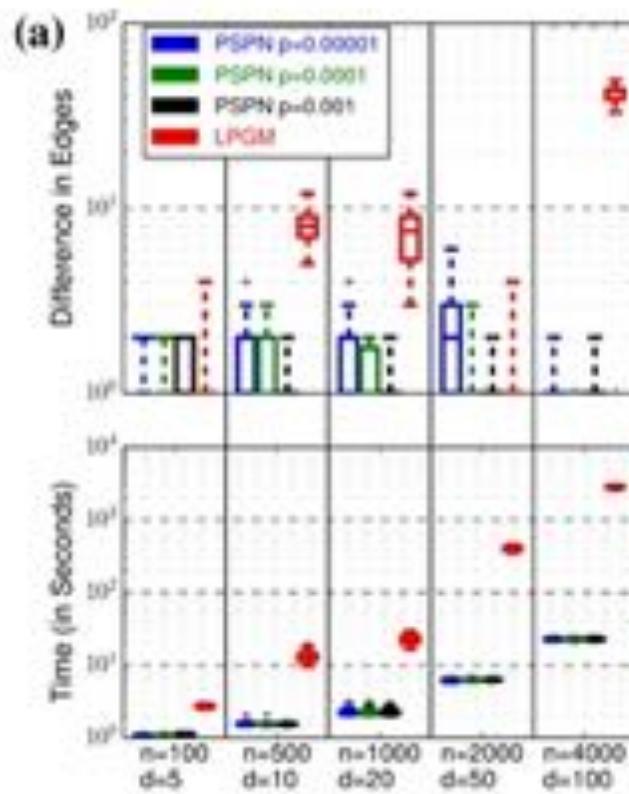
[Bishop, Fienberg, Holland. Discrete Multivariate Analysis: Theory and Practice. Springer, 2007]

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

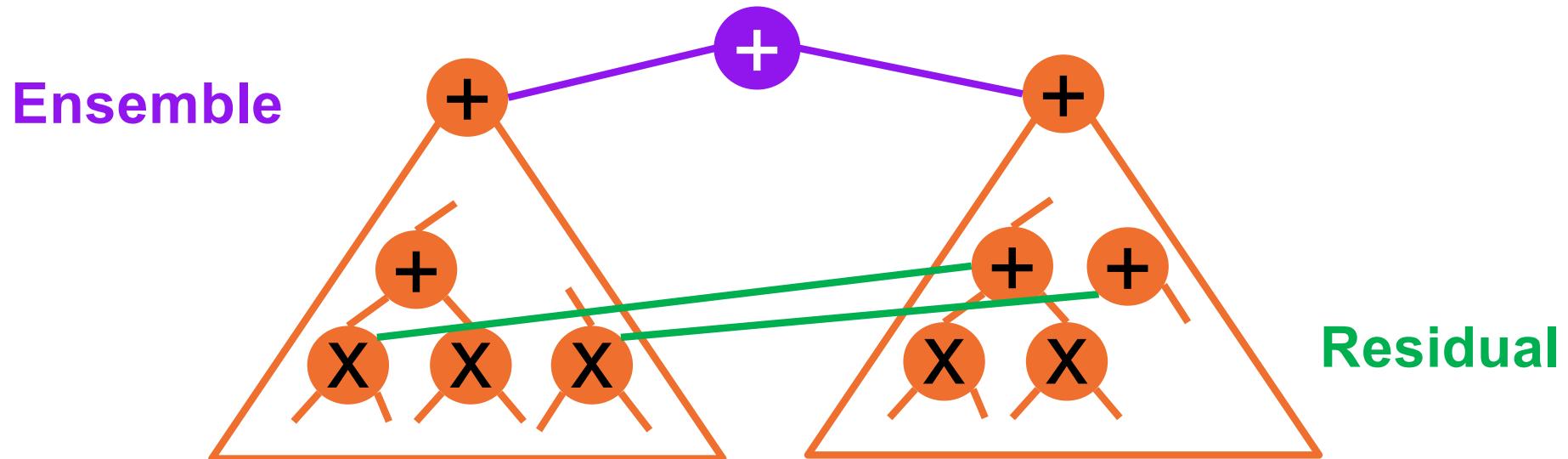
$$\begin{aligned}
 & \Pr_{\text{Poiss}} \left(\tilde{x} \mid \tilde{\lambda} \right) \Pr_{\text{Mult}} \left(\mathbf{x} \mid \theta = (\lambda_1, \dots, \lambda_p) / \tilde{\lambda}, \mathbf{N} = \tilde{x} \right) \\
 &= \frac{e^{-\tilde{\lambda}}}{\tilde{x}!} \tilde{\lambda}^{\tilde{x}} \frac{\tilde{x}!}{\prod_{s=1}^p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



Competitive Predictive Performance



Residual Sum-Product Networks



ResSPNs = Ensemble SPNs + Residual Links

	Test Accuracy				
	Baseline Singleton SPN Learners			Ensemble SPN	
	Best ExtraSPN	LearnSPN	ID-SPN	ResSPN	
NLTCS	-6.153	-6.11	-6.02	-6.040↑	
MSNBC	-6.433	-6.11	-6.04	-6.097↑	
Plants	-16.683	-12.98	-12.54	-13.908●	
Audio	-42.639	-40.5	-39.79	-40.762●	
Jester	-55.335	-53.48	-52.86	-53.956●	
Netflix	-60.330	-57.33	-56.36	-57.867●	

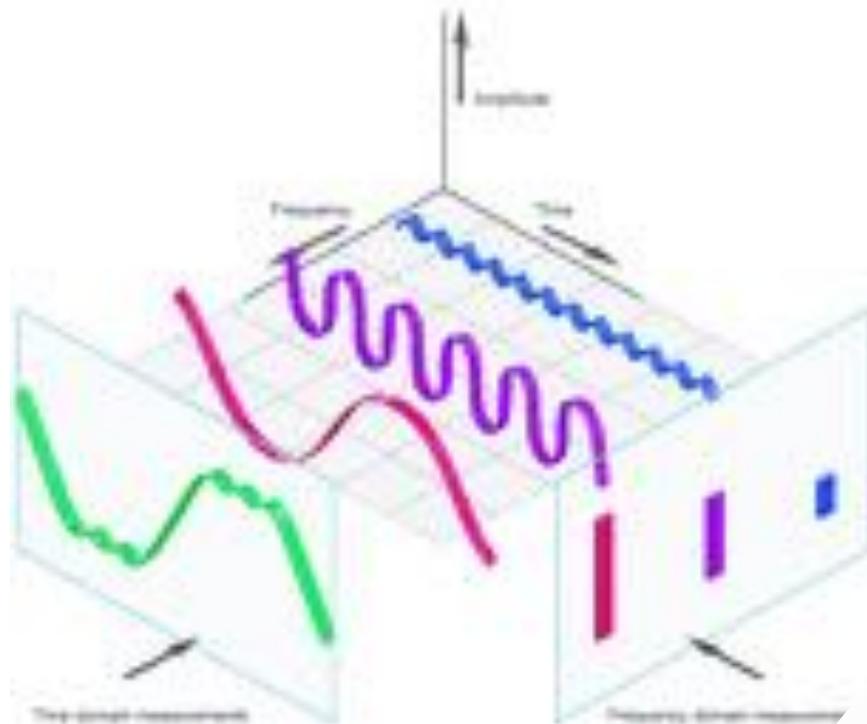
Whittle Networks



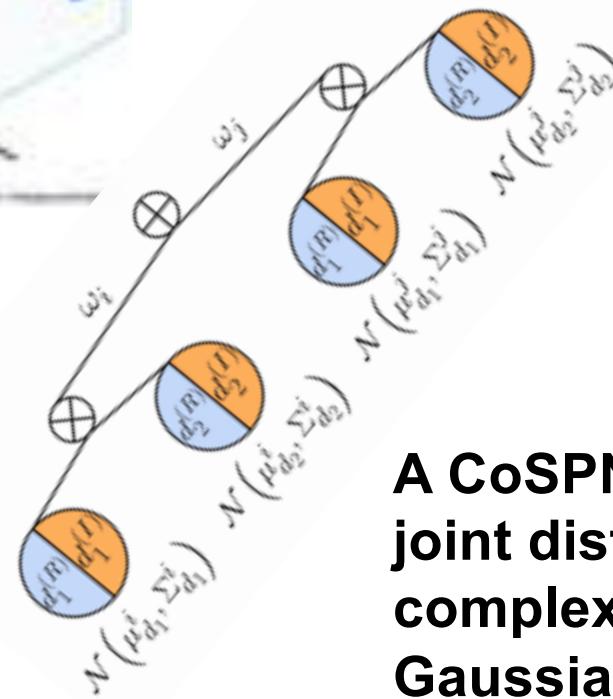
ICML 2021

38th International Conference
on Machine Learning

[Yu, Ventola, Kersting ICML 2021]



Encode the Whittle likelihood
for times series using a
complex-valued SPN (CoSPN)



Whittle Likelihood:
The Fourier coefficients of
stationary times series are
independent complex-valued
Gaussians with zero mean

