ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200381

Logical Interpretations of Autoencoders

Anton Fuxjaeger[1] and **Vaishak Belle**[2]

Abstract. The unification of low-level perception and high-level reasoning is a long-standing problem in artificial intelligence, which has the potential to not only bring the areas of logic and learning closer together but also demonstrate how abstract concepts might emerge from sensory data. Precisely because deep learning methods dominate perception-based learning, including vision, speech, and linguistic grammar, there is fast-growing literature on how to integrate symbolic reasoning and deep learning. Broadly, efforts seem to fall into three camps: those focused on defining a logic whose formulas capture deep learning, ones that integrate symbolic constraints in deep learning, and others that allow neural computations and symbolic reasoning to co-exist separately, to enjoy the strengths of both worlds. In this paper, we identify another dimension to this inquiry: what do the hidden layers really capture, and how can we reason about that logically? In particular, we consider variational autoencoders that are widely used for dimensionality reduction and inject a symbolic generative framework onto the feature layer. This allows us, among other things, to generate example images for a class to get a sense of what was learned. Moreover, the modular structure of the proposed model makes it possible to learn relations over multiple images at a time, as well as handle noisy labels. Our empirical evaluations show the promise of this inquiry.

1 INTRODUCTION

The unification of low-level perception and high-level reasoning is a long-standing problem in artificial intelligence, which has the potential to not only bring the areas of logic and learning closer together but also demonstrate how abstract concepts might emerge from sensory data. Precisely because deep learning methods dominate perception-based learning, including vision, speech, and linguistic grammar, there is fast-growing literature on how to integrate symbolic reasoning and deep learning. Efforts have ranged from providing a truth-theory to deep learning [31, 9], neural architectures that enable differential computation for symbolic constraints [36, 2, 29, 30], and embeddings for graph and relational data [37, 21, 24, 7]. Approaches such as DeepProbLog [23], on the other hand, treat deep learning as an external computation and integrate its predictions as an external predicate in a probabilistic logic programming framework. Broadly, efforts seem to fall into three camps: those focused on *semantic characterizations* (i.e., define a logic whose formulas capture deep learning), *constrained learning* (i.e., integrate symbolic constraints in deep learning), and *hybrid methods* (allow neural computations and symbolic reasoning to co-exist separately, to enjoy the strengths of both worlds).

In this paper, we identify another dimension to this inquiry: *what do the hidden layers really capture, and how can we reason about that logically?* In particular, we consider autoencoders (AEs) [12, 15, 27]. As a variant of neural networks, AE frameworks are perhaps the most popular for dimensionality reduction, but its inner workings are entirely opaque and mysterious. Basically, given an encoder *e*, one first applies it to input data x to obtain a feature layer (*FL*) and then attempts to recover x from *FL* using a decoder *d*. Constraints on *FL* can lead to massive reductions on the dimensionality and identify salient features for applications such as anomaly detection. (See, for example, [7] that is a purely logical approach inspired by autoencoding principles.) Thus, we ask the question: *can we inject a logical language onto the FL to perform Boolean reasoning over the FL's variables?*

The exact choice of the language would depend on what we intend to do with the logic. A purely discrete representation such as propositional logic may not be very interesting or insightful about what the *FL* really captures, especially in cases where there may be probabilities assigned to image labels. In that regard, there has been an interesting development in knowledge representation over the last few years. As a special case of probabilistic logical models [6, 11] tractable probabilistic models have emerged as an extension to data structures such as binary decision diagrams (BDDs). In particular probabilistic sentential decision diagram (PSDDs) [16], for example, are a complete and canonical representation of a probabilistic distribution defined over the models of a propositional theory. By imposing certain properties on the propositional representation, such as decomposability and determinism, probabilistic queries can be answered in polynomial time in the size of the data structure by way of model counting. Its parameters can be learned efficiently from data, which allows us to view the representation through a generative lens over a logical base. We discuss below how these features are put to use, but more generally, we see the work as a step in repurposing deep learning in logical space to contributing to the emergence of high-level reasoning from a low-level system. Our contributions are orthogonal in many regards to the existing literature on neuro-symbolic systems and thus we imagine there would be space for looking at other kinds of integration with the existing literature.

Interestingly, we take note of multiple approaches for visually inspecting and interpreting NNs in the literature [33, 38], with a special focus on understanding convolutions of deep networks after training [32, 39]. While many of these methods yield various analysis of what happens in a given NN, including saliency maps [32], they differ in thrust significantly from our contributions. For example, optimization methods are usually used to infer and decode regions of interest in a specific layer of a pre-trained network. In contrast to this, our logical approach uses a symbolic framework to make sense of the NN over a generative model. As such we do not infer the meaning of individual variables, although it is possible to visualize them, but

[1] University of Edinburgh, UK, email: anton.fuxjaeger@ed.ac.uk
[2] University of Edinburgh & Alan Turing Institute, UK, email: vaishak@ed.ac.uk

rather compute conditional probabilities over those variables.

It should also be noted that recent circuit models attempt to tackle vision problems too (e.g., [26, 10, 20]), and so it is possible to realize the entire image classification pipeline using these tractable probabilistic models (however usually as a discriminative model). We think this is a very exciting development. What our work attempts to do, however, is to inspect state-of-the-art deep learning architectures (especially models like AEs that are very powerful for dimensionality reduction) via such symbolic generative models, in case such architectures are already in place, or are tackling problems still to be addressed using a pure circuit scheme. In that regard, as mentioned, our work is to be seen as attempting to re-purpose the latent space in a logical manner.

Our approach offers the following capabilities. We learn a PSDD over a discretized *FL*, which yields a joint distribution over the individual variables, including image labels, of the *FL*. This allows us, among other things, to visualize these individual variables by conditional sampling. In particular, this enables us to generate example images for a class to get a sense of what was learned. Moreover, the modular structure of the proposed model makes it possible to learn relations over multiple images at a time. Finally, because of the logical structure that we impose, noisy labels can also be handled. We also discuss how we can evaluate the learned representation over well-known datasets, and also discuss both reconstructability (i.e., generative capabilities) and classification accuracy.

