Tree Variational Autoencoders

Laura Manduchi [*1] **Moritz Vandenhirtz** [*1] **Alain Ryser** [1] **Julia Vogt** [1]

Abstract

We propose a new generative hierarchical clustering model that learns a flexible tree-based posterior distribution over latent variables. The proposed *Tree Variational Autoencoder* (TreeVAE) hierarchically divides samples according to their intrinsic characteristics, shedding light on hidden structures in the data. It adapts its architecture to discover the optimal tree for encoding dependencies between latent variables, improving generative performance. We show that TreeVAE uncovers underlying clusters in the data and finds meaningful hierarchical relations between the different groups on several datasets. Due to its generative nature, TreeVAE can generate new samples from the discovered clusters via conditional sampling.

1. Introduction

Discovering structure and hierarchies in the data has been a long-standing goal in machine learning (Jordan & Mitchell, 2015; Bishop, 2006; Bengio et al., 2012). Our work advances the state-of-the-art in structured VAEs by combining the complementary strengths of hierarchical clustering and deep generative models. We propose TreeVAE, a novel tree-based generative model that encodes *hierarchical* dependencies between latent variables. We propose a training procedure to learn the optimal tree structure to model the posterior distribution of latent variables. An example of a tree learned by TreeVAE is depicted in Fig. 1. Each edge and each split are encoded by neural networks, while the circles depict latent variables. Each sample is associated with a probability distribution over *paths*. The resulting tree thus organizes the data into an interpretable hierarchical structure in an unsupervised fashion, optimizing the amount of shared information between samples. In CIFAR-10, for example, the method divides the vehicles and animals into two different subtrees and similar groups (such as planes and ships) share common ancestors.

Our main contributions are as follows: (*i*) We propose a novel, deep probabilistic approach to hierarchical clustering that learns the optimal generative binary tree to mimic the hierarchies present in the data. (*ii*) We provide a thorough empirical assessment of the proposed approach. In particular, we show that (a) TreeVAE achieves superior hierarchical clustering performance compared to related work on deep hierarchical clustering, (b) it discovers meaningful patterns in the data and their hierarchical relationships, and (c) TreeVAE achieves a more competitive log-likelihood lower bound compared to VAE and LadderVAE, its non-hierarchical and sequential counterparts, respectively.

2. TreeVAE

We propose TreeVAE, a novel deep generative model that learns a flexible tree-based posterior distribution over latent variables. Each sample travels through the tree from root to leaf in a probabilistic manner as TreeVAE learns sample-specific probability distributions of paths. As a result, the data is divided in a hierarchical fashion, with more refined concepts for deeper nodes in the tree. The proposed graphical model is depicted in Fig. 2. The inference and generative model share the same top-down tree structure, permitting interaction between the bottom-up and top-down architecture, similar to Sønderby et al. (2016).

2.1. Model Formulation

Given H, the maximum depth of the tree, and given a dataset $\boldsymbol{X}$, the model is defined by three components that are learned during training:

- the *global* structure of the binary tree $\mathcal{T}$, which specifies the set of nodes $\mathbb{V} = \{0, \dots, V\}$, the set of leaves $\mathbb{L}$, where $\mathbb{L} \subset \mathbb{V}$, and the set of edges $\mathcal{E}$.
- the *sample-specific* latent embeddings $\mathbf{z} = \{\mathbf{z}_0, \dots, \mathbf{z}_V\}$, which are random variables assigned to each node.
- the *sample-specific* decisions $\mathrm{c} = \{\mathrm{c}_0, \dots, \mathrm{c}_{V-|\mathbb{L}|}\}$, which are Bernoulli random variables indicating the probability of going to the left child of the underlying node. They take values $c_i \in \{0, 1\}$ for $i \in \mathbb{V} \setminus \mathbb{L}$, with

[*]Equal contribution [1]Department of Computer Science, ETH Zurich, Zurich, Switzerland. Correspondence to: Laura Manduchi <laura.manduchi@inf.ethz.ch>, Moritz Vandenhirtz <moritz.vandenhirtz@inf.ethz.ch>.

Accepted to ICML workshop on Structured Probabilistic Inference & Generative Modeling, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

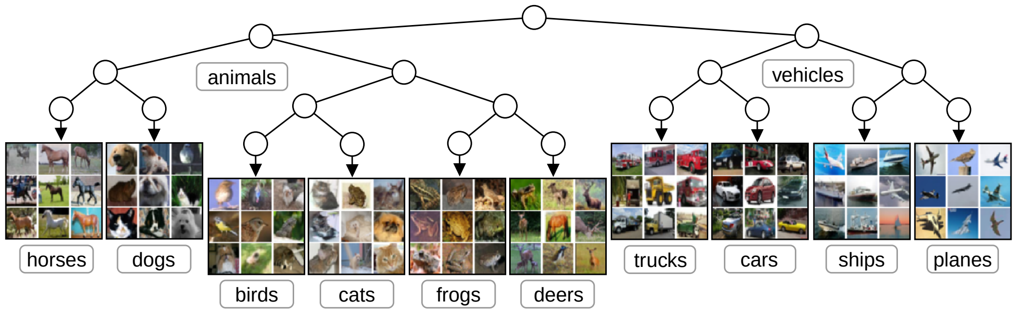

Figure 1: The hierarchical structure discovered by TreeVAE on the CIFAR-10 dataset. We display random subsets of images that are probabilistically assigned to each leaf of the tree.

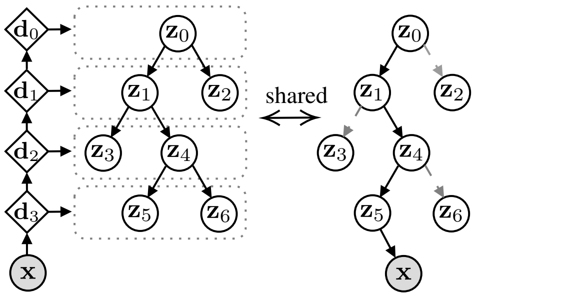

Figure 2: The proposed inference (left) and generative (right) model for TreeVAE. Circles are stochastic variables while diamonds are deterministic. The global topology of the tree is learned during training.

$c_i = 0$ if the left child is selected. A decision path $\mathcal{P}_l$ indicates the path from root 0 to leaf l, e.g., in Fig. 2 $\mathcal{P}_5 = \{0, 1, 4, 5\}$.

The tree structure is shared across the entire dataset and is learned iteratively by growing the tree node-wise. The latent embeddings and the decision paths, on the other hand, are learned using variational inference by conditioning the model on the current tree structure.

2.2. Generative Model

The generative process of TreeVAE for a given $\mathcal{T}$ is depicted in Fig. 2 (right). The generation of a new sample $\boldsymbol{x}$ starts from the root. First, the latent embedding of the root node $\mathbf{z}_0$ is sampled from a standard Gaussian $p_\theta(\boldsymbol{z}_0) = \mathcal{N}(\boldsymbol{z}_0 \mid \mathbf{0}, \boldsymbol{I})$. Then, given the sampled $\boldsymbol{z}_0$, the decision of going to the left or the right node is sampled from a Bernoulli distribution $p(\mathrm{c}_0 \mid \boldsymbol{z}_0) = Ber(r_{p,0}(\boldsymbol{z}_0))$, where $\{r_{p\ ,i} \mid i \in \mathbb{V} \setminus \mathbb{L}\}$ are functions parametrized by neural networks defined as *routers*, and cause the splits in Fig. 2. The subscript p is used to indicate the parameters of the generative model. The latent embedding of the selected child, let us assume it is $\boldsymbol{z}_1$, is then sampled from a Gaussian distribution $p_\theta(\boldsymbol{z}_1 \mid \boldsymbol{z}_0) = \mathcal{N}\left(\mathbf{z}_1 \mid \mu_{p,1}(\boldsymbol{z}_0), \sigma^2_{p,1}(\boldsymbol{z}_0)\right)$, where $\{\mu_{p,i}, \sigma_{p,i} \mid i \in \mathbb{V} \setminus \{0\}\}$ are functions parametrized by neural networks defined as *transformations*. They are indicated by the top-down arrows in Fig. 2. This process continues until a leaf is reached.

Let us define the set of latent variables selected by the path $\mathcal{P}_l$, which goes from the root to the leaf l, as $\mathbf{z}_{\mathcal{P}_l} = \{\mathbf{z}_i \mid i \in \mathcal{P}_l\}$, the parent node of the node i as $pa(i)$, and $p(c_{pa(i)\to i} \mid \boldsymbol{z}_{pa(i)})$ the probability of going from $pa(i)$ to i. Note that the path $\mathcal{P}_l$ defines the sequence of decisions. The prior probability of the latent embeddings and the path given the tree $\mathcal{T}$ can be summarized as

$$p_\theta(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l) = p(\boldsymbol{z}_0) \prod_{i \in \mathcal{P}_l \setminus \{0\}} p(c_{pa(i)\to i} \mid \boldsymbol{z}_{pa(i)}) p(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}). \quad (1)$$

Finally, $\boldsymbol{x}$ is sampled from a distribution that is conditioned on the selected leaf l:

$$p_\theta(\boldsymbol{x} \mid \boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l) = \mathcal{N}\left(\mathbf{x} \mid \mu_{x,l}(\boldsymbol{z}_l), \sigma^2_{x,l}(\boldsymbol{z}_l)\right), \quad (2)$$

where $\mathbf{z}_{\mathcal{P}_l}$ is the set of latent variables selected by the path and $\{\mu_{x,l}, \sigma_{x,l} \mid l \in \mathbb{L}\}$ are functions parametrized by leaf-specific neural networks defined as *decoders*.

2.3. Inference Model

The inference model is described by the variational posterior distribution of both the latent embeddings and the paths. It follows a similar structure as in the prior probability defined in (1), with the difference that the probability of the root and of the decisions are now conditioned on the sample $\boldsymbol{x}$:

$$q(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l \mid \boldsymbol{x}) = q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i \in \mathcal{P}_l \setminus \{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}). \quad (3)$$

To compute the variational probability of the latent embeddings $q(\boldsymbol{z}_0 \mid \boldsymbol{x})$ and $q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)})$, where

$$q(\boldsymbol{z}_0 \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}_0 \mid \mu_{q,0}(\boldsymbol{x}), \sigma^2_{q,0}(\boldsymbol{x})\right) \quad (4)$$

$$q_\phi\left(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}\right) = \mathcal{N}\left(\mathbf{z}_i \mid \mu_{q,i}\left(\boldsymbol{z}_{pa(i)}\right), \sigma^2_{q,i}\left(\boldsymbol{z}_{pa(i))}\right)\right)\right) \forall i \in \mathcal{P}_l, \quad (5)$$

we follow a similar approach to the one proposed by Sønderby et al. (2016). Note that we use the subscript q to indicate the parameters of the inference model.
First, a deterministic bottom-up pass computes the node-specific approximate likelihood contributions

$$\mathbf{d}_h = \text{MLP}\left(\mathbf{d}_{h+1}\right) \tag{6}$$
$$\hat{\boldsymbol{\mu}}_{q,i} = \text{Linear}\left(\mathbf{d}_{depth(i)}\right), i \in \mathbb{V} \tag{7}$$
$$\hat{\boldsymbol{\sigma}}^2_{q,i} = \text{Softplus}\left(\text{Linear}\left(\mathbf{d}_{depth(i)}\right)\right), i \in \mathbb{V}, \tag{8}$$

where $\mathbf{d}_H$ is parametrized by a domain-specific neural network defined as *encoder*, and $\text{MLP}(\mathbf{d}_h)$ for $h \in \{1, \ldots, H\}$, indicated by the bottom-up arrows in Fig. 2, are neural networks, shared among the parameter predictors, $\hat{\mu}_{q,i}, \hat{\sigma}^2_{q,i}$, of the same depth. They are characterized by the same architecture as the *transformations* defined in Sec.2.2. A stochastic downward pass then recursively computes the approximate posteriors defined as

$$\boldsymbol{\sigma}_{q,i} = \frac{1}{\hat{\boldsymbol{\sigma}}^{-2}_{q,i} + \boldsymbol{\sigma}^{-2}_{p,i}}, \quad \boldsymbol{\mu}_{q,i} = \frac{\hat{\boldsymbol{\mu}}_{q,i}\hat{\boldsymbol{\sigma}}^{-2}_{q,i} + \boldsymbol{\mu}_{p,i}\boldsymbol{\sigma}^{-2}_{p,i}}{\hat{\boldsymbol{\sigma}}^{-2}_{q,i} + \boldsymbol{\sigma}^{-2}_{p,i}}, \tag{9}$$

where all operations are performed elementwise. Finally, the variational distributions of the decisions $q(\mathrm{c}_i \mid \boldsymbol{x})$ are defined as

$$q(\mathrm{c}_i \mid \boldsymbol{x}) = q(\mathrm{c}_i \mid \mathbf{d}_{\text{depth(i)}}) = Ber(r_{q,i}(\mathbf{d}_{\text{depth(i)}})), \tag{10}$$

where $\{r_{q,i} \mid i \in \mathbb{V} \setminus \mathbb{L}\}$ are functions parametrized by neural networks and are characterized by the same architecture as the *routers* of the generative model defined in Sec.2.2.

