NeurASP: Neural-Probabilistic Answer Set Programming

Skryagin, A., Stammer, W., Ochs, D., Singh Dhami, D.,& Kristian, K. (2022). Neural-Probabilistic Answer Set Programming.

Presented by:
John Cava
Joshua Elkins
Sahil Patel

28 September 2022

Overview

- Introduction
- Neural Probabilistic Predicates
- Probabilistic Circuits
- SLASH
- Empirical Illustrations
- Conclusion and Related Works

Introduction

Neuro-Symbolic AI

This field's approach to learning recently gained popularity.

- Blend low-level perception with high-level reasoning by combining data-driven neural modules and logic-based symbolic modules
- Creates several advantages for various tasks:
  - Visual question answering and reasoning
  - Concept learning
  - Improved properties for explainable and revisable models

The Problem with Neuro-Symbolic Architectures

Problem:

- Have to design specific neuro-symbolic architectures where we often disjoint the neural and symbolic modules and train them independently

Solution:

- Use deep probabilistic programming languages as an alternative

What do Deep Probabilistic Programming Languages (DPPLs) do?

- Definition: A language for specifying both deep neural networks and probabilistic models.
  - Draws upon programming languages, Bayesian statistics, and deep learning to make the development of machine learning applications easier
- Integrate the neural and symbolic modules via a unifying programming framework
  - Acts as a sort of “glue” between separate modules and reason with noisy data
- Allows for prior knowledge and biases in the form of logical rules to easily be added into the learning process
- Easily assimilate neural networks into downstream logical reasoning tasks

Recently Created DPPLs

- DeepProbLog, NeurASP
- Allows conditional class probability estimates
  - Centers their probability estimates on neural predicates
- Necessary to be able to integrate and process joint probability estimates for the overall expressive and utilization power of a DPPL
- “The world is uncertain, and often it becomes necessary to reason in settings”
  - E.g., variables of an observation might be missing or manipulated

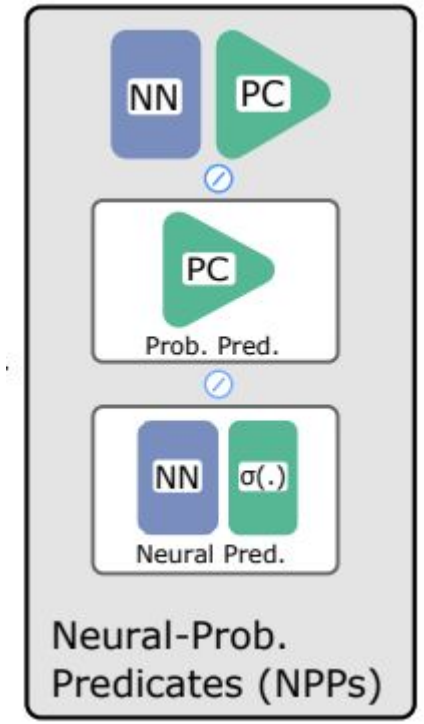
Neural Probabilistic Predicates

Neural Probabilistic Predicates (NPPs)

- A novel design principle that allows for the unification of all deep model types and combinations thereof to be represented as a single probabilistic predicate
- They consist of neural and/or probabilistic circuit (PC) modules
  - Acts as a unifying term that encompasses the neural predicates of DeepProbLog and NeurASP as well as purely probabilistic predicates

Neural-Probabilistic Predicates (NPPs)

- Neural Network + Probabilistic Circuit
- Probabilistic Circuit (Probabilistic Circuit)
- Neural Network with Softmax (Neural predicates)



Probabilistic Circuits

Probabilistic Circuits

- What are probabilistic circuits?

Probabilistic Circuits

- What are probabilistic circuits?

A probabilistic circuit $\mathcal{C}$ over variables $\mathbf{X}$ is a computational graph encoding a (possibly unnormalized) probability distribution $p(\mathbf{X})$

Vergari et al. Probabilistic Circuits: Representations, Inference, Learning, Applications.

Probabilistic Circuits

- What are probabilistic circuits?

Base case: a single node encoding a distribution

$\Rightarrow$ *e.g., Gaussian PDF continuous random variable*

Vergari et al. Probabilistic Circuits: Representations, Inference, Learning, Applications.

Probabilistic Circuits

- What are probabilistic circuits?

$\neg X$

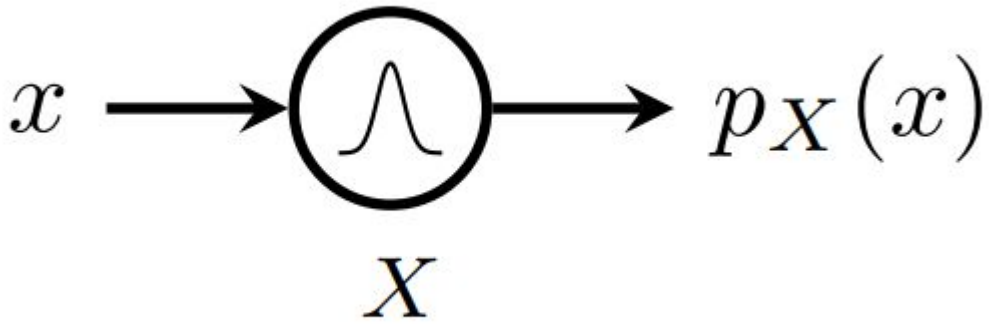
Base case: a single node encoding a distribution

$\Rightarrow$ *e.g., indicators for* X *or* $\neg X$ *for Boolean random variable*

Vergari et al. Probabilistic Circuits: Representations, Inference, Learning, Applications.

