Sum-Product Logic: Integrating Probabilistic Circuits into DeepProbLog

Arseny Skryagin et. Al.
AI&ML Lab, TU Darmstadt, Germany

Abstract: We introduce **Sum-Product Logic** (**SPLog**), a deep probabilistic logic programming language (**DPPL**) that **incorporates learning through predicates encoded as probabilistic circuits**, specifically sum-product networks. Our **empirical illustrations demonstrate the benefits of supporting symbolic and deep representations**, both neural and probabilistic circuit ones for inference and (deep) learning from examples.

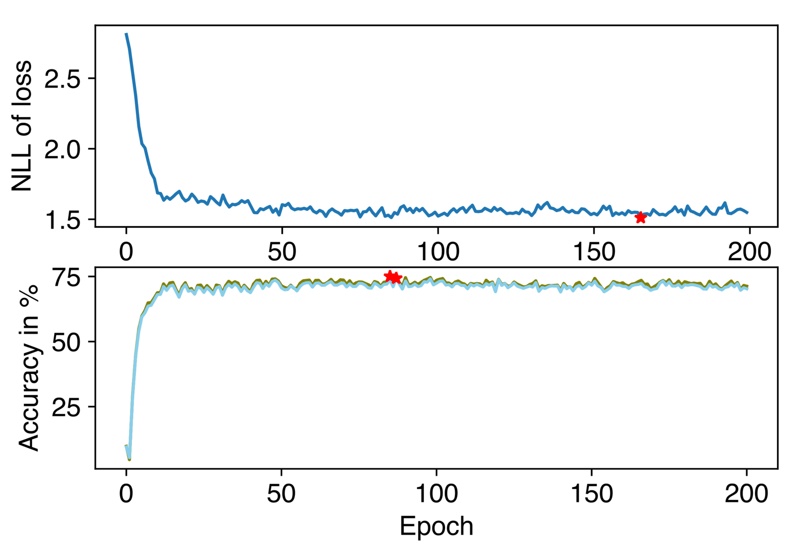
METHOD

1. An SPLog program is a ProbLog program that is extended with a set of ground sum-product annotated disjunctions (spADs) of the form $spn(m_a, \vec{Q}, \vec{E}) :: a(\vec{e}, \vec{q}_1); \dots ; a(\vec{e}, \vec{q}_1) : -b_1, \dots, b_n$
2. We use the learning from entailment (i.e. learning from queries). Given an SPLog program with parameters X and a set D of pairs (q, p) where q is a query and p its desired success probability, we compute the loss L:

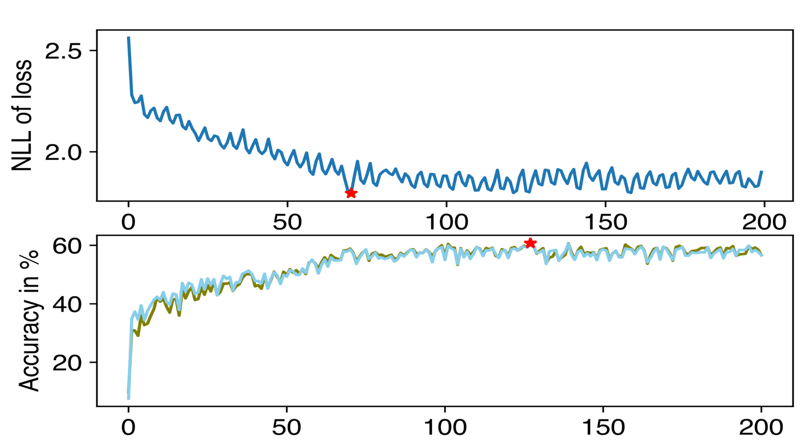
$$\arg\min_{\vec{x}} \frac{1}{|D|} \sum_{(q,p)\in D} L(P_{X=\vec{x}}(q), p)$$

