Variational Mixture of HyperGenerators for Learning Distributions over Functions

Batuhan Koyuncu [1] **Pablo Sánchez-Martín** [2] **Ignacio Peis** [3] **Pablo M. Olmos** [3] **Isabel Valera** [1]

Abstract

Recent approaches build on implicit neural representations (INRs) to propose generative models over function spaces. However, they are computationally costly when dealing with inference tasks, such as missing data imputation, or directly cannot tackle them. In this work, we propose a novel deep generative model, named VaMoH. VaMoH combines the capabilities of modeling continuous functions using INRs and the inference capabilities of Variational Autoencoders (VAEs). In addition, VaMoH relies on a normalizing flow to define the prior, and a mixture of hypernetworks to parametrize the data log-likelihood. This gives VaMoH a high expressive capability and interpretability. Through experiments on a diverse range of data types, such as images, voxels, and climate data, we show that VaMoH can effectively learn rich distributions over continuous functions. Furthermore, it can perform inference-related tasks, such as conditional super-resolution generation and in-painting, as well or better than previous approaches, while being less computationally demanding.

1. Introduction

While many real-world applications lead to data over continuous coordinate systems, such data is often discretized, e.g., by fixing the resolution of images (Simonyan & Zisserman, 2014) or assuming a fixed sample frequency in time-series (Hochreiter & Schmidhuber, 1997). In contrast, recent advances in Implicit Neural Representations (INR) have been shown to be powerful approaches for directly parameterizing continuous functions by mapping coordinates into data features. To name a few examples, INRs have been successfully applied in diverse fields such as image representation (Stanley, 2007; Ha, 2016), shape and scene representation (Mescheder et al., 2019; Genova et al., 2019; 2020; Chen & Zhang, 2019; Zeng et al., 2022; Sitzmann et al., 2019; Jiang et al., 2020; Mildenhall et al., 2021), audio (Sitzmann et al., 2020), graphs (Grattarola & Vandergheynst, 2022), and data manifolds (Dupont et al., 2022b;a).

A couple of recent works (Dupont et al., 2022b;a) have relied on INRs to generate data at any continuous coordinate (e.g., to generate images of different resolutions). Dupont et al. (2022a) disentangle the task of learning functions using an INR from the data generation task; and, Dupont et al. (2022b) proposes a *hypergenerator* based on generative adversarial networks (GANs), where the parameters of its generator are the outputs of another network. However, they both suffer from limitations, especially with regard to conditional generation tasks such as image in/out-painting. The implicit nature of Dupont et al. (2022b) does not provide straightforward ways for conditional generation, and Dupont et al. (2022a) requires solving a computationally expensive numerical optimization problem to generate the modulation vector of each new data point (i.e., unseen during training).

In this paper, we propose a Variational Mixture of HyperGenerators for learning distributions over functions, referred to as VaMoH. Our model relies on a mixture of hyper variational autoencoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014; Nguyen et al., 2021), where:

i) a planar normalizing flow (Rezende & Mohamed, 2015) is used as prior distribution over its latent variables to be able to fit and generate complex data;
ii) a hypernetwork is used to parameterize the mixture of decoders that, in turn, partition the function space into meaningful regions (e.g., into visual segmentation maps as shown in Figure 9); and,
iii) analogous to Dupont et al. (2022b), a PointConv network (Wu et al., 2019) is used in the encoder to map any cloud of points, i.e., any set of continuous coordinates and features, into a fixed-sized vector.

As demonstrated by our extensive experiments on several benchmark datasets, VaMoH can accurately and efficiently

[1]Saarland University, Saarbrücken, Germany [2]Max Planck Institute for Intelligent Systems, Tübingen, Germany [3]Universidad Carlos III de Madrid, Madrid, Spain. Correspondence to: Batuhan Koyuncu <koyuncu@cs.uni-saarland.de>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

learn distributions over functions and, thus, generate data over continuous coordinate systems. Remarkably, in contrast to prior work, conditional generation (e.g., in/out-painting tasks or generating a higher-resolution version of a given image) is straightforward in VaMoH as it just requires a forward pass on the model (independently of whether the conditioning data were seen during training).

2. Related Work & Background

Variational Autoencoders (VAEs). (Kingma & Welling, 2013; Rezende et al., 2014) approximate the intractable posterior over latent variables $p(\boldsymbol{z}|\boldsymbol{x})$ by performing amortized variational inference (Cremer et al., 2018; Zhang et al., 2018) with an auxiliary model that obtains the approximation $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, using an encoder-decoder architecture.

Their objective is the Evidence Lower Bound (ELBO),

$$\mathcal{L}(\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}|\boldsymbol{z}\right)\right] - D_{KL}(q_{\boldsymbol{\phi}}\left(\boldsymbol{z}|\boldsymbol{x}\right)||p(\boldsymbol{z})), \quad (1)$$

which encourages proper data reconstruction via the first term, whilst minimizing mismatch between posterior and prior via the second term.

When complex data spaces are to be encoded in the latent space, flexible priors are required to avoid a significant mismatch between the aggregated posterior and the prior, typically referred to as the *prior hole problem* (Rezende & Viola, 2018). In previous works, this issue has been alleviated by various strategies, including using multimodal priors mimicking the aggregated posterior (VampPrior) (Tomczak & Welling, 2018), or training flow-based (Rezende & Mohamed, 2015; Kingma et al., 2016; Papamakarios et al., 2021; Gatopoulos & Tomczak, 2021), autoregressive (Chen et al., 2017) or hierarchical priors (Klushyn et al., 2019; Maaløe et al., 2019; Peis et al., 2022; Zeng et al., 2022). However, these methods are all tailored for structured data under grid representations, whereas our proposed model is specifically designed for unstructured data.

Implicit Neural Representations (INRs) . INRs represent a powerful approach for parameterizing non-linear continuous functions that map coordinates to data using deep neural networks. This allows for efficient and independent querying of continuous locations, which is useful for various tasks like learning, graphics, vision, and graphs (Stanley, 2007; Ha, 2016; Mescheder et al., 2019; Genova et al., 2019; 2020; Chen & Zhang, 2019; Zeng et al., 2022; Sitzmann et al., 2019; Jiang et al., 2020; Mildenhall et al., 2021; Sitzmann et al., 2020; Grattarola & Vandergheynst, 2022; Dupont et al., 2022b;a).

Earlier versions of INRs struggled to capture high-frequency details but advancements have addressed this issue through improved input encoding (Tancik et al., 2020), activation functions, and network architectures. Recently, Hao et al. (2022) proposes a novel INR Levels-of-Experts (LoE) model that generalizes INRs based on MLPs with position-dependent weights, greatly increasing the model capacity. On the downside, the complexity of coordinate-dependent generation functions rules out the use of hypernetwork-based approaches.

HyperNetworks. HyperNetworks (Ha et al., 2017) are a powerful class of neural networks that generate the weights for a principal network. Recently, Nguyen et al. (2021) combined hypernetworks with VAEs in order to improve performance and generalization when modeling different tasks concurrently. In this approach, hypernetworks are used to generate the parameters of the approximate posterior ($\boldsymbol{\phi}$) and the likelihood ($\boldsymbol{\theta}$). In contrast, our proposed approach draws inspiration from the generator of GASP (Dupont et al., 2022b) and utilizes the hypernetwork to output the parameters of our data generator.

Deep Generative Models for INRs. Recently, several works have proposed to use INRs for learning distributions of functions, rather than distributions of data directly. Dupont et al. (2022a) propose to disentangle the task of learning functions by first learning the named *functas*, or modulation vectors that configure an INR for each datapoint. SIREN (Sitzmann et al., 2020) is used as the base INR network. In the second stage, any deep generative model can be trained on the learned *functaset*. Then, for conditional generation tasks, computing the modulation vector for a test point requires solving a numerical optimization problem. In (Rodríguez-Santana et al., 2022), the INR generator is constructed by combining a Bayesian NN with Gaussian weight priors that takes as input both the coordinates and a sample from a latent Gaussian noise distribution. α-divergence Variational Inference is developed by jointly approximating the function at different coordinates using a Gaussian Process. Inducing points are introduced to make the model scalable. While the performance of the method is remarkable in a small-to-moderate dimension and it provides posterior inference, scaling it to model high-dimensional objects such as images is certainly not trivial.

In GASP (Dupont et al., 2022b), a generator of functions is built by transforming samples from a standard Gaussian latent variable to sets of weights using a hypernetwork. They train this generator using a GAN-style approach jointly with a PointConv-based discriminator that tries to discern fake from real samples. The LoE model in (Hao et al., 2022) is also reformulated in a generative way by taking as input both the coordinate vector and a Gaussian latent noise sample and it is also adversarially trained. Therefore, we can state that across Functa, GASP and LoE, inference over test data to perform conditional generation, for instance, image completion or super-resolution, is not trivial as they all require

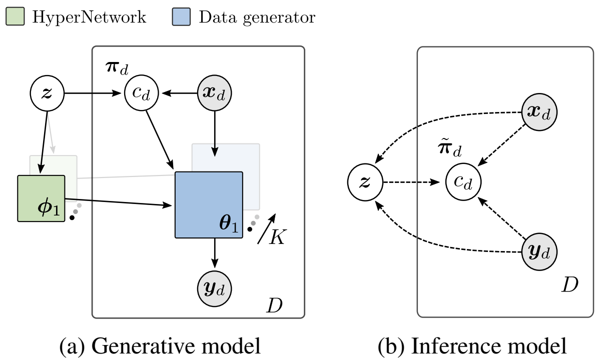


Figure 1. The VAMoH generative (a) and inference (b) model.

numerical optimization to find the latent codes. In VaMoH, we use a similar generator approach as in GASP, but we generalize it by using a mixture of hypergenerators combined with a flexible latent space constructed using normalizing-flows (Rezende & Mohamed, 2015). Additionally, VaMoH can be robustly trained using stochastic variational inference and do not require any extra optimization to perform inference on unseen data. The proposed method is a significant advancement in the field of INRs and deep generative models, and we demonstrate it has the potential to achieve state-of-the-art results in various tasks.

3. Variational Mixture of HyperGenerators

In this section, we introduce the **Va**riational **M**ixture **o**f **H**yperGenerators (VaMoH) model. VaMoH seamlessly integrates the capabilities of Variational Autoencoders (VAEs), mixtures of generative models, and hypernetworks to handle continuous domain data points effectively. Additionally, by incorporating normalizing flows (Rezende & Mohamed, 2015) as an expressive prior and utilizing Implicit Neural Representations (INRs) and Point Cloud encoders (Wu et al., 2019), VaMoH achieves superior performance and interpretability in a variety of both sample generation and inference tasks, such as in-painting and out-painting. The proposed generative model is depicted in Figure 1a.

Notation. We denote $[D] = \{1 \dots D\}$ to the set of positive integers from 1 to D. Let $(\boldsymbol{X}^{(i)}, \boldsymbol{Y}^{(i)})$, $i \in [N]$ be a set of N data samples (e.g., images). The i-th sample comprises a point cloud of D_i coordinate vectors, $\boldsymbol{X}^{(i)} \doteq \{\boldsymbol{x}_j^{(i)}\}_{j=1}^{D_i}$, and the set of corresponding feature vectors $\boldsymbol{Y}^{(i)} \doteq \{\boldsymbol{y}_j^{(i)}\}_{j=1}^{D_i}$. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the space of coordinate and feature vectors, respectively. As an example, in the context of image analysis, D_i represents the number of pixels in image i, and $(\boldsymbol{X}^{(i)}, \boldsymbol{Y}^{(i)})$ correspond to the set of $\mathbb{R}^2$ coordinates and values (e.g. RGB values) of the pixels, respectively.

3.1. Mixture of HyperGenerators

VaMoH generates a feature set $\boldsymbol{Y}$ given a set of corresponding coordinates $\boldsymbol{X}$. For simplicity, let us assume that $(\boldsymbol{X}, \boldsymbol{Y})$ is an image with D pixels. To generate such an image, first a continuous latent variable $\boldsymbol{z}$ is sampled from a prior distribution $p_{\boldsymbol{\psi}_z}(\boldsymbol{z})$ parameterized by $\boldsymbol{\psi}_z$. The resulting vector $\boldsymbol{z}$ acts as the input to K different *hypergenerators*. Here, we refer as a hypergenerator to both an MLP-based hypernetwork $g_{\phi_k}(\boldsymbol{z})$, with input $\boldsymbol{z}$ that outputs a set of parameters $\boldsymbol{\theta}_k = g_{\phi_k}(\boldsymbol{z})$; and, a data generator, f_{θ_k}, parametrized by the output of the hypernetwork. Thus, both $\boldsymbol{z}$ and $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K$ encode the information shared among the D coordinates (e.g., pixel location) in the data (e.g., pixel values) generation process.

