# 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 [34, 35, 58] 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 [12, 19, 45, 47] and mixed-type data [24, 44, 50], 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 [63, 69]. More specifically, Hamiltonian Monte Carlo (HMC) [6, 17] 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 [10] 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 [11, 46, 66, 72]. 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 [34, 58] 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 [13, 76] optimizing a lower bound (ELBO) on the log evidence: $\mathbb{E}_{q_{\boldsymbol{z}}(\boldsymbol{z}|\boldsymbol{x})} \log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\psi(\boldsymbol{z}|\boldsymbol{x})}$, 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, 50]. Nonetheless, the problem of handling unbalanced likelihoods leads to the domination of some dimensions during the optimization process. In [44], 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. This methodology is completely valid when using the missing-at-random (MAR) assumption [41], i.e. assuming the missing mechanism is independent of the missing values. In this our work, we use the same assumption. As proposed in [50] and [47], the ELBO objective is transformed into a lower bound on the observed data, and the unobserved data is replaced with zeros.

## 2.2 Hierarchical VAEs

Hierarchical models have been successfully employed in deep generative modeling, [3, 61, 62]. 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* [46, 55, 73]. In the last few years, several works have investigated possible hierarchical structures for VAEs. In [66], 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 [46], the authors use a bidirectional stochastic inference path. More recently, [72] or [11] have adapted these architectures to complex datasets and high quality images. Possibly motivated by the

residual connections in [32], 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, 17, 52] 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 following updates are repeated for $l = 1 : LF$ steps:

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

where $\odot$ refers to the Hadamard product, and $\phi$ 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

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

For the consecutive $T$ proposals, a new momentum $\boldsymbol{r}$ is resampled and the updates of eq. (1) are repeated for $LF$ steps to update the state if $(\boldsymbol{z}', \boldsymbol{r}')$ is accepted.

### 2.4 Active Feature Acquisition

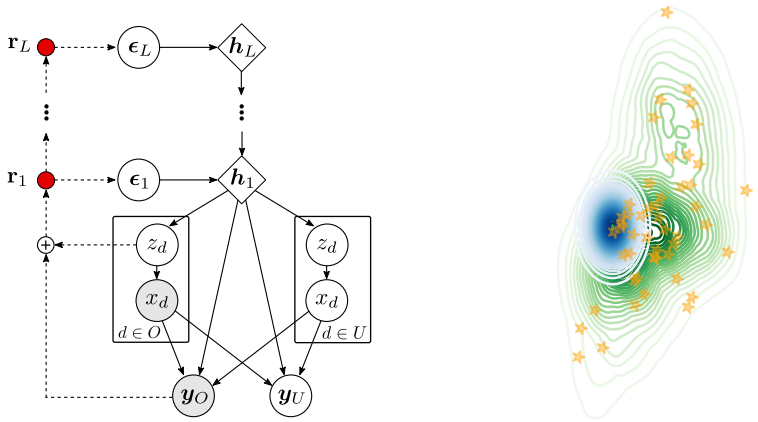
Among all the Active Learning techniques, Active Feature Acquisition [29, 48, 60, 68] is of special interest in cost-sensitive applications for modeling a trade-off between the improvement of predictions and the cost of acquiring new data at the feature level. Several works in the recent literature have 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 [45], it is proposed an efficient method for approximating a non-tractable reward by using the encoder of a VAE that handles missing data. In [44] 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.

### 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 $\boldsymbol{x}$ is denoted by $x_d$. A




(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 $M = 2$.

training set 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 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 [44], 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

$$\mathcal{L}_d(x_d; \{\theta_d, \gamma_d\}) = \mathbb{1}(x_d \in \boldsymbol{x}_O)\mathbb{E}_{q_{\gamma_d}(z_d|x_d)} \log \frac{p_{\theta_d}(x_d, z_d)}{q_{\gamma_d}(z_d|x_d)}, \tag{3}$$

where $\mathbb{1}(x_d \in \boldsymbol{x}_O)$ is an indicator function that activates the ELBO when the feature is observed. Under the missing-at-random assumption, which the one considered in this work, Equation (3) leads to a lower bound on observed data likelihoods. 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 [18, 50] 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

$$\mathcal{L}(\boldsymbol{x}_O, \boldsymbol{y}_O; \{\theta, \psi\}) = \mathbb{E}_{q_\psi}\left[\log \frac{p_\theta(\boldsymbol{z}_O, \boldsymbol{y}_O, \boldsymbol{\epsilon})}{q_\psi(\boldsymbol{\epsilon}|\boldsymbol{z}_O, \boldsymbol{x}_O, \boldsymbol{y}_O)}\right] \tag{4}$$

where $\boldsymbol{\epsilon} = \{\boldsymbol{\epsilon}_1, ..., \boldsymbol{\epsilon}_L\}$ is a set of reparameterized hierarchical latent variables. Further details on the design of the hierarchical dependency VAE are provided below.

### 3.3 Predictive enhancement

The combination of generative and discriminative models is an effective well-studied strategy for dealing with predictive models under missing data [21, 71]. In [21], they model $p(\boldsymbol{x})$ for imputing missing data using a Gaussian Mixture Model. In a deep learning context, recent supervised VAE models have revisited this combination [30, 64] or used factorisations of type $p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{y}|\boldsymbol{z})$ [31] to learn meaningful representations. In [39] the authors propose a deep generative model with factorisation $p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z})$ for detecting adversarial attacks.

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 vector $\hat{\boldsymbol{x}} = (x_i \in \boldsymbol{x}_O,\ \hat{x}_j \in \boldsymbol{x}_U)$ includes the observed part and imputation of the missing part $\hat{x}_j$ by decoding the latent $\boldsymbol{h}$ into $\boldsymbol{z}$ using $p(\boldsymbol{z}|\boldsymbol{h_1})$, and each dimension $z_j \in \boldsymbol{z}$ into $\hat{x}_j$ using $p(x_j|z_j)$. The predictor parameters $\theta_y$ are optimized along with the decoder parameters $\theta_z$.

### 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, as stated in [5, 6], HMC can be pathological when used for sampling from hierarchical densities, where the magnitude of autoregressive variations increase with the depth. For approximating the Hamiltonian dynamics, inside each Leapfrog integrator step, gradients $\nabla_{\boldsymbol{h}_{1:L}} \log p^*(\boldsymbol{h}_{1:L})$ are required. Due to the strong curvature regions, huge norm of high-order derivatives are backpropagated and might eventually explode, ending in overflow issues (Figure 35 in [6]).