A CoSPN models the
joint distribution over
complex (normal)
Gaussians

Whittle Networks

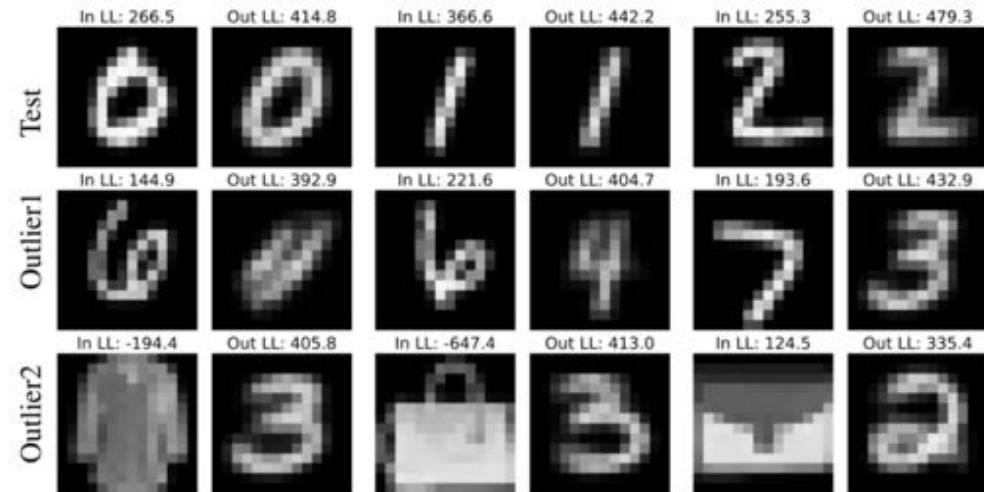
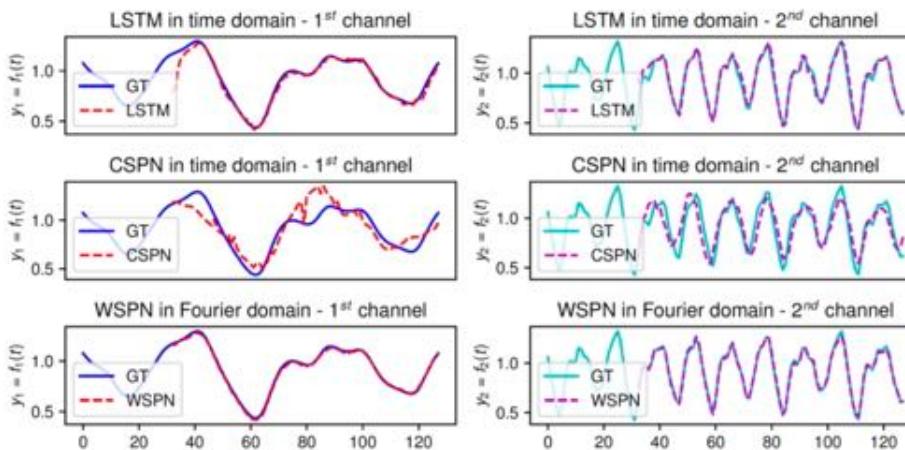
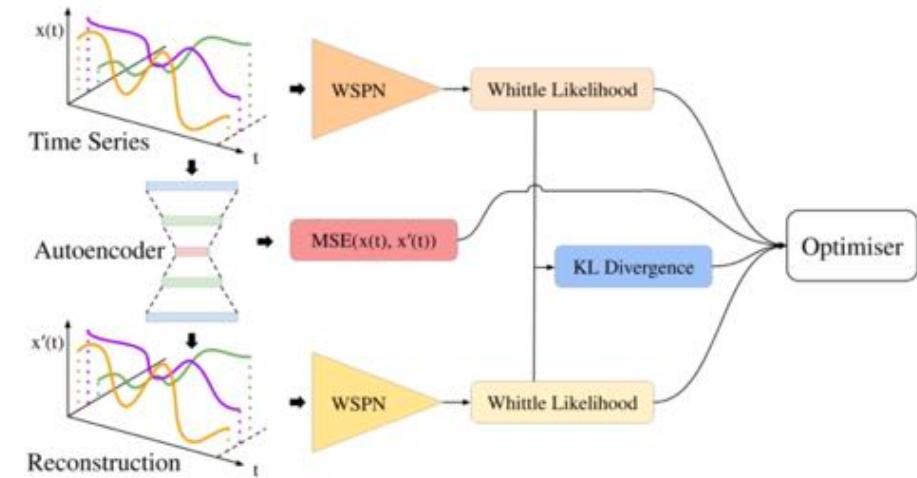
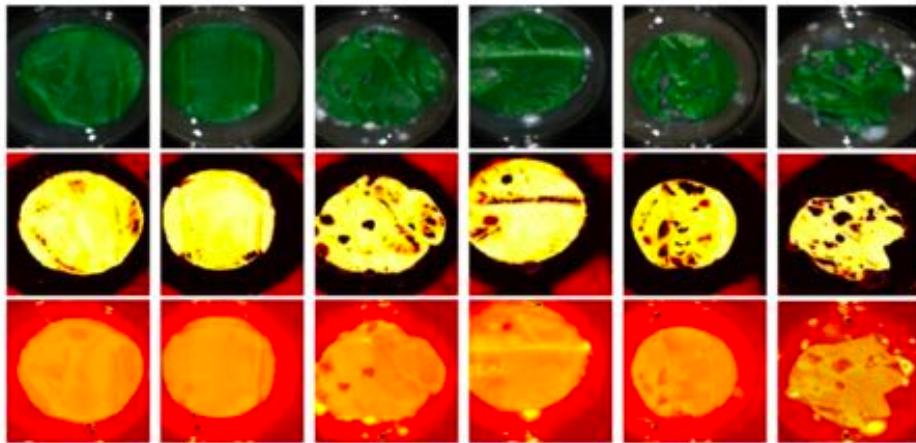


ICML 2021

38th International Conference
on Machine Learning

[Yu, Ventola, Kersting ICML 2021]

SAMMIE Whittle LL Test Images



Random sum-product networks



TECHNISCHE
UNIVERSITÄT
DARMSTADT



UNIVERSITY OF
CAMBRIDGE

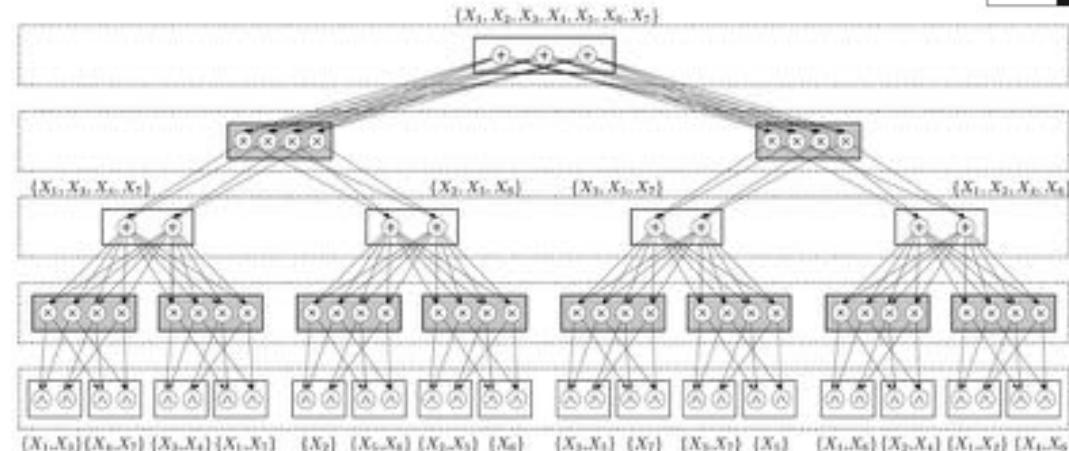


Max Planck Institute for
Intelligent Systems

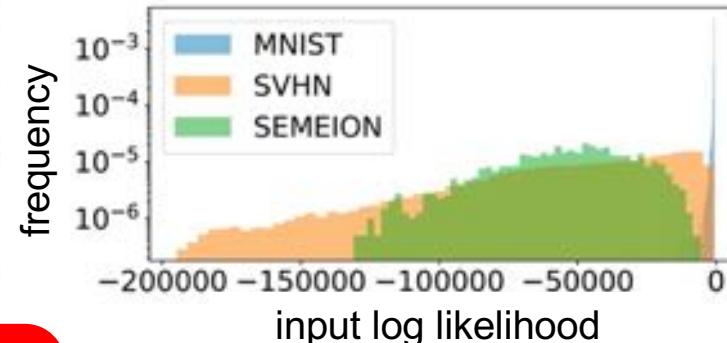


TECHNISCHE
UNIVERSITÄT
DARMSTADT

UBER AI Labs



	RAT-SPN	MLP	vMLP
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)
MNIST	0.0852 (17M)	0.0874 (0.82M)	0.0974 (0.22M)
F-MNIST	0.3525 (0.65M)	0.2965 (0.82M)	0.3225 (0.29M)
20-NG	1.6954 (1.63M)	1.6180 (0.22M)	1.6263 (0.22M)



SPNs can have similar predictive performances as (simple) DNNs

SPNs can distinguish the datasets

SPNs know when they do not know by design

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



Einsum Networks

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



TU/e



TECHNISCHE
UNIVERSITÄT
DARMSTADT

UBER AI Labs

UCLA



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.

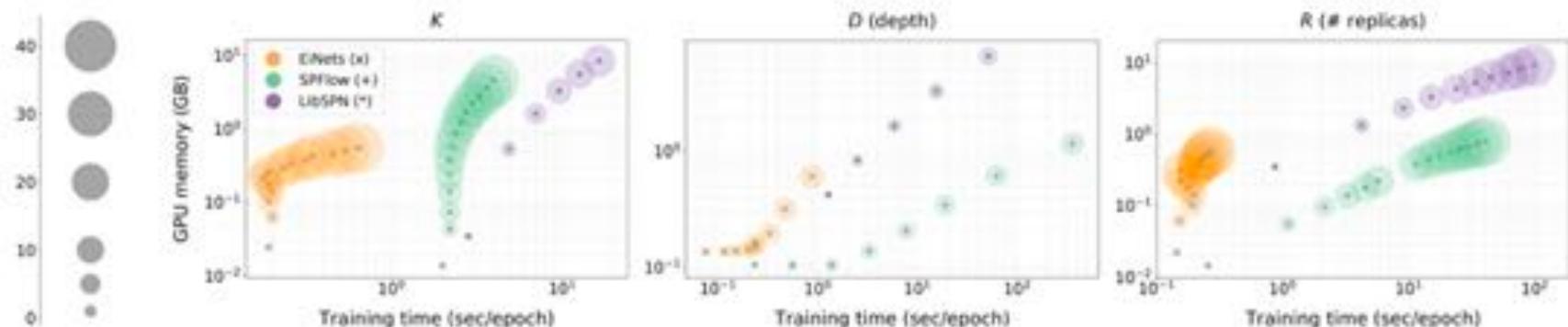
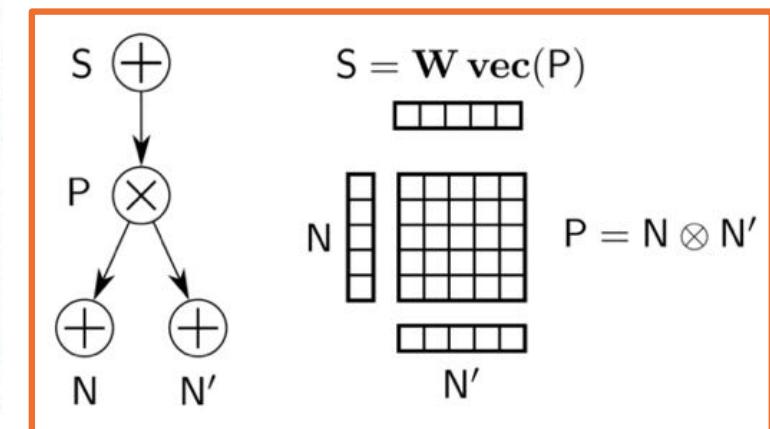


(e) EiNet Celeba samples.



