Computing with Categories in Machine Learning

Eli Sennesh[1](✉), Tom Xu[2], and Yoshihiro Maruyama[2]

[1] Northeastern University, Boston, USA
sennesh.e@northeastern.edu
[2] Australian National University, Canberra, Australia
{tom.xu,yoshihiro.maruyama}@anu.edu.au

Abstract. Category theory has been successfully applied in various domains of science, shedding light on universal principles unifying diverse phenomena and thereby enabling knowledge transfer between them. Applications to machine learning have been pursued recently, and yet there is still a gap between abstract mathematical foundations and concrete applications to machine learning tasks. In this paper we introduce DisCoPyro as a categorical structure learning framework, which combines categorical structures (such as symmetric monoidal categories and operads) with amortized variational inference, and can be applied, e.g., in program learning for variational autoencoders. We provide both mathematical foundations and concrete applications together with comparison of experimental performance with other models (e.g., neuro-symbolic models). We speculate that DisCoPyro could ultimately contribute to the development of artificial general intelligence.

Keywords: Structure learning · Program learning · Symmetric monoidal category · Operad · Amortized variational Bayesian inference

1 Introduction

Category theory has been applied in various domains of mathematical science, allowing us to discover universal principles unifying diverse mathematical phenomena and thereby enabling knowledge transfer between them [7]. Applications to machine learning have been pursued recently [21]; however there is still a large gap between foundational mathematics and applicability in concrete machine learning tasks. This work begins filling the gap. We introduce the categorical structure learning framework DisCoPyro, a probabilistic generative model with amortized variational inference. We both provide mathematical foundations and compare with other neurosymbolic models on an example application.

Here we describe why we believe that DisCoPyro could contribute, in the long run, to developing human-level artificial general intelligence. Human intelligence supports graded statistical reasoning [15], and evolved to represent spatial (geometric) domains before we applied it to symbolic (algebraic) domains. Symmetric monoidal categories provide a mathematical framework for constructing both

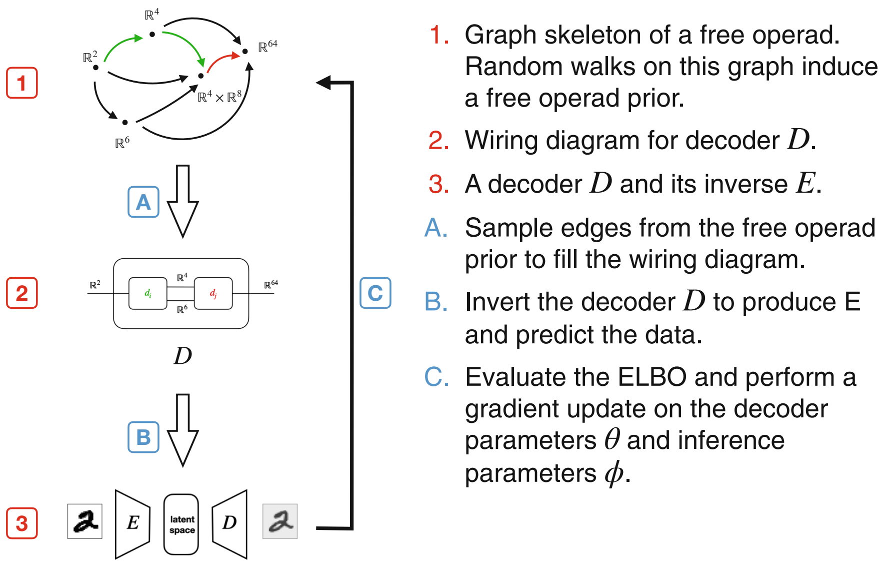
Supported by the JST Moonshot Programme on AI Robotics (JPMJMS2033-02).

symbolic computations (as in this paper) and geometrical spaces (e.g. [17]). We take Lake [15]'s suggestion to represent graded statistical reasoning via probability theory, integrating neural networks into variational inference for tractability. In terms of applications, we get competitive performance (see Subsect. 3.2 below) by variational Bayes, without resorting to reinforcement learning of structure as with modular neural networks [13,20].

The rest of the paper is organized as follows. In Sect. 2, we first introduce mathematical foundations of DisCoPyro (Subsects. 2 and 2.1). In Sect. 3 we then explain how to train DisCoPyro on a task (Subsect. 2.1) and provide experimental results and performance comparisons (Subsect. 3.2). Figure 1 demonstrates the flow of execution during the training procedure for the example task. We conclude and discuss further applications in Sect. 4. We provide an example implementation at https://github.com/neu-pml/discopyro with experiments at https://github.com/esennesh/categorical_bpl. DisCoPyro builds upon Pyro [2] (a deep universal probabilistic programming language), DisCoCat [4] (a distributional compositional model for natural language processing [4]), and the DisCoPy [6] library for computing with categories.

1.1 Notation

This paper takes symmetric monoidal categories (SMCs) $\mathcal{C}$ and their corresponding operads $\mathcal{O}$ as its mathematical setting. The reader is welcome to see Fong [7] for an introduction to these. SMCs are built from objects $Ob(\mathcal{C})$ and sets of



1. Graph skeleton of a free operad. Random walks on this graph induce a free operad prior.
2. Wiring diagram for decoder D.
3. A decoder D and its inverse E.

A. Sample edges from the free operad prior to fill the wiring diagram.
B. Invert the decoder D to produce E and predict the data.
C. Evaluate the ELBO and perform a gradient update on the decoder parameters θ and inference parameters ϕ.

Fig. 1. Example experiment. In each epoch of training, DisCoPyro learns variational autoencoder structures by sampling them from its skeleton according to a wiring diagram, then learning their faithful inverses as approximate posteriors.

morphisms $\mathcal{C}(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2)$ between objects $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2 \in Ob(\mathcal{C})$. Operads are built from types $Ty(\mathcal{O})$ and sets of morphisms $\mathcal{O}(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2)$ between types $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2 \in Ty(\mathcal{O})$. An SMC is usually written $(\mathcal{C}, \otimes, I)$ with a product operation $\otimes$ over objects and morphisms and a unit I of $\otimes$. In both settings, every object/type $\boldsymbol{\tau}$ has a unique identity morphism $id_{\boldsymbol{\tau}}$. Categories support composition $g \circ f$ on morphisms, and operads support indexed composition $g \circ_i f$ (for $i \in \mathbb{N}$) on morphisms.

2 Foundations of DisCoPyro

In essence, Definition 1 below exposes a finite number of building blocks (generators) from an SMC, and the morphisms constructed by composing those generators with $\circ$ and $\otimes$. For example, in categories of executable programs, a monoidal signature [6] specifies a domain-specific programming language.

Definition 1 (Monoidal signature in an SMC). *Given a symmetric monoidal category (SMC) $\mathcal{C}$ with the objects denoted by $Ob(\mathcal{C})$, a* monoidal signature[1] *$\mathcal{S} = (O, M)$ in that SMC consists of*

- *A finite set $O \subseteq Ob(\mathcal{C})$; and*
- *A finite set M consisting of elements $m : \mathcal{C}(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2)$ for some $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2 \in O$, such that $\forall \boldsymbol{\tau} \in O, m \neq id_{\boldsymbol{\tau}}$.*

The following free operad over a monoidal signature represents the space of all possible programs synthesized from the above building blocks (generators specified by the monoidal signature). Employing an operad rather than just a category allows us to reason about composition as nesting rather than just transitive combination; employing an operad rather than just a grammar allows us to reason about both the inputs and outputs of operations rather than just their outputs.

Definition 2 (Free operad over a signature). *The free operad $\mathcal{O}_{\mathcal{S}}$ over a signature $\mathcal{S} = (O, M)$ consists of*

- *A set of types (representations) $Ty(\mathcal{O}_{\mathcal{S}}) = \{I\} \cup O^{\otimes}$;*
- *For every $n \in \mathbb{N}^+$ a set of operations (mappings) $\mathcal{O}_{\mathcal{S}}(\boldsymbol{\tau}_0, \ldots, \boldsymbol{\tau}_{n-1}; \boldsymbol{\tau}_n)$ consisting of all trees with finitely many branches and leaves, in which each vertex v with $n-1$ children is labeled by a generator $m(v) \in M$ such that* $\mathrm{dom}(m(v))$ *has product length $n-1$;*
- *An identity operation $id_{\boldsymbol{\tau}} : \mathcal{O}_{\mathcal{S}}(\boldsymbol{\tau}; \boldsymbol{\tau})$ for every $\boldsymbol{\tau} \in Ty(\mathcal{O}_{\mathcal{S}})$; and*
- *A substitution operator $\circ_i$ defined by nesting a syntax tree $\mathcal{O}_{\mathcal{S}}(\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_m; \boldsymbol{\tau}_i)$ inside another $\mathcal{O}_{\mathcal{S}}(\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{n-1}; \boldsymbol{\tau}_n)$ when $i \in [1...n-1]$ to produce a syntax tree $\mathcal{O}_{\mathcal{S}}(\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{i-1}, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_m, \boldsymbol{\tau}_{i+1}, \ldots \boldsymbol{\tau}_{n-1}; \boldsymbol{\tau}_n)$.*

Intuitively, free operads share a lot in common with context-free grammars, and in fact Hermida [10] proved that they share a representation as directed acyclic hypergraphs. The definition of a signature in an SMC already hints at the structure of the appropriate hypergraph, but Algorithm 1 will make it explicit

```
**Input:** signature $\mathcal{S} = (O, M)$
**Output:** hypergraph $H = (V, E)$, recursion sites $R$
$V \leftarrow O$;
$E \leftarrow$ map($\lambda m.(\mathrm{dom}(m), \mathrm{cod}(m))$, $M$);
$R \leftarrow \emptyset$;
stack $\leftarrow \{v \in V \mid |v| > 1\}$;
while stack $\neq \emptyset$ do
    ty $\leftarrow$ pop(stack);
    inhabitants $\leftarrow$ map($\lambda c.\{(\mathrm{dom}(e), c) \mid e \in E, \mathrm{cod}(e) = c\}$, chunks(ty, $V$));
    foreach $((d_1, c_1), \dots, (d_k, c_k)) \in \bigotimes$ inhabitants do
        if not sublist($\bigotimes_{i \in [1..k]} d_i$, ty) then
            $R \leftarrow R \cup \{\otimes[(d_1, c_1), \dots, (d_k, c_k)]\}$
             $E \leftarrow E \cup \{(\bigotimes_{i \in [1..k]} d_i, \bigotimes_{i \in [1...k]} c_i)\}$;
            if $d \notin V$ then
                push(stack, $d$);
                $V \leftarrow V \cup \{\bigotimes_{i \in [1..k]} d_i\}$;
            end
        end
    end
end
return $(V, E), R$
```

Algorithm 1: Algorithm to represent a free operad as a hypergraph. The function `chunks` partitions `ty` into sublists, each an element of the set V.

and add edges to the hypergraph corresponding to nesting separate operations in parallel (or equivalently, to monoidal products in the original SMC). In the hypergraph produced by Algorithm 1, each vertex corresponds to a non-product type and each hyperedge has a list of vertices as its domain and codomain. Each such hypergraph admits a representation as a graph as well, in which the hyperedges serve as nodes and the lists in their domains and codomains serve as edges. We will use this graph representation $G \simeq H$ to reason about morphisms as paths between their domain and codomain.

We will derive a probabilistic generative model over morphisms in the free operad from this graph representation's directed adjacency matrix A_G.

Definition 3 (Transition distance in a directed graph). *The "transition distance" between two indexed vertices v_i, v_j is the negative logarithm of the i, j entry in the exponentiated adjacency/transition matrix*

$$d(v_i, v_j) = -\log\left(\left[e^{A_G}\right]_{i,j}\right), \tag{1}$$

where the matrix exponential is defined by the series

$$e^{A_G} = \sum_{n=1}^{\infty} \frac{(A_G)^n}{n!}.$$

[1] Also called a "hypersignature".

```
Data: hypergraph $(V, E)$
function Path($\boldsymbol{\tau}_-, \boldsymbol{\tau}_+, \beta, \mathbf{w}$)
    $i \leftarrow 1$;
    $\boldsymbol{\tau}_i \leftarrow \boldsymbol{\tau}_-$;
    $f \leftarrow id_{ty-}$;
    while $\boldsymbol{\tau}_i \neq \boldsymbol{\tau}_+$ do
        $e_i \sim \pi(e \in E \mid \boldsymbol{\tau}_i, \boldsymbol{\tau}_+ \ \beta)$;
        $\boldsymbol{\tau}_i \leftarrow \mathrm{cod}(e_i)$;
        $f \leftarrow f \fatsemi$ Generator($e_i, \beta, \mathbf{w}$);
        $i \leftarrow i + 1$;
    end
    return $f$
end
```

Algorithm 2: The Markov chain constructing paths between types

A soft-minimization distribution over this transition distance will, in expectation and holding the indexed target vertex constant, define an probabilistic generative model over paths through the hypergraph.

Definition 4 (Free operad prior). *Consider a signature $\mathcal{S} = (O, M)$ and its resulting graph representation $G = (V, E)$ and recursion sites R, and then condition upon a domain and codomain $\boldsymbol{\tau}_-, \boldsymbol{\tau}_+ \in Ty(\mathcal{O}_\mathcal{S})$ represented by vertices in the graph. The* free operad prior *assigns a probability density to all finite paths $\mathbf{e} = (e_1, e_2, \ldots, e_n)$ with* $\mathrm{dom}(\mathbf{e}) = \boldsymbol{\tau}_-$ *and* $\mathrm{cod}(\mathbf{e}) = \boldsymbol{\tau}_+$ *by means of an absorbing Markov chain. First the model samples a "precision" β and a set of "weights" $\mathbf{w}$*

$$\beta \sim \gamma(1, 1) \qquad\qquad \mathbf{w} \sim \mathrm{Dirichlet}\left(\vec{\mathbf{1}}^{(|M|+|R|)}\right).$$

Then it samples a path (from the absorbing Markov chain in Algorithm 2) by soft minimization (biased towards shorter paths by β) of the transition distance

$$\pi(e \in E \mid \boldsymbol{\tau}_1, \boldsymbol{\tau}_2; \beta) := \frac{\exp\left(-\frac{1}{\beta} d(\mathrm{cod}(e), \boldsymbol{\tau}_2)\right)}{\sum_{e' \in E : \mathrm{dom}(e') = \boldsymbol{\tau}_1} \exp\left(-\frac{1}{\beta} d(\mathrm{cod}(e'), \boldsymbol{\tau}_2)\right)}. \tag{2}$$

Equation 2 will induce a transition operator T which, by Theorem 2.5.3 in Latouche and Ramaswami [16], will almost-surely reach its absorbing state corresponding to $\boldsymbol{\tau}_2$. This path can then be filled in according to Algorithm 3. The precision β increases at each recursion to terminate with shorter paths. We denote the induced joint distribution as

$$p(f, \mathbf{w}, \beta; \boldsymbol{\tau}_-, \boldsymbol{\tau}_+) = p(f \mid \beta, \mathbf{w}; \boldsymbol{\tau}_-, \boldsymbol{\tau}_+) p(\mathbf{w}) p(\beta). \tag{3}$$

Having a probabilistic generative model over operations in the free operad over a signature, we now need a way to specify a structure learning problem. Definition 5 provides this by specifying what paths to sample (each box specifies a call to Algorithm 2) and how to compose them.

```
Data: hypergraph $(V, E)$, generators $M$, recursion sites $R$
function Generator($e, \beta, \mathbf{w}$)
    $gs \leftarrow \{m \in M \mid (\mathrm{dom}(m), \mathrm{cod}(m)) = (\mathrm{dom}(e), \mathrm{cod}(e))\}$;
    $gs \leftarrow gs \cup \{\otimes[(d_1, c_1), \ldots, (d_k, c_k)] \in R \mid \bigotimes_{i \in [1 \ldots k]} d_i = \mathrm{dom}(e) \wedge \bigotimes_{i \in [1 \ldots k]} c_i = \mathrm{cod}(e)\}$;
    foreach $j \in \{1, \ldots, |gs|\}$ do
        if $gs_j = \otimes(\ldots)$ then
            $\mathbf{w}_j \leftarrow \mathbf{w}_j / \beta$;
        end
    end
    $\mathbf{w}_e \leftarrow [\mathbf{w}_n \mid g \in gs, g \in M, n = \mathtt{index}(g, M)]$;
    $\mathbf{w}_e \leftarrow \mathbf{w}_e + [\mathbf{w}_{|M|+n} \mid g \in gs, g \in R, n = \mathtt{index}(g, R)]$;
    $j \sim \mathrm{Discrete}(\mathbf{w}_e)$;
    if $gs_j = \otimes[(d_1, c_1), \ldots, (d_k, c_k)]$ then
        return $\bigotimes_{l=1}^{k} \mathtt{Path}(d_l, c_l, \beta + 1, \mathbf{w})$
    else
        return $gs_j$
    end
end
```

Algorithm 3: Filling in an edge in the path with a morphism

Definition 5 (Wiring diagram). *An acyclic, O-typed* wiring diagram *[7, 22] is a map from a series of* internal boxes*, each one defined by its domain and codomain pair $(\boldsymbol{\tau}_i^-, \boldsymbol{\tau}_i^+)$ to an* outer box *defined by domain and codomain $(\boldsymbol{\tau}_n^-, \boldsymbol{\tau}_n^+)$*

$$\varPhi : \mathcal{O}_{\mathcal{S}}(\boldsymbol{\tau}_1^-, \boldsymbol{\tau}_1^+) \times \ldots \times \mathcal{O}_{\mathcal{S}}(\boldsymbol{\tau}_{n-1}^-, \boldsymbol{\tau}_{n-1}^+) \rightarrow \mathcal{O}_{\mathcal{S}}(\boldsymbol{\tau}_n^-, \boldsymbol{\tau}_n^+).$$

Acyclicity requires that connections ("wires") can extend only from the outer box's domain to the domains of inner boxes, from the inner boxes codomains to the outer box's codomain, and between internal boxes such that no cycles are formed in the directed graph of connections between inner boxes.

Given a **user-specified** wiring diagram $\varPhi$, we can then wite the complete prior distribution over all latent variables in our generative model.

$$p(f, \mathbf{w}, \beta; \varPhi, \mathcal{S}) = p(\beta)p(\mathbf{w}) \prod_{(\boldsymbol{\tau}_i^-, \boldsymbol{\tau}_i^+) \in \varPhi} p(f_i \mid \beta, \mathbf{w}; \boldsymbol{\tau}_i^-, \boldsymbol{\tau}_i^+). \tag{4}$$

If a user provides a likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z} \mid f)$ relating the learned structure f to data $\mathbf{x}$ (via latents $\mathbf{z}$) we will have a joint density

$$p(\mathbf{x}, \mathbf{z}, f, \mathbf{w}, \beta; \varPhi, \mathcal{S}) = p(\mathbf{x} \mid f)p(f, \mathbf{w}, \beta; \varPhi, \mathcal{S}), \tag{5}$$

and Eq. 5 then admits inference from the data $\mathbf{x}$ by Bayesian inversion

$$p(\mathbf{z}, f, \mathbf{w}, \beta \mid \mathbf{x}; \varPhi, \mathcal{S}) = \frac{p(\mathbf{x}, \mathbf{z}, f, \mathbf{w}, \beta; \varPhi, \mathcal{S})}{p_{\boldsymbol{\theta}}(\mathbf{x}; \varPhi, \mathcal{S})}. \tag{6}$$

Section 2.1 will explain how to approximate Eq. 6 by stochastic gradient-based optimization, yielding a maximum-likelihood estimate of $\boldsymbol{\theta}$ and an optimal approximation for the parametric family $\boldsymbol{\phi}$ to the true Bayesian inverse.

2.1 Model Learning and Variational Bayesian Inference

Bayesian inversion relies on evaluating the model evidence $p_{\boldsymbol{\theta}}(\mathbf{x}; \Phi, \mathcal{S})$, which typically has no closed form solution. However, we can transform the high-dimensional integral over the joint density into an expectation

$$\begin{aligned} p_{\boldsymbol{\theta}}(\mathbf{x}; \Phi, \mathcal{S}) &= \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta; \Phi, \mathcal{S}) d\mathbf{z}\, df_{\boldsymbol{\theta}}\, d\mathbf{w}\, d\beta \\ &= \mathbb{E}_{p(f_{\boldsymbol{\theta}}, \mathbf{w}, \beta; \Phi, \mathcal{S})} \left[p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta; \Phi, \mathcal{S}) \right], \end{aligned}$$

and then rewrite that expectation into one over the proposal

$$\begin{aligned} &\mathbb{E}_{p(f_{\boldsymbol{\theta}}, \mathbf{w}, \beta; \Phi, \mathcal{S})} \left[p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta; \Phi, \mathcal{S}) \right] = \\ &\qquad\qquad \mathbb{E}_{q_{\phi}(\mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta \mid \mathbf{x}; \Phi, \mathcal{S})} \left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta; \Phi, \mathcal{S})}{q_{\boldsymbol{\phi}}(\mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta \mid \mathbf{x}; \Phi, \mathcal{S})} \right]. \end{aligned}$$

For constructing this expectation, DisCoPyro provides both the functorial inversion described in Sect. 3.1 and an amortized form of Automatic Structured Variational Inference [1] suitable for any universal probabilistic program.

Jensen's Inequality says that expectation of the log density ratio will lower-bound the log expected density ratio

$$\begin{aligned} &\mathbb{E}_{q_{\phi}(\mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta \mid \mathbf{x}; \Phi, \mathcal{S})} \left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta; \Phi, \mathcal{S})}{q_{\boldsymbol{\phi}}(\mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta \mid \mathbf{x}; \Phi, \mathcal{S})} \right] \leq \\ &\qquad\qquad \log \mathbb{E}_{p(f_{\boldsymbol{\theta}}, \mathbf{w}, \beta; \Phi, \mathcal{S})} \left[p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta; \Phi, \mathcal{S}) \right], \end{aligned}$$

so that the left-hand side provides a lower bound to the true model evidence

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{q_{\phi}(\mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta \mid \mathbf{x}; \Phi, \mathcal{S})} \left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta; \Phi, \mathcal{S})}{q_{\boldsymbol{\phi}}(\mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta \mid \mathbf{x}; \Phi, \mathcal{S})} \right] \leq \log p_{\boldsymbol{\theta}}(\mathbf{x}; \Phi, \mathcal{S}).$$

Maximizing this *evidence lower bound* (ELBO) by Monte Carlo estimation of its values and gradients (using Pyro's built-in gradient estimators) will estimate the model parameters $\boldsymbol{\theta}$ by maximum likelihood and train the proposal parameters $\boldsymbol{\phi}$ to approximate the Bayesian inverse (Eq. 6) [12].

3 Example Application and Training

The framework of connecting a morphism to data via a likelihood with intermediate latent random variables allows for a broad variety of applications. This section will demonstrate the resulting capabilities of the DisCoPyro framework. Section 3.1 will describe an example application of the framework to deep probabilistic program learning for generative modeling. Section 3.2 that describe application's performance as a generative model.

Table 1. Average log-evidence on the Omniglot evaluation set across models. Our free operad model obtains the highest higher log-evidence per data dimension.

Model	Image Size	Learns Structure	log-$\hat{Z}$/dim
Sequential Attention [19]	28×28	✗	-0.1218
Variational Homoencoder [11] (PixelCNN)	28×28	✗	-0.0780
Graph VAE [9]	28×28	✓	-0.1334
Generative Neurosymbolic [5]	105×105	✓	-0.0348
Free Operad DGM (ours)	28×28	✓	$\mathbf{-0.0148}$

3.1 Deep Probabilistic Program Learning with DisCoPyro

As a demonstrative experiment, we constructed an operad $\mathcal{O}$ whose generators implemented Pyro building blocks for deep generative models $f_{\boldsymbol{\theta}}$ (taken from work on structured variational autoencoders [12,19,24]) with parameters $\boldsymbol{\theta}$. We then specified the one-box wiring diagram $\Phi : (I, \mathbb{R}^{28\times28}) \to (I, \mathbb{R}^{28\times28})$ to parameterize the DisCoPyro generative model. We trained the resulting free operad model on MNIST just to check if it worked, and on the downsampled (28×28) Omniglot dataset for few-shot learning [14] as a challenge. Since the data $\mathbf{x} \in \mathbb{R}^{28\times28}$, our experimental setup induces the joint likelihood

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}, f_{\boldsymbol{\theta}}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}, f_{\boldsymbol{\theta}}), \boldsymbol{I\tau})$$
$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z} \mid f_{\boldsymbol{\theta}}) = p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}, f_{\boldsymbol{\theta}}) p_{\boldsymbol{\theta}}(\mathbf{z} \mid f_{\boldsymbol{\theta}}).$$

DisCoPyro provides amortized variational inference over its own random variables via neural proposals $q_{\boldsymbol{\phi}}$ for the "confidence" $\beta \sim q_{\boldsymbol{\phi}}(\beta \mid \mathbf{x})$ and the "preferences" over generators $\mathbf{w} \sim q_{\boldsymbol{\phi}}(\mathbf{w} \mid \mathbf{x})$. Running the core DisCoPyro generative model over structures $f_{\boldsymbol{\theta}}$ then gives a proposal over morphisms in the free operad, providing a generic proposal for DisCoPyro's latent variables

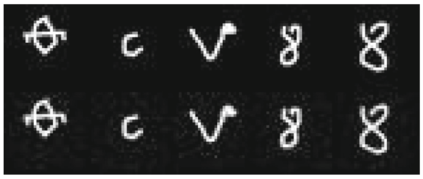
$$q_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}, \mathbf{w}, \beta \mid \mathbf{x}; \Phi, \mathcal{S}) = p(f_{\boldsymbol{\theta}} \mid \mathbf{w}, \beta; \Phi, \mathcal{S}) q_{\boldsymbol{\phi}}(\beta \mid \mathbf{x}) q_{\boldsymbol{\phi}}(\mathbf{w} \mid \mathbf{x}).$$

Since the morphisms in our example application are components of deep generative models, each generating morphism can be simply "flipped on its head" to get a corresponding neural network design for a proposal. We specify that proposal as $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}, f_{\boldsymbol{\theta}})$; it constructs a faithful inverse [23] compositionally via a dagger functor (for further description of Bayesian inversion as a dagger functor, please see Fritz [8]). Our application then has a complete proposal density

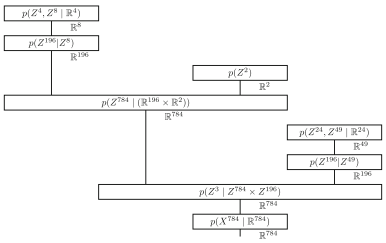
$$q_{\boldsymbol{\phi}}(\mathbf{z}, f_{\boldsymbol{\theta}}, \mathbf{w}, \beta \mid \mathbf{x}; \Phi, \mathcal{S}) = q_{\boldsymbol{\phi}}(\mathbf{z} \mid f_{\boldsymbol{\theta}}, \mathbf{x}) q_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}, \mathbf{w}, \beta \mid \mathbf{x}; \Phi, \mathcal{S}). \qquad (7)$$

3.2 Experimental Results and Performance Comparison

Table 1 compares our free operad model's performance to other structured deep generative models. We report the estimated log model evidence. Our free operad



(a) Omniglot characters (above) and their reconstructions (below)


$p(Z^4, Z^8 \mid \mathbb{R}^4)$
$\mathbb{R}^8$
$p(Z^{196}|Z^8)$
$\mathbb{R}^{196}$
$p(Z^2)$
$\mathbb{R}^2$
$p(Z^{784} \mid (\mathbb{R}^{196} \times \mathbb{R}^2))$
$\mathbb{R}^{784}$
$p(Z^{24}, Z^{49} \mid \mathbb{R}^{24})$
$\mathbb{R}^{49}$
$p(Z^{196}|Z^{49})$
$\mathbb{R}^{196}$
$p(Z^3 \mid Z^{784} \times Z^{196})$
$\mathbb{R}^{784}$
$p(X^{784} \mid \mathbb{R}^{784})$
$\mathbb{R}^{784}$

(b) A string diagram sampled from the free operad model's Bayesian inverse.

Fig. 2. Reconstructions (left) generated by inference in the diagrammatic generative model (right) on handwritten characters in the Omniglot evaluation set. The string diagram shows a model that generates a glimpse, decodes it into an image canvas via a variational ladder decoder, and then performs a simpler process to generate another glimpse and insert it into the canvas.

prior over deep generative models achieves the best log-evidence per data dimension, although standard deviations for the baselines do not appear to be available for comparison. Some of the older baselines, such as the sequential attention model and the variational homoencoder, fix a composition structure ahead of time instead of learning it from data as we do. Figure 2 shows samples from the trained model's posterior distribution, including reconstruction of evaluation data (Fig. 2a) and an example structure for that data (Fig. 2b).

Historically, Lake [14] proposed the Omniglot dataset to challenge the machine learning community to achieve human-like concept learning by learning a single generative model from very few examples; the Omniglot challenge requires that a model be usable for classification, latent feature recognition, concept generation from a type, and exemplar generation of a concept. The deep generative models research community has focused on producing models capable of few-shot reconstruction of unseen characters. [11,19] fixed as constant the model architecture, attempting to account for the compositional structure in the data with static dimensionality. In contrast, [5,9] performed joint structure learning, latent variable inference, and data reconstruction as we did.

4 Discussion

This paper described the DisCoPyro system for generative Bayesian structure learning, along with its variational inference training procedures and an example application. Section 2 described DisCoPyro's mathematical foundations in category theory, operad theory, and variational Bayesian inference. Section 3 showed DisCoPyro to be competitive against other models on a challenge dataset.

As Lake [15] suggested, (deep) probabilistic programs can model human intelligence across more domains than handwritten characters. Beyond programs, neural network architectures, or triangulable manifolds, investigators have applied operads and SMCs to chemical reaction networks, natural language processing, and the systematicity of human intelligence [3, 18]. This broad variety of applications motivates our interest in representing the problems a generally intelligent agent must solve in terms of operadic structures, and learning those structures jointly with their contents from data.

Acknowledgements. The authors would like to thank the anonymous reviewers for their constructive feedback and encouragement. This work was supported by the Japan Science and Technology Agency (JST JPMJMS2033) and National Science Foundation of the United States of America (NSF 2047253).

References

1. Ambrogioni, L., et al.: Automatic structured variational inference. In: Proc. AISTATS, pp. 676–684 (2021)
2. Bingham, E., et al.: Pyro: deep universal probabilistic programming. JMLR **20**, 973–978 (2019)
3. Bradley, T.D.: What is applied category theory? arXiv preprint arXiv:1809.05923 (2018)
4. Coecke, B., Sadrzadeh, M., Clark, S.: Mathematical foundations for a compositional distributional model of meaning. arXiv preprint arXiv:1003.4394 (2010)
5. Feinman, R., Lake, B.M.: Learning task-general representations with generative neuro-symbolic modeling. In: Proc. ICLR (2021)
6. de Felice, G., Toumi, A., Coecke, B.: DisCoPy: monoidal categories in python. In: Applied Category Theory Conference, pp. 1–20. EPTCS (2020). http://arxiv.org/abs/2005.02975arXiv: 2005.02975
7. Fong, B., Spivak, D.I.: Seven Sketches in Compositionality: An Invitation to Applied Category Theory. Cambridge University Press, Cambridge (2019)
8. Fritz, T.: A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. Adv. Math. **370**, 107239 (2020)
9. He, J., et al.: Variational autoencoders with jointly optimized latent dependency structure. In: Proc. ICLR 2019, pp. 1–16 (2019)
10. Hermida, C., Makkaiy, M., Power, J.: Higher dimensional multigraphs. In: Proceedings - Symposium on Logic in Computer Science, pp. 199–206 (1998)
11. Hewitt, L.B., et al.: The variational homoencoder: learning to learn high capacity generative models from few examples. Proc. UAI **2018**, 988–997 (2018)
12. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
13. Kirsch, L., Kunze, J., Barber, D.: Modular networks: learning to decompose neural computation. In: Proc. NeurIPS 2018, pp. 2408–2418 (2018)
14. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: The Omniglot challenge: a 3-year progress report. Curr. Opin. Behav. Sci. **29**, 97–104 (2019)
15. Lake, B.M., Ullman, T.D., Tenenbaum, J.B., Gershman, S.J.: Building machines that learn and think like people. Behav. Brain Sci. **40**, e253 (2017)
16. Latouche, G., Ramaswami, V.: Introduction to matrix analytic methods in stochastic modeling. In: Society for Industrial and Applied Mathematics (1999)
17. Milnor, J.: On spaces having the homotopy type of CW-complex. Trans. Am. Math. Soc. **90**, 272–280 (1959)

18. Phillips, S., Wilson, W.H.: Categorial compositionality: a category theory explanation for the systematicity of human cognition. PLoS Comput. Biol. **6**, e1000858 (2010)
19. Rezende, D.J., et al.: One-shot generalization in deep generative models. In: International Conference on Machine Learning, vol. 48 (2016)
20. Rosenbaum, C., Klinger, T., Riemer, M.: Routing networks: adaptive selection of non-linear functions for multi-task learning. In: Proc. ICLR 2018, pp. 1–10 (2018)
21. Shiebler, D., Gavranović, B., Wilson, P.: Category theory in machine learning. In: Proc. ACT 2021 (2021)
22. Spivak, D.I.: The operad of wiring diagrams: formalizing a graphical language for databases, recursion, and plug-and-play circuits (2013). arxiv: 1305.0297
23. Webb, S., et al.: Faithful inversion of generative models for effective amortized inference. In: Proc. NIPS 2018, pp. 3074–3084 (2018)
24. Zhao, S., Song, J., Ermon, S.: Learning hierarchical features from generative models. In: International Conference on Machine Learning (2017)