ARD-VAE: A Statistical Formulation to Find the Relevant Latent Dimensions of Variational Autoencoders

Surojit Saha[1], Sarang Joshi[1], and Ross Whitaker[1]

Scientific Computing and Imaging Institute, Kahlert School of Computing, The University of Utah, Salt Lake City, USA
{surojit.saha,sarang.josh,ross.whitaker}@utah.edu

Abstract. The variational autoencoder (VAE) [34, 15] is a popular, deep, latent-variable model (DLVM) due to its simple yet effective formulation for modeling the data distribution. Moreover, optimizing the VAE objective function is more manageable than other DLVMs. The bottleneck dimension of the VAE is a crucial design choice, and it has strong ramifications for the model's performance, such as finding the hidden explanatory factors of a dataset using the representations learned by the VAE. However, the size of the latent dimension of the VAE is often treated as a hyperparameter estimated empirically through trial and error. To this end, we propose a statistical formulation to discover the relevant latent factors required for modeling a dataset. In this work, we use a *hierarchical* prior in the latent space that estimates the variance of the latent axes using the encoded data, which identifies the relevant latent dimensions. For this, we replace the fixed prior in the VAE objective function with a hierarchical prior, keeping the remainder of the formulation unchanged. We call the proposed method the *automatic relevancy detection in the variational autoencoder* (ARD-VAE). We demonstrate the efficacy of the ARD-VAE on multiple benchmark datasets in finding the relevant latent dimensions and their effect on different evaluation metrics, such as FID score and disentanglement analysis.

Keywords: Variational autoencoders · Automatic relevancy detection · Intrinsic dimensions.

1 Introduction

Deep latent variable models (DLVMs) have emerged as powerful tools for modeling data distributions, generating novel examples from a distribution, and learning useful encodings for downstream applications. Many deep latent variable models have been proposed, including the variational autoencoder (VAE), adversarial autoencoder (AAE) [24], Wasserstein autoencoder(WAE) [43], and diffusion-based models [40, 11]. Besides the unsupervised representation learning [15, 24], the latent mappings produced by DLVMs find potential application in semi-supervised [15, 18], weakly supervised [44, 45] and few-shot learning [39, 23, 8]. Moreover, the quality of the generated samples produced by the diffusion probabilistic models [40, 11] is comparable to state-of-the-art (SOTA) GANs [13]. All these properties make DLVMs an important and active area of research.

Among DLVM approaches, the VAE [34, 15] has a strong theoretical foundation that uses *variational inference* to approximate the true posterior by a surrogate distribution. The VAE uses a *inference/recognition* model to estimate the parameters of the surrogate posterior distribution in an amortized framework (shared parameters). The use of the reparameterization trick in VAEs leads to efficient optimization of the lower bound on the data log-likelihood. In addition, it is simple to train a neural network with the VAE objective function relative to the AAE [24] and WAE [43](due to the use of the discriminators in these models). These advantages have made the VAE a popular DLVM, which has even helped in progressing basic science research [7, 29, 31]. However, the VAE often fails to match the aggregate distributions in the latent space (manifested by *pockets/holes* in the encoded distribution) [35, 3] and suffers from posterior collapse (uninformative latent variables) [12, 21, 20]. Several methods have been proposed for better matching of marginal posteriors [14, 41, 36] and to alleviate posterior collapse [30, 32]

In DLVMs, we generally assume that the observed data existing in a high dimensional space ($\mathbb{R}^D$), such as images, can be represented by a low dimensional manifold ($\mathbb{R}^M$) embedded in that space, where $M << D$. The size of the manifold, aka *intrinsic* data dimension, can be interpreted as the number of latent factors used to produce the observed data (ignoring the noise). The bottleneck size of DLVMs represents the *intrinsic* data dimension. However, the true *intrinsic* dimension is generally unknown for real-world datasets, and finding the correct dimension remains an open and challenging problem. Studies have shown that incorrect bottleneck dimensions of DLVMs result in inaccurate modeling of the data distribution, which subsequently affects the interpretation of the latent representations produced by a DLVM [43, 3]. However, only a few methods [26, 4] have addressed this problem in DLVMs. For instance, the MaskAAE [26] introduces a trainable masking vector in the latent space of a deterministic autoencoder (used in [24, 43]) to determine the number of active dimensions. In [4], the authors apply a L_0 regularization on the masking vector to prune unnecessary dimensions in the VAE. In both methods, additional regularization loss is added to the primary objective function with associated hyperparameters. It is observed that the proposed methods are sensitive to the choice of the neural architecture architectures and settings of the hyperparameters used in the objective function. Thus, the methods are useful only with certain architectures or training strategies and do not generalize well to different scenarios.