(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>

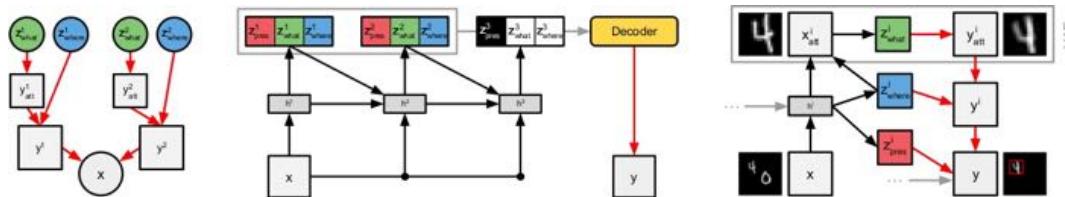


TECHNISCHE
UNIVERSITÄT
DARMSTADT

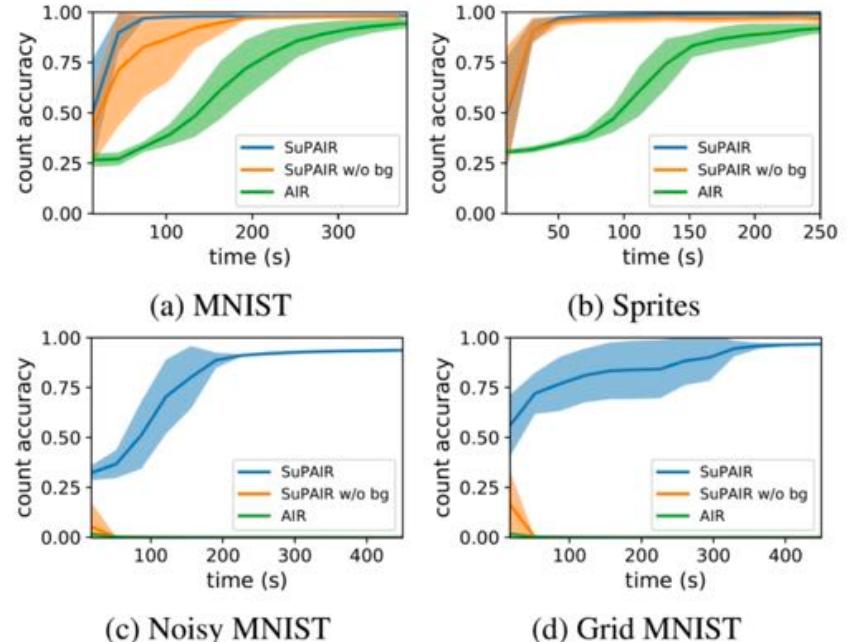
ICML | 2019

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]



VAE

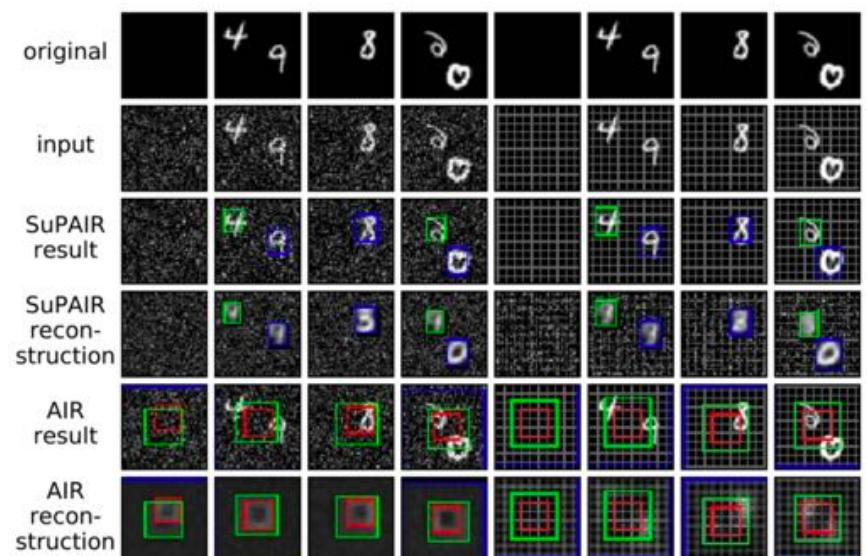


Replace VAE
by SPN as
object model

SPN



- 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

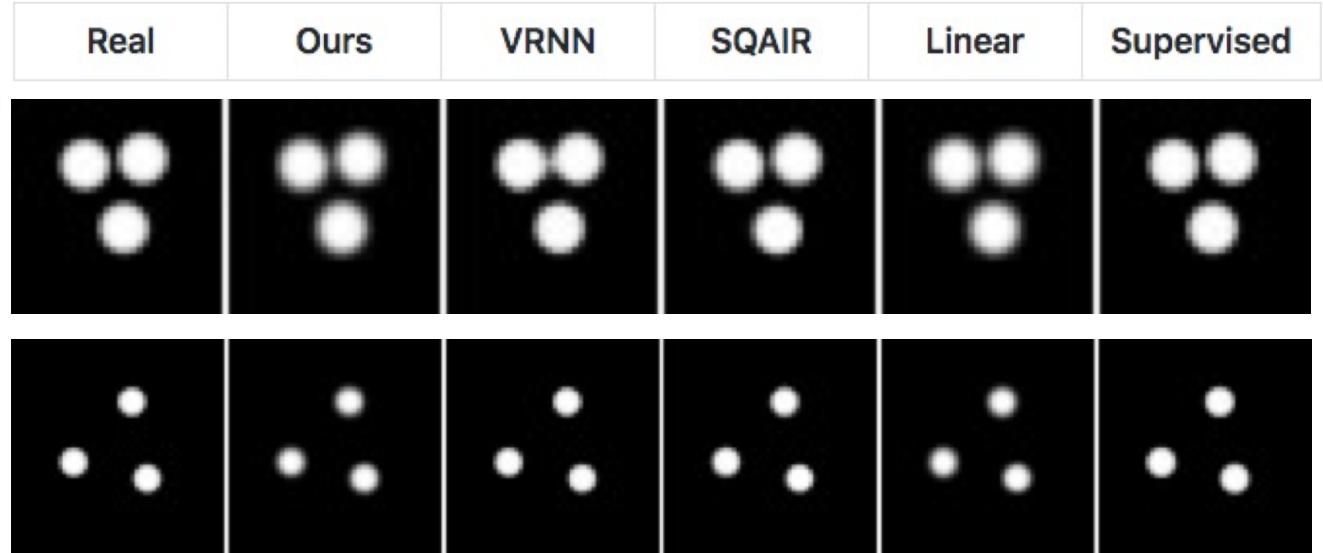
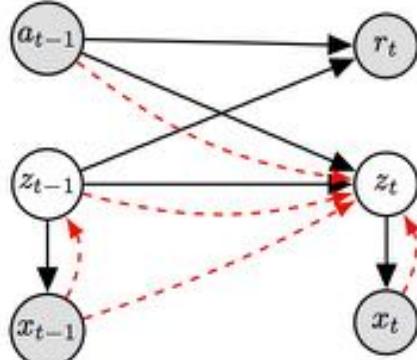
[Kossa, Stelzner, Hussing, Voelcker, Kersting ICLR 2020]

ICLR | 2020

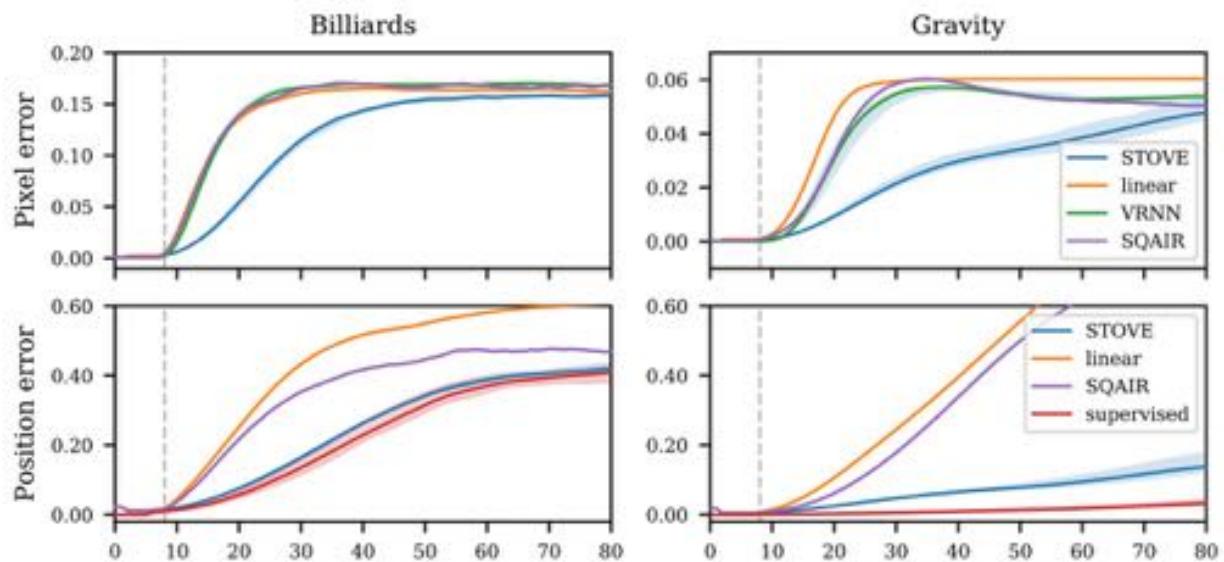
Eighth International Conference on
Learning Representations



TECHNISCHE
UNIVERSITÄT
DARMSTADT



putting
structure and
tractable
inference into
deep models



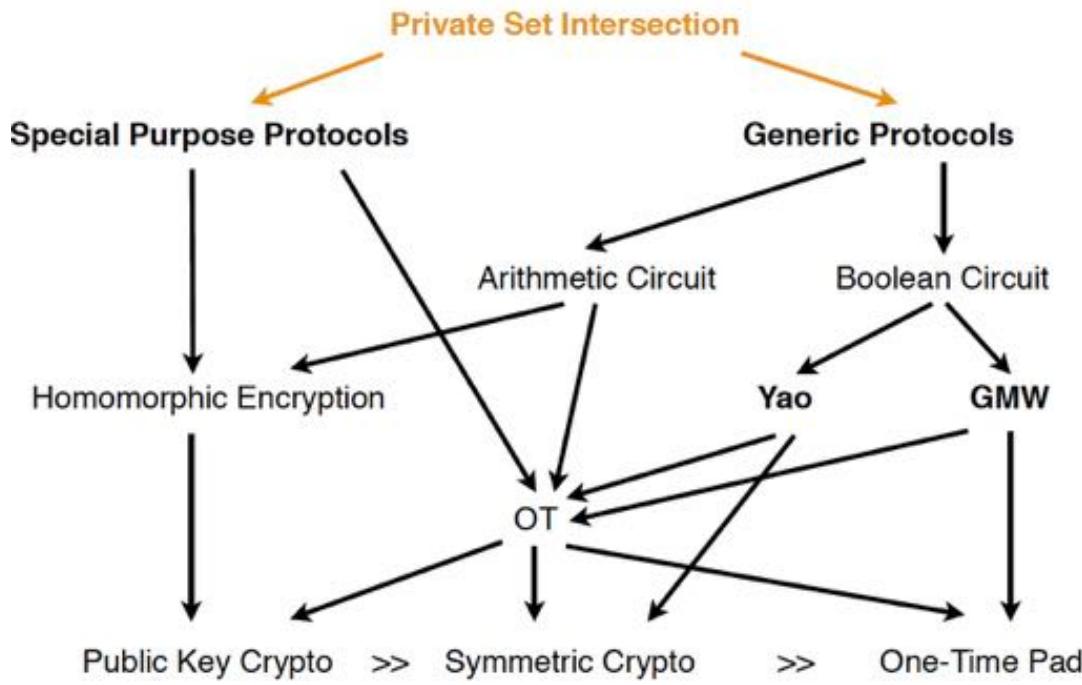
Semi-ring structure allows one to go beyond the standard settings