Probabilistic Circuits

- What are probabilistic circuits?

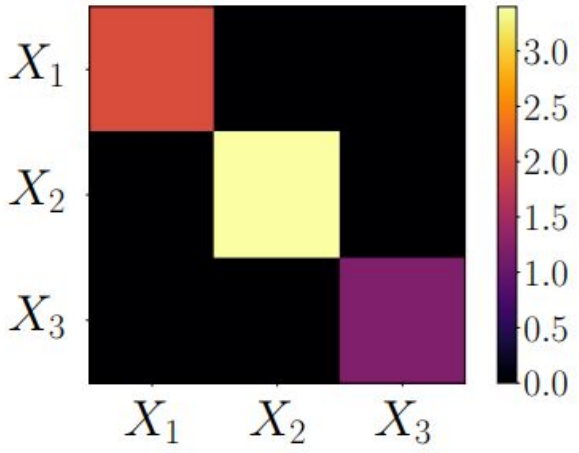


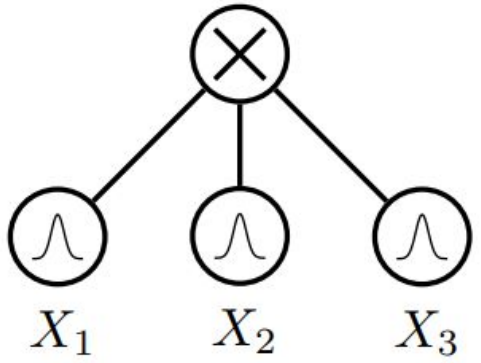
Vergari et al. Probabilistic Circuits: Representations, Inference, Learning, Applications.

Probabilistic Circuits

- Factorizations as product of nodes

$$p(X_1, X_2, X_3) = p(X_1) \cdot p(X_2) \cdot p(X_3)$$

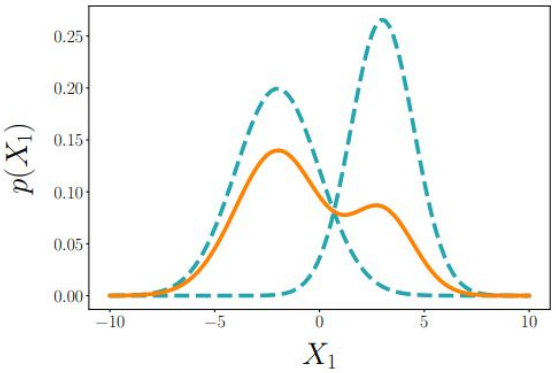




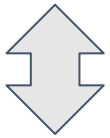
Vergari et al. Probabilistic Circuits: Representations, Inference, Learning, Applications.