RESULTS

I. Joint training with the variational AE



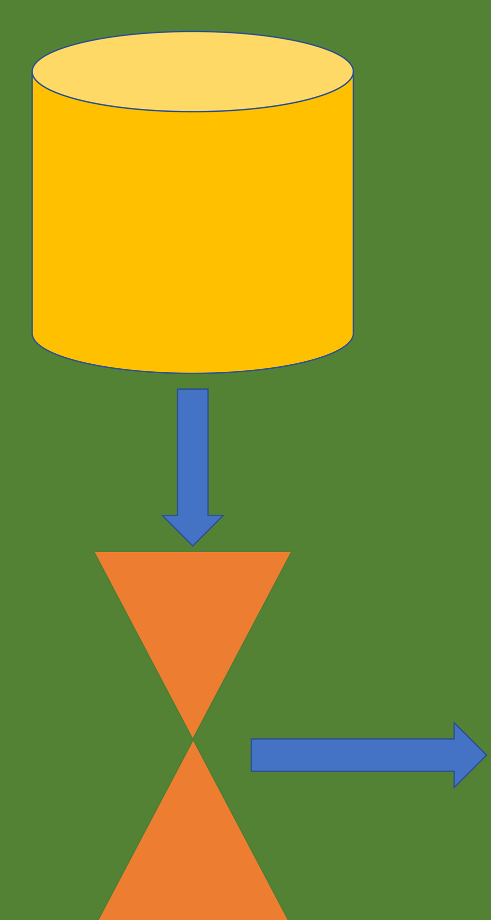
II. Training with incomplete data



The **integration** of **Sum-Product Networks into DeepProbLog** paved the way for **System 2 approaches** and so allowed to perform **inference**, **marginalization**, and **sampling in linear time**, as well as **training with incomplete data**.