In order for the resulting model to be expressive and interpretable, we assume that *each pixel is sampled from a mixture of K hypergenerators*. Thus, for each pixel $d \in [D]$, we introduce a latent categorical variable $c_d \in [K]$ in order to select the hypergenerator responsible of the pixel distribution, such that

$$p_{\psi_c}(\boldsymbol{C}|\boldsymbol{X}, \boldsymbol{z}) = \prod_{d=1}^{D} \prod_{k=1}^{K} \pi_{dk}^{[\![c_d = k]\!]}, \tag{2}$$

where $[\![c_d = k]\!]$ is the indicator function. The probability mass function (pmf) of c_d is also parameterized using a neural network f_{ψ_c} with parameters ψ_c that takes both $\boldsymbol{x}_d$ and $\boldsymbol{z}$ as input, and outputs $\boldsymbol{\pi}_d = f_{\psi_c}(\boldsymbol{x}_d, \boldsymbol{z})$ using a softmax function, where $\pi_{dk} = P(c_d = k|\boldsymbol{x}_d, \boldsymbol{z})$.

In summary, the overall generative process is given by:

$$p(\boldsymbol{Y}, \boldsymbol{C}, \boldsymbol{z}|\boldsymbol{X}) = p_{\psi_z}(\boldsymbol{z}) \prod_{d=1}^{D} \prod_{k=1}^{K} \pi_{dk}^{[\![c_d = k]\!]} p_{\boldsymbol{\theta_k}}(\boldsymbol{y}_d|\boldsymbol{x}_d), \tag{3}$$

where, importantly, we rely on Random Fourier Features (RFF) (Tancik et al., 2020) to encode $\boldsymbol{x}_d$ to capture high frequency details with our generative functions $f_{\boldsymbol{\theta}}$, as in Dupont et al. (2022b).

Flow-based prior on $\boldsymbol{z}$. The use of a fixed prior distribution for generating new unconditional samples in VAEs often results in the well-documented *prior hole problem* (Rezende & Viola, 2018), illustrated in Figure 2. This limitation stems from the poor expressiveness of a fixed prior distribution as compared to the approximate posterior. To tackle this issue, we rely on normalizing flows (NF) to learn a prior distribution of the form $p_{\psi_z}(\boldsymbol{z})$ with parameters ψ_z. This improves the prior expressiveness and, thus, addresses the aforementioned problem. Specifically, VaMoH integrates T layers of a planar flow (Rezende & Mohamed, 2015) with a Gaussian $p_0(\boldsymbol{z})$ as base distribution, resulting in a flexible prior distribution that significantly enhances the

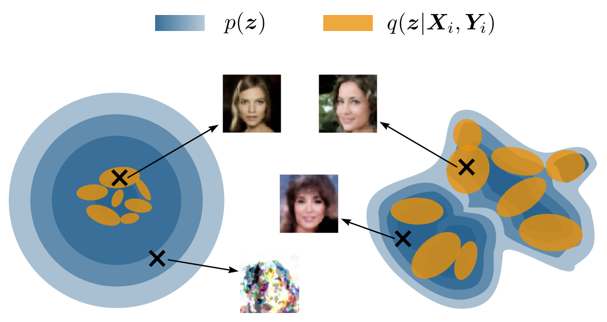


Figure 2. Illustration of the *prior hole problem*. Blue contours are the prior $p(\boldsymbol{z})$, whilst orange contours are approximate posteriors $q(\boldsymbol{z}|\boldsymbol{X}_i, \boldsymbol{Y}_i)$. Left: simple standard prior does not accurately cover the encoder complexity. Decoding samples from the prior that fall far from the aggregated posterior from training data gives unrealistic images. Right: a more flexible prior properly matches the complexity of the encoder, leading to better quality of the images generated from the prior.

generated samples. Extended empirical analysis is provided in Appendix A.5.

3.2. Inference model

In this subsection, we present the inference model that we propose to approximate the posterior of the latent variables $\boldsymbol{z}$ and $\boldsymbol{C}$. The model is defined as follows:

$$q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X}) = q_{\boldsymbol{\gamma}_z}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X}) \prod_{d=1}^{D} q_{\boldsymbol{\gamma}_c}(c_d|\boldsymbol{z}, \boldsymbol{y}_d, \boldsymbol{x}_d). \tag{4}$$

We illustrate the inference model in Figure 1b. Throughout the following, we refer to all the parameters of the inference model as $\boldsymbol{\gamma} = \{\gamma_z, \gamma_c\}$.

Inference for the continues latent variable. We propose to model the posterior distribution of $\boldsymbol{z}$ as $q_{\boldsymbol{\gamma}_z}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X}) = \mathcal{N}\left(\boldsymbol{z}|f_{\boldsymbol{\gamma}_z}(\boldsymbol{Y}, \boldsymbol{X})\right)$, parameterized by γ_z. It's important to note that this distribution is shared among the complete sample (e.g., image), thus $\boldsymbol{z}$ contains global information. To parametrize $f_{\boldsymbol{\gamma}_z}$, we use a PointConv (Wu et al., 2019). This entails several advantages. Firstly, it generalizes convolutional operations to continuous space coordinate systems, in contrast to the fixed grids used in CNNs (LeCun et al., 1995). Secondly, it is independent of data resolution, geometry of the grid, and missingness of the data. Of particular interest is the latter, as grid-based architectures typically require missing dimensions to be filled (e.g., with zeros), which introduces bias to the model (Simkus et al., 2021).

Inference of the mixture components. We model the posterior over the categorical latent variables c_d of each coordinate d as

$$q_{\boldsymbol{\gamma}_c}(c_d|\cdot) = \mathrm{Cat}\left(c_d|f_{\boldsymbol{\gamma}_c}(\boldsymbol{z}, \boldsymbol{y}_d, \boldsymbol{x}_d)\right) = \prod_{k=1}^{K} \tilde{\pi}_{dk}^{[\![c_d=k]\!]}, \tag{5}$$

where we parametrize $\boldsymbol{\gamma}_c$ with a simple MLP. It is important to notice that this posterior depends on the local information of the coordinate, i.e., $(\boldsymbol{x}_d, \boldsymbol{y}_d)$, and only on the global information through $\boldsymbol{z}$. This choice is made to encourage the model to learn to use different hypergenerators for different parts of the data.

3.3. Training

The evidence lower bound (ELBO) of our proposed model is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}) &= \mathbb{E}_{q_{\boldsymbol{\gamma}}}\left[\log p_{\boldsymbol{\theta}}\left(\boldsymbol{Y}|\boldsymbol{z}, \boldsymbol{X}, \boldsymbol{C}\right)\right] \\ &\quad - D_{KL}(q_{\boldsymbol{\gamma}_z}(\boldsymbol{z}|\boldsymbol{X}, \boldsymbol{Y})||p_{\psi_z}(\boldsymbol{z})) \\ &\quad - \mathbb{E}_{q_{\boldsymbol{\gamma}_z}}\left[D_{KL}(q_{\boldsymbol{\gamma}_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{X}, \boldsymbol{Y})||p_{\psi_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{X}))\right].\end{aligned} \tag{6}$$

We provide the complete derivation in Appendix A.2. We train VaMoH maximizing this ELBO by stochastic gradient descent on randomly selected mini-batches. To stabilize the initialization of the normalizing flow and accelerate convergence, we employ a standard prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ during the initial epochs of the training to allow the model to focus on organizing the approximate posterior and producing accurate reconstructions. After some iterations, we start training the Planar Flow $p_{\psi_z}(\boldsymbol{z})$. While KL between discrete distributions in Equation (6) is easily computed in closed form, the continuous KL is approximated by Monte Carlo sampling, just as we do for the reconstruction term.

Point dropout. The design of VAMoH allows for easily handling point clouds $\{\boldsymbol{X}, \boldsymbol{Y}\}$ of arbitrary sizes. The PointConv encoder is able to convolve the observed points, regardless of their coordinates, and map them into the approximate posterior, which can then be decoded to generate new points at any desired location. To enhance the conditional generation capabilities and robustness of VAMoH in inferring information from partial data, we apply dropout to the points within a set $\{\boldsymbol{X}, \boldsymbol{Y}\}$ according to a probability $p \sim U(0, \alpha)$, which is sampled independently for each batch. The maximum dropout probability, α, is fixed for ensuring that the reduced set contains at least as many points as centroids to be found at the first layer of PointConv. In previous VAE-based models (Ma et al., 2020; Peis et al., 2022), this strategy has been successfully employed for masking training batches in order to improve missing data imputation tasks. Nevertheless, since these models deal

Table 1. Comparison of FID and Precision and Recall scores of image generation for VaMoH, GASP, and Functa. Low FID, high precision, and high recall indicate the best performance. The best results are highlighted in bold. Note that for Functa and CELEBA HQ we just report their FID value, since they do not report the precision and recall.

	CELEBA HQ			SHAPES3D		
Model	↓ FID	↑ Precision	↑ Recall	↓ FID	↑ Precision	↑ Recall
GASP (Dupont et al., 2022b)	**14.01 ± 0.18**	**0.81 ± 0.0**	**0.43 ± 0.01**	118.66 ± 0.64	0.01 ± 0.0	0.16 ± 0.01
Functa (Dupont et al., 2022a)	40.40	-	-	57.81 ± 0.15	0.06 ± 0.0	0.13 ± 0.0
VaMoH	66.27 ± 0.18	0.65 ± 0.0	0.0 ± 0.0	**56.25 ± 0.57**	**0.08 ± 0.0**	**0.64 ± 0.01**

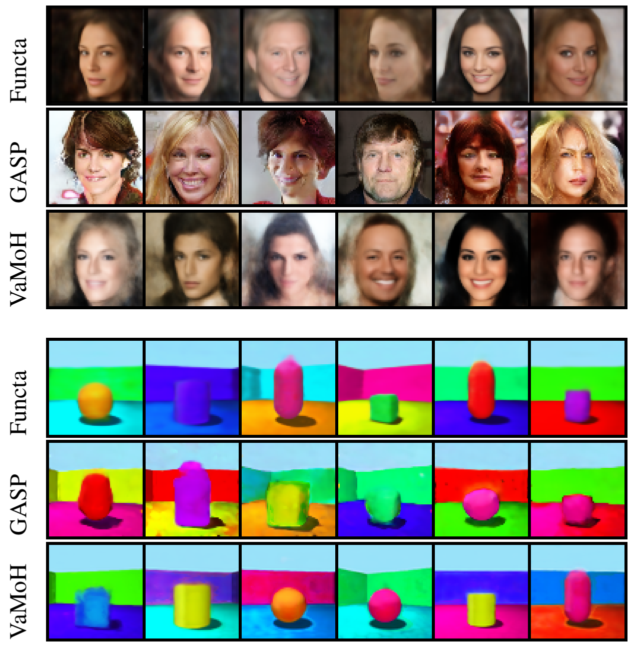


Figure 3. Comparison of generation quality at original resolution. CELEBA HQ samples of Functa obtained from (Dupont et al., 2022a).

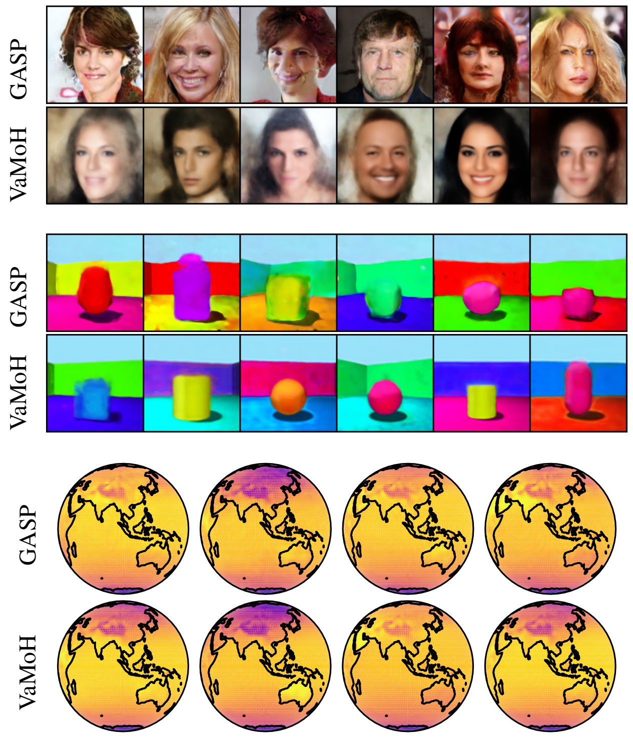


Figure 4. Comparison of uncurated generated samples at super-resolution.

with grid-type incomplete data, some pre-imputation is required before feeding the encoder, mean imputation or zero filling being typical choices, with the cost of introducing bias in the model (Simkus et al., 2021). In contrast, in our work, the PointConv encoder easily handles missing data, without requiring any pre-imputation strategy. Additional empirical evaluations can be found in the Appendix A.4.

