Nonparametric Variational Auto-encoders for Hierarchical Representation Learning

Prasoon Goyal[1] Zhiting Hu[1,2] Xiaodan Liang[1] Chenyu Wang[2] Eric P. Xing[1,2]
[1]Carnegie Mellon University [2]Petuum Inc.
{prasoongoyal13,chenyu.wanghao}@gmail.com, {zhitingh,xiaodan1,epxing}@cs.cmu.edu

Abstract

The recently developed variational autoencoders (VAEs) have proved to be an effective confluence of the rich representational power of neural networks with Bayesian methods. However, most work on VAEs use a rather simple prior over the latent variables such as standard normal distribution, thereby restricting its applications to relatively simple phenomena. In this work, we propose hierarchical nonparametric variational autoencoders, which combines tree-structured Bayesian nonparametric priors with VAEs, to enable infinite flexibility of the latent representation space. Both the neural parameters and Bayesian priors are learned jointly using tailored variational inference. The resulting model induces a hierarchical structure of latent semantic concepts underlying the data corpus, and infers accurate representations of data instances. We apply our model in video representation learning. Our method is able to discover highly interpretable activity hierarchies, and obtain improved clustering accuracy and generalization capacity based on the learned rich representations.

1. Introduction

Variational Autoencoders (VAEs) [11] are among the popular models for unsupervised representation learning. They consist of a standard autoencoder component, that embeds the data into a latent code space by minimizing reconstruction error, and a Bayesian regularization over the latent space, which enforces the posterior of the hidden code vector matches a prior distribution. These models have been successfully applied to various representation learning tasks, such as sentence modeling [3, 8] and image understanding [19, 5].

However, most of these approaches employ a simple prior over the latent space, which is often the standard normal distribution. Though convenient inference and learning is enabled, converting the data distribution to such fixed, single-mode prior distribution can lead to overly simplified representations which lose rich semantics present in the data. This is especially true in the context of unsupervised learning where large amount of available data with complex hidden structures is of interest which is unlikely to be presented in the restricted latent space. For example, a large video corpus can encode rich human activity with underlying intricate temporal dependencies and hierarchical relationships. For accurate encoding and new insights into the datasets, it is desirable to develop new representation learning approaches with great modeling flexibility and structured interpretability.

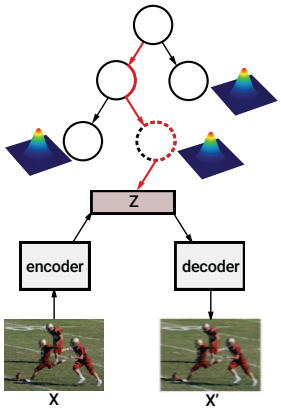

Figure 1. Illustration of the nonparametric hierarchical variational autoencoder. We combines hierarhical Bayesian nonparametric priors with variational autoencoders.

In this paper, we propose *hierarchical nonparametric variational autoencoders*, which combines Bayesian nonparametric priors with VAEs. Bayesian nonparametric methods as the code space prior can grow information capacity with the amount and complexity of data, which endows great representational power of the latent code space. In particular, we employ nested Chinese Restaurant Process (nCRP) [2], a stochastic process allowing infinitely deep and branching trees for representing the data. As opposed to fixed prior distributions in previous work, we learn both the VAE parameters and the nonparametric priors *jointly* from the data, for self-calibrated model capacity. The induced tree hierarchies serve as an aggregated structured representation of the whole corpus, summarizing the gist for convenient navigation and better generalization. On the other hand, each data instance is assigned with a probabil-

2380-7504/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCV.2017.545

ity distribution over the paths down the trees, from which an *instance-specific* prior distribution is induced for regularizing the instance latent code. Figure 1 gives a schematic overview of our approach.

The resulting model unifies the Bayesian nonparametric flexibility with neural inductive biases, by viewing it as a nonparametric topic model [1] on the latent code space, in which raw data examples are first transformed to compact (probabilistic) semantic vectors with deep neural networks (i.e., the encoder networks). This enables invariance to distracting transformations in the raw data [12, 4], resulting in robust topical inference. We derive variational inference updates for estimating all parameters of the neural autoencoder and Bayesian priors jointly. A tailored split-merge process is incorporated for effective exploration of the unbounded tree space.

Our work is the first to combine tree-structured BNPs and VAE neural models in a unified framework, with all parameters learned jointly. From the VAE perspective, we propose the first VAE extension that learns priors of the latent space from data. From the BNP perspective, our model is the first to integrate neural networks for efficient generation and inference in Dirichlet process models.

We present an application on video corpus summarization and representation learning, in which each video is modeled as a mixture of the tree paths. Each frame in the video is embedded to the latent code space and attached to a path sampled from the mixture. The attachment dynamics effectively clusters the videos based on sharing of semantics (e.g., activities present in the video) at multiple level of abstractions, resulting in a hierarchy of abstract-to-concrete activity topics. The induced rich latent representations can enable and improve a variety of downstream applications. We experiment on video classification and retrieval, in which our model obtains superior performance over VAEs with parametric priors. Our method also shows better generalization on test set reconstruction. Qualitative analysis reveals interpretability of the modeling results.

