Interpretable Generative Modeling Using a Hierarchical Topological VAE

Etienne Desticourt, Véronique Letort, Florence d'Alché-Buc

Interpretable Generative Modeling using a Hierarchical Topological VAE

Etienne Desticourt
Centre d'Excellence Systme, ALMS
Air Liquide
Antony, France
etienne.desticourt@airliquide.com

Véronique Letort
MICS
CentraleSuplec, Université Paris-Saclay
Gif-sur-Yvette, France
veronique.lechevalier@centralesupelec.fr

Florence d'Alché–Buc
LTCI
Télécom Paris, IP Paris
Palaiseau, France
florence.dalche@telecom-paris.fr

***Abstract*—Generating realistic datasets with fine-grained control over their properties can help overcome challenges linked to the scarcity of data in many domains, such as medical applications. To that end, we extend Variational Autoencoders by using a hierarchical and topological prior consisting of a sequence of Self-Organizing Maps (SOM), which are stacked in the latent space and learned without supervision, jointly with the parameters of the variational autoencoder. We induce a hierarchy between the codes of the SOM sequence, each SOM corresponding to a different hierarchical level and learning increasingly finer-grained representations of the data. Our model combines the power of deep learning with the interpretability of hierarchical and topological clustering and produces competitive results when evaluated on three well-known computer vision benchmarks and a custom medical dataset.**

***Index Terms*—Deep-Learning, Generative Modeling, Hierarchical models, Variational Autoencoders**

I. Introduction

Realistic data generation has attracted a lot of attention in recent years, especially with the advent of generative adversarial networks [1], normalizing flows, as well as variational autoencoders [2], [3]. More than solely generating data, the effectiveness of these models lies in their ability to learn representations of their inputs and enable the generation of specific classes or properties. Work has been undertaken to improve these representations through disentanglement [4]–[6], structure and manifold learning, and more generally interpretable generation [7]. In the context of medical applications, generative models have been applied with some success [8] and hold the promise to complete, or even replace in some cases, model-based approaches like differential equations. However, the generation of realistic virtual patient data comes with higher requirements about the nature of generated data and their ability to cover a variety of patients. This is the case, for instance, when a medical ventilator is developed. Such a medical device must be able to handle a variety of different situations without the benefit of large annotated training datasets whose collection is a tremendous undertaking. Therefore, learning interpretable representations that allow fine control over the generated data is an essential prerequisite for this particular application.

National Association Technological Research (ANRT), CIFRE N 2018/0264

In this context, the Variational autoencoder (VAE) framework [2] stands out as it benefits from both the autoencoder framework and generative modeling. In particular, a prior probability can be defined and exploited during learning. One such type of prior are graph- and tree-based priors, which are well fit to represent data presenting correspondingly graphical or hierarchical properties. Hierarchical clustering in the latent space of generative networks has been the subject of some research, mostly by taking advantage of the hierarchical nature of neural networks, which learn increasingly abstract features as the layer depth increases [9]–[12]. But so far, few works have focused on imposing explicitly hierarchical, structured priors on the embedding space of autoencoders. Among them, VaDE [13] learns a Gaussian Mixture Model (GMM) in the latent space of its VAE. The LTVAE [14] learns a two-level tree in the latent space of its VAE where each node and its children form a GMM, the parameters of the prior being updated in an alternating fashion with its network parameters using message-passing. The nCRP-VAE [15] and HCRL [16] learn a Dirichlet Process in their latent space to recursively build a tree of arbitrary depth and determine its parameter using a custom variational inference method. However, all these models share a relative complexity both in terms of the prior expression and the separate learning algorithms required to learn their probabilistic parameters. Additionally, some of those models are limited to elementary structures of at most two levels, preventing them from learning representative hierarchies from the data.

In this paper, pursuing these lines of works, we propose a new VAE architecture, Hierarchical Topological VAE (HTVAE), which jointly learns deep non-linear embeddings along with a structured hierarchical prior. We take advantage of recent advances in discrete VAEs [17], [18] to learn a sequence of self-organizing maps in the latent space with parenthood constraints between the nodes in two successive self-organizing maps. The model assigns to each observation a sequence of nodes from these maps, forming a branch of the hierarchical prior. In doing so, our model is able to learn deep representations of high-dimensional data while capturing its inherent hierarchical properties. The interpretability of this representation allows generating realistic data, with fine

control over the classes or properties of the samples, by sampling from nodes at any desired level in the hierarchy.

The paper is organized as follows. After detailing some components upon which our approach is built in Section II, we describe our novel architecture and its learning algorithm in Section III. In Section IV, we briefly review other existing methods incorporating hierarchical architectures and compare them with our HTVAE model. Section V presents numerical experiments on three well-known computer vision benchmarks, MNIST, FashionMNIST and CIFAR-100, as well as on a custom dataset of breath recordings from medical ventilators. Eventually, a conclusion is drawn in Section VI.

