Missing Data Imputation and Acquisition with Deep Hierarchical Models and Hamiltonian Monte Carlo

Anonymous Author(s)
Affiliation
Address
email

Abstract

Variational Autoencoders (VAEs) have recently been highly successful at imputing and acquiring heterogeneous missing data. However, within this specific application domain, existing VAE methods are restricted by using only one layer of latent variables and strictly Gaussian posterior approximations. To address these limitations, we present HH-VAEM, a Hierarchical VAE model for mixed-type incomplete data that uses Hamiltonian Monte Carlo with automatic hyper-parameter tuning for improved approximate inference. Our experiments show that HH-VAEM outperforms existing baselines in the tasks of missing data imputation and supervised learning with missing features. Finally, we also present a sampling-based approach for efficiently computing the information gain when missing features are to be acquired with HH-VAEM. Our experiments show that this sampling-based approach is superior to alternatives based on Gaussian approximations.

1 Introduction

Many real-world unsupervised learning tasks require dealing with complicated datasets with mixed types (real, positive-valued, continuous, or discrete) and missing values. For this purpose, variational autoencoders [22, 23, 37] stand out in the recent literature as robust generative models that efficiently handle high-dimensional data. However, in their naive configuration, every data dimension is assumed to have similar statistical properties (i.e., homogeneity), and all dimensions are considered to be completely observed. Both assumptions won't hold in many real-world scenarios. Recent works have adapted VAEs to handle incomplete [10, 15, 29, 31] and mixed-type data [17, 28, 33], and demonstrated improved performance in downstream tasks such as missing data imputation and active information acquisition. Despite these advances, existing approaches are far from optimal as they are based on restrictive design choices: 1), only one layer of latent variables are considered; 2), Gaussian posterior approximations are usually adopted. These will lead to limited flexibility and additional bias, especially under real-world settings with complex mixed-type incomplete data.

In the literature, the issue of model flexibility and inference bias are often addressed separately. For example, approximate inference bias can be reduced by using Monte Carlo sampling [42, 45]. More specifically, Hamiltonian Monte Carlo (HMC) [6, 13] stands out among MCMC methods in machine learning due to its superior efficiency for exploring the target density. In the context of VAE, HMC has also been combined with stochastic variational inference [8] for improving the training of VAEs. On the other hand, the flexibility of VAEs can be improved by considering hierarchical VAEs with multiple layers of hidden variables [9, 30, 43, 46]. By using a hierarchical structure in the latent space, they enforce the information to flow from high-level representations to more specific observable factors, imitating the way information is often organized in the real world.

However, the issue of modeling flexibility and approximate inference bias are often heavily intertwined, and addressing them simultaneously in a *joint* manner is highly non-trivial. The hierarchical

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

organization of the latent variables creates complicated posterior dependencies that are not straightforward to deal with and require special consideration. To improve Gaussian approximate inference, most works opt by defining shared paths between the recognition and generative networks. While this makes hierarchical VAEs practical, the bias introduced by the Gaussian approximations is still present. To the best of our knowledge, none of the aforementioned hierarchical VAEs has been previously combined with Monte Carlo algorithms for improving over standard Gaussian approximate inference.

To overcome such limitations, we focus on training new hierarchical VAE models for heterogeneous mixed-type data with HMC. Our models can be used for missing data imputation and for supervised learning with missing data. We also present a sampling-based framework that allows our models to perform accurate sequential active information acquisition.

Our main contributions are as follows:

- We present HH-VAEM, a deep hierarchical model for handling mixed-type incomplete data that uses HMC with automatic hyper-parameter tuning for outperforming amortized variational inference by generating low bias samples from the true posterior.
- We propose a sampling-based strategy for missing feature acquisition that benefits from the improved inference of HH-VAEM. By using histograms to estimate the mutual information, this strategy achieves lower bias than other Gaussian-based alternatives.
- We exhaustively evaluate HH-VAEM in the tasks of 1) missing data imputation, 2) supervised learning with missing data and 3) information acquisition with our sampling-based strategy. In all cases we report significant gains with respect to baselines.

2 Related work

2.1 VAEs for mixed-type incomplete data

Variational Autoencoders [22, 37] are deep generative models that make use of encoder and decoder networks for mapping data into a latent Gaussian distribution, and reconstructing the latent codes into the original observational space, respectively. The parameters of these networks are trained using amortized Variational Inference [11, 50] optimizing a lower bound (ELBO) on the log evidence:

$$\mathcal{L}(\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{z}}(\boldsymbol{z}|\boldsymbol{x})} \log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\psi(\boldsymbol{z}|\boldsymbol{x})}, \tag{1}$$

where the generative model $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ can be expressed in terms of the likelihood $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ and the prior $p(\boldsymbol{z})$. In mixed-type data, the vector $\boldsymbol{x}$ is composed by data from different types: real, positive real, categorical, binary, etc. A naive approach is to consider a factorized decoder using different likelihood contributions $p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \prod_d p_\theta(x_d|\boldsymbol{z})$ [2, 33]. Nonetheless, the problem of handling unbalanced likelihoods leads to the domination of some dimensions during the optimization process. In [28], authors propose a solution using a set of *marginal* VAEs that encode each feature into a Gaussian uni-dimensional space, and a *dependency* VAE that captures the inter-dimensional dependencies more effectively using balanced Gaussian likelihoods.

By marginalizing each dimension of the decoder, incomplete data can be easily handled by dividing the vector $\boldsymbol{x}$ into the observed $\boldsymbol{x}_O$ and unobserved $\boldsymbol{x}_U$ parts. As proposed in [33] and [31], the ELBO objective is transformed into a lower bound on the observed data, and the unobserved data is replaced with zeros. All these methods rely on Gaussian approximations to the true posterior and thus operate by optimizing lower bounds on the log evidence.

2.2 Hierarchical VAEs

Hierarchical models have been successfully employed in deep generative modeling, [3, 40, 41]. In VAEs, defining a hierarchical latent space for VAEs can be straightforward. Nevertheless, potential pitfalls require special attention. Concretely, if the decoder is powerful enough, the model tends to uniquely use the shallowest layers, ignoring the deepest ones and falling into the well-known problem of *posterior collapse* [30, 35, 47]. In the last few years, several works have investigated possible hierarchical structures for VAEs. In [43], a bottom-up deterministic path is used along with a top-down inference path that shares parameters with the top-down structure of the generative model.

In [30], the authors use a bidirectional stochastic inference path. More recently, [46] or [9] have adapted these architectures to complex datasets and high quality images. Possibly motivated by the residual connections in [21], all these works use a shared path between recognition and generative models that helps in tying the divergences between approximations and priors in the ELBO.

2.3 Hamiltonian Monte Carlo

HMC [5, 13, 34] is a particularly effective MCMC algorithm for sampling from a target distribution $p(\boldsymbol{z}) = \frac{1}{\mathcal{Z}} p^*(\boldsymbol{z})$ where $\mathcal{Z}$ is the normalization constant and $\boldsymbol{z}$ is a d-dimensional vector. By augmenting this model to $p(\boldsymbol{r}, \boldsymbol{z}) = \mathcal{N}(\boldsymbol{r}; \boldsymbol{0}, \boldsymbol{M}) p(\boldsymbol{z})$, and denoting $\boldsymbol{r}$ as the *momentum* variable with diagonal covariance matrix $\boldsymbol{M}$, with the same dimensionality as $\boldsymbol{z}$, HMC samples from the distribution by simulating the time-evolution of a fictitious physical system.