If we were to run our HMC method over the hierarchical variables without any reparameterization (Figure 4a), by the time the states reached the aforementioned problematic regions, the integrator would diverge and we would experience the aforementioned overflow problems. By rejecting these problematic states, chains would get stuck close to the proposal and the hierarchical density would not be properly explored (concluding that HMC would not improve the Gaussian proposal). To give an example, in [72], the AR path $p(\boldsymbol{h}_l|\boldsymbol{h}_{<l})$ would provoke instabilities inside the HMC integrator due to huge gradients in $\nabla_{\boldsymbol{h}_{1:L}} \log p^*(\boldsymbol{h}_{1:L})$. This is considered as a pathological design and thus is not included in the validation experiments.

We successfully solved this issue by introducing a hierarchical reparameterization technique. The representation at each layer is reparameterized from variable $\boldsymbol{\epsilon}_l$ with standard Gaussian prior $p(\boldsymbol{\epsilon}_l)$

$$\boldsymbol{h}_l = f_{\mu_l}(\boldsymbol{h}_{l+1}) + f_{\sigma_l}(\boldsymbol{h}_{l+1}) \cdot \boldsymbol{\epsilon}_l, \tag{5}$$

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, thanks to this trick, 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 our HMC optimization, detailed in section 3.5, and avoids the necessity of employing more advanced HMC samplers like [6, 22]. We include further details on the pathological behavior and demonstration of the effectiveness of our solution in Section A.9. Provided the promising results we obtain in Section 5, we propose our reparameterization trick as a novel contribution for solving the pathological behavior of HMC combined with hierarchical VAEs.

For the sake of simplicity, we name the generative parameters as $\theta = \{\theta_z, \theta_y, \theta_1, ..., \theta_L\}$. The dependency ELBO under this hierarchical reparameterized model becomes

$$\begin{gathered}\mathcal{L}_{VI}(\boldsymbol{x}_O, \boldsymbol{y}_O; \{\theta, \psi\}) = \mathbb{E}_{q_\psi}\left[\log \frac{p_\theta(\boldsymbol{z}_O, \boldsymbol{y}_O, \boldsymbol{\epsilon})}{q_\psi(\boldsymbol{\epsilon}|\boldsymbol{z}_O, \boldsymbol{x}_O, \boldsymbol{y}_O)}\right] = \\ \mathbb{E}_{q_\psi}\left[\log p_\theta(\boldsymbol{z}_O|\boldsymbol{h}_1) + \log p_\theta(\boldsymbol{y}_O|\hat{\boldsymbol{x}}, \boldsymbol{h}_1)\right] - \sum_{l=1}^{L} D_{\mathrm{KL}}\left(q_\psi(\boldsymbol{\epsilon}_l|\boldsymbol{x}_O, \boldsymbol{y}_O)||p(\boldsymbol{\epsilon}_l)\right),\end{gathered} \tag{6}$$

where we name $\boldsymbol{r}_l$ the hidden representation at each layer, and defining $\boldsymbol{r}_0 = \{\boldsymbol{x}_O, \boldsymbol{y}_O\}$, we employ NNs with parameters $\psi_{r_l}$ 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 $\psi_{\mu_l}$ and $\psi_{\sigma_l}$. With compactness purposes, we will denote the encoder parameters as $\psi = \{\psi_1, ..., \psi_L\}$, with $\psi_l = \{\psi_{r_l}, \psi_{\mu_l}, \psi_{\sigma_l}\}$.

### 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 [10, 28]. 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 [9] 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 (6), and ii) using the pre-trained encoder for starting from a good proposal [28], 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 $\phi$, a $T \times d$ matrix containing the step sizes for each dimension at each step of the Leapfrog integration. Within this perspective, the hyper-parameters can be optimized using variational inference by maximizing the ELBO

$$\mathbb{E}_{q_\phi^{(T)}(\boldsymbol{\epsilon})}[\log p(\boldsymbol{z}_O, \boldsymbol{y}_O, \boldsymbol{\epsilon})] + H[q_\phi^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{x}_O, \boldsymbol{y}_O)], \tag{7}$$

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

$$\mathcal{L}_{HMC}(\boldsymbol{z}_O, \boldsymbol{y}_O; \{\theta, \psi, \phi\}) = \mathbb{E}_{q_\phi^{(T)}(\boldsymbol{\epsilon})}[\log p_\theta(\boldsymbol{z}_O|\boldsymbol{h}_1) + \log p_\theta(\boldsymbol{y}_O|\hat{\boldsymbol{x}}, \boldsymbol{h}_1) + \sum_{l=1}^{L} p(\boldsymbol{\epsilon}_l^{(T)})]. \tag{8}$$

Nevertheless, the entropy term $H[q_\phi^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{x}_O, \boldsymbol{y}_O)]$ in Equation (7) 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, ending in the proposal $\prod_{l=1}^{L} \mathcal{N}(g_{\mu_l}(\boldsymbol{r}_l),\ s_l \cdot g_{\sigma_l}(\boldsymbol{r}_l))$. Whilst in [9] 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 inflations we ensure a wider coverage of the density by minimizing the Sliced Kernelized Stein Discrepancy (SKSD) [23]

$$\mathcal{L}_{SKSD}(\boldsymbol{x}_O, \boldsymbol{y}_O; \boldsymbol{s}) = \text{SKSD}\left(q_\phi^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{z}_O, \boldsymbol{x}_O, \boldsymbol{y}_O; \boldsymbol{s}),\ p(\boldsymbol{\epsilon}|\boldsymbol{z}_O, \boldsymbol{x}_O, \boldsymbol{y}_O)\right), \tag{9}$$

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 [42] in high dimensional latent spaces.

We provide in Section A.3 a toy demonstration on the efficacy of the HMC optimization, and in Section A.4 a demonstration of the optimization convergence.

### 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 (1) 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 with smaller lengths. To reduce the computational cost, we optimize hyperparameters in a final training stage, since convergence is rapidly achieved (as observed empirically in Fig. 7). At test, samples from the Gaussian $q^{(0)}$ (faster and cheaper), or from HMC $q^{(T)}$ (slower and better) can be used to fit computational constraints.