We begin by reviewing related work in §2. We then present our approach in the problem setting of learning hierarchical representations of sequential data (e.g., videos). §3 describes the problem and provides background on the nCRP prior. §4 develops our nonparametric variational autoencoders and derives variational inference for joint estimation of both the neural parameters and Bayesian priors. In §5 we apply the model for video representation learning. §6 shows quantitative and qualitative experimental results. We conclude the paper in §7.

2. Related Work

Variational autoencoders and variants. Variational Autoencoders (VAEs) [11] provide a powerful framework for deep unsupervised representation learning. VAEs consist of

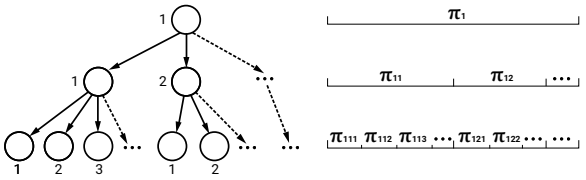

Figure 2. **Left**: a sample tree structure draw from nCRP. **Right**: The respective tree-based stick-breaking construction. The stick length of the root node is $\pi_1 = 1$. Each node performs a stick-breaking process on its stick segment to construct its children.

encoder and decoder networks which encode a data example to a latent representation and generate samples from the latent space, respectively. The model is trained by minimizing an expected reconstruction error of observed data under the posterior distribution defined by the encoder network, and at the same time regularizing the posterior of the hidden code to be close to a prior distribution, by minimizing the KL divergence between the two distributions. Vanilla VAEs typically use a standard normal distribution with zero mean and identity covariance matrix as the prior, which enables closed-form optimization while restricting the expressive power of the model. Adversarial autoencoders [13] replace the KL divergence with an adversarial training criterion to allow richer families of priors. Our work differs in that we compose VAEs with Bayesian nonparametric methods for both flexible prior constraints of individual instances and structured representation induction of the whole corpus. Previous research has combined VAEs with graphical models in different context. Siddharth et al., [16] replace the encoder networks with structured graphical models to enable disentangled semantics of the latent code space. Johnson et al., [10] leverage the encoder networks to construct graphical model potentials to avoid feature engineering. Our work is distinct as we aim to combine Bayesian nonparametric flexibility with VAEs, and address the unique inferential complexity involving the hierarchical nonparametric models. Other VAE variants that are orthogonal to our work are proposed. Please refer to [9] for a general discussion of VAEs and their connections to a broad class of deep generative models.

Bayesian nonparametric methods. Bayesian nonparametric methods allow infinite information capacity to capture rich internal structure of data. For example, mixture models with Dirichlet process priors can be used to cluster with an unbounded number of centers. A few recent works have developed powerful hierarchical nonparametric priors [2, 7] to induce tree structures with unbounded width and depth. Nested Chinese Restaurant Process (nCRP) assigns data instances with paths down the trees. The attachment dynamics lead to hierarchical clustering of the data where high-level clusters represent abstract semantics while low-level clusters represent concrete content. We leverage

nCRP as the prior over the latent code space for enhanced representational power. Gaussian processes [15] are another line of Bayesian nonparametric approach which has been incorporated with deep neural networks for expressive kernel learning [21, 6]. These methods have typically been applied in supervised setting, while we are targeting on unsupervised representation learning using hierarchical Dirichlet nonparametrics.

3. Preliminaries

For concreteness, we present our approach in the problem setting of unsupervised hierarchical representation learning of sequential data. We start by describing the problem statement, followed by an overview of nCRP. All the notations used in the paper have been consolidated in Table 1 for quick reference.

3.1. Problem Description

Let $\boldsymbol{x}^m = (x_{mn})_{n=1}^{N_m}$ denote a sequence $\boldsymbol{x}^m$ of length N_m with n^{th} element denoted as x_{mn}. Given unlabeled sequences $\{\boldsymbol{x}^m\}$ of data, we want to learn compact latent representation for each instance as well as capture the gist of the whole corpus. To this end, we build a generative probabilistic model that assigns high probability to the given data. Further, to capture rich underlying semantic structures, we want the probabilistic model to be hierarchical, that is, coarse-grained concepts are higher up in the hierarchy, and fine-grained concepts form their children.

For instance, video data can be modeled as above, wherein each video can be represented as a sequence $\boldsymbol{x}^m$. Each element x_{mn} of the sequence is a temporal segment of the video, such as a raw frame or sub-clip of the video, or some latent representation thereof. In such data, the hierarchy should capture high-level activities, such as, "playing basketball" higher up in the hierarchy, while more fine-grained activities, such as "running" and "shooting" should form its children nodes. These hierarchies can then be used for a wide variety of downstream tasks, such as, video retrieval, summarization, and captioning.

3.2. Nested Chinese Restaurant Process

We use nCRP priors [2], which can be recursively defined in terms of Dirichlet process (DP). A draw from a Dirchlet process $DP(\gamma, G_0)$ is described as

$$\begin{aligned} v_i &\sim Beta(1,\gamma), \qquad \pi_i = v_i \prod_{j=1}^{i-1}(1-v_j) \\ w_i &\sim G_0, \qquad\qquad G = \sum_{i=1}^{\infty} \pi_i \delta_{w_i} \end{aligned} \tag{1}$$

Here, γ is the scaling parameter, G_0 is the base distribution of the DP, and δ_w is an indicator function that takes value 1 at w and 0 otherwise. The above construction admits an intuitive stick-breaking interpretation, in which, a unit length stick is broken at a random location, and π_1 is the length of the resulting left part. The right part is broken further, and the length of left part so obtained is assigned to π_2. The process is continued to infinity. Note that, $\sum_{i=1}^{\infty} \pi_i = 1$. Therefore, a draw from a DP defines a discrete probability distribution over a countably infinite set.