4. Experiments

In this section, we provide a thorough empirical evaluation of VaMoH. We evaluate our model on the tasks of data generation, reconstruction, and imputation, including the superresolution results.

Baselines. We compare VaMoH with GASP (Dupont et al., 2022b) and Functa (Dupont et al., 2022a) with a Normalizing Flow as generator of modulation vectors, which we denote simply by Functa. More specifically, GASP is only considered for data generation, since it does not allow for any inference-related task.

Datasets. We evaluate VaMoH on POLYMNIST (28×28), CELEBA HQ (64×64) (Karras et al., 2017), SHAPES3D (64×64) (Burgess & Kim, 2018), climate data from the ERA5 dataset (Hersbach et al., 2019), and 3D chair voxels from the SHAPENET dataset (Chang et al., 2015). The architecture and hyperparameters used for each dataset are detailed in Appendix B.1.

We implemented VaMoH in PyTorch and performed all experiments on a single V100 with 32GB of RAM. The code with the model implementation and experiments is available at `https://github.com/bkoyuncu/vamoh`. In this section, we provide a glimpse into the results of our experiments on all datasets. Additional experiments and their outcomes can be found at Appendix B.

4.1. Generation

In this section, we evaluate VaMoH for the task of synthetic data generation over continuous coordinate systems. Samples are obtained from the learned models in both the original and twice the resolution, referred to as *super resolution*.

Metrics. We use two metrics to evaluate the quality of the generated samples. We report the usually employed Fréchet Inception Distance (FID) (Heusel et al., 2017) as well as the *improved precision and recall* (Kynkäänniemi et al., 2019), which measures the quality of the generated data (i.e, high precision) and the coverage of the true data distribtuion (i..e, high recall).

Results. We present the summary of the quantitative comparison of VaMoH with both baselines for two image datasets in Table 1. Our analysis indicates that, quantitatively speaking, there is not a clear winner among the models. While GASP achieves the highest performance on the CELEBA HQ dataset, visual inspection of the generated samples in Figure 3 reveals that the quality of Functa and VaMoH is also high. This aligns with previous studies (Dupont et al., 2022a) that have noted that FID may over-penalize blurriness. Additionally, it is worth noting that VaMoH obtains a recall of 0.0 on this dataset, despite the generated samples displaying diversity in features such as facial expressions and hairstyles, as seen in Figure 3. In analyzing the performance of VaMoH on the SHAPES3D dataset, we observe a clear superiority compared to other models. As depicted in Figure 3, VaMoH is able to generate objects with diverse shapes and colors, as well as walls and floors delimited with sharp edges. Furthermore, it achieves the highest quantitative metrics, as demonstrated in the right columns of Table 1. Despite this, it is important to note that the low precision values obtained by all models require a comprehensive evaluation approach, incorporating both quantitative and qualitative metrics when assessing generative models.

Finally, in Figure 4, we evaluate the super-resolution capabilities of the models by visually inspecting the generated samples at twice the original resolution. We utilize the same latent code as in the generated samples in Figure 3 for the CELEBA HQ and SHAPES3D datasets. Additionally, to demonstrate the versatility of our VaMoH to model diverse types of data, we also include super resolution samples from the ERA5 dataset. Comparison with super-resolution samples generated by Functa, and additional results for all the datasets can be found in Appendix B.2.

Discussion. A significant advantage of VaMoH is its ability to achieve the capability of generation through a single optimization procedure. In contrast, GASP requires solving the min-max GAN optimization, which has been acknowledged to be unstable (Jabbar et al., 2020); and Functa requires first learning the SIREN model and modulations for each sample, followed by the training of an additional generative model, such as a normalizing flow.

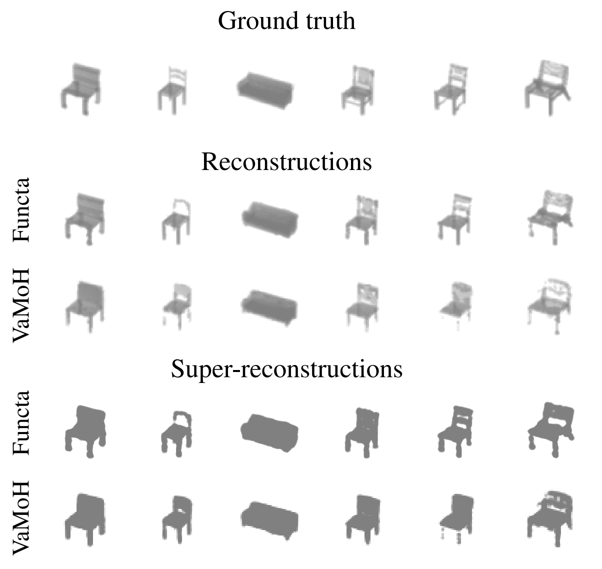
Overall, these results demonstrate that the generation quality of VaMoH is comparable to existing alternatives. In the following, we show that VaMoH exhibits superior efficiency and performance in inference tasks in comparison with Functa. It is worth noting that we do not compare VaMoH with GASP, as it is purely a generative model.

4.2. Reconstruction

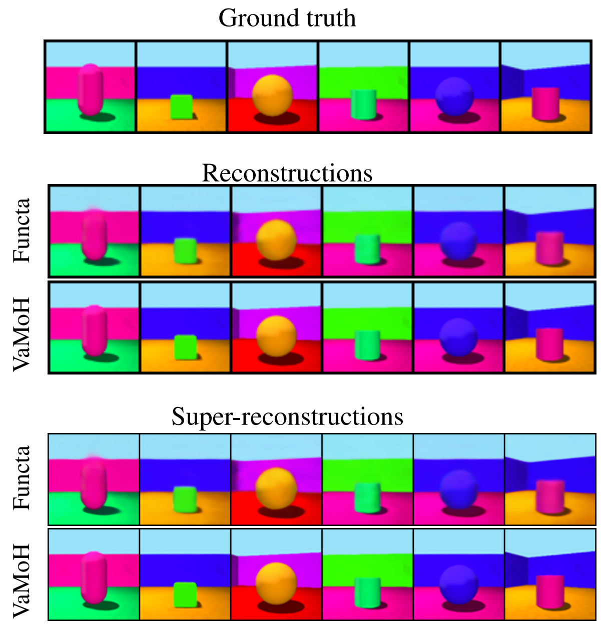
In this section, we evaluate the performance of VaMoH in reconstructing data and compare it with Functa. As before, we conduct the assessment at both the original resolution and at double the resolution, which we refer to as *super-reconstruction*. This latter scenario involves generating features at new coordinate positions.

Metrics. We use the Peak Signal-to-Noise-Ratio (PSNR) to quantify the quality of reconstructions (See Appendix B.3 for further details). We also compare inference times (in seconds) of VaMoH and Functa in Tables 2 and 5.

Results. Figure 5 shows reconstructions and super-reconstructions for the first 6 samples of the test set of SHAPENET and SHAPES3D for Functa and VaMoH. We observe that both methods produce high-quality reconstructions. When evaluating the SHAPENET dataset (as shown in Figure 5a), we observe that both models are capable of capturing the details of the chairs, such as the various patterns present on the back of the chairs. Notably, VaMoH achieves more detailed edges in both reconstructions and super-reconstructions for the SHAPES3D dataset (see Figure 5b) as well as overall less blurriness. A more quantitative assessment is provided in Figure 6, which compares PSNR values obtained for all the test set samples of the image-like datasets. We observe that VaMoH achieves similar quality on POLYMNIST and CELEBA HQ, and outperforms Functa on ERA5 and SHAPES3D. This is quite remarkable as Functa requires solving an optimization problem per sample, and thus it 'overfits' each of the samples. In stark contrast, VaMoH efficiently generates reconstructions with a single forward pass. As a consequence, VaMoH is



(a) SHAPENET



(b) SHAPES3D

Figure 5. Comparison of reconstruction quality of VaMoH and Functa on ground truth data from the first samples of the test set at original and super-resolution.

Table 2. Comparison of inference time (seconds) for reconstruction task of VaMoH and Functa. On the right-most two columns, we show the speed improvement of VaMoH compared to Functa (3) which is trained with 3 gradient steps as suggested in the original paper (Dupont et al., 2022a) and Functa (10) which is trained with 10 gradient steps to obtain the results of Functa depicted in Figures 16,17. Please note that these experiments are run on the same GPU device.

	Model Inference Time (secs)			Speed Improvement	
Dataset	VaMoH	Functa (3)	Functa (10)	vs. Functa (3)	vs. Functa (10)
POLYMNIST	**0.00453**	0.01648	0.05108	**x 3.64**	**x 11.28**
SHAPES3D	**0.00536**	0.01759	0.05480	**x 3.28**	**x 10.22**
CELEBA HQ	**0.00757**	0.01733	0.05381	**x 2.29**	**x 7.11**
ERA5	**0.00745**	0.01899	0.05932	**x 2.55**	**x 7.96**
SHAPENET	**0.00689**	0.02095	0.06576	**x 3.04**	**x 9.54**

significantly faster. In Table 2, we show that VaMoH is more than 2 times faster than Functa when trained with 3 gradient steps (as stated in (Dupont et al., 2022a)); and at least 7 times faster when trained with 10 gradient steps. Finally, Figure 7 shows the ability of VaMoH to generate quality super-reconstructions on the CELEBA HQ and ERA5 datasets. Another inference time comparison for super-reconstruction can be found in Table 5 in Appendix B.3.

Discussion. Both VaMoH and Functa are able to reconstruct and super-reconstruct data with high quality with different data modalities (e.g., images and voxels). On top of this, VaMoH achieves comparable or better PSNR values for the test set, which indicates it may reconstruct more details. Additionally, in Table 2, we show that VaMoH is significantly faster than Functa for the reconstruction task. This happens because VaMoH requires a simple forward pass. In contrast, Functa requires finding a new modulation for each new sample, which is an optimization in itself; therefore, its inference time highly depends on the number of gradient steps needed, which does not only have a major impact on the quality of the results but also on its computational efficiency. Note that this computational burden applies to any inference-related task, e.g., image completion. Given the quantitative and qualitative assessment of the results of VaMoH, we argue that it might be preferable to rely on

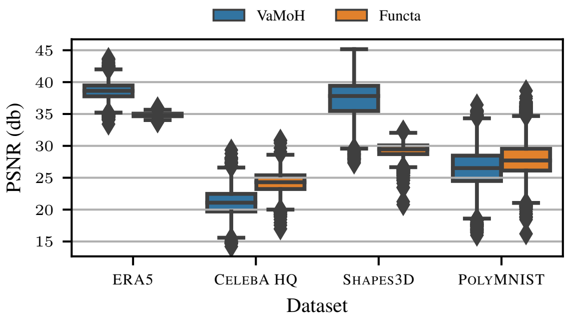


Figure 6. Comparison of PSNR (db) of reconstructed images in the test set.