Considering the importance of the bottleneck dimension in the VAE and limitations in the existing methods, we propose a method to determine the relevant latent dimensions for a dataset using a *hierarchical* prior [22, 42] introduced in [28], where the prior distribution, $p(\mathbf{z})$, is dependent on another random variable, $\boldsymbol{\alpha}$, a *hyperprior*. In the proposed statistical formulation, the hierarchical prior distribution is defined as $p(\mathbf{z} \mid \boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ estimates the precision of the latent dimensions using encoded data. The variance ($\boldsymbol{\alpha}^{-1}$) along the latent axes is used to determine the relevant latent factors required to represent a dataset, such that the unwanted latent variables, through training, obtain very low variance. Thus, the ARD-VAE introduces a natural extension to the fixed prior used in the VAE that is flexible and is learned from the data. Unlike other approaches [26, 4], we do not modify the ELBO objective of the VAE, except the use of the hierarchical prior. Following are the contributions of this work:

– We propose the ARD-VAE to discover relevant axes in the VAE using a hierarchical prior without modification of the ELBO in the VAE.
– The ARD-VAE is insensitive to the choice of the autoencoder architecture and optimization strategies.
– Empirical evaluations demonstrate the effectiveness of the ARD-VAE in modeling data distributions and finding relevant generative factors.

2 Related Work

VAEs often fail to match the aggregate posterior that subsequently affects the modeling of the data distribution [35, 3, 37]. Matching the aggregate posterior also makes the latent dimensions independent, thus encouraging disentanglement of the latent factors [16, 14, 41]. The regularized autoencoder (RAE) [5] estimates the aggregate encoded distribution (using an ex-post density estimator) to model the data distribution, where it replaces the stochastic autoencoders used in VAEs with its deterministic counterpart. All these analyses encourage the matching of aggregate distribution in VAEs as done in other DLVMs, such as the AAE [24] and WAE [43], to improve the performance of the model. In this work, we study the dimensionality of the original VAE formulation (which matches conditional distributions to the prior), with the understanding that it applies readily to aggregate-matching methods.

The mismatch between the size of the bottleneck dimension (chosen in VAEs) and the intrinsic data dimension is shown to be a critical factor in designing a VAE. A detailed analysis of the effect of mismatch between the dimensions is done in [3], which elucidates the importance of finding the correct size of the latent space. Finding the intrinsic data dimension is important for any DLVM [43] and is not unique to VAEs.

A few methods have been proposed in the recent past to address this issue; one is designed for VAEs [4] and the other one for AAEs [26]. The method in [4] (referred to herein as GECO-L_0-ARM-VAE) introduces a boolean gating vector (matching the bottleneck dimension) to find the cardinal latent axes required for modeling a dataset in VAEs. The latent encodings are combined with the shared gating vector (using the element-wise product) to optimize the ELBO objective function of the VAE. To encourage sparsity in the gating vector, the authors have augmented the ELBO with a L_0 regularization loss on the gating vector. However, optimization of the L_0 loss using gradient descent is challenging, and it uses the L_0-ARM [17] to update the gradients. To make the effect of the regularization of the latent representations more interpretable to the users, they use a threshold for the reconstruction loss, as in [33]. In optimizing the objective function of the GECO-L_0-ARM-VAE, the parameters of the gating vector are updated after the desired reconstruction loss is achieved, i.e., after the model parameters have learned sufficient statistics of the data. After that, all model parameters are trained jointly.

The MaskAAE [26] uses a trainable masking vector similar to [4] to determine the relevant latent dimensions. Unlike the method in [4], the MaskAAE is designed for the AAE [24] and WAE [43] that uses deterministic autoencoders and matches the aggregate (or marginal) posterior to the prior. The Haramard products of the latent encodings with the global masking vector are used as inputs to the decoder and discriminator. The masking vector is initialized with continuous values for gradient updates. A regularization loss, the masking loss, is added to the objective function of the AAE, along with an additional loss term to the reconstruction loss.

Both the GECO-L_0-ARM-VAE and MaskAAE use a lot of hyperparameters and different initialization strategies, which are most likely to be sensitive to neural architectures and other optimization settings. The MaskAAE uses a complex training recipe to optimize different loss functions and update hyperparameters. Thus, the typical settings used in both methods raise questions about using the methods on new datasets, which might require changes in the VAE/AAE architecture. In contrast, the proposed method has a single hyperparameter to balance the reconstruction and regularization loss in the objective function (similar to any DLVM). Furthermore, it is flexible enough to be used with virtually any VAE architecture and, thus, can be trained on any new dataset.