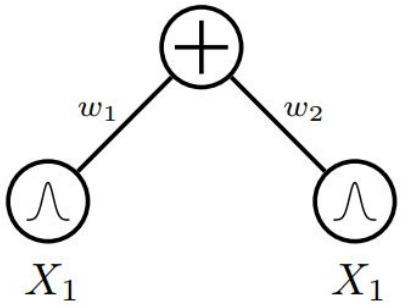
Probabilistic Circuits

- Mixtures as sum of nodes



$$p(X) = w_1 \cdot p_1(X) + w_2 \cdot p_2(X)$$



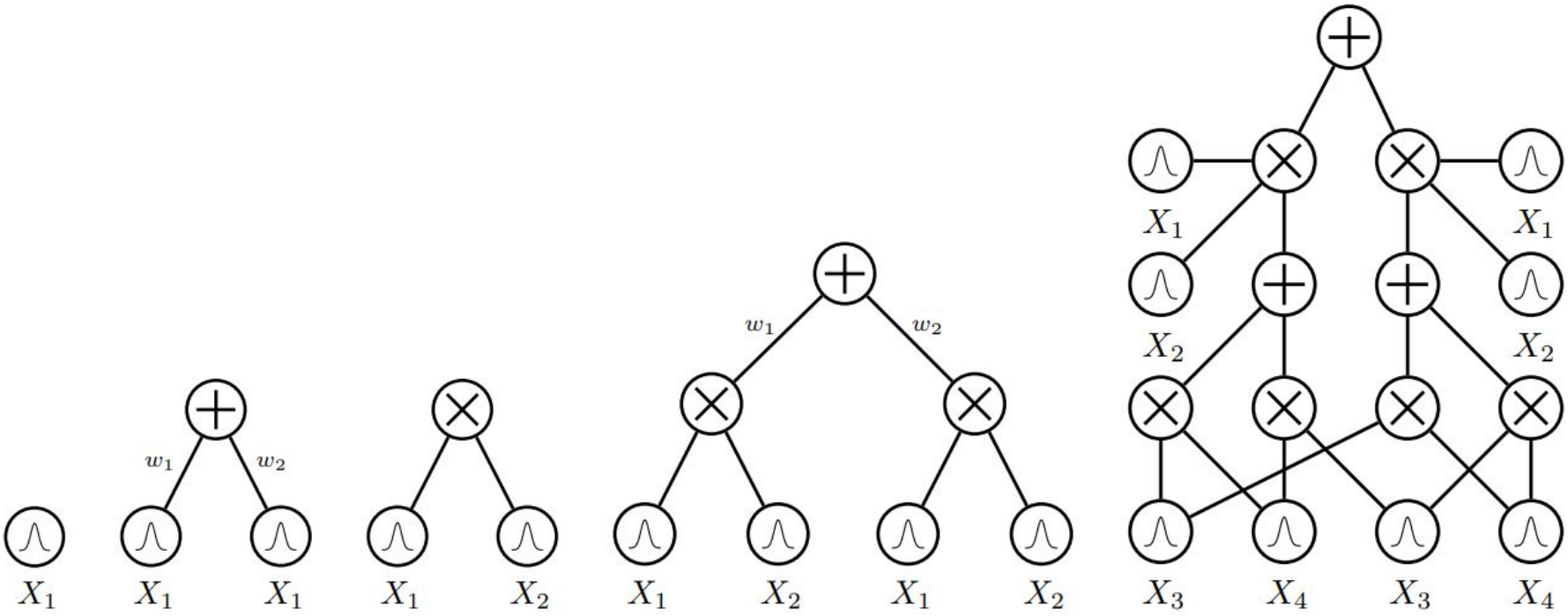


$$p(x) = 0.2 \cdot p_1(x) + 0.8 \cdot p_2(x)$$

Vergari et al. Probabilistic Circuits: Representations, Inference, Learning, Applications.

Probabilistic Circuits

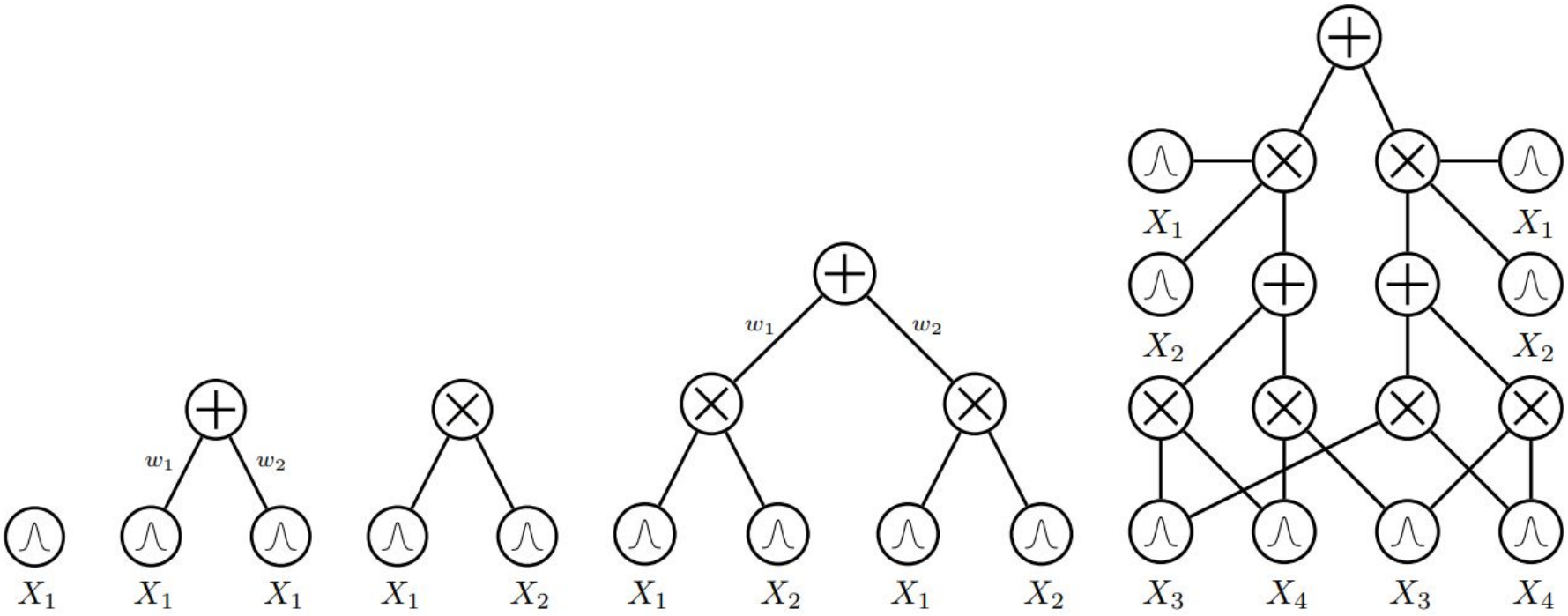
- Build a family of circuits



Vergari et al. Probabilistic Circuits: Representations, Inference, Learning, Applications.

Probabilistic Circuits

- Sound Familiar?



Vergari et al. Probabilistic Circuits: Representations, Inference, Learning, Applications.

Probabilistic Circuits

- So what?

Probabilistic Circuits

- So what?