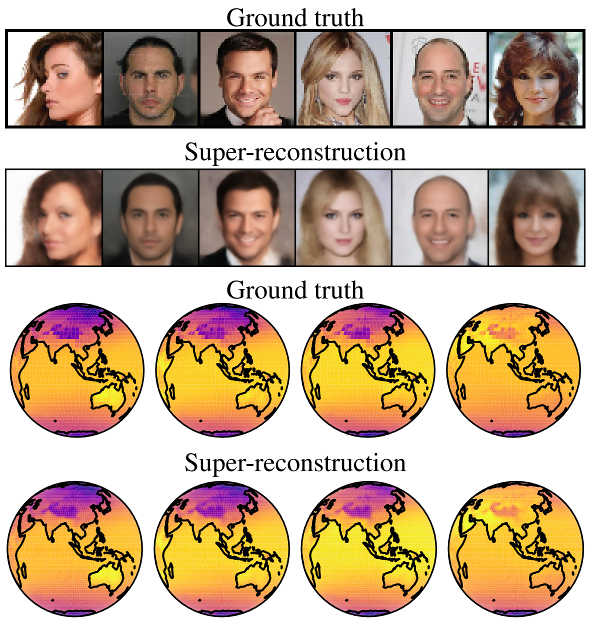
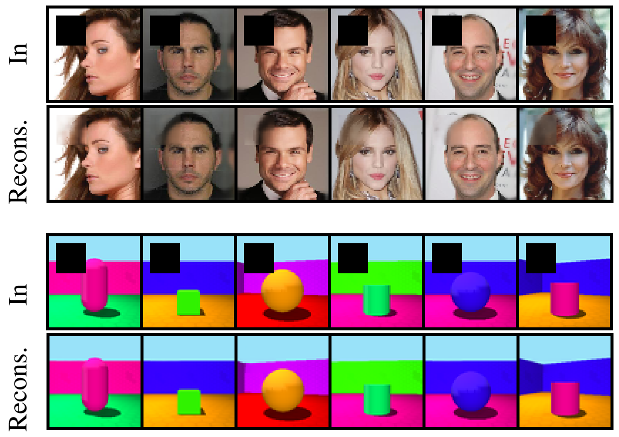


Figure 7. Ground truth for the first six test set samples and the corresponding super-reconstruction obtained by VaMoH for CELEBA HQ and ERA5 datasets.

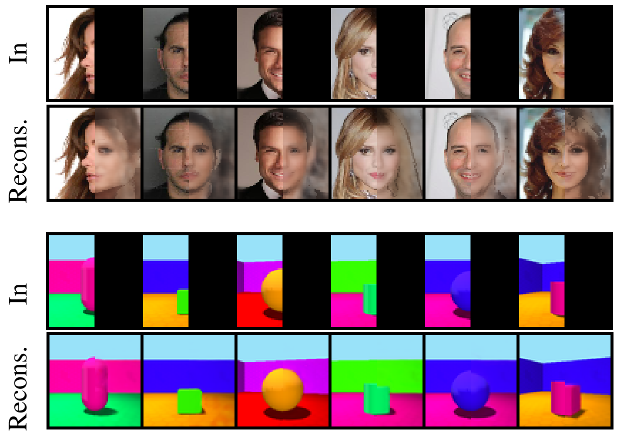
VaMoH due to its inference time efficiency in comparison to Functa. Thus, in the following, we keep understanding the capabilities of VaMoH without comparison.

4.3. Image completion

We evaluate the performance of VaMoH on the task of image completion. We consider two scenarios: (i) missing a patch (i.e., image in-painting) and (ii) missing half of the image, as a challenging case. Figure 8 illustrates the results on unseen test samples from CELEBA HQ and SHAPES3D. Our results demonstrate that VaMoH can reconstruct high-quality images, even when half of the image is missing, as in Figure 8b. Thanks to the use of PointConv (Wu et al., 2019) as the encoder, VaMoH can infer missing information without the need of inputting dummy values (e.g., zeros) for the features of the missing coordinates. This is something needed in standard VAEs, and allows VaMoH to mitigate any potential biases in the results. Complete results for all datasets evaluated, including those for image out-painting, can be found in Appendix B.4.



(a) Missing a patch



(b) Missing half of the image

Figure 8. Imputation of different amounts of missing parts using VaMoH. For each dataset, the top row shows the input with the missing part (In) and the bottom row shows the reconstructed image.

4.4. The flexibility of VAMoH

In this Section, we provide evidence of the interpretability of VaMoH. In particular, we show an example in which the mixture of hypergenerators, with $K = 4$, allows for splitting the generation of each pixel into several modes.

In Figure 9, we show the entropy (middle row) and map (bottom row) of the posterior distribution of the categorical latent variables $\boldsymbol{C}$. By examining the entropies per pixel, we observe low values for most pixels, but higher values for the borders. This indicates that the model is uncertain about which components to use at the borders, while it is using

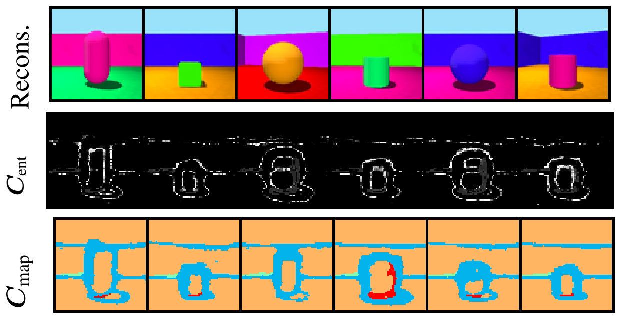


Figure 9. Reconstructed samples (top), entropy (middle) $\boldsymbol{C}_{\text{ent}} = \mathcal{H}(\boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})$, and map (bottom) $\boldsymbol{C}_{\text{map}} = \max_{c_d} q_{\gamma_c}(\boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})$ of the posterior distribution over the mixture.

fewer components for the rest of the image. Furthermore, the map values of $\boldsymbol{C}$ resemble a segmentation map. Here, we observe that one hypergenerator (orange) is specialized in filling the colors of different parts of the image, while the others (blue, red, and green) are used for the borders.

5. Conclusion

In this paper, we introduced VaMoH, a novel VAE-based model for learning distributions of functions that enables efficient and accurate generation of data over continuous coordinate systems. Notably, VaMoH allows for straightforward conditional generation with a simple forward pass on the model. VaMoH can perform tasks such as in-painting and out-painting or generate higher-resolution versions of a test set sample (that is, unseen during training) more efficiently than competing methods. Our experimental results demonstrate the effectiveness of VaMoH, both in generation and inference tasks, in a wide range of applications and datasets.

Although our model has a couple of limitations, they are not significant factors that would impede its effectiveness. Firstly, one limitation relates to the number of parameters in the hypernet, which scales linearly with the number of components (i.e., K). Secondly, we could improve the speed of the PointConv encoder with a sampling-based algorithm for the selection of centroid points used for the convolution step.

As a future research direction, we aim to enhance VaMoH by exploring methods for achieving disentanglement in the latent space. This would allow for controlled modifications of generated and reconstructed samples and may open up new possibilities for applications such as data editing at various resolutions. We have not identified any social concerns associated with this work. In fact, VaMoH reduces the time required for inference during testing, which could broaden the range of applications and areas where INR methods can be utilized.

6. Acknowledgements

Batuhan Koyuncu and Isabel Valera acknowledge the support by the German Federal Ministry of Education and Research (BMBF) through the Cell-o project. Pablo Sánchez Martín thanks the German Research Foundation through the Cluster of Excellence “Machine Learning – New Perspectives for Science”, EXC 2064/1, project number 390727645 for generous funding support. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Pablo Sánchez Martín. Pablo M. Olmos and Ignacio Peis acknowledge the support by the Spanish government MCIN/AEI/10.13039/501100011033/ FEDER, UE, under grant PID2021-123182OB-I00, by Comunidad de Madrid under grant IND2022/TIC-23550, and by Comunidad de Madrid and FEDER through IntCARE-CM. The work of Ignacio Peis has been also supported by the Spanish government (MIU) under grant FPU18/00516. We thank Adrián Javaloy and Jonas Klesen for their valuable comments and discussions.

References

Burgess, C. and Kim, H. 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=BysvGP5ee.

Chen, Z. and Zhang, H. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948, 2019.

Cremer, C., Li, X., and Duvenaud, D. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pp. 1078–1086. PMLR, 2018.

Dupont, E., Kim, H., Eslami, S. A., Rezende, D. J., and Rosenbaum, D. From data to functa: Your data point is a function and you can treat it like one. In *International Conference on Machine Learning*, pp. 5694–5725. PMLR, 2022a.

Dupont, E., Teh, Y. W., and Doucet, A. Generative models as distributions of functions. In *International Conference on Artificial Intelligence and Statistics*, pp. 2989–3015. PMLR, 2022b.

Gatopoulos, I. and Tomczak, J. M. Self-supervised variational auto-encoders. *Entropy*, 23(6):747, 2021.

Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W. T., and Funkhouser, T. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7154–7164, 2019.

Genova, K., Cole, F., Sud, A., Sarna, A., and Funkhouser, T. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4857–4866, 2020.

Grattarola, D. and Vandergheynst, P. Generalised implicit neural representations. *Advances in Neural Information Processing Systems*, 2022.

Ha, D. Generating large images from latent vectors. *blog.otoro.net*, 2016. URL https://blog.otoro.net/2016/04/01/generating-large-images-from-latent-vectors/.

Ha, D., Dai, A. M., and Le, Q. V. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Hao, Z., Mallya, A., Belongie, S., and Liu, M.-Y. Implicit neural representations with levels-of-experts. *Advances in Neural Information Processing Systems*, 2022.

Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Muñoz Sabater, J., Nicolas, J., Peubey, C., Radu, R., Rozum, I., et al. Era5 monthly averaged data on single levels from 1979 to present. *Copernicus Climate Change Service (C3S) Climate Data Store (CDS)*, 10:252–266, 2019.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jabbar, A., Li, X., and Omar, B. A survey on generative adversarial networks: Variants, applications, and training. *ACM Computing Surveys (CSUR)*, 54, 2020.

Jiang, C., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T., et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6001–6010, 2020.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

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

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

Klushyn, A., Chen, N., Kurle, R., Cseke, B., and van der Smagt, P. Learning hierarchical priors in vaes. *Advances in neural information processing systems*, 32, 2019.

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., and Aila, T. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

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

Loaiza-Ganem, G. and Cunningham, J. P. The continuous bernoulli: fixing a pervasive error in variational autoencoders. *Advances in Neural Information Processing Systems*, 32, 2019.

Ma, C., Tschiatschek, S., Turner, R., Hernández-Lobato, J. M., and Zhang, C. Vaem: a deep generative model for heterogeneous mixed type data. *Advances in Neural Information Processing Systems*, 33:11237–11247, 2020.

Maaløe, L., Fraccaro, M., Liévin, V., and Winther, O. Biva: A very deep hierarchy of latent variables for generative modeling. *Advances in neural information processing systems*, 32, 2019.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4460–4470, 2019.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Nguyen, P., Tran, T., Gupta, S., Rana, S., Dam, H.-C., and Venkatesh, S. Variational hyper-encoding networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 100–115. Springer, 2021.

Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57):1–64, 2021.

Peis, I., Ma, C., and Hernández-Lobato, J. M. Missing data imputation and acquisition with deep hierarchical models and hamiltonian monte carlo. In *Advances in Neural Information Processing Systems 35*, 2022.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015.

Rezende, D. J. and Viola, F. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

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

Rodríguez-Santana, S., Zaldivar, B., and Hernandez-Lobato, D. Function-space inference with sparse implicit processes. In *International Conference on Machine Learning*, pp. 18723–18740. PMLR, 2022.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Simkus, V., Rhodes, B., and Gutmann, M. U. Variational gibbs inference for statistical model estimation from incomplete data. *arXiv preprint arXiv:2111.13180*, 2021.

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

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

Stanley, K. O. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7537–7547. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/55053683268957697aa39fba6f231c68-Paper.pdf.

Tomczak, J. and Welling, M. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223. PMLR, 2018.

Wu, W., Qi, Z., and Fuxin, L. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630, 2019.

Zeng, X., Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., and Kreis, K. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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

A. Further details about VaMoH

A.1. HyperGenerator networks

As a reminder, we work with data samples which are point clouds $(\boldsymbol{X}, \boldsymbol{Y})$ with coordinate vectors $\boldsymbol{X}$ and feature vectors $\boldsymbol{Y}$. VaMoH aims to generate a feature set $\boldsymbol{Y}$ given a set of corresponding coordinates $\boldsymbol{X}$. If the data sample is an image with D pixels, $(\boldsymbol{X}, \boldsymbol{Y})$ correspond to the set of $\mathbb{R}^2$ coordinates and values (e.g. RGB values) of the pixels, respectively.