3 Automatic Relevancy Detection in the Variational Autoencoder

3.1 Background

The VAE is a generative model that approximates the unknown data distribution, $p(\mathbf{x})$, by learning the joint distribution of the latent variables, $\mathbf{z}$, where $\mathbf{z} \in \mathbb{R}^L$, and the observed variables, $\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^D$. The VAE maximizes the lower bound on the data log likelihood in 1, known as the evidence lower bound (ELBO),

$$\max_{\theta,\phi} \mathbb{E}_{p(\mathbf{x})}\Big[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log\Big(p_\theta(\mathbf{x} \mid \mathbf{z})\Big) - \mathrm{KL}\Big(q_\phi(\mathbf{z} \mid \mathbf{x})||p(\mathbf{z})\Big)\Big]. \tag{1}$$

VAEs use a probabilistic encoder, $\mathbf{E}_\phi$, and a probabilistic decoder, $\mathbf{D}_\theta$, to represent $q_\phi(\mathbf{z} \mid \mathbf{x})$ and, $p_\theta(\mathbf{x} \mid \mathbf{z})$, respectively. Both $\mathbf{E}_\phi$ and $\mathbf{D}_\theta$ are usually deep neural networks parameterized by ϕ and θ, respectively. The prior distribution, $p(\mathbf{z})$, is chosen to be $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and the surrogate posterior is a Gaussian distribution with diagonal covariance (under the assumption of independent latent dimensions), which is defined as follows:

$$q_\phi(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}\left(\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\sigma}^2_\mathbf{x}\mathbf{I}\right), \text{where } \boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\sigma}^2_\mathbf{x} \leftarrow \mathbf{E}_\phi(\mathbf{x}). \tag{2}$$

The choice of the Gaussian distribution as the posterior, $q_\phi(\mathbf{z} \mid \mathbf{x})$, helps in efficient computation of the $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x} \mid \mathbf{z})$ using the reparameterization trick introduced by the VAE,

$$\mathbf{z} \leftarrow \boldsymbol{\mu}_\mathbf{x} + \epsilon \odot \boldsymbol{\sigma}_\mathbf{x}, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \text{ and } \mathbf{z} \text{ representing samples from } q_\phi(\mathbf{z} \mid \mathbf{x}). \quad (3)$$

Moreover, we have a closed form solution for the KL divergence between Gaussian distributions in 1. The multivariate Gaussian distribution or Bernoulli distribution is used as the likelihood distribution, $p_\theta(\mathbf{x} \mid \mathbf{z})$. The VAE jointly optimizes the parameters of the $\mathbf{E}_\phi$ and $\mathbf{D}_\theta$ using the objective function in 1, under the modeling assumptions discussed above.

3.2 Formulation for Relevance-Aware Prior

In this paper, we propose a statistical formulation to determine the number of relevant latent axes for a dataset using a given encoder-decoder architecture. To this end, we allow the prior distribution, $p(\mathbf{z})$ (a Gaussian), to have different variance along each latent axis, represented by another random variable $\boldsymbol{\alpha}$, which forms a hyperprior. Under this setting, the latent axes with high variances should represent important latent factors and unnecessary latent dimensions are expected to exhibit relatively low variances. To this end, we use a hierarchical prior [28, 22] on independent latent dimensions (as in VAEs) defined as,

$$p(\mathbf{z} \mid \boldsymbol{\alpha}) = \prod_{l=1}^{L} \mathcal{N}\left(z_l; 0, \alpha_l^{-1}\right), \quad (4)$$

$$p(\boldsymbol{\alpha}) = \prod_{l=1}^{L} \mathrm{Gamma}(\alpha_l; a_l^0, b_l^0). \quad (5)$$

where α_l^{-1} is the variance along latent axis l. The distribution over the independent precision variable (α_l) is chosen as the associated conjugate prior, which is the Gamma distribution.