The algorithm starts by firstly sampling $\boldsymbol{z}$ and $\boldsymbol{r}$ from an initial proposal and the momentum distribution, respectively. Chains with length T are built by recurrently proposing and accepting new states. To propose a new state, the Hamiltonian dynamics are simulated using a symplectic integrator, Leapfrog being the most common choice. The three following updates are repeated for L steps:

$$\begin{aligned} \boldsymbol{r}_{l+\frac{1}{2}} &= \boldsymbol{r}_l + \frac{1}{2} \boldsymbol{\phi} \odot \nabla_{z_l} \log p^*(\boldsymbol{z}_l) \,, \\ \boldsymbol{z}_{l+1} &= \boldsymbol{z}_k + \boldsymbol{r}_{l+\frac{1}{2}} \odot \boldsymbol{\phi} \odot \frac{1}{\boldsymbol{M}} \,, \\ \boldsymbol{r}_{l+1} &= \boldsymbol{r}_{l+\frac{1}{2}} + \frac{1}{2} \boldsymbol{\phi} \odot \nabla_{z_{l+1}} \log p^*(\boldsymbol{z}_{l+1}), \end{aligned} \tag{2}$$

where $\odot$ refers to the Hadamard product, and ϕ is the *step size* hyperparameter. Although it is typically defined as a scalar for simplicity, a d-dimensional vector can be considered to apply different step sizes per dimension, or further, as considered in this work, a $T \times d$ matrix to apply different steps per each proposal of the chain. The new proposal $(\boldsymbol{z}', \boldsymbol{r}')$ is accepted with probability $min\,[1, \exp(-H(\boldsymbol{z}', \boldsymbol{r}') + H(\boldsymbol{z}, \boldsymbol{r}))]$, where

$$H(\boldsymbol{z}, \boldsymbol{r}) = -\log p^*(\boldsymbol{z}) + \frac{1}{2} \boldsymbol{r}^T \boldsymbol{M}^{-1} \boldsymbol{r}. \tag{3}$$

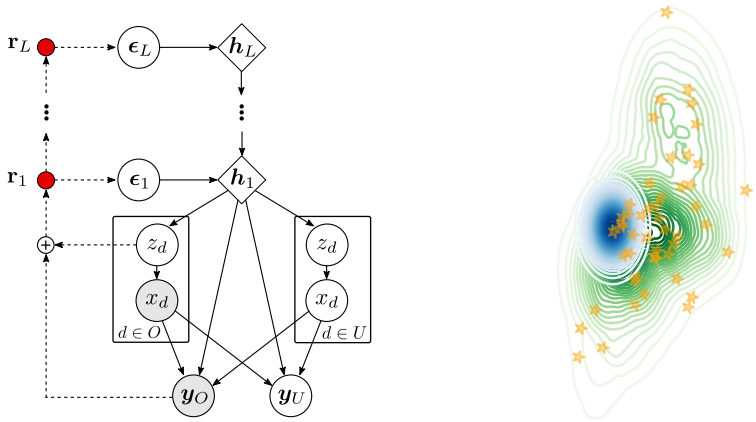
For the consecutive T proposals, a new momentum $\boldsymbol{r}$ is resampled and the updates of eq. (2) are repeated for L steps to update the state if $(\boldsymbol{z}', \boldsymbol{r}')$ is accepted.

2.4 Active Feature Acquisition

Among all the Active Learning techniques, Active Feature Acquisition [20, 32, 39, 44] is of special interest in cost-sensitive applications for modeling a trade-off between the improvement of predictions and the cost of acquiring new data at the feature level. Several works in the recent literature have studied methods for performing the task of sequentially acquiring high-value information by selecting features that maximize an information theoretical reward function and enhance the accuracy of the predictions. This task is denoted by SAIA (Sequential Active Information Acquisition). In [29], it is proposed an efficient method for approximating a non-tractable reward by using the encoder of a VAE that handles missing data. In [28] they extend this method for handling mixed-type data. Both works computing the reward by relying on Gaussian approximations given by the encoder networks.

3 Hamiltonian hierarchical VAE for mixed-type incomplete data (HH-VAEM)

The HH-VAEM model (Figure 1 (a)) is a Hierarchical VAE for mixed-type, incomplete data that incorporates HMC with automatic hyper-parameter optimization for sampling from the posterior of the latent variables. In a first stage, the mixed-type data is encoded into marginal Gaussian posterior distributions as given by univariate VAEs fitted to each data dimension. In a second stage, a hierarchical structure captures the dependencies among the standarized, homogeneous dimensions with well-balanced Gaussian likelihoods. The model is trained using samples from the posterior of the hierarchical latent variables by means of HMC, whilst the HMC hyper-parameters are automatically tuned. A more detailed description is provided in the following subsections.



(a) The HH-VAEM model

(b) HMC sampling

Figure 1: The HH-VAEM model (a). Illustrative example (b): samples $\boldsymbol{\epsilon}^{(T)}$ obtained with HMC (orange) following the true posterior $p(\boldsymbol{\epsilon}|\boldsymbol{x}_O, \boldsymbol{y}_O)$ (green) using the Gaussian distribution given by the encoder $q^{(0)}(\boldsymbol{\epsilon}|\boldsymbol{x}_O, \boldsymbol{y}_O)$ (blue) as the initial proposal, with latent dimension $M = 2$.

3.1 Notation

The model generates both data $\boldsymbol{x} \in \mathbb{R}^D$ and output $\boldsymbol{y} \in \mathbb{R}^P$, where each of these variables is divided into observed parts $\boldsymbol{x}_O$, $\boldsymbol{y}_O$ and unobserved parts $\boldsymbol{x}_U$, $\boldsymbol{y}_U$. Each dimension of the input data is denoted by x_d. A training dataset is composed of N observations as tuples $(\boldsymbol{x}_O^{(n)}, \boldsymbol{y}_O^{(n)})$. For ease of notation, we omit the observation index n, and the objectives are presented for a single observation. The dependency latent space is composed of L latent variables $[\boldsymbol{\epsilon}_1, ..., \boldsymbol{\epsilon}_L]$, with $\boldsymbol{\epsilon}_l \in \mathbb{R}^{m_l}$. The total dimension of the joint latent distribution is $\sum_l m_l = M$. In the marginal VAEs, the latent variables z_d are unidimensional.

3.2 Handling heterogeneous incomplete data

Following the strategy proposed by [28], we perform a two-staged approach for handling heterogeneous data. The marginal distribution of each feature $p_{\theta_d}(x_d)$ is modeled by a one-dimensional VAE. First, the D marginal VAEs are trained independently by maximizing the marginal ELBO over observed points

$$\mathcal{L}_d(x_d; \{\theta_d, \gamma_d\}) = \mathbb{1}(x_d \in \boldsymbol{x}_O)\mathbb{E}_{q_{\gamma_d}(z_d|x_d)} \log \frac{p_{\theta_d}(x_d, z_d)}{q_{\gamma_d}(z_d|x_d)}, \tag{4}$$

where $\mathbb{1}(x_d \in \boldsymbol{x}_O)$ is an indicator function that activates the ELBO when the feature is observed. Second, a dependency VAE encodes a vector $\boldsymbol{z}$ with the concatenated samples from the marginal posteriors $q_{\gamma_d}(z_d|x_d)$ into the global latent variable $\boldsymbol{h}$, using zero-filling for the unobserved variables. By using this approach, $\boldsymbol{z}$ is now homogeneous and can then be easily modeled using a standard Gaussian decoder. By contrast, other works [14, 33] directly operate with different decoding likelihoods per data type. These often have very different magnitudes in the ELBO and may reduce learning efficiency. The ELBO for the second stage dependency VAE is

$$\mathcal{L}(\boldsymbol{x}_O, \boldsymbol{y}_O; \{\theta_z, \psi\}) = \mathbb{E}_{q_\psi(\boldsymbol{h}|\boldsymbol{z}_O, \boldsymbol{y}_O)} \log \frac{p_{\theta_z}(\boldsymbol{z}_O, \boldsymbol{h})}{q_\psi(\boldsymbol{h}|\boldsymbol{z}_O, \boldsymbol{y}_O)}. \tag{5}$$

3.3 Predictive enhancement

With the aim at reinforcing the prediction of the variable of interest, we turn into a supervised model by including a separate predictor for $p_{\theta_y}(\boldsymbol{y}|\hat{\boldsymbol{x}}, \boldsymbol{h})$, apart from the decoder $p_{\theta_z}(\boldsymbol{z}|\boldsymbol{h})$. The notation $\hat{\boldsymbol{x}}$ refers to the whole observational vector, including the imputation of the missing part by decoding the latent $\boldsymbol{h}$ into $\boldsymbol{z}$ and each marginal z_d into $\hat{\boldsymbol{x}}_u$. The predictor parameters θ_y are optimized along with the decoder parameters θ_z.