II. Background

A. VAE

The variational autoencoder introduced by [2] is a generative neural network that learns to encode and model an i.i.d. sample $X = \{x_1, ..., x_N\}$ from a fixed but unknown distribution. The VAE relies on the following modeling assumption: the observed data X is supposed to be generated by a continuous random process using a random latent variable z distributed according the prior density $p_\theta(z)$ such that

$$p_\theta(x) = p_\theta(x|z)p_\theta(z). \tag{1}$$

As the posterior density $p_\theta(z|x)$ is intractable, the authors use a variational approximation $q_\phi(z|x)$ instead. The VAE assumes $p_\theta(z)$ is a multivariate Gaussian and learns the function $q_\phi(z|x) = \mathcal{N}(\mu, \sigma^2 I)$ using a probabilistic encoder in the form of a neural network which outputs μ and σ. The posterior is then sampled from and fed to a probabilistic decoder, another neural network, which learns the function $p_\theta(x|z)$. The encoder and decoder are jointly trained using a loss function composed of a reconstruction error and a regularization term based on the Kullback-Leibler divergence, which ensures that the prior is close to the approximate posterior. For a given observation x, we have:

$$\mathcal{L}(x) = -KL\left(q_\phi(z|x) \,||\, p_\theta(z)\right) + E_{q_\phi(z|x)}[\, log\, p_\theta(x|z)]. \tag{2}$$

B. SOM-VAE

Self-Organizing Map-VAE (SOM-VAE) [17] belongs to the family of discrete VAEs, along with recent works such as the Vector-Quantized VAE [18]. These models assign the continuous encodings sampled from the approximate posterior to their closest counterpart in a set of discrete codes in the latent space of the VAE. The assignment is done using a deterministic categorical distribution so that for the continuous encoding $z \in \mathbb{R}^L$ and a codebook of K codes $S = \{e_1, ..., e_K\} \in \mathbb{R}^{K \times L}$, we can sample discrete encodings $z_q \sim p(z_q|z)$ defined by the following equation:

$$p(z_q = e_i|z) = \mathbb{1}[i = argmin_j ||z - e_j||^2] \tag{3}$$

In the case of a SOM-VAE, the codebook is a self-organizing map (SOM) [19], a lattice graph in $\mathbb{R}^L$ in which the different codes organize themselves in the latent space with respect to their topological neighbours according to constraints defined in the loss.

For a given datapoint x, the loss $\mathcal{L}(x)$ is defined as the sum of three terms: $\mathcal{L}_{rec}(x)$, $\mathcal{L}_{com}(x)$, and $\mathcal{L}_{neigh}(x)$. The reconstruction loss $\mathcal{L}_{rec}(x)$ ensures that reconstructions sampled from the generative model are close to the observations and is defined as $\mathcal{L}_{rec}(x, x', x'_q) = ||x - x'_q||^2 + ||x - x'||^2$ where $x' \sim p(x|z)$ and $x'_q \sim p(x|z_q)$ are respectively the reconstructions of the continuous and discrete latent variables. Both reconstructions are used in this part of the loss as the inference model $q(z|x)$ is a neural network trained with back-propagation. As the discretization bottleneck breaks the backward gradient flow at z_q, the continuous reconstruction is used to provide gradients to the inference model. The commitment loss $\mathcal{L}_{com}(x) = ||z_e(x) - z_q(x)||^2$ ensures the continuous encodings and their discrete counterparts are close to each others. Eventually the neighbourhood loss $\mathcal{L}_{neigh}(x)$ enforces the self-organizing constraints on the SOM so that codes are close to the continuous encodings that are assigned to their topological neighbours so that $\mathcal{L}_{neigh}(x) = \sum_{e_i \in neigh(z_q)} \|e_i - sg[z_e(x)]\|^2$ with sg the stop-gradient operator and $neigh$ the neighbourhood function.

III. Proposed Model

A. Intuition

To get a fine control over data generation, the generative model induced by the VAE should rely on a hierarchy of classes. We first notice that in the same way that similar observations are encoded close to each others in the latent space of a Gaussian VAE, similar observations are encoded in the same code neighbourhood of a SOM-VAE's latent space. Put another way, the self-organizing map builds an unsupervised partition of the latent space where each discrete code represents a different intrinsic class. We propose to stack self-organizing maps in the latent space so that the whole model learns a hierarchical representation of the data. Moreover, we introduce a probabilistic perspective to the maps by making each code the center of a Gaussian distribution so that data may be generated from any code after training. Maps that have fewer codes will cover larger regions of space per code and therefore learn broader representations while maps with a larger amount of codes will learn more granular representations. By adding penalties between the different maps in the loss function, we can ensure that a hierarchy emerges between the codes at different levels and that these codes cover the same region of the latent space, thereby enabling generation at all levels of the hierarchy.

The hierarchy or tree can be built in an agnostic way or benefit from prior knowledge about some classes.

B. HTVAE's principle

The dataset is denoted $\mathbf{X} = \{x_1, ..., x_N | x_i \in \mathbb{R}^D\}$ and is supposed to be an i.i.d. sample from a fixed but unknown probability distribution. Similarly to a VAE, an HTVAE is composed of an encoder network f_θ which maps variable x to the latent space variable $z \in \mathbb{R}^L$ and of a generative decoder

network g_ϕ that is able to generate x from a given z. Using $\mathbf{X}$, the encoder learns the parameters of the approximate posterior distribution $q(z|x)$, while the generative network learns the distribution $p(x|z)$. However unlike a VAE, which uses an isotropic Gaussian prior, we use a topological hierarchical prior. We define a sequence of M maps $(S_i)_{i=1}^M$ at initialization time. Each map S_i is itself a set of K_i codes denoted $\{e_k^i\}_{k=1}^{K_i}$ in the latent space so that $S_i \in \mathbb{R}^{L \times K_i}$. The number of maps as well as the number of codes in each map are hyper-parameters of our model. The maps are ordered with decreasing depth so that S_1 is the deepest and S_M is the highest in the hierarchy. As explained in the previous section, we initialize the maps with $K_i > K_{i+1}$ so that deeper maps have a larger number of codes to learn a more granular discrete representation than their ancestors. Each map's codes are initialized using a single multivariate Gaussian in R^L with zero mean and a standard deviation σ_{init} which is an hyper-parameter of our model shared by all maps. In practice the initialization method does not have a big impact on learning as the maps self-organize in the latent space.

1) Inference: For a given observation our encoder outputs the mean and standard deviation vectors of an isotropic Gaussian in R^L so that $(\mu_0, \sigma_0) = f_\theta(x)$ which defines the approximate posterior $q(z|x) = \mathcal{N}(\mu_0, \sigma_0^2 I)$ of our model. From this distribution we sample the latent variable $z_0 \in \mathbb{R}^L$. These encodings are then discretized sequentially by each map $S_i = \{e_k^i\}_{k=1}^{K_i}$ in the hierarchy by mapping them to the code which is closest to them in the latent space. This produces a sequence of M discrete encodings $\mu_i \in \mathbb{R}^L \sim \prod_{j=1}^i p(\mu_j|\mu_{j-1})$ which defines the branch of the hierarchy to which the observation is assigned and is calculated according to the following categorical distribution, with $\mu_0 = z_0$ for $i = 1$:

$$p(\mu_i = e_j^i|\mu_{i-1}) = \mathbb{1}[j = \arg\min_k \|\mu_{i-1} - e_k^i\|^2] \quad (4)$$

In essence this distribution induces a child-parent relationship between maps so that a code in a map at a specific level of the hierarchy is assigned as a child of the code that is closest to it in the map at the next level. Additionally each discrete encoding μ_i is interpreted as the mean of a multivariate Gaussian whose standard deviation $\sigma_i \in \mathbb{R}^L$ is proportional to the distance to its closest topological neighbours $neigh(e_{i,j}) = \{e_{i-1,j}, e_{i+1,j}, e_{i,j-1}, e_{i,j+1}\}$ along each dimension of $\mathbb{R}^L$, weighed by a scalar hyper-parameter ρ, so that

$$\sigma_i = \frac{1}{\rho} min_{e \in neigh(z_i)} \|z_i - e\|^2 \quad (5)$$

We can sample from this Gaussian to create $z_i \sim \mathcal{N}(\mu_i, \sigma_i^2 I)$ at level i of the hierarchy, this sampling is done using the reparametrization trick described by [2] to allow the back-propagation to the code at μ_i. In enforcing the discretization of codes at deeper level by codes at higher level we ensure all maps cover the same region of the latent space. Moreover since the standard deviation of a Gaussian code depends on the distance to its closest neighbour, this ensures each map covers this region fully and similarly that the decoder will learn to build reconstructions from any point in this region, as described in the following section.

2) Generation: At training time the decoder is used to build $\{x_i'\}_{i=0}^M \in \mathbb{R}^D$ the reconstructions of the continuous encodings z_0 as well as all sampled map encodings z_1 to z_M by sampling from the generative model $p(x|z)$ parametrized by our deterministic decoder g_ϕ. This ensures the space covered by the Gaussians at every level of the hierarchy maps to real observations in data space and new samples can be generated at any level of the hierarchy.

Once trained, the model is able to generate samples of a given class at a chosen level of granularity: for a specific code e_k from map S_i, we sample $x' \sim p(z)p(x|z)$ with $p(z) \sim \mathcal{N}(\mu_k, \sigma_k^2 I)$.

C. Learning

Hierarchical prior $(S_i)_{i=1}^M$ and neural network parameters (θ, ϕ) are learnt jointly during training by minimizing a loss $\mathcal{L}$ that promotes the reconstruction of realistic samples from the hierarchical prior, the topological organization of each level of the hierarchy, and the learning of a good representation in the latent space. For sake of simplicity, we note the complete loss function as $\mathcal{L}(x, \{\mu_i\}_{i=1}^M, \{z_i\}_{i=0}^M, \{x_i'\}_{i=0}^M)$. Then, for a given x, we have:

$$\mathcal{L} = \mathcal{L}_{top}(\{\mu_i\}_{i=1}^M) + \mathcal{L}_{hierarchy}(\{z_i\}_{i=0}^M) + \mathcal{L}_{rec}(x, \{x_i'\}_{i=0}^M) \quad (6)$$

We define $\mathcal{L}_{hierarchy}$ to enforce the closeness between parent and children encodings and the codes of our SOM sequence.

$$\mathcal{L}_{hierarchy}(\{z_i\}_{i=0}^M) = \sum_{i=1}^M \|z_i - z_{i-1}\|^2 \quad (7)$$

This term also allows the gradient to back-propagate through the first part of the autoencoder as z_{i-1} corresponds to the continuous encodings z_0 produced by the encoder for $i = 1$. In order to enforce faithful reconstruction of the observations at any level of the hierarchy, given x the observation and $x_i' \sim p(x|z_i) \in \mathbb{R}^D$ its reconstruction from the encodings at level i, we define the reconstruction loss as follows:

$$\mathcal{L}_{rec}(x, \{x_i'\}_{i=0}^M) = \sum_{i=0}^M \|x_i' - x\|^2 \quad (8)$$

Finally we add a topological constraint $\mathcal{L}_{top}$ on each level so that the codes of each map are close to each others. This is achieved by forcing the sampled means $\mu_{i>=1} \sim \prod_{j=1}^i p(\mu_j|\mu_{j-1})q(z|x)$, as described in 4, to be close to their neighbours. We use a function $neigh\colon \mathbb{R}^L \to \mathbb{R}^{L \times 4}$ which takes a code as input and returns the codes which are connected to it by an edge to find the topological neighbours of the code located at μ_i in the S_i lattice.

$$\mathcal{L}_{top}(\{\mu_i\}_{i=1}^M) = \sum_{i=1}^M \sum_{e \in neigh(\mu_i)} \|\mu_i - e\|^2 \quad (9)$$

This topological loss enforces a neighborhood relationship for the discrete encodings μ_i of a given observation x, and not for every code e^i of the SOM S_i, which means the effect of this term on a code will be proportional to the number of observations discretized by this code.

We further describe the training procedure in algorithm III-C.

Algorithm 1 Training the HTVAE

Inputs:

$\mathbf{X}$ the observations

$\boldsymbol{M}$ the number of self-organizing maps

$\boldsymbol{\{K\}_{i=1}^{M}}$ the number of codes per map

$\boldsymbol{\sigma_{init}}$ the global variance hyper-parameter to initialize the SOM stack.

$\boldsymbol{\rho}$ the factor hyper-parameter that controls the variance of each code Gaussian relative to the distance to its closest topological neighbour

$\boldsymbol{E}$ the number of training epochs

Function `Train` ($\mathbf{X}$, $\boldsymbol{\sigma_0}$, $\boldsymbol{\rho}$, $\boldsymbol{M}$, $\boldsymbol{\{K\}_{i=1}^{M}}$, $\boldsymbol{E}$) **:**

- **for** $i = 1$ *to* M **do**
  - $S_i \leftarrow K_i$ codes drawn from $\mathcal{N}(0, \sigma_{init}^2 I)$
- **end**
- **for** $epoch = 0$ *to* E **do**
  - **for** ***each*** $x \in \mathbf{X}$ **do**
    - $z_0 \leftarrow$ drawn from $q(z|x)$
    - $x'_0 \leftarrow$ drawn from $p(x|z_0)$
    - **for** $i = 1$ *to* M **do**
      - $\mu_i \leftarrow$ drawn from $p(\mu_i|\mu_{i-1})$ using 4, with $\mu_0 = z_0$
      - $z_i \leftarrow$ drawn from $\mathcal{N}(\mu_i, \sigma_i^2 I)$
      - $x'_i \leftarrow$ drawn from $p_\theta(x|z_i)$
    - **end**
    - $g = \nabla_{(S_i)_{i=1}^M, \theta, \phi} \mathcal{L}(\{\mu_i\}_{i=1}^M, \{z_i\}_{i=0}^M, \{x'_i\}_{i=0}^M, x)$ using 6
    - $(S_i)_{i=1}^M, \theta, \phi \leftarrow$ Update parameters using g and Adam [20]
  - **end**
- **end**

return

D. Semi-Supervised Constraints

In some applications, there is not enough datapoints from a given class to ensure that the HTVAE will be able to learn to generate such data. In this case, prior knowledge under the form of Must-Link constraints can be imposed during learning. A Must-Link constraint forces pairs of given observations to belong to the same cluster. There are many ways to implement these constraints [21], they can be enforced directly when partitioning the observations, bypassing the normal assignment procedure [22], or by minimizing a dedicated metric [23], [24].

Implementing such a constraint in our framework can be challenging due to the discretization bottleneck which breaks the gradient backpropagation. To encode such a prior knowledge while preserving the gradient, we implement a soft must-link penalty term in the loss function which decreases as the linked pairs of of operation get closer topologically.

For two observations $x^1, x^2 \in \mathbb{R}^D$ whose continuous encodings are $z^1, z^2 \sim q(z|x)$, and codes that discretize these encodings are $e_1^i, e_2^i \in \mathbb{R}^L$ in SOM S_i, we define a loss function $\mathcal{L}_{must-link}^i \colon \mathbb{R}^{L\times 2} \rightarrow \mathbb{R}$ which maps a pair of observations to its continuous distance in the latent space weighed by its topological distance $d_i \colon \mathbb{R}^{L\times 2} \rightarrow \mathbb{N}$ at a chosen level i of the hierarchy so that

$$\mathcal{L}_{must-link}^i(z_1, z_2) = \ d_i(z_1, z_2).\|z_1 - z_2\|^2 \qquad (10)$$

IV. Related Works

The following approaches can be related to our works.

HKM & RGMM: The Hierarchical KMeans model [25] is a recursive application of the K-Means algorithm. A tree is defined with specific depth and branching factors at each level. The data space is then partitioned a first time into Vorono regions by learning centroids using KMeans. The procedure is repeated for each region to learn new centroids until the full depth of the tree is reached. In a similar manner, a Gaussian Mixture Model (GMM) can be applied recursively with a predefined depth and branching factors to build a tree from the data space. Moreover, one may use the encoder of a variational autoencoder to extract low-dimensional features from the data and learn a model on those features, improving the quality of the clustering.

nCRP-VAE: The nCRP-VAE [15] is a variational autoencoder which uses a nested Chinese Restaurant Process (nCRP) [26] as prior. The nCRP is akin to a recursive Dirichlet process and describes a tree whose root is a zero-centered Gaussian distribution in the latent space of the VAE. The root children are also Gaussians and their means are sampled from the root distribution. Recursively applying this sampling process, a full tree can be constructed. All the nodes share the same variance which is an hyper-parameter of the model. The parameters of the tree prior are updated using variational inference [27]. Note that, unlike in our model, the tree prior of the nCRP-VAE structure and its neural networks parameters are not learned jointly but alternatively. Moreover, the nCRP prior is highly complex, requiring multiple distributions and numerous probabilistic parameters as well as a customized variational inference method.

LTVAE: The Latent-Tree VAE [14] is a variational autoencoder that learns a tree-structured Bayesian network as its prior. The tree is composed of two levels: A set of super variables and their children. Each super-variable and its children form a single Gaussian mixture model. The structure is not learned jointly with the variational autoencoder but rather alternatively. The VAE is first fully trained for multiple epochs, then the structure itself is modified, and the whole process is sequentially repeated until a satisfactory Bayesian information criterion (BIC) score is achieved.

Growth and pruning: Other models that learn unsupervised hierarchical clusters have to incorporate growth and pruning procedures to control the shape of the learned hierarchy [14]–[16], [28]. The rationale is that regions of the latent

space that include a larger portion of the input distribution should be represented by more nodes in the learned hierarchy. Conversely, sparser regions need fewer nodes for an equivalent representation level. Various metrics and hyper-parameters are used to empirically control this growth, such as the BIC [14] or the mean quantization error [28]. As the approximate posterior distribution changes over time during training of the neural network, it can be difficult to control this growth. Indeed, a branch can become superfluously large if the corresponding region of the latent space becomes sparser than it was at previous training steps. Our model elegantly solves this problem, since each level of the hierarchy is self-organizing. Thus it can easily match the density of the approximate posterior. Moreover since the parenthood of nodes is defined by their distance between each other at different levels, a node may mechanistically switch from a branch to another without any need of explicitly implementing rules to control it. These properties of our model enable it to self-adjust and to learn even complex imbalanced hierarchies in order to match the true data distribution, while retaining a relative simplicity compared to other hierarchical models.

V. Experiments

Evaluation of HTVAE is perfomed on multiple datasets, including the MNIST [29] and FashionMNIST [30] benchmarks, the CIFAR100 [31] dataset which presents complex hierarchies of concepts, and a custom real-world physiological dataset of respiratory data recorded on ventilated patients presenting various breathing activity patterns. We show that the HTVAE is able to learn complex hierarchies in an unsupervised manner, and showcase how to incorporate prior knowledge using semi-supervised constraints on the respiratory dataset. Training was performed on a Tesla M60 GPU with 8GB of RAM. All cited models including competitors were implemented using Pytorch [32].

A. Metrics and Baselines

We evaluate the learned clustering using two metrics, namely purity and normalized mutual information (NMI). Given a set of clusters Ω, and $\mathcal{C}$ a set of classes, purity is defined as:

$$purity(\Omega, \mathcal{C}) = \frac{1}{N} \sum_k max_j |w_k \cap c_j| \tag{11}$$

Normalized Mutual Information measures to what extent the clustering learned in the latent space of the model is close to the true labels, and is penalized by the number of cluster whereas purity tends toward 1 as the number of clusters increases.

It relies on I, the mutual information, and H, the entropy, as defined below:

$$I(\Omega; \mathcal{C}) = \sum_{\omega_k \in \Omega} \sum_{c_j \in \mathcal{C}} P(\omega_k \cap c_j) log \frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)} \tag{12}$$

$$H(\Omega) = - \sum_{\omega_k \in \Omega} P(\omega_k) log P(\omega_k) \tag{13}$$

and is itself defined as:

$$NMI(\Omega, \mathcal{C}) = \frac{2 * I(\Omega; \mathcal{C})}{[H(\Omega) + H(\mathcal{C}]} \tag{14}$$

We also introduce a graph-based metric to measure how much a learned hierarchy differs from a given (true) one (for instance, the hierarchy of classes). For that purpose we rely on the notion of topological distance between two nodes in an oriented graph (length of the shortest path between two nodes ancestor). Let (V_S, E_S) be the directed acyclic graph obtained after learning HTVAE using sample $\mathbf{X}$. V_S is the set of nodes defined in all the self-organizing maps while E_S is the set of edges between nodes. Note that each node refers to a code. Denote $D_S^{\mathbf{X}}$ the $N \times N$ matrix of the topological distance between the nodes (codes) in S_1 that are associated to the datapoints $x \in \mathbf{X}$. Similarly, denote $D_{\mathcal{H}}^{\mathbf{X}}$, the $N \times N$ topological distance matrix between the nodes of the desired hierarchy associated to datapoints of $\mathbf{X}$.

$$D_{tree}^{\mathbf{X}}(S, \mathcal{H}) = \frac{1}{N^2} \| D_S^{\mathbf{X}} - D_{\mathcal{H}}^{\mathbf{X}} \|_F^2, \tag{15}$$

where $\| \cdot \|_F$ is the Frobenius norm.

We compare our model to several baselines, all of which are described in section IV. Our first baseline is the SOM-VAE [17] which learns a single non-probabilistic self-organizing map over the latent space. We also use two hierarchical VAE baselines, the VAE-HKM and the VAE-RGMM which were used as baselines in prior work on hierarchical VAEs [16]. We did not compare with selected hierarchical models including the LTVAE [14] whose comparatively shallow hierarchical structure we felt would not be as amenable to compare with our deep hierarchies. We were unable to get the published code working for some of the hierarchical approaches previously mentioned and were therefore unable to use them as baselines. All experiments were done using our own implementations of the models above.

B. MNIST and FashionMNIST

Dataset: We trained our model and the baselines on the MNIST digit dataset [29], randomly split into 59400 training samples and 600 validation samples. The models were evaluated on the predefined 10000 samples testing set. The same procedure was followed for the FashionMNIST dataset [30]. For the latter we defined a coarser set of super-classes over the standard fine labels provided by the authors. These classes partition the original 10 classes into 6 classes including light tops, heavy tops, bottoms, bags, dresses, and boots which are themselves partitioned into two coarser super-categories: clothes and accessories. These super-classes form a 4-level deep tree (including the root) which we define as the true tree in all metrics that require one.

Experimental Setting: All experiments used a convolutional autoencoder with four convolutional layers in total and ReLU activations for all layers except the last for which sigmoid is used. The encoder uses maxpooling layers to reduce the input size and the decoder uses bilinear upsampling to upsize the latent encodings. The models were trained for 20 epochs with

MNIST			
	Purity	NMI	D_{tree}
SOM-VAE	0.86 ± 0.01	0.47 ± 0.004	N.A
VAE-HKMeans	0.90 ± 0.002	0.47 ± 0.001	4.68 ± 0.005
VAE-RGMM	0.92 ± 0.003	0.49 ± 0.004	4.66 ± 0.013
HTVAE	$\mathbf{0.95 \pm 0.003}$	$\mathbf{0.54 \pm 0.006}$	$\mathbf{4.61 \pm 0.033}$
FashionMNIST			
	Purity	NMI	D_{tree}
SOM-VAE	0.73 ± 0.008	0.44 ± 0.007	N.A
VAE-HKMeans	0.79 ± 0.005	0.43 ± 0.001	3.58 ± 0.010
VAE-RGMM	0.79 ± 0.009	0.44 ± 0.003	3.59 ± 0.051
HTVAE	$\mathbf{0.81 \pm 0.004}$	$\mathbf{0.48 \pm 0.003}$	$\mathbf{3.49 \pm 0.055}$

TABLE I
EVALUATION METRICS FOR THE MNIST AND FASHIONMNIST DATASETS: PURITY SCORE, NORMALIZED MUTUAL INFORMATION (NMI) AND TOPOLOGICAL DISTANCE BETWEEN THE HTVAE HIERARCHY AND THE TRUE ONE.

the Adam optimizer [20] with a learning rate of 5×10^{-4} and a batch size of 32.

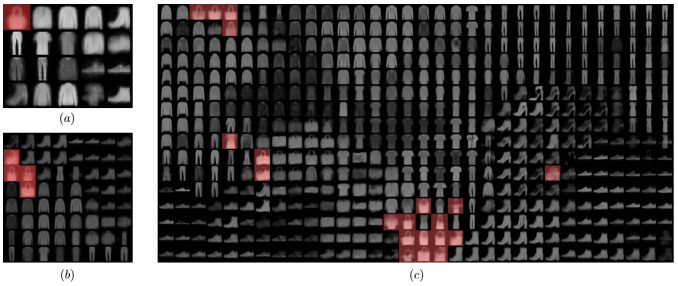

Fig. 1. The HTVAE learns a 4-level tree in the latent space including a virtual root and three levels (a, b, c). A single class at level (a) is highlighted and its children are similarly highlighted in the deeper level. The HTVAE was able to learn a hierarchy of items without supervision.

Comments: The learned self-organizing maps visible in Fig. 1 as well as the proportion of classes in each level of the tree display the model's ability to learn the implicit structure present in the data in a completely unsupervised manner. Table I shows the unsupervised accuracy and $D_{tree}$of the models on the MNIST and FashionMNIST datasets averaged over 5 runs. Our model HTVAE achieves the best accuracy with up to a 3% improvement over the second best method. It also learns a tree in the latent space that is closest to that of the true hierarchy of the data. In addition it can be seen from Fig.

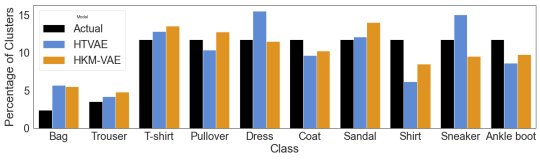

Fig. 2. We train our model on an artificially imbalanced subset of FashionMNIST. The learned tree is able to match the density of the data closely without explicit growth or pruning procedures. We also show the results for the HKM-VAE; however since the HKM component is not used as prior nor learned jointly with the VAE, it is only shown as reference.

2 that when introducing an artificial imbalance in the classes, our method learns a hierarchical prior that matches the density of the dataset well. This supports our analysis in section IV that the topological nature of each hierarchical level is able to match the true hierarchy of an input dataset without explicit growth strategies.

C. CIFAR-100

Dataset: CIFAR-100 [31] is a dataset of 60000 32x32 colour images, evenly split between 100 classes. These classes are themselves split between 20 super-classes, which grants the dataset hierarchical properties amenable to being learned by our model. We use the pre-defined training set (50000 samples) split into sets of 49400 and 600 for training and validation. For testing, we use the predefined testing set of 10000 samples.

Experimental Setting: We use a similar architecture as previously.

	Purity	NMI	D_{tree}
SOM-VAE	0.09 ± 0.005	0.21 ± 0.023	N.A
VAE-HKMeans	0.15 ± 0.005	0.31 ± 0.004	4.80 ± 0.023
VAE-RGMM	0.16 ± 0.007	0.32 ± 0.006	4.96 ± 0.058
HTVAE	$\mathbf{0.19 \pm 0.004}$	$\mathbf{0.37 \pm 0.004}$	$\mathbf{4.62 \pm 0.047}$

TABLE II
EVALUATION METRICS FOR THE CIFAR-100 DATASET.

Comments: Table II shows that our model achieves good clustering scores and learns a tree significantly closer to the right labeling than the other methods. These satisfactory results allow considering the application of our approach to a real-life problem: breathing data sequences.

D. Flow and Pressure Ventilation Dataset

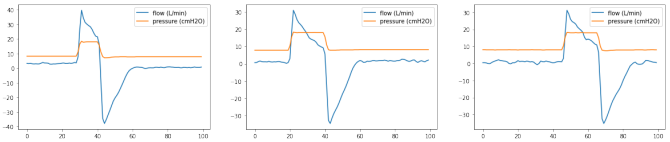

Fig. 3. Three random breaths from the dataset, displaying various amplitudes and lengths of inspiration/expiration.

Dataset: We evaluate our model on breathing data from people using a medical ventilator. The dataset includes one-hour long recordings of the flow and pressure signals of three subjects using the same model ventilator. The participants were asked to perform various breathing patterns and movements at different times of the recording session. These events were classified into ten categories: normal breathing, speaking, coughing, snoring, apnea, lying down, turning to the left, turning to the right, lying down with their knees to their chest, and swallowing. These classes were further aggregated into three super-classes representing normal breathing, respiratory events, and movement/positioning. These patterns were selected to reflect the normal behaviour of a ventilated patient and affect both the pressure and flow recordings, as well as their respiratory parameters (lung compliance and resistance). In order to train our model, we split the recordings into a set of 1000 cycles, each cycle comprising a single breath (inspiration followed by

expiration) and padded to 200 points using their edge values. The data of two of the participants was used as training data and that of the third participant for testing.

Experimental Setting: We use the same architecture as in previous experiments, modified to use temporal (1D, multi-channel) convolutions.

Comments: The approach reaches 39.5% unsupervised accuracy when trained on the model without any constraints and 65% with the must-link loss term. With this semi-supervised loss term, expert knowledge is introduced as a soft prior enables to learn more useful groupings that eventually provide a way to generate data more appropriate to the user's application. Moreover it enables the sampling of recordings at any level of the tree which allows the user to create datasets with a specific balance of classes. For example one might want a dataset composed of 50% normal breathing, and 50% apneas and snoring for application that require the detection of problematic events, in which case the user can sample half the samples from the first level of the tree in a coarse manner, and the other half from the second with a finer grained control.

VI. Conclusion

The proposed method learns interpretable hierarchical structures in the latent space of a VAE, enabling fine-grained generation of realistic data, and granular representations at different levels of the hierarchy. HTVAE is able to learn imbalanced hierarchies that match the input distribution in an unsupervised manner. The prior is learned jointly with the neural network parameters, and this does not require neither an explicit growth/pruning methodology, nor a large number of hyper-parameters. On 3 benchmarks, HTVAE achieves competitive results in terms of clustering ability and quality of the learned hierarchies. The application on a real-world application (respiratory signal generation) shows the relevance of multi-level generative architectures and constraints integration.

Acknowledgment

The authors are thankful to ANRT for participating to the funding of this project.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS) 27*, 2014.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR*, 2014.

[3] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *CoRR*, vol. abs/1605.08803, 2016.

[4] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework." in *Proc. of ICLR*, 2017.

[5] F. Falck, H. Zhang, M. Willetts, G. Nicholson, C. Yau, and C. C. Holmes, "Multi-facet clustering variational autoencoders," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8676–8690, 2021.

[6] A. F. Ansari and H. Soh, "Hyperprior induced unsupervised disentanglement of latent representations," in *Proc. of the AAAI Conf. on Artificial Intelligence*, 2019.

[7] S. K. Ainsworth, N. J. Foti, A. K. C. Lee, and E. B. Fox, "oi-vae: Output interpretable vaes for nonlinear group factor analysis." in *ICML*, ser. Proc. of Machine Learning Research, J. G. Dy and A. Krause, Eds., 2018.

[8] C. Esteban, S. L. Hyland, and G. Rtsch, "Real-valued (medical) time series generation with recurrent conditional gans." *CoRR*, 2017.

[9] C. K. Sonderby, T. Raiko, L. Maaloe, S. K. Sonderby, and O. Winther, "Ladder variational autoencoders." in *Adv. on Neural Information Processing Systems (NeurIPS)*, 2016.

[10] S. Zhao, J. Song, and S. Ermon, "Learning hierarchical features from generative models." *CoRR*, 2017.

[11] M. Tschannen, O. Bachem, and M. Lucic, "Recent advances in autoencoder-based representation learning." *arXiv preprint*, vol. arXiv:1812.05069, 2018.

[12] D. P. de Mello, R. M. Assunção, and F. Murai, "Top-down deep clustering with multi-generator gans," in *Proc. of the AAAI Conf. on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7770–7778.

[13] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," in *Proc. of the 36^{th} Int. Joint Conf. on Artificial Intelligence, IJCAI-17*, 2017, pp. 1965–1972.

[14] X. Li, Z. Chen, L. K. M. Poon, and N. L. Zhang, "Learning latent superstructures in variational autoencoders for deep multidimensional clustering." in *Proc. of ICLR*, 2019.

[15] P. Goyal, Z. Hu, X. Liang, C. Wang, and E. P. Xing, "Nonparametric variational auto-encoders for hierarchical representation learning." *CoRR*, 2017.

[16] S.-J. Shin, K. Song, and I.-C. Moon, "Hierarchically clustered representation learning." in *Proc. of AAAI*, 2020.

[17] V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch, "Somvae: Interpretable discrete representation learning on time series," in *7th Int. Conf. on Learning Representations (ICLR)*, May 2019.

[18] A. Van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems (NeurIPS) 30*, 2017.

[19] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of the 3rd Int. Conf. on Learning Representations (ICLR)*, 2015.

[21] T. A. Lampert, T.-B.-H. Dao, B. Lafabregue, N. Serrette, G. Forestier, B. Crmilleux, C. Vrain, and P. Ganarski, "Constrained distance based clustering for time-series: a comparative and experimental study." *Data Min. Knowl. Discov.*, 2018.

[22] K. Wagsta, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," in *Proc. of 18th Int. Conf. on Machine Learning (ICML-01)*, 2001.

[23] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning a mahalanobis metric from equivalence constraints." *Journal of Machine Learning Research*, vol. 6, p. 937965, 2005.

[24] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *ICML '04: Proc. of the 21^{rst} Int. Conference on Machine learning*, 2004.

[25] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *2006 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 2161–2168.

[26] D. M. Blei, T. L. Griffiths, and M. I. Jordan, "The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies." *Journal of the ACM*, vol. 57, no. 2, 2010.

[27] C. Wang and D. M. Blei, "Variational inference for the nested chinese restaurant process." in *NIPS*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds., 2009.

[28] M. Dittenbach, D. Merkl, and A. Rauber, "The growing hierarchical self-organizing map." in *Proc. of IJCNN (6)*, 2000.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha, "Gradient-Based Learning Applied to Document Recognition," *Proc. of the IEEE*, vol. 86, no. 11, p. 22782324, 1998.

[30] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arxiv preprint*, 2017.

[31] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-100 (canadian institute for advanced research)," 2009.

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, and T. e. a. Killeen, "Pytorch: An imperative style, high-performance deep learning library," in *Adv. in Neural Information Processing Systems 32*, 2019.