2.4. Evidence Lower Bound

The likelihood of the data conditioned on the tree structure $\mathcal{T}$ is defined as

$$p(\boldsymbol{x} \mid \mathcal{T}) = \sum_{l \in \mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} p(\boldsymbol{x}, \boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l) \tag{11}$$
$$= \sum_{l \in \mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} p_\theta(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l) p_\theta(\boldsymbol{x} \mid \boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l). \tag{12}$$

We use variational inference to derive the Evidence Lower Bound (ELBO) of the log-likelihood:

$$\begin{aligned}\mathcal{L}(\boldsymbol{x} \mid \mathcal{T}) := \mathbb{E}_{q(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l \mid \boldsymbol{x})}[\log p(\boldsymbol{x} \mid \boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l)] \\ - \text{KL}\left(q\left(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l \mid \boldsymbol{x}\right) \| p\left(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l\right)\right).\end{aligned} \tag{13}$$

where the variational posterior $q\left(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l \mid \boldsymbol{x}\right)$ is defined in Section 2.3. The first component of the ELBO is the reconstruction term:

$$\mathcal{L}_{rec} = \mathbb{E}_{q(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l \mid \boldsymbol{x})}[\log p(\boldsymbol{x} \mid \boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l)] \tag{14}$$
$$\begin{aligned}\approx \frac{1}{M} \sum_{m=1}^{M} \sum_{l \in \mathbb{L}} P(l; \boldsymbol{c}) \\ \times \log \mathcal{N}(\mathbf{x} \mid \mu_{x,l}(\boldsymbol{z}_l^{(m)}), \sigma^2_{x,l}(\boldsymbol{z}_l^{(m)})),\end{aligned} \tag{15}$$

where $\mathcal{P}_i$ for $i \in \mathbb{V}$ is the path from root to node i, $P(i; \boldsymbol{c})$ is the probability of reaching node i, which is the product over the probabilities of the decisions in the path until i, $\boldsymbol{z}_l^{(m)}$ are the Monte Carlo (MC) samples, and M the number of the MC samples. Intuitively, the reconstruction loss is the sum of the leaf-wise reconstruction losses weighted by the probabilities of reaching the respective leaf. Note that here we sum over all possible paths in the tree, which is equal to the number of leaves.

The second term of (13) is the Kullback–Leibler divergence (KL) between the prior and the variational posterior of the tree. It can be written as a sum of the KL_{root}, the KL_{nodes}, and the $\text{KL}_{decisions}$:

$$\text{KL}_{root} = \text{KL}(q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \| p(\boldsymbol{z}_0)), \tag{16}$$
$$\begin{aligned}\text{KL}_{nodes} \approx \frac{1}{M} \sum_{m=1}^{M} \sum_{i \in \mathbb{V} \setminus \{0\}} P(i; \boldsymbol{c}) \\ \times \text{KL}(q(\boldsymbol{z}_i^{(m)} \mid pa(\boldsymbol{z}_i^{(m)})) \| p(\boldsymbol{z}_i^{(m)} \mid pa(\boldsymbol{z}_i^{(m)}))),\end{aligned} \tag{17}$$
$$\begin{aligned}\text{KL}_{decisions} \approx \frac{1}{M} \sum_{m=1}^{M} \sum_{i \in \mathbb{V} \setminus \mathbb{L}} P(i; \boldsymbol{c}) \\ \times \sum_{c_i \in \{0,1\}} q(c_i \mid \boldsymbol{x}) \log \left(\frac{q(c_i \mid \boldsymbol{x})}{p(c_i \mid \boldsymbol{z}_i^{(m)})} \right),\end{aligned} \tag{18}$$

where M is the number of MC samples. We refer to Appendix A for the full derivation. The KL_{root} is the KL between the standard Gaussian prior $p(\boldsymbol{z}_0)$ and the variational posterior of the root $q(\boldsymbol{z}_0 \mid \boldsymbol{x})$, thus enforcing the root to be compact. The KL_{nodes} is the sum of the node-specific KLs weighted by the probability of reaching their node i: $P(i; \boldsymbol{c})$. The node-specific KL of node i is the KL between the two Gaussians $q(\boldsymbol{z}_i \mid pa(\boldsymbol{z}_i))$, $p(\boldsymbol{z}_i \mid pa(\boldsymbol{z}_i))$. Finally, the last term, $\text{KL}_{decisions}$, is the weighted sum of all the KLs of the decisions, which are Bernoulli random variables, $KL(q(\mathrm{c}_i \mid \boldsymbol{x}) \mid p(\mathrm{c}_i \mid \boldsymbol{z}_i))) = \sum_{c_i \in \{0,1\}} q(c_i \mid \boldsymbol{x}) \log \left(\frac{q(c_i \mid \boldsymbol{x})}{p(c_i \mid \boldsymbol{z}_i))} \right)$. The hierarchical specification of the binary tree allows encoding expressive models while retaining the computational efficiency of fully factorized models.

2.5. Growing The Tree

In the previous sections, we discussed the variational objective to learn the parameters of the model given $\mathcal{T}$. To learn the structure of the binary tree $\mathcal{T}$ we follow an incremental approach. TreeVAE starts by training a tree composed of a root and two leaves for N_t epochs by maximising the ELBO. Once the model converges, a leaf is selected, and two children are attached. The selection criteria can be determined by e.g. the reconstruction loss or the ELBO. In our experiments, we selected the leaves with the maximum number of samples to retain balanced clusters. The resulting sub-tree is then trained and the process is repeated until the tree reaches its maximum capacity or until a predefined maximum number of leaves is met. The entire model is then fine-tuned for N_f epochs by unfreezing all weights. During fine-tuning, the tree is pruned by removing empty branches. For a schematic overview, we refer to Appendix A.3.

3. Results

We evaluate the clustering and generative performance of TreeVAE by setting the number of leaves to the true number of clusters and compute accuracy (ACC) and normalized mutual information (NMI). In terms of generative performance, we compute the approximated true log-likelihood and the reconstruction loss (15). We compare the generative performance of TreeVAE to the VAE (Rezende et al., 2014; Kingma & Welling, 2014), its non-hierarchical counterpart, and the LadderVAE (Sønderby et al., 2016), its sequential counterpart. We compare TreeVAE to non-generative hierarchical clustering baselines: Ward's agglomerative clustering (Agg) (Ward, 1963; Murtagh & Legendre, 2014), and the DeepECT (Mautz et al., 2020). We propose two additional baselines, where we perform Ward's clustering on the latent space of the VAE and of the last layer of the LadderVAE. For details on the experimental setup, we refer to Appendix D.

Hierarchical Clustering Results Table 1 shows the quantitative hierarchical clustering results. As can be seen, TreeVAE outperforms the baselines in all datasets. This suggests that the proposed approach successfully builds an optimal tree based on the data's intrinsic characteristics. Among the different baselines, agglomerative clustering using Ward's method (Agg) trained on the last layer of LadderVAE shows competitive performances. To the best of our knowledge, we are the first to report these results. Notably, it consistently outperforms VAE + Agg, indicating that the last layer of LadderVAE captures more cluster information than the VAE. In Appendix E.1, we show additional experiments where we assume *unknown* true number of clusters.

Generative Results In Table 2, we evaluate the generative performance of TreeVAE. The proposed approach outperforms the baselines on the majority of datasets, indicating that the proposed ELBO (13) provides a tighter lower bound of the log-likelihood. The most notable improvement appears to be reflected in the reconstruction loss, showing the advantage of using L cluster-specialized decoders. However, this improvement comes at the expense of a larger neural network architecture and an increase in the number of parameters. While this requires more computational resources at training time, during deployment the tree structure of TreeVAE permits lightweight inference through conditional sampling, thus matching the inference time of LadderVAE. Finally, we notice that more complex methods are prone to overfitting on the 20Newsgroups dataset.

Table 1: Test set hierarchical clustering performances (%) of TreeVAE compared with baselines. Means and standard deviations are computed across 10 runs. The star "*" indicates the inclusion of contrastive learning.

Dataset	Method	ACC	NMI
MNIST	Agg	69.5 ± 0.0	71.1 ± 0.0
	VAE + Agg	86.6 ± 4.9	81.6 ± 2.0
	LadderVAE + Agg	80.3 ± 5.6	82.0 ± 2.1
	DeepECT	74.9 ± 6.2	76.7 ± 4.2
	TreeVAE (ours)	$\mathbf{90.2} \pm 7.5$	$\mathbf{90.0} \pm 4.6$
Fashion	Agg	51.3 ± 0.0	52.6 ± 0.0
	VAE + Agg	54.9 ± 4.4	56.1 ± 3.2
	LadderVAE + Agg	55.9 ± 3.0	60.7 ± 1.4
	DeepECT	51.8 ± 5.7	57.7 ± 3.7
	TreeVAE (ours)	$\mathbf{63.6} \pm 3.3$	$\mathbf{64.7} \pm 1.4$
20Newsgroups	Agg	26.1 ± 0.0	27.5 ± 0.0
	VAE + Agg	15.2 ± 0.4	11.6 ± 0.3
	LadderVAE + Agg	17.4 ± 0.9	17.8 ± 0.6
	DeepECT	15.6 ± 3.0	18.1 ± 4.1
	TreeVAE (ours)	$\mathbf{32.8} \pm 2.3$	$\mathbf{34.4} \pm 1.5$
Omniglot-5	Agg	53.2 ± 0.0	33.3 ± 0.0
	VAE + Agg	52.9 ± 4.2	34.4 ± 2.9
	LadderVAE + Agg	$\mathbf{59.6} \pm 4.9$	44.2 ± 4.7
	DeepECT	41.1 ± 4.2	23.5 ± 4.3
	TreeVAE (ours)	$\mathbf{63.9} \pm 7.0$	$\mathbf{50.0} \pm 5.9$
CIFAR-10*	VAE + Agg	14.4 ± 0.2	1.9 ± 1.7
	LadderVAE + Agg	19.3 ± 0.6	7.4 ± 0.4
	DeepECT	10.3 ± 0.4	0.2 ± 0.1
	TreeVAE (ours)	$\mathbf{53.0} \pm 1.3$	$\mathbf{41.4} \pm 1.1$

Table 2: Test set generative performances of TreeVAE with 10 leaves compared with baselines. Means and standard deviations are computed across 10 runs.