I. Joint training

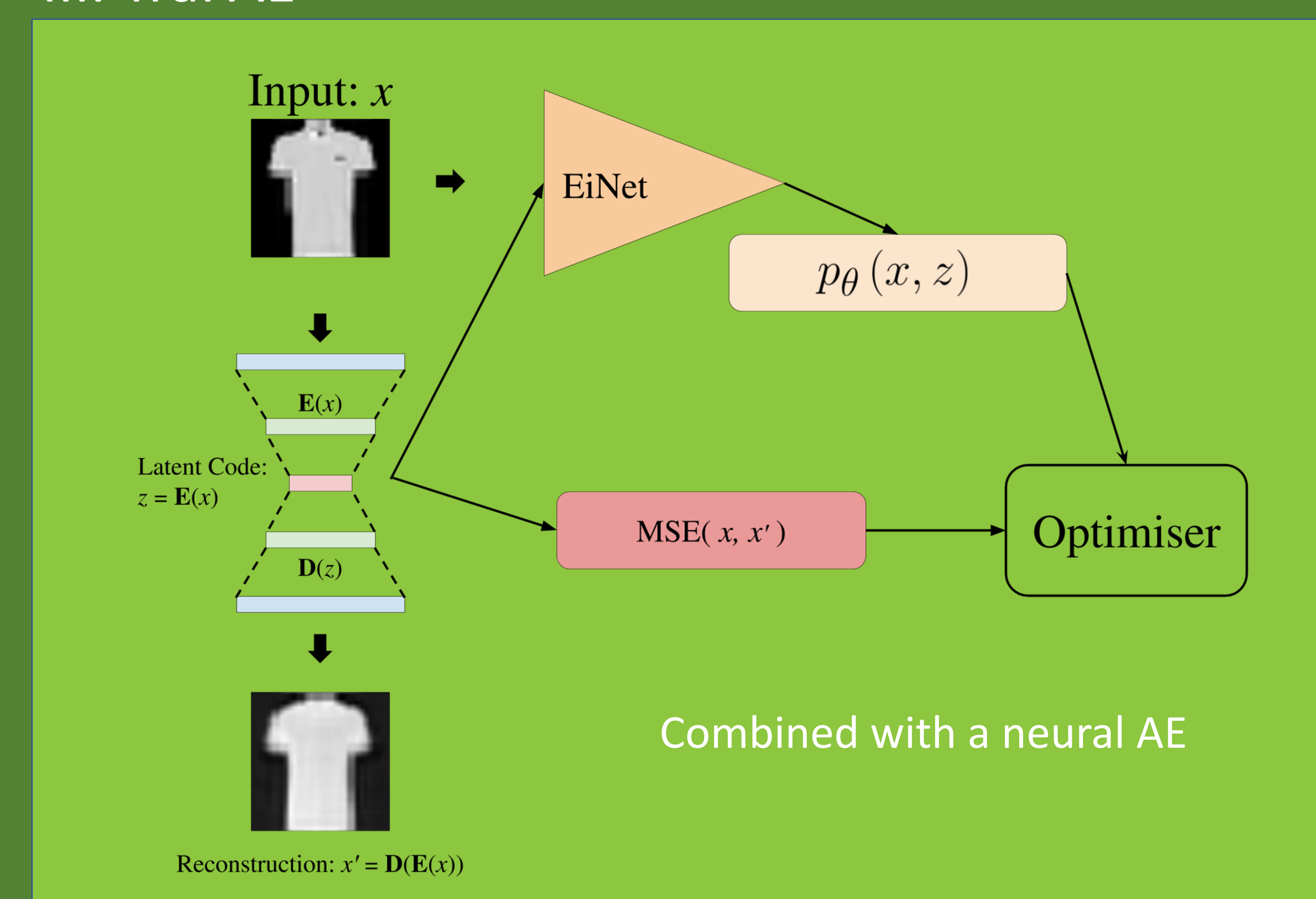
Data set



VAE

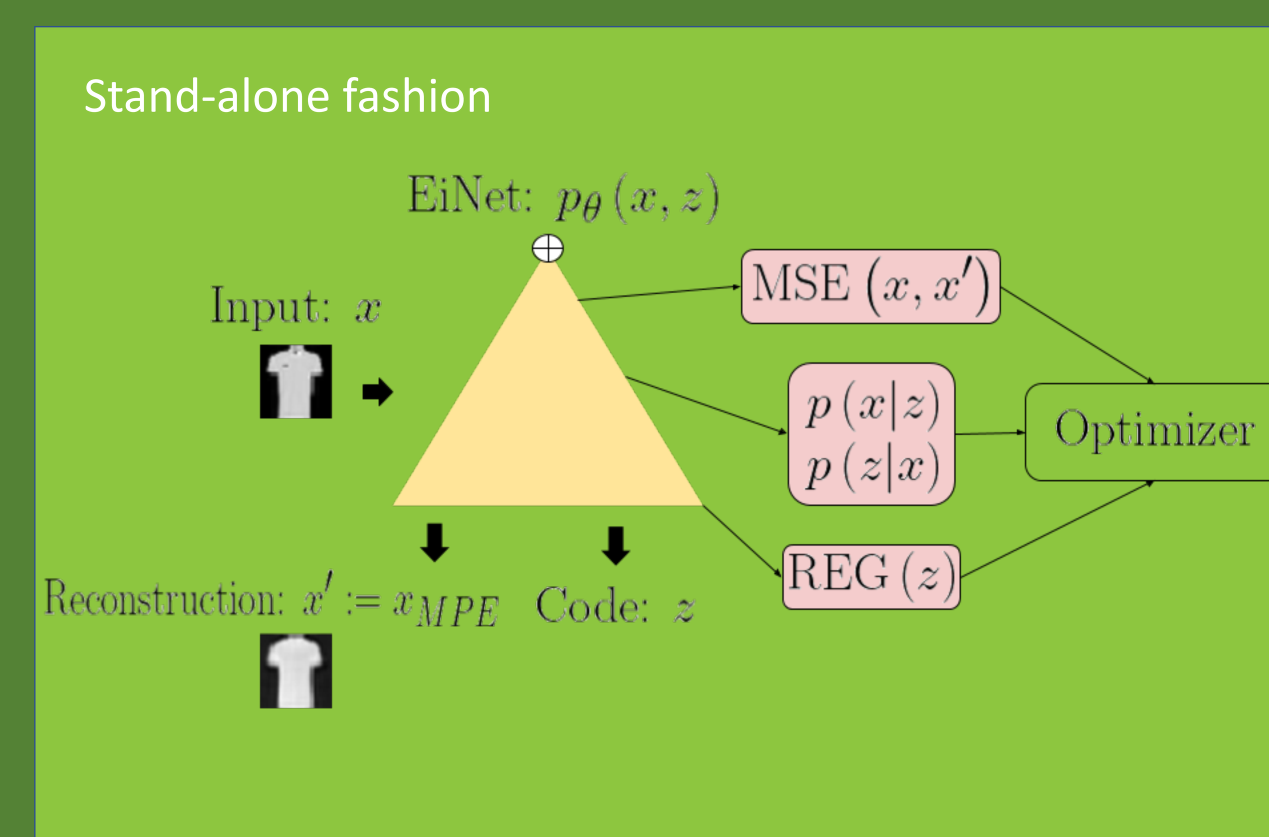
$z_1 + z_4 = 5$

III. TraPAE



II. Training with incomplete data

+ = 5



Learn from queries and not from data!

https://www.aiml.informatik.tu-darmstadt.de

Code at:

III. Tractable Probabilistic Auto-Encoder

Combined with a neural autoencoder the total loss function is defined as:
$\mathcal{L} = \omega_1 \cdot \mathcal{L}_{\text{REC}} + \omega_2 \cdot \mathcal{L}_{\text{nLL}}$

Specifically, with $z = \boldsymbol{E}(x)$ and $x' = \boldsymbol{D}(\boldsymbol{E}(x))$, each loss with a size- B batch is defined as:

$\mathcal{L}_{\text{REC}} = MSE(x, x') = \frac{1}{B}\sum_b \sum_n (x_{bn} - x'_{bn})^2$
-- Reconstruction Error. x_{bn} x_{bn} represents the n^{th} feature of the b^{th} data.

$\mathcal{L}_{\text{nLL}} = -\frac{1}{B}\sum_b \log p_\theta(x_b, z_b)$
-- Negative Log-likelihood

1 2 3 4 5 6 7 8

Results from 10 classes in the dataset
Column 1: original input x;
