1st Symposium on Advances in Approximate Bayesian Inference, 2018 1–27

The LORACs prior for VAEs: Letting the Trees Speak for the Data

Sharad Vikram* SVIKRAM@CS.UCSD.EDU
U.C. San Diego

Matthew D. Hoffman MHOFFMAN@GOOGLE.COM
Google AI

Matthew J. Johnson MATTJJ@GOOGLE.COM
Google Brain

1. Introduction

The "default" prior for variational autoencoders (VAEs; Kingma and Welling, 2014; Rezende et al., 2014) is an isotropic normal, but if the natural factors of variation in the dataset exhibit discrete structure or are not independent, then the isotropic-normal prior will actually encourage learning representations that *mask* this structure (Hoffman and Johnson, 2016). In this abstract, (1) we propose using the time-marginalized coalescent (TMC; Boyles and Welling, 2012), as a prior for VAEs, resulting in a deep Bayesian nonparametric model that can discover hierarchical cluster structure in complex, high-dimensional datasets, (2) we develop a minibatch-friendly inference procedure for fitting TMCs based on an inducing-point approximation, which scales to arbitrarily large datasets and (3) show that our model's learned latent representations consistently outperform those learned by other variational (and classical) autoencoders when evaluated on downstream classification and retrieval tasks. Please refer to Appendix B for related work.

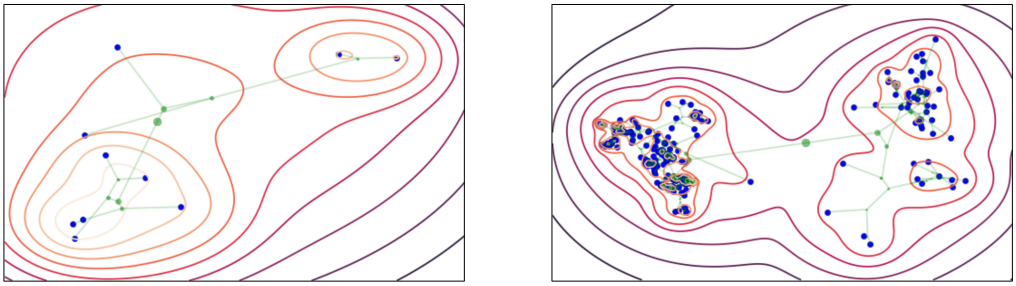
2. Background

2.1. Bayesian priors for hierarchical clustering

Bayesian nonparametric hierarchical clustering introduces a prior distribution over trees $r(\tau)$ and a likelihood model for data $r(z_{1:N} \mid \tau)$, with the goal of sampling the posterior distribution $r(\tau \mid z_{1:N})$.[1] In this abstract, we focus on rooted binary trees with N labeled leaves adorned with branch lengths, called *phylogenies*, and the time-marginalized coalescent (TMC; Boyles and Welling, 2012). The TMC is an exchangeable distribution over phylogenies which first samples a tree structure (V, E) (vertex and edge sets) and times $\{t_v\}_{v \in V}$ for each node. The times implicitly define branch lengths over the tree by taking the difference between a child and parent's time. We then define $r(z_{1:N} \mid \tau)$ to be a d-dimensional

* Work done while interning at Google.

1. We use r to denote probability distributions relating to the TMC and distinguish from p and q distributions used later in the abstract.



(a) (b)

Figure 1: Independent samples from a time-marginalized coalescent (TMC) prior and two-dimensional Gaussian random walk likelihood model (10 and 300 leaves respectively). Contours in the plots correspond to posterior predictive density $r(z_{N+1} \mid z_{1:N}, \tau)$. As the number of leaves grow, the predictive density grows more complex.

Gaussian random walk (GRW) over the tree structure, using branch lengths as variances. The GRW when at observed at the leaves corresponds to vectors $z_{1:N}$. The TMC's associated posterior predictive density, $r(z_{N+1} \mid z_{1:N}, \tau)$, first samples a branch e_{N+1} and a time t_{N+1} and computes the posterior distribution over a leaf node attached at e_{N+1} and t_{T+1}, $r(z_{N+1} \mid e_{N+1}, t_{N+1}, z_{1:N}, \tau)$ according to the GRW. Shown in Figure 1 are samples from the TMC/GRW prior, along with their associated posterior predictive densities. Inference in the TMC, i.e. computing the posterior $r(\tau \mid z_{1:N})$, can performed with Metropolis-Hastings, as the joint likelihood $r(\tau, z_{1:N})$ can be calculated efficiently. Details of the TMC generative process, posterior predictive density, and inference are in Appendix A.

3. The LORACs prior for VAEs

Consider the TMC prior for a VAE, generating the latent values $z_{1:N}$ of a VAE according to a TMC, i.e. $\tau \sim r(\tau)$, $z_{1:N}|\tau \sim r(z_{1:N} \mid \tau)$, then generating observations $x_{1:N}$ using a neural network observation model $p_\theta(x_n \mid z_n)$. Inference in this model, the TMC-VAE, is computationally limiting. $r(z_{1:N} \mid \tau)$ does not factor independently over z_n's, rendering minibatch-gradient ascent of the ELBO impossible. Furthermore, a phylogeny τ will have as many leaves as the size of the data, so inference over tree structure slows down significantly as the dataset grows. For details of the TMC-VAE and its limitations, see Appendix C.1.

We thus introduce a novel approximation to the TMC prior. Our key idea is to use a set of learnable *inducing points* as the leaves of the tree in the latent space, analogous to inducing-input approximations for Gaussian processes (Snelson and Ghahramani, 2006). In this model, latent vectors $z_{1:N}$ are not directly hierarchically clustered, but rather are independent samples from the posterior predictive density of a TMC. We call this the Latent ORganization of Arboreal Clusters (*LORACs*, pronounced "lorax") prior.

To define the LORACs prior $p(\tau, z_{1:N})$, we first define an auxiliary TMC distribution $r(\tau, s_{1:M})$ with M leaf locations $s_{1:M}$. We treat $s_{1:M}$ as a set of learnable free parameters, and define the conditional $r(\tau \mid s_{1:M})$ as the LORACs prior on phylogenies τ: $p(\tau; s_{1:M}) \triangleq$

$r(\tau \mid s_{1:M})$. That is, we choose the prior on phylogenies τ to be the posterior distribution of a TMC with pseudo-observations $s_{1:M}$. Next, we define the LORACs prior on locations $z_n \mid \tau$ as a conditionally independent draw from the posterior predictive distribution $r(s_{M+1} = z_n \mid \tau, s_{1:M})$. To complete the model, we use an observation likelihood parameterized by a neural network. By using the learned inducing points $s_{1:M}$, we avoid the main difficulty of inference in the TMC-VAE, namely the need to do inference over all N points in the dataset. Instead, dependence between datapoints is mediated by the set of inducing points $s_{1:M}$, which has a size independent of N. As a result, with the LORACs prior, minibatch-based learning becomes tractable even for very large datasets.

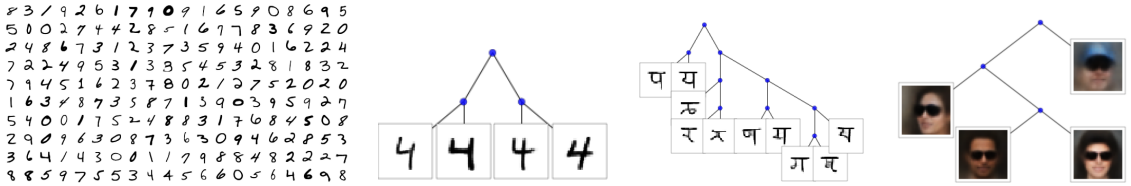
This technique presents its own inference challenges. However, we are still able to efficiently optimize the ELBO in the LORACs-VAE through a hybrid of amortized inference via recognition networks and Metropolis-Hastings over tree structure. Details of this inference procedure are in Appendix C.2.

4. Results

We evaluated the LORACs prior on three separate datasets: dynamically binarized MNIST (LeCun and Cortes, 2010), Omniglot (Lake et al., 2015), and CelebA (Liu et al., 2015). Details of this setup can be found in Appendix D.1.