**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:** $\gamma$, $\theta$, $\psi$, $\phi$, $s$
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**

# 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 [44, 45] 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 [40]. Following [4], we can define it as

$$R(i, \boldsymbol{x}_O) = \mathbb{E}_{p(x_i|\boldsymbol{x}_O)} \, D_{\text{KL}} \left( p(\boldsymbol{y}|x_i, \boldsymbol{x}_O) p(\boldsymbol{y}|\boldsymbol{x}_O) \right), \tag{10}$$

where $i$ is the index of each unobserved feature. Intuitively, this quantity can be interpreted as the expected change in the predictive distribution when $\boldsymbol{x}_i$ is observed. 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 (10) 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 [44, 45] 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 one or two dimensions), an approximation using histograms is more effective. Concretely, the reward in (10) can be rewritten as

$$D_{\text{KL}} \left[ p(\boldsymbol{y}, x_i|\boldsymbol{x}_O) || p(\boldsymbol{y}|\boldsymbol{x}_O) p(x_i|\boldsymbol{x}_O) \right] = \mathcal{I}(\boldsymbol{y}; x_i \, |\boldsymbol{x}_O), \tag{11}$$

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 [37, 59], many are not easilly adapted for parallelization. We demonstrate that the simplest one, based on binning the $x_i$ and $\boldsymbol{y}$ domains

$$\hat{I}(\boldsymbol{y}; \boldsymbol{x}_i \, |\boldsymbol{x}_O) \approx \sum_{ij} p(i,j) \log \frac{p(i,j)}{p_x(i) p_y(j)}, \tag{12}$$

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. The number of bins defines width of uniformly distributed intervals over $x_d$ and $\boldsymbol{y}$ supports. Since this estimator is sampling-based, under certain conditions (namely, if all densities exist as proper functions), Equation (13) indeed converges to $\mathcal{I}(\boldsymbol{y}; x_i \, |\boldsymbol{x}_O)$ if we first let the number of samples $N \to \infty$ [37].

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

Table 1: Test NLL of the unobserved features for our model and baselines.

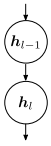
| | Bank | Insurance | Avocado | Naval | Yatch | Diabetes | Concrete | Wine | Energy | Boston |
|---|---|---|---|---|---|---|---|---|---|---|
| VAEM | $0.56 \pm 0.06$ | $1.20 \pm 0.03$ | $1.18 \pm 0.02$ | $2.69 \pm 0.01$ | $0.61 \pm 0.02$ | $1.59 \pm 0.19$ | $1.07 \pm 0.09$ | $0.28 \pm 0.09$ | $0.61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

Table 2: Test NLL of the predicted target for our model and baselines.

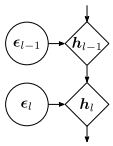
# 5 Experiments

The evaluation of the HH-VAEM model is organized into three quantitative experiments and one qualitative experiment. The ablation study includes the validation of our proposed HMC-based, hierarchical model with respect to the Gaussian and one-layered alternatives. Namely, the comparison is performed with the following baseline models:

- *VAEM*: 1 layer, Gaussian-based VAEM [44] (without including the Partial VAE).
- *MIWAEM*: 1 layer, Gaussian-based, importance weighted IWAEM (VAEM + IWAE [47]).
- *H-VAEM*: 2 layers, Gaussian-based VAEM.
- *HMC-VAEM*: 1 layer, HMC-based (with our optimization method) VAEM.




(a) Autoregressive




(b) Reparameterization

Figure 4: AR (a) vs reparameterized (b).

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

Table 3: Test NLL of the unobserved features for the MNIST datasets.

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

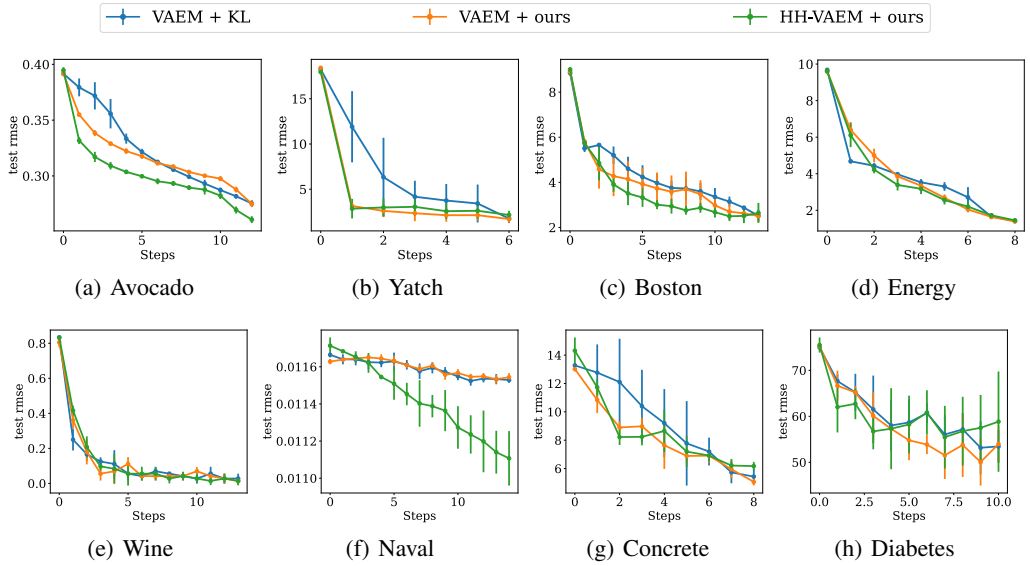
Table 4: Test NLL of the predicted target for the MNIST datasets.

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

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

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.

For the quantitative experiments, a total of 10 UCI datasets [16] that include mixed-type data are employed for the evaluation. We include both MNIST [38] and Fashion-MNIST [75] datasets for evaluating our model in higher dimensional observational and latent spaces and bigger architectures (3 layered convolutional nets for encoder/decoder). The qualitative results are included in Section A.8, where we evaluate our model in the image inpainting task on MNIST and CelebA [43]. For these two image datasets, 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 experiments and details on the employed architectures and evaluation times are provided in the Appendix.




(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.

### 5.1 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

$$\log p(\boldsymbol{x}_U|\boldsymbol{x}_O) \approx \log \mathbb{E}_{\boldsymbol{\epsilon}\sim q^{(T)}(\boldsymbol{\epsilon}|\boldsymbol{x}_O)}\left[p(\boldsymbol{x}_U|\boldsymbol{\epsilon})\right] \approx \log \frac{1}{k}\sum_{i}^{k} p(\boldsymbol{x}_U|\boldsymbol{\epsilon}_i), \tag{13}$$

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 for most of the datasets, incremental improvement is obtained: VAEM $<$ H-VAEM $<$ HMC-VAEM $<$ HH-VAEM. Extended results with the imputation error, included in Section A.5, corroborate this.

### 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 the same incremental improvement in the prediction task.

### 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 [44]. 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.

## 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 both quantitative and qualitative 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] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance Weighted Autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[8] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in beta-VAE. *arXiv preprint arXiv:1804.03599*, 2018.