At the outset, from an engineering (as opposed to mathematical) viewpoint, it should be noted that, at this point, since circuit software packages have not enjoyed the same amount of maturity as deep learning packages, the reported accuracy is not as competitive as state-of-the-art systems. Nonetheless, although this reported accuracy is lower, the model has considerably more functionality, including the ability to sample prototypical images for each of the learned classes, ultimately aiding us in understanding what has been learned by the model (i.e., visually showing us what the model thinks a given class represents).

2 PRELIMINARIES

2.1 Probabilistic Sentential Decision Diagrams

Sentential decision diagrams (SDDs) were first introduced in [5] and are tractable representations of propositional knowledge bases. SDDs are shown to be a strict subset of deterministic decomposable negation normal form (d-DNNF), a popular representation for probabilistic reasoning applications [3] due to their desirable properties. Decomposability and determinism especially ensure tractable probabilistic inference. PSDDs extend SDDs with probabilities, and are a complete and canonical representation of joint probability distributions [16].

Intuitively, PSDDs are parametrized directed acyclic graphs (DAGs), as seen in Figure 1. Here, each terminal node represents a univariate (Bernoulli) distribution over a binary variable (e.g. B_j) with a probability θ represented by the tuple $(\theta_j : B_j)$. Within the tree, each node is either an AND or an OR node. An AND node has two inputs termed prime p for the left one and sub s for the right one. The OR node can have an arbitrary number of inputs, where each of the n input wires is annotated by a probability $\theta_1, ..., \theta_n$ together making up a normalised distribution over the variables represented by the corresponding *vtree*. Moreover, OR and AND gates always alternate, such that a given OR node can also be represented as a set of AND nodes or decisions: $\{(p_1, s_1, \theta_1), ..., (p_n, s_n, \theta_n)\}$.

In order to retain the desirable properties of SDDs for inference (tractability) and canonicity, similar syntactic restrictions hold here as well. Firstly, each of the AND gates has to be *decomposable*, meaning that the *vtree* nodes represented by prime and sub share no variables. In other words, the prime and sub have to represent probability distributions over disjoint sets of variables. Analogously, *determinism* demands that for each possible world (or assignment), there can be at most one prime that assigns a non-zero probability to it (the specific world).

In [19, 1], a learning regime is proposed that is capable of learning a PSDD, called learnPSDD, as well as the underlying SDD and *vtree* [25] directly from data in an unsupervised manner. It works by iteratively updating and improving the structure of the PSDD to better fit the data. It does so by applying specified *clone* and *split* operations to a PSDD r at each step. Learning is then carried out until a time limit is reached or a pre-defined *score* converges on the validation data (if present, otherwise training data). This *score* is based on the log-likelihood of the model given the data, but takes the size of the tree into account as well. The log-likelihood of PSDD r given data $\mathcal{D}$ is then a sum of log-likelihood contributions per node:

$$ln\mathcal{L}(r|\mathcal{D}) = lnPr_r(\mathcal{D}) = \sum_{q\in r}\sum_{i\in q} ln\theta_{q,i}\mathcal{D}\#(\gamma_q, [p_{q,i}]) \quad (1)$$

where $\#(\gamma_q, [p_{q,i}])$ is the number of examples that satisfy the node context of q and the base of a prime q, that is, $p_{q,i}$. Additionally, [19] also proposed an algorithm for learning ensembles of PSDDs (EM-LearnPSDD) which is built on the learnPSDD algorithm and the soft structural EM algorithm from [8]. This algorithm consists of two nested learners, where the outer EM is learning the structure and the inner EM is learning the parameters. This is the algorithm we predominantly use in our experiments.

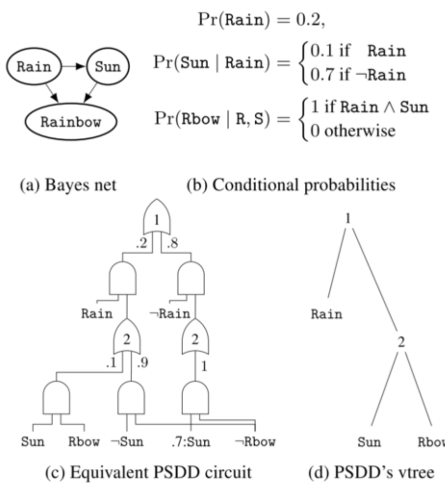

Pr(Rain) = 0.2,
Pr(Sun | Rain) = 0.1 if Rain; 0.7 if ¬Rain
Pr(Rbow | R, S) = 1 if Rain ∧ Sun; 0 otherwise
Rain
Sun
Rainbow
(a) Bayes net
(b) Conditional probabilities
1
.2
.8
Rain
¬Rain
2
.1
.9
1
Sun
Rbow
¬Sun
.7:Sun
¬Rbow
(c) Equivalent PSDD circuit
(d) PSDD's vtree

Figure 1: A Bayesian network and its equivalent PSDD [19].

2.2 Neural Networks & Autoencoders

An AE is a specific instance of an artificial neural network (NN) that is intended to reproduce the given input as an output [12]. It consists of two parts: the encoder *e* and decoder *d*. Internally, it has a hidden layer referred to as the *feature layer* (*FL*) such that $FL = e(x)$ for some input data x and $x_{rec} = d(FL)$ where x_{rec} is the reconstructed input, $FL \in \mathbb{R}^{dim}$, and *dim* is the dimensionality of the *FL*.

Restrictions are usually imposed on the structure of the network such as reduced dimentionality in the *FL* or added noise on the input. By reducing the dimension of the *FL* relative to the dimension of the input x, the network is intended to learn the *most valuable* features for reconstructing the original image. The *learning* procedure then constitutes learning the encoder and decoder function simultaneously by minimising the reconstruction loss (e.g., mean squared error) penalising $d(e(x))$ for being dissimilar to x.