Edge AI, Crypto, Rankings, AutoML



Dataset	Rows	CPU (μs)	T-CPU (rows/ μs)	CPUF (μs)	T-CPUF (rows/ μs)	GPU (μs)	T-GPU (rows/ μs)	FPGA Cycle Counter	FPGAC (μs)	T-FPGAC (rows/ μs)	FPGA (μs)	T-FPGA (rows/ μs)
Accidents	17009	2798.27				7.87	63000.94	0.27			696.00	24.44
Audio	20000	4271.78				5.4			20317		761.00	26.28
Netflix	20000	4892.22				4.8			20322		654.00	30.58
MSNBC200	388434	15476.05				30.5			388900	19	608.00	77.56
MSNBC300	388434	10060.78				41.2			388810	19	933.00	78.74
NLTCS	21574	791.80				31.2			21904		566.00	38.12
Plants	23215	3621.71	6.41	3521.04		6.59	67004.41	0.35	23592	117.96	196.80	778.00
NIPS5	10000	25.11	398.31	26.37		379.23	8210.32	1.22	10236	51.18	195.39	337.30
NIPS10	10000	83.60	119.61	84.39		118.49	11550.82	0.87	10279	51.40	194.57	464.30
NIPS20	10000	191.30	52.27	182.73	54.72	18689.04	0.54	10285	51.43	194.46	543.60	18.40
NIPS30	10000	387.61	25.80	349.84	28.58	25355.93	0.39	10308	51.80	193.06	592.30	16.88
NIPS40	10000	551.64	18.13	471.26	21.22	30820.49	0.32	10306	51.53	194.06	632.20	15.82
NIPS50	10000	812.44	12.31	792.13	17.62	36355.60	0.28	10559	52.80	189.41	720.60	13.88
NIPS60	10000	1046.38	9.56	662.53	15.09	40778.36	0.25	12271	61.36	162.99	799.20	12.51
NIPS70	10000	1148.17	8.71	1134.80	8.81	46759.26	0.21	14022	70.11	142.63	858.60	11.65
NIPS80	10000	1556.99	6.42	1277.81	7.83	63217.99	0.16	14275	78.51	127.37	961.80	10.40

How do we do deep learning offshore?



There are generic protocols to validate computations on authenticated data without knowledge of the secret key

DNA MSPN ####
 Gates: 298208 Yao Bytes: 9542656 Depth: 615

DNA PSPN ####
 Gates: 228272 Yao Bytes: 7304704 Depth: 589

NIPS MSPN ####
 Gates: 1001477 Yao Bytes: 32047264 Depth: 970

Secure multi-party computation for SPNs?

Crypto SPNs

[Molina, Weinert, Treiber, Schneider, Kersting ECAI 2020]



From Probabilities to Preferences

Rain	R	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
Neighbour's Grad	N	T	T	T	T	F	F	F	T	T	T	F	F	F	F	F
Grass	H	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F
Sprinkler	S	T	F	T	F	T	F	T	F	T	F	T	F	T	F	F
	Rank	4	1	∞	∞	∞	∞	∞	5	∞	8	2	3	∞	6	0

Let us abstract probabilities into rankings

Not surprising events = rank 0

Surprising = rank 1

Very surprising = rank 2

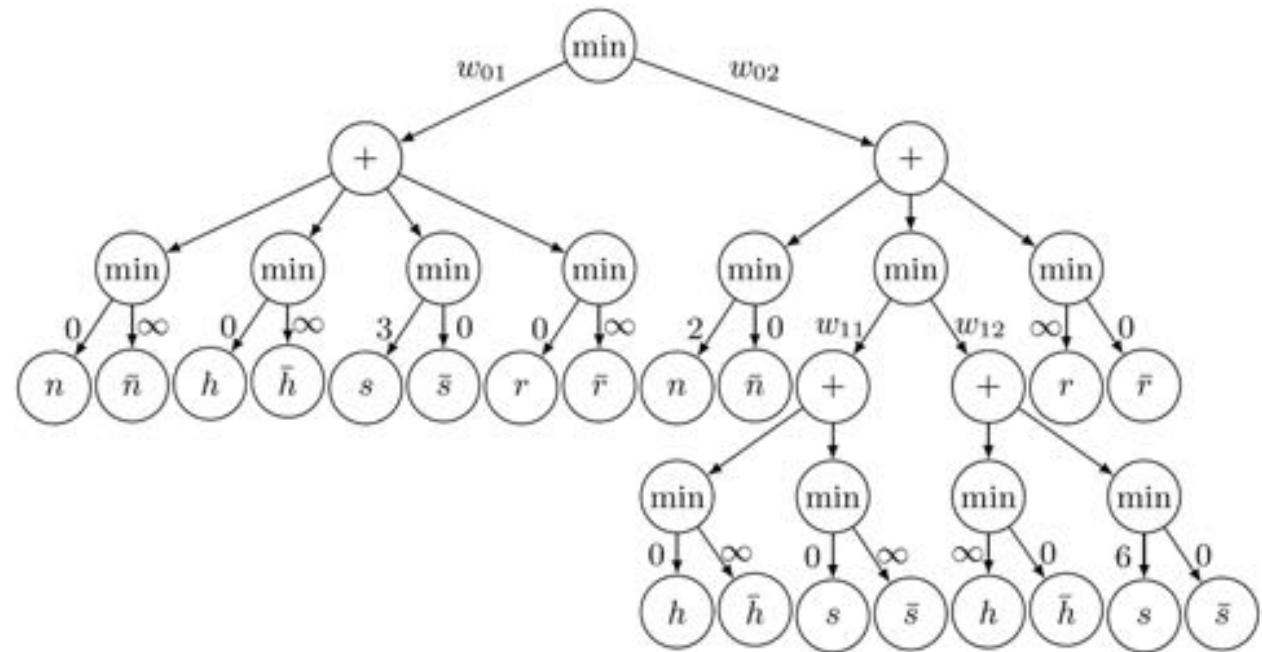
...



Rain	R	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F
Neighbour's Grad	N	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F
Grass	H	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F
Sprinkler	S	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T
Rank	4	1	∞	5	∞	8	2	3	∞	6						

Min-Sum Networks

Let us abstract probabilities
Not surprising events = 1
Surprising = rank 1
Very surprising = rank 2

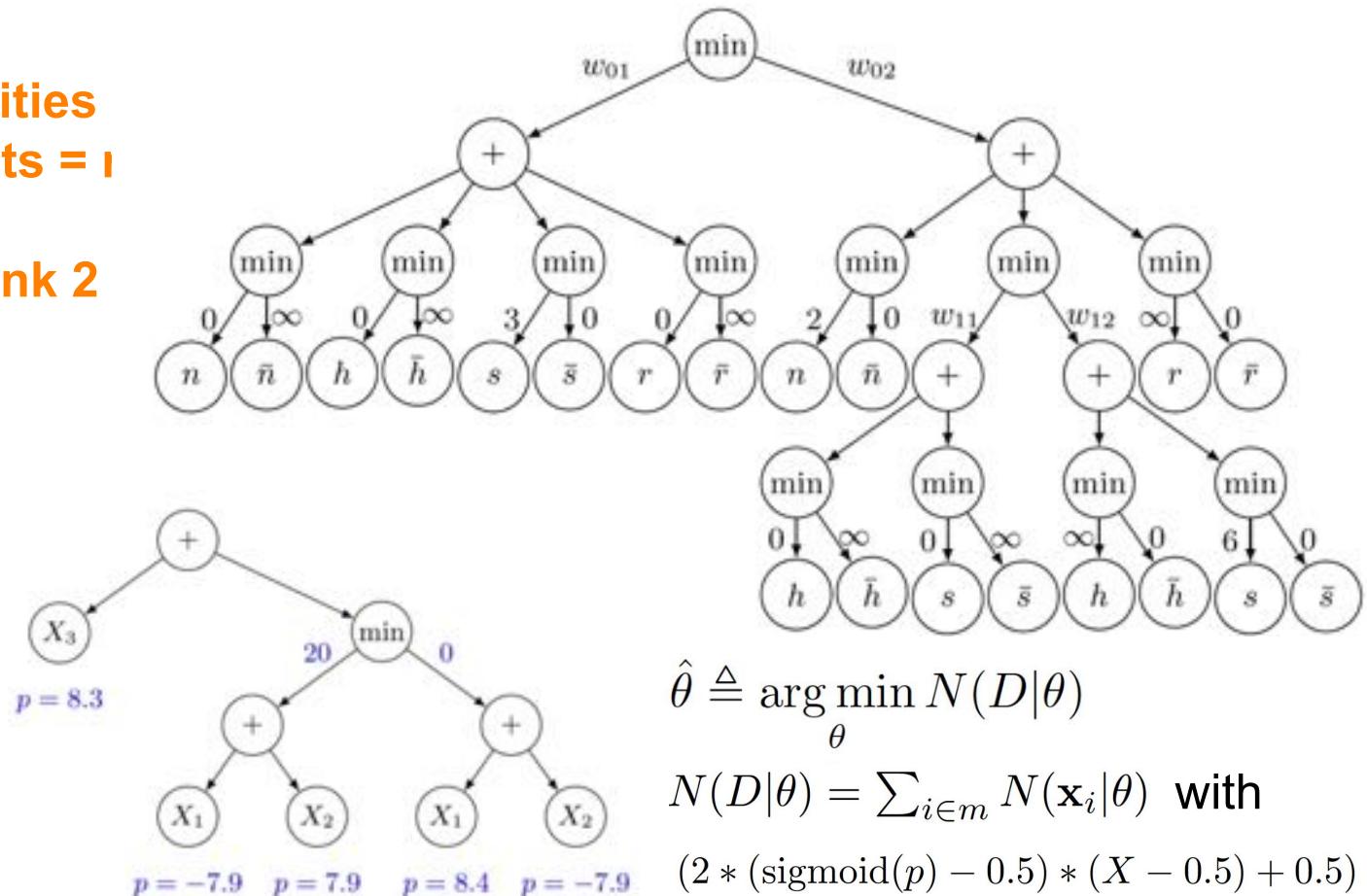
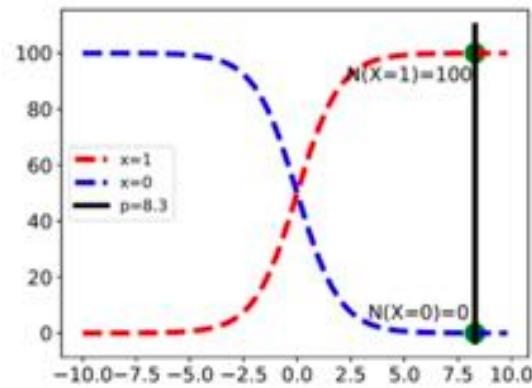