[9] 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.

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

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

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

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

[14] D. Dheeru and E. K. Taniskidou. UCI machine learning repository. 2017.

[15] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

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

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

[18] 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.

[19] 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.

[20] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[21] Z. Ghahramani and M. I. Jordan. Learning from incomplete data. 1995.

[22] M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

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

[24] 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.

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

[26] J. Hernandez-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernández-Lobato, and R. Turner. Black-box alpha divergence minimization. In *International Conference on Machine Learning*, pages 1511–1520. PMLR, 2016.

[27] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. 2016.

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

[29] 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.

[30] N. Ipsen, P.-A. Mattei, and J. Frellsen. How to deal with missing data in supervised deep learning? In *Artemiss-ICML Workshop on the Art of Learning with Missing Values*, 2020.

[31] T. Joy, S. Schmon, P. Torr, S. Narayanaswamy, and T. Rainforth. Capturing label characteristics in vaes. In *Proceedings of the ICLR Conference 2021*. OpenReview, 2021.

[32] 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.

[33] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving Variational Inference with Inverse Autoregressive Flow. *arXiv preprint arXiv:1606.04934*, 2016.

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

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

[36] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing Neural Networks. In *Proceedings of the 31st international conference on neural information processing systems*, pages 972–981, 2017.

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

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

[39] Y. Li, J. Bradshaw, and Y. Sharma. Are generative classifiers more robust to adversarial attacks? In *International Conference on Machine Learning*, pages 3804–3814. PMLR, 2019.

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

[41] R. J. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[42] 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.

[43] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] E. Nalisnick, L. Hertel, and P. Smyth. Approximate Inference for Deep Latent Gaussian Mixtures. In *NIPS Workshop on Bayesian Deep Learning*, volume 2, page 131, 2016.

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

[51] R. M. Neal. *Probabilistic inference using Markov Chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, ON, Canada, 1993.

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

[53] L. Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

[54] I. Peis, P. M. Olmos, and A. Artés-Rodríguez. Unsupervised learning of global factors in deep generative models. *arXiv preprint arXiv:2012.08234*, 2020.

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

[56] 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.

[57] D. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[58] 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.

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

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

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

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

[63] 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.

[64] M. Śmieja, Ł. Struski, J. Tabor, B. Zieliński, and P. Spurek. Processing of missing data by neural networks. *Advances in neural information processing systems*, 31, 2018.

[65] J. Sohl-Dickstein and B. J. Culpepper. Hamiltonian Annealed Importance Sampling for partition function estimation. *arXiv preprint arXiv:1205.1925*, 2012.

[66] 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.

[67] D. J. Stekhoven and P. Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

[68] 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.

[69] 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.

[70] J. Tomczak and M. Welling. VAE with a VampPrior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.

[71] V. Tresp, S. Ahmad, and R. Neuneier. Training neural networks with deficient data. *Advances in neural information processing systems*, 6, 1993.

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

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

[74] 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.

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

[76] 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. For the experiments with CelebA, we use 5 layered Deep CNNs with {32, ,32, 64, 64, 512} output channels, kernel size 4, stride 2 and padding 1, followed by batch norm layers. 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 $1 \times 10^{-3}$ for all the parameters except for the inflation parameter $\boldsymbol{s}$, where we increase to $1 \times 10^{-2}$ 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 $20 \times 10^3$ for Boston, Energy, Wine, Yatch, Concrete, Diabetes and Yatch, and $50 \times 10^3$ for Naval, Avocado, Bank, and Insurance. For MNIST and Fashion-MNIST, we have $100 \times 10^3$ training steps. For the marginal VAEs stage, we employ $1 \times 10^3$ 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 $LF = 5$ 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 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.3 Efficacy of training HMC

We include in this experiment results that demonstrate the efficacy of training the hyperparameters of HMC using our proposed gradient-based strategy on several 2D densities. In Figure 6, each row correspond to an example with a different density: first row is a *wave* density, while second row is a *dual moon* density. In the first column, the initial set up is showed, including the density contour (dark blue), the initial Gaussian proposal contour (light blue) and samples from HMC (green). In both cases, due to the tightness of the proposal, chains do not properly explore the density and get stuck close the initial state. In the second column, results after training HMC hyperparameters and the inflation parameter are included. Again, in both cases, and more specially in the first one, the inflation of the horizontal variance of the proposal successfully increases to better cover the surface, and final HMC samples vastly improve the exploration. In the third column, the approximations of the HMC objective, the SKSD discrepancy and the inflation parameters over the optimization steps are included.

### A.4 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 7 we include the validation metrics obtained during the optimization of Yatch dataset, where $T_{HMC} = 2e3$.

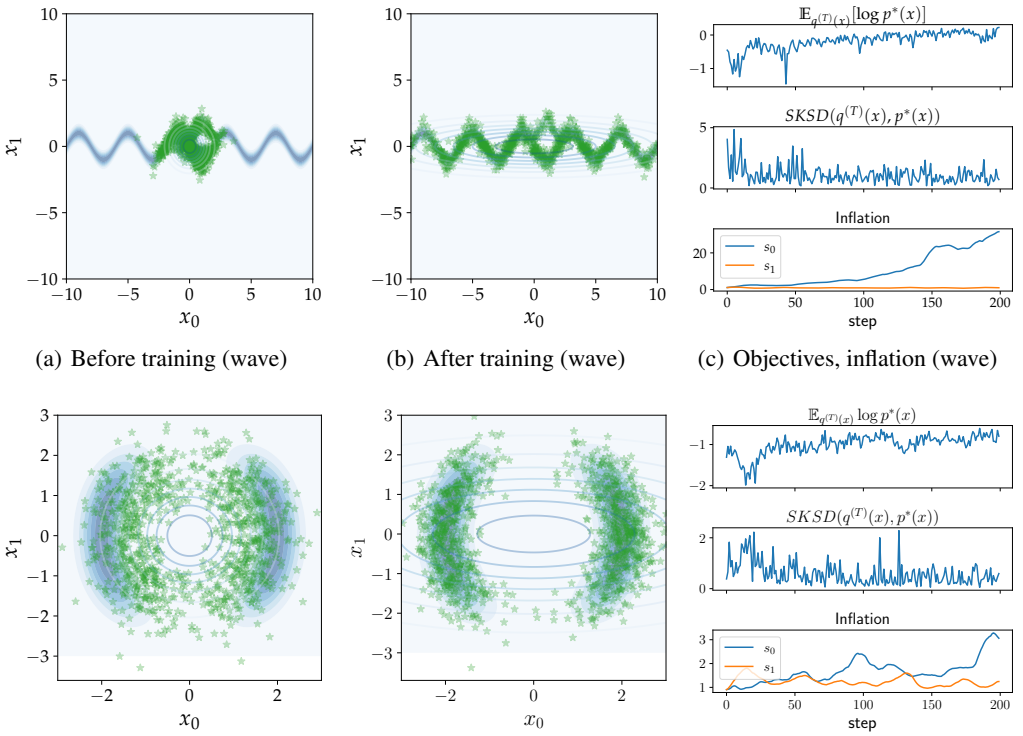
In figure 8 (a), the mean acceptance rate of the HMC sampler over the training steps and the step sizes (b) are included. The steps are initialized from $U(0.05, 0.2)$. After $2 \times 10^3$ steps, the mean acceptance rate converges to a value closer to $\bar{p}_a = 0.65$, which is defined as the optimal desired acceptance probability [52]. This empirical result provides evidence that, apart from reducing the computational cost, reducing the HMC training step to this value is sufficient for achieving convergence.