“In this way, with PCs it is possible to answer a much richer set of probabilistic queries, i.e. P(X, C), P(X|C), P(C|X) and P(C).”

Probabilistic Circuits

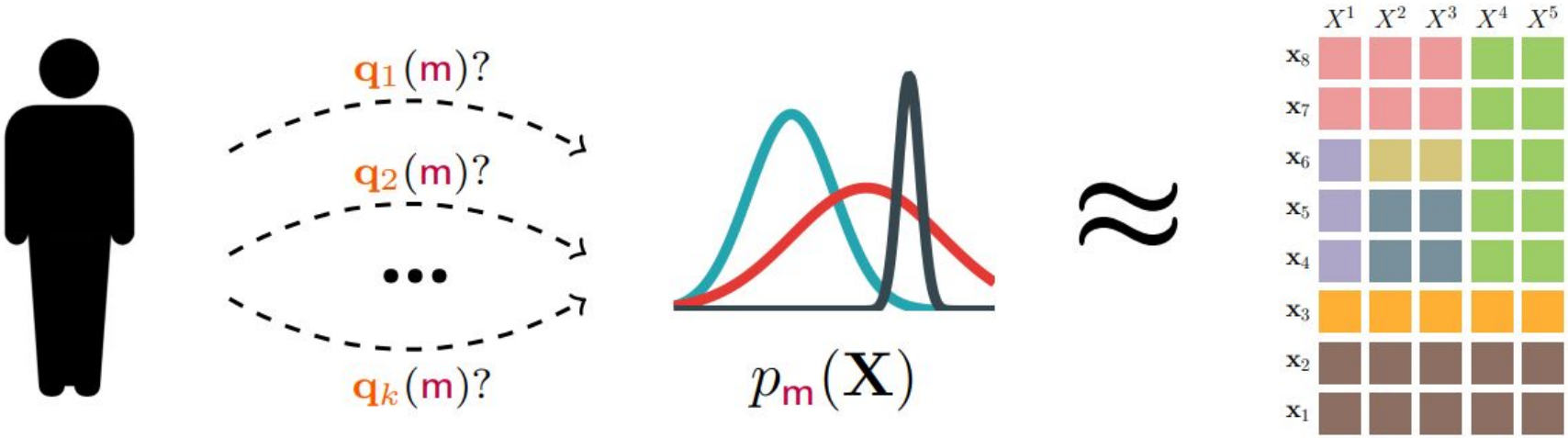
- So what?

"In this way, with PCs it is possible to answer a much richer set of probabilistic queries, i.e. P(X, C), P(X|C), P(C|X) and P(C)."

- color_attr(+X, +C) ⇔ P(X,C)
- color_attr(+X, -C) ⇔ P(C|X)
- color_attr(-X, +C) ⇔ P(X|C)
- color_attr(-X, -C) ⇔ P(C)

Probabilistic Circuits

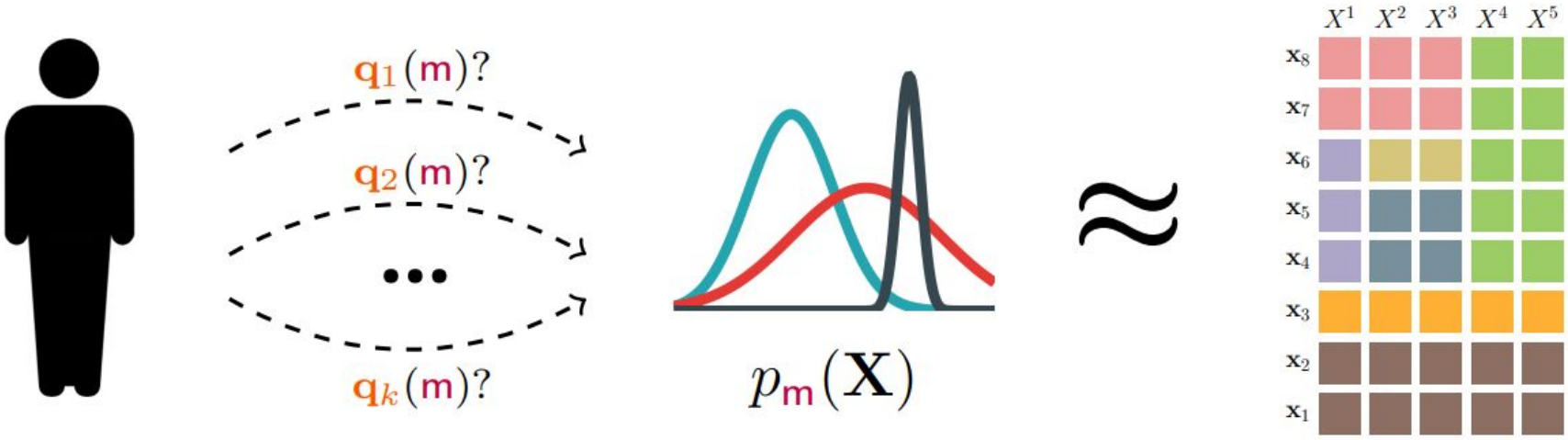
- Moreover, PCs are Generative



Vergari et al. Probabilistic Circuits: Representations, Inference, Learning, Applications.