From Probabilities to Preferences

Rain	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
Neighbour's Grad	N	T	T	T	T	F	F	F	T	T	T	F	F	F	F
Grass	H	T	T	F	F	T	T	F	F	T	T	F	T	T	F
Sprinkler	S	T	F	T	F	T	F	T	F	T	F	T	F	T	F
Rank	4	1	∞	∞	∞	∞	∞	5	∞	8	2	3	∞	6	0

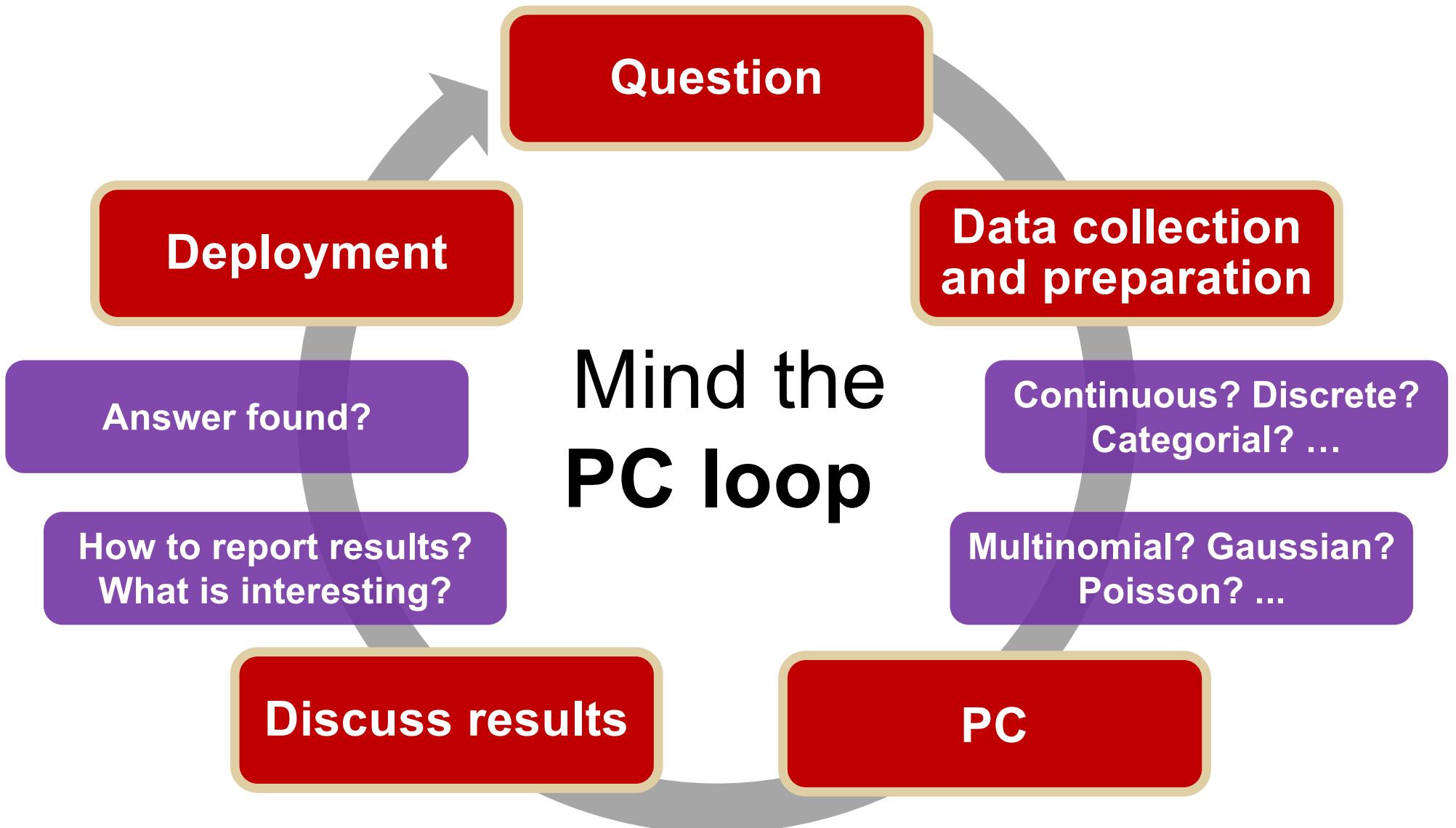
Let us abstract probabilities
 Not surprising events = 1
 Surprising = rank 1
 Very surprising = rank 2

...