To generate such an image, first a continuous latent variable $\boldsymbol{z}$ is sampled from a prior distribution $p_{\boldsymbol{\psi}_z}(\boldsymbol{z})$ parameterized by $\boldsymbol{\psi}_z$. The resulting vector $\boldsymbol{z}$ can be thought as a global summary vector of the output image and acts as the input to K different *hypergenerators*. Here, we refer as a hypergenerator to both an MLP-based hypernetwork $g_{\phi_k}(\boldsymbol{z})$, with input $\boldsymbol{z}$ that outputs a set of parameters $\boldsymbol{\theta}_k = g_{\phi_k}(\boldsymbol{z})$; and, a data generator, f_{θ_k}, parametrized by the output of the hypernetwork as shown in Figure 1. In other words, each of the MLP-based hypernetwork outputs the set of parameters $\boldsymbol{\theta}_k = g_{\phi_k}(\boldsymbol{z})$ that parameterizes the corresponding data generator f_{θ_k} which operates over each input coordinate $\boldsymbol{x}_d \in \boldsymbol{X}$ to generate corresponding feature values $\boldsymbol{y}_d \in \boldsymbol{Y}$, i.e. $\boldsymbol{y}_d = f_{\theta_k}(\boldsymbol{x}_d)$ where $d \in [D]$.

The intuition for the usage of hypergenerators can be explained as follows: VaMoH uses a continous latent variable $\boldsymbol{z}$ as an input to a hypernetwork. This hypernetwork is used for parameterizing a data generator, i.e. an INR, that can model a data sample. In the inference step, VaMoH can infer the latent variable $\boldsymbol{z}$ using a PointConv based encoder. Therefore, VaMoH can perform conditional tasks by performing an inference step. Moreover, each of the conditional tasks requires only a single forward pass without any extra optimization step per samples. Lastly, K different hypergenerators increase the expresiveness of VaMoH since they act as separate INRs for modeling different patterns in the output.

A.2. ELBO derivation

Our objective is to maximize the evidence lower bound (ELBO) of a set of D features $\boldsymbol{Y}$ given the corresponding set of coordinates $\boldsymbol{X}$. To do so, we introduce a family of variational distributions for the continuous and discrete latent variables, namely $\boldsymbol{z}$ and c_d where $d \in [D]$:

$$q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X}) = q_{\boldsymbol{\gamma}_z}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X}) \prod_{d=1}^{D} q_{\boldsymbol{\gamma}_c}(c_d|\boldsymbol{z}, \boldsymbol{y}_d, \boldsymbol{x}_d). \tag{7}$$

where

$$q_{\boldsymbol{\gamma}_z}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X}) = \mathcal{N}\left(\boldsymbol{z}|f_{\boldsymbol{\gamma}_z}(\boldsymbol{Y}, \boldsymbol{X})\right) \quad \text{and} \quad q_{\boldsymbol{\gamma}_c}(c_d|\boldsymbol{z}, \boldsymbol{y}_d, \boldsymbol{x}_d) = \text{Cat}\left(c_d|f_{\boldsymbol{\gamma}_c}(\boldsymbol{z}, \boldsymbol{y}_d, \boldsymbol{x}_d)\right). \tag{8}$$

Note that these two distributions are parameterized by $\boldsymbol{\gamma}_z$ and $\boldsymbol{\gamma}_c$, which we aim to learn. We refer to the set of all inference parameters as $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}_z, \boldsymbol{\gamma}_c\}$. Moving on to the generative distribution, it factorizes as

$$p_{\psi,\phi}(\boldsymbol{Y}, \boldsymbol{C}, \boldsymbol{z}|\boldsymbol{X}) = p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{C}, \boldsymbol{z}) p_{\psi_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{X}) p_{\psi_z}(\boldsymbol{z}) \tag{9}$$

where $p_{\psi_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{X})$ is the prior over the categorical latent variable, $p_{\psi_z}(\boldsymbol{z})$ is the prior over the continuous latent variable (parameterized by a Normalizing Flow, see Appendix A.5), and $p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{C}, \boldsymbol{z})$ is the likelihood distribution over the set of features. More in detail, the likelihood factorizes over coordinates and we define each of them as a mixture:

$$p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{C}, \boldsymbol{z}) = \prod_{d=1}^{D} p_{\boldsymbol{\theta}}(\boldsymbol{y}_d|\boldsymbol{x}_d, c_d, \boldsymbol{z}), = \prod_{d=1}^{D} \prod_{k=1}^{K} \pi_{dk}^{[\![c_d=k]\!]} p_{\boldsymbol{\theta}_k}(\boldsymbol{y}_d|\boldsymbol{x}_d) \quad \text{where} \quad \boldsymbol{\theta}_k = g_{\phi_k}(\boldsymbol{z}). \tag{10}$$

At this point, it is important to remark that we do not learn the parameters $\boldsymbol{\theta}_k$ $k \in [K]$ of the likelihood directly. Instead, we learn k the parameters of k different hypernetworks g_{ϕ_k} $k \in [K]$ that output the corresponding likelihood parameters. Thus, the set of all the parameters of the generative models are $\boldsymbol{\phi} = \{\boldsymbol{\phi}_k\}_{k=1}^{K}$ and $\boldsymbol{\psi} = \{\psi_z, \psi_c\}$.

With this choice of the inference and generative model, we define the ELBO as

$$\mathcal{L}(\boldsymbol{Y}, \boldsymbol{X}; \psi, \phi, \boldsymbol{\gamma}) = \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})}\left[\log \frac{p_{\psi,\phi}(\boldsymbol{Y}, \boldsymbol{C}, \boldsymbol{z}|\boldsymbol{X})}{q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})}\right] \leq \log p(\boldsymbol{Y}|\boldsymbol{X}). \tag{11}$$

Using the factorization of the generative and inference model in Equations (8) and (10), we can split it in various terms:

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{Y}, \boldsymbol{X}; \psi, \phi, \boldsymbol{\gamma}) = & \ \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{C}, \boldsymbol{z})\right] \\
& + \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})} \left[\log \frac{p_{\psi_z}(\boldsymbol{z})}{q_{\boldsymbol{\gamma}_z}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X})}\right] \\
& + \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})} \left[\log \frac{p_{\psi_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{X})}{q_{\boldsymbol{\gamma}_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{Y}, \boldsymbol{X})}\right]
\end{aligned} \tag{12}
$$

We can marginalize some variables from the expectations.

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{Y}, \boldsymbol{X}; \psi, \phi, \boldsymbol{\gamma}) = & \ \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{C}, \boldsymbol{z})\right] \\
& + \mathbb{E}_{q_{\boldsymbol{\gamma}_z}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X})} \left[\log \frac{p_{\psi_z}(\boldsymbol{z})}{q_{\boldsymbol{\gamma}_z}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X})}\right] \\
& + \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})} \left[\log \frac{p_{\psi_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{X})}{q_{\boldsymbol{\gamma}_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{Y}, \boldsymbol{X})}\right]
\end{aligned} \tag{13}
$$

And rewriting the last two terms as KL divergences we get:

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{Y}, \boldsymbol{X}; \psi, \phi, \boldsymbol{\gamma}) = & \ \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{C}, \boldsymbol{z})\right] \\
& - D_{KL}(q_{\gamma_z}(\boldsymbol{z}|\boldsymbol{X}, \boldsymbol{Y}) || p_{\psi_z}(\boldsymbol{z})) \\
& - \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X})} \left[D_{KL}(q_{\gamma_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{X}, \boldsymbol{Y}) || p_{\psi_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{X}))\right].
\end{aligned} \tag{14}
$$

Finally, in the first term, we can marginalize over the categorical latent variable such that we only need to approximate by Monte Carlo expectation over $\boldsymbol{z}$:

$$
\begin{aligned}
\mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{C}, \boldsymbol{z})\right] &= \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{C}|\boldsymbol{Y}, \boldsymbol{X})} \left[\sum_{d=1}^{D} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_d|\boldsymbol{x}_d, \boldsymbol{z}, c_d)\right] \\
&= \sum_{d=1}^{D} \mathbb{E}_{q_{\boldsymbol{\gamma}_z}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X})} \left[\sum_{k=1}^{K} \log p_{\boldsymbol{\theta}_k}(\boldsymbol{y}_d|\boldsymbol{x}_d) \cdot \pi_{dk}\right]
\end{aligned} \tag{15}
$$

Then ELBO becomes

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{Y}, \boldsymbol{X}; \psi, \phi, \boldsymbol{\gamma}) = & \sum_{d=1}^{D} \mathbb{E}_{q_{\boldsymbol{\gamma}_z}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X})} \left[\sum_{k=1}^{K} \log p_{\boldsymbol{\theta}_k}(\boldsymbol{y}_d|\boldsymbol{x}_d) \cdot \pi_{dk}\right] \\
& - D_{KL}(q_{\gamma_z}(\boldsymbol{z}|\boldsymbol{X}, \boldsymbol{Y}) || p_{\psi_z}(\boldsymbol{z})) \\
& - \mathbb{E}_{q_{\boldsymbol{\gamma}}(\boldsymbol{z}|\boldsymbol{Y}, \boldsymbol{X})} \left[D_{KL}(q_{\gamma_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{X}, \boldsymbol{Y}) || p_{\psi_c}(\boldsymbol{C}|\boldsymbol{z}, \boldsymbol{X}))\right].
\end{aligned} \tag{16}
$$

A.3. Algorithm details

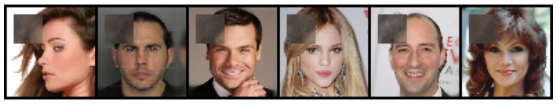
The training methodology is outlined in Algorithm 1. As discussed in Section 3.3, the training process begins with an initial warming stage, in which a standard prior $p(\boldsymbol{z})$ is utilized to stabilize the optimization of the encoder. The second and primary stage involves the introduction of the learnable prior $p_{\psi_z}(\boldsymbol{z})$.

Algorithm 1 Minibatch training of VaMoH

```
1: Define hyperparameters, estimator $\widetilde{\mathcal{L}}$ using Eq. (6)
2: $\gamma_{z,c}$ (inference), $\psi_{z,c}$ (prior), $\phi_{1:K}$ (hypernetwork) $\leftarrow$ initialize
3: **repeat**
4:    $\{\boldsymbol{X}, \boldsymbol{Y}\}_{1:M} \leftarrow$ random minibatch of $M$
5:    $\{\boldsymbol{X'}, \boldsymbol{Y'}\}_{1:M} \leftarrow$ apply dropout with probability $p \sim U(0, \alpha)$
6:    **if** NF is available **then**
7:       parameterize $p_\psi(\boldsymbol{z})$ with planar flow
8:    **else**
9:       fix $p(\boldsymbol{z})$ with standard Gaussian
10:   **end if**
11:   $\mathbf{g} \leftarrow \nabla_{\gamma,\psi,\phi_{1:K}} \widetilde{\mathcal{L}}^M(\gamma, \psi, \phi_{1:K}; \{\boldsymbol{X}, \boldsymbol{Y}\}_{1:M})$ (gradients)
12:   $\gamma, \psi, \phi_{1:K} \leftarrow$ Update parameters using gradients $\mathbf{g}$
13: **until** convergence of parameters $\gamma, \psi, \phi_{1:K}$
```

A.4. Point dropout for conditional generation

In this section, we present experimental evidence supporting the effectiveness of the point dropout strategy for training VAMoH, as discussed in Section 3.3 of the paper. Figure 10 illustrates the results of patch imputation on test images from the test set of CelebA HQ. When the model is trained on full images, as shown in Figure 10a, the PointConv encoder is unable to learn from a diverse set of centroids, resulting in uninformative posteriors when partial images are used during testing. However, as shown in Figure 10b, when VAMoH is trained on images that have undergone point dropout, as outlined in Section 3.3, improved robustness for conditional generation is obtained.


(a) Without point dropout

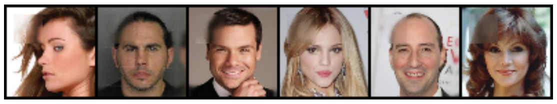
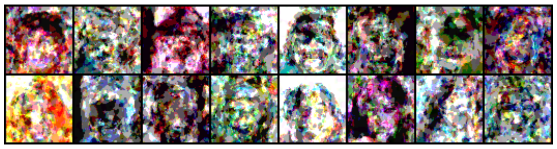

(b) With point dropout

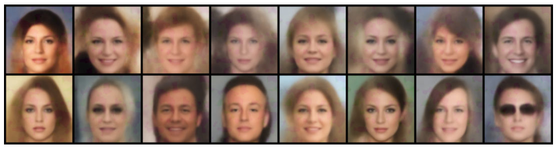
Figure 10. Imputation of missing patches in test CelebA HQ images using VAMoH. In (a), full images are fed to the model during training. In (b), points of training images are deleted with a dropout probability per batch.