3.4 Hierarchical reparameterized latent space

A hierarchical structure over the latent space $\boldsymbol{h} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_L\}$ enriches the prior assumptions and permits a flexible generation of data in a more natural fashion. Nevertheless, Hierarchical VAEs require special care with their optimization. Most of the recent works employ a deterministic top-down path, shared between the inference and generative models, that helps in tying the prior and the posterior approximate distributions during the optimization. On the contrary, in our model, the latent variables $\boldsymbol{\epsilon}_l$ are disconnected from each other and the hierarchical structure present in $\boldsymbol{h}_l$ is modeled by deterministic networks, thereby this shared path is no longer needed. Thus, the representation at each layer is a reparametrization from variable $\boldsymbol{\epsilon}_l$ with standard Gaussian prior $p(\boldsymbol{\epsilon}_l)$

$$\boldsymbol{h}_l = f_{\mu_l}(\boldsymbol{h}_{l+1}) + f_{\sigma_l}(\boldsymbol{h}_{l+1}) \cdot \boldsymbol{\epsilon}_l, \tag{6}$$

where the functions $f_{\mu_l}(\boldsymbol{h}_{l+1})$ and $f_{\sigma_l}(\boldsymbol{h}_{l+1})$ are applied by NNs with parameters $\theta_l = \{\theta_{\mu_l}, \theta_{\sigma_l}\}$. The result is equivalent as learning the mean and covariance of autoregressive variables (see Figure 4 for illustrative details). However, by choosing this reparameterization, we relax the dependencies among the latent variables, resulting in a smoother joint posterior density $p(\boldsymbol{\epsilon}|\boldsymbol{x}_O, \boldsymbol{y}_O)$. Performing the inference over $\boldsymbol{\epsilon} = \{\boldsymbol{\epsilon}_1, ..., \boldsymbol{\epsilon}_L\}$ leads to a better posed basis for running HMC, as detailed in section 3.5. To give an example, in [46], the AR path $p(z_l|z_{<l})$ would provoke instabilities inside the HMC integrator due to huge gradients in $\nabla_{z_{1:L}} \log p^*(z_{1:L})$ (Betancourt et at., 2015). We solve this by disconnecting the latent variables ($p(\boldsymbol{\epsilon}_l)$ are standard Gaussians). For the sake of simplicity, we name the generative parameters as $\theta = \{\theta_z, \theta_y, \theta_1, ..., \theta_L\}$.

In our work, a deterministic bottom-up path is built from the observational space to each latent layer for encoding the information. Naming $\boldsymbol{r}_l$ the hidden representation at each layer, and defining $\boldsymbol{r}_0 = \{\boldsymbol{x}_O, \boldsymbol{y}_O\}$, we employ NNs with parameters ψ_{r_l} for computing $\boldsymbol{r}_l = f_r(\boldsymbol{r}_{l-1})$. These vectors are mapped into the parameters of the variational posterior $q_{\psi_l}(\boldsymbol{\epsilon_l}|\boldsymbol{x}_O, \boldsymbol{y}_O)$, using NNs for computing the mean as $g_{\mu_l}(\boldsymbol{r}_l)$ and the covariance as $g_{\sigma_l}(\boldsymbol{r}_l)$, with parameters ψ_{μ_l} and ψ_{σ_l}. With compactness purposes, we will denote the encoder parameters as $\psi = \{\psi_1, ..., \psi_L\}$, with $\psi_l = \{\psi_{r_l}, \psi_{\mu_l}, \psi_{\sigma_l}\}$. The dependency ELBO under this hierarchical reparameterized model becomes

$$\begin{gathered}\mathcal{L}_{VI}(\boldsymbol{x}_O, \boldsymbol{y}_O; \{\theta, \psi\}) = \mathbb{E}_{q_\psi}\left[\log \frac{p_\theta(\boldsymbol{z}_O, \boldsymbol{\epsilon})}{q_\psi(\boldsymbol{\epsilon}|\boldsymbol{z}_O, \boldsymbol{x}_O, \boldsymbol{y}_O)}\right] = \\ \mathbb{E}_{q_\psi}\left[\log p_\theta(\boldsymbol{z}_O|\boldsymbol{h}_1) + \log p_\theta(\boldsymbol{y}_O|\hat{\boldsymbol{x}}, \boldsymbol{h}_1)\right] - \sum_{l=1}^{L} D_{\mathrm{KL}}\left(q_\psi(\boldsymbol{\epsilon}_l|\boldsymbol{x}_O, \boldsymbol{y}_O)||p(\boldsymbol{\epsilon}_l)\right).\end{gathered} \tag{7}$$

3.5 HMC over the hierarchical density

In recent works, HMC has been combined with deep generative models for improving the inference of the latent variables by obtaining better samples from the posterior [8, 19]. In this work, we propose to transcend these previous approaches and build a generalized method for sampling from complicated, hierarchical latent structures composed by several layers. Inspired by [7] and their method for sampling from the posterior within a vanilla VAE framework while tuning the HMC hyperparameters, we follow a procedure for training the dependency model where i) during a pre-training stage, the encoder and decoder are optimized using standard VI and the ELBO from equation (7), and ii) using the pre-trained encoder for starting from a good proposal [19], HMC samples are obtained to follow the true posterior and jointly optimize the generative model and the HMC hyperparameters. In Figure 1 (b) we include an illustrative example.

We denote by $q_\phi^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{z}_O, \boldsymbol{x}_O, \boldsymbol{y}_O)$ the implicit distribution for the posterior after T HMC steps. The hyper-parameters of HMC are named ϕ, a $T \times d$ matrix containing the step sizes for each dimension at each step of the Leapfrog integration. As exposed by [7], optimizing the momentum variances does no improve performance. Within this perspective, the hyper-parameters can be optimized using variational inference by maximizing the ELBO

$$\mathbb{E}_{q_\phi^{(T)}(\boldsymbol{\epsilon})}\left[\log p(\boldsymbol{z}_O, \boldsymbol{y}_O, \boldsymbol{\epsilon})\right] + H\left[q_\phi^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{x}_O, \boldsymbol{y}_O)\right], \tag{8}$$

where the first part is the HMC objective, and can be easily estimated via Monte Carlo

$$\mathcal{L}_{HMC}(\boldsymbol{z}_O, \boldsymbol{y}_O; \{\theta, \psi, \phi\}) = \mathbb{E}_{q_\phi^{(T)}(\boldsymbol{\epsilon})}\left[\log p_\theta(\boldsymbol{z}_O|\boldsymbol{h}_1) + \log p_\theta(\boldsymbol{y}_O|\hat{\boldsymbol{x}}, \boldsymbol{h}_1) + \sum_{l=1}^{L} p(\boldsymbol{\epsilon}_l^{(T)})\right]. \tag{9}$$

Nevertheless, the entropy term $H\left[q_\phi^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{x}_O, \boldsymbol{y}_O)\right]$ in Equation (8) is not tractable since we are not able to explicitly evaluate the distribution $q_\phi^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{x}_O, \boldsymbol{y}_O)$. Although optimizing the first term might result in a well-posed algorithm, this would bring consequences that must be considered. Namely, without a proper regularization term, and in case the initial proposal $q_\phi^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{x}_O, \boldsymbol{y}_O)$ is concentrated in high density regions, the chains would barely move from the initial state and only these regions with high density would be explored. To cope with this problem, we define an inflation parameter $\boldsymbol{s}$ to increase the variance of the proposal $q_\phi(\boldsymbol{\epsilon}|\boldsymbol{z}_O, \boldsymbol{x}_O, \boldsymbol{y}_O)$ given by the Gaussian encoder. Whilst in [7] the authors define this parameter as a scalar factor applied to all the latent dimensions, in our work, we extend this to apply a different inflation factor at each latent level $\boldsymbol{s} = \{s_1, ..., s_L\}$. In order to tune these inflation parameters, we minimize the SKSD discrepancy measure [16]

$$\mathcal{L}_{SKSD}(\boldsymbol{x}_O, \boldsymbol{y}_O; \boldsymbol{s}) = \text{SKSD}\left(q_\phi^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{z}_O, \boldsymbol{x}_O, \boldsymbol{y}_O; \boldsymbol{s}),\, p(\boldsymbol{\epsilon}|\boldsymbol{z}_O, \boldsymbol{x}_O, \boldsymbol{y}_O)\right), \tag{10}$$

which fits perfectly our requirements, since only requires samples from HMC and gradients $\nabla_\epsilon \log p(\boldsymbol{z}_O, \boldsymbol{y}_O, \boldsymbol{\epsilon})$ for measuring a discrepancy between the implicit and the true posterior. Further, the SKSD performs better than other discrepancies like [27] in high dimensional latent spaces.