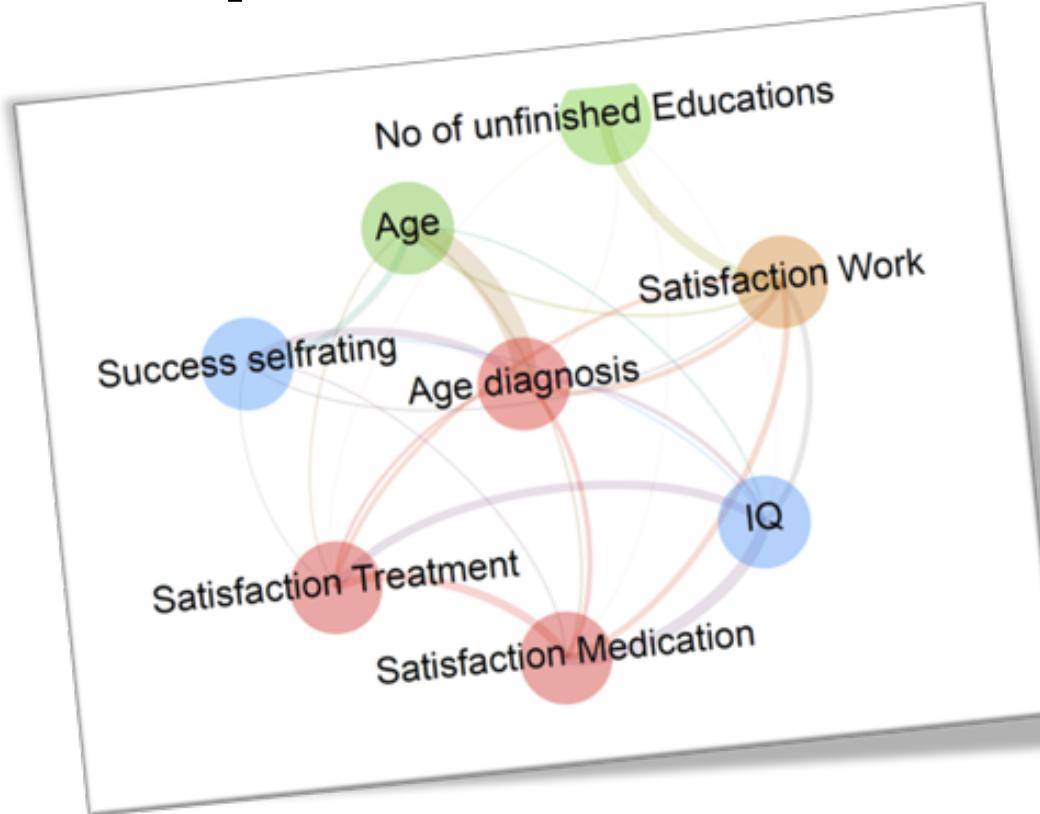




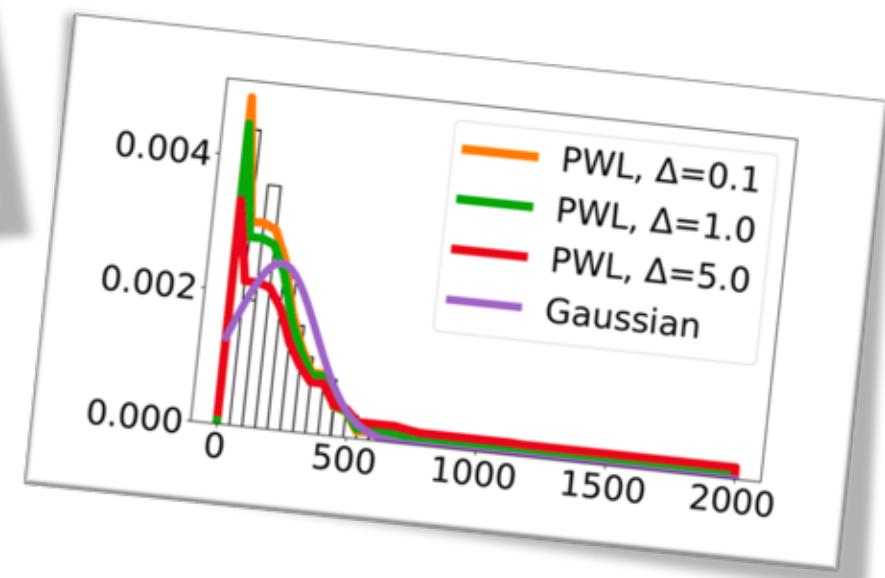
Automated Density Estimation



Distribution-agnostic Deep Probabilistic Learning

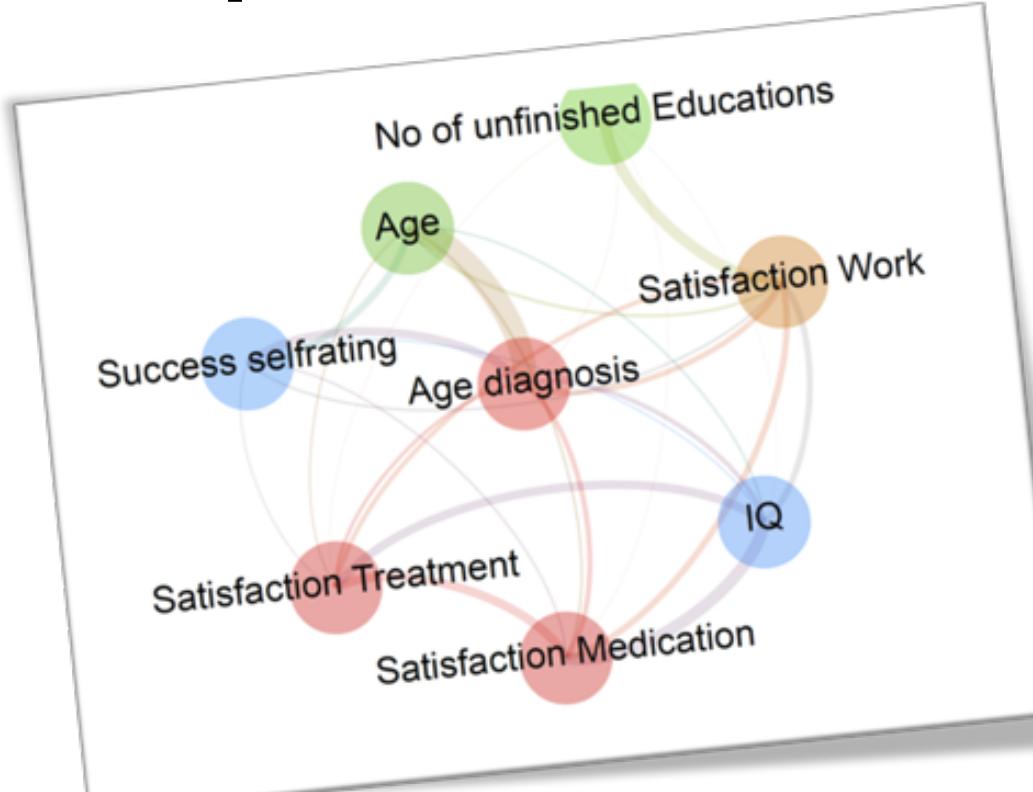


Use nonparametric
independency tests
and piece-wise linear
approximations

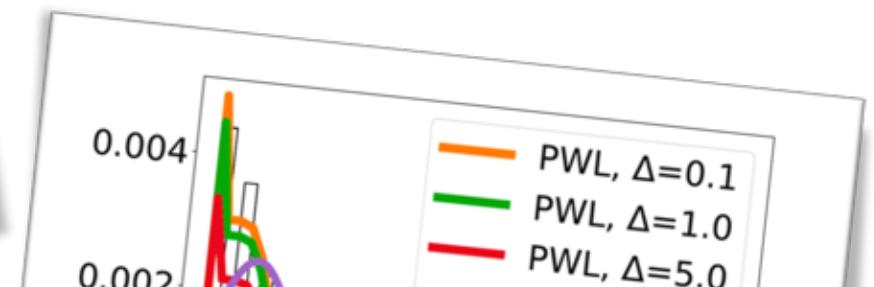




Distribution-agnostic Deep Probabilistic Learning



Use nonparametric
independency tests
and piece-wise linear
approximations



However, we have to provide the statistical types and do not gain insights into the parametric forms of the variables.
Are they Gaussians? Gammas? ...

The Explorative Automatic Statistician

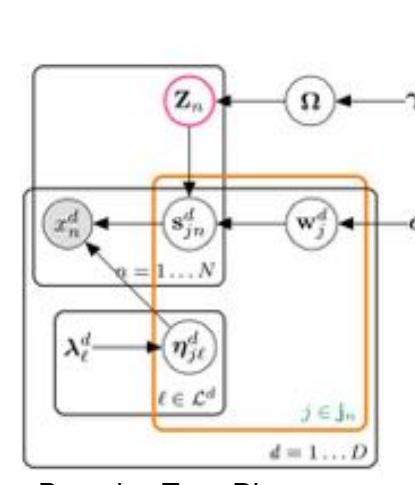


UNIVERSITY OF
CAMBRIDGE

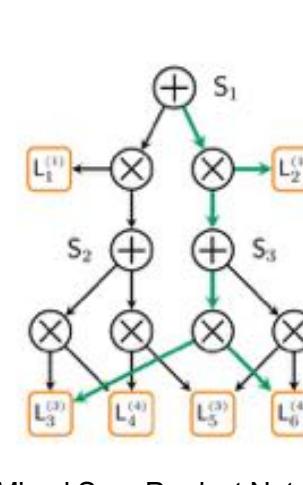


	X^1	X^2	X^3	X^4	X^5
x_8					
x_7		?			
x_6					
missing value	x_5	?			
x_4			?		
x_3					
x_2		?			
x_1					

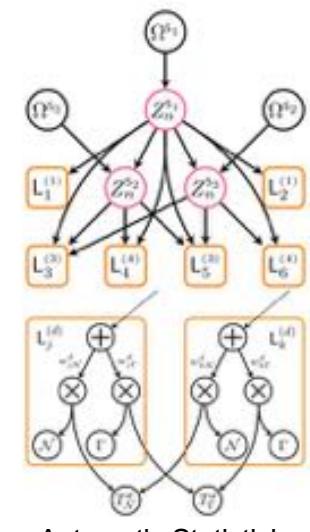
We can even automatically discovers the statistical types and parametric forms of the variables



Bayesian Type Discovery



Mixed Sum-Product Network

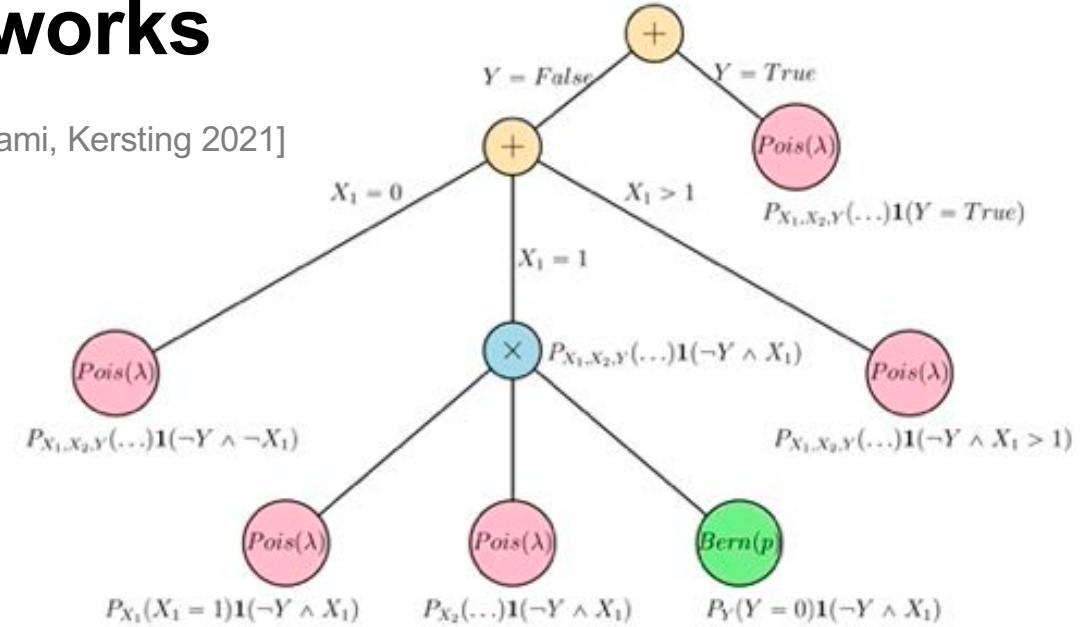
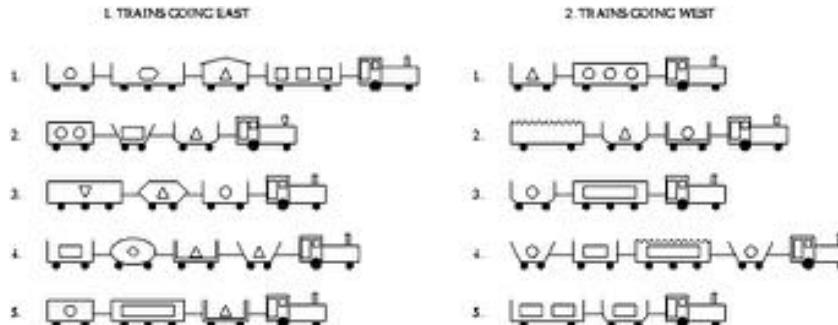


Automatic Statistician

Generative Clausal Networks

[Correia, Peharz, de Campos NeurIPS 2020, Ventola, Dhami, Kersting 2021]

Propositionalization



First-order feature construction

$f1(T):-\text{hasCar}(T,C), \text{clength}(C,\text{short}).$

$f2(T):-\text{hasCar}(T,C), \text{hasLoad}(C,L),$
loadShape(L,circle)

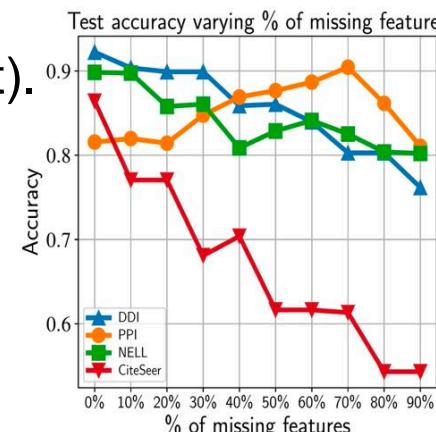
$f3(T) :- \dots$

Propositional Learning:

$t(T) \leftarrow f1(T), f4(T)$

PC learning:

$\#\!f1(T), \#\!f2(T), \#\!f2(T)$



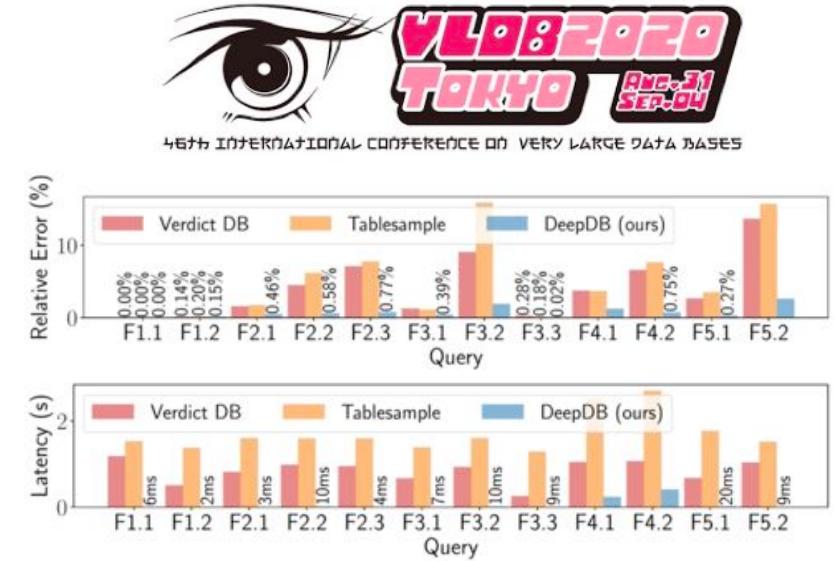
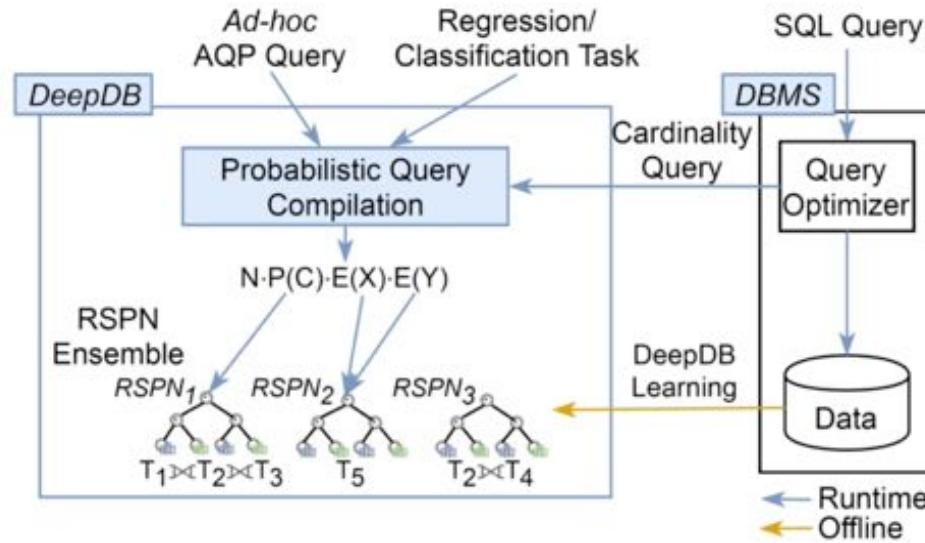
Data	Methods	Accuracy	AUC-ROC	AUC-PR	Data	Methods	Accuracy	AUC-ROC	AUC-PR
<i>DDI</i>	LR	93.79	87.62	83.64	<i>NELL</i>	LR	83.96	58.90	28.27
	GB	86.77	86.17	67.95		GB	87.69	75.28	48.07
	NN	87.54	85.03	69.21		NN	88.26	74.42	48.35
	DT	85.52	83.22	65.12		DT	87.27	71.77	45.71
	TILDE	72.52	73.43	70.79		TILDE	81.48	86.27	78.23
<i>PPI</i>	RDN-B	75.54	82.87	83.13	<i>CiteSeer</i>	RDN-B	81.26	88.47	83.41
	MLN-B	63.80	79.83	78.40		MLN-B	60.54	89.44	85.30
	GCLN-B	85.05	70.57	70.84		GCLN-B	80.74	38.72	23.15
	GCLN-P	92.22 ↑	87.53 ↑	83.81 ↑		GCLN-P	89.83 ↑	89.44 ↑	56.39 ↑
	LR	78.13	81.54	52.44		LR	76.33	83.78	53.30
<i>NELL</i>	GB	77.21	78.25	49.54	<i>CiteSeer</i>	GB	96.05	97.21	87.24
	NN	76.94	75.75	47.49		NN	96.50	96.88	88.10
	DT	76.47	77.52	48.71		DT	95.33	96.79	85.38
	TILDE	62.20	62.87	58.27		TILDE	91.47	83.33	73.45
	RDN-B	67.15	72.84	74.02		RDN-B	94.72	97.11	89.23
<i>CiteSeer</i>	MLN-B	54.87	74.39	73.34		MLN-B	81.98	94.67	80.54
	GCLN-B	79.72	82.75	59.43		GCLN-B	77.18	71.34	41.20
	GCLN-P	81.56 ↑	91.16 ↑	80.08 ↑		GCLN-P	86.42 ↑	71.57 ↑	42.57 ↑

Partial Counts

As observations: $P(X > k) = 1 - \sum_{i=0}^k P(X = i)$

Deep Databases

[Hilprecht, Schmidt, Kulessa, Molina, Kersting, Binnig VLDB 2020]



Hybrid AI: Sum Product Logic

= Probabilistic Circuits + (Deep NNs + (Probabilities + Logic Programming))

Sum Product Logic

DeepProbLog

ProbLog



[Skryagin, Stelzner, Molina, Ventola, Yu, Kersting BBPR-GNN+ 2020]



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Multi-Output GP-SPNs

[Yu, Zhu, Trapp, Skryagin, Kersting UAI 2021]

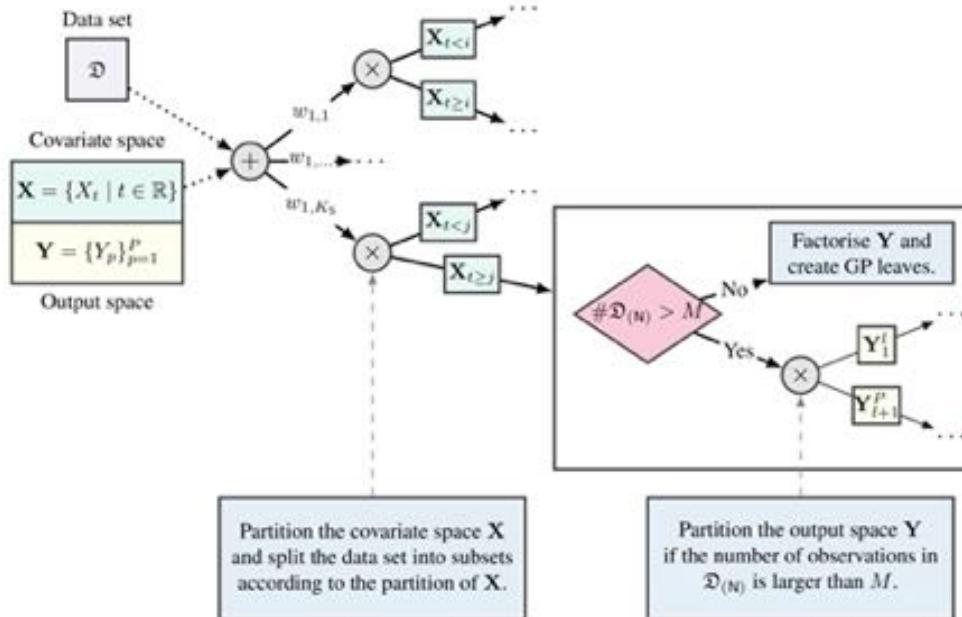
37th Conference on Uncertainty in Artificial Intelligence
July 27-30, 2021
Online
uai2021



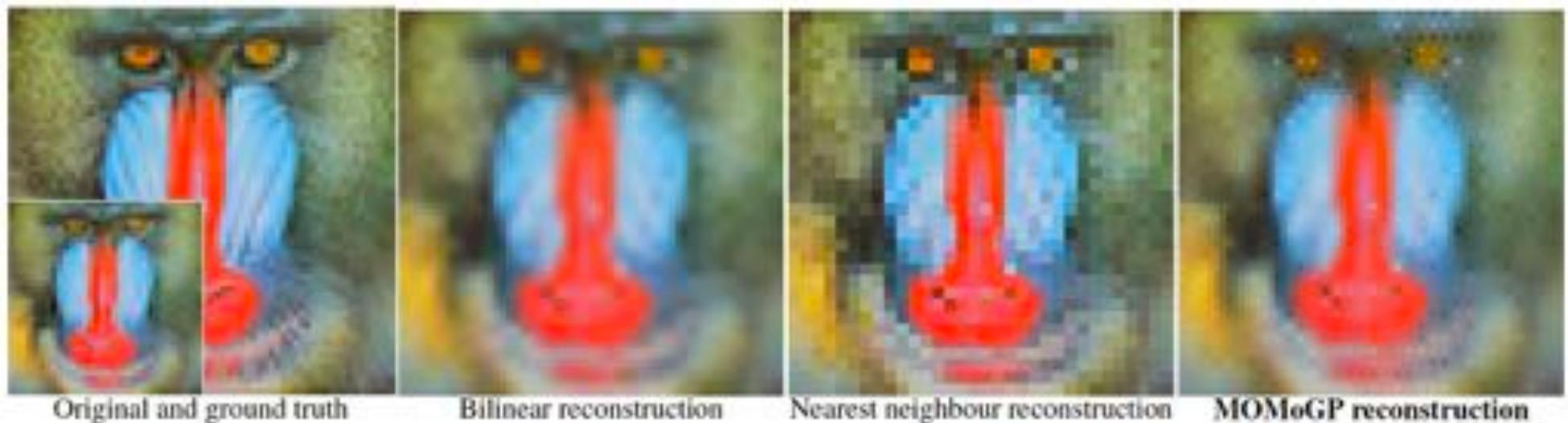
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Nanjing University
of Aeronautics & Astronautics