Dataset	Method	LL	RL
MNIST	VAE	-101.9 ± 0.2	87.2 ± 0.3
	LadderVAE	-99.9 ± 0.5	87.8 ± 0.7
	TreeVAE (ours)	$\mathbf{-92.9} \pm 0.2$	$\mathbf{80.3} \pm 0.2$
Fashion	VAE	-242.2 ± 0.2	231.7 ± 0.5
	LadderVAE	-239.4 ± 0.5	231.5 ± 0.6
	TreeVAE (Ours)	$\mathbf{-234.7} \pm 0.1$	$\mathbf{226.5} \pm 0.3$
20Newsgroups	VAE	$\mathbf{-44.26} \pm 0.01$	45.52 ± 0.03
	LadderVAE	-44.30 ± 0.03	$\mathbf{43.52} \pm 0.03$
	TreeVAE (Ours)	-51.67 ± 0.59	45.83 ± 0.36
Omniglot	VAE	-115.3 ± 0.3	101.6 ± 0.3
	LadderVAE	-113.1 ± 0.5	100.7 ± 0.7
	TreeVAE (Ours)	$\mathbf{-110.4} \pm 0.5$	$\mathbf{96.9} \pm 0.5$

Real-world Imaging Data & Contrastive Learning Clustering real-world imaging data is extremely difficult as there are endless possibilities of how the data can be partitioned. We therefore inject prior information through augmentations to guide TreeVAE and the baselines to semantically meaningful splits. We regularize the model to have similar embeddings, as well as paths, for augmented versions of the same sample. For more details, we refer to Appendix B. Table 1 (bottom) shows the hierarchical

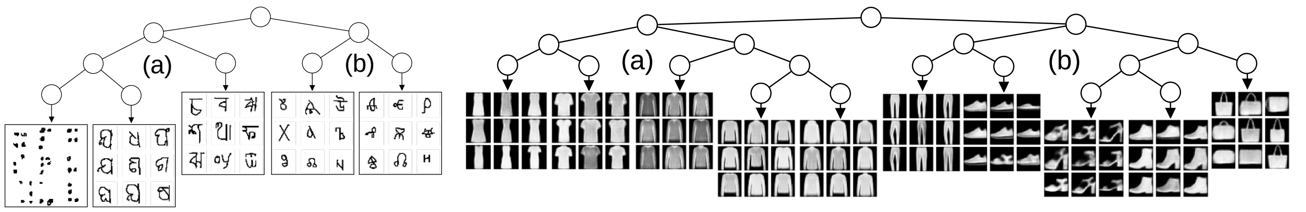

Figure 3: Hierarchical structures learned by TreeVAE on Omniglot-5 (left) and Fashion (right). Left: Subtree (a) learns a hierarchy over Braille and the Indian alphabets, while (b) groups Slavic alphabets. Right: Subtree (a) encodes tops, while (b) encodes shoes, purses, and pants.

clustering performance of TreeVAE and its baselines, all employing contrastive learning, on CIFAR-10. We observe that DeepECT struggles in separating the data as their contrastive approach leads to all samples falling into the same leaf. TreeVAE is able to group the data into contextually meaningful hierarchies and groups, evident from its superior performance compared to the baselines.

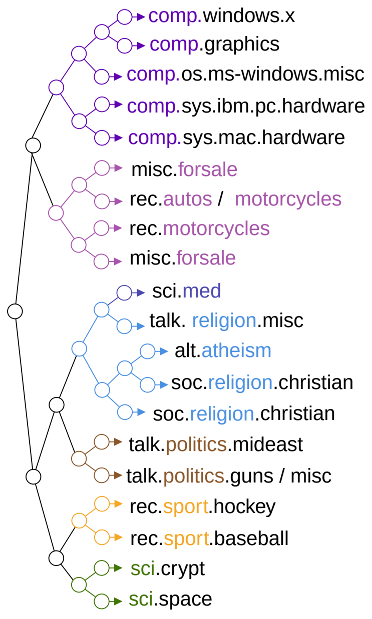

Figure 4: Hierarchical structure learned by TreeVAE on 20Newsgroups.

Discovery of Hierarchies In addition to solely clustering data, TreeVAE is able to discover meaningful hierarchical relations between the clusters, thus allowing for more insights into the dataset. In the introductory Fig. 1 and in Fig. 3 we present the hierarchical structures learned by TreeVAE on CIFAR-10 (Krizhevsky & Hinton, 2009), Fashion-MNIST (Xiao et al., 2017), Omniglot-5 (Lake et al., 2015), and 20Newsgroups (Lang, 1995). In Fig.3 (left), TreeVAE learns to split alphabets into Indian (Odia and Bengali) and Slavic (Glagolitic and Cyrillic) subtrees. In Fig. 3 (right) we additionally display conditional cluster generations from the leaf-specific decoders. TreeVAE separates the fashion items into two subtrees, one containing shoes and bags, and the other containing the tops, which are further refined into long and short sleeves. In Fig. 4, TreeVAE learns to separate technological and societal subjects and discovers semantically meaningful subtrees. In Appendix E we present more learned trees and how TreeVAE can additionally be used to sample unconditional generations for all clusters simultaneously, where the generations differ across the leaves by their cluster-specific features, whereas cluster-independent properties are retained across all generations.

4. Conclusion

In this paper, we introduced TreeVAE, a new generative method that leverages a tree-based posterior distribution of latent variables to capture the hierarchical structures present in the data. TreeVAE optimizes the balance between shared and specialized architecture, enhancing the learning and adaptation capabilities of generative models. Empirically, we showed that our model offers a substantial improvement in hierarchical clustering performance compared to the related work, while also providing a more competitive lower bound to the log-likelihood of the data. We presented qualitatively how the hierarchical structures learned by TreeVAE enable a more comprehensive understanding of the data, thereby facilitating enhanced analysis, interpretation, and decision-making. Our findings highlight the versatility of the proposed approach, which we believe to hold significant potential for unsupervised representation learning.

Limitations & Future Work: While deep latent variable models, such as VAEs, provide a framework for modelling explicit relationships through graphical structures, they often exhibit poor performance on synthetic image generation. However, more complex architectural design (Vahdat & Kautz, 2020) or recent advancement in diffusion latent models (Rombach et al., 2021), present potential solutions to enhance image quality generation, thus striking an optimal balance between generating high-quality images and capturing meaningful representations.

References

Arenas, M., Barceló, P., Orth, M. A. R., and Subercaseaux, B. On computing probabilistic explanations for decision trees. In *NeurIPS*, 2022.

Bae, J., Zhang, M. R., Ruan, M., Wang, E., Hasegawa, S., Ba, J., and Grosse, R. B. Multi-rate VAE: Train once, get the full rate-distortion curve. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OJ8aSjCaMNK.

Basak, J. and Krishnapuram, R. Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE Trans. Knowl. Data Eng.*, 17(1):121–132, 2005.

Bengio, Y., Courville, A. C., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35: 1798–1828, 2012.

Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Blei, D. M., Jordan, M. I., Griffiths, T. L., and Tenenbaum, J. B. Hierarchical topic models and the nested chinese restaurant process. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pp. 17–24, Cambridge, MA, USA, 2003. MIT Press.

Blockeel, H. and De Raedt, L. Top-down induction of clustering trees. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 55–63. Morgan Kaufmann Publishers, 1998.

Bredell, G., Flouris, K., Chaitanya, K., Erdil, E., and Konukoglu, E. Explicitly minimizing the blur error of variational autoencoders. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9krnQ-ue9M.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Wadsworth, 1984. ISBN 0-534-98053-8.

Campello, R. J., Moulavi, D., Zimek, A., and Sander, J. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):1–51, 2015.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Ester, M., Kriegel, H., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Simoudis, E., Han, J., and Fayyad, U. M. (eds.), *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pp. 226–231. AAAI Press, 1996. URL http://www.aaai.org/Library/KDD/1996/kdd96-037.php.

Falck, F., Williams, C., Danks, D., Deligiannidis, G., Yau, C., Holmes, C. C., Doucet, A., and Willetts, M. A multi-resolution framework for u-nets with applications to hierarchical VAEs. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=PQFr7FbGbO.

Fraiman, R., Ghattas, B., and Svarc, M. Interpretable clustering using unsupervised binary trees. *Adv. Data Anal. Classif.*, 7(2):125–145, 2013. doi: 10.1007/s11634-013-0129-3. URL https://doi.org/10.1007/s11634-013-0129-3.

Frosst, N. and Hinton, G. E. Distilling a neural network into a soft decision tree. In Besold, T. R. and Kutz, O. (eds.), *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), Bari, Italy, November 16th and 17th, 2017*, volume 2071 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017. URL https://ceur-ws.org/Vol-2071/CExAIIA_2017_paper_3.pdf.

Goyal, P., Hu, Z., Liang, X., Wang, C., Xing, E. P., and Mellon, C. Nonparametric variational auto-encoders for hierarchical representation learning. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5104–5112, 2017.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. Draw: A recurrent neural network for image generation. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1462–1471, Lille, France, 2015. PMLR. URL https://proceedings.mlr.press/v37/gregor15.html.

He, J., Gong, Y., Marino, J., Mori, G., and Lehrmann, A. M. Variational autoencoders with jointly optimized latent dependency structure. In *ICLR*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Heller, K. A. and Ghahramani, Z. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*, pp. 297–304, 2005.

Jordan, M. I. and Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015. doi: 10.1126/science.aaa8415. URL https://www.science.org/doi/abs/10.1126/science.aaa8415.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Klushyn, A., Chen, N., Kurle, R., Cseke, B., and Smagt, P. v. d. *Learning Hierarchical Priors in VAEs*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Kobren, A., Monath, N., Krishnamurthy, A., and McCallum, A. A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pp. 255–264. ACM, 2017a. doi: 10.1145/3097983.3098079. URL https://doi.org/10.1145/3097983.3098079.

Kobren, A., Monath, N., Krishnamurthy, A., and McCallum, A. A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 255–264, 2017b.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350 (6266):1332–1338, 2015. doi: 10.1126/science.aab3050. URL https://www.science.org/doi/abs/10.1126/science.aab3050.

Lang, K. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331–339, 1995.

Laptev, D. and Buhmann, J. M. Convolutional decision trees for feature learning and segmentation. In *Pattern Recognition: 36th German Conference, GCPR 2014, Münster, Germany, September 2-5, 2014, Proceedings 36*, pp. 95–106. Springer, 2014.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86:2278–2324, 1998.

Li, Y., Hu, P., Liu, J. Z., Peng, D., Zhou, J. T., and Peng, X. Contrastive clustering. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 8547–8555. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17037.

Liu, B., Xia, Y., and Yu, P. S. Clustering through decision tree construction. In *International Conference on Information and Knowledge Management*, 2000.

Maaløe, L., Fraccaro, M., Liévin, V., and Winther, O. Biva: A very deep hierarchy of latent variables for generative modeling. In *NeurIPS*, 2019.

Mattei, P.-A. and Frellsen, J. Leveraging the exact likelihood of deep latent variable models. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Mautz, D., Plant, C., and Böhm, C. Deepect: The deep embedded cluster tree. *Data Science and Engineering*, 5: 419 – 432, 2020.

Monath, N., Zaheer, M., Silva, D., McCallum, A., and Ahmed, A. Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 714–722, 2019.

Moshkovitz, M., Yang, Y., and Chaudhuri, K. Connecting interpretability and robustness in decision trees through separation. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7839–7849. PMLR, 2021.

Murtagh, F. and Contreras, P. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1): 86–97, 2012.

Murtagh, F. and Legendre, P. Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion? *Journal of Classification*, 31:274–295, 2014.

Nasiri, A. and Bepler, T. Unsupervised object representation learning using translation and rotation group equivariant VAE. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=qmm__jMjMlL.

Nistér, D. and Stewénius, H. Scalable recognition with a vocabulary tree. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2:2161–2168, 2006.