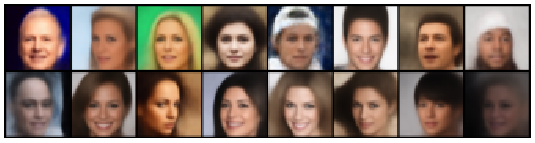
A.5. Flow-based prior

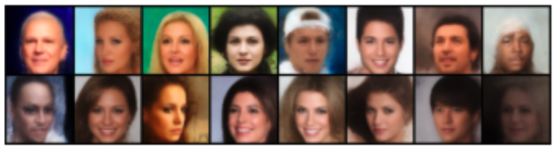
In this section, we evaluate the effectiveness of the Planar Flow as a prior for our continuous latent variable in addressing the *hole problem* described in Section 3.1. We first train our model on the CELEBA dataset (Liu et al., 2015), utilizing a fixed standard prior $p(\boldsymbol{z})$. Upon convergence, we sample from the learned prior and present the results in Figure 11a. The standard prior places significant probability mass in regions distant from the aggregated posterior, resulting in poor image quality when samples are drawn from these regions. By reducing the variance and approaching the posterior probability mass, as demonstrated in Figure 11b, we observe improved realism in the generated images. However, this also leads to a lack of diversity in the generated samples. In Figure 11c we include decoded samples from the posterior parameterized by the flexible encoder that reveals diversity for reconstructing images.

In contrast, by introducing the Planar Flow after a warming stage and training its parameters, we observe a more aligned reconstruction-generation process. This is evident by comparing the samples obtained by sampling from the posterior (Figure 11d) and the learned prior (Figure 11e), respectively.


(a) Standard prior, $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$


(b) Standard prior, $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, 0.01 \cdot \boldsymbol{I})$


(c) Standard prior, $\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{X}, \boldsymbol{Y})$


(d) Flow-based prior, $\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{X}, \boldsymbol{Y})$

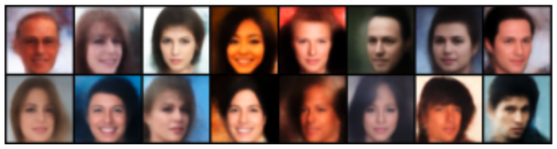

(e) Flow-based prior, $\boldsymbol{z} \sim p_{\psi_z}(\boldsymbol{z})$

Figure 11. Decoding samples from different latent distributions. Images in (a), (b), and (c) are generated/reconstructed by our model after being trained assuming a standard prior over $\boldsymbol{z}$. In (d) and (e) we include reconstructions and generations, respectively, when modeling a more flexible prior over $\boldsymbol{z}$ by training a Planar Flow.

A.6. Logistic likelihood

Inspired by (Kingma et al., 2016), we utilize a Discretized Logistic distribution for parameterizing the likelihood of discrete data with high number of categories, such as color channels in the RGB space for images. By doing that, we obtain a smooth and memory efficient predictive distribution for $\boldsymbol{y}_d$, as opposed to, for example, a Categorical likelihood with parameterized with 256-way softmax. The likelihood parameterized by each hypergenerator is given by

$$p_{\boldsymbol{\theta}_k}(\boldsymbol{y}_d|\boldsymbol{x}_d) = \sigma\left(\left(\boldsymbol{y}_d + 0.5 - \boldsymbol{\mu}_{dk}\right)/\boldsymbol{s}\right) - \sigma\left(\left(\boldsymbol{y}_d - 0.5 - \boldsymbol{\mu}_{dk}\right)/\boldsymbol{s}\right), \tag{17}$$

where the means $\boldsymbol{\mu}_{dk}$ are outputs of the hypergenerator, and the scales $\boldsymbol{s}$ are learnable parameters. Thanks to the design of VAMoH, our approach using a mixture for the likelihood is in line with (Salimans et al., 2017), and results in a Discretized Logistic Mixture Likelihood of the form

$$p(\boldsymbol{y}_d|\boldsymbol{x}_d) = \prod_{k=1}^{K} \pi_k \left[\sigma\left(\left(\boldsymbol{y}_d + 0.5 - \boldsymbol{\mu}_{dk}\right)/\boldsymbol{s}\right) - \sigma\left(\left(\boldsymbol{y}_d - 0.5 - \boldsymbol{\mu}_{dk}\right)/\boldsymbol{s}\right)\right], \tag{18}$$

which allows to accurately model the conditional distributions of the points by using a relatively small number of mixture components, as we show in our paper, and in concordance with (Salimans et al., 2017).

A.7. Effect of mixture components

Using a mixture of decoders which are parameterized by hypernetworks increases the expressiveness of our model as provide a mixture of INRs to map coordinates to features in the decoding step. Furthermore, it provides the flexibility of acquiring segmentation and entropy maps as shown in Figure 9. In addition, in Figure 12, we show the effect of not using a mixture (K=1), we have lower quality outputs. Moreover, if we increase too much the number of components (K=10), we achieve similar quality; however, we get over-segmented images where regions of interest are less informative.

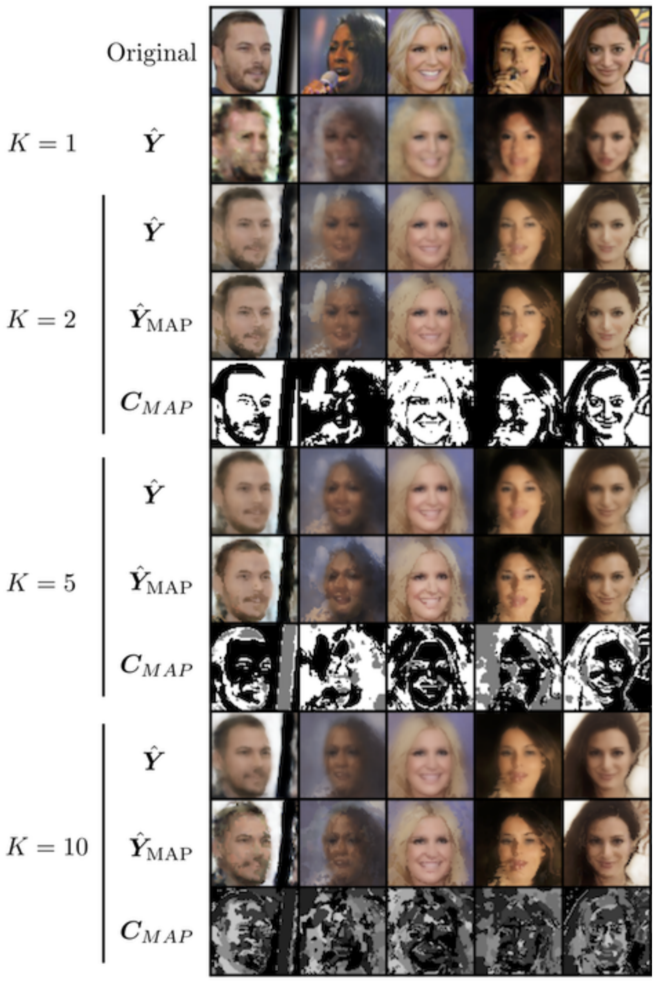


Figure 12. Ablation study on mixture of generators and effects of different number of mixture components in VaMoH for reconstruction task with CELEBA HQ dataset.

B. Experimental extension

In this section, we describe the experimental setup for VaMoH, as detailed in Appendix B.1. We also present a comprehensive set of results, including comparisons with the baselines Functa and GASP. Of note, we used the original code from GASP to replicate their results and also trained it on new datasets in our experimental setup. For Functa, we report results from the original paper (Dupont et al., 2022a) for the datasets shared with our study, and for the remaining datasets, we implemented and generated results independently with our own implementation.

B.1. Experimental setup

Implementation details for VaMoH are provided in Table 3. We base our choice of architecture on (Dupont et al., 2022b) as it provides a baseline setting. We train all of our model configurations with Adam optimizer. Also, in all of our models, we train Planar Flow and MLPs with activation of LeakyReLU with the exception of Sigmoid activation in the final layer of function generator. For datasets CelebA HQ, Shapes3D and PolyMNIST, Discretized Logistic likelihood (Kingma et al., 2016; Salimans et al., 2017) is utilized, whilst for ShapeNet and ERA5, Bernoulli and Continuous Bernoulli (Loaiza-Ganem & Cunningham, 2019) are employed, respectively.

Table 3. Implementation details of VaMoH.

		CelebA-HQ	Shapes3D	PolyMNIST	ShapeNet	ERA5
	dim_z	64	32	16	32	32
	K	10	5	4	3	3
	epochs	1000	600	600	500	600
	bs	64	64	256	22	64
	lr	1e-4	1e-4	1e-3	1e-3	1e-4
PointConv Encoder	h_weights	[16,16,16,16]	[16,16]	[16,16]	[16,16,16,16]	[16,16,16,16]
	neighbors	[9,9,9,9,9]	[16,16,16]	[9,9,9]	[8,27,27,27,8]	[9,9,9,9,9]
	centroids	[4096,1024,256,64,1]	[1024,256,64]	[196,49,25]	[4096,512,64,16,16]	[1024,256,64,32,16]
	out_channels	[64,128,256,512,512]	[32,64,256]	[32,32,32]	[32,64,128,256,16]	[32,64,256,512,32]
	avg_pooling_neighbors	[9,9,9,9,None]	None	None	None	None
	avg_pooling_centroids	[1024,256,64,16,None]	None	None	None	None
Categorical Encoder	layers	[64,32]	[32,32]	[32,32]	[32,32]	[32,32]
Hypernetwork	layers	[256,512]	[128,256]	[16,32]	[256,512]	[256,512]
Generator	layers	[64,64,64]	[32,32]	[4,4,4]	[64,64,64]	[64,64,64]
	RFF	m $= 128, \sigma = 2$	m $= 128, \sigma = 2$	m $= 128, \sigma = 2$	m $= 128, \sigma = 2$	m $= 128, \sigma = 2$
Flow	T	80	20	3	40	10

B.2. Generation

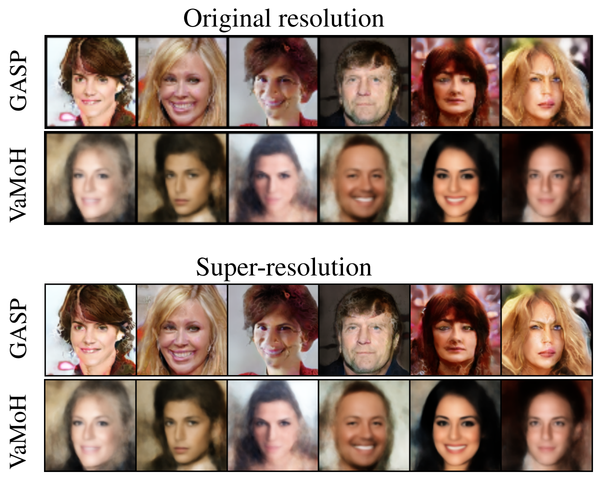
In this section, we present a comprehensive evaluation of the generation capabilities of VaMoH in comparison to Functa and GASP. Table 4 contains the obtained values for the FID, Precision, and Recall metrics for the image datasets. These values are presented as the mean and standard deviation computed over five independent sets of generated images. For each of the metrics, we report the values using real embeddings coming from the training set (tr) and the test set (tst), since we also want to evaluate whether any model is overfitting to the training set, or as wished, it is able to generalize.

Results for the CELEBA HQ and SHAPES3D datasets were previously discussed in Section 4.1. Additionally, we include in this Section results for the POLYMNIST dataset, where we observe that GASP obtains the best quantitative results, which might be related to its ability to generate sharp samples. Quantitative values for Functa and POLYMNIST are not reported as they were not used in the original paper and we found that normalizing flows failed to avoid overfitting to the modulations, even when trying different values of dropout, number of layers, and dimensionality of the hidden space. This led to low quality of the generations, as shown in 14b.