Symbol	Description
$(\boldsymbol{x})_N$	a sequence of length N, with elements $\boldsymbol{x_1}, \ldots, \boldsymbol{x}_N$
$\boldsymbol{x}_{mn}$	n^{th} element of sequence $(\boldsymbol{x}^m)_N$
$\boldsymbol{z}_{mn}$	the latent code corresponding to $\boldsymbol{x}_{mn}$
$par(p)$	the parent node of node p
$\boldsymbol{\alpha}_p$	the parameter vector for node p
$\boldsymbol{\alpha}^*$	the prior parameter over α_p for the root node
σ_N	the variance parameter for node parameters
σ_D	the variance parameter for data
v_{me}	the nCRP variable for m^{th} sequence on edge e
$\boldsymbol{V}_m$	the set of all v_{me} for m^{th} sequence
γ^*	the prior parameter shared by all v_{me}
c_{mn}	the path assignment for data point $\boldsymbol{x}_{mn}$
c_{mn}	the path assignment for data point $\boldsymbol{x}_{mn}$
$\boldsymbol{\mu}_p, \sigma_p$	parameters for variational distribution of α_p
$\gamma_{me,0}, \gamma_{me,1}$	parameters for variational distribution of v_{me}
$\boldsymbol{\phi}_{mn}$	parameter for variational distribution of c_{mn}

Table 1. Notations used in the paper

The above process can be extended to obtain nCRP, or equivalently, a tree-based stick-breaking process, in which, we start at the root node (level 0), and obtain probabilities over its child nodes (level 1) using a DP. Then we recursively run a DP on each level 1 node to get probabilities over level 2 nodes, and so on. This defines a probability distribution over paths of an infinitely wide and infinitely deep tree. Figure 2 gives an illustration of the process. More formally, we label all the nodes recursively using a sequence of integers – the root node has label '1', its children nodes have labels '11', '12', ..., children nodes of '11' have labels '111', '112', and so on. Now, we can assign probability to every node p based on draws of stick-breaking weights v as follows:

- For the root node (level 0), $\pi_1 = 1$
- For i^{th} node at level 1, $\pi_{1i} = \pi_1 v_{1i} \Pi_{j=1}^{i}(1 - v_{1j})$.
- For j^{th} child at level 2 of i^{th} level-1 node,
$$\pi_{1ij} = \pi_1 \pi_{1i} v_{1ij} \Pi_{k=1}^{j}(1 - v_{1ik}).$$

This process is repeated to infinity. Please refer to [2, 20] for more details.

4. Hierarchical Nonparametric Variational Autoencoders

We first give a high-level overview of our framework, and then describe various components in detail.

Formally, a VAE takes in an input $\boldsymbol{x}$, that is passed through an encoder with parameters $\boldsymbol{\phi}$ to produce a distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ over the latent space. Then, a latent code

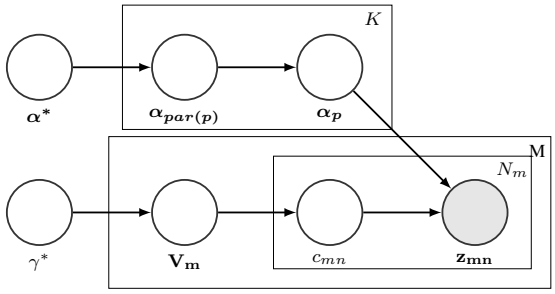

Figure 3. The proposed generative model. This diagram only shows the BNP component. Thus, the latent codes z_{mn} that are learnt in VAE training are treated as observations.

$\boldsymbol{z}$ is sampled from this distribution, and passed through a decoder to obtain the reconstructed data point $\tilde{\boldsymbol{x}}$. Thus, minimizing the reconstruction error amounts to maximizing $\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$, where $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ corresponds to the decoder parameterized by $\boldsymbol{\theta}$. The encoder and the decoder can be arbitrary functions; however, they are typically modeled as neural networks. Further, a prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is imposed on the latent space. Thus we want to solve for parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, which, using the standard variational inference analysis gives the following lower bound on the data likelihood:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^m) &\geq \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x}^m)\\ &= \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^m)}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^m|\boldsymbol{z})]\\ &\quad - D_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^m)\|p_{\boldsymbol{\theta}}(\boldsymbol{z}))\end{aligned} \tag{2}$$

Therefore, the prior and the decoder together act as the generative model of the data, while the encoder network acts as the inference network, mapping data to posterior distributions over the latent space. Typically, the prior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is assumed to be standard normal distribution $\mathcal{N}(0, I)$, which implies that maximizing the above lower bound amounts to optimizing only the neural network parameters, since in that case, the prior is free of parameters.

In this work, we use a much richer prior, namely the nCRP prior described in 3.2. This allows growing information capacity of the latent code space with the amount and complexity of data, and thus obtains accurate latent representations. The tree-based prior also enables automatic discovery of rich semantic structures underlying the data corpus. To this end, we need to jointly optimize for the neural network parameters and the parameters of the nCRP prior. We make use of alternating optimization, wherein we first fix the nCRP parameters and perform several backpropagation steps to optimize for the neural network parameters, and then fix the neural network, and perform variational inference updates to optimize for the nCRP parameters.