Pace, A., Chan, A. J., and van der Schaar, M. POETREE: interpretable policy learning with adaptive decision trees. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=AJsI-ymaKn_.

Ram, P. and Gray, A. G. Density estimation trees. In *Knowledge Discovery and Data Mining*, 2011.

Ranganath, R., Tran, D., and Blei, D. M. Hierarchical variational models. *ArXiv*, abs/1511.02386, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pp. 1278–1286. PMLR, 2014.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, 2021.

Rota Bulo, S. and Kontschieder, P. Neural decision forests for semantic image labelling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 81–88, 2014.

Shin, S.-J., Song, K., and Moon, I.-C. Hierarchically clustered representation learning. In *AAAI Conference on Artificial Intelligence*, 2019.

Sneath, P. H. The application of computers to taxonomy. *Microbiology*, 17(1):201–226, 1957.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder variational autoencoders. In *NIPS*, 2016.

Souza, V. F., Cicalese, F., Laber, E. S., and Molinaro, M. Decision trees with short explainable rules. In *NeurIPS*, 2022.

Steinbach, M. S., Karypis, G., and Kumar, V. A comparison of document clustering techniques. 2000.

Suárez, A. and Lutsko, J. F. Globally optimal fuzzy decision trees for classification and regression. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(12):1297–1311, 1999. doi: 10.1109/34.817409. URL https://doi.org/10.1109/34.817409.

Tanno, R., Arulkumaran, K., Alexander, D., Criminisi, A., and Nori, A. Adaptive neural trees. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6166–6175. PMLR, 2019. URL https://proceedings.mlr.press/v97/tanno19a.html.

Vahdat, A. and Kautz, J. Nvae: A deep hierarchical variational autoencoder. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 19667–19679. Curran Associates, Inc., 2020.

van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Vikram, S., Hoffman, M. D., and Johnson, M. J. The loracs prior for vaes: Letting the trees speak for the data. *ArXiv*, abs/1810.06891, 2018.

Wan, A., Dunlap, L., Ho, D., Yin, J., Lee, S., Petryk, S., Bargal, S. A., and Gonzalez, J. E. NBDT: neural-backed decision tree. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=mCLVeEpplNE.

Ward, J. H. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.

Webb, S., Goliński, A., Zinkov, R., Narayanaswamy, S., Rainforth, T., Teh, Y. W., and Wood, F. Faithful inversion of generative models for effective amortized inference. In *Neural Information Processing Systems*, 2017.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL http://arxiv.org/abs/1708.07747.

Xiao, T. Z. and Bamler, R. Trading information between latents in hierarchical variational autoencoders. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=eWtMdr6yCmL.

You, C., Robinson, D. P., and Vidal, R. Scalable sparse subspace clustering by orthogonal matching pursuit. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3918–3927, 2015.

Zharmagambetov, A. and Carreira-Perpiñán, M. Á. Semi-supervised learning with decision trees: Graph laplacian tree alternating optimization. In *NeurIPS*, 2022.

A. Evidence Lower Bound

In this section, we provide a closer look at the loss function of TreeVAE. In Appendix A.1 we focus on the derivations of the Kullback-Leibler divergence term and of the reconstruction term of the Evidence Lower Bound and provide an interpretable factorization. Furthermore, in Appendix A.2 we address the computational complexity, thus offering an in-depth understanding of the loss function, its practical implications, and the trade-offs involved in its computation. Finally, in Appendix A.3 we provide a schematic overview that illustrates the step-by-step process of the TreeVAE growing procedure.

A.1. ELBO Derivations

We hereby derive both terms of the ELBO (13), namely the KL loss, which is the Kullback–Leibler divergence (KL) between the prior and the variational posterior of TreeVAE, and the reconstruction loss. Let us define $\mathcal{P}_l$ as the decision path from root 0 to leaf l, L is the number of leaves, which is equal to the number of paths in $\mathcal{T}$, $\mathbf{z}_{\mathcal{P}_l} = \{\mathbf{z}_i \mid i \in \mathcal{P}_l\}$ the set of latent variables selected by the path $\mathcal{P}_l$, the parent node of the node i as $pa(i)$, $p(c_{pa(i)\to i} \mid \boldsymbol{z}_{pa(i)})$ the probability of going from $pa(i)$ to i. For example, if we consider the path in Fig. 2 (right) we will observe $c_0 = 0$, $c_1 = 1$, and $c_4 = 0$, where $c_i = 0$ means the model selects the left child of node i.

A.1.1. KL Loss

The KL loss can be expanded using Eq. 1/3:

$$\mathrm{KL}\left(q\left(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l \mid \boldsymbol{x}\right) \| p\left(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l\right)\right) \tag{19}$$

$$= \mathrm{KL}\Bigg(q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \Big\| p(\boldsymbol{z}_0) \prod_{i\in\mathcal{P}_l\setminus\{0\}} p(c_{pa(i)\to i} \mid \boldsymbol{z}_{pa(i)}) p(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)})\Bigg) \tag{20}$$

$$= \sum_{l\in\mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \times \log\left(\frac{q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{j\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(j)\to j} \mid \boldsymbol{x}) q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}{p(\boldsymbol{z}_0) \prod_{k\in\mathcal{P}_l\setminus\{0\}} p(c_{pa(k)\to k} \mid \boldsymbol{z}_{pa(k)}) p(\boldsymbol{z}_k \mid \boldsymbol{z}_{pa(k)})}\right) \tag{21}$$

$$= \sum_{l\in\mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log\left(\frac{q(\boldsymbol{z}_0 \mid \boldsymbol{x})}{p(\boldsymbol{z}_0)}\right) \tag{22}$$

$$+ \sum_{l\in\mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log\left(\frac{\prod_{j\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(j)\to j} \mid \boldsymbol{x})}{\prod_{k\in\mathcal{P}_l\setminus\{0\}} p(c_{pa(k)\to k} \mid \boldsymbol{z}_{pa(k)})}\right) \tag{23}$$

$$+ \sum_{l\in\mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log\left(\frac{\prod_{j\in\mathcal{P}_l\setminus\{0\}} q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}{\prod_{k\in\mathcal{P}_l\setminus\{0\}} p(\boldsymbol{z}_k \mid \boldsymbol{z}_{pa(k)})}\right). \tag{24}$$

In the following, we will simplify each of the three terms 22, 23, and 24 separately.

KL Root The term (22) corresponds to the KL of the root node. We can integrate out all the latent variables $\boldsymbol{z}_i$ for $i \neq 0$ and all decisions c_i. The first term can be then written as follows:

$$\begin{aligned}
\mathrm{KL}_{root} &= \sum_{i=1,2} \int_{\boldsymbol{z}_0} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) q(c_{0\to i} \mid \boldsymbol{z}_0) \log \left(\frac{q(\boldsymbol{z}_0 \mid \boldsymbol{x})}{p(\boldsymbol{z}_0)} \right) && (25)\\
&= \int_{\boldsymbol{z}_0} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \left[\sum_{i\in\{1,2\}} q(c_{0\to i} \mid \boldsymbol{z}_0) \right] \log \left(\frac{q(\boldsymbol{z}_0 \mid \boldsymbol{x})}{p(\boldsymbol{z}_0)} \right) && (26)\\
&= \int_{\boldsymbol{z}_0} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \left[q(\mathrm{c}_0 = 0 \mid \boldsymbol{z}_0) + q(\mathrm{c}_0 = 1 \mid \boldsymbol{z}_0) \right] \log \left(\frac{q(\boldsymbol{z}_0 \mid \boldsymbol{x})}{p(\boldsymbol{z}_0)} \right) && (27)\\
&= \int_{\boldsymbol{z}_0} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \log \left(\frac{q(\boldsymbol{z}_0 \mid \boldsymbol{x})}{p(\boldsymbol{z}_0)} \right) = \mathrm{KL}\left(q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \| p(\boldsymbol{z}_0) \right), && (28)
\end{aligned}$$

where $q(\mathrm{c}_0 = 0 \mid \boldsymbol{z}_0) + q(\mathrm{c}_0 = 1 \mid \boldsymbol{z}_0) = 1$ and $\mathrm{KL}\left(q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \| p(\boldsymbol{z}_0) \right)$ is the KL between two Gaussians, which can be computed analytically.

KL Decisions The second term (23) corresponds to the KL of the decisions. We can pull out the product from the log, yielding

$$\begin{aligned}
\mathrm{KL}_{decisions} = \sum_{l\in\mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \\
\times \log \left(\prod_{j\in\mathcal{P}_l\setminus\{0\}} \frac{q(c_{pa(j)\to j} \mid \boldsymbol{x})}{p(c_{pa(j)\to j} \mid \boldsymbol{z}_{pa(j)})} \right) && (29)
\end{aligned}$$

$$= \sum_{l\in\mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} \sum_{j\in\mathcal{P}_l\setminus\{0\}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log \left(\frac{q(c_{pa(j)\to j} \mid \boldsymbol{x})}{p(c_{pa(j)\to j} \mid \boldsymbol{z}_{pa(j)})} \right) \quad (30)$$

Let us define as $\mathcal{P}_{l\in j}$ all paths that go through node j, as $\mathcal{P}_{\leq j}$ (denoted as $\mathcal{P}_j$ in the main text for brevity) the unique path that ends in the node j, and as $\mathcal{P}_{>j}$ all the possible paths that start from the node j and continue to a leaf $l \in \mathbb{L}$. Similarly, let us define as $\boldsymbol{z}_{\leq j}$ all the latent embeddings that are contained in the path from the root to node j and as $\boldsymbol{z}_{>j}$ all the latent embeddings of the nodes $i > j$ that can be reached from node j.
To factorize the above equation, we first change from a pathwise view to a nodewise view. Instead of summing over all possible leaves in the tree ($\sum_{l\in\mathbb{L}}$) and then over each contained node ($\sum_{j\in\mathcal{P}_l\setminus\{0\}}$), we sum over all nodes ($\sum_{j\in\mathbb{V}\setminus\{0\}}$) and then over each path that leads through the selected node ($\sum_{\mathcal{P}_{l\in j}}$).

$$\begin{aligned}
&\sum_{l\in\mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} \sum_{j\in\mathcal{P}_l\setminus\{0\}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log \left(\frac{q(c_{pa(j)\to j} \mid \boldsymbol{x})}{p(c_{pa(j)\to j} \mid \boldsymbol{z}_{pa(j)})} \right) \\
&= \sum_{j\in\mathbb{V}\setminus\{0\}} \sum_{\mathcal{P}_{l\in j}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log \left(\frac{q(c_{pa(j)\to j} \mid \boldsymbol{x})}{p(c_{pa(j)\to j} \mid \boldsymbol{z}_{pa(j)})} \right) \qquad (31)
\end{aligned}$$

The above can be proved with the following Lemma, where we rewrite $\sum_{\mathcal{P}_{l\in j}} = \sum_{l\in\mathbb{L}} \mathbb{1}[j \in \mathcal{P}_l]$.

Lemma A.1. *Given a binary tree $\mathcal{T}$ as defined in Section 2.1, composed of a set of nodes $\mathbb{V} = \{0, \ldots, V\}$ and leaves $\mathbb{L} \subset \mathbb{V}$, where $\mathcal{P}_l$ is the decision path from root 0 to leaf l, and $\mathbf{z}_{\mathcal{P}_l} = \{\mathbf{z}_i \mid i \in \mathcal{P}_l\}$ the set of latent variables selected by the path $\mathcal{P}_l$. Then it holds*

$$\sum_{l\in\mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} \sum_{j\in\mathcal{P}_l\setminus\{0\}} f(j, l, \boldsymbol{z}_{\mathcal{P}_l}) = \sum_{j\in\mathbb{V}\setminus\{0\}} \sum_{l\in\mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} \mathbb{1}[j \in \mathcal{P}_l] f(j, l, \boldsymbol{z}_{\mathcal{P}_l}), \qquad (32)$$

Proof. The proof is as follows:

$$\sum_{j\in\mathbb{V}\setminus\{0\}}\sum_{l\in\mathbb{L}}\int_{\boldsymbol{z}_{\mathcal{P}_l}}\mathbb{1}[j\in\mathcal{P}_l]f(j,l,\boldsymbol{z}_{\mathcal{P}_l})=\sum_{j\in\mathbb{V}\setminus\{0\}}\sum_{l\in\mathbb{L}}\int_{\boldsymbol{z}_{\mathcal{P}_l}}f(j,l,\boldsymbol{z}_{\mathcal{P}_l})\sum_{i\in\mathcal{P}_l\setminus\{0\}}\mathbb{1}[i=j] \tag{33}$$

$$=\sum_{l\in\mathbb{L}}\sum_{j\in\mathbb{V}\setminus\{0\}}\int_{\boldsymbol{z}_{\mathcal{P}_l}}\sum_{i\in\mathcal{P}_l\setminus\{0\}}f(j,l,\boldsymbol{z}_{\mathcal{P}_l})\mathbb{1}[i=j] \tag{34}$$

$$=\sum_{l\in\mathbb{L}}\sum_{j\in\mathbb{V}\setminus\{0\}}\int_{\boldsymbol{z}_{\mathcal{P}_l}}\sum_{i\in\mathcal{P}_l\setminus\{0\}}f(i,l,\boldsymbol{z}_{\mathcal{P}_l})\mathbb{1}[i=j] \tag{35}$$

$$=\sum_{l\in\mathbb{L}}\int_{\boldsymbol{z}_{\mathcal{P}_l}}\sum_{i\in\mathcal{P}_l\setminus\{0\}}f(i,l,\boldsymbol{z}_{\mathcal{P}_l})\sum_{j\in\mathbb{V}\setminus\{0\}}\mathbb{1}[i=j] \tag{36}$$

$$=\sum_{l\in\mathbb{L}}\int_{\boldsymbol{z}_{\mathcal{P}_l}}\sum_{i\in\mathcal{P}_l\setminus\{0\}}f(i,l,\boldsymbol{z}_{\mathcal{P}_l}) \tag{37}$$

$$=\sum_{l\in\mathbb{L}}\int_{\boldsymbol{z}_{\mathcal{P}_l}}\sum_{j\in\mathcal{P}_l\setminus\{0\}}f(j,l,\boldsymbol{z}_{\mathcal{P}_l}). \tag{38}$$

□

Having proven the equality, we can continue with the KL of the decisions as follows:

$$\mathrm{KL}_{decisions}=$$

$$\sum_{j\in\mathbb{V}\setminus\{0\}}\sum_{\mathcal{P}_{l\in j}}\int_{\boldsymbol{z}_{\mathcal{P}_l}}q(\boldsymbol{z}_0\mid\boldsymbol{x})\prod_{i\in\mathcal{P}_l\setminus\{0\}}q(c_{pa(i)\to i}\mid\boldsymbol{x})q(\boldsymbol{z}_i\mid\boldsymbol{z}_{pa(i)})\log\left(\frac{q(c_{pa(j)\to j}\mid\boldsymbol{x})}{p(c_{pa(j)\to j}\mid\boldsymbol{z}_{pa(j)})}\right) \tag{39}$$

$$\begin{aligned}=&\sum_{j\in\mathbb{V}\setminus\{0\}}\sum_{\mathcal{P}_{l\in j}}\int_{\boldsymbol{z}_{\mathcal{P}_l}}[q(\boldsymbol{z}_0\mid\boldsymbol{x})\prod_{i\in\mathcal{P}_{\leq j}\setminus\{0\}}q(c_{pa(i)\to i}\mid\boldsymbol{x})q(\boldsymbol{z}_i\mid\boldsymbol{z}_{pa(i)})\log\left(\frac{q(c_{pa(j)\to j}\mid\boldsymbol{x})}{p(c_{pa(j)\to j}\mid\boldsymbol{z}_{pa(j)})}\right)\\&\times\prod_{k\in\mathcal{P}_{>j}}q(c_{pa(k)\to k}\mid\boldsymbol{x})q(\boldsymbol{z}_k\mid\boldsymbol{z}_{pa(k)})]\end{aligned} \tag{40}$$

$$\begin{aligned}=&\sum_{j\in\mathbb{V}\setminus\{0\}}\sum_{\mathcal{P}_{>j}}\int_{\boldsymbol{z}_{\leq j},\boldsymbol{z}_{>j}}[q(\boldsymbol{z}_0\mid\boldsymbol{x})\prod_{i\in\mathcal{P}_{\leq j}\setminus\{0\}}q(c_{pa(i)\to i}\mid\boldsymbol{x})q(\boldsymbol{z}_i\mid\boldsymbol{z}_{pa(i)})\log\left(\frac{q(c_{pa(j)\to j}\mid\boldsymbol{x})}{p(c_{pa(j)\to j}\mid\boldsymbol{z}_{pa(j)})}\right)\\&\times\prod_{k\in\mathcal{P}_{>j}}q(c_{pa(k)\to k}\mid\boldsymbol{x})q(\boldsymbol{z}_k\mid\boldsymbol{z}_{pa(k)})]\end{aligned} \tag{41}$$

$$\begin{aligned}=&\sum_{j\in\mathbb{V}\setminus\{0\}}\int_{\boldsymbol{z}_{\leq j}}q(\boldsymbol{z}_0\mid\boldsymbol{x})\prod_{i\in\mathcal{P}_{\leq j}\setminus\{0\}}q(c_{pa(i)\to i}\mid\boldsymbol{x})q(\boldsymbol{z}_i\mid\boldsymbol{z}_{pa(i)})\log\left(\frac{q(c_{pa(j)\to j}\mid\boldsymbol{x})}{p(c_{pa(j)\to j}\mid\boldsymbol{z}_{pa(j)})}\right)\\&\times\sum_{\mathcal{P}_{>j}}\int_{\boldsymbol{z}_{>j}}\left[\prod_{k\in\mathcal{P}_{>j}}q(c_{pa(k)\to k}\mid\boldsymbol{x})q(\boldsymbol{z}_k\mid\boldsymbol{z}_{pa(k)})\right]\end{aligned} \tag{42}$$

From Eq. 39 to Eq. 40, we split the inner product into the nodes of the paths $\mathcal{P}_{l\in j}$ that are before and after the node j.
From Eq. 40 to Eq. 41, we observe that the sum over all paths going through j can be reduced to the sum over all paths starting from j, because there is only one path to j, which is specified in the product that comes after.
From Eq. 41 to Eq. 42, we observe that the sum over paths starting from j and integral over $\boldsymbol{z}_{>j}$ concern only the terms of

the second line. Observe that the term on the second line of Eq. 42 integrates out to 1 and we get

$$\begin{aligned}
&\mathrm{KL}_{decisions} = \\
&\qquad \sum_{j\in\mathbb{V}\setminus\{0\}} \int_{\boldsymbol{z}_{\leq j}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_{\leq j}\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log\left(\frac{q(c_{pa(j)\to j} \mid \boldsymbol{x})}{p(c_{pa(j)\to j} \mid \boldsymbol{z}_{pa(j)})}\right) &(43)\\
&= \sum_{j\in\mathbb{V}\setminus\{0\}} \int_{\boldsymbol{z}_{<j}} \int_{\boldsymbol{z}_j} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_{\leq j}\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log\left(\frac{q(c_{pa(j)\to j} \mid \boldsymbol{x})}{p(c_{pa(j)\to j} \mid \boldsymbol{z}_{pa(j)})}\right) &(44)\\
&= \sum_{j\in\mathbb{V}\setminus\{0\}} \int_{\boldsymbol{z}_{<j}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_{<j}\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log\left(\frac{q(c_{pa(j)\to j} \mid \boldsymbol{x})}{p(c_{pa(j)\to j} \mid \boldsymbol{z}_{pa(j)})}\right) \\
&\qquad \times \int_{\boldsymbol{z}_j} q(c_{pa(j)\to j} \mid \boldsymbol{x}) q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)}) &(45)\\
&= \sum_{j\in\mathbb{V}\setminus\{0\}} \int_{\boldsymbol{z}_{<j}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_{<j}\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) q(c_{pa(j)\to j} \mid \boldsymbol{x}) \\
&\qquad \times \log\left(\frac{q(c_{pa(j)\to j} \mid \boldsymbol{x})}{p(c_{pa(j)\to j} \mid \boldsymbol{z}_{pa(j)})}\right). &(46)
\end{aligned}$$

From Eq. 44 to Eq. 45, we single out the term in the product that corresponds to $j = i$, which is the only term that depends on $\int_{z_i}$.
From Eq. 45 to Eq. 46, we observe that in the singled-out term, $\int_{z_j} q(z_j \mid \boldsymbol{z}_{pa(j)}) = 1$, which leaves only $q(c_{pa(j)\to j} \mid \boldsymbol{x})$.

This equation can be rewritten in a more interpretable way. Let us define the probability of reaching node j and observing $\boldsymbol{z}_j$ as

$$P(j; \boldsymbol{z}, \boldsymbol{c}) = q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_{\leq j}\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}). \tag{47}$$

Then the KL term of the decisions can be simplified as

$$\begin{aligned}
\mathrm{KL}_{decisions} &= \sum_{j\in\mathbb{V}\setminus\{0\}} \int_{\boldsymbol{z}_{<j}} P(pa(j); \boldsymbol{z}, \boldsymbol{c}) q(c_{pa(j)\to j} \mid \boldsymbol{x}) \log\left(\frac{q(c_{pa(j)\to j} \mid \boldsymbol{x})}{p(c_{pa(j)\to j} \mid \boldsymbol{z}_{pa(j)})}\right) &(48)\\
&= \sum_{i\in\mathbb{V}\setminus\mathbb{L}} \sum_{k\in\{0,1\}} \int_{\boldsymbol{z}_{<i}} P(i; \boldsymbol{z}, \boldsymbol{c}) q(c_i = k \mid \boldsymbol{x}) \log\left(\frac{q(c_i = k \mid \boldsymbol{x})}{p(c_i = k \mid \boldsymbol{z}_i)}\right). &(49)
\end{aligned}$$

This term requires Monte Carlo sampling for the expectations over the latent variables $\boldsymbol{z}$, while we can analytically compute the sum over all decisions $\mathcal{P}_l$.

$$\begin{aligned}
&\mathrm{KL}_{decisions} = \sum_{i\in\mathbb{V}\setminus\mathbb{L}} \int_{\boldsymbol{z}_{<i}} P(i; \boldsymbol{z}, \boldsymbol{c}) \\
&\qquad \times \left[q(c_i = 0) \mid \boldsymbol{x}) \log\left(\frac{q(\mathrm{c}_i = 0 \mid \boldsymbol{x})}{p(\mathrm{c}_i = 0 \mid \boldsymbol{z}_i)}\right) + q(\mathrm{c}_i = 1 \mid \boldsymbol{x}) \log\left(\frac{q(\mathrm{c}_i = 1 \mid \boldsymbol{x})}{p(\mathrm{c}_i = 1 \mid \boldsymbol{z}_i)}\right)\right] &(50)\\
&\approx \frac{1}{M} \sum_{m=1}^{M} \sum_{i\in\mathbb{V}\setminus\mathbb{L}} P(i; \boldsymbol{z}^{(m)}, \boldsymbol{c}) \\
&\qquad \times \left[q(c_i = 0) \mid \boldsymbol{x}) \log\left(\frac{q(\mathrm{c}_i = 0 \mid \boldsymbol{x})}{p(\mathrm{c}_i = 0 \mid \boldsymbol{z}_i^{(m)})}\right) + q(\mathrm{c}_i = 1 \mid \boldsymbol{x}) \log\left(\frac{q(\mathrm{c}_i = 1 \mid \boldsymbol{x})}{p(\mathrm{c}_i = 1 \mid \boldsymbol{z}_i^{(m)})}\right)\right], &(51)
\end{aligned}$$

where $P(i; \boldsymbol{z}^{(m)}, \boldsymbol{c}) = \prod_{j\in\mathcal{P}_{\leq i}} q(c_{pa(j)\to j} \mid \boldsymbol{x})$ is the probability of reaching node i, defined as $P(i; \boldsymbol{c})$ in Eq. 15/17/18 for simplicity.

KL Nodes Finally, we can analyze the last term of the KL term, which corresponds to the KL of the nodes (24). The reasoning is similar to the equations above and we will use the same notation. The KL of the nodes can be written as

$$\begin{aligned}\mathrm{KL}_{nodes} = &\sum_{l\in\mathbb{L}}\int_{\boldsymbol{z}_{\mathcal{P}_l}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \\ &\times \log\left(\frac{\prod_{j\in\mathcal{P}_l\setminus\{0\}} q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}{\prod_{k\in\mathcal{P}_l\setminus\{0\}} p(\boldsymbol{z}_k \mid \boldsymbol{z}_{pa(k)})}\right)\end{aligned} \tag{52}$$

$$= \sum_{l\in\mathbb{L}}\int_{\boldsymbol{z}_{\mathcal{P}_l}} \sum_{j\in\mathcal{P}_l\setminus\{0\}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log\left(\frac{q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}{p(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}\right) \tag{53}$$

We now change from a pathwise view to a nodewise view.

$$= \sum_{j\in\mathbb{V}\setminus\{0\}} \sum_{\mathcal{P}_{l\in j}} \int_{\boldsymbol{z}_{\leq j}, \boldsymbol{z}_{>j}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_l\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log\left(\frac{q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}{p(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}\right) \tag{54}$$

$$\begin{aligned}= &\sum_{j\in\mathbb{V}\setminus\{0\}} \int_{\boldsymbol{z}_{\leq j}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_{\leq j}\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log\left(\frac{q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}{p(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}\right) \\ &\times \sum_{\mathcal{P}_{>j}} \int_{\boldsymbol{z}_{>j}} \left[\prod_{k\in\mathcal{P}_{>j}} q(c_{pa(k)\to k} \mid \boldsymbol{x}) q(\boldsymbol{z}_k \mid \boldsymbol{z}_{pa(k)})\right]\end{aligned} \tag{55}$$

$$= \sum_{j\in\mathbb{V}\setminus\{0\}} \int_{\boldsymbol{z}_{<j}} \int_{\boldsymbol{z}_j} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i\in\mathcal{P}_{\leq j}\setminus\{0\}} q(c_{pa(i)\to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log\left(\frac{q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}{p(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}\right) \tag{56}$$

$$= \sum_{j\in\mathbb{V}\setminus\{0\}} \int_{\boldsymbol{z}_{<j}} \int_{\boldsymbol{z}_j} P(pa(j); \boldsymbol{z}, \boldsymbol{c}) q(c_{pa(j)\to j} \mid \boldsymbol{x}) q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)}) \log\left(\frac{q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}{p(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}\right) \tag{57}$$

$$= \sum_{j\in\mathbb{V}\setminus\{0\}} \int_{\boldsymbol{z}_{<j}} P(pa(j); \boldsymbol{z}, \boldsymbol{c}) q(c_{pa(j)\to j} \mid \boldsymbol{x}) \int_{\boldsymbol{z}_j} q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)}) \log\left(\frac{q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}{p(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})}\right) \tag{58}$$

$$= \sum_{j\in\mathbb{V}\setminus\{0\}} \int_{\boldsymbol{z}_{<j}} P(pa(j); \boldsymbol{z}, \boldsymbol{c}) q(c_{pa(j)\to j} \mid \boldsymbol{x}) \,\mathrm{KL}\left(q(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)}) \mid p(\boldsymbol{z}_j \mid \boldsymbol{z}_{pa(j)})\right) \tag{59}$$

$$\begin{aligned}\approx &\frac{1}{M}\sum_{m=1}^{M} \sum_{i\in\mathbb{V}\setminus\{0\}} P(pa(i); \boldsymbol{z}^{(m)}, \boldsymbol{c}) q(c_{pa(i)\to i} \mid \boldsymbol{x}) \\ &\times \mathrm{KL}(q(\boldsymbol{z}_i^{(m)} \mid pa(\boldsymbol{z}_i^{(m)})) \| p(\boldsymbol{z}_i^{(m)} \mid pa(\boldsymbol{z}_i^{(m)})))\end{aligned} \tag{60}$$

$$= \frac{1}{M}\sum_{m=1}^{M} \sum_{i\in\mathbb{V}\setminus\{0\}} P(i; \boldsymbol{z}^{(m)}, \boldsymbol{c}) \,\mathrm{KL}(q(\boldsymbol{z}_i^{(m)} \mid pa(\boldsymbol{z}_i^{(m)})) \| p(\boldsymbol{z}_i^{(m)} \mid pa(\boldsymbol{z}_i^{(m)}))), \tag{61}$$

where $P(pa(j); \boldsymbol{z}, \boldsymbol{c})$ is defined in Eq. 47 and where $P(i; \boldsymbol{z}^{(m)}, \boldsymbol{c}) = P(i; \boldsymbol{c}) = \prod_{j\in\mathcal{P}_{\leq i}} q(c_{pa(j)\to j} \mid \boldsymbol{x})$ is the probability of reaching node i.

KL terms Using the above factorization, the KL term of the ELBO can be written as

$$\begin{aligned}\mathrm{KL}\left(q\left(\boldsymbol{z}, \mathcal{P}_l \mid \boldsymbol{x}\right) \| p\left(\boldsymbol{z}, \mathcal{P}_l\right)\right) \approx\ &\mathrm{KL}\left(q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \| p(\boldsymbol{z}_0)\right) \\ &+ \frac{1}{M}\sum_{m=1}^{M} \sum_{i\in\mathbb{V}\setminus\mathbb{L}} P(i; \boldsymbol{z}^{(m)}, \boldsymbol{c}) \sum_{c_i\in\{0,1\}} q(c_i \mid \boldsymbol{x}) \log\left(\frac{q(c_i \mid \boldsymbol{x})}{p(c_i \mid \boldsymbol{z}_i^{(m)}))}\right) \\ &+ \frac{1}{M}\sum_{m=1}^{M} \sum_{i\in\mathbb{V}\setminus\{0\}} P(i; \boldsymbol{z}^{(m)}, \boldsymbol{c}) \,\mathrm{KL}(q(\boldsymbol{z}_i^{(m)} \mid pa(\boldsymbol{z}_i^{(m)})) \| p(\boldsymbol{z}_i^{(m)} \mid pa(\boldsymbol{z}_i^{(m)}))),\end{aligned} \tag{62}$$

where $P(i; \boldsymbol{z}^{(m)}, \boldsymbol{c}) = P(i; \boldsymbol{c}) = \prod_{j\in\mathcal{P}_{\leq i}} q(c_{pa(j)\to j} \mid \boldsymbol{x})$.

A.1.2. Reconstruction Loss

Finally, to compute the full ELBO, the KL terms are added to the reconstruction loss defined in (15):

$$\mathcal{L}_{rec} = \mathbb{E}_{q(\boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l \mid \boldsymbol{x})}[\log p(\boldsymbol{x} \mid \boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l)] \tag{63}$$

$$= \sum_{l \in \mathbb{L}} \int_{\boldsymbol{z}_{\mathcal{P}_l}} q(\boldsymbol{z}_0 \mid \boldsymbol{x}) \prod_{i \in \mathcal{P}_l \setminus \{0\}} q(c_{pa(i) \to i} \mid \boldsymbol{x}) q(\boldsymbol{z}_i \mid \boldsymbol{z}_{pa(i)}) \log p(\boldsymbol{x} \mid \boldsymbol{z}_{\mathcal{P}_l}, \mathcal{P}_l) \tag{64}$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} \sum_{l \in \mathbb{L}} P(l; \boldsymbol{c}) \log \mathcal{N}\left(\mathbf{x} \mid \mu_{x,l}\left(\boldsymbol{z}_l^{(m)}\right), \sigma_{x,l}^2\left(\boldsymbol{z}_l^{(m)}\right)\right), \tag{65}$$

$$P(i; \boldsymbol{c}) = \prod_{j \in \mathcal{P}_i \setminus \{0\}} q(c_{pa(j) \to j} \mid \boldsymbol{x}) \quad \text{for } i \in \mathbb{V}, \tag{66}$$

where $\mathcal{P}_i$ for $i \in \mathbb{V}$ is the path from root to node i, $P(i; \boldsymbol{c})$ is the probability of reaching node i, which is the product over the probabilities of the decisions in the path until i, $\boldsymbol{z}_l^{(m)}$ are the Monte Carlo (MC) samples, and M the number of the MC samples. Here, assumptions about the distribution of the inputs are required. For the grayscale datasets such as MNIST, Fashion-MNIST, and Omniglot, as well as the one-hot-encoded 20Newsgroup, we assume that the inputs are Bernoulli distributed, such that the resulting reconstruction loss is the binary cross entropy. On the other hand, for the colored dataset CIFAR-10 we assume that the pixel values are normally distributed, which leads to the mean squared error as loss function, where we assume that $\sigma = 1$.

A.2. Computational Complexity

All terms of the Evidence Lower Bound, Eq. 13, can be computed efficiently and the computational complexity of a single joint update of the parameters is $O(LBVDC_p)$, where L is the number of MC samples, B is the batch size, V is the number of nodes in the tree, D is the maximum depth of the tree, and C_p is the cost to compute the KL between Gaussians. It should be noted that the computational complexity is, in practice, reduced to $O(LBVC_p)$, as the term $P(i; \boldsymbol{c})$ can be computed dynamically from parent nodes.

A.3. Growing The Tree

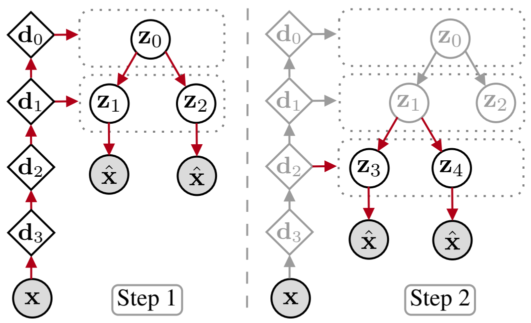

Figure 5: The first two steps of the growing process to learn the global structure of the tree during training. Highlighted in red are the trainable weights. (left) TreeVAE starts by optimizing the ELBO of the tree composed of a root and two leaves. (right) Upon convergence, a node is selected, $\boldsymbol{z}_1$, and two children are attached. The new sub-tree is then trained for M epochs by freezing the weights of the rest of the model. For efficiency, the subtree is trained using only the subset of data that have a high probability (higher than a threshold t) of being assigned to the parent node.

B. Integrating Prior Knowledge

Retrieving semantically meaningful clustering structures of real-world images is extremely challenging, as there are several underlying factors according to which the data can be clustered. Therefore, it is often crucial to integrate domain knowledge that guides the model toward desirable cluster assignments. Thus, we propose an extension of TreeVAE where we integrate recent advances in contrastive learning (Chen et al., 2020; van den Oord et al., 2018; Li et al., 2021), whereby prior knowledge on data invariances can be encoded through augmentations. For a batch $\boldsymbol{X}$ with N samples, we randomly augment every sample twice to obtain the augmented batch $\tilde{\boldsymbol{X}}$ with $2N$ samples. For all i, j where $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$ stem from the same sample, we compute the *NT-Xent* (Chen et al., 2020) defined as $\ell_{i,j} = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$, where $s_{i,j}$ denotes the cosine similarity between the representations of $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$, and τ is a temperature parameter. We calculate $\ell_{i,j}$ for (a) the output of separate projection heads $g_h(\mathbf{d}_h)$, which take as input the bottom-up embeddings $\mathbf{d}_h$, and (b) the probability output of the routers $r_{q,i}(\mathbf{d}_h)$. Finally, we average these terms and add them to the negative ELBO (13) for the real-world image experiment.

C. Related Work

We provide a review of relevant work in the domains of deep latent variable models, hierarchical clustering and decision trees. By doing so, we hope to shed further light on the current state-of-the-art approaches and contribute to a deeper understanding of the challenges and opportunities that lie in the intersection of hierarchical clustering, decision trees, and latent variable models.

Deep Latent Variable Models Deep latent variable models automatically learn structure from data by combining the flexibility of deep neural networks and the statistical foundations of generative models (Mattei & Frellsen, 2018). Variational autoencoders (VAEs) (Rezende et al., 2014; Kingma & Welling, 2014) are among the most used frameworks (Bae et al., 2023; Bredell et al., 2023; Nasiri & Bepler, 2022). A variety of works has been proposed to integrate more complex empirical prior distributions, thus reducing the gap between approximate and true posterior distributions (Ranganath et al., 2015; Webb et al., 2017; Klushyn et al., 2019). Among these, the most related to our work is the VAE-nCRP (Goyal et al., 2017; Shin et al., 2019) and the TMC-VAE (Vikram et al., 2018). Both works use Bayesian nonparametric hierarchical clustering priors based on the nested Chinese restaurant process (nCRP) prior (Blei et al., 2003), and on the time-marginalized coalescent (TMC). However, even if they allow more flexible prior distributions these models suffer from restrictive posterior distributions (Kingma et al., 2016).To overcome the above issue, deep hierarchical VAEs (Gregor et al., 2015; Kingma et al., 2016) have been proposed to employ structured approximate posteriors, which are composed of hierarchies of conditional stochastic variables that are connected *sequentially*. Among a variety of proposed methods (Vahdat & Kautz, 2020; Xiao & Bamler, 2023; Falck et al., 2022), Ladder VAE (Sønderby et al., 2016) is most related to TreeVAE. The authors propose to model the approximate posterior by combining a "bottom-up" recognition distribution with the "top-down" prior. Further extensions include BIVA (Maaløe et al., 2019), which introduces a bidirectional inference network, and GraphVAE (He et al., 2019), that integrates a more flexible dependency structure of latent variables. Contrary to the previous approaches, TreeVAE models a *tree-based* posterior distribution of latent variable, thus allowing hierarchical clustering of samples.

C.1. Hierarchical Clustering

Hierarchical clustering algorithms have long been employed in the field of data mining and machine learning to extract hierarchical structures from data (Sneath, 1957; Ward, 1963; Murtagh & Contreras, 2012). Agglomerative clustering is among the earliest and most well-known hierarchical clustering algorithms. These methods start with each data point as an individual cluster and then iteratively merge the closest pairs of clusters, according to a predefined distance metric, until a stopping criterion is met. While single-linkage and complete-linkage agglomeration clustering are widely used as baselines, we observe better performance when using the bottom-up strategy proposed by Ward (1963). Ward's minimum variance criterion minimizes the total within-cluster variance (Murtagh & Legendre, 2014), thus providing balanced and compact clusters. In contrast, the Bayesian Hierarchical Clustering (BHC) proposed by Heller & Ghahramani (2005) takes a different approach by employing hypothesis testing to determine when to merge the clusters. The divisive clustering algorithms, on the other hand, provide a different strategy to hierarchical clustering. Unlike agglomerative methods, divisive clustering starts with all data points in a single cluster and recursively splits clusters into smaller ones. The proposed TreeVAE is an example of a divisive clustering method. Among a variety of proposed methods, the Bisecting-K-means algorithm (Steinbach et al., 2000; Nistér & Stewénius, 2006) is widely used for its simplicity; it applies k-means with two clusters recursively. More recent approaches include PERCH (Kobren et al., 2017b), which is a non-greedy, incremental algorithm that scales to both the number of data points and the number of clusters, GHC (Monath et al., 2019), which leverages continuous representations of trees in a hyperbolic space and optimizes a differentiable cost function, and RSSCOMP (You et al., 2015), which explores a subspace clustering method based on an orthogonal matching pursuit. Finally, Deep ECT (Mautz et al., 2020) proposes a divisive hierarchical embedded clustering method, which jointly optimizes an autoencoder that compresses the data into an embedded space and a hierarchical clustering layer on top of it. Density-based clustering algorithms, such as DBSCAN, belong to a distinct category of clustering techniques. They aim to identify regions in a dataset where points are densely concentrated and classify outliers as noise (Ester et al., 1996). Campello et al. (2015) build on this idea to learn a hierarchy based on the distances between datapoints where distance is roughly determined by the density. However, one limitation of density-based methods lie in their performance when confronted with complex datasets requiring high-dimensional representations. In such cases, estimating density requires an exponentially growing number of data points, which leads to scalability issues that TreeVAE does not have.

Decision Trees Decision trees (Breiman et al., 1984) are interpretable, non-parametric supervised learning techniques commonly employed in classification and regression tasks. They rely on the data itself to build a hierarchical structure. This is done by recursively partitioning the data into subsets, each of which corresponds to a specific node in the tree. At each node, a decision rule is generated based on one of the input features that best discriminates the data in that subset. One of the key advantages of decision trees is their interpretability. The learned tree structure can be easily visualized to get insights into the data and the model's decision-making process. Suárez & Lutsko (1999) argue that deterministic splits lead to overfitting and introduce fuzzy decision trees with probabilistic decisions, implicitly allowing for backpropagation. With the advancement of neural networks, many works (Rota Bulo & Kontschieder, 2014; Laptev & Buhmann, 2014; Frosst & Hinton, 2017) leverage MLPs or CNNs for learning a more complex decision rule. However, the input at every node remains the original features, which limits their performance, as they are unable to learn meaningful representations. Thus, Tanno et al. (2019) introduces Adaptive Neural Trees (ANT), a method that learns flexible, hierarchical representations through NNs, hereby facilitating hierarchical separation of task-relevant features. Additionally, ANTs architectures grow dynamically such that they can adapt to the complexity of the training dataset. At inference time, ANTs allow for lightweight conditional computation via the most likely path, leading to inbuilt interpretable decision-making.
While decision trees were initially designed with the goal of achieving high predictive performance, they are also used for many auxiliary goals such as semi-supervised learning (Zharmagambetov & Carreira-Perpiñán, 2022), robustness (Moshkovitz et al., 2021) or interpretability (Souza et al., 2022; Arenas et al., 2022; Pace et al., 2022; Wan et al., 2021). In the context of interpretability, various approaches have been proposed to enhance accuracy and interpretability. Wan et al. (2021) introduce Neural-Backed Decision Trees (NBDTs), which improve accuracy by replacing the final layer of a neural network with a differentiable sequence of decisions to increase its interpretability while retaining predictive performance. Souza et al. (2022) focus on optimizing the structural parameters of decision trees, introducing the concept of "explanation size" as being the expected number of attributes required for prediction to measure interpretability. Pace et al. (2022) develop the POETREE framework for interpretable policy learning via decision trees in time-varying clinical decision environments. Arenas et al. (2022) investigate explanations in decision trees, considering both deterministic and probabilistic approaches and showing the limitations thereof. These works collectively contribute to advancing the use of decision trees in interpretable machine learning, providing insights into trade-offs, criteria, and frameworks for improving accuracy and interpretability.
While decision trees are most commonly known for their application in supervised tasks, they can be repurposed to partition the data space into clusters in an unsupervised way (Liu et al., 2000; Ram & Gray, 2011; Fraiman et al., 2013; Blockeel & De Raedt, 1998; Basak & Krishnapuram, 2005). Most methods, however, have the downside of not learning meaningful representations for their splits, such that they are unfit for modelling complex interactions. Therefore, similar to Tanno et al. (2019) in the supervised setting, in this work, we combine the simplicity and interpretability of clustering decision trees with the flexibility of NNs.

D. Experimental Setup

Datasets and Metrics: We evaluate the clustering and generative performance of TreeVAE on MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), 20Newsgroups (Lang, 1995), Omniglot (Lake et al., 2015), and Omniglot-5, where only 5 vocabularies (Braille, Glagolitic, Cyrillic, Odia, and Bengali) are selected and used as true labels. We preprocessed the 20Newsgroups data by selecting the top $2,000$ tf-idf words and we reshaped Omniglot to 28×28. We assess the hierarchical clustering performance by computing dendrogram purity (DP) and leaf purity (LP), as defined by (Kobren et al., 2017a) using the datasets labels, where we assume the number of true clusters is unknown. We also report standard clustering metrics, accuracy (ACC) and normalized mutual information (NMI), by setting the number of leaves for TreeVAE and for the baselines to the true number of clusters. In terms of generative performance, we compute the approximated true log-likelihood calculated using 1000 importance-weighted samples, together with the reconstruction loss (15). We also perform a hierarchical clustering experiment on real-world imaging data, namely CIFAR-10, using the contrastive extension described in Appendix B.

Baselines: We compare the generative performance of TreeVAE to the VAE (Rezende et al., 2014; Kingma & Welling, 2014), its non-hierarchical counterpart, and the LadderVAE (Sønderby et al., 2016), its sequential counterpart. For a fair comparison, all methods share the same architecture and hyperparameters whenever possible. We compare TreeVAE to non-generative hierarchical clustering baselines for which the code was publicly available: Ward's minimum variance agglomerative clustering (Agg) (Ward, 1963; Murtagh & Legendre, 2014), and the DeepECT (Mautz et al., 2020). We propose two additional baselines, where we perform Ward's agglomerative clustering on the latent space of the VAE (VAE + Agg) and of the last layer of the LadderVAE (LadderVAE + Agg). For the contrastive clustering experiment, we apply a contrastive loss similar to TreeVAE to the VAE and the LadderVAE, while for DeepECT we use the contrastive loss proposed by the authors.

Implementation Details: While we believe that more complex architectures could have a substantial impact on the performance of TreeVAE, we choose to employ rather simple settings to validate the proposed approach. We set the dimension of all latent embeddings $\mathbf{z} = \{\mathbf{z}_0, \dots, \mathbf{z}_V\}$ to 8 for MNIST, Fashion, and Omniglot, to 4 for 20Newsgroups, and to 64 for CIFAR-10. The maximum depth of the tree is set to 6 for all datasets, except 20Newsgroups where we increased the depth to 7 to capture more clusters. To compute DP and LP, we allow the tree to grow to a maximum of 30 leaves for 20Newsgroups, and 20 for the rest, while for ACC and NMI we fix the number of leaves to the number of true classes. The transformations consist of one-layer MLPs of size 128 and the routers of two-layers of size 128 for all datasets except for the real-world imaging data where we slightly increase the MLP complexity to 512. As encoders we use an MLP with 5 layers on the tf-idf vectorized 20Newsgroup, and CNN's for all others. For MNIST and Fashion-MNIST, we use 3 layers of 3×3 convolutions with stride 2 and a final linear layer on the flattened CNN output to reach the stated embedding dimensionality. For Omniglot the encoder consists of 4×4 convolutions with alternating stride 1 and 2 for 6 layers in total, again with a final linear layer. Lastly, for CIFAR-10, we follow a ResNet (He et al., 2016) inspired approach; 4 layers of 3×3 convolutions with stride 1 and residual connections followed by 2D average-pooling of kernel size 3 and stride 2, again with a final linear layer. The decoders for all datasets are symmetric to the encoders, where the average pooling is replaced by upsampling. We apply batch normalization followed by LeakyReLU non-linearities after all convolutional and dense layers. The trees are trained for $M = 150$ epochs at each growth step, and the final tree is finetuned for $M_f = 200$ epochs. We anneal the KL terms of the ELBO to reduce the risk of posterior collapse during training. For the real-world imaging experiment, we set the weight of the contrastive loss to 100.

E. Further Experiments

Here, we provide additional experimental results, including further clustering experiments, hierarchical structures, and unconditional generation of samples.

E.1. Clustering

In order to measure not only the flat clustering performance, but also evaluate the learned hierarchical structure, we conduct experiments to compute DP and LP. The results are shown in Table 3

E.2. Discovery of Hierarchies

We show further hierarchical structures learned by TreeVAE on MNIST in Fig. 6/7, Omniglot-50 in Fig. 8, and 20Newsgroups in Fig. 4. Additionally, the figures that include conditional generations are generated by sampling from the root and following the most probable path to generate every leaf-specific image.

E.3. Generations

We further show how TreeVAE can be used to sample unconditional generations for all clusters simultaneously for MNIST in Fig. 9. Differently from the conditional generation, where we follow the path of the tree with the highest probability, here we follow all paths of the tree regardless of their probabilities. As a result, we generate 10 images, one for every decoder. These generated samples exhibit distinct characteristics based on their respective cluster-specific features while maintaining cluster-independent properties across all generated instances.

Table 3: Test set hierarchical clustering performances (%) of TreeVAE compared with baselines. Means and standard deviations are computed across 10 runs. The star "*" indicates the inclusion of contrastive learning.

Dataset	Method	DP	LP
MNIST	Agg	63.7 $\pm$ 0.0	78.6 $\pm$ 0.0
	VAE + Agg	79.9 $\pm$ 2.2	90.8 $\pm$ 1.4
	LadderVAE + Agg	81.6 $\pm$ 3.9	90.9 $\pm$ 2.5
	DeepECT	74.6 $\pm$ 5.9	90.7 $\pm$ 3.2
	TreeVAE (ours)	**87.9** $\pm$ 4.9	**96.0** $\pm$ 1.9
Fashion	Agg	45.0 $\pm$ 0.0	67.6 $\pm$ 0.0
	VAE + Agg	44.3 $\pm$ 2.5	65.9 $\pm$ 2.3
	LadderVAE + Agg	49.5 $\pm$ 2.3	67.6 $\pm$ 1.2
	DeepECT	44.9 $\pm$ 3.3	67.8 $\pm$ 1.4
	TreeVAE (ours)	**54.4** $\pm$ 2.4	**71.4** $\pm$ 2.0
20Newsgroups	Agg	13.1 $\pm$ 0.0	30.8 $\pm$ 0.0
	VAE + Agg	7.1 $\pm$ 0.3	18.1 $\pm$ 0.5
	LadderVAE + Agg	9.0 $\pm$ 0.2	20.0 $\pm$ 0.7
	DeepECT	9.3 $\pm$ 1.8	17.2 $\pm$ 3.8
	TreeVAE (ours)	**17.5** $\pm$ 1.5	**38.4** $\pm$ 1.6
Omniglot-5	Agg	41.4 $\pm$ 0.0	63.7 $\pm$ 0.0
	VAE + Agg	46.3 $\pm$ 2.3	68.1 $\pm$ 1.6
	LadderVAE + Agg	49.8 $\pm$ 3.9	71.3 $\pm$ 2.0
	DeepECT	33.3 $\pm$ 2.5	55.1 $\pm$ 2.8
	TreeVAE (ours)	**58.8** $\pm$ 4.0	**77.7** $\pm$ 3.9
CIFAR-10*	VAE + Agg	10.5 $\pm$ 0.1	16.3 $\pm$ 0.2
	LadderVAE + Agg	12.8 $\pm$ 0.2	25.4 $\pm$ 0.6
	DeepECT	10.0 $\pm$ 0.1	10.3 $\pm$ 0.4
	TreeVAE (ours)	**35.3** $\pm$ 1.2	**53.9** $\pm$ 1.2

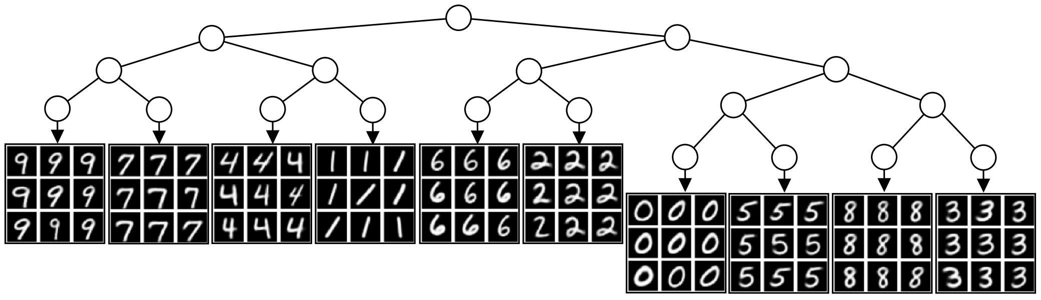

Figure 6: Hierarchical structure learned by TreeVAE with ten leaves on MNIST dataset with generated images through conditional sampling. The right subtree contains rounded digits, while the left subtree contains digits with a straight line.

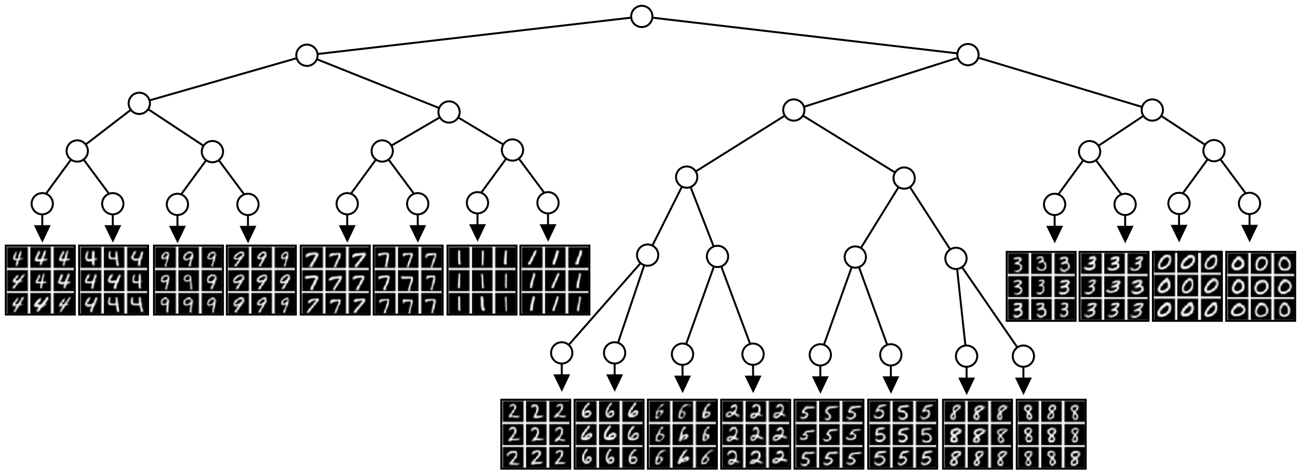

Figure 7: Hierarchical structure learned by TreeVAE with 20 leaves on MNIST dataset with generated images through conditional sampling. The digits are further divided according to style.

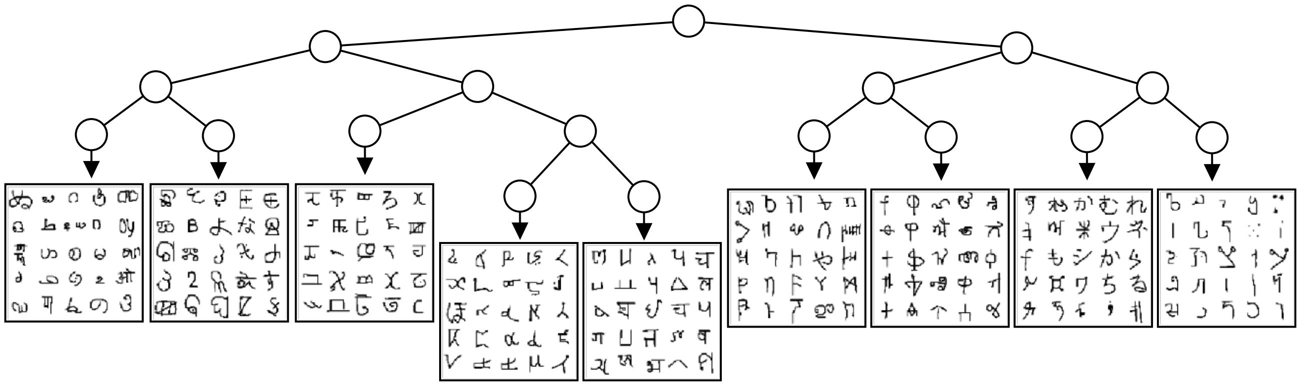

Figure 8: Hierarchical structure learned by TreeVAE on the full Omniglot dataset. We display random subsets of images that are probabilistically assigned to each leaf of the tree. Similar to the results in the main text, we can again find regional hierarchies in some individual subtrees. For instance, the leftmost subtree seems to find structures in different Indian alphabets. We can also observe that this subtree seems to cluster smaller characters in its left child, whereas the right child contains bigger shapes. Another example is the rightmost tree, which seems to encode the more straight shapes of Japanese writing styles. There, the left child encodes more complex shapes that contain many different strokes, and the right subtree groups simpler shapes with only a few lines.

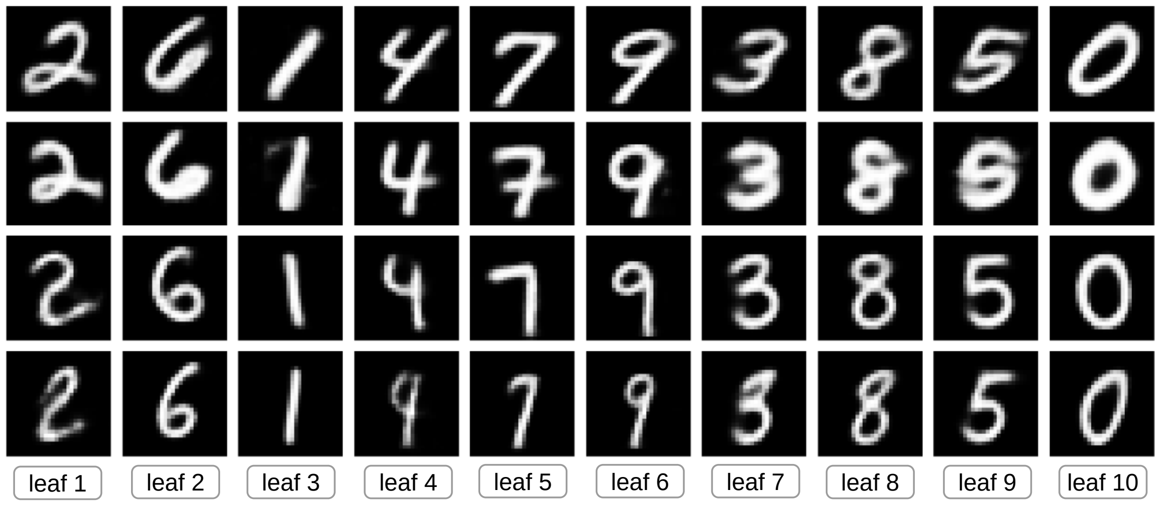

Figure 9: Selected unconditional generations of MNIST. One row corresponds to one sample from the root, for which we depict the visualizations obtained from the 10 leaf-decoders. Each row retains similar properties across the different leaves. In the first row, all digits are rotated; the second are bold; the third are straight; and the last are squeezed.