Despite of the robustness and success of HMC methods, it remains a well-known issue that hierarchical objectives lead to computational issues in the integrator [5]. The autoregressive dependencies among variables create regions of high curvature in the typical set of the joint density that can be pathological for symplectic integrators. These type of issues can be solved by manipulating the HMC algorithm by means of a parameterised covariance for the momentum distribution [6]. Nonetheless, by using our reparameterized version, we are able to relax these dependencies and maintain the standard HMC method for obtaining samples that approximately follow the true posterior.

Algorithm 1 Training algorithm for HH-VAEM

Input: data $\left(\boldsymbol{x}_O^{(1:N)}, \boldsymbol{y}_O^{(1:N)}\right)$, steps: T_d, T_{VI}, T_{HMC}
Parameters: γ, θ, ψ, ϕ, s
STAGE 1: MARGINAL VAEs
for $d = 1$ **to** D **do**
  Initialize marginal VAE $\{\theta_d, \gamma_d\}$
  for $t = 1$ **to** T_d **do**
    $\gamma_d^{t+1}, \theta_d^{t+1} \leftarrow \text{Adam}_{\gamma_d^t, \theta_d^t}(\mathcal{L}_d)$
  end for
end for
STAGE 2: DEPENDENCY VAE
for $t = 1$ **to** T_{VAE} **do**
  $\theta^{t+1}, \psi^{t+1} \leftarrow \text{Adam}_{\theta^t, \psi^t}(\mathcal{L}_{VI})$
end for
STAGE 3: JOINTLY OPTIMIZING VAE + HMC
for $t = 1$ **to** T_{HMC} **do**
  $\psi^{t+1} \leftarrow \text{Adam}_{\psi^t}(\mathcal{L}_{VI})$
  $\theta^{t+1}, \phi^{t+1} \leftarrow \text{Adam}_{\theta^t, \phi^t}(\mathcal{L}_{HMC})$
  $s^{t+1} \leftarrow \text{Adam}_{s^t}(\mathcal{L}_{SKSD})$
end for

3.6 HH-VAEM optimization

The optimization of the HH-VAEM is divided into three stages. In a first stage, we train one independent marginal VAE per dimension. In a second stage, the dependency VAE is trained, using as inputs the concatenation of the encoded dimensions $\boldsymbol{z} = \{z_1, ..., z_D\}$ and the target $\boldsymbol{y}$. Finally, in a third stage, the HMC hyperparameters, the decoder and the predictor are tuned using the HMC objective, the inflation parameter is trained using the SKSD discrepancy, and the encoder is trained used the ELBO. The pseudocode for HH-VAEM training is shown in Algorithm 1.

3.7 Computational cost

In a VAE, the data is fed to the encoder with aim at obtaining the variational parameters for sampling from the Gaussian approximated posterior. In our method, the data is similarly encoded to obtain the initial Gaussian proposal $q^{(0)}$, and the samples from this distribution are updated for T cycles to obtain the implicit $q^{(T)}$ using HMC. Within each of these iterations, L leapfrog steps (2) are executed. For each of these steps, the computation of the gradients $\nabla_{\boldsymbol{\epsilon}_l} \log p(\boldsymbol{z}, \boldsymbol{y}, \boldsymbol{\epsilon}_l)$ and $\nabla_{\boldsymbol{\epsilon}_{l+1}} \log p(\boldsymbol{z}, \boldsymbol{y}, \boldsymbol{\epsilon}_{l+1})$ is required. To obtain these gradients, we need to *i)* compute the parameters of the likelihood $p(\boldsymbol{z}, \boldsymbol{y}|\boldsymbol{\epsilon})$ that are given by the decoder ($p(\boldsymbol{z}|\boldsymbol{h}_1)$) and predictor ($p(\boldsymbol{y}|\hat{\boldsymbol{x}}, \boldsymbol{h}_1)$), *ii)* evaluate it and *iii)* perform the automatic differentiation. Thus, for running our method, an additional cost from both decoding and performing differentiation a total of $2TL$ times is introduced.

By jointly optimizing the HMC hyperparameters we are able to achieve faster convergence of the chains with smaller lengths (T), as demonstrated in [7]. Further, to reduce the computational cost as much as possible, we optimize the HMC hyperparameters only for a few steps in a final training stage, since convergence is rapidly achieved (as observed empirically in Fig. 6). At test time, samples from the Gaussian proposal $q^{(0)}$ (faster and cheaper), or from HMC $q^{(T)}$ (slower and better) can be used to fit computational constraints.

4 Sampling-based Active Learning

Considering that the input data are tuples of observed and missing features $\{\boldsymbol{x}_O, \boldsymbol{x}_U\}$, our Active Learning framework follows [28, 29] and determines which feature $\boldsymbol{x}_i \in \boldsymbol{x}_U$ will enhance the prediction of the target $\boldsymbol{y}$ the most for a particular $\boldsymbol{x}_O$. Concretely, in a Sequential Active Information Acquisition (SAIA) scenario, this decision is taken sequentially to optimally increase knowledge and accurately predict $\boldsymbol{y}$. From a information theoretical perspective, this task can be performed recurrently by maximizing a reward function R at each step d. This reward might represent abstract quantities of interest like the cost or benefit of acquiring $\boldsymbol{x}_i$ (depending on the sign). In Bayesian experimental analysis, R is the expected gain of information [26]. Following [4], we can define it as

$$R(i, \boldsymbol{x}_O) = \mathbb{E}_{p(x_i|\boldsymbol{x}_O)} \, D_{\mathrm{KL}} \left(p(\boldsymbol{y}|x_i, \boldsymbol{x}_O) p(\boldsymbol{y}, |\boldsymbol{x}_O) \right), \tag{11}$$

where i is the index of each unobserved feature. Intuitively, this reward can be interpreted as the expected change in the predictive distribution when $\boldsymbol{x}_i$ is observed. As one might appreciate, the reward needs to be estimated via Monte Carlo by sampling from $p(x_i|\boldsymbol{x}_O)$. With a robust generative model that handles missing data like HH-VAEM, these samples are easily obtained: first, using HMC, we sample $\boldsymbol{\epsilon}^{(T)}$ from $p(\boldsymbol{\epsilon}|\boldsymbol{x}_O)$. Second, we decode these samples to obtain x_i from $p(\boldsymbol{z}|\boldsymbol{h}_1)$ and $p(x_i|z_d)$. Nonetheless, the reward defined in (11) is intractable since both $p(\boldsymbol{y}|x_i)$ and $p(\boldsymbol{y}|x_i, \boldsymbol{x}_O)$ are intractable: computing them requires to integrate out the latent variables. This motivates the authors of [28, 29] to present a transformation of the reward for being computed in the latent space using the encoder network. Although they prove that this transformation effectively provides a good estimation in several datasets, we demonstrate that for low dimensional targets (commonly with one or two dimensions), the estimation of the reward with histograms is more effective.

The reward in (11) can be rewritten as

$$D_{\mathrm{KL}} \left[p(\boldsymbol{y}, x_i|\boldsymbol{x}_O) || p(\boldsymbol{y}|\boldsymbol{x}_O) p(x_i|\boldsymbol{x}_O) \right] = \mathcal{I}(\boldsymbol{y}; x_i \, |\boldsymbol{x}_O), \tag{12}$$

which is exactly the Mutual Information between the target and the missign feature x_i, conditioned on observing $\boldsymbol{x}_O$. While a set of advanced non-parametric estimators of the mutual information are available [24, 38], many are not easilly adapted for parallelization. We demonstrate that the simplest one, based on binning the x_i and $\boldsymbol{y}$ domains

$$\hat{I}(\boldsymbol{y}; \boldsymbol{x}_i \, |\boldsymbol{x}_O) \approx \sum_{ij} p(i,j) \log \frac{p(i,j)}{p_x(i) p_y(j)}, \tag{13}$$

is effective and easy to parallelize. In the equation, $p_x(i) = n_x(i)/N$, $p_y(j) = n_y(j)/N$ and $p(i,j) = n(i,j)/N$ are the relative frequencies that approximate $p(x)$, $p(y)$ and $p(x,y)$ for each bin. Thus, $n_x(i)$, $n_y(j)$ and $n(i,j)$ are the number of samples inside each interval. This biased estimator is sufficient for discriminating among important features that will enhance predictions.