Probabilistic Circuits

- Thus, we can work with missing data



Vergari et al. Probabilistic Circuits: Representations, Inference, Learning, Applications.

Neural Probabilistic Predicates

- The Loss

$$L_{SLASH} = L_{NPP} + L_{ENT}$$

Neural Probabilistic Predicates

- The Loss

$$L_{SLASH} = L_{NPP} + L_{ENT}$$

- Loss for the Neural Probabilistic Predicate

$$L_{NPP} := -\log LH(x, \hat{x}) = \sum_{i=1}^{n} LH(x_i, \hat{x}_i) =$$

$$-\sum_{i=1}^{n} x_i \cdot \log(P_{\xi}^{(X,C)}(x_i)) = -\sum_{i=1}^{n} \log(P_{\xi}^{(X,C)})$$

Neural Probabilistic Predicates

- The Loss

$$L_{SLASH} = L_{NPP} + L_{ENT}$$

- Loss for the Neural Probabilistic Predicate

$$L_{NPP} := -\log LH(x, \hat{x}) = \sum_{i=1}^{n} LH(x_i, \hat{x}_i) =$$

$$-\sum_{i=1}^{n} x_i \cdot \log(P_{\xi}^{(X,C)}(x_i)) = -\sum_{i=1}^{n} \log(P_{\xi}^{(X,C)})$$

- Loss for “Entailment” (loss of queries when NNP is fixed)

$$L_{ENT} :=$$

$$-\log LH\left(\log(P_{\Pi(\theta)}(\mathbf{Q})), P^{(X_{\mathbf{Q}},C)}(x_{\mathbf{Q}})\right) =$$

$$-\sum_{j=1}^{m} \log(P_{\Pi(\theta)}(Q_{ij})) \cdot \log\left(P^{(X_{\mathbf{Q}},C)}(x_{Q_{ij}})\right).$$

Neural Probabilistic Predicates

- Loss for “Entailment” (loss of queries when NNP is fixed)

$$
\begin{aligned}
&L_{ENT} := \\
&\quad -\log LH\left(\log(P_{\Pi(\theta)}(\mathbf{Q})), P^{(X_{\mathbf{Q}},C)}(x_{\mathbf{Q}})\right) = \\
&\quad -\sum_{j=1}^{m} \log(P_{\Pi(\theta)}(Q_{ij})) \cdot \log\left(P^{(X_{\mathbf{Q}},C)}(x_{Q_{ij}})\right).
\end{aligned}
$$

- Derivative of L_ENT

$$
\sum_{Q\in\mathbf{Q}} \frac{\partial \log\left(P_{\Pi(\boldsymbol{\theta})}(Q)\right)}{\partial \boldsymbol{\theta}} = \sum_{Q\in\mathbf{Q}} \frac{\partial \log\left(P_{\Pi(\boldsymbol{\theta})}(Q)\right)}{\partial \mathbf{p}} \times \frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}}.
$$

Neural Probabilistic Predicates

- Loss for “Entailment” (loss of queries when NNP is fixed)

$$L_{ENT} := -\log LH\left(\log(P_{\Pi(\theta)}(\mathbf{Q})), P^{(X_{\mathbf{Q}},C)}(x_{\mathbf{Q}})\right) = -\sum_{j=1}^{m} \log(P_{\Pi(\theta)}(Q_{ij})) \cdot \log\left(P^{(X_{\mathbf{Q}},C)}(x_{Q_{ij}})\right).$$

- Derivative of L_ENT

$$\sum_{Q\in\mathbf{Q}} \frac{\partial \log\left(P_{\Pi(\boldsymbol{\theta})}(Q)\right)}{\partial \boldsymbol{\theta}} = \sum_{Q\in\mathbf{Q}} \frac{\partial \log\left(P_{\Pi(\boldsymbol{\theta})}(Q)\right)}{\partial \mathbf{p}} \times \frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}}.$$

Neural Probabilistic Predicates

- Loss for “Entailment” (loss of queries when NNP is fixed)

$$L_{ENT} := -\log LH\left(\log(P_{\Pi(\theta)}(\mathbf{Q})), P^{(X_{\mathbf{Q}},C)}(x_{\mathbf{Q}})\right) = -\sum_{j=1}^{m} \log(P_{\Pi(\theta)}(Q_{ij})) \cdot \log\left(P^{(X_{\mathbf{Q}},C)}(x_{Q_{ij}})\right).$$

- Derivative of L_ENT

$$\sum_{Q\in\mathbf{Q}} \frac{\partial \log\left(P_{\Pi(\boldsymbol{\theta})}(Q)\right)}{\partial \boldsymbol{\theta}} = \sum_{Q\in\mathbf{Q}} \frac{\partial \log\left(P_{\Pi(\boldsymbol{\theta})}(Q)\right)}{\partial \mathbf{p}} \times \frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}}.$$