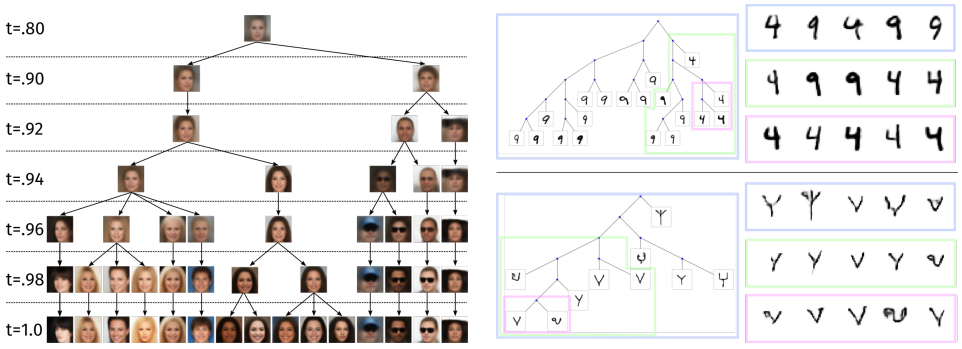
Qualitative results We show inducing points and learned subtrees in Figure 2. We also visualize "evolution"(taking slices of the tree at particular times) and conditional sampling in Figure 3. These visual results indicate that LORACs prior offers new and useful ways to explore and interpret latent representations of data. For details and additional visualizations, please see Appendix D.2.

Quantitative results We first trained baseline VAEs with the following priors: no prior, standard normal, VampPrior (Tomczak and Welling, 2018), DVAE♯ (Vahdat et al., 2018), and masked autoregressive flow (MAF; Papamakarios et al., 2017). We evaluated the learned representations in few-shot classification, information retrieval, and held-out log-likelihood tasks and found that LORACs prior representations are better at downstream tasks, despite middling held-out log-likelihood numbers (see Figure 4, Figure 5, Appendix D.3).



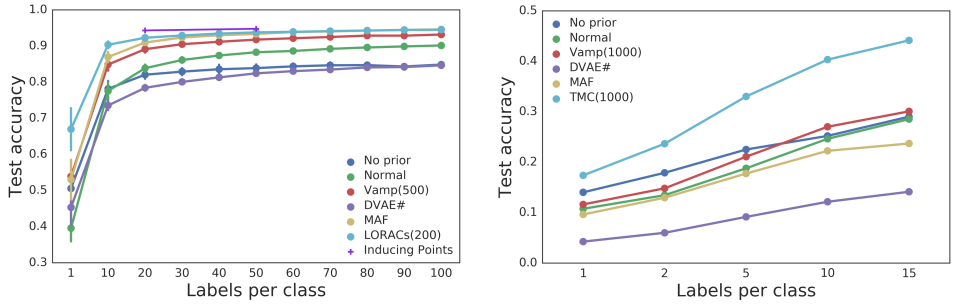
(*a*) LORACs(200) inducing points for MNIST. (*b*) MNIST (*c*) Omniglot (*d*) CelebA

Figure 2: In (a) are inducing points visualized by passing them through the decoder. (b), (c), and (d) are example learned subtrees from each dataset.



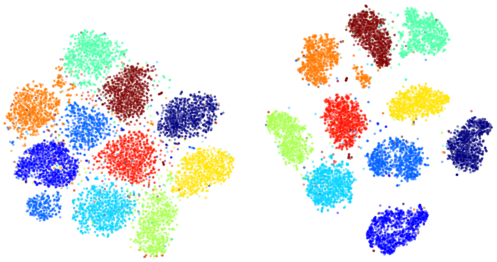
(a) (b)

Figure 3: (a) shows evolution of a CelebA over some inducing points. We create this visualization by taking slices of the tree at particular times and observing their decoded expected values under a GRW. (b) shows conditionally sampling the posterior predictive density restricted to particular subtrees for MNIST and Omniglot.



(a) MNIST (b) Omniglot

Figure 4: Few-shot classification accuracy across different priors.



(a) Normal (b) LORACs(200)

Prior	MNIST	Omniglot
No prior	0.429	0.078
Normal	0.317	0.057
VAMP	0.502	0.063
DVAE#	0.490	0.024
MAF	0.398	0.070
LORACs	**0.626**	**0.087**

(c)

Figure 5: (a) and (b) show t-SNE visualizations of the latent space of MNIST with a normal and LORACs prior. (c) shows averaged-precision recall AUC for latent representations on MNIST/Omniglot.

References

Levi Boyles and Max Welling. The time-marginalized coalescent prior for hierarchical clustering. In *Advances in Neural Information Processing Systems*, pages 2969–2977, 2012.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296, 2009.

Prasoon Goyal, Zhiting Hu, Xiaodan Liang, Chenyu Wang, Eric P Xing, and Carnegie Mellon. Nonparametric variational auto-encoders for hierarchical representation learning. In *ICCV*, pages 5104–5112, 2017.

Thomas L Griffiths, Michael I Jordan, Joshua B Tenenbaum, and David M Blei. Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems*, pages 17–24, 2004.

Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, 2016.

Matthew J Johnson, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. Composing graphical models with neural networks for structured representations and fast inference. In *Neural Information Processing Systems*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2014.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL http://science.sciencemag.org/content/350/6266/1332.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Wu Lin, Nicolas Hubacher, and Mohammad Emtiyaz Khan. Variational message passing with structured inference networks. *arXiv preprint arXiv:1803.05589*, 2018.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

Radford M Neal. Density modeling and clustering using dirichlet diffusion trees. *Bayesian statistics*, 7:619–629, 2003.

George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286, 2014.

Tim Salimans, Andrej Karpathy, Xi Chen, Diederik P Kingma, and Yaroslav Bulatov. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations (ICLR)*, 2017.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.

Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223, 2018.

Arash Vahdat, Evgeny Andriyash, and William G Macready. Dvae#: Discrete variational autoencoders with relaxed Boltzmann priors. In *Neural Information Processing Systems (NIPS)*, 2018.

Appendix A. Time-marginalized coalescent (TMC)

The time-marginalized coalescent defines a prior distribution over phylogenies. A phylogeny $\tau = (V, E, T)$ is a directed rooted full binary tree, with vertex set V and edges E, together with time labels $T : V \to [0, 1]$ where we denote $t_v = T(v)$. The vertex set V is partitioned into N leaf vertices V_{leaf} and $N - 1$ internal vertices V_{int}, so that $V = V_{\text{int}} \cup V_{\text{leaf}}$, and we take $V_{\text{leaf}} = \{1, 2, \ldots, N\}$ to simplify notation for identifying leaves with N data points. The directed edges of the tree are encoded in the edge set $E \subset V_{\text{int}} \times V$, where we denote the root vertex as v_{root} and for $v \in V \setminus \{v_{\text{root}}\}$ we denote the parent of v as $\pi(v) = w$ where $(w, v) \in E$.

The TMC samples a random tree structure (V, E) by a stochastic process in which the N leaves are recursively merged uniformly at random until only one vertex is left. This process yields the probability mass function on valid (V, E) pairs given by

$$r(V, E) = \frac{(N-1)!}{\prod_{v \in V_{\text{int}}} c(v)} \prod_{i=1}^{N-1} \binom{i+1}{2}^{-1}, \tag{A.1}$$

where $c(v)$ denotes the number of internal vertices in the subtree rooted at v. Given the tree structure, time labels are generated via the stick-breaking process

$$t_v = \begin{cases} 0 & v = v_{\text{root}}, \\ 1 & v \in V_{\text{leaf}}, \\ t_{\pi(v)} - \beta_v(1 - t_{\pi(v)}) & v \in V_{\text{int}} \setminus \{v_{\text{root}}\}, \end{cases} \tag{A.2}$$

where $\beta_v \overset{\text{iid}}{\sim} \text{Beta}(a, b)$ for $v \in V$. These time labels encode a branch length $t_v - t_{\pi(v)}$ for each edge $e = (\pi(v), v) \in E$. We denote the overall density on phylogenies with N leaves as $\text{TMC}_N(\tau; a, b)$.

Finally, to connect the TMC prior to data in $\mathbb{R}^d$, we define a likelihood model $r(z_{1:N} \mid \tau)$ on N data points, with z_n corresponding to the leaf vertex $n \in V_{\text{leaf}}$. We use a Gaussian random walk (GRW), where for each vertex $v \in V$ a location $z_v \mid z_{\pi(v)}$ is sampled according to a Gaussian distribution centered at its parent's location with variance equal to the branch length,

$$z_v \mid z_{\pi(v)} \sim \mathcal{N}(z_{\pi(v)}, (t_v - t_{\pi(v)})I), \quad v \in V \setminus \{v_{\text{root}}\},$$

and we take $z_{v_{\text{root}}} \sim \mathcal{N}(0, I)$. As a result of this choice, we can exploit the Gaussian graphical model structure to efficiently marginalize out the internal locations z_v associated with internal vertices $v \in V_{\text{int}}$ and evaluate the resulting marginal density $r(z_{1:N} \mid \tau)$. For details about this marginalization, please refer to subsection A.2.

The final overall density is written as

$$r(z_{1:N}, \tau) = \text{TMC}_N(\tau; a, b) r(z_{1:N} \mid \tau). \tag{A.3}$$

For further details and derivations related to the TMC, please refer to Boyles and Welling (2012).

A.1. TMC posterior predictive density

The TMC with N leaves and a GRW likelihood model can be a prior on a set of N hierarchically-structured data, i.e. data that correspond to nodes with small tree distance should have similar location values. In addition, it also acts as a density from which we can sample new data. The posterior predictive density $r(z_{N+1} \mid z_{1:N}, \tau)$ is easy to sample thanks to the exchangeability of the TMC.

To sample a new data point z_{N+1}, we select a branch (edge) and a time to attach a new leaf node. The probability $r(e_{N+1} \mid V, E)$ of selecting branch e_{N+1} is proportional to the probability under the TMC prior of the tree with a new leaf attached to branch e_{N+1}. The density $r(t_{N+1} \mid e_{N+1}, V, E)$ for a time label t_{N+1} is determined by the stick-breaking process. Consider inserting a node $N+1$ into the tree in between vertices u and v such that $t_v > t_u$, creating branch e_{N+1}. The inserted node has time t_{N+1} with probability according to the stick breaking process, i.e.

$$r(t_{N+1} \mid e_{N+1}, V, E) = \text{Beta}\left(\tfrac{t_v - t_{N+1}}{1 - t_{N+1}}; a, b\right) \text{Beta}\left(\tfrac{t_{N+1} - t_u}{1 - t_u}; a, b\right). \tag{A.4}$$

The new location z_{N+1} can be sampled from $r(z_{N+1} \mid e_{N+1}, t_{N+1}, \tau)$, which is the Gaussian distribution that comes out of the GRW likelihood model. Pictured in Figure 1 are samples from a TMC prior and GRW likelihood, where contours correspond to $r(z_{N+1} \mid z_{1:N}, \tau)$.

In addition to modeling hierarchical structure, the TMC is a flexible nonparametric density estimator.

A.2. TMC inference

The posterior distribution $r(\tau \mid z_{1:N})$ is analytically intractable due to the normalization constant $r(z_{1:N})$ involving a sum over all tree structures, but it can be approximately sampled via Markov chain Monte-Carlo (MCMC) methods. We utilize the Metropolis-Hastings algorithm with a subtree-prune-and-regraft (SPR) proposal distribution (Neal, 2003). An SPR proposal picks a subtree uniformly at random from τ and detaches it. It is then attached back on the tree to a branch and time picked uniformly at random. The Metropolis-Hastings acceptance probability is efficient to compute because the joint density $r(\tau, z_{1:N})$ can be evaluated using belief propagation to marginalize the latent values at internal nodes of τ, and many of the messages can be cached.

Recall that the TMC is a prior over phylogenies τ, and after attaching a Gaussian random walk (GRW), we obtain a distribution over N vectors in $\mathbb{R}^d$, corresponding to the leaves, $r(z_{1:N} \mid \tau)$. However, the GRW samples latent vectors at internal nodes $z_{V_{\text{int}}}$. Rather than explicitly representing these values, in this work we marginalize them out, i.e.

$$r(z_{1:N} \mid \tau) = \int r(z_{1:N} \mid z_{V_{\text{int}}}, \tau) p(z_{V_{\text{int}}} \mid \tau) dz_{V_{\text{int}}} \tag{A.5}$$

This marginalization process can be done efficiently, because our graphical model is tree-shaped and all nodes have Gaussian likelihoods. Belief propagation is a message-passing framework for marginalization and we utilize message-passing for several TMC inference queries. The main queries we are interested in are:

1. $r(z_{1:N}, \tau)$ - for the purposes of MCMC, we are interested in computing the joint likelihood of a set of observed leaf values and a phylogeny.

2. $r(z_n \mid z_{\setminus n}, \tau)$ - this query computes the posterior density over one leaf given all the others; we use this distribution when computing the posterior predictive density of a TMC.

3. $\nabla_{z_{\setminus n}} r(z_n \mid z_{\setminus n}, \tau)$ - this query is the gradient of the predictive density of a single leaf with respect to the values at all other leaves. This query is used when computing gradients of the ELBO w.r.t $s_{1:M}$ in the LORACs prior.

Message passing Message passing treats the tree as an undirected graph. We first pick start node v_{start} and request messages from each of v_{start}'s neighbors.

Message passing is thereafter defined recursively. When a node v has requested messages from a source node s, it thereafter requests messages from all its neighbors but s. The base case for this recursion is a leaf node v_n, which returns a message with the following contents:

$$\nu_n = \mathbf{0}; \quad \mu_n = z_n; \quad \log Z_n = 0; \quad \nabla_{\nu_n}(\nu) = \mathbf{1}; \quad \nabla_{\nu_n}(\mu) = \mathbf{0}; \quad \nabla_{\mu_n}(\mu) = \mathbf{1} \tag{A.6}$$

where bold numbers $\mathbf{0} \triangleq (0, \ldots, 0)^\top$ and $\mathbf{1} \triangleq (1, \ldots, 1)^\top$ denote vectors obtained by repeating a scalar d times.

In the recursive case, consider being at a node i and receiving a set of messages from its neighbors M.

$$\nu_i = \frac{1}{\sum_{m \in M} \frac{1}{\nu_m + e_{im}}}; \quad \mu_i = v_i \sum_{m \in M} \frac{\mu_m}{\nu_m + e_{im}} \tag{A.7}$$

where e_{im} is the length of the edge between nodes i and m. These messages are identical to those used in Boyles and Welling (2012).

Additionally, our messages include gradients w.r.t. *every* leaf node downstream of the message. We update each of these gradients when computing the new message and pass them along to the source node. Gradients with respect to one of these nodes j are calculated as

$$\begin{aligned} \nabla_{\nu_j}(\nu) &= \nabla_{\nu_j} \nu_i \\ \nabla_{\nu_j}(\mu) &= \nabla_{\nu_j} \mu_i \\ \nabla_{\mu_j}(\mu) &= \nabla_{\mu_j} \mu_i \end{aligned} \tag{A.8}$$

The most complicated message is the $\log Z_i$ message, which depends on the number of incoming messages. v_{start} gets three incoming messages, all other nodes get only two. Consider two messages from nodes v_k and v_l:

$$\begin{aligned} \Sigma_i &\triangleq (\nu_k + e_{ik} + \nu_l + e_{il}) I \\ \log Z_i &= -\frac{1}{2} \|\mu_k - \mu_l\|^2_{\Sigma_i} - \frac{1}{2} \left(\log |\Sigma_i|_d \log 2\pi \right) \end{aligned} \tag{A.9}$$

For three messages from nodes v_k, v_l, and v_m:

$$\begin{aligned} \Sigma_i &\triangleq \left((\nu_k + e_{ik})(\nu_l + e_{il}) + (\nu_l + e_{il})(\nu_m + e_{im}) + (\nu_m + e_{im})(\nu_k + e_{ik}) \right) I \\ \log Z_i &= -\frac{1}{2} \left((\nu_m + e_{im}) \|\mu_k - \mu_l\|^2_{\Sigma_i} + (\nu_k + e_{ik}) \|\mu_l - \mu_m\|^2_{\Sigma_i} + (\nu_l + e_{il}) \|\mu_m - \mu_k\|^2_{\Sigma_i} \right) - \frac{1}{2} \log |\Sigma_i| - \log 2\pi \end{aligned} \tag{A.10}$$

With these messages, we can answer all the aforementioned inference queries.

1. We can begin message passing at any internal node and compute: $\log r(z_{1:N}, \tau) = \sum_{v \in V} \log Z_v$
2. We start message passing at v_n. $r(z_n \mid z_{\setminus n}, \tau)$ is a Gaussian with mean μ_n and variance ν_n.
3. $\nabla_{z_{\setminus n}} r(z_n \mid z_{\setminus n}, \tau)$ is $\nabla_{z_{\setminus n}} \mathcal{N}(z_n \mid \mu_n, \nu_n I)$, which in turn utilizes gradients sent via message passing.

Implementation We chose to implement the TMC and message passing in Cython because we found raw Python to be too slow due to function call and type-checking overhead. Furthermore, we used diagonal rather than scalar variances in the message passing implementation to later support diagonal variances handed from the variational posterior over z_n.

Appendix B. Related work

LORACs connects various ideas in the literature, including Bayesian nonparametrics (Boyles and Welling, 2012), inducing-point approximations (e.g.; Snelson and Ghahramani, 2006; Tomczak and Welling, 2018), and amortized inference (Kingma and Welling, 2014; Rezende et al., 2014).

Also relevant is a recent thread of efforts to endow VAEs with the interpretability of graphical models (e.g.; Johnson et al., 2016; Lin et al., 2018). In this vein, Goyal et al. (2017) propose using a different Bayesian nonparametric tree prior, the nested Chinese restaurant process (CRP) (Griffiths et al., 2004), in a VAE. We chose to base LORACs on the TMC instead, as the posterior predictive distribution of an nCRP is a finite mixture, whereas the TMC's posterior predictive distribution has more complex continuous structure. Another distinction is that Goyal et al. (2017) only consider learning from pretrained image features, whereas our approach is completely unsupervised.

Appendix C. Inference in the LORACs prior

C.1. TMC-VAE

The choice of prior distribution in the VAE significantly affects the autoencoder and resulting latent space. The default standard normal prior, which takes $z_n \overset{\text{iid}}{\sim} \mathcal{N}(0, I)$, acts as a regularizer on an otherwise unconstrained autoencoder, but can be restrictive and result in overpruning (Burda et al., 2015). Extremely flexible, learnable distributions like masked autoregressive flow (MAF) priors (Papamakarios et al., 2017) enable very rich latent spaces, but don't encode any interpretable bias for organizing the latent space (except perhaps smoothness).

We explore the TMC prior for the VAE, which could potentially strike a sweet spot between restrictive and flexible priors. We generate the latent values $z_{1:N}$ of a VAE according to the TMC prior, then generate observations $x_{1:N}$ using a neural network observation model,

$$\tau \sim \mathrm{TMC}_N(\tau; a, b), \tag{C.11}$$

$$z_{1:N} \,|\, \tau \sim r(z_{1:N} \,|\, \tau), \qquad x_n \,|\, z_n \sim p_\theta(x_n \,|\, z_n). \tag{C.12}$$

The TMC-VAE is a coherent generative process that captures discrete, interpretable structure in the latent space. A phylogeny not only has an intuitive inductive bias, but can be useful for exploratory data analysis and introspecting the latent space itself.

Consider doing inference in this model: first assume variational distributions $q_\phi(z_n \,|\, x_n)$ (as in the VAE) and $q(\tau)$, which results in the ELBO,

$$\mathcal{L}[q] = \mathbb{E}_q \left[\log \frac{p(\tau, z_{1:N}, x_{1:N})}{q(\tau) \prod_n q(z_n \,|\, x_n)} \right] \tag{C.13}$$

For fixed $q_\phi(z_n \,|\, x_n)$, we can sample the optimal $q^*(\tau)$,

$$q^*(\tau) \propto \exp\{\mathbb{E}_q \left[\log p(\tau, z_{1:N}, x_{1:N})\right]\} \tag{C.14}$$

Because $p(z_{1:N} \mid \tau)$ is jointly Gaussian (factorizing according to tree structure) and $q_\phi(z_n \mid x_n)$ is Gaussian, expectations with respect to $z_{1:N}$ can move into $\log p(\tau, z_{1:N}, x_{1:N})$. This enables sampling the expected joint likelihood $\mathbb{E}_q[\log p(\tau, z_{1:N})]$ using SPR Metropolis-Hastings. However, optimizing this ELBO is problematic. $p(z_{1:N}|\tau)$ does not factorize independently, so computing unbiased gradient estimates from minibatches is impossible and requires evaluating all the data. Furthermore, the TMC is limiting from a computational perspective. Since a phylogeny has as many leaves as points in the dataset, belief propagation over internal nodes of the tree slows down linearly as the size of the dataset grows. In addition, SPR proposals mix very slowly for large trees. We found these limitations make the model impractical for datasets of more than 1000 examples.

C.2. LORACs prior

The LORACs prior involves first sampling a tree from the posterior distribution over TMCs with $s_{1:M}$ as leaves. We then sample a branch and time for each data z_n according to the posterior predictive distribution described in subsection 2.1. We then sample a z_n from the distribution induced by the GRW likelihood model. Finally, we pass the sampled z_n through the decoder.

$$\begin{aligned} \tau &\sim p(\tau; s_{1:M}) \\ e_n, t_n &\sim p(e_n, t_n|\tau) \\ z_n|e_n, t_n, \tau &\sim p(z_n|e_n, t_n, \tau; s_{1:M}) \triangleq r(s_{M+1} = z_n|e_n, t_n, \tau) \\ x_n|z_n &\sim p_\theta(x_n|z_n) \end{aligned} \tag{C.15}$$

Consider sampling the optimal $q^*(\tau; s_{1:M})$.

$$\begin{aligned} q^*(\tau; s_{1:M}) &\propto \exp\{\mathbb{E}_q[\log p(\tau, z_{1:N}, x_{1:N})]\} \\ &\propto \exp\{\log p(\tau; s_{1:M}) + \sum_n \mathbb{E}_q[p(z_n|e_n, t_n, \tau)]\} \\ &\propto \exp\{\log \mathrm{TMC}_N(\tau; a, b) + \sum_{m=1}^{M} \log r(s_m|s_{1:m-1}, \tau) \\ &\quad + \sum_n \mathbb{E}_q[\log p(z_n|e_n, t_n, \tau)]\} \end{aligned} \tag{C.16}$$

This term has a sum over N expectations; therefore computing this likelihood for the purpose of MCMC would involve passing the entire dataset through a neural network. Furthermore, the normalizer for this likelihood is intractable, but necessary for computing gradients w.r.t $s_{1:M}$. We therefore avoid using the optimal $q^*(\tau; s_{1:M})$ and set $q(\tau; s_{1:M})$ to the prior. This has the additional computational advantage of cancelling out the $\mathbb{E}_q[\log p(\tau)]$ term in the ELBO, which also has an intractable normalizing constant. If the inducing points are chosen so that they contain most of the information about the hierarchical organization of the dataset, then the approximation $p(\tau \mid z) \approx r(\tau \mid s_{1:M}) = p(\tau)$ will be reasonable. Thus, $q(\tau; s_{1:M})$ can be sampled using vanilla SPR Metropolis-Hastings, so samples from this distribution are readily available.

We use additional variational factors $q(e_n)$, $q_\xi(t_n|e_n, z_n; s_{1:M})$, and $q_\phi(z_n|x_n)$. $q_\xi(t_n|e_n, z_n; s_{1:M})$ is a recognition network that outputs the attach time for a particular branch. Since the $q(\tau; s_{1:M})$ and $p(\tau; s_{1:M})$ terms cancel out, we obtain the following ELBO.

$$\mathcal{L}[q] \triangleq \mathbb{E}_q \left[\log \frac{\prod_n p(e_n, t_n|\tau) p(z_n|e_n, t_n, \tau; s_{1:M}) p_\theta(x_n|z_n)}{\prod_n q(e_n) q_\xi(t_n|e_n, z_n; s_{1:M}) q_\phi(z_n|x_n)} \right] \tag{C.17}$$

For each data in the minibatch x_n, we pass it through the encoder to obtain $q(z_n|x_n)$. We then compute

$$q^*(e_n) = \exp\left\{ \mathbb{E}_q \left[\log p(e_n|t_n, z_n, \tau; s_{1:M}) \right] \right\} \tag{C.18}$$

This quantity is computed by looping over every branch b of a sample from $q(\tau)$, storing incoming messages at each node, passing the μ and ν and a sample from $q(z_n|x_n)$ into $q_\xi(t_n|e_n, s_{1:M}, z_n)$, outputting a logistic-normal distribution over times for that branch. We sample that logistic normal to obtain a time t to go with branch b. We can then compute the log-likelihood of z_n if it were to attach to b and t, using TMC inference query #2. This log-likelihood is added to the TMC prior log-probability of the branch being selected to obtain a joint probability $\mathbb{E}_q \left[\log p(e_n) p(t_n) p(z_n|e_n, t_n, \tau; s_{1:M}) \right]$ over the branch. After doing this for every branch, we normalize the joint likelihoods to obtain the optimal categorical distribution over every branch for z_n, $q^*(e_n)$. We then sample this distribution to obtain an attach location and time e_n, t_n for each data in the minibatch.

The next stage is to compute gradients w.r.t. to the learnable parameters of the model (θ, $s_{1:M}$, ϕ, and ξ). In the process of calculating $q^*(e_n)$, we have obtained samples from its corresponding $q_\xi(t_n|e_n, z_n, \tau; s_{1:M})$ and $q(z_n|x_n)$. We plug these into the ELBO and can compute gradients via automatic differentiation w.r.t. ϕ, θ, and ξ. Computing gradients w.r.t. $s_{1:M}$ is more tricky. We first examine the ELBO.

$$\mathcal{L}[q] = \mathbb{E}_q \left[\log \frac{\prod_n p(e_n|\tau) p(t_n) p(z_n|e_n, t_n, \tau; s_{1:M}) p_\theta(x_n|z_n)}{\prod_n q(e_n) q_\xi(t_n|e_n, z_n; s_{1:M}) q_\phi(z_n|x_n)} \right] \tag{C.19}$$

Consider the gradient of the ELBO with respect to $s_{1:M}$.

$$
\begin{aligned}
\nabla_{s_{1:M}} \mathcal{L}[q] &= \nabla_{s_{1:M}} \mathbb{E}_q \left[\log \frac{\prod_n p(e_n|\tau) p(t_n) p(z_n|e_n, t_n, \tau; s_{1:M}) p_\theta(x_n|z_n)}{\prod_n q(e_n) q_\xi(t_n|e_n, z_n; s_{1:M}) q_\phi(z_n|x_n)} \right] \\
&= \nabla_{s_{1:M}} \sum_\tau q(\tau; s_{1:M}) \mathbb{E}_q \left[\log \frac{\prod_n p(e_n|\tau) p(t_n) p(z_n|e_n, t_n, \tau; s_{1:M}) p_\theta(x_n|z_n)}{\prod_n q(e_n) q_\xi(t_n|e_n, z_n; s_{1:M}) q_\phi(z_n|x_n)} \right] \\
&= \sum_\tau q(\tau; s_{1:M}) \nabla_{s_{1:M}} \mathbb{E}_q \left[\log \frac{\prod_n p(e_n|\tau) p(t_n) p(z_n|e_n, t_n, \tau; s_{1:M}) p_\theta(x_n|z_n)}{\prod_n q(e_n) q_\xi(t_n|e_n, z_n; s_{1:M}) q_\phi(z_n|x_n)} \right] \\
&+ \sum_\tau \left(\nabla_{s_{1:M}} q(\tau; s_{1:M}) \right) \mathbb{E}_q \left[\log \frac{\prod_n p(e_n|\tau) p(t_n) p(z_n|e_n, t_n, \tau; s_{1:M}) p_\theta(x_n|z_n)}{\prod_n q(e_n) q_\xi(t_n|e_n, z_n; s_{1:M}) q_\phi(z_n|x_n)} \right] \\
&= \sum_\tau q(\tau; s_{1:M}) \nabla_{s_{1:M}} \mathbb{E}_q \left[\log \frac{\prod_n p(e_n|\tau) p(t_n) p(z_n|e_n, t_n, \tau; s_{1:M}) p_\theta(x_n|z_n)}{\prod_n q(e_n) q_\xi(t_n|e_n, z_n; s_{1:M}) q_\phi(z_n|x_n)} \right] \\
&+ \sum_\tau \left(q(\tau; s_{1:M}) \nabla_{s_{1:M}} \log q(\tau; s_{1:M}) \right) \mathbb{E}_q \left[\log \frac{\prod_n p(e_n|\tau) p(t_n) p(z_n|e_n, t_n, \tau; s_{1:M}) p_\theta(x_n|z_n)}{\prod_n q(e_n) q_\xi(t_n|e_n, z_n; s_{1:M}) q_\phi(z_n|x_n)} \right] \\
&= \mathbb{E}_{q(\tau)} \left[\nabla_{s_{1:M}} \mathbb{E}_q \left[\log \frac{\prod_n p(e_n|\tau) p(t_n) p(z_n|e_n, t_n, \tau; s_{1:M}) p_\theta(x_n|z_n)}{\prod_n q(e_n) q_\xi(t_n|e_n, z_n; s_{1:M}) q_\phi(z_n|x_n)} \right] \right. \\
&\left. + \nabla_{s_{1:M}} \log q(\tau; s_{1:M}) \mathbb{E}_q \left[\log \frac{\prod_n p(e_n|\tau) p(t_n) p(z_n|e_n, t_n, \tau; s_{1:M}) p_\theta(x_n|z_n)}{\prod_n q(e_n) q_\xi(t_n|e_n, z_n; s_{1:M}) q_\phi(z_n|x_n)} \right] \right] \\
&= \mathbb{E}_q \left[\nabla_{s_{1:M}} \left(-\log q(e_n) - \log q(t_n \,|\, z_n, e_n, \tau; s_{1:M}) + \log p(z_n \,|\, e_n, t_n, \tau; s_{1:M}) \right) \right] \\
&+ \mathbb{E}_q \left[\nabla_{s_{1:M}} \left(\log q(\tau) + \log q(e_n) \right) \log \frac{p(e_n|\tau)}{q(e_n)} \frac{p(z_n \,|\, z_n, e_n, t_n, \tau; s_{1:M})}{q(t_n \,|\, z_n, e_n, \tau; s_{1:M})} \right]
\end{aligned}
\tag{C.20}
$$

In the last step, we expand out expectation over e_n and then pass the derivative through like we did for τ. The gradients w.r.t. $q(e_n)$ are zero, since $q^*(e_n)$ is a partial optimum of the ELBO and we are left with:.

$$
\begin{aligned}
\nabla_{s_{1:M}} \mathcal{L}[q] &= \mathbb{E}_q \left[\nabla_{s_{1:M}} \log p(z_N|e_n, t_n, \tau; s_{1:M}) \right] - \mathbb{E}_q \left[\log q(t_n \,|\, z_n, e_n, \tau; s_{1:M}) \right] \\
&+ \mathbb{E}_q \left[\nabla_{s_{1:M}} \log q(\tau; s_{1:M}) \log \frac{p(e_n|\tau)}{q(e_n)} \frac{p(z_n \,|\, z_n, e_n, t_n, \tau; s_{1:M})}{q(t_n \,|\, z_n, e_n, \tau; s_{1:M})} \right]
\end{aligned}
\tag{C.21}
$$

The first term of the gradient is the expected gradient of the posterior predictive density w.r.t $s_{1:M}$. This can be calculated by using TMC inference query #3 using samples from $q(e_n)$ and $q(t_n \,|\, z_n, e_n, \tau; s_{1:M})$. The second term also uses the same gradients, by means of the chain rule to differentiate through the time-amortization network. The third term of this gradient is a score function gradient, which we decide to not use due to the high-variance nature of score function gradients. We found that we were able to obtain strong results even with biased gradients.

Appendix D. Detailed results

D.1. Experimental setup

We evaluated the LORACs prior on three separate datasets: dynamically binarized MNIST (LeCun and Cortes, 2010), Omniglot (Lake et al., 2015), and CelebA (Liu et al., 2015). For

all three experiments, we utilized convolutional/deconvolutional encoders/decoders and a 40-dimensional latent space. We used 200, 1000, and 500 inducing points for MNIST, Omniglot, and CelebA respectively with TMC parameters $a = b = 2$. $q_\xi(t_n \mid e_n, z_n; s_{1:M})$ was a two-layer 500-wide neural network with ReLU activations that output parameters of a logistic-normal distribution over stick size and all parameters were optimized with Adam (Kingma and Ba, 2015).

We implemented the LORACs prior in Tensorflow and Cython. For MNIST and Omniglot, our architectures are in Table D.1 and CelebA is in Table D.2.

(*a*) Encoder

Layer type	Shape
Conv + ReLU	[3, 3, 64], stride 2
Conv + ReLU	[3, 3, 32], stride 1
Conv + ReLU	[3, 3, 16], stride 2
FC + ReLU	512
Gaussian	40

(*b*) Decoder

Layer type	Shape
FC + ReLU	3136
Deconv + ReLU	[3, 3, 32], stride 2
Deconv + ReLU	[3, 3, 32], stride 1
Deconv + ReLU	[3, 3, 1], stride 2
Bernoulli	

Table D.1: Network architectures for MNIST and Omniglot

(*a*) Encoder

Layer type	Shape
Conv + ReLU	[3, 3, 64], stride 2
Conv + ReLU	[3, 3, 32], stride 1
Conv + ReLU	[3, 3, 16], stride 2
FC + ReLU	512
Gaussian	40

(*b*) Decoder

Layer type	Shape
FC + ReLU	4096
Deconv + ReLU	[3, 3, 32], stride 2
Deconv + ReLU	[3, 3, 32], stride 1
Deconv + ReLU	[3, 3, 3], stride 2
Bernoulli	

Table D.2: Network architectures for CelebA

In general, we trained the model interleaving one gradient step with 100 sampling steps for $q(\tau; s_{1:M})$. We also found that experimenting with values of a and b in the TMC prior did not impact results significantly. We initialized the networks with weights from a VAE trained for 100 epochs and inducing points were initialized using k-means. All parameters were trained using Adam (Kingma and Ba, 2015) with a 10^{-3} learning rate for an 100 epochs with learning rate decay to 10^{-5} for the last 20 epochs. Finally, we initialized trees with all node times close to 0, to emulate a VAE prior.

D.1.1. Baseline details

All baselines were trained with the default architecture. They were trained for 400 epochs, with KL warmup (β started at 10^{-2}, and ramped up to $\beta = 1$ linearly over 50 epochs). They were trained using Adam with a learning rate of 10^{-3}, with a learning rate of 10^{-5} for the last 80 epochs.

DVAE# was trained using the default implementation from https://github.com/QuadrantAI/dvae, which is hierarchical VAE consisting of two Bernoulli latent variables,

Figure D.6: Learned inducing points for a LORACs(200) prior on MNIST.

200-dimensional each. Each is learned via a feed-forward neural network 4-layers deep. The default DVAE# implementation also uses statically binarized MNIST where we use dynamically binarized.

D.2. Qualitative results

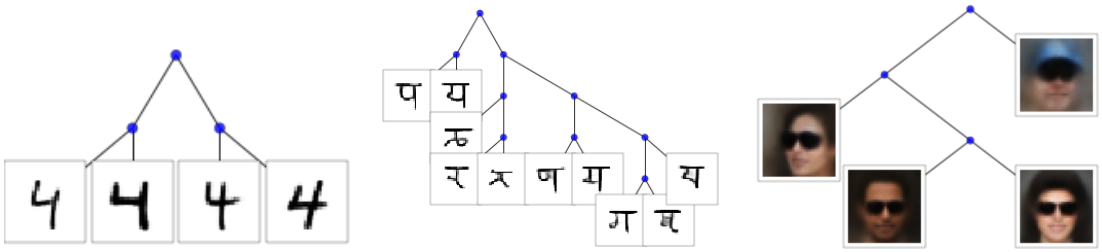
A hierarchical clustering in the latent space offers a unique opportunity for interpretability and exploratory data analysis, especially when the data are images. Here are some methods for users to obtain useful data summaries and explore a dataset.

Visualizing inducing points We first inspect the learned inducing points $s_{1:M}$ by passing them through the decoder. Visualized in Figure D.6 are the 200 learned inducing points for MNIST. The inducing points are all unique and are cleaner than pseudo-input reconstructions from VampPrior (shown in Figure D.16). Inducing points can help summarize a dataset, as visualizations of the latent space indicate they spread out and cover the data (see Figure D.14). Inducing points are also visually unique and sensible in Omniglot and CelebA (see Figure D.17 and D.18).

Hierarchical clustering We can sample $q(\tau; s_{1:M})$ to obtain phylogenies over the inducing points, and can visualize these clusterings using the decoded inducing points; subtrees from a sample in each dataset are visualized in Figure D.7. In MNIST, we find large subtrees correspond to the discrete classes in the dataset. In Omniglot, subtrees sometimes correspond to language groups and letter shapes. In CelebA, we find subtrees sometimes correspond to pose or hair color and style.

We can further use the time at each internal node to summarize the data at many levels of granularity. Consider "slicing" the hierarchy at a particular time t by taking every branch $(\pi(v), v) \in E$ with $t_{\pi(v)} \leq t < t_v$ and computing the corresponding expected Gaussian random walk value at time t. At times closer to zero, we slice fewer branches and are closer to the root of the hierarchy, so the value at the slice looks more like the mean of the data. In Figure D.8, we visualize this process over a subset of the inducing points of CelebA. Visualizing the dataset in this way reveals cluster structure at different granularities and offers an evolutionary interpretation to the data, as leaves that coalesce more "recently" are likely to be closer in the latent space.

Although the hierarchical clustering is only over inducing points, we can still visualize where real data belong on the hierarchy by computing $q^*(e_n)$ and attaching the data to the



(*a*) MNIST (*b*) Omniglot (*c*) CelebA

Figure D.7: An example learned subtree from a sample of $q(\tau; s_{1:M})$ for each dataset. Leaves are visualized by passing inducing points through the decoder.

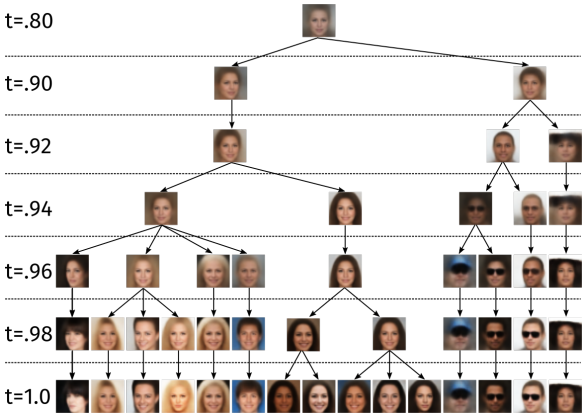
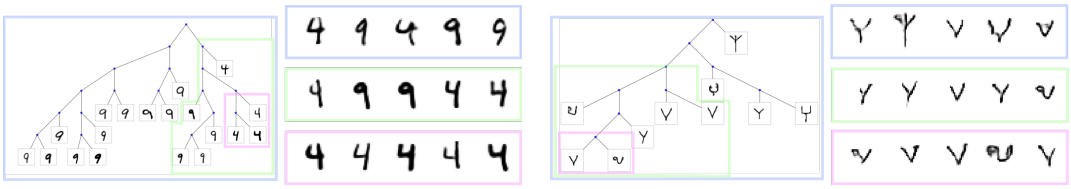


Figure D.8: The evolution of a CelebA over a subset of inducing points. We create this visualization by taking slices of the tree at particular times and looking at the latent distribution at each of the sliced branches.

tree. By doing this for many points of data, and removing the inducing points from the tree, we obtain an induced hierarchical clustering.

Generating samples Having fit a generative model to our data, we can visualize samples from the model. Although we do not expect the samples to have fidelity and sharpness comparable to those from GANs or state-of-the-art decoding networks (Radford et al., 2015; Salimans et al., 2017), sampling with the LORACs prior can help us understand the latent space. To draw a sample from a TMC's posterior predictive density, we first sample a branch and time, assigning the sample a place in the tree. This provides each generated sample a *context*, i.e., the branch and subtree it was generated from. However, learning a LORACs prior allows us to conditionally sample in a novel way. By restricting samples to a subtree, we can generate samples from the support of the posterior predictive density limited to that subtree. This enables conditional sampling at many levels of the hierarchy. We visualize examples of this in Figure D.9 and Figure D.10.



(*a*) MNIST (*b*) Omniglot

Figure D.9: Conditional samples from subtrees.

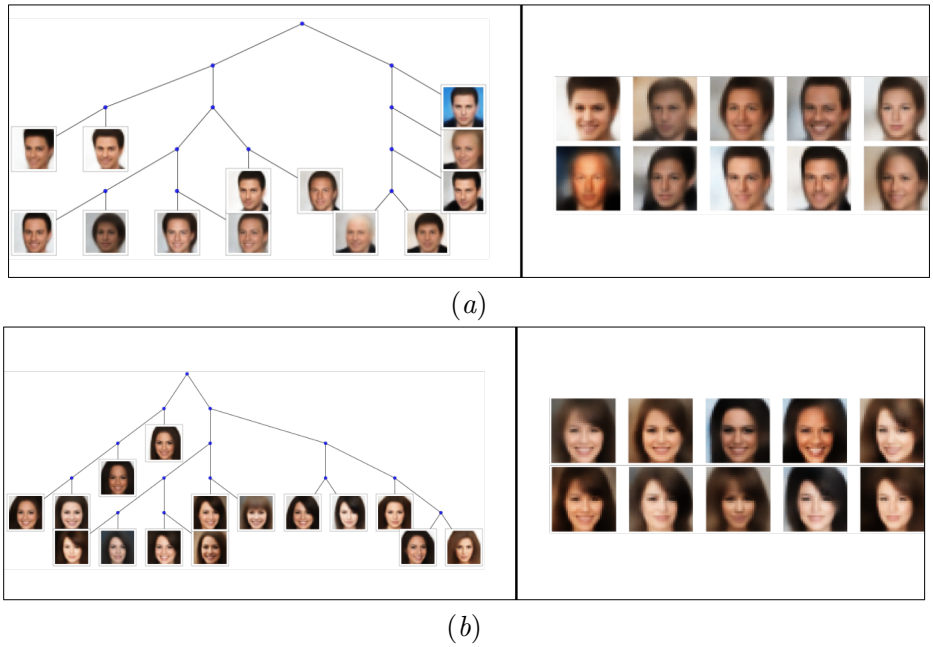


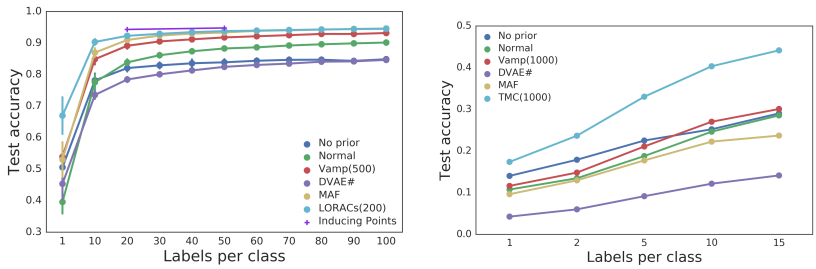
Figure D.10: Samples from subtrees of CelebA.

D.3. Quantitative results

We ran experiments designed to evaluate the usefulness of the LORACs's learned latent space for downstream tasks. We compare the LORACs prior against a set of baseline priors on three different tasks: few-shot classification, information retrieval, and generative modeling. Our datasets are dynamically binarized MNIST and Omniglot (split by instance) and our baselines are representations learned with the same encoder-decoder architecture and latent dimensionality[2] but substituting the following prior distributions over z:

- No prior
- Standard normal prior
- VampPrior (Tomczak and Welling, 2018) - 500 pseudo-inputs for MNIST, 1000 for Omniglot

2. Following the defaults in the author's reference implementation, we evaluated DVAE# on statically binarized MNIST with smaller neural networks, but with a higher-dimensional latent space.



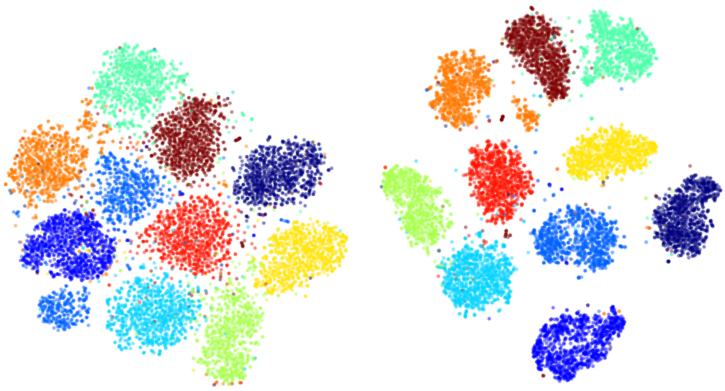
(*a*) MNIST (*b*) Omniglot

Figure D.11: Few-shot classification results

- DVAE♯ (Vahdat et al., 2018) - latent vectors are 400-dimensional, formed from concatenating binary latents, encoder and decoder are two-layer feed-forward networks with ReLU nonlinearities
- Masked autoregressive flow (MAF; Papamakarios et al., 2017) - two layer, 512 wide MADE

Few-shot classification In this task, we train a classifier with varying numbers of labels and measure test accuracy. We pick equal numbers of labels per class to avoid imbalance and we use a logistic regression classifier trained to convergence to avoid adding unnecessary degrees of freedom to the experiment. We replicated the experiment across 20 randomly chosen label sets for MNIST and 5 for Omniglot. The test accuracy on these datasets is visualized in Figure D.11. For MNIST, we also manually labeled inducing points and found that training a classifier on 200 and 500 inducing points achieved significantly better test accuracy than randomly chosen labeled points, hinting that the LORACs prior has utility in an active learning setting.

The representations learned with the LORACs consistently achieve better accuracy, though in MNIST, LORACs prior and MAF reach very similar test accuracy at 100 labels per class. The advantage of the LORACs prior is especially clear in Omniglot (Table D.5 and Table D.6 contain the exact numbers). We believe our advantage in this task comes from ability of the LORACs prior to model discrete structure. TSNE visualizations in Figure D.12 and Figure D.13 indicate clusters are more concentrated and separated with the LORACs prior than with other priors, though TSNE visualizations should be taken with a grain of salt.



(*a*) Normal prior (*b*) LORACs(200) prior

Figure D.12: TSNE visualizations of the latent space of the MNIST test set with different priors, color-coded according to class. LORACs prior appears to learn a space with more separated, concentrated clusters.

Information retrieval We evaluated the meaningfulness of Euclidean distances in the learned latent space by measuring precision-recall when querying the test set. We take each element of the test set and sort all other members according to their L_2 distance in the latent space. From this ranking, we produce precision-recall curves for each of the query and plot the average precision-recall over the entire test set in Figure D.19. We also report the area-under-the-curve (AUC) measure for each of these curves in Table D.3.

AUC numbers for Omniglot are low across the board because of the large number of classes and low number of instances per class. However, in both datasets the LORACs prior consistently achieves the highest AUC, especially with MNIST. The LORACs prior encourages tree-distance to correspond to squared Euclidean distance, as branch lengths in the tree are variances in a Gaussian likelihoods. We thus suspect distances in a LORACs prior latent space to be more informative and better for information retrieval.

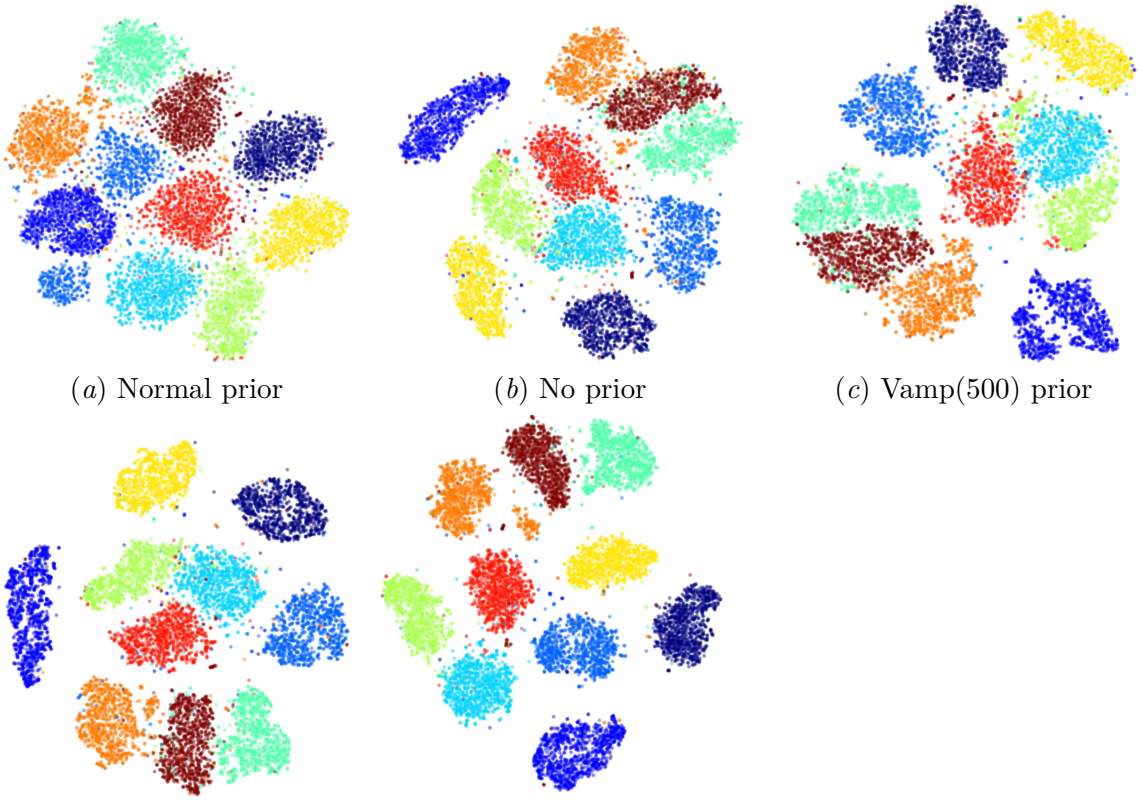
Prior	MNIST	Omniglot
No prior	0.429	0.078
Normal	0.317	0.057
VAMP	0.502	0.063
DVAE#	0.490	0.024
MAF	0.398	0.070
LORACs	**0.626**	**0.087**

Table D.3: Averaged precision-recall AUC on MNIST/Omniglot test datasets

Prior	MNIST	Omniglot
Normal	-83.789	-89.722
MAF	**-80.121**	**-86.298**
Vamp	-83.0135	-87.604
LORACs	-83.401	-87.105

Table D.4: MNIST/Omniglot test log-likelihoods

Held-out log-likelihood We estimate held-out log-likelihoods for the four VAEs we trained with comparable architectures and different priors. (We exclude DVAE♯ since its architecture is substantially different, and the classical autoencoder since it lacks generative semantics.) We use 1000 importance-weighted samples (Burda et al., 2015) to estimate held-out log-likelihood, and report the results in Table D.4. We find that, although LORACs outperforms the other priors on downstream tasks, it only achieves middling likelihood numbers. This result is consistent with the findings of Chang et al. (2009) that held-out log-likelihood is not necessarily correlated with interpretability or usefulness for downstream tasks.



(*a*) Normal prior (*b*) No prior (*c*) Vamp(500) prior

(*d*) MAF prior (*e*) LORACs(200) prior

Figure D.13: TSNE visualizations of the latent space of the MNIST test set with various prior distributions, color-coded according to class.

Figure D.14: A TSNE visualization of the latent space for the TMC(200) model with inducing points and one sample from $q(\tau; s_{1:M})$ plotted. Internal nodes are visualized by computing their expected posterior values, and branches are plotted in 2-d space.

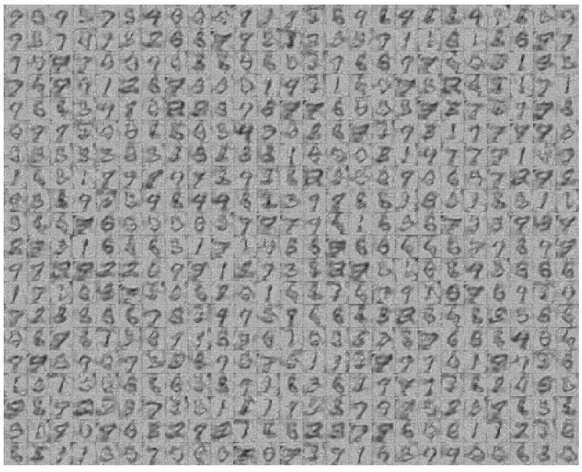


Figure D.15: MNIST VampPrior learned pseudo-inputs.

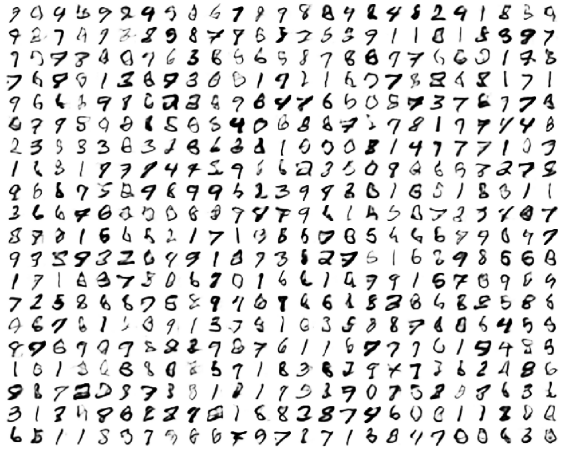


Figure D.16: MNIST VampPrior reconstructed pseudo-inputs obtained by deterministically encoding and decoding each pseudo-input.

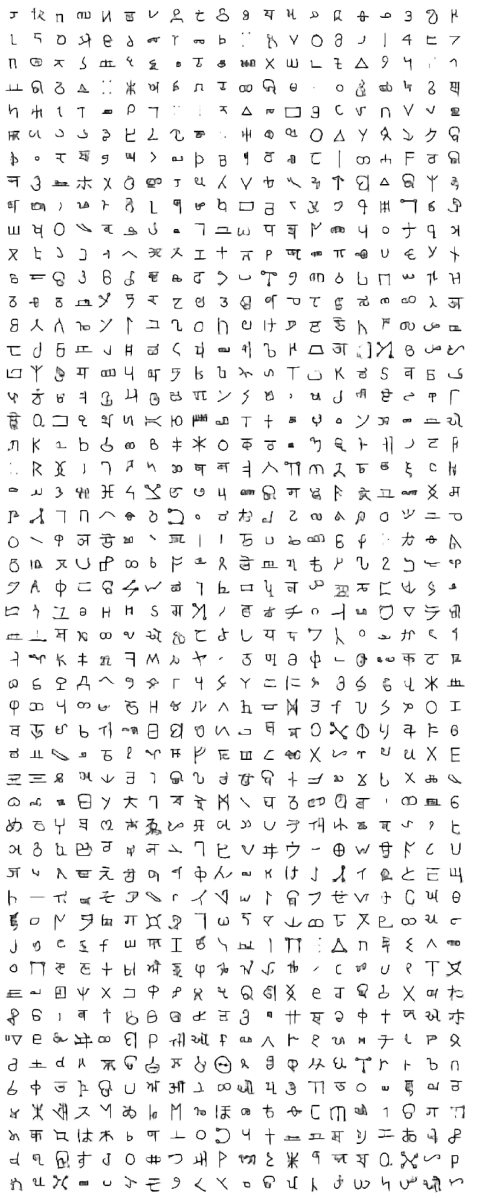


Figure D.17: Omniglot learned inducing points.

Figure D.18: CelebA learned inducing points.

Labels per class	1	10	20	30	40	50	60	70	80	90	100
No prior	0.506 ± 0.095	0.781 ± 0.045	0.820 ± 0.023	0.829 ± 0.020	0.836 ± 0.026	0.839 ± 0.021	0.844 ± 0.017	0.846 ± 0.017	0.847 ± 0.015	0.843 ± 0.015	0.848 ± 0.014
Normal	0.396 ± 0.076	0.775 ± 0.051	0.838 ± 0.020	0.861 ± 0.016	0.874 ± 0.011	0.883 ± 0.011	0.886 ± 0.011	0.892 ± 0.010	0.896 ± 0.010	0.899 ± 0.011	0.901 ± 0.008
Vamp(500)	0.539 ± 0.094	0.849 ± 0.035	0.891 ± 0.019	0.905 ± 0.013	0.911 ± 0.016	0.918 ± 0.012	0.921 ± 0.009	0.925 ± 0.008	0.929 ± 0.007	0.928 ± 0.005	0.932 ± 0.005
DVAE#	0.453 ± 0.101	0.735 ± 0.027	0.784 ± 0.017	0.801 ± 0.012	0.813 ± 0.013	0.824 ± 0.014	0.830 ± 0.012	0.835 ± 0.011	0.841 ± 0.007	0.842 ± 0.007	0.846 ± 0.008
MAF	0.530 ± 0.113	0.869 ± 0.029	0.910 ± 0.012	0.923 ± 0.012	0.930 ± 0.007	0.933 ± 0.010	0.938 ± 0.008	0.940 ± 0.008	0.942 ± 0.006	0.944 ± 0.006	$\mathbf{0.946 \pm 0.005}$
LORACs(200)	$\mathbf{0.670 \pm 0.120}$	$\mathbf{0.903 \pm 0.019}$	$\mathbf{0.923 \pm 0.011}$	$\mathbf{0.929 \pm 0.009}$	$\mathbf{0.934 \pm 0.006}$	$\mathbf{0.938 \pm 0.004}$	$\mathbf{0.939 \pm 0.005}$	$\mathbf{0.941 \pm 0.004}$	$\mathbf{0.943 \pm 0.004}$	$\mathbf{0.944 \pm 0.003}$	0.945 ± 0.003

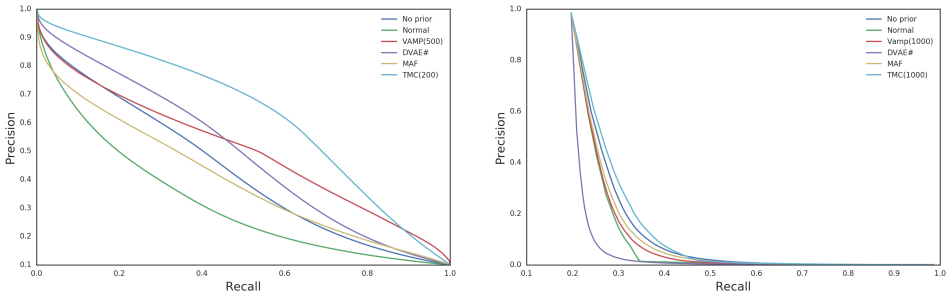
Table D.5: MNIST few-shot classification results.

Labels per class	1	2	5	10	15
No prior	0.140 ± 0.012	0.179 ± 0.008	0.225 ± 0.006	0.252 ± 0.009	0.290 ± 0.001
Normal	0.107 ± 0.007	0.134 ± 0.010	0.187 ± 0.008	0.246 ± 0.006	0.285 ± 0.000
Vamp(1000)	0.116 ± 0.011	0.148 ± 0.009	0.210 ± 0.003	0.270 ± 0.005	0.300 ± 0.000
DVAE#	0.042 ± 0.004	0.060 ± 0.006	0.091 ± 0.003	0.121 ± 0.001	0.141 ± 0.000
MAF	0.096 ± 0.008	0.129 ± 0.006	0.177 ± 0.010	0.222 ± 0.007	0.237 ± 0.002
LORACs(1000)	$\mathbf{0.173 \pm 0.005}$	$\mathbf{0.236 \pm 0.005}$	$\mathbf{0.330 \pm 0.008}$	$\mathbf{0.403 \pm 0.006}$	$\mathbf{0.441 \pm 0.000}$

Table D.6: Omniglot few-shot classification results.

# of inducing points	200	500
	0.9428	0.9474

Table D.7: MNIST few-shot classification with labeled inducing points.



(a) MNIST (b) Omniglot

Figure D.19: Averaged precision-recall curves over test datasets.