5 Experiments

The evaluation of the HH-VAEM model is organized into four experiments. The comparison is performed with the following baseline models:

- *VAEM*: The VAEM strategy [28] (without including the Partial VAE).
- *MIWAEM*: VAEM combined with the importance weighted estimation proposed in [31].
- *H-VAEM*: A Hierarchical VAEM with two layers of latent variables and a Gaussian encoder.
- *HMC-VAEM*: A VAEM that includes a tuned HMC sampler for the true posterior.

	Bank	Insurance	Avocado	Naval	Yatch	Diabetes	Concrete	Wine	Energy	Boston
VAEM	2.84 ± 0.07	1.81 ± 0.03	1.89 ± 0.01	0.55 ± 0.05	3.15 ± 0.28	2.78 ± 0.16	2.45 ± 0.26	3.01 ± 0.61	2.09 ± 0.10	2.01 ± 0.23
MIWAEM	2.74 ± 0.05	1.88 ± 0.04	1.92 ± 0.04	0.57 ± 0.03	2.66 ± 0.11	2.55 ± 0.09	2.34 ± 0.51	2.76 ± 0.48	2.06 ± 0.14	1.94 ± 0.23
H-VAEM	2.82 ± 0.06	1.80 ± 0.04	1.89 ± 0.01	0.48 ± 0.06	3.06 ± 0.31	2.74 ± 0.09	2.42 ± 0.21	2.85 ± 0.56	1.72 ± 0.11	1.89 ± 0.24
HMC-VAEM	2.69 ± 0.05	1.77 ± 0.06	1.89 ± 0.02	0.49 ± 0.07	$\mathbf{2.21 \pm 0.24}$	2.72 ± 0.20	2.28 ± 0.29	2.83 ± 0.46	1.73 ± 0.05	1.83 ± 0.16
HH-VAEM	$\mathbf{2.63 \pm 0.04}$	$\mathbf{1.75 \pm 0.03}$	$\mathbf{1.88 \pm 0.05}$	$\mathbf{0.40 \pm 0.05}$	2.47 ± 0.27	$\mathbf{2.54 \pm 0.13}$	$\mathbf{2.28 \pm 0.09}$	$\mathbf{1.90 \pm 0.17}$	$\mathbf{1.71 \pm 0.04}$	$\mathbf{1.83 \pm 0.11}$

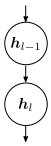
Table 1: Test negative log likelihood of the unobserved features for our model and baselines.

	Bank	Insurance	Avocado	Naval	Yatch	Diabetes	Concrete	Wine	Energy	Boston
VAEM	0.56 ± 0.06	1.20 ± 0.03	1.18 ± 0.02	2.69 ± 0.01	0.61 ± 0.02	1.59 ± 0.19	1.07 ± 0.09	0.28 ± 0.09	0.61 ± 0.14	0.85 ± 0.21
MIWAEM	0.51 ± 0.03	1.15 ± 0.03	1.15 ± 0.03	2.70 ± 0.01	0.60 ± 0.03	$\mathbf{1.36 \pm 0.10}$	0.95 ± 0.22	0.28 ± 0.13	0.54 ± 0.12	0.80 ± 0.21
H-VAEM	0.50 ± 0.03	1.06 ± 0.02	1.18 ± 0.02	2.68 ± 0.01	0.60 ± 0.02	1.71 ± 0.14	1.02 ± 0.09	0.26 ± 0.11	0.46 ± 0.14	0.90 ± 0.22
HMC-VAEM	0.52 ± 0.02	1.00 ± 0.03	1.12 ± 0.03	2.71 ± 0.01	$\mathbf{0.52 \pm 0.15}$	1.55 ± 0.29	0.95 ± 0.26	0.28 ± 0.09	0.41 ± 0.07	0.71 ± 0.13
HH-VAEM	$\mathbf{0.49 \pm 0.03}$	$\mathbf{0.93 \pm 0.06}$	$\mathbf{1.10 \pm 0.01}$	$\mathbf{2.62 \pm 0.01}$	0.56 ± 0.02	1.38 ± 0.18	$\mathbf{0.95 \pm 0.08}$	$\mathbf{0.20 \pm 0.04}$	$\mathbf{0.32 \pm 0.05}$	$\mathbf{0.55 \pm 0.04}$

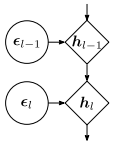
Table 2: Test negative log likelihood of the predicted target for our model and baselines.

For all the models, we manually introduce missing data in the training set by randomly setting per data point a feature as missing with a probability sampled uniformly in the interval $[0.01, 0.99]$ within each batch. Both the input data $\boldsymbol{x}$ and the target $\boldsymbol{y}$ can be missing. For the test set, a fixed probability of 0.5 leads to about half of the input data being observed, whilst the target is completely unobserved.

A total of 10 UCI datasets [12] that include mixed-type data are employed for the evaluation, Further, we include both MNIST [25] and Fashion-MNIST [49] datasets for evaluating our model in higher dimensional observational and latent spaces and bigger architectures (3 layered convolutional nets for encoder/decoder). For these two special cases, the marginal VAEs are not included and the dependency VAE is fed directly with the Bernoulli-distributed pixels. We name this model HH-VAE, and similarly, the baselines are renamed as VAE, MIWAE, HMC-VAE and H-VAE. Extended details on the employed architectures and evaluation times are provided in Appendix A.1.



(a) AR hierarchy



(b) Reparameterization

Figure 4: Autoregressive hierarchy (a) vs reparameterized version (b).

	VAE	MIWAE	H-VAE	HMC-VAE	**HH-VAE**
MNIST	0.124 ± 0.001	0.121 ± 0.001	0.119 ± 0.001	0.101 ± 0.004	$\mathbf{0.094 \pm 0.003}$
F-MNIST	0.162 ± 0.002	0.160 ± 0.002	0.156 ± 0.002	0.150 ± 0.002	$\mathbf{0.144 \pm 0.002}$

Table 3: Test negative log likelihood of the unobserved features for the MNIST datasets.

	VAE	MIWAE	H-VAE	HMC-VAE	**HH-VAE**
MNIST	0.153 ± 0.009	0.151 ± 0.007	0.146 ± 0.006	0.067 ± 0.007	$\mathbf{0.056 \pm 0.019}$
F-MNIST	0.501 ± 0.012	0.496 ± 0.008	0.494 ± 0.007	0.357 ± 0.060	$\mathbf{0.337 \pm 0.069}$

Table 4: Test negative log likelihood of the predicted target for the MNIST datasets.

	VAE	MIWAE	H-VAE	HMC-VAE	**HH-VAE**
MNIST	0.953 ± 0.004	0.953 ± 0.003	0.953 ± 0.003	0.978 ± 0.003	$\mathbf{0.981 \pm 0.005}$
F-MNIST	0.824 ± 0.005	0.824 ± 0.004	0.824 ± 0.004	0.869 ± 0.015	$\mathbf{0.876 \pm 0.017}$

Table 5: Test accuracy of the predicted digits for the MNIST datasets.