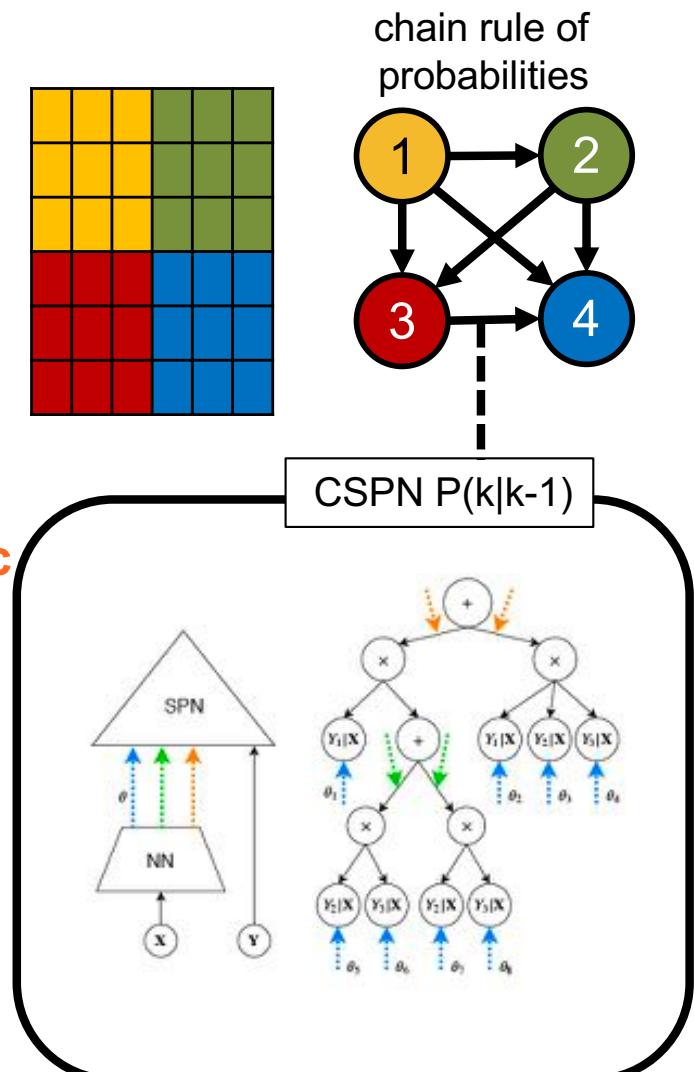


Data Set	LR	GP	MOGP	MOSVGP	DSMGP	sumGP	MOMoGP
Parkinsons	RMSE	0.974	0.783	0.793	0.864	0.774	0.784
	MAE	0.816	0.610	0.603	0.708	0.604	0.610
	NLPD	2.787	2.389	1.766	2.515	2.319	2.388
scm20d	RMSE	0.854	0.832	0.816	0.824	0.839	0.829
	MAE	0.652	0.643	0.630	0.636	0.646	0.641
	NLPD	21.876	21.013	21.310	17.319	19.059	14.859
WindTurbine	RMSE	0.391	0.133	0.139	0.302	0.143	0.133
	MAE	0.311	0.074	0.080	0.236	0.073	0.074
	NLPD	1.435	-2.649	2.594	0.104	-8.749	-2.627
Energy	RMSE	0.752	NA	NA	0.659	0.547	NA
	MAE	0.605	NA	NA	0.516	0.400	NA
	NLPD	14.775	NA	NA	14.169	11.745	NA
usFlight	RMSE	0.983	NA	NA	0.955	0.927	NA
	MAE	0.529	NA	NA	0.494	0.492	NA
	NLPD	2.331	NA	NA	2.251	2.178	NA



Putting a little bit of structure into SPN models allows one to realize autoregressive deep models akin to PixelCNNs

[van den Oord et al. NIPS 2016]

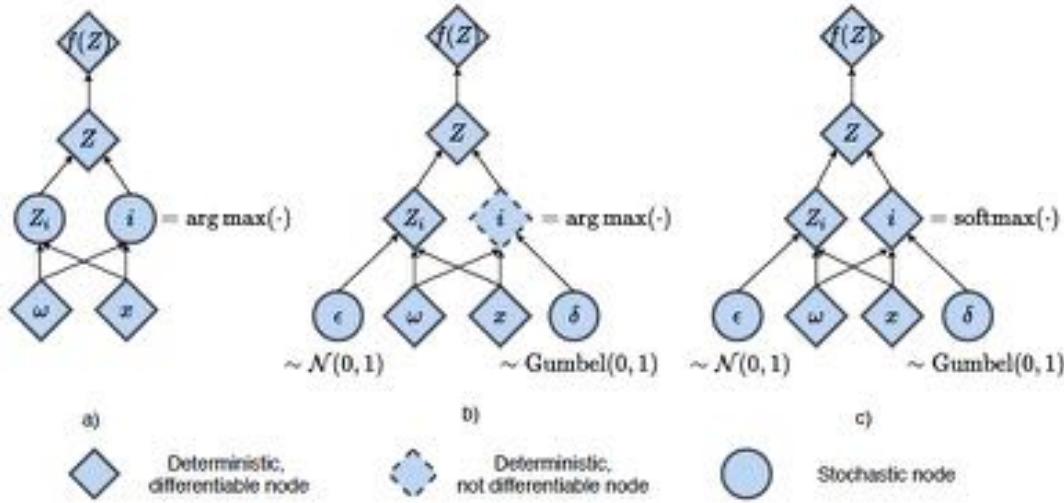
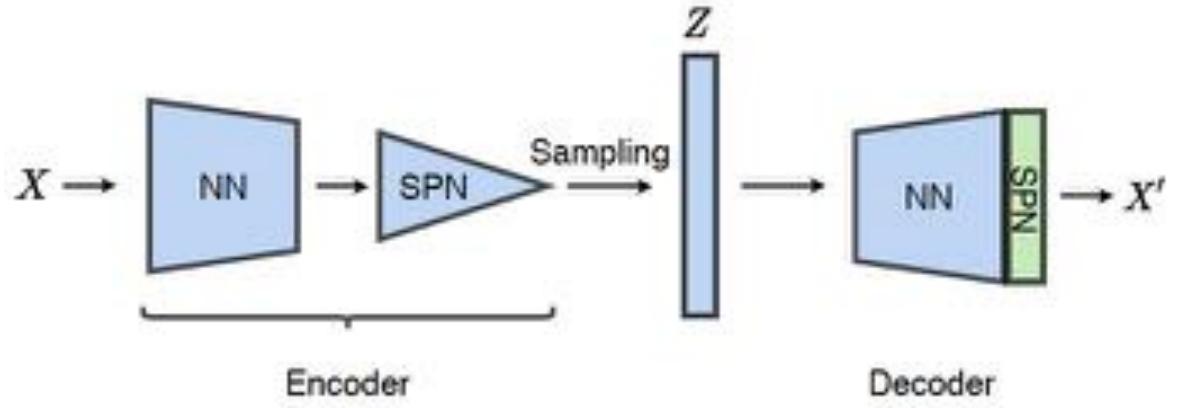


Learn Conditional SPN (CSPNs) by non-parametric conditional independence testing and conditional clustering [Zhang et al. UAI 2011; Lee, Honavar UAI 2017; He et al. ICDM 2017; Zhang et al. AAAI 2018; Runge AISTATS 2018] encoded using gating functions

Conditional SPNs

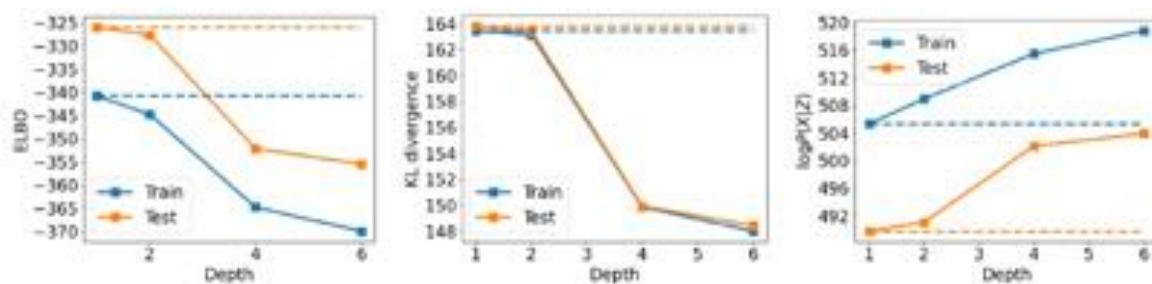
[Shao, Molina, Vergari, Peharz, Liebig, Kersting PGM 2020] **PGM** 2020

Use a (neural) CPSN as encoder and decoder together with differentiable sampling



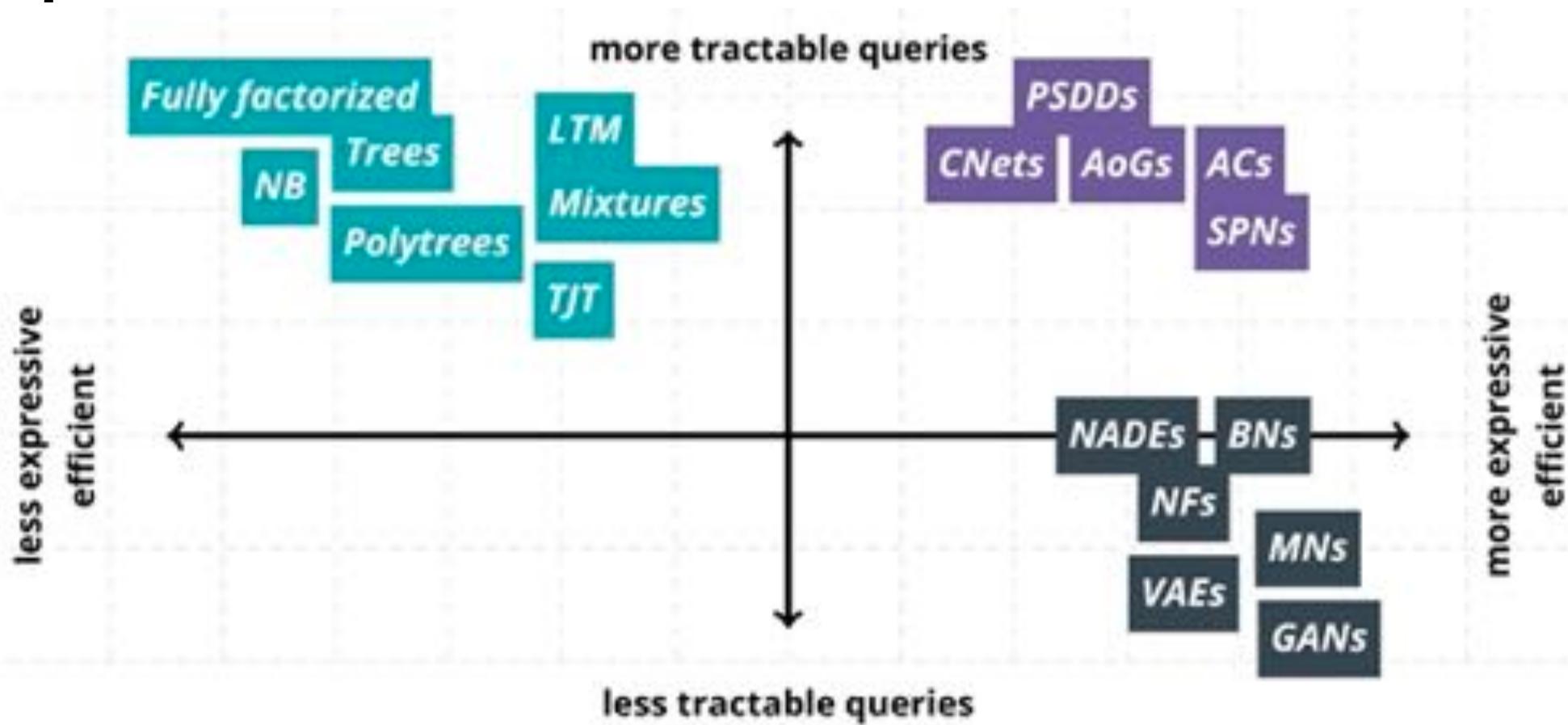
Sum-Product VAE

[Shao, Molina, Vergari, Peharz, Liebig, Kersting 2021]



(e) On CelebA. Prior: isotropic multivariate Gaussian.

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



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