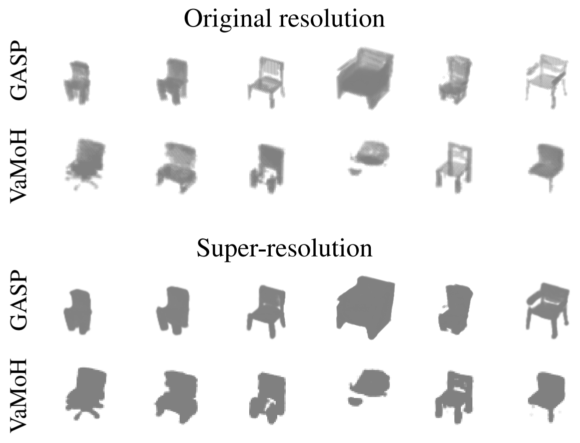
Visual inspection of the generated samples in Figure 13, Figure 14 illustrates that VaMoH generates high-quality samples, capturing details such as smiles in CELEBA HQ (Figure 13a), variety in chair legs in SHAPENET (Figure 13b), backgrounds in POLYMNIST (Figure 14b), as well as different configurations of temperatures (Figure 15). In Figure 13, Figure 14, we did not include generation samples of CELEBA HQ and SHAPENET for Functa since we were unable to reproduce the results reported in Dupont et al. (2022a). We refer the readers to Dupont et al. (2022a) for accessing the generation results for the corresponding datasets.

Table 4. Comparison of FID and Precision and Recall scores of image generation for VaMoH, GASP, and Functa. Low FID, high precision, and high recall indicate the best performance. The best results are highlighted in bold. Note that for Functa and CELEBA HQ we just report their FID value, since they do not report the precision and recall.

	CELEBA HQ			SHAPES3D		
Model	$\downarrow$ FID	$\uparrow$ Precision	$\uparrow$ Recall	$\downarrow$ FID	$\uparrow$ Precision	$\uparrow$ Recall
GASP(tst)	**17.8 $\pm$ 0.17**	**0.83 $\pm$ 0.0**	**0.42 $\pm$ 0.01**	119.35 $\pm$ 0.66	0.01 $\pm$ 0.0	0.16 $\pm$ 0.02
Functa(tst)	-	-	-	58.3 $\pm$ 0.14	0.07 $\pm$ 0.0	0.13 $\pm$ 0.0
VaMoH(tst)	72.14 $\pm$ 0.21	0.43 $\pm$ 0.01	0.0 $\pm$ 0.0	**56.63 $\pm$ 0.58**	0.09 $\pm$ 0.01	0.63 $\pm$ 0.02
GASP(tr)	**14.01 $\pm$ 0.18**	**0.81 $\pm$ 0.0**	**0.43 $\pm$ 0.01**	118.66 $\pm$ 0.64	0.01 $\pm$ 0.0	0.16 $\pm$ 0.01
Functa(tr)	40.40	-	-	57.81 $\pm$ 0.15	0.06 $\pm$ 0.0	0.13 $\pm$ 0.0
VaMoH(tr)	66.27 $\pm$ 0.18	0.65 $\pm$ 0.0	0.0 $\pm$ 0.0	**56.25 $\pm$ 0.57**	**0.08 $\pm$ 0.0**	**0.64 $\pm$ 0.01**

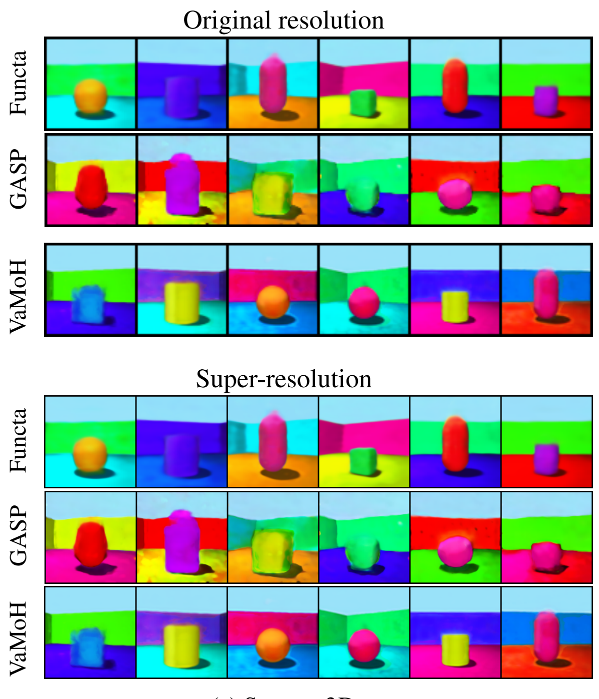


(a) CELEBA HQ

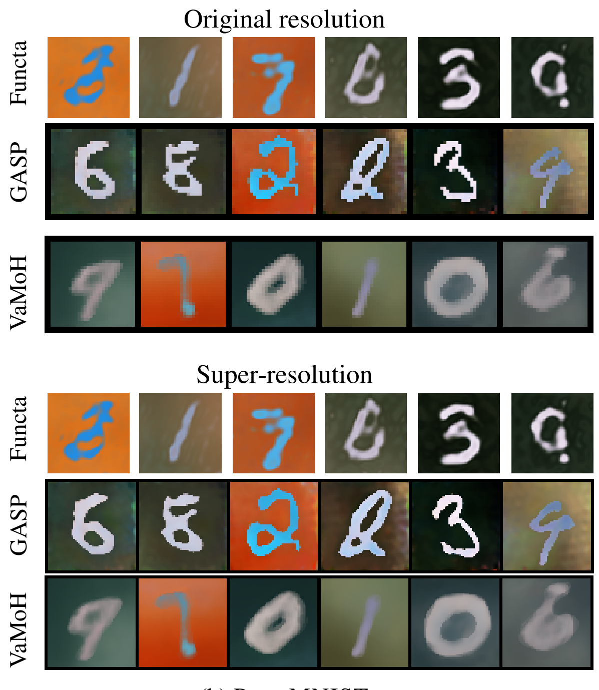


(b) SHAPENET

Figure 13. Comparison of generation quality of VaMoH and GASP at original resolution (top) and x2 resolution (bottom).

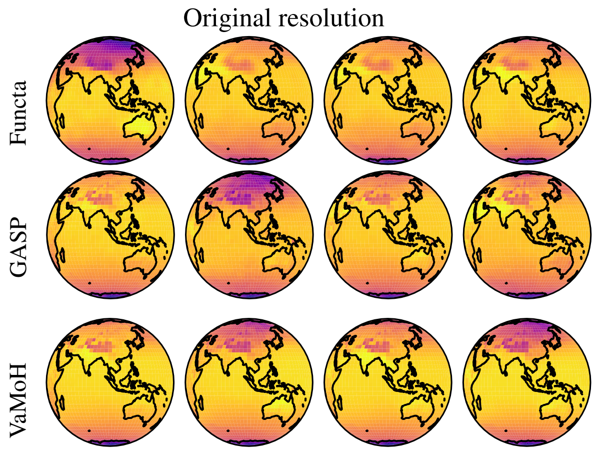


(a) SHAPES3D



(b) POLYMNIST

Figure 14. Comparison of generation quality of VaMoH, Functa, and GASP at original resolution (top) and x2 resolution (bottom).



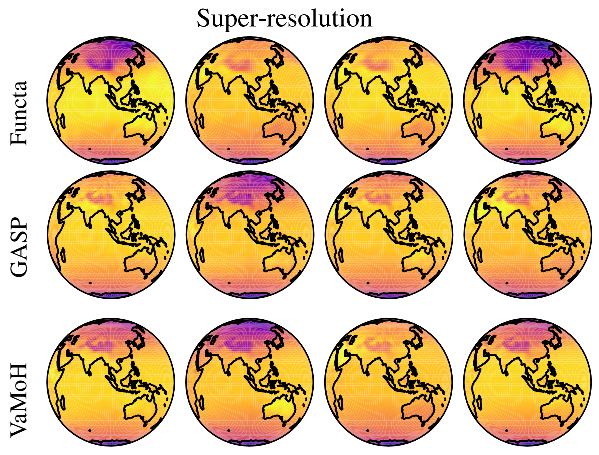


Figure 15. Comparison of generation quality of VaMoH, Functa, and GASP at original resolution (left) and x2 resolution (right).

B.3. Reconstruction

In this section, we provide a comprehensive set of figures demonstrating the capability of VaMoH in reconstructing data at both the original resolution and double the original resolution. Figure 16 and Figure 17 present the results for all datasets under examination. These figures reveal that VaMoH is able to produce high-quality reconstructions and super-reconstructions, with visual fidelity comparable or better to that of Functa, while requiring only a simple forward pass.

Furthermore, in Section 4.2, the Peak Signal to Noise Ratio (PSNR) is employed to evaluate the quality of the reconstruction, denoted as $\tilde{\boldsymbol{Y}}$, of an image, denoted as $\boldsymbol{Y}$. The computation of PSNR begins by calculating the root mean squared error (RMSE) and then PSNR itself as follows

$$\text{RMSE} = \sqrt{\frac{1}{D}\sum_{d} ||\boldsymbol{y}_d - \tilde{\boldsymbol{y}}_d||_2^2}, \tag{19}$$

$$\text{PSNR} = 20\log\left(\frac{255}{\text{RMSE}}\right). \tag{20}$$

Table 5. Comparison of inference time (seconds) for super-reconstruction task of VaMoH and Functa. On the right-most two columns, we show the speed improvement of VaMoH compared to Functa (3) which is trained with 3 gradient steps as suggested in the original paper (Dupont et al., 2022a) and Functa (10) which is trained with 10 gradient steps to obtain the results of Functa depicted in Figures 16,17. Please note that these experiments are run on the same GPU device.

	Model Inference Time (secs)			Speed Improvement	
Dataset	VaMoH	Functa (3)	Functa (10)	vs. Functa (3)	vs. Functa (10)
PolyMNIST	**0.00455**	0.01649	0.05109	**x 3.62**	**x 11.23**
Shapes3D	**0.00544**	0.01768	0.05489	**x 3.25**	**x 10.09**
CelebA HQ	**0.00833**	0.01729	0.05377	**x 2.08**	**x 6.46**
ERA5	**0.00790**	0.01997	0.06030	**x 2.53**	**x 7.63**
ShapeNet	**0.01440**	0.02089	0.06569	**x 1.45**	**x 4.56**

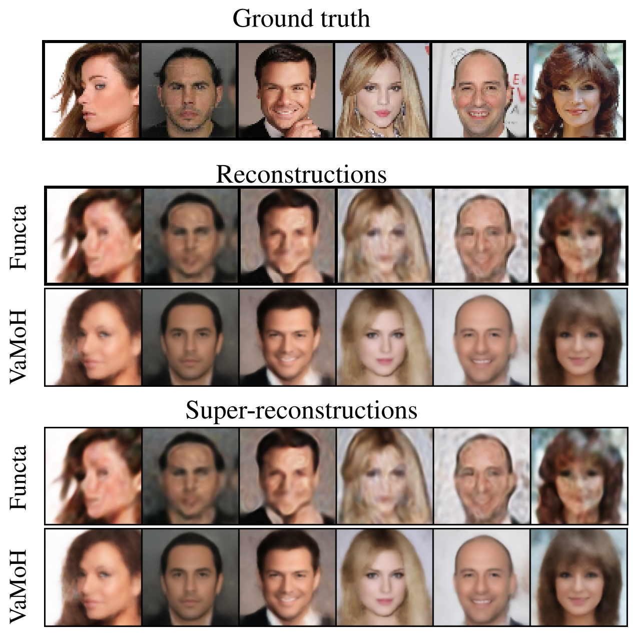
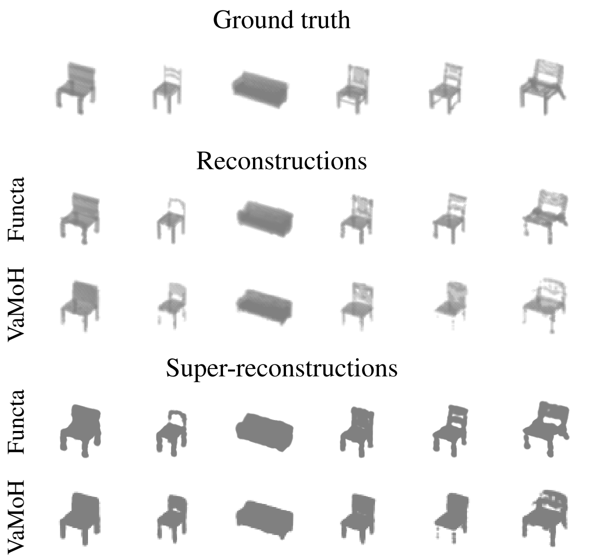
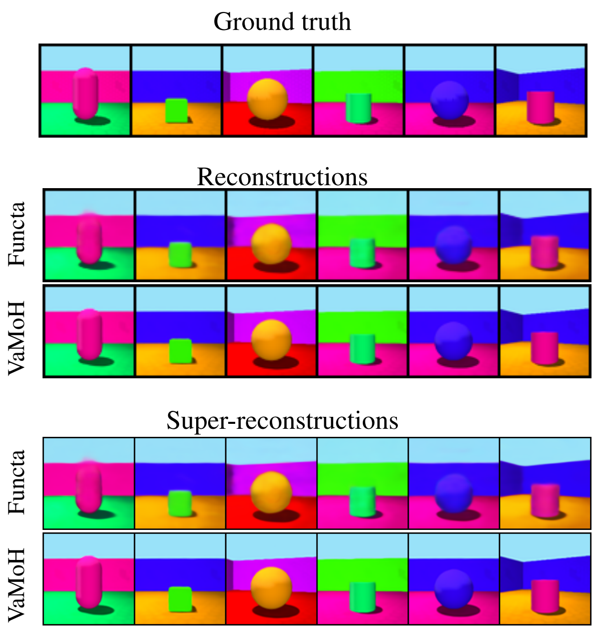