5.1 Mixed type conditional data imputation

In order to evaluate the performance of the model in terms of data imputation, we opt by computing the negative log likelihood of the unobserved features. We make use of the Monte Carlo approximation

$$\log p(\boldsymbol{x}_U|\boldsymbol{x}_O) = \log \mathbb{E}_{\boldsymbol{\epsilon}\sim q^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{x}_O)} \left[p(\boldsymbol{x}_U|\boldsymbol{\epsilon})\right] \approx \log \frac{1}{k} \sum_{i}^{k} p(\boldsymbol{x}_U|\boldsymbol{\epsilon}_i), \tag{14}$$

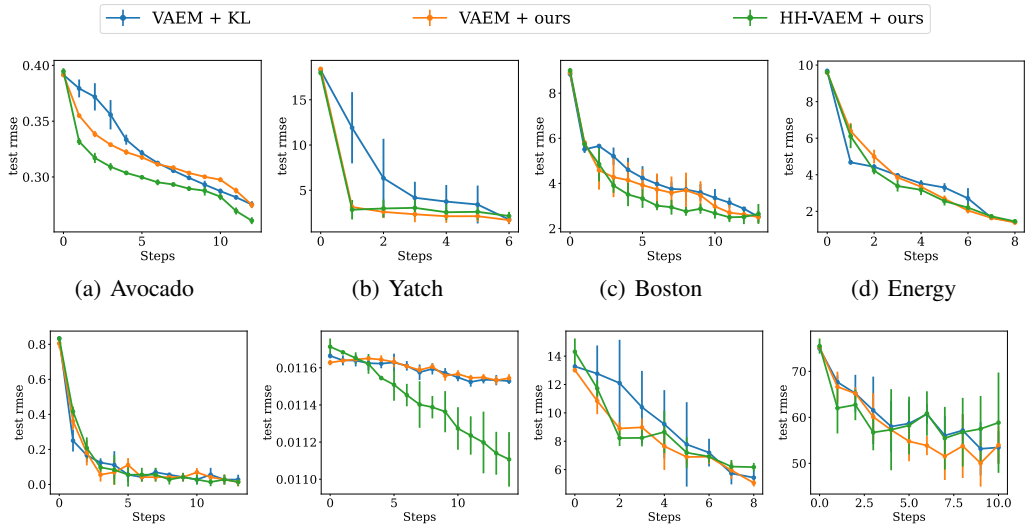
which is averaged over features in order to compare the imputation performance with the baselines. Results on the 10 UCI datasets and the MNIST datasets are included in Tables 1 and 3, showing that both HH-VAE and HH-VAEM outperform the baselines in almost all the cases.

5.2 Target prediction

For this experiment, we compute the negative log likelihood of the target under the predictive distribution using the same procedure as in the previous section. Results included in Tables 2 and 4 show that our model again outperforms the baselines in almost all the cases.

5.3 Sequential active information acquisition (SAIA)

In this experiment, the HH-VAEM model and our proposed acquisition function are evaluated in a SAIA task. We start by predicting from completely unobserved inputs. At each step, the missing feature that maximizes the reward is acquired sequentially. Figure 5 shows the resulting error curves for the UCI datasets. Blue lines correspond to the transformed reward proposed by [28]. Orange lines are our reward in a VAEM framework (not including the hierarchy and the HMC sampler). Green lines correspond to HH-VAEM with our reward. In most of the cases, our model and acquisition technique outperform the baselines, obtaining lower errors and a faster discovery of information.



(a) Avocado (b) Yatch (c) Boston (d) Energy

(e) Wine (f) Naval (g) Concrete (h) Diabetes

Figure 5: SAIA curves. Horizontal axis shows number of discovered features. Vertical axis is RMSE.

5.4 Training times

Table 6 shows the average training time in minutes for each model in the experiments for Tables 1 and 2. The ratio between training times for our method and the Gaussian baselines is approximately between 5 and 10.

	Bank	Avocado	Yatch	Diabetes	Concrete	Wine	Energy	Boston
VAEM	29.92 ± 0.39	21.49 ± 1.64	5.89 ± 0.01	8.79 ± 0.30	7.70 ± 0.58	7.92 ± 0.11	8.18 ± 0.09	10.01 ± 0.34
MIWAEM	63.33 ± 6.20	37.17 ± 2.28	8.21 ± 0.24	13.29 ± 0.33	11.51 ± 0.52	11.81 ± 0.13	16.71 ± 0.16	15.01 ± 0.13
H-VAEM	41.44 ± 0.38	33.83 ± 0.81	15.84 ± 0.11	13.38 ± 0.47	11.80 ± 0.38	10.61 ± 0.54	12.71 ± 1.20	13.74 ± 1.43
HMC-VAEM	281.88 ± 9.14	356.22 ± 5.94	23.50 ± 1.60	50.42 ± 1.18	42.70 ± 3.14	63.46 ± 1.97	90.87 ± 7.05	103.72 ± 7.24
HH-VAEM	316.81 ± 9.49	388.28 ± 6.47	27.08 ± 0.12	68.29 ± 4.06	65.78 ± 0.43	79.97 ± 5.53	140.33 ± 7.36	129.30 ± 4.69

Table 6: Training times (in minutes) of our model and baselines.

6 Conclusion

We presented HH-VAEM, to our knowledge, the first hierarchical VAE for mixed-type incomplete data that uses HMC with automatic hyper-parameter tuning for improved inference. We provide experiments that demonstrate its superiority with respect the baselines in the tasks of missing data imputation and supervised learning, placing HH-VAEM as a robust model for real-world datasets. Further, we have developed a novel sampling-based technique for dynamic feature selection that outperforms the Gaussian-based alternatives and results in an efficient method for active learning in deep generative models.

References

[1] J. An and S. Cho. Variational Autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[2] D. Barrejón, P. M. Olmos, and A. Artés-Rodríguez. Medical data wrangling with sequential Variational Autoencoders. *arXiv preprint arXiv:2103.07206*, 2021.

[3] Y. Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[4] J. M. Bernardo. Expected information as expected utility. *the Annals of Statistics*, pages 686–690, 1979.

[5] M. Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[6] M. Betancourt and M. Girolami. Hamiltonian Monte Carlo for hierarchical models. *Current trends in Bayesian methodology with applications*, 79(30):2–4, 2015.

[7] A. Campbell, W. Chen, V. Stimper, J. M. Hernandez-Lobato, and Y. Zhang. A gradient Based Strategy for Hamiltonian Monte Carlo Hyperparameter optimization. In *International Conference on Machine Learning*, pages 1238–1248. PMLR, 2021.

[8] A. L. Caterini, A. Doucet, and D. Sejdinovic. Hamiltonian Variational Auto-Encoder. *arXiv preprint arXiv:1805.11328*, 2018.

[9] R. Child. Very deep VAEs generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020.

[10] M. Collier, A. Nazabal, and C. K. Williams. VAEs in the presence of missing data. *arXiv preprint arXiv:2006.05301*, 2020.

[11] C. Cremer, X. Li, and D. Duvenaud. Inference Suboptimality in Variational Autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR, 2018.

[12] D. Dua, C. Graff, et al. UCI machine learning repository. 2017.

[13] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

[14] S. Eduardo, A. Nazábal, C. K. Williams, and C. Sutton. Robust Variational Autoencoders for outlier detection and repair of mixed-type data. In *International Conference on Artificial Intelligence and Statistics*, pages 4056–4066. PMLR, 2020.

[15] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. A. Eslami. Conditional Neural Processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018.

[16] W. Gong, Y. Li, and J. M. Hernández-Lobato. Sliced Kernelized Stein Discrepancy. *arXiv preprint arXiv:2006.16531*, 2020.

[17] Y. Gong, H. Hajimirsadeghi, J. He, T. Durand, and G. Mori. Variational Selective Autoencoder: Learning from partially-observed heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 2377–2385. PMLR, 2021.

[18] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[19] M. D. Hoffman. Learning deep latent Gaussian models with Markov Chain Monte Carlo. In *International conference on machine learning*, pages 1510–1519. PMLR, 2017.

[20] S.-J. Huang, M. Xu, M.-K. Xie, M. Sugiyama, G. Niu, and S. Chen. Active feature acquisition with supervised matrix completion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1571–1579, 2018.

[21] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved Variational Inference with Inverse Autoregressive Flow. *Advances in neural information processing systems*, 29:4743–4751, 2016.

[22] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[23] D. P. Kingma and M. Welling. An introduction to Variational Autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

[24] A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

[25] Y. LeCun. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[26] D. V. Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, pages 986–1005, 1956.

[27] Q. Liu, J. Lee, and M. Jordan. A Kernelized Stein Discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284. PMLR, 2016.

[28] C. Ma, S. Tschiatschek, J. M. Hernández-Lobato, R. Turner, and C. Zhang. VAEM: a Deep Generative Model for Heterogeneous Mixed Type Data. *arXiv preprint arXiv:2006.11941*, 2020.

[29] C. Ma, S. Tschiatschek, K. Palla, J. M. Hernández-Lobato, S. Nowozin, and C. Zhang. EdDI: Efficient Dynamic Discovery of High-Value Information with Partial VAE. *arXiv preprint arXiv:1809.11142*, 2018.

[30] L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther. BIVA: A very deep hierarchy of latent variables for generative modeling. *arXiv preprint arXiv:1902.02102*, 2019.

[31] P.-A. Mattei and J. Frellsen. MIWAE: Deep generative modelling and imputation of incomplete data sets. In *International Conference on Machine Learning*, pages 4413–4423. PMLR, 2019.

[32] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. Active feature-value acquisition for classifier induction. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 483–486. IEEE, 2004.

[33] A. Nazabal, P. M. Olmos, Z. Ghahramani, and I. Valera. Handling Incomplete Heterogeneous Data using VAEs. *Pattern Recognition*, 107:107501, 2020.

[34] R. M. Neal et al. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[35] A. Razavi, A. v. d. Oord, B. Poole, and O. Vinyals. Preventing posterior collapse with delta-VAEs. *arXiv preprint arXiv:1901.03416*, 2019.

[36] S. Redyuk, S. Schelter, T. Rukat, V. Markl, and F. Biessmann. Learning to validate the predictions of black box machine learning models on unseen data. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pages 1–4, 2019.

[37] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[38] B. C. Ross. Mutual information between discrete and continuous data sets. *PloS one*, 9(2):e87357, 2014.

[39] M. Saar-Tsechansky, P. Melville, and F. Provost. Active feature-value acquisition. *Management Science*, 55(4):664–684, 2009.

[40] R. Salakhutdinov. Learning deep generative models. *Annual Review of Statistics and Its Application*, 2:361–385, 2015.

[41] R. Salakhutdinov and G. Hinton. Deep Boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455. PMLR, 2009.

[42] T. Salimans, D. Kingma, and M. Welling. Markov Chain Monte Carlo and Variational Inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226. PMLR, 2015.

[43] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder Variational Autoencoders. *Advances in neural information processing systems*, 29:3738–3746, 2016.

[44] M. Thahir, T. Sharma, and M. K. Ganapathiraju. An efficient heuristic method for active feature acquisition and its application to protein-protein interaction prediction. In *BMC proceedings*, volume 6, pages 1–9. BioMed Central, 2012.

[45] A. Thin, N. Kotelevskii, A. Doucet, A. Durmus, E. Moulines, and M. Panov. Monte carlo variational auto-encoders. In *International Conference on Machine Learning*, pages 10247–10257. PMLR, 2021.

[46] A. Vahdat and J. Kautz. NVAE: A Deep Hierarchical Variational Autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

[47] Y. Wang and J. P. Cunningham. Posterior collapse and latent variable non-identifiability. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2020.

[48] Y. Wang, B. Dai, G. Hua, J. Aston, and D. P. Wipf. Green generative modeling: Recycling dirty data using recurrent Variational Autoencoders. In *UAI*, 2017.

[49] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[50] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt. Advances in Variational Inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 3.7 about the computational cost.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not believe there could be any potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include the code for reproducing all the presented experiments in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.1 with all the training details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We include error intervals in all our experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.1 with all the training details.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We have cited every dataset/model employed or mentioned in our work.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include the code for reproducing all the presented experiments in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Experimental setup

The networks for the encoder of the model with the MNIST datasets are 2 layered Deep CNNs with $\{16, 32, 32\}$ output channels, kernel size 5, stride 2 and padding 2. They are followed by MLPs for obtaining the variational parameters for each layer. The decoder that obtains $p_\theta(\boldsymbol{x}|\boldsymbol{h}_1)$ is the symmetric convolutional network. All the neural networks employed in the models trained with UCI datasets are one single layer MLPs with 256 hidden units. The noise variance for Gaussian likelihoods is set up to 0.1.

We employ a learning rate of 1×10^{-3} for all the parameters except for the inflation parameter $\boldsymbol{s}$, where we increase to 1×10^{-2} for a faster convergence. A batch size of 100 is used for all the models except for Yatch and Wine dataset, where we use 20 samples per batch. The number of training steps is 20×10^3 for Boston, Energy, Wine, Yatch, Concrete, Diabetes and Yatch, and 50×10^3 for Naval, Avocado, Bank, and Insurance. For MNIST and Fashion-MNIST, we have 100×10^3 training steps. For the marginal VAEs stage, we employ 1×10^3 training steps. The dimension of the latent variables is $[d_1 = 10, d_2 = 5]$ for Boston, Energy, Wine, Naval Avocado, Bank and Insurance $[d_1 = 4, d_2 = 2]$ for Concrete, Yatch and Diabetes and $[d_1 = 20, d_2 = 10]$ for MNIST and Fashion-MNIST.

We use $L = 5$ Leapfrog steps, chains of $T = 10$ for Boston, Energy, Wine, Naval Avocado, Bank, Insurance, MNIST and Fashion-MNIST, and $T = 5$ for Concrete, Yatch and Diabetes. The SKSD function is estimated using 30 HMC samples.

For the MNIST datasets, the use of Nvidia P100 GPU with Pascal architecture sped up the training with the CNN-based architecture. For the UCI datasets, due to the use of small networks, the differences using CPU or GPU are almost imperceptible.

A.2 Balancing the KLs

Following [46], we define a short initial warming stage (10% of the total training steps) during the optimization where the KLs are balanced according to their magnitude and the corresponding latent dimension, preventing the model for falling into posterior collapse. A factor is applied to each KL, following

$$\gamma_l = \frac{d_l \, \mathbb{E}_{x\sim B}\left[\mathrm{KL}(q(\boldsymbol{\epsilon}_l|\boldsymbol{x})||p(\boldsymbol{\epsilon}))\right]}{\sum_{i=1}^{L} d_i \, \mathbb{E}_{x\sim B}\left[\mathrm{KL}(q(\boldsymbol{\epsilon}_i|\boldsymbol{x})||p(\boldsymbol{\epsilon}))\right]}. \tag{15}$$

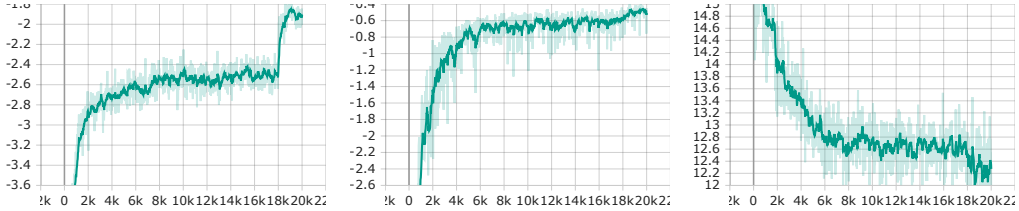
The factors penalises the fact that a layer might be ignored by making the KL smaller when its magnitude is small compared to the rest layers.

A.3 Datasets information

Apart from the MNIST and Fashion-MNIST datasets, a total of 10 UCI datasets have been employed in this work including: Bank Marketing, Insurance Company Benchmark, Avocado sales, Naval Propulsion Plants, Yatch Hydrodynamics, Diabetes, Boston Housing, Wines, Energy efficiency and Bank Marketing.

A.4 Convergence of HMC optimization

We face the computational cost of running HMC by defining a small percentage of training steps for the last stage in Algorithm 1. A 10% of the total training steps for T_{HMC} is sufficient for obtaining the convergence. In Figure 6 we include the validation metrics obtained during the optimization of Yatch dataset, where $T_{HMC} = 2e3$.



(a) $\log p(\boldsymbol{x}_U|\boldsymbol{x}_O)$ (b) $\log p(\boldsymbol{y}|\boldsymbol{x}_O)$ (c) RMSE

Figure 6: Validation curves during optimization for Naval dataset.