We next describe the nCRP-based generative model and variational inference updates.

4.1. Generative Model

The generative model assumes a tree with infinite depth and branches, and generates data sequences through root-to-leaf random walks along the paths of the tree. Each node p has a parameter vector $\boldsymbol{\alpha}_p$ which depends on the parameter vector of the parent node to encode the hierarchical relation. That is, for every node p of the tree, draw a D-dimensional parameter vector $\boldsymbol{\alpha}_p$, according to

$$\boldsymbol{\alpha}_p \sim \mathcal{N}(\boldsymbol{\alpha}_{par(p)}, \sigma_N^2 I) \tag{3}$$

where $par(p)$ denotes the parent node of p, and σ_N is a variance parameter shared by all nodes of the tree. For the root node, we define $\boldsymbol{\alpha}_{par(p)} = \boldsymbol{\alpha}^*$, for some constant vector $\boldsymbol{\alpha}^*$.

Each data sequence $\boldsymbol{x}^m$ is modeled as a mixture of the paths down the tree, and each element x_{mn} is attached to one path sampled from the mixture. Specifically, $\boldsymbol{x}^m$ is drawn as follows (Figure 3 gives the graphical model representation):

1. For each edge e of the tree, draw $v_{me} \sim Beta(1, \gamma^*)$. We denote the collection of all v_{me} for sequence m as $\boldsymbol{V}_m$. This defines a distribution over the paths of the tree, as described in section 3.2. Let $\pi(\boldsymbol{V}_m)$ denote the probabilities assigned to each leaf node through this process.
2. For each element $\boldsymbol{x}_{mn}$ in $\boldsymbol{x}^m$, draw a path c_{mn} according to the multinomial distribution $Mult(\pi(\boldsymbol{V}_m))$.
3. Draw the latent representation vector $\boldsymbol{z}_{mn}$ according to $\mathcal{N}(\boldsymbol{\alpha}_{c_{mn}}, \sigma_D^2 I)$ which is the emission distribution defined by the parameter associated in the leaf node of path c_{mn}. Here σ_D is a variance parameter shared by all nodes.

This process generates a latent code $\boldsymbol{z}_{mn}$, which is then passed through the decoder to get the observed data $\boldsymbol{x}_{mn}$. To summarize the above generative process, the node parameters of the tree depend on their parent node, and the tree is shared by the entire corpus. For each sequence, draws $\boldsymbol{V}_m$ define a distribution over the paths of the tree. For each element of the sequence, a path is sampled according to the above distribution, and finally, the data element is drawn according to the node parameter of the sampled path.

Our goal is to estimate the parameters of the tree model, including the node parameters $\boldsymbol{\alpha}_p$, sequence-level parameters $\boldsymbol{V}_m$, and path assignments c, as well as the neural parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, given the hyperparameters $\{\boldsymbol{\alpha}^*, \gamma^*, \sigma_N, \sigma_D\}$ and the data.

4.2. Parameter Learning

In this section, we first describe the variational inference updates for estimating the parameters of nCRP prior (section 4.2.1), and the update equations for our neural network

parameters (section 4.2.2). Finally, we describe a procedure for joint optimization of the nCRP prior parameters and the neural network.

4.2.1 Variational Inference

Using the mean-field approximation, we assume the following forms of the variational distributions:

- For each node p of the tree, the parameter vector $\boldsymbol{\alpha}_p$ is distributed as $\boldsymbol{\alpha}_p \sim \mathcal{N}(\boldsymbol{\mu}_p, \sigma_p^2 I)$, where $\boldsymbol{\mu}_p$ is a D-dimensional vector and σ_p is a scalar.
- For sequence m, the DP variable at edge e, v_{me} is distributed as $v_{me} \sim Beta(\gamma_{me,0}, \gamma_{me,1})$, where $\gamma_{me,0}$ and $\gamma_{me,1}$ are scalars.
- For data $\boldsymbol{x}_{mn}$, the path assignment variable c_{mn} is distributed as $c_{mn} \sim Mult(\boldsymbol{\phi}_{mn})$, where the dimension of $\boldsymbol{\phi}_{mn}$ is equal to the number of paths in the tree.

We want to find optimal variational parameters that maximize the variational lower bound

$$\mathcal{L} = \mathbb{E}_q[\log p(W, X|\Theta)] - \mathbb{E}_q[\log q_\nu(W)] \tag{4}$$

where W denotes the collection of latent variables, $X = \{z_{mn}\}$ are the latent vector representations of observations, Θ are the hyperparameters, and $\nu = \{\boldsymbol{\mu}_p, \sigma_p, \gamma_{me,0}, \gamma_{me,1}, \boldsymbol{\phi}_{mn}\}$ are variational parameters. We use $p(W, X|\Theta)$ to denote the generative model described in section 4.1. We derive variational inference for a *truncated* tree [7, 20]. We achieve this by setting the components corresponding to all other paths of $\boldsymbol{\phi}_{mn}$ equal to 0 for all $m \in \{1, \ldots, M\}$ and $n \in \{1, \ldots, N_m\}$. We later describe how we can dynamically grow and prune the tree during training. Thus, the generative distribution above simplifies to the following:

$$\begin{aligned} &p(W, X|\Theta) \\ &= \sum_p \log p(\boldsymbol{\alpha}_p|\boldsymbol{\alpha}_{par(p)}, \sigma_N) + \sum_{m,e} \log p(v_{me}|\gamma^*) \\ &+ \sum_{m,n} \log p(c_{mn}|\boldsymbol{V_m}) + \log p(\boldsymbol{z}_{mn}|\boldsymbol{\alpha}, c_{mn}, \sigma_D) \end{aligned} \tag{5}$$

Here, $p \in \{1, \ldots, P\}$ and $e \in \{1, \ldots, E\}$ index the paths and the edges of the truncated tree respectively. Note that the above truncation is nested. That is, for two trees T_1 and T_2 such that the set of nodes of T_1 is a subset of the set of nodes of T_2, the model generated from T_2 subsumes all possible configurations that can be generated from T_1.

Proceeding as in standard derivation of posterior estimate, we obtain the following variational updates:

$$q^*(\boldsymbol{\alpha}_p|\boldsymbol{\mu}_p, \sigma_p) \sim \mathcal{N}(\boldsymbol{\mu}_p, \sigma_p^2) \tag{6}$$

where, for a leaf node,

$$\frac{1}{\sigma_p^2} = \frac{1}{\sigma_N^2} + \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_m}\phi_{mnp}}{\sigma_D^2} \tag{7}$$

$$\boldsymbol{\mu}_p = \sigma_p^2 \cdot \left(\frac{\boldsymbol{\mu}_{par(p)}}{\sigma_N^2} + \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_m}\phi_{mnp}\boldsymbol{z}_{mn}}{\sigma_D^2}\right) \tag{8}$$

while for an internal node:

$$\frac{1}{\sigma_p^2} = \frac{1 + |ch(p)|}{\sigma_N^2} \tag{9}$$

$$\mu_p = \sigma_p^2 \cdot \left(\frac{\boldsymbol{\mu}_{par(p)} + \sum_{r \in ch(p)} \boldsymbol{\mu}_r}{\sigma_N^2}\right) \tag{10}$$

Here, $ch(p)$ denotes the set of all children of node p, and $|\cdot|$ denotes the cardinality of a set. Intuitively, for a leaf node, σ_p is small when we have many points (high ϕ_{mnp}) associated with this node, which corresponds to a good estimate of parameter $\boldsymbol{\alpha_p}$. The mean for a leaf node, $\boldsymbol{\mu}_p$, is a weighted mean of the latent codes of the data. For an internal node, the mean parameter, $\boldsymbol{\mu}_p$ is a simple average of all the child nodes (and the parent node). However, a child node with larger amount of data is farther from its parent node, and thereby has a greater effect on the mean implicitly.

$$q^*(v_{me}|\gamma_{me,0}, \gamma_{me,1}) \sim Beta(\gamma_{me,0}, \gamma_{me,1}) \tag{11}$$

where

$$\gamma_{me,0} = 1 + \sum_{n=1}^{N_m}\sum_{p:e\in p} \phi_{mnp} \tag{12}$$

$$\gamma_{me,1} = \gamma^* + \sum_{n=1}^{N_m}\sum_{p:e<p} \phi_{mnp} \tag{13}$$

Here, $e \in p$ denotes the set of all edges that lie on path p, while $e < p$ denotes the set of all edges that lie to the left of p in the tree.

$$q^*(c_{mn}|\boldsymbol{\phi}_{mn}) \sim Mult(\boldsymbol{\phi}_{mn}) \tag{14}$$

where

$$\begin{aligned} \phi_{mnp} \propto \exp\Bigg\{ &\sum_{e:e\in p} [\Psi(\gamma_{me,0}) - \Psi(\gamma_{me,0} + \gamma_{me,1})] \\ &+ \sum_{e:e<p} [\Psi(\gamma_{me,1}) - \Psi(\gamma_{me,0} + \gamma_{me,1})] \\ &- \frac{1}{2\sigma_D^2}\left[(\boldsymbol{z}_{mn} - \boldsymbol{\mu}_p)^T(\boldsymbol{z}_{mn} - \boldsymbol{\mu}_p) + \sigma_p^2\right] \Bigg\} \end{aligned} \tag{15}$$

Here, $\Psi(\cdot)$ is the digamma function.

4.2.2 Neural network parameter updates

The goal of neural network training is to maximize the following lower bound on the data log-likelihood function:

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^m)}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|p_{\boldsymbol{\theta}}(\boldsymbol{z})) \tag{16}$$

with respect to the neural network parameters. Note that ϕ denotes the parameters of the encoder network, while $\boldsymbol{\theta}$ denotes the parameters of the decoder network and the nCRP prior. Defining $\boldsymbol{\theta}_{NN}$ as the set of parameters of the decoder network, we need to learn parameters $\{\phi, \boldsymbol{\theta}_{NN}\}$.

The update equations for a parameter $\beta \in \{\phi, \boldsymbol{\theta}_{NN}\}$ is given by

$$\beta^{(t+1)} \leftarrow \beta^{(t)} + \eta \cdot \frac{\partial \mathcal{L}}{\partial \beta} \tag{17}$$

where the partial derivative is computed using backpropagation algorithm, while η is an appropriate learning rate.

4.2.3 Joint training

In order to jointly learn the nCRP parameters and the NN parameters, we employ alternating optimization, wherein, we first fix the nCRP prior parameters and perform several steps of NN parameter updates, and then fix the NN parameters and perform several steps of nCRP parameter updates. This enables the variational inference to use increasingly accurate latent codes to build the hierarchy, and the continuously improving hierarchy guides the neural network to learn more semantically meaningful latent codes.

4.3. Dynamically adapting the tree structure

Since our generative model is non-parametric, it admits growing or pruning the tree dynamically, depending on the richness of the data. Here, we list the heuristics we use for dynamically growing and pruning the tree. Note that each data point $\boldsymbol{x}_{mn}$ has soft assignments to paths, given by $\boldsymbol{\phi}_{mn}$. We use these soft assignments to make decisions about dynamically adapting the tree structure.

Growing the tree We define *weighted radius* of leaf node p as

$$r_p = \sqrt{\frac{\sum_{m=1}^{M}\sum_{n=1}^{N_m}\phi_{mnp}(\boldsymbol{z}_{mn}-\boldsymbol{\mu}_p)^T(\boldsymbol{z}_{mn}-\boldsymbol{\mu}_p)}{\sum_{m=1}^{M}\sum_{n=1}^{N_m}\phi_{mnp}}} \tag{18}$$

If the weighted radius r_p is greater than a threshold R, then we split the leaf node into K children nodes.

Pruning the tree For a leaf node p, we can compute the total fraction of the data assigned to this node as

$$f_p = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_m}\phi_{mnp}}{\sum_{m=1}^{M}N_m} \tag{19}$$

If the data fraction f_p is less than a threshold F, then the leaf node is eliminated. If an internal node is left with only one child, then it is replaced by the child node, thus effectively eliminating the internal node. The parameters R and K for growing the tree, and the parameter F for pruning the tree are set using the validation set.

5. Video Hierarchical Representation Learning

In this section, we describe how we can apply our proposed model to learn meaningful hierarchical representations for video data.

Consider an unlabeled set of videos. We want to build a hierarchy in which the leaf nodes represent fine-grained activities, while as we move up the hierarchy, we obtain more coarse-grained activities. For instance, a node in the hierarchy may represent "sports", its child nodes may represent specific sports, such as "basketball" and "swimming". The node "swimming" can, in turn, have child nodes representing "diving", "backstroke", etc.

To use the above framework, we treat each video as a discrete sequence of frames, by sampling frames from the video. Then, each frame is passed through a pre-trained convolutional neural network (CNN) to obtain frame features. The resulting frame features are then used as sequence elements $\boldsymbol{x}_{mn}$ in our framework. Note, however, that our framework is sufficiently general, and therefore, instead of using the frame features extracted from a pre-trained CNN, we can use the raw frames directly, or even model the video as a discrete sequence of subshots, instead of a discrete sequence of frames.

We optimize the neural and nCRP parameters jointly as described in section 4.2. This process gives us a posterior estimate of the nCRP parameters, which we can use to build a hierarchy for the corpus, as follows. We obtain a distribution $\mathcal{N}(\boldsymbol{\mu}_p, \sigma_p)$ for each node parameter $\boldsymbol{\alpha}_p$. Thus, we can pass a frame feature vector $\boldsymbol{x}$ through the trained encoder network to get a latent code $\boldsymbol{z}$, and then assign the frame to the path whose $\boldsymbol{\alpha}_p$ is closest to $\boldsymbol{z}$. Doing this for all frames results in each node being associated with the most representative frames of the activity the node represents.

6. Experiments

Here, we present quantitative and qualitative analysis of our proposed framework. It is worth pointing out that because of the unavailability of data labeled both with coarse-grained and fine-grained activities, we conduct quantitative analysis on the video classification task and video retrieval task, and qualitatively show the interpretable hierarchy generated by our non-parametric model.

6.1. Experimental Settings

Dataset We evaluate the models on TRECVID Multimedia Event Detection (MED) 2011 dataset [14], which consists of 9746 videos in total. Each video is labeled with one of 15 event classes or supplied as a background video without any assigned action label. In our experiments, we used only the labeled videos of MED dataset, where 1241 videos are used for training, 138 for validation and 1169 for testing. The mean length of these videos is about 3 minutes.

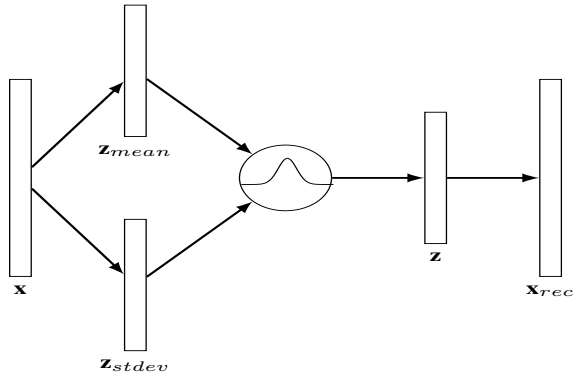

Figure 4. The neural network architecture. The input $\mathbf{x}$ is a 4096-dim VGG feature vector, that is mapped to 48-dim vectors $\mathbf{z}_{mean}$ and $\mathbf{z}_{stdev}$ using one fully connected layer each. A latent code $\mathbf{z}$ is then sampled from a Gaussian distribution defined by $\mathbf{z}_{mean}$ and $\mathbf{z}_{stdev}$, which is decoded to $\mathbf{x}_{rec}$ using one fully connected layer.

Algorithm	Mean test log-likelihood
VAE-StdNormal	-28886.90
VAE-nCRP	**-28438.32**

Table 2. Test-set log-likelihoods by our model "VAE-nCRP" and traditional variational autoencoder "VAE-StdNormal".

Feature extraction For each video, we extract one frame for every five seconds, resulting in 42393 frames for training, 5218 frames for validation, and 41144 frames for evaluation. Then, each frame is passed through a VGG-16 network [17] trained on ImageNet dataset. The output of the first fully-connected layer is used as the 4096-dimensional feature vector.

Neural network architecture Both the encoder and the decoder networks were multi-layer perceptions (MLPs). The detailed network is shown in Fig 4. In the alternating optimization procedure we performed one iteration of variational inference updates after every epoch of neural network training. We used RMSProp optimizer [18] with an initial learning rate of 0.01 and a decay rate of 0.98 per 1000 iterations. Our model converged in about 20 epochs.

6.2. Test Set Reconstruction

To better demonstrate the effectiveness of learning hierarchical prior distribution by our non-parametric VAE, we compare the test-set log likelihood of conventional VAE with our model. Formally, the log likelihood of data point $\boldsymbol{x}$ is given by $\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$, where $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ corresponds to the encoder network and $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ indicates the decoder network.

We computed the average log likelihood of the test set across 3 independent runs of variational autoencoders with standard normal prior, and with nCRP prior. We report the sum of log likelihood over all frames in the test set. The results are summarized in Table 2. Our model obtains a higher log likelihood, implying that it can better model the underlying complex data distribution embedded in natural diverse videos. This supports our claim that richer prior distributions are beneficial for capturing the rich semantics embedded in the data, especially for complex video content.

Category	K-Means	VAE-GMM	VAE-nCRP
Board_trick	44.6	**47.2**	31.3
Feeding_an_animal	**57.0**	42.5	53.8
Fishing	33.7	39.0	**48.9**
Woodworking	38.9	40.5	**60.8**
Wedding_ceremony	59.8	54.3	**63.6**
Birthday_party	6.5	7.4	**27.8**
Changing_a_vehicle_tire	31.9	39.7	**45.3**
Flash_mob_gathering	**43.4**	40.1	38.2
Getting_a_vehicle_unstuck	52.9	50.6	**65.9**
Grooming_an_animal	2.9	14.5	**17.3**
Making_a_sandwich	47.1	**54.7**	49.3
Parade	28.4	**33.8**	19.8
Parkour	4.5	19.8	**27.7**
Repairing_an_appliance	42.3	**58.6**	47.4
Sewing_project	1.6	**24.3**	18.4
Aggregate over all classes	34.9	39.1	**42.4**

Table 3. Classification Accuracy (%) on TRECVID MED 2011.

6.3. Video Classification

We compared our model (denoted as `VAE-nCRP`) with two clustering baselines, namely, K-Means clustering (denoted as `K-Means`) and variational autoencoders with Gaussian mixture model prior (denoted as `VAE-GMM`).

In order to evaluate the quality of the obtained hierarchy for the data, our model learns to assign an action label to each node (either leaf nodes or internal nodes) by taking a majority vote of the labels assigned to the data points. For each frame in the test data, we can obtain the latent representation of the frame feature, and then find the leaf node to which it is assigned by minimizing the Euclidean distance between the latent representation and the leaf node parameter $\boldsymbol{\alpha}_p$, and the predicted label of this frame is then given by the label assigned to this leaf node. The classification accuracy is then a measure of the quality of the hierarchy. Similarly, for other clustering baselines, we assign a label to each cluster, and then assign new data points to the closest cluster to predict their labels.

Note that, in applications, we would typically use the standard variational inference framework to find the path assignments, in which case, an unseen frame can also be assigned to a new path, exploiting the non-parametric nature of our model. However, for the purpose of our evaluations, we need to assign a label to each frame, and therefore, it must be assigned to one of the paths created during training. We would also like to point out that the hierarchy is

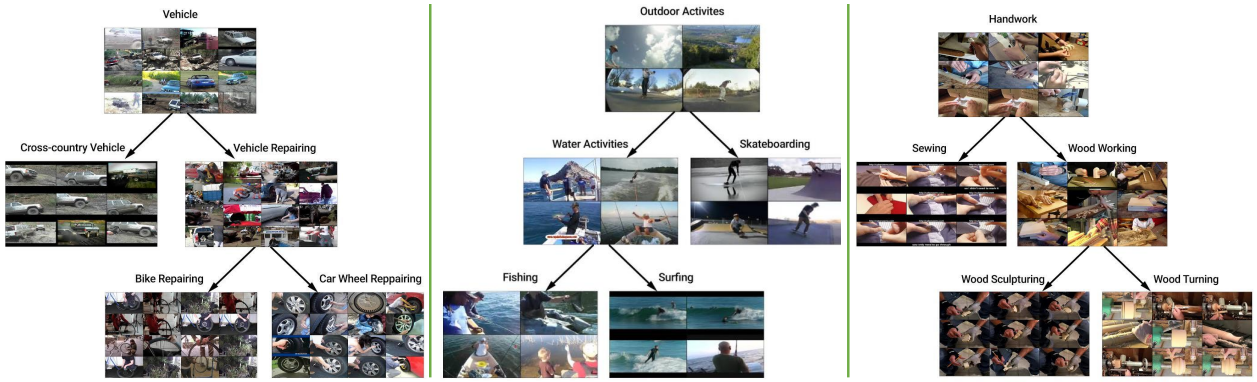

Figure 5. Some example hierarchical structures learned by our model.

constructed in a purely unsupervised manner, and the class labels are used only for evaluation.

We report the mean accuracy of each model, averaged over three independent runs. The results are summarized in Table 3. As can be seen, VAE-nCRP outperforms the baseline models on 8 out of 15 classes, and also has an overall highest accuracy. This suggests that the clusters formed by VAE-nCRP are more separated than those formed by K-means clustering and VAE-GMM.

6.4. Video retrieval

We also conduct experiments on video retrieval task to further verify the capability of our model. This task aims to retrieve all frames from the test set that belong to each class, which is closely related to video classification task. We report the F-1 scores of the models, which incorporates both the false positive rate and false negative rate. The results are summarized in Table 4. Again, it can be observed that VAE-nCRP outperforms the baseline models on 8 out of 15 classes, and achieves the highest overall F-1 score.

Category	K-Means	VAE-GMM	VAE-nCRP
Board_trick	32.1	**38.9**	32.1
Feeding_an_animal	33.7	33.8	**36.2**
Fishing	44.9	45.9	**59.9**
Woodworking	32.1	29.5	**38.0**
Wedding_ceremony	41.0	**51.2**	51.0
Birthday_party	14.0	11.0	**30.5**
Changing_a_vehicle_tire	38.3	45.5	**54.5**
Flash_mob_gathering	**45.8**	41.6	42.3
Getting_a_vehicle_unstuck	37.9	43.2	**56.9**
Grooming_an_animal	6.7	**21.5**	20.1
Making_a_sandwich	51.7	53.0	**60.3**
Parade	24.7	**37.7**	29.8
Parkour	6.8	28.2	**39.1**
Repairing_an_appliance	39.9	**41.2**	36.8
Sewing_project	1.7	**32.5**	25.8
Aggregate over all classes	32.4	38.5	**42.4**

Table 4. F-1 scores of video retrieval on TRECVID MED 2011.

6.5. Qualitative analysis

In addition to the quantitative analysis, we also performed a qualitative analysis of the hierarchy learned by our model as shown in Figure 5. We visualized the hierarchy structure by representing each node with the several closest frames assigned to it. Observed from the first hierarchy, the model puts a variety of vehicle-related frames into a single node. These frames are then refined into frames about cross-country vehicles and frames about vehicle-repairing. The frames on vehicle-repairing are further divided into bike repairing and car wheel repairing. These informative hierarchies learned by our model demonstrates its effectiveness of capturing meaningful hierarchical patterns in the data as well as exhibits interpretability.

7. Conclusions

We presented a new unsupervised learning framework to combine rich nCRP prior with VAEs. This embeds the data into a latent space with rich hierarchical structure, which has more abstract concepts higher up in the hierarchy, and less abstract concepts lower in the hierarchy. We developed a joint optimization framework for variational updates of both the neural and nCRP parameters. We showed an application of our model to video data, wherein, our experiments demonstrate that our model outperforms other models on two downstream tasks, namely, video classification and video retrieval. Qualitative analysis of our model by visualizing the learned hierarchy shows that our model captures rich interpretable structure in the data.

Acknowledgements. This work was partly funded by NSF IIS1563887, NSF IIS1447676, ONR N000141410684, and ONR N000141712463. Xiaodan Liang is supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

References

[1] D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012. 2

[2] 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 (JACM)*, 57(2):7, 2010. 1, 2, 3

[3] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015. 1

[4] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016. 2

[5] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015. 1

[6] G. E. Hinton and R. R. Salakhutdinov. Using deep belief nets to learn covariance kernels for Gaussian processes. In *Advances in neural information processing systems*, pages 1249–1256, 2008. 3

[7] Z. Hu, Q. Ho, A. Dubey, and E. P. Xing. Large-scale distributed dependent nonparametric trees. In *ICML*, pages 1651–1659, 2015. 2, 5

[8] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. Controlled generation of text. In *ICML*, 2017. 1

[9] Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing. On unifying deep generative models. *arXiv preprint arXiv:1706.00550*, 2017. 2

[10] M. Johnson, D. K. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, pages 2946–2954, 2016. 2

[11] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1, 2

[12] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015. 2

[13] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 2

[14] P. Over, J. Fiscus, G. Sanders, D. Joy, M. Michel, G. Awad, A. Smeaton, W. Kraaij, and G. Quénot. Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID*, page 52, 2014. 6

[15] C. E. Rasmussen. Gaussian processes for machine learning. 2006. 3

[16] N. Siddharth, B. Paige, A. Desmaison, J.-W. v. d. Meent, F. Wood, N. D. Goodman, P. Kohli, and P. H. Torr. Learning disentangled representations in deep generative models. 2017. 2

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 7

[18] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012. 7

[19] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer, 2016. 1

[20] C. Wang and D. M. Blei. Variational inference for the nested Chinese restaurant process. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2009. 3, 5

[21] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 370–378, 2016. 3