### A.5 Deterministic imputation metrics

We include in Table 6 results on the RMSE obtained with other discriminative validated predictors, using mean imputation under the same missing rates. Additionally, we include here the missForest in the baselines, a wide-spread method for missing data imputation using a Random Forest approach [67]. For classification tasks, the error rate is considered. In almost all cases, HH-VAEM outperforms the baselines.

| | Bank | Insurance | Avocado | Naval | Yatch | Diabetes | Concrete | Wine | Energy | Boston |
|---|---|---|---|---|---|---|---|---|---|---|
| missForest | $0.64 \pm 0.01$ | $0.61 \pm 0.06$ | $0.59 \pm 0.02$ | $0.30 \pm 0.01$ | $\mathbf{0.86 \pm 0.12}$ | $0.76 \pm 0.08$ | $0.76 \pm 0.07$ | $0.77 \pm 0.11$ | $0.64 \pm 0.08$ | $0.59 \pm 0.06$ |
| VAEM | $0.57 \pm 0.01$ | $0.40 \pm 0.01$ | $0.59 \pm 0.00$ | $0.33 \pm 0.01$ | $0.93 \pm 0.04$ | $0.79 \pm 0.02$ | $0.74 \pm 0.04$ | $0.64 \pm 0.03$ | $0.75 \pm 0.02$ | $0.65 \pm 0.02$ |
| MIWAEM | $0.56 \pm 0.00$ | $0.39 \pm 0.00$ | $0.59 \pm 0.01$ | $0.34 \pm 0.01$ | $0.94 \pm 0.04$ | $0.75 \pm 0.01$ | $0.71 \pm 0.03$ | $0.63 \pm 0.02$ | $0.75 \pm 0.02$ | $0.63 \pm 0.01$ |
| H-VAEM | $0.56 \pm 0.01$ | $0.39 \pm 0.00$ | $0.58 \pm 0.01$ | $0.32 \pm 0.01$ | $0.92 \pm 0.05$ | $0.77 \pm 0.01$ | $0.71 \pm 0.02$ | $0.60 \pm 0.04$ | $0.55 \pm 0.03$ | $0.59 \pm 0.01$ |
| HMC-VAE | $0.55 \pm 0.01$ | $\mathbf{0.38 \pm 0.00}$ | $0.58 \pm 0.01$ | $0.30 \pm 0.02$ | $0.91 \pm 0.05$ | $0.76 \pm 0.03$ | $0.71 \pm 0.02$ | $0.60 \pm 0.02$ | $0.55 \pm 0.02$ | $0.57 \pm 0.02$ |
| HH-VAEM | $\mathbf{0.54 \pm 0.01}$ | $\mathbf{0.38 \pm 0.00}$ | $\mathbf{0.57 \pm 0.00}$ | $\mathbf{0.29 \pm 0.02}$ | $0.90 \pm 0.00$ | $\mathbf{0.75 \pm 0.01}$ | $\mathbf{0.70 \pm 0.01}$ | $\mathbf{0.59 \pm 0.04}$ | $\mathbf{0.54 \pm 0.01}$ | $\mathbf{0.56 \pm 0.03}$ |

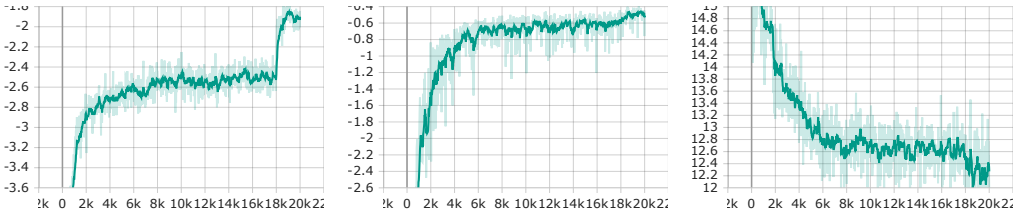
Table 6: Test NLL of the unobserved features for our model and baselines.




(a) Before training (wave) (b) After training (wave) (c) Objectives, inflation (wave)

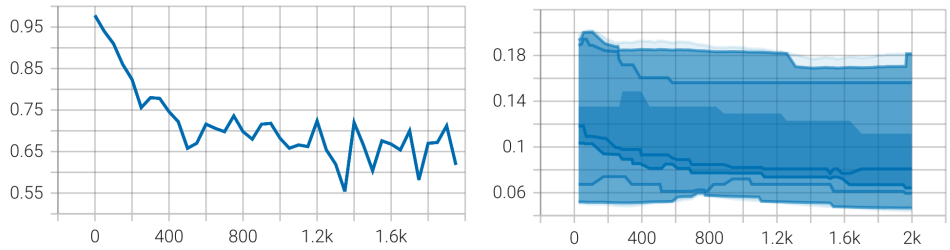
(d) Before training (dual moon) (e) After training (dual moon) (f) Objectives, inflation (dual moon)

Figure 6: Toy example showing the efficacy of training HMC hyperpameters using our method on two densities: wave (top row) with $T = 5$ and dual moon (bottom row) with $T = 10$. Left column illustrates the non desired behavior when chains hardly explore the density and stuck in a small region close to the mass of the tight Gaussian initial proposal (light blue contour ellipses). Right column shows how optimizing the step sizes and the inflation parameter leads to a vast improvement of the exploration. More specifically, (b) justifies scaling each dimension of the target, since the inflation is bigger on the horizontal axis.



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

Figure 7: Validation curves during optimization for Naval dataset.