The parameters of the Gamma distribution are estimated analytically using data in the latent space, $D_\mathbf{z}$, produced using the posterior distribution of the VAE estimated by the encoder, $\mathbf{E}_\phi$ (refer to algorithm 1 in the appendix). Thus, the distribution over $\boldsymbol{\alpha}$ conditioned on data, $D_\mathbf{z}$, is defined as $p(\boldsymbol{\alpha} \mid D_\mathbf{z})$. In this limited context, $p(\boldsymbol{\alpha} \mid D_\mathbf{z})$ can be interpreted as a posterior distribution, which is $\propto p(D_\mathbf{z} \mid \boldsymbol{\alpha})p(\boldsymbol{\alpha})$. Given $p(\boldsymbol{\alpha})$ as the Gamma distribution (prior), the posterior $p(\boldsymbol{\alpha} \mid D_\mathbf{z})$ is also a Gamma distribution (conjugate prior) when $p(D_\mathbf{z} \mid \boldsymbol{\alpha})$ is a Gaussian distribution (likelihood). The parameters of $p(\boldsymbol{\alpha} \mid D_\mathbf{z})$ can be derived analytically for each latent axis independently. The parameters of $p(\alpha_l \mid D_\mathbf{z})$ for each axis for a given mean $\boldsymbol{\mu}^{\boldsymbol{\alpha}} \in \mathbb{R}^L$ (mean of the likelihood distribution, $p(D_\mathbf{z} \mid \boldsymbol{\alpha})$) are [27]:

$$a_l = a_l^0 + \frac{n}{2}, \quad (6)$$

$$b_l = b_l^0 + \frac{\sum_i^n (z_l^i - \mu_l^{\boldsymbol{\alpha}})^2}{2}, \tag{7}$$

where n is the number of samples in $D_{\mathbf{z}}$ and $\mathbf{z}^i \in D_{\mathbf{z}}$. The parameters of the Gamma distribution for all the latent axes are denoted by $\boldsymbol{a}_L$ and $\boldsymbol{b}_L$.

With the introduction of data, $D_{\mathbf{z}}$, the hierarchical prior distribution in 4 ($p(\mathbf{z} \mid \boldsymbol{\alpha})$) is modified to $p(\mathbf{z} \mid \boldsymbol{\alpha}, D_{\mathbf{z}})$. The new distribution can be factorized as $p(\mathbf{z} \mid \boldsymbol{\alpha}, D_{\mathbf{z}}) = p(\mathbf{z} \mid \boldsymbol{\alpha})p(\boldsymbol{\alpha} \mid D_{\mathbf{z}})$. For optimization of the ELBO objective in 1, we need to remove $\boldsymbol{\alpha}$ from $p(\mathbf{z} \mid \boldsymbol{\alpha}, D_{\mathbf{z}})$. The hyperprior $\boldsymbol{\alpha}$ is removed by marginalization (as in 8), and that integration results in a Student's t-distribution for each latent variable, as shown in 9 (please refer to exercise 2.46 of [1]). The Student's t-distribution encourages sparsity in the latent variable $\mathbf{z}$ (refer to section 5.1 in [42]). The degrees of freedom, ν_l (10), and t-statistics, t_l (11), of the t-distributions are linear functions of the parameters of the Gamma distribution used for modeling $\boldsymbol{\alpha}$ (defined in 6 and 7).

$$p(\mathbf{z} \mid D_{\mathbf{z}}) = \int p(\mathbf{z} \mid \boldsymbol{\alpha}) p(\boldsymbol{\alpha} \mid D_{\mathbf{z}}) d\boldsymbol{\alpha} \tag{8}$$

$$p(\mathbf{z} \mid D_{\mathbf{z}}) = \prod_{l=1}^{L} \frac{1}{s_l} \frac{\Gamma(\frac{\nu_l+1}{2})}{\sqrt{\pi\ \nu_l}\ \Gamma(\frac{\nu_l}{2})} \left(1 + \frac{t_l^2}{\nu_l} \right)^{-(\nu_l+1)/2}, \tag{9}$$

$$\text{where, } \nu_l = 2 * a_l \tag{10}$$

$$t_l^2 = \frac{(z_l - \mu_l)^2}{b_l / a_l} \tag{11}$$

$$s_l = \sqrt{b_l / a_l}. \tag{12}$$

$$\tag{13}$$

With the marginalized distribution, $p(\mathbf{z} \mid D_{\mathbf{z}})$ (*target distribution* in VAEs), we can optimize the KL divergence term in the ELBO objective function in 1. Thus, the objective function of the ARD-VAE is defined as follows:

$$\max_{\theta,\phi} \mathbb{E}_{p(\mathbf{x})} \Big[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \Big(p_\theta(\mathbf{x} \mid \mathbf{z}) \Big) - \mathrm{KL} \Big(q_\phi(\mathbf{z} \mid \mathbf{x}) || p(\mathbf{z} \mid D_{\mathbf{z}}) \Big) \Big]. \tag{14}$$

The objective function of the proposed method did not modify the ELBO derivation used in the conventional VAE [34], except the use of the hierarchical prior distribution. Unlike methods in [26, 4], we do not add a regularizer to the objective function in 1 to determine the relevant axes. The number of samples in the set $D_{\mathbf{z}}$ gives us the degrees of freedom, ν, of the Student's t-distribution in 9. The t-distribution is approximately the Gaussian distribution for higher ν, which allows for a more efficient, analytical expression for the KL-divergence as in 15. Thus, the proposed formulation supports optimization of the ELBO under different scenarios, depending on the size of $D_{\mathbf{z}}$.