NeurASP (Yang, Ishay, and Lee 2020)

$$\frac{\partial \log\left(P_{\Pi(\boldsymbol{\theta})}(Q)\right)}{\partial \mathbf{p}} = \frac{\sum_{\substack{I:I\models Q \\ I\models c=v}} \frac{P_{\Pi(\boldsymbol{\theta})}(I)}{P_{\Pi(\boldsymbol{\theta})}(c=v)} - \sum_{\substack{I:I,v'\models Q \\ I\models c=v', v\neq v'}} \frac{P_{\Pi(\boldsymbol{\theta})}(I)}{P_{\Pi(\boldsymbol{\theta})}(c=v')}}{\sum_{I:I\models Q} P_{\Pi(\boldsymbol{\theta})}(I)}.$$

Neural Probabilistic Predicates

- Loss for the Neural Probabilistic Predicate

$$L_{NPP} := -\log LH(x, \hat{x}) = \sum_{i=1}^{n} LH(x_i, \hat{x}_i) =$$

$$-\sum_{i=1}^{n} x_i \cdot \log(P_{\xi}^{(X,C)}(x_i)) = -\sum_{i=1}^{n} \log(P_{\xi}^{(X,C)})$$

- Derivative for Neural Probabilistic Predicate

$$\frac{\partial}{\partial \xi} L_{NPP} =$$

$$\frac{\partial x_{Q_i}}{\partial \xi} \left(-\sum_{i=1}^{n} \frac{1}{\left(P_{\xi}^{(X_{\mathbf{Q}},C)}(x_{Q_i}) \right)} \frac{\partial}{\partial x_{Q_i}} \left(P_{\xi}^{(X_{\mathbf{Q}},C)}(x_{Q_i}) \right) \right)$$

Neural Probabilistic Predicates

- Loss for the Neural Probabilistic Predicate

$$L_{NPP} := -\log LH(x, \hat{x}) = \sum_{i=1}^{n} LH(x_i, \hat{x}_i) =$$

$$-\sum_{i=1}^{n} x_i \cdot \log(P_\xi^{(X,C)}(x_i)) = -\sum_{i=1}^{n} \log(P_\xi^{(X,C)})$$

- Derivative for Neural Probabilistic Predicate

$$\frac{\partial}{\partial \xi} L_{NPP} =$$

$$\frac{\partial x_{Q_i}}{\partial \xi} \left(-\sum_{i=1}^{n} \frac{1}{\left(P_\xi^{(X_\mathbf{Q},C)}(x_{Q_i})\right)} \frac{\partial}{\partial x_{Q_i}} \left(P_\xi^{(X_\mathbf{Q},C)}(x_{Q_i})\right) \right)$$

Dependent on NNP

SLASH

SLASH

- We can now construct a novel DPPL, called SLASH, to efficiently combine NPPs with logic programming
- SLASH acts similar to the punctuation symbol, where it can be used to efficiently combine several paradigms into one
- SLASH represents the first efficient and scalable programming language that seamlessly integrates probabilistic logical programming with neural representations and tractable probabilistic estimations
  - This allows for the integration of all forms of probability estimates that is not just class conditionals

SLASH (Continued)

- SLASH consists of a logical program where it contains a set of facts and logical statements that help define the state of the world of an underlying task
- An Answer Set Programming (ASP) module is used to combine with the NPP(s) with the logic program
  - ASP = logic programming paradigm that does combinatorial searches and knowledge intensive tasks

SLASH (Continued)

- When given a logical query about the input data, the ASP module will produce a probability estimate about the truth value of the query
  - Input data = the logical program and the probability estimates obtained from the NPP(s)
- Training in SLASH is efficiently performed in a batch-wise, end-to-end fashion where we integrate the parameters of all modules into a single loss term

SLASH Attention

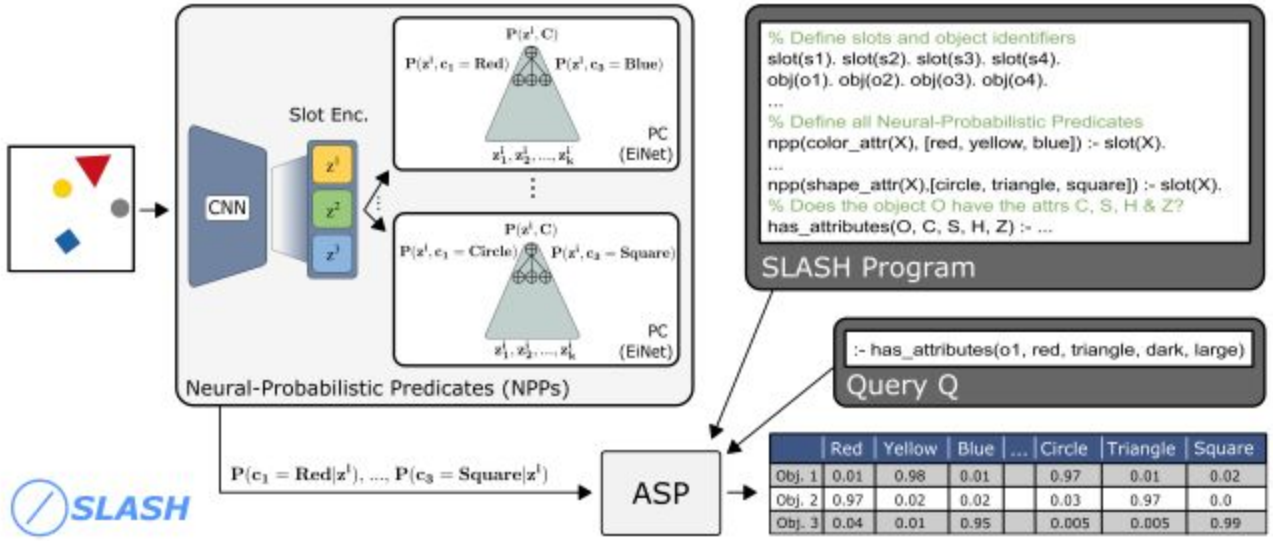


Figure 1: NPPs consist of neural and/or probabilistic circuit modules and can produce task-specific probability estimates. In our novel DPPL, SLASH, NPPs are integrated with a logic program via an ASP module to answer logical queries about data samples. In the depicted instantiation of SLASH (*SLASH Attention*) the Neural-Probabilistic Predicates consist of a slot attention encoder and Probabilistic Circuits (PCs) realized via EiNets. The slot encoder is shared over all NPPs, where each EiNet computes a joint distribution at the root node, thereby, learning the joint distribution over slot encodings, z^i, and object attributes, C, of a specific category, e.g., color attributes. Via targeted queries to the NPPs, one can obtain task-related probabilities, e.g., conditional probabilities for a visual reasoning task.

Empirical Illustrations

Overview

Primary Advantage of SLASH: “efficient integration of neural, probabilistic and symbolic computations”

Did 5 evaluations to show the advantages of SLASH on DPPLs.

Details

Two Benchmark data sets: MNIST (MNIST-Addition) | ShapeWorld (object-centric set prediction)

Methodology: Present average + standard deviation of five runs

Use random seeds for parameter initialization

Shapeworld: Generate ShapeWorld4. Each image in SW4 utilizes 1-4 objects with a color, shade, shape, and size. There are 84 combinations for objects.

Performance is classification accuracies for MNIST-Addition

SW4 experiments based on average precision.

Generative MNIST-Addition only model not trained by discriminative fashion only.

Results

5 Evaluations:

Evaluation 1: SLASH outperforms SOTA DPPLs in MNIST-Addition.

Evaluation 2: Handling Missing Data with SLASH.

Evaluation 3: True density estimation allows for generative learning.

Evaluation 4: Improved Concept Learning via SLASH.

Evaluation 5: Improved Compositional Generalization with SLASH.

Evaluation 1

MNIST Addition: Determine handwritten digits and then sum them.

SLASH (PC) is lower than the rest, but SLASH (DNN) is more accurate. Reasoning: PCs learn true mixture densities and not conditional probabilities

	Test Acc. (%)
DeepProbLog	98.49 ± 0.18
NeurASP	98.21 ± 0.30
SLASH (PC)	95.52 ± 0.32
SLASH (DNN)	$\mathbf{98.99 \pm 0.04}$

	DeepProbLog	NeurASP	**SLASH** (DNN)	**SLASH** (PC)
50%	$\mathbf{97.73 \pm 0.12}$	25.75 ± 31.32	96.0 ± 0.02	97.59 ± 0.01
80%	76.07 ± 18.38	10.29 ± 0.83	95.4 ± 0.01	$\mathbf{96.8 \pm 0.00}$
90%	69.15 ± 29.15	13.28 ± 4.94	75.6 ± 0.18	$\mathbf{95.00 \pm 0.00}$
97%	32.46 ± 22.48	11.94 ± 0.95	67.4 ± 0.09	$\mathbf{84.2 \pm 0.03}$

Evaluation 2

MNIST-Addition: Missing data in which the pixels of the image are missing. SLASH (PC) is noted to have a reduced variation compared to the rest such as DeepProbLog

	Test Acc. (%)
DeepProbLog	98.49 ± 0.18
NeurASP	98.21 ± 0.30
SLASH (PC)	95.52 ± 0.32
SLASH (DNN)	$\mathbf{98.99 \pm 0.04}$

	DeepProbLog	NeurASP	**SLASH** (DNN)	**SLASH** (PC)
50%	$\mathbf{97.73 \pm 0.12}$	25.75 ± 31.32	96.0 ± 0.02	97.59 ± 0.01
80%	76.07 ± 18.38	10.29 ± 0.83	95.4 ± 0.01	$\mathbf{96.8 \pm 0.00}$
90%	69.15 ± 29.15	13.28 ± 4.94	75.6 ± 0.18	$\mathbf{95.00 \pm 0.00}$
97%	32.46 ± 22.48	11.94 ± 0.95	67.4 ± 0.09	$\mathbf{84.2 \pm 0.03}$

Evaluation 3

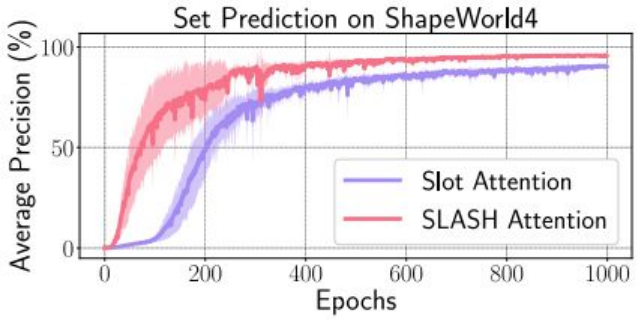
MNIST-Addition (Generative Learning): Models receive the output solution (Sum) and one of the digits handwritten, and it must output the missing digit. DPPLs cannot handle this, so output was compared to PC trained using EM.

Results: SLASH is able to correctly predict the missing digit.

Evaluation 4

The addition of logical constraints in SLASH Attention provides better accuracy and better Average Precision in less epochs compared to Slot Attention

	Slot Att.	**SLASH Att.**
	ShapeWorld4	
Test Set	90.24 ± 0.93	$\mathbf{95.58 \pm 0.61}$
	CoGenT	
Test Cond. A	90.37 ± 2.19	$\mathbf{96.85 \pm 0.43}$
Test Cond. B	27.15 ± 2.36	$\mathbf{40.58 \pm 1.99}$



Evaluation 5

Compositional Generalization with SLASH (Two Conditions)

- Condition A: squares with gray, blue, brown yellow; triangles with red, green, magenta, and cyan; circles with all colors
- Condition B: same as A, but test set has squares with colors of triangles' from training set, triangles with squares' colors from training set, and keeping circles with all colors.

SLASH Attention was able to perform 13% higher precision on condition B (Indicating higher generalizability)

Summary

Expressiveness and flexibility of SLASH can outperform the state-of-the-art.

Easy to combine NN, PCs, and logic with SLASH.

Results show that NPPs have great potential!

Conclusion and Related Works

Future Work

Interesting avenues to take the authors suggested for SLASH:

- Benchmarking SLASH on additional data types and tasks
- Explore unsupervised and weakly supervised learning
  - Investigate how far logical constraints can help unsupervised object discovery

Conclusion

- SLASH is a novel DPPL that integrates neural computations with tractable probability estimates and logical statements using Neural-Probabilistic Predicates
- Using Answer Set Programming, the logical SLASH program and probability estimates from NPPs are combined to estimate the truth value of a task-specific query
- SLASH demonstrates improved performances and generalizability

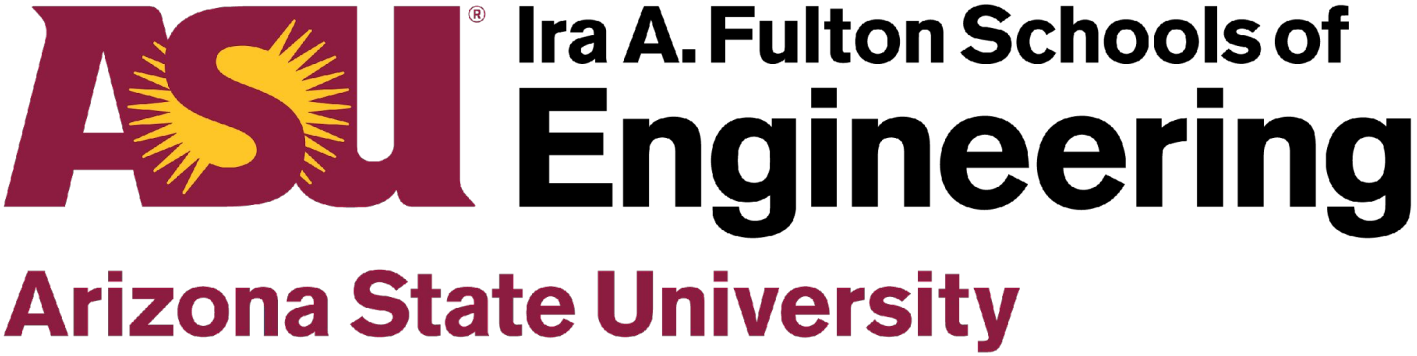
DPPLs to Look Into

Several DPPLs have been proposed:

- **Pyro**
  - A DPPL build on PyTorch, a GPU-accelerated deep learning framework
  - YouTube link: https://www.youtube.com/watch?v=aLFJ5ERxt2c
- **DeepPropLog**
  - A fully expressive DPPL where it uses probabilistic logic with well defined semantics
  - YouTube link: https://www.youtube.com/watch?v=b-AC428qhdQ

References

1. A. Skryagin, W. Stammer, D. Ochs, D. Singh Dhami, and K. Kersting, “Neural-Probabilistic Answer Set Programming,” Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning (KR), 2022.
2. A. Vergari, Y. Choi, R. Peharz, G. Van den Broeck, “Probabilistic Circuits: Representations, Inference, Learning, Applications,” AAAI 2020, 2020.
3. G. Baudart, M. Hirzel, and L. Mandell, “Deep Probabilistic Programming Languages: A Qualitative Study,” arXiv, 2018.
4. J. Lee and Z. Yang, “NeurASP = Neural Network + ASP,” Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), 2022.
5. London Machine Learning Meetup, “Robin Manhaeve - DeepProbLog: Neural Probabilistic Logic Programming,” Dec. 2019. [Video]. YouTube. https://www.youtube.com/watch?v=b-AC428qhdQ
6. Uber Engineering, “[Uber Open Summit 2018] Pyro: Deep Probabilistic Programming,” Nov. 2018. [Video]. YouTube. https://www.youtube.com/watch?v=aLFJ5ERxt2c


Ira A. Fulton Schools of
Engineering
Arizona State University