Column 2: direct reconstruction by PAE, x'_{PAE};
Column 3: reconstruction from MPE of code, x'_{PAE};
Column 4 to 8: reconstruction from conditional sampled code. x'_{sample}

The total loss function in case of the stand-alone fashion is defined as:

$$\mathcal{L} = \omega_1 \cdot \mathcal{L}_{\text{REC}} + \omega_2 \cdot p(z|x) + \omega_3 \cdot p(x|z) + \omega_4 \cdot \mathcal{L}_{\text{REG}}$$

Specifically, with $z = z_{Cond}$ and $x' = x_{MPE}$, each loss with a size- B batch is defined as:

$$\mathcal{L}_{\text{REC}} = MSE(x, x') = \frac{1}{B}\sum_b \sum_n (x_{bn} - x'_{bn})^2$$

-- Reconstruction Error. x_{bn} x_{bn} represents the n^{th} feature of the b^{th} data.

$$p(z|x) = -\frac{1}{B}\sum_b \log p_\theta(z_b|x_b)$$

$$p(x|z) = -\frac{1}{B}\sum_b \log p_\theta(x_b|z_b)$$

-- negative Log-likelihoods

$$\mathcal{L}_{\text{REG}} = L_2 = \frac{1}{2} \cdot \|z^2\|$$

-- Regularization Term. E.g. the Euclidian norm of the latent code.

TECHNISCHE UNIVERSITÄT DARMSTADT