A.5 Outlier Detection

The unsupervised task of Outlier Detection (OD) or Identification can be naturally handled with VAEs by using the reconstruction term $\mathbb{E}_{q_z}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right]$. Nonetheless, they are not always robust against outliers, due to their high flexibility [18]. All the existing approaches approximate this score by sampling from the Gaussian variational approximation of the true posterior [1, 14, 36, 48].

	RF	SVM	MLP	**HH-VAEM**
Avocado	0.49 ± 0.01	0.48 ± 0.01	0.50 ± 0.01	$\mathbf{0.32 \pm 0.00}$
Yatch	19.64 ± 6.20	19.62 ± 5.94	19.63 ± 6.12	$\mathbf{11.82 \pm 0.22}$
Diabetes	92.47 ± 6.13	94.73 ± 7.04	106.69 ± 4.87	$\mathbf{58.40 \pm 3.18}$
Boston	12.37 ± 0.85	11.23 ± 0.65	12.86 ± 0.92	$\mathbf{4.58 \pm 0.08}$

Table 7: Test RMSE/error rate for deterministic baselines.

	VAE	H-VAE	HMC-VAE	**HH-VAE**
MNIST vs F-MNIST	0.95 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	$\mathbf{0.98 \pm 0.00}$
F-MNIST vs MNIST	0.81 ± 0.00	$\mathbf{0.84 \pm 0.00}$	0.81 ± 0.00	$\mathbf{0.84 \pm 0.01}$

Table 8: Average Precision-Recall intervals obtained when detecting OoD samples with our model in the MNIST dataset.

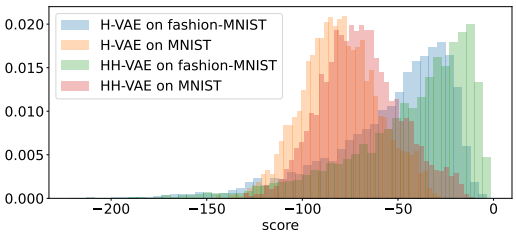
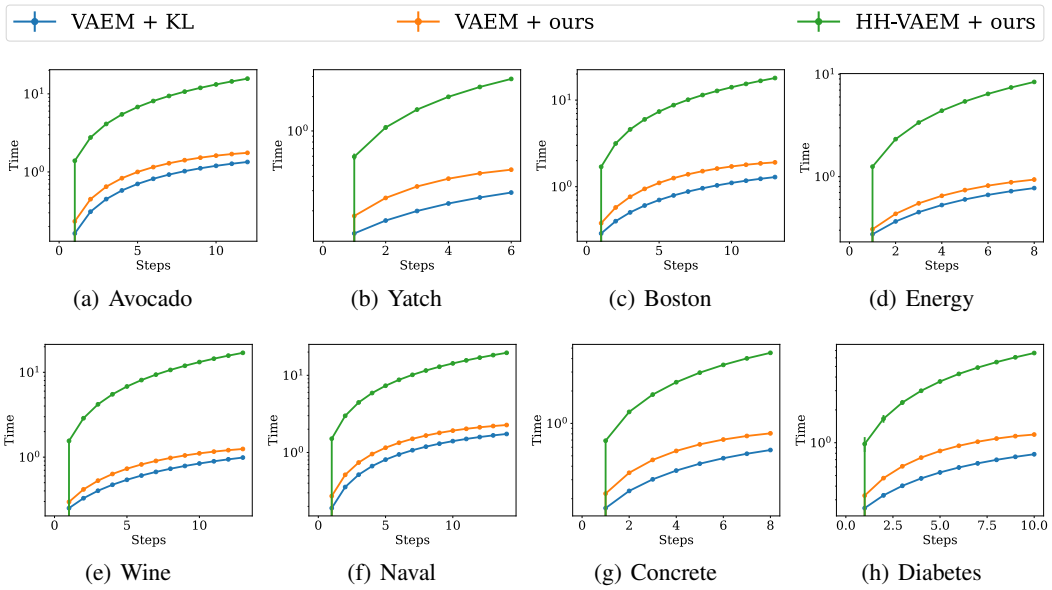


Figure 7: Normalized histograms of the scores obtained on test samples from the original dataset (Fashion-MNIST) and the OoD dataset (MNIST) employing the studied models.

In this experiment, we evaluate our model in detecting Out of Distributions (OoD) samples. We use the metric in equation (14) on the observed data as score. Results for this evaluation on the MNIST datasets are included in Table 8, where the second dataset within each line is the OoD dataset. The hierarchy plays a vital role due to the increased flexibility of the model, while the gain provided by incorporating HMC is not considerably big. HMC is expected to increase the performance of the model in reconstructing data, which affects both in-domain and out-of-domain data.

In Figure 7, histograms of the scores provided by HH-VAEM and the Gaussian standard version without HMC (H-VAEM) are included. As one may appreciate, HMC improves the reconstruction in both the domain and out-of-domain data, which shifts both histograms to the right.



(a) Avocado (b) Yatch (c) Boston (d) Energy

(e) Wine (f) Naval (g) Concrete (h) Diabetes

Figure 8: SAIA time curves. Horizontal axis shows acquisition steps (number of discovered features). Vertical axis is the elapsed time.

A.6 Deterministic baselines

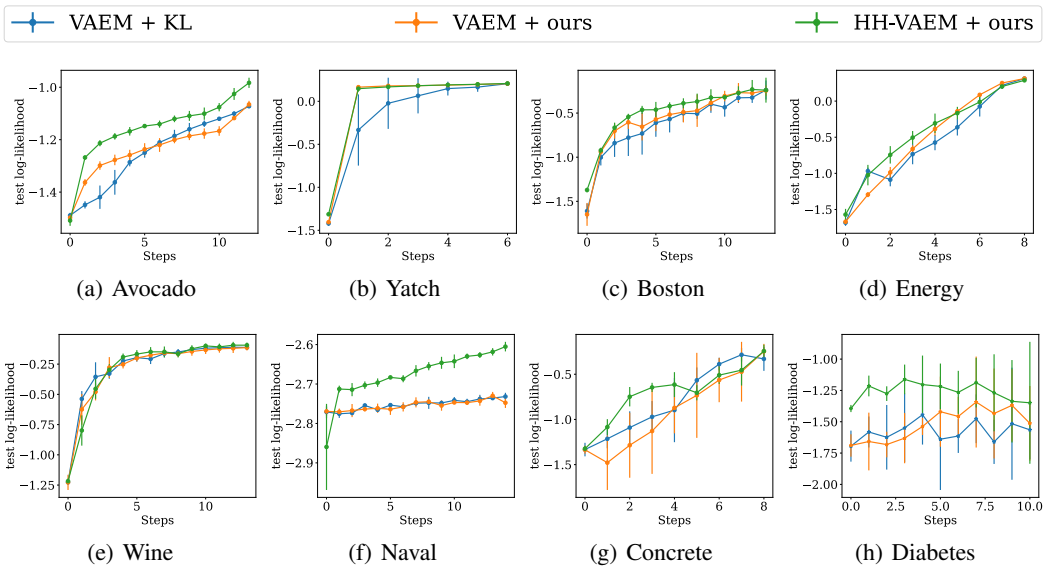
We include in Table 7 results on the RMSE obtained with other discriminative validated predictors, using mean imputation under the same missing rates. In all cases, HH-VAEM outperforms the baselines.

A.7 SAIA times

The times for obtaining the SAIA metric curves in Section 5.3 are included in Figure 8. Although the performance is improved with HH-VAEM, it requires considerably higher time than the baselines to evaluate the reward, due to the HMC algorithm for sampling from the true posterior. Future work might be oriented in proposing ways to measure and reduce this gap.

A.8 SAIA log-likelihoods

In order to extend the results provided in Section 5.3, we include here the log-likelihoods curves when dynamically selecting features using the same procedure (Figure 9).



(a) Avocado (b) Yatch (c) Boston (d) Energy

(e) Wine (f) Naval (g) Concrete (h) Diabetes

Figure 9: SAIA log-likelihood curves. Horizontal axis shows acquisition steps (number of discovered features). Vertical axis is the log-likelihood of the target $\log p(\boldsymbol{y}|\boldsymbol{x}_O)$.