(a) Mean acceptance rate $\bar{p}_a$ (b) Step sizes

Figure 8: Evolution of mean acceptance rate $\bar{p}_a$ (a) and step sizes (b) over the training steps.

### A.6 Likelihood of the observed features

We include in this section results on the negative log likelihood of the observed features

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

Results are included in Table 7. In almost all the cases, we confirm the incremental superiority when adding each part of our proposed design.

| | bank | insurance | avocado | naval | yatch | diabetes | concrete | wine | energy | boston |
|---|---|---|---|---|---|---|---|---|---|---|
| VAEM | $0.51 \pm 0.05$ | $0.99 \pm 0.05$ | $0.44 \pm 0.01$ | $0.21 \pm 0.01$ | $0.62 \pm 0.13$ | $0.92 \pm 0.12$ | $0.63 \pm 0.18$ | $0.73 \pm 0.18$ | $1.86 \pm 0.09$ | $0.56 \pm 0.11$ |
| MIWAEM | $0.63 \pm 0.02$ | $1.06 \pm 0.03$ | $0.60 \pm 0.03$ | $0.33 \pm 0.01$ | $0.75 \pm 0.07$ | $1.05 \pm 0.06$ | $0.76 \pm 0.09$ | $0.80 \pm 0.06$ | $1.77 \pm 0.15$ | $0.67 \pm 0.03$ |
| H-VAEM | $0.40 \pm 0.04$ | $0.93 \pm 0.04$ | $0.42 \pm 0.05$ | $0.19 \pm 0.07$ | $0.58 \pm 0.09$ | $0.70 \pm 0.13$ | $0.53 \pm 0.18$ | $0.71 \pm 0.15$ | $0.38 \pm 0.02$ | $0.49 \pm 0.07$ |
| HMC-VAE | $0.37 \pm 0.07$ | $\mathbf{0.92 \pm 0.04}$ | $0.39 \pm 0.06$ | $0.18 \pm 0.05$ | $0.54 \pm 0.10$ | $\mathbf{0.68 \pm 0.07}$ | $0.49 \pm 0.22$ | $\mathbf{0.55 \pm 0.07}$ | $0.40 \pm 0.06$ | $0.41 \pm 0.04$ |
| HH-VAEM | $\mathbf{0.33 \pm 0.03}$ | $0.95 \pm 0.05$ | $\mathbf{0.36 \pm 0.01}$ | $\mathbf{0.17 \pm 0.04}$ | $\mathbf{0.45 \pm 0.04}$ | $0.68 \pm 0.16$ | $\mathbf{0.40 \pm 0.16}$ | $0.64 \pm 0.17$ | $\mathbf{0.37 \pm 0.06}$ | $\mathbf{0.41 \pm 0.04}$ |

Table 7: Test NLL of the observed features for our model and baselines.

### A.7 Heterogeneous likelihoods

In experiment 5.1 we reported an average likelihood across heterogeneous variables. Although this quantity is not a valid joint likelihood probability, we employed this average to provide a fair comparison on models that have been trained on the same heterogeneous likelihoods. In this section, we show the comparison on averaging separately the three considered marginal likelihoods (Gaussian, Bernoulli and Categorical) for two of the biggest datasets considered on Tables 8-10. Again, we show the incremental superiority when adding the different design choices of our model.

| | Bank | Avocado |
|---|---|---|
| VAEM | $0.36 \pm 0.29$ | $0.26 \pm 0.08$ |
| MIWAEM | $0.33 \pm 0.27$ | $0.32 \pm 0.06$ |
| H-VAEM | $0.26 \pm 0.22$ | $0.29 \pm 0.07$ |
| HMC-VAEM | $0.25 \pm 0.21$ | $0.25 \pm 0.08$ |
| **HH-VAEM** | $\mathbf{0.20 \pm 0.22}$ | $\mathbf{0.22 \pm 0.07}$ |

Table 8: Average test Gaussian NLL of the observed features.

| | Bank | Avocado |
|---|---|---|
| VAEM | $0.13 \pm 0.00$ | $0.06 \pm 0.00$ |
| MIWAEM | $0.15 \pm 0.00$ | $0.09 \pm 0.00$ |
| H-VAEM | $0.11 \pm 0.00$ | $0.07 \pm 0.00$ |
| HMC-VAEM | $0.08 \pm 0.00$ | $0.05 \pm 0.00$ |
| **HH-VAEM** | $\mathbf{0.07 \pm 0.00}$ | $\mathbf{0.04 \pm 0.00}$ |

Table 9: Average test Bernoulli NLL of the observed features.

| | Bank | Avocado |
|---|---|---|
| VAEM | $0.24 \pm 0.16$ | $0.33 \pm 0.00$ |
| MIWAEM | $0.26 \pm 0.17$ | $0.36 \pm 0.00$ |
| H-VAEM | $0.23 \pm 0.16$ | $0.32 \pm 0.00$ |
| HMC-VAEM | $0.22 \pm 0.15$ | $\mathbf{0.30 \pm 0.00}$ |
| **HH-VAEM** | $\mathbf{0.21 \pm 0.15}$ | $\mathbf{0.30 \pm 0.00}$ |

Table 10: Average test Cat. NLL of the observed features.

### A.8 Conditional image inpainting

In this section, we include qualitative results when comparing our model with the baselines on image conditional inpainting task. A better pixel imputation is expecting when conditioning on the observed pixels, since samples from the approximate posterior are required. We evaluate the inpainting experiment by applying the following sequence of steps:

(i) We generate a square mask of missing pixels covering part of the image (with side of 14 pixels for $28 \times 28$ images, and 32 for $64 \times 64$ images).

(ii) We use the model encoder to obtain $q(\boldsymbol{z}|\boldsymbol{x}_O)$ (or $q(\boldsymbol{\epsilon}|\boldsymbol{x}_O)$ in our reparameterized hierarchy).

(iii) We draw $S$ samples $\{\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(S)}\}$ or $\{\boldsymbol{\epsilon}^{(1)}, ..., \boldsymbol{\epsilon}^{(S)}\}$ of the posterior approximation.

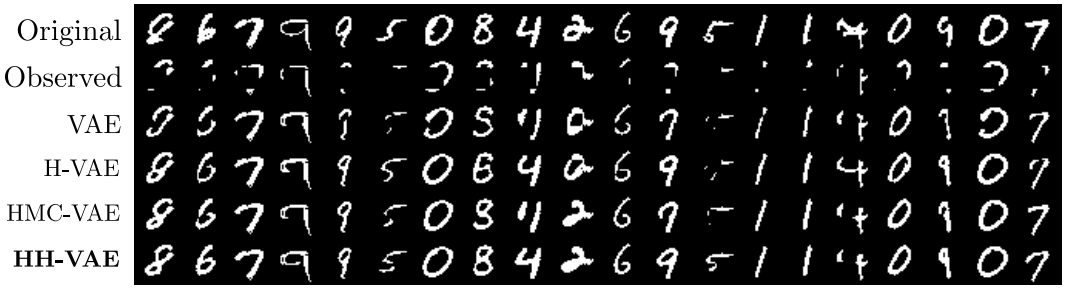
(iv) We use the decoder to obtain $p(\boldsymbol{x}_U|\boldsymbol{z}^{(s)})$ (or $p(\boldsymbol{x}_U|\boldsymbol{h}_1^{(s)})$ for the hierarchical models).

(v) The imputation is obtained by averaging

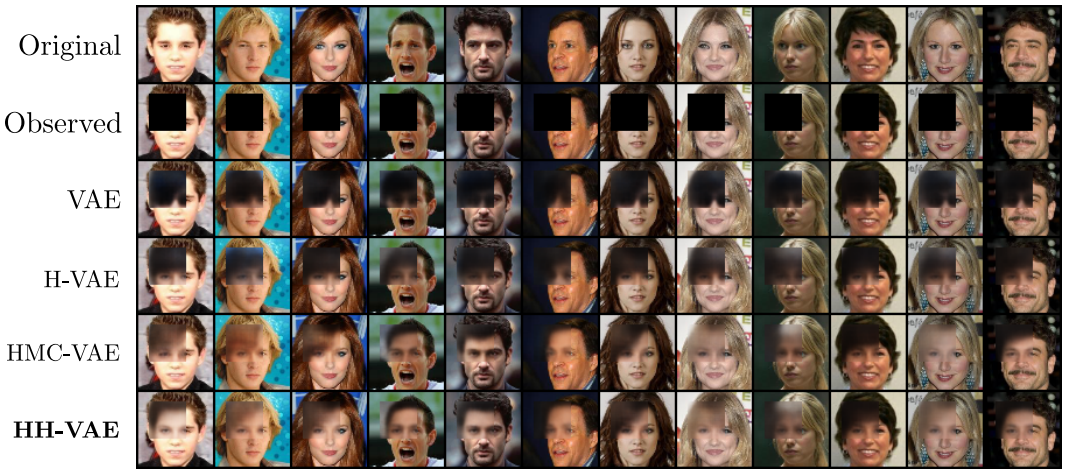
$$\hat{\boldsymbol{x}}_U = \frac{1}{S}\sum_{s=1}^{S} \hat{\boldsymbol{x}}_U^{(s)}$$

where $\hat{\boldsymbol{x}}_U^{(s)}$ are the decoded Bernoulli parameters for each sample.

We include the results for MNIST and CelebA in Figure 9 (a) and (b), respectively. From top to bottom, each row correspond to: i) original image, ii) observed pixels. From iii) to vi): inpainting with VAE, HVAE, HMC-VAE and HH-VAE (last row), respectively. As observed, we prove the hypothesis of obtaining improved imputation when using better samples from our tuned HMC. By observing the results, several conclusions can be made. First, the hierarchical Gaussian-based model (fourth row) considerably improves the one-layered Gaussian alternative. Second, the HMC-based methods (two last rows) vastly improve the Gaussian alternatives. Third, in some specific cases, an extra improvement is added by HH-VAE with respect to the one-layered HMC-based alternative (for example, columns 2, 4, 8, 9, 11, 12 and 13 in Figure 9 (a) or column 1 in (b)).




(a) MNIST




(b) CelebA

Figure 9: Image conditional inpainting on MNIST (a) and CelebA (b). Original images from the test set are included in the first row. Second row is the input to the model, $\boldsymbol{x}_O$, where the black box is the square missing mask $\boldsymbol{x}_U$ manually introduced. In third, fourth, fifth and sixth rows, imputed $\hat{\boldsymbol{x}}_U$ is included for VAE, H-VAE, HMC-VAE and HH-VAE, our model, where $\hat{x}_U$ is decoded from samples of the approximate posterior (Gaussian for VAE and H-VAE, or HMC-based for HMC-VAE and HH-VAE).

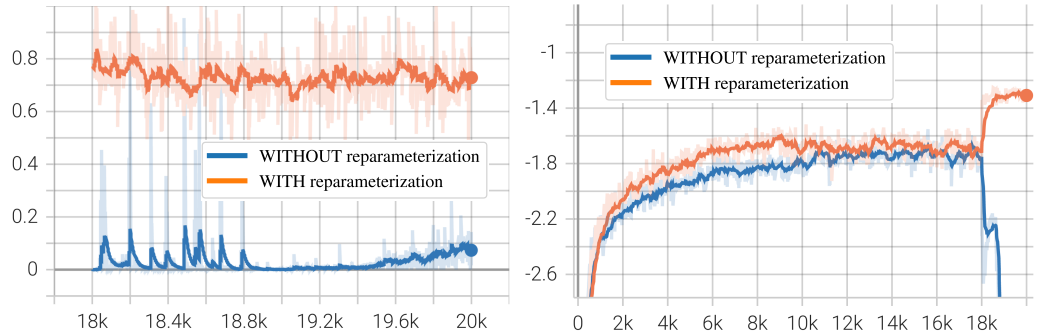
### A.9 Reparameterization trick for solving ill-posed HMC with hierarchical densities

We provide in this section strong empirical demonstration on the efficacy of our proposed reparameterization trick. As stated in Section 3.4, naïve implementations of HMC are ill-posed when combined with hierarchical densities. The autoregressive correlations lead to non-smooth densities with huge peaks. The large gradients $\nabla_{\boldsymbol{h}_{1:L}} \log p^*(\boldsymbol{h}_{1:L})$ evaluated on these regions inside the Leapfrog steps of Equations 1 make huge modifications of the proposed states. If we denote these diverged states by $\boldsymbol{h}_{1:L}^{(ill)}$, evaluating the objective $\log p^*(\boldsymbol{h}_{1:L}^{(ill)})$ lead to overflow issues. This undesired behavior is what we call *divergence of the Leapfrog integrator*, and is also illustrated by [5] in Figure 35.

In order to avoid the aforementioned overflow issues, we can directly reject these problematic states. Nevertheless, by doing this, HMC will not properly explore the density by the time the states fall into the problematic regions, leading to extremely low acceptance rates. To demonstrate this, we include in Figure 10 (a) with blue line the evolution of the acceptance rates when optimizing HMC with the HH-VAEM variant without reparameterization (as illustrated in Figure 4 (a)). The acceptance rate is extremely low as expected, which is a clear indicator of a poorly mixing sampler.

On the contrary, by using our proposed reparameterization trick (Figure 4 (b)), we are able to make HMC work properly and tune the step sizes, getting closer to the ideal acceptance rate $\bar{p}_a = 0.65$ [52]. Evaluations are applied to the same validation split of the Boston dataset. The step sizes of HMC are initialized from $U(0.05, 0.2)$ in both cases.

Further, we also include in Figure 10 (b) the evolution of the imputation log likelihood metric of Equation (13) during the whole optimization for both approaches. First $18 \times 10^3$ steps correspond to the pretraining stage using only the ELBO. Without the reparameterization, due to the low acceptance rate, the states hardly move from the initial proposal. These samples are even worse than the samples provided by the encoder, thus provoking the metric to decrease. We demonstrate again the effectiveness of the reparameterization trick by observing an increase in this metric.



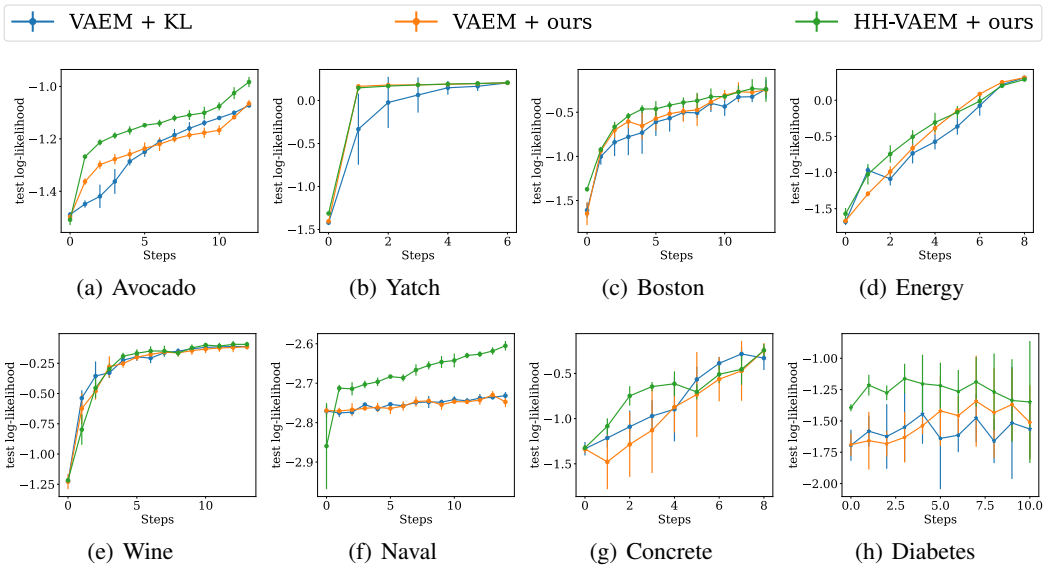

(a) Mean acceptance rate $\bar{p}_a$ (b) $\log p(\boldsymbol{x}_U|\boldsymbol{x}_O)$

Figure 10: Demonstration of the efficacy of our reparameterization trick. Our model is showed with orange lines, and the same model but without the reparameterization with blue lines. In (a), the mean acceptance rate of the HMC proposals over the optimization steps is included. We demonstrate that without the reparameterization, HMC is ill-posed and by rejecting the proposals that make the integrator diverge, the acceptance rate is extremely low, thus not properly exploring the density. In (b), the imputation log likelihood metric is showed for the whole optimization. With the reparameterization trick, we successfully solve the pathological issues, leading to a considerable increase of the metric.

### A.10 Balancing the KLs

Following [72], 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

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




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

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

Figure 11: SAIA log-likelihood curves. Horizontal axis shows acquisition steps (number of discovered features). Vertical axis is the log-likelihood of the target $\log p(\boldsymbol{y}|\boldsymbol{x}_O)$.

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.11 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 11).

### A.12 Training times

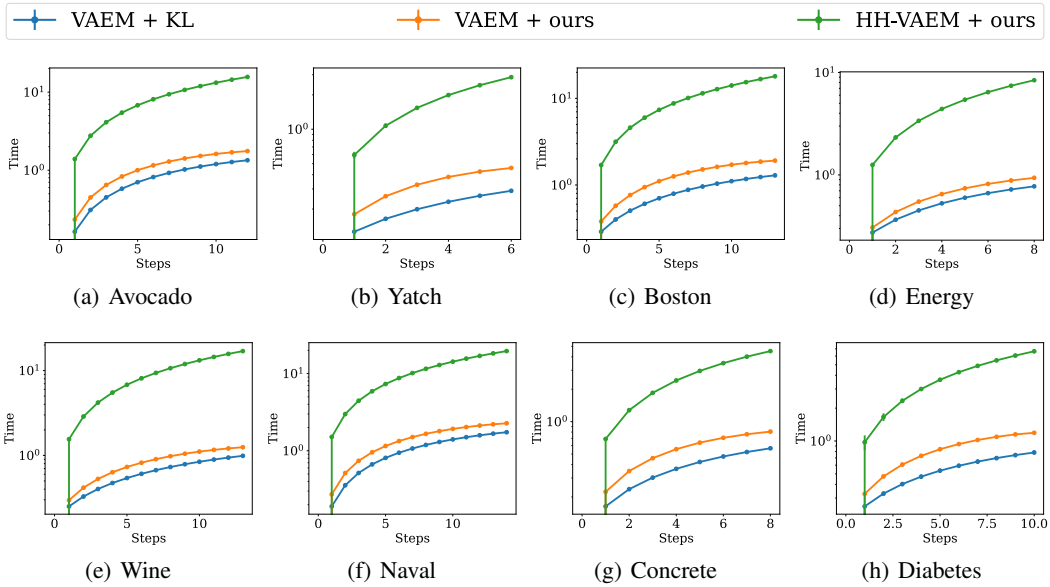
Table 11 shows the average training time in minutes for each model in the experiments for Tables 1 and 2. The ratio between training times for our method and the Gaussian baselines is approximately between 5 and 10.

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

Table 11: Training times (in minutes) of our model and baselines.

### A.13 SAIA times

The times for obtaining the SAIA metric curves in Section 5.3 are included in Figure 12. 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) Avocado (b) Yatch (c) Boston (d) Energy

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

Figure 12: SAIA time curves. Horizontal axis shows acquisition steps (number of discovered features). Vertical axis is the elapsed time.