The **variational autoencoder** (VAE) [15, 27] is a variant of AE that builds on the stochastic generalisation of the classical AE architecture, where instead of a deterministic function, *e* and *d* are stochastic mappings $p_e(FL|x)$ and $p_d(x|FL)$. Thus, we can also view *e* and *d* as conditional probability distributions. Utilising this probabilistic interpretation, the VAE framework defines a distribution $q(FL|x)$ (e.g the Gaussian distribution) such that *FL* samples can be drawn from that distribution. Then we can use the Kullback-Leibler divergence (D_{KL}) to enforce the encoder network to be as similar as possible to our chosen distribution $q(FL|x)$ while at the same time maximizing the $\log p(x)$ prior which is achieved by updating the weights based on the gradient of [12]:

$$\log p(x|FL) - D_{KL}(q(FL|x) \| p(FL)) \tag{2}$$

The *reparameterization trick* [15] then allows us to enable stochastic gradient descent by reformulating the task using stochastic input layers.

Finally, we leverage the Gumbel-Max trick [13, 22] which yields the *Gumbel-Softmax Distribution* that is defined as a continuous distribution over the simplex that can approximate samples from a categorical distribution [14].

3 METHODOLOGY & EVALUATION METRICS

In this section, we propose a novel model for representing and learning a symbolic generative model from a neural network that is trained over unstructured data $\mathcal{D}$. This is possible by means of the intermediate *FL*, defined over n discrete variables. The model is to be considered as generative but over a set of *domains*, meaning that we can perform conditional sampling for a given domain with respect to other domains. Domains, written as $D^A, D^B, D^C, \ldots$, represent disjoint subsets of $\mathcal{D}$. For example, suppose we are given an image dataset such as MNIST. Here, the images are denoted by domain D^A (say, $D^A \subset \mathbb{R}^{28\times 28}_{[0,1,\ldots,255]}$) and the corresponding labels by domain D^Y (say, $D^Y \subset \mathbb{N}$) such that $D^A \sqcup D^Y = \mathcal{D}$. The model is then able to approximate the distributions $p(D^A, D^Y)$ and moreover, can sample from $p(D^A \mid D^Y)$ and $p(D^Y \mid D^A)$.

3.1 Architecture

We now discuss the formal and architectural components of our model.

Definition 1: (Feature layer) The *FL* is a finite set of discrete (typically, Boolean) variables (represented as C_i with instantiation $c_i \in \Delta^{C_i} \subset \mathbb{N}$); that is, $FL = \{C_0, C_1, ..., C_l\}$ for some $l \geq 0$. Furthermore, $FL_i = C_i$ represents the i^{th} variable of the *FL*.

Intuitively, *FL* represents an encoded discretized version of the original data $\mathcal{D}$. If the data is split into different domains (disjoint sets, as explained above) $\mathcal{D} = D^A \sqcup \ldots \sqcup D^Y$, then the *FL* can also be split into disjoint domains: $FL = FL^A \sqcup \ldots \sqcup FL^Y$. The size of *FL* is then the number of variables $|FL| = l$ and note that, $|FL| = |FL^A| + \ldots + |FL^Y|$. Further, for a given domain FL^X, we use $dom(FL^X) > 0$ to denote the number of possible values that the discrete variables can take.

The reason we define the *FL* in such a general manner with discrete variables although PSDDs only handle binary variables, is to keep the formulation separate from the implementation. This will allows us to appeal to different high-level models (e.g., SPNs, probabilistic relational models) in the future with the same formulation.

3.1.1 *Encoders and Decoder Specification*

For a given domain $D^X \in \{D^A, D^B, D^C, ..\}$, we have an encoder e^X and a decoder d^X such that $FL^X = e^X(D^X)$ and $d^X(FL^X) = d^X(e^X(D^X)) \subset D^X$. This is like the usual AE setup, where we map from the input domain (e.g. D^X) to an intermediate discrete representation (e.g. FL^X) using the encoder and then map back to the original domain using the decoder. This formulation also works for stochastic encoder/decoder networks with $p_e(FL^X \mid D^X)$ and $p_d(D^X \mid FL^X)$ utilising the Gumbel-Softmax distribution.

Essentially, encoders and decoders are functions tasked with mapping the inputs data to the discrete *FL* representation with respect to the domain. While some encoder-decoder pairs may be learned from data, others can be defined deterministically or are simply the identity mapping. Revisiting the MNIST dataset, for example, here we may define domain D^A to be the images (e.g. $a \in D^A \subset \mathbb{R}^{28 \text{ x } 28}_{[0,1,..,255]}$) and domain D^Y to represent the corresponding labels (e.g. $y \in D^Y = \{0, 1, .., 9\}$). In particular, the encoder and decoder for domain D^A (e^A, d^A) are deep convolutional neural networks mapping a given image to a discrete *FL*. As for domain D^Y, the encoder/decoder is simply the identity mapping.

The learning of these functions (if applicable) is done in an unsupervised manner using VAEs in our setup and is referred to as *learning phase I* throughout this article. A visualization of the pipeline can be seen in Figure 2.

3.1.2 *Optional binarization of FL*

For the sake of generality, we defined the *FL* as a set or vector of discrete variables. However PSDDs are essentially Boolean and so we need to provide a binary encoding for the data. This has been explored in two possible ways: using a one-hot encoding or a mapping to binary. Considering, for example, $fl_i = c_i \in \{0, 1, 2, 3\}$, a discrete variable ($C_i$) that can take one of four values. Then $c_i = 1 \mapsto_{one\_hot} [0100]$, whereas $c_i = 1 \mapsto_{binary\_code} [01]$. Analogously $c_i = 3 \mapsto_{one\_hot} [0001]$, whereas $c_i = 1 \mapsto_{binary\_code} [11]$. In our work we mostly explored the binary encoding which is consistent with the underlying language being propositional. Since this is binary conversion is essential for the technical treatment, we will not refer to this conversion in the sequel.

3.1.3 *The Logical Interpretation*

The (logical) generative model represents the dependencies between the individual variables of the *FL*; in other words, it represents the joint probability distribution over the variables of the *FL*, i.e., $Pr(FL)$. In this paper, we chose to use PSDDs [16] though other models such as sum product networks (SPNs) [26] could have been used as well. This choice was based on the reported ability of PSDDs to handle constraints in the learning regime. These could be one-of label constraints or any other kind of Boolean function over the inputs.

3.1.4 *Learning*

The learning of the system is done in two phases (see Figure 2). Learning phase I denotes the learning of the encoder-decoder function pairs for each domain (independently and unsupervised). Once learning phase I is completed we can use the encoders to map the data to the *FL* representation and learn the logical generative model (learning phase II), that is, the PSDD. This is done in an unsupervised manner by iteratively approximating the joint probability distribution over the data. Essentially, the variables of the *FL* are the propositions of the PSDD.

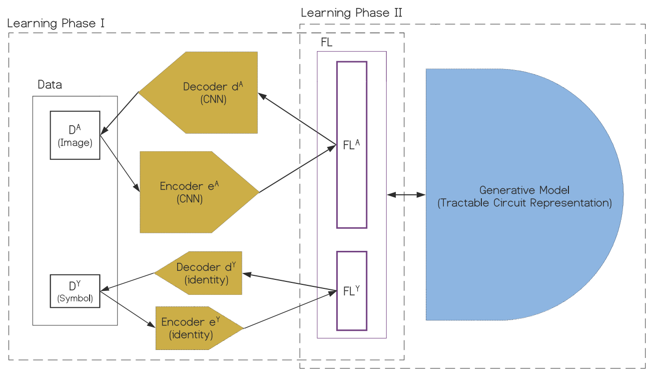


Figure 2: The two phases of the proposed learning system w.r.t. the MNIST example

3.2 Querying

Due to the generative property of PSDDs, we are able to perform any query of the form: $Pr(q|v) = \frac{Pr(q \wedge v)}{Pr(v)}$ where q is the query and v is the evidence, both Boolean functions over the variables of the *FL* (e.g. $q = \{c_{32}\}$, $v = \{c_0, c_1, .., c_{31}\}$). Specifically, in the MNIST example, such queries would take the form: $Pr(e^X(q) \mid e^Y(v))$ where X and Y correspond to two domains of the data ($D^X, D^Y \subset \mathcal{D}, q \in D^X \subset \mathbb{R}^{28 \times 28}_{[0,1,..,255]}$ and $v \in D^Y = \{0, 1, .., 9\}$). Furthermore, by Theorem 7 of [16] such probabilities can be computed in one pass through the tree and thus in polynomial time w.r.t. the size of the graph. (Thus, these are referred to as tractable models in the literature.)

3.2.1 *Generative Query*

Given evidence v, we define the task of a *generative query* as one that samples values for all variables in the *FL* which are not assigned in the evidence. That is, $fl = generativeQuery(\Gamma, v)$, which is discussed in Algorithm 1 and is equivalent to $fl \sim Pr(FL \mid v)$. Here Γ represents the high-level model, the learned PSDD in out case.

As mentioned before the resulting assignments of variables returned from the algorithm can then be decoded using the decoder *d*.

3.3 Evaluation

In order to evaluate our model on image datasets, we focus on two main aspects. First, classification accuracy and secondly, the *recoverability* of the trained model in terms of how it *interprets* a given domain, explained below.

Algorithm 1 generativeQuery(PSDD Γ, evidence v, categorical dimension k)

```
 1: $assigned \leftarrow$ variables_appearing_in($v$)
 2: $not\_assigned \leftarrow$ variables_appearing_in($\Gamma$) $- \; assigned$
 3: $generated \leftarrow dict()$
 4: **while** not empty(not_assigned) **do**
 5:     $var \leftarrow pop\_random(not\_assigned)$
 6:     $dist \leftarrow zeros(dim = k)$
 7:     **for** $j \in range(k)$ **do**
 8:         $dist_j \leftarrow \frac{Pr_\Gamma(var = onehot(j)|v)}{Pr_\Gamma(v)}$
 9:     **end for**
        {Sample from the categorical distribution}
10:     $inst \leftarrow sample(dist)$
        {Add sampled assignment to $v$}
11:     $v \leftarrow (v \wedge (var = inst))$
        {Add sampled assignment to output dictionary}
12:     $generated[var] \leftarrow inst$
13: **end while**
14: **return** $generated$
```

Classification accuracy is used as a quantifiable score that is easily comparable to other learning systems. We train the model on multiple image datasets before asking the model to classify unseen images ($a \in D^A$) into one of the possible categories ($y \in D^Y$) using a maximum likelihood formulation. That is, we compute the label of an image by:

$$y = \underset{y_i \in \Delta^Y}{\arg\max}\, Pr(e^Y(y_i) \mid e^A(a))$$

However we could also obtain the label by sampling (here $'$ denotes a sample drawn from a distribution):

$$y' = d^Y(gernerativeQuery(\Gamma, e^A(a))) \tag{3}$$

$$e^Y(y') \sim Pr(FL^Y \mid e^A(a)) \tag{4}$$

$$y' = d^Y(fl'^Y \sim Pr(FL^Y \mid [fl'^A \sim p_e(FL^A \mid a)])) \tag{5}$$

We investigate the interpretability of the model by manually inspecting samples drawn from the distribution for some evidence. For the MNIST dataset, for example, we sample images for each category or class and check if the images correspond to the class. Such samples will be computed as follows (where $a \in D^A$ denotes the images as before, and $y \in D^Y$ are the class labels):

$$a' = d^A(generativeQuery(\Gamma, e^Y(y)) \tag{6}$$

$$e^A(a') \sim Pr(FL^A \mid e^Y(y)) \tag{7}$$

$$a' \sim p_d(D^A \mid [fl'^a \sim Pr(FL^A \mid e^Y(y))]) \tag{8}$$

Finally, we analyse the variables of the *FL*. This sheds some light on the inner workings of the model, and gives us an insight into what the individual variables capture. Basically, we approximate the expectation of decoded *FL* samples where, in a binary setting, samples would be drawn conditional on a specific variable being true or false:

$$diff_{FL_i} = \mathbb{E}_{fl' \sim p(FL|fl_i)} d^A(fl') - \mathbb{E}_{fl' \sim p(FL|\neg fl_i)} d^A(fl') \tag{9}$$

This is approximated by N samples:

$$\frac{1}{N} * \sum_{fl' \sim p(FL|fl_i)}^{N} d^A(fl') * p(fl') - \frac{1}{N} * \sum_{fl' \sim p(FL|\neg fl_i)} d^A(fl') * p(fl')$$

Here we define the decoded image to be w pixels in width and h pixels in height. Then each greyscale image a is normalized to be an element of $a \in [0,1]^{w*h}$. What follows is that $\mathit{diff}_{FL_i} \in [-1,1]^{w*h}$ and as such it has to be normalized accordingly in order to produce an image:

$$\mathit{visual}_{FL_i} = \left[\frac{\mathit{diff}_{FL_i}+1}{2}, \frac{-\mathit{diff}_{FL_i}+1}{2}\right] \tag{10}$$

Here, visual_{FL_i} is depicted as a tuple of images corresponding to the variable i being true vs. false and vice versa.

4 EXPERIMENTS

In this section, we investigate the predictive accuracy on unseen data (via a held out test set) as well as the generative power of the model. Firstly, we consider the standard classification task. Secondly, we run experiments where noise is added to the label of each entry; that is, each training entry has k additional random labels specified (in addition to the correct one). Thirdly, we explore tasks which consist of at least two images and possibly a symbolic value. In one such experiment for example, a data point is defined over two images, representing successive integers and we are then interested in generating one image given the other (e.g., generate *7* if the first image is *6*.). These are referred to as *functional tasks*. We conclude with an analysis of the *FL*.

4.0.1 Data

In order to get a comprehensive understanding of the capabilities of the proposed model, we used three different datasets. First, the MNIST dataset [18] containing 10^5 (grayscale) images of dimension 28x28 that represent handwritten digits, along with the corresponding class label. After the first set of experiments on the MNIST dataset, we used the hyper-parameters for the best performing models and re-run the experiments on the FASHION dataset [35], which contains 10^4 (grayscale) images of fashion items of dimension 28x28 and the corresponding labels belong to one of the 10 categories. Finally, to investigate the scalability of the model, we used the EMNIST (extended-MNIST) [4] dataset, where the images are handwritten numbers and letters of the English alphabet with the corresponding labels (47 classes in total).

4.0.2 Hardware

Since most experiments involved two training phases using very different optimization methods, we made use of two different cluster architectures in order to improve performance. Learning phase I is concerned with learning the parameters of a deep neural network using mini-batch gradient descent, and backpropagation were run on GPU clusters. The cluster nodes used here are a combination of Dell PowerEdge R730 and Dell PowerEdge T630. Each has two 16 core Xeon CPUs, where the GPUs use NVIDIA cards Tesla K40m, GeForce GTX Titan X and GeForce Titan X. Learning phase II, on the other hand, uses the learnPSDD structure learning algorithm, and this was run on CPU clusters, where each of the 21 nodes is a Dell PowerEdge R815 with four 16 core Opteron CPUs and 256GB of memory.

4.0.3 Learning and Stopping Criteria

Learning phase I, which is tasked with learning deep convolutional neural networks representing the encoder and decoder mapping was in all cases run for 400 epochs. For learning phase II, where we learn the structure and parameters of the PSDD, the learning was stopped after 72 hours or if the score converged below a given threshold.

4.1 Classification Task

Given training examples consisting of images and their labels, we trained the encoder-decoder pair unsupervised in the first instance, and the PSDD on the whole *FL* (FL^A + FL^Y representing the image and label respectively) in the second instance. The hyper-parameters explored here include the *vtree*-search algorithm used, as well as the option of compressing the label to a one_hot or binary_code representation (see Section 3.1.2). Furthermore, we varied the number of variables in FL^A and the categorical dimension of such variables (denoted by $|FL^A|$ and $dom(FL^A)$ respectively). Note that the categorical dimension essentially corresponds to whether we interpret the features in a Boolean space vs finite multi-valued space.

4.1.1 MNIST

The best classification accuracy on the MNIST dataset was measured at: 89.55% using 32 binary variables (and a categorical dimension of 2) and a one-hot encoded FL^Y. In comparison, we note that discriminative models such as convolutional NN achieve 99.3% [18] on the same task. A more comprehensive overview is given in Figure 3. Here we can see that there is a clear trade off between expressiveness of the *FL* and the ability for the PSDD learning to interpret this *FL*. In other words, if the *FL* is too small, then the neural model will not be able to learn a meaningful mapping, retaining valuable information in the encoding, and if the *FL* is too large, the PSDD learner struggles to find correlations between the variables.

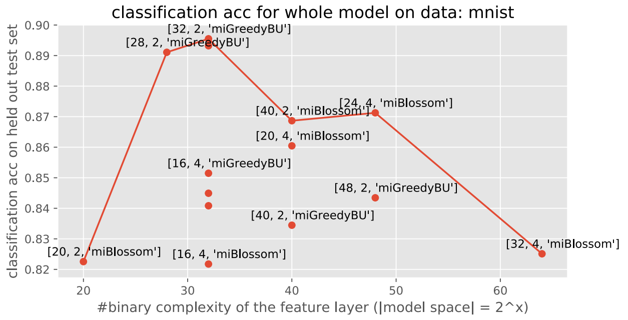
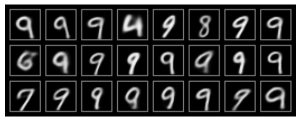
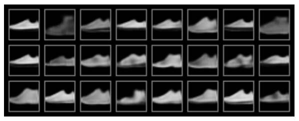


Figure 3: Experiment classification results MNIST (label format: $[|FL^A|, dom(FL^A)$, vtree seach algorithm)

From the best setup above, to test the generative abilities, we sampled images for each of the 10 categories using the proposed conditional-sampling algorithm. These samples are depicted in Figures 4a, 4b for 2 of the 10 classes. Since these are samples, we should expect to see some variation, corresponding to the figures. In a sense, the system demonstrates a prototypical understanding of what the labels represent. It is interesting to relate this insight to approaches such as [17] that involve an explicit token construction framework for generating images. We imagine that it might be possible to use the variables induced in our frameworks as a token generator, which we leave for the future.

(a) class:3 (MNIST)


(b) class:9 (MNIST)


(c) class:Sneaker (FASHION)

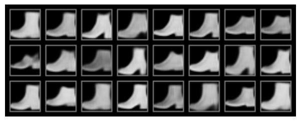

(d) class:Ankle boot (FASHION)

Figure 4: Sampled images for specified classes of models trained on classification task (MNIST and FASHION) with $|FL^A| = 32$, $dom(FL^A) = 2$

4.1.2 *FASHION*

For the FASHION dataset, we achieve a classification accuracy of 75% using the hyperparameters of the best performing MNIST model. Interestingly, the reconstruction loss (binary-cross-entropy) of the neural model is smaller (thus, better) in this scenario than in the MNSIT case, and the PSDD score is larger (thus, better) as well. The predictive accuracy on the held out test set is still considerably lower. This can be due to many reasons, most notably perhaps due to the additional complexity of the images. The additional variability of FASHION influences the computed reconstruction loss of the VAE, as it computes an average over pixel difference between the original image and the reconstructed one. For testing the generative abilities, once again, we sampled images for 2 of the 10 classes, as shown in Figures 4c and 4d.

4.1.3 *EMNIST*

Finally, when running experiments on the EMNIST dataset, we found that the system is not quite capable of scaling to such a large number of image classes, which we suspect seriously affects the performance of the PSDD learner. Additionally, the VAE is confronted with a much more complex task in differentiating symbols (e.g., “1” and “l”). Here we only recorded an overall best accuracy of 29%, where $1/47 = 0.021$ would be the expected random accuracy. It is an interesting question for the future to consider how to handle so many image classes with a Boolean learner.

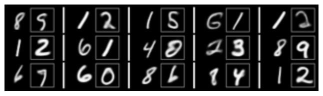
4.2 Noisy Label Task

As mentioned earlier, we are interested in challenging the system by providing k randomly generated additional labels to the correct one during training. To evaluate the experiment, we computed the accuracy on a held out set (for MNIST) only containing the right label (no noise). Generally speaking, as expected, we observed a decreasing accuracy with increasing noise: for example, adding one additional label (noisy-1) decreases accuracy by .05 to 85.0%. Even if 2 and 3 noise labels are added, the accuracy only decreases to 82.1% and 71.8% respectively. Intuitively, the experiment requires the PSDD to reason about the possible labels for a given image and thus, we show that the logical model performs this reasoning in a satisfactory manner.

4.3 Functional Tasks

The idea here is to have training examples consisting of at least two images and maybe a symbolic value that denotes the relationship between these images. However, no semantic characterisation is provided for this symbolic value in our setup, so the system tries to map the image pairs to the value purely from visual features. (Thus, the machinery of logically defining such functions, as seen in, e.g., DeepProbLog [23], could be used to extend our framework further.)

The simplest one is where we provide an image, and expect the system to generate a second image such that the integers present in the images are successors. This demonstration is depicted in Figure 5, where we observe that the predecessor/successor integer's image was generated successfully.


(a) Sampled images for FL^A

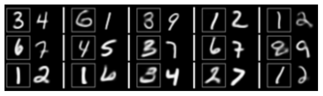
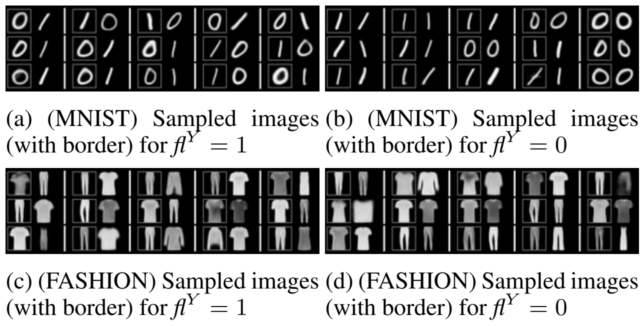

(b) Sampled images for FL^B

Figure 5: Image generation for successor task, one datapoint is a tuple, where the image with border was sampled for image with no border

In an additional set of experiments, we also provide a symbolic variable (FL^Y) that is the evaluation of a mathematical function over the two images. One example of such a function is the Boolean logic *XOR*. Here, we first train the unsupervised VAE on the whole (e.g., MNIST) data, and then create a custom training dataset where each entry contains two images, either “0” or “1”, and the result of applying the logical *XOR* operation on the label of these two images (FL^Y). Thus, the FL in this task is made of three individual parts: $FL^A = e(imgA), FL^B = e(imgB)$ representing two images and $FL^Y = bool(label(imgA))\ XOR\ bool(label(imgB))$, representing the evaluation of the *XOR* function on the original labels of the two images. We can then evaluate the accuracy on the correctness of the predicted symbolic value on a held out test set. Conversely, we can sample for one of the images given the other image and a specified FL^Y value. To reiterate, this is purely visual reasoning, so to clarify that, we can also repeat the experiment with the FASHION dataset, treating T-shirts and Trousers to correspond to *true* and *false* respectively (e.g., $bool(\text{Trouser}) = 1$). (All other digits in MNIST and all other image classes in FASHION are discarded.) The classification accuracy that we measured on a held out test set were 99.4% and 88.2% for MNIST and FASHION respectively. Generated samples are shown in Figure 6. As an example of a more complex function, we also conducted experiments on MNIST, where FL^A and FL^B range over all images present in the dataset but FL^Y constitutes the result of the arithmetic *plus* operation on the original labels of the two images, such that $FL^Y = label(imgA) + label(imgB)$. Here, we have many more possible FL^Y values and multiple combinations of images that correspond to the same FL^Y value. However, the recorded classification accuracy on the held out test set is only 9.92%. Thus, the conclusion to be drawn here is although the logical generative model does allow us to formulate challenging tasks over mathematical and logical functions, it currently only resolves this in terms of the visual features. So a second interesting direction for the future is to understand how to go beyond this and find a way to incorporate (or learn) the semantic meaning of the mathematical function. PSDDs [19], for example, can be trained with constraints which might offer a possible way to make progress in this direction.



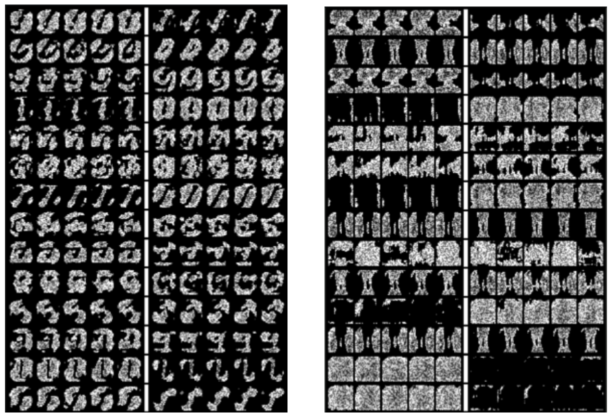
(a) (MNIST) Sampled images (with border) for $fl^Y = 1$ (b) (MNIST) Sampled images (with border) for $fl^Y = 0$

(c) (FASHION) Sampled images (with border) for $fl^Y = 1$ (d) (FASHION) Sampled images (with border) for $fl^Y = 0$

Figure 6: Image generation for binary-logic-*XOR* task, one datapoint is a tuple, where the image with border was sampled for image with no border and $fl^Y \in \{0, 1\}$

4.4 *FL* Analysis

To understand what a given variable in the *FL* actually represents, we use the generative query algorithm and Equation 10. To evaluate this more concretely, we sample images for our best performing model on MNIST and FASHION. In Figure 7, we computed $visual_{FL_i}$ five times with $N = 200$ for each variable of FL^A. In Figure 7, each row corresponds to one of the first 14 variables of the *FL* (in order from 1 to 14, top to bottom). What this demonstrates is that the model accords meaningful elements of the images to each variable of the *FL*. Indeed, we see that individual variables correspond to different shapes such as slightly bent lines in row/variable 1 of Figure 7a or circular objects in row/variable 4. In a sense, the *FL* is able to identify discrete visual components for the images, which hints at a compact and compositional understanding of the domain in question.



(a) $visual_{fl_i}$ on MNIST (b) $visual_{fl_i}$ on FASHION

Figure 7: Visualisation of the first 14 binary variables in the classification models for two datasets (variables are ordered)

5 CONCLUSION & DISCUSSION

In this work, we were interested in understanding what precisely the latent space of AEs capture, and whether that space could be inspected from a logical viewpoint. In that regard, we motivated the learning of a symbolic generative model on the *FL*, which allows us to inspect the hidden layers and perform logical reasoning over the variables of these layers. For example, by means of a conditional sampling algorithm, we were able to generate prototypical images for a label, and moreover, generate labels for images.

As mentioned previously, with regards to the standard classification task we see that our model can not compete with other state-of-the-art systems such as deep convolutional neural networks (CNNs). However, when making such comparisons, one should consider that discriminate models such as CNNs are not generative, whereas generative models (e.g. VAEs) are not appropriate for classifying images as they may not discriminate between individual images.

Although one might, in general, consider that a lower performance is a reasonable trade-off in exchange for increased functionality and interpretability, we do not think this is "fundamental" tradeoff: our observation has been that the PSDD software seems very capable of handling intricate Boolean reasoning, but it struggles somewhat when considering multi-valued discrete variables. Note that by increasing the "encoding" space, we are allowing for more granular reconstructions of the latent space, and so we should expect to reach the performance of state-of-the-art models. So there is an engineering effort required. In contrast, many conventional deep learning software packages have benefited from considerable optimizations.

In addition to classifying images, the model was put to test in challenging tasks capturing structural, logical or mathematical relationships between pairs of images, as well as the handling of noisy labels. While we did observe scalability issues when considering a very large set of classes, the underlying framework still offers an insightful logical view of the hidden layers. This provides the space for interesting avenues for the future, such as integrating our framework with existing neuro-symbolic frameworks. In particular, can one of these frameworks provide a way to reason about mathematical functions in a semantic manner (perhaps also learn them), rather than the purely visual quality exploited in the current setup? Can proposals from statistical relational learning [11] help us capture and reason about intricate logical relationships between variables? The overall goal, then, is to get a better grasp of how abstract concepts and high-level reasoning might emerge from low-level sensory data. We hope that this work, which attempts to re-purpose a deep learning framework in logical space, provides some of the insights on how that is possible, and at the same time, shows the benefits of using a symbolic generative model in a differential latent space.

It is worth remarking that PSDDs may not be the high-level representation that one considers to be human-readable. However, owing to it's probabilistic semantics, one can pose arbitrary conditional queries to analyse what has been learned. Moreover, there has been a lot of work in knowledge representation and knowledge compilation where certain high-level languages have been reduced to (arithmetic) circuits at the time of inference [34, 23, 28]. Therefore, we are very interested in considering the problem from a different angle in the future: given that we have a circuit representing the data, can we construct high-level concepts perhaps with some weak supervision? This remains to be explored, and we hope our work is providing a reasonable first step.

ACKNOWLEDGEMENTS

We would like to thank John Quinn for his valuable input on this work and many fruitful conversations. Anton Fuxjaeger was supported by the Engineering and Physical Sciences Research Council (EPSRC) Centre for Doctoral Training in Pervasive Parallelism (grant EP/L01503X/1) at the School of Informatics, University of Edinburgh. Vaishak Belle was supported by a Royal Society University Research Fellowship. We would also like to thank our reviewers for their helpful suggestions.

REFERENCES

[1] Jessa Bekker, Jesse Davis, Arthur Choi, Adnan Darwiche, and Guy Van den Broeck, 'Tractable learning for complex probability queries', in *Advances in Neural Information Processing Systems*, pp. 2242–2250, (2015).

[2] Matko Bošnjak, Tim Rocktäschel, Jason Naradowsky, and Sebastian Riedel, 'Programming with a differentiable forth interpreter', in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 547–556. JMLR. org, (2017).

[3] Mark Chavira and Adnan Darwiche, 'On probabilistic inference by weighted model counting', *Artificial Intelligence*, **172**(6-7), 772–799, (2008).

[4] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik, 'Emnist: an extension of mnist to handwritten letters', *arXiv preprint arXiv:1702.05373*, (2017).

[5] Adnan Darwiche, 'Sdd: A new canonical representation of propositional knowledge bases', in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, p. 819, (2011).

[6] Luc De Raedt and Angelika Kimmig, 'Probabilistic (logic) programming concepts', *Machine Learning*, **100**(1), 5–47, (2015).

[7] Sebastijan Dumancic, Tias Guns, Wannes Meert, and Hendrik Blockleel, 'Auto-encoding logic programs', in *International Conference on Machine Learning, Location: Stockholm, Sweden*, (2018).

[8] Nir Friedman, 'The bayesian structural em algorithm', in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 129–138. Morgan Kaufmann Publishers Inc., (1998).

[9] Artur S d'Avila Garcez, Krysia B Broda, and Dov M Gabbay, *Neural-symbolic learning systems: foundations and applications*, Springer Science & Business Media, 2012.

[10] Robert Gens and Pedro Domingos, 'Discriminative learning of sum-product networks', in *Advances in Neural Information Processing Systems*, pp. 3239–3247, (2012).

[11] Lise Getoor and Ben Taskar. Introduction statistical relational learning, 2007.

[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio, *Deep learning*, volume 1, MIT press Cambridge, 2016.

[13] Emil Julius Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33, US Government Printing Office, 1954.

[14] Eric Jang, Shixiang Gu, and Ben Poole, 'Categorical reparameterization with gumbel-softmax', *arXiv preprint arXiv:1611.01144*, (2016).

[15] Diederik P Kingma and Max Welling, 'Auto-encoding variational bayes', *arXiv preprint arXiv:1312.6114*, (2013).

[16] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche, 'Probabilistic sentential decision diagrams.', in *KR*, (2014).

[17] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum, 'Human-level concept learning through probabilistic program induction', *Science*, **350**(6266), 1332–1338, (2015).

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al., 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, **86**(11), 2278–2324, (1998).

[19] Yitao Liang, Jessa Bekker, and Guy Van den Broeck, 'Learning the structure of probabilistic sentential decision diagrams', in *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, (2017).

[20] Yitao Liang and Guy Van den Broeck, 'Learning logistic circuits', *In Proceedings of the 33rd Conference on Artificial Intelligence (AAAI)*, (2019).

[21] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu, 'Learning entity and relation embeddings for knowledge graph completion', in *Twenty-ninth AAAI conference on artificial intelligence*, (2015).

[22] Chris J Maddison, Daniel Tarlow, and Tom Minka, 'A* sampling', in *Advances in Neural Information Processing Systems*, pp. 3086–3094, (2014).

[23] Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt, 'Deepproblog: Neural probabilistic logic programming', *arXiv preprint arXiv:1805.10872*, (2018).

[24] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov, 'Learning convolutional neural networks for graphs', in *International conference on machine learning*, pp. 2014–2023, (2016).

[25] Thammanit Pipatsrisawat and Adnan Darwiche, 'A lower bound on the size of decomposable negation normal form.', in *AAAI*, (2010).

[26] Hoifung Poon and Pedro Domingos, 'Sum-product networks: A new deep architecture', in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 689–690. IEEE, (2011).

[27] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra, 'Stochastic backpropagation and approximate inference in deep generative models', *arXiv preprint arXiv:1401.4082*, (2014).

[28] Matthew Richardson and Pedro Domingos, 'Markov logic networks', *Machine Learning*, **62**(1), 107–136, (February 2006).

[29] Tim Rocktäschel and Sebastian Riedel, 'End-to-end differentiable proving', in *Advances in Neural Information Processing Systems*, pp. 3788–3800, (2017).

[30] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap, 'A simple neural network module for relational reasoning', in *Advances in neural information processing systems*, pp. 4967–4976, (2017).

[31] Luciano Serafini and Artur d'Avila Garcez, 'Logic tensor networks: Deep learning and logical reasoning from data and knowledge', *arXiv preprint arXiv:1606.04422*, (2016).

[32] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 'Deep inside convolutional networks: Visualising image classification models and saliency maps', *arXiv preprint arXiv:1312.6034*, (2013).

[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, 'Intriguing properties of neural networks', *arXiv preprint arXiv:1312.6199*, (2013).

[34] Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt, 'Lifted probabilistic inference by first-order knowledge compilation', in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, ed., Toby Walsh, pp. 2178–2185. AAAI Press/International Joint Conferences on Artificial Intelligence, (2011).

[35] Han Xiao, Kashif Rasul, and Roland Vollgraf, 'Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms', *arXiv preprint arXiv:1708.07747*, (2017).

[36] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck, 'A semantic loss function for deep learning with symbolic knowledge', *CoRR*, **abs/1711.11157**, (2017).

[37] Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim, 'Joint relational embeddings for knowledge-based question answering', in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 645–650, (2014).

[38] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 'Understanding neural networks through deep visualization', *arXiv preprint arXiv:1506.06579*, (2015).

[39] Matthew D Zeiler and Rob Fergus, 'Visualizing and understanding convolutional networks', in *European conference on computer vision*, pp. 818–833. Springer, (2014).