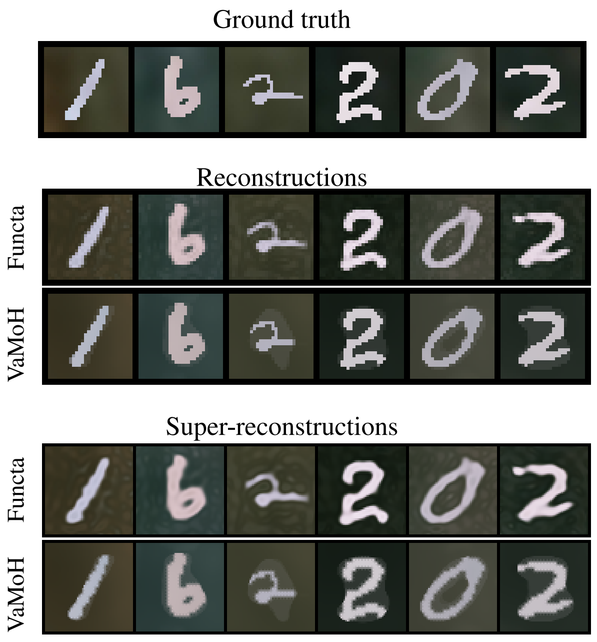
Figure 16. Comparison of reconstruction quality of VaMoH and Functa on ground truth images of CelebA HQ dataset at original resolution (top) and x2 resolution (bottom).



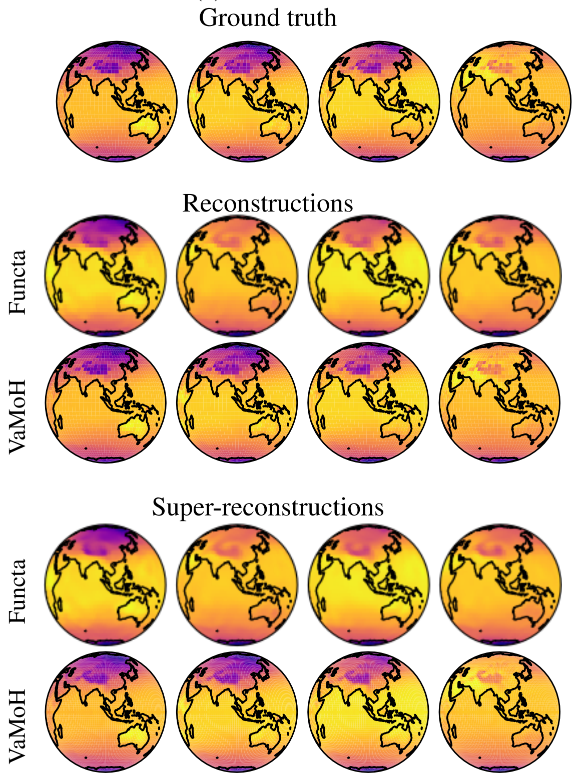
(a) SHAPENET



(b) SHAPES3D



(c) POLYMNIST

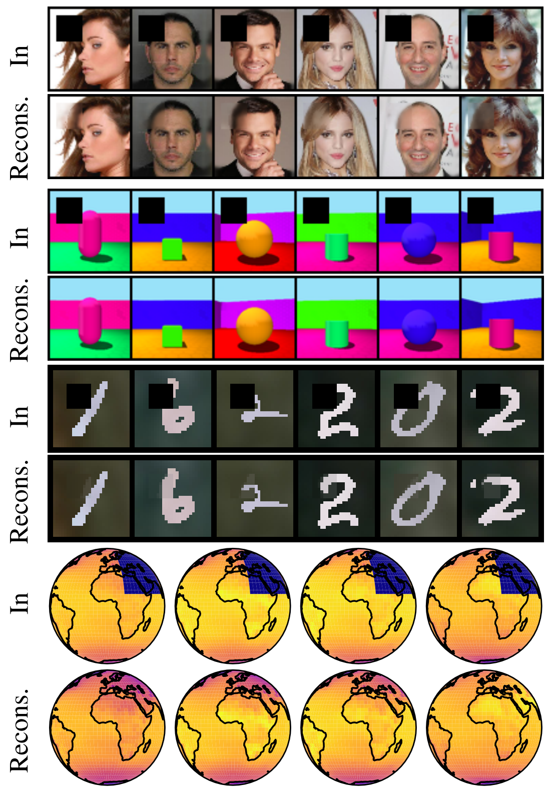


(d) ERA5

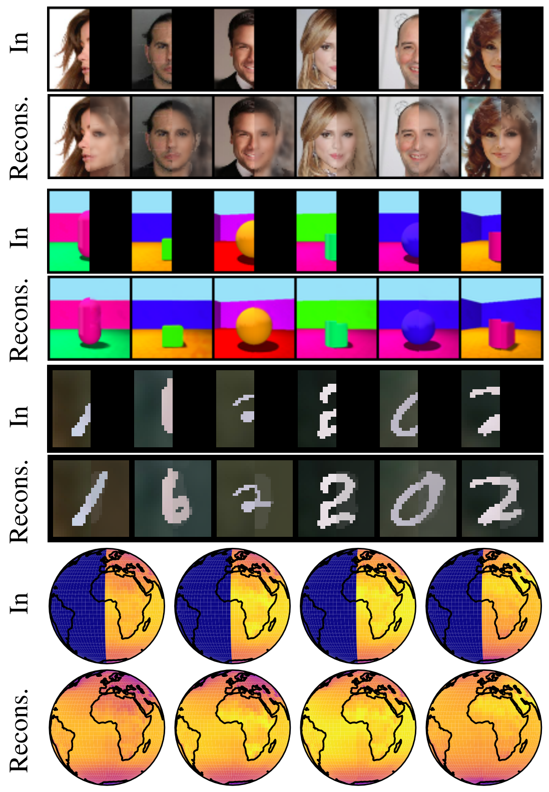
Figure 17. Comparison of reconstruction quality of VaMoH and Functa on ground truth data from the first samples of the test set at original and super-resolution.

B.4. Image completion

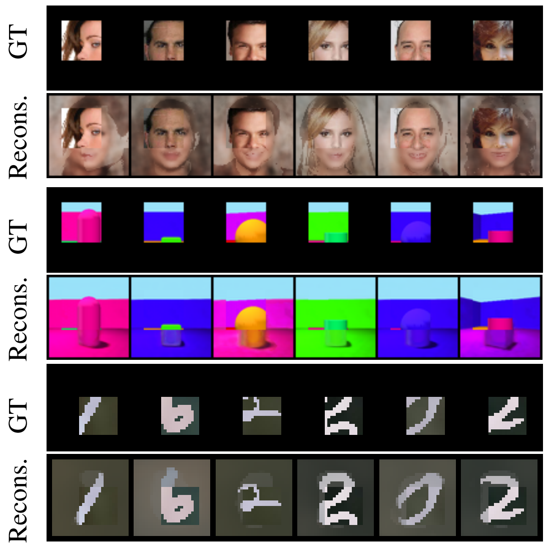
In this section, we provide qualitative results of VaMoH for various imputation tasks for different missingness patterns. As it is already highlighted in Section 3.3 and Appendix A.4, we use point dropout to increase the robustness of the encoder against missing data. In Figure 18, we provide results for image completion task on with different datasets.



(a) Missing a patch (in-painting)



(b) Missing half of the image

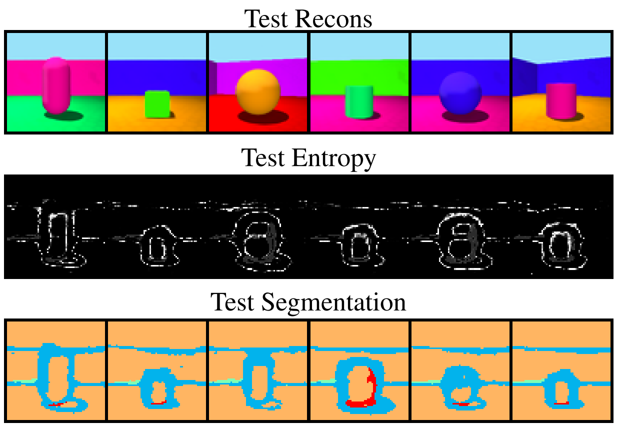


(c) Image out-painting

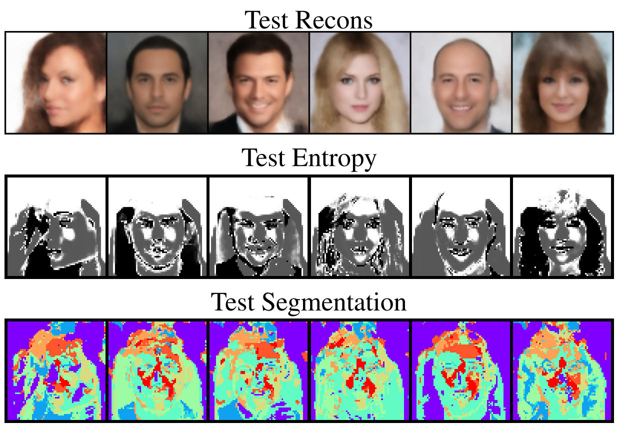
Figure 18. Imputation of different amounts of missing parts using VaMoH. For each dataset, the top row shows the input (In) and the bottom row shows the reconstructed image (Recons.).

B.5. Entropy and Segmentation

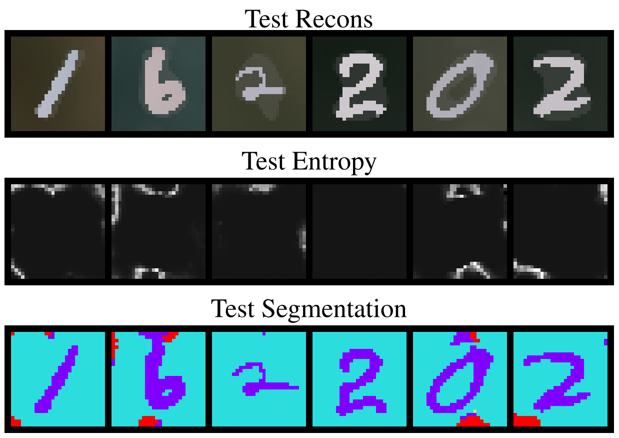
In this section, we present figures that illustrate the contribution of each generator in the image reconstruction task. Specifically, we use entropy maps to visualize the uncertainty of the posterior probabilities of the categorical latent variable. A higher value in the entropy map indicates that the probabilities are more evenly distributed among the components for a given pixel, while a lower value suggests that a smaller number of components contribute to the reconstruction of that pixel. Additionally, we use segmentation maps to visualize the component with the highest probability per pixel, where different components are denoted by different colors. As an example, Figure 19b shows that there is a high degree of uncertainty among the components that are responsible for reconstructing the background in the CELEBA HQ dataset. Conversely, for simpler datasets such as POLYMNIST, as shown in Figure 19c, the components are able to differentiate between the object and background in the images. For SHAPES3D dataset results presented in Figure 19a, the uncertainty is lower in the Regions of Interest with high variations (borders), where only one of the components focuses on generating the shape and shadow of the object, as well as the perspective of the background wall. Thus, we provide evidence that meaningful interpretations can be obtained from our Mixture of HyperGenerators.



(a) SHAPES3D Test



(b) CELEBA HQ Test



(c) POLYMNIST Test

Figure 19. Visualization of used components in mixture of generators of VaMoH for entropy (top) and segmentation (bottom) maps for SHAPES3D, CELEBA HQ, and POLYMNIST.