3.3 Training the ARD-VAE

In addition to the reconstruction loss, the objective function of the ARD-VAE (defined in 14) has a KL-divergence loss for matching the conditional posterior,

$q_\phi(\mathbf{z} \mid \mathbf{x})$, to the target, $p(\mathbf{z} \mid D_\mathbf{z})$. The parameters of the target distribution, $p(\mathbf{z} \mid D_\mathbf{z})$ (in 10 and 11), are dependent on the parameters of $p(\boldsymbol{\alpha} \mid D_\mathbf{z})$ (in 6 and 7), which are learned from data. Thus, the ARD-VAE requires sample data to optimize the objective function in 14 and learn the parameters of the hyperprior distribution (in 6 and 7). To address this issue, the training data, $\mathcal{X}_{train}$ is split into $\mathcal{X}_{sgd}$ and $\mathcal{X}_{\boldsymbol{\alpha}}$, such that $\mid \mathcal{X}_{\boldsymbol{\alpha}} \mid < \mid \mathcal{X}_{sgd} \mid$. The set $\mathcal{X}_{sgd}$ is used to produce minibatches for optimization of the ARD-VAE objective function in 14. Samples in $\mathcal{X}_{\boldsymbol{\alpha}}$ are used to estimate the parameters, $\mathbf{a}_L$ and $\mathbf{b}_L$, of $p(\boldsymbol{\alpha} \mid D_\mathbf{z})$ using the data $D_\mathbf{z}$ produced by the algorithm 1 (refer to the appendix).

The update of the dataset, $D_\mathbf{z}$, is set to lag for a few minibatches, as its update in every minibatch (an ideal scenario) would change the prior(/target) distribution in the KL divergence term in 14, leading to optimization challenges. The lagging $D_\mathbf{z}$ gives stability to the optimization of the ARD-VAE objective function. Moreover, the delayed update of $D_\mathbf{z}$ offers computational benefits. Using samples in the $D_\mathbf{z}$, the parameters of $p(\boldsymbol{\alpha} \mid D_\mathbf{z})$ are updated using 6 and 7. In our work, we set $\boldsymbol{a}_L^0 = 0$ and $\boldsymbol{b}_L^0 = 0$ (uninformative) as in [42]. The mean vector $\boldsymbol{\mu}^{\boldsymbol{\alpha}}$ used in 7 is also set to $\mathbf{0}$. Under this configuration, $\boldsymbol{a}_L^0$ is fixed (samples size of $D_\mathbf{z}$) and only $\boldsymbol{b}_L^0$ is modified using the latest $D_\mathbf{z}$.

VAEs are trained in an unsupervised framework, having access to a large amount of data. Thus, the prior distribution can be approximated as a Gaussian distribution using sufficient samples in $D_\mathbf{z}$. With $p(\mathbf{z} \mid D_\mathbf{z})$ as an approximate Gaussian distribution, the KL divergence in 14 results in

$$-\frac{L}{2} - \frac{1}{2}\sum_{i=1}^{L} \log \sigma_i^2 + \frac{1}{2}\sum_{i=1}^{L} \log \hat{\sigma}_i^2 + \frac{1}{2}\sum_{i=1}^{L} \frac{\mu_i^2 + \sigma_i^2}{\hat{\sigma}_i^2}, \tag{15}$$

where $\boldsymbol{\mu}, \boldsymbol{\sigma}^2 \leftarrow \mathbf{E}_\phi(\mathbf{x}), \mathbf{x} \in \mathcal{X}_{sgd}$ and

$\hat{\boldsymbol{\sigma}}^2 = \boldsymbol{b}_L / \boldsymbol{a}_L, \boldsymbol{a}_L$ and $\boldsymbol{b}_L$ are estimated using the latest $D_\mathbf{z}$.

In this work, we use the closed-form expression of the KL divergence in 14. However, one can still use the Student's t-distributions in 9 as the prior distribution. In that scenario, one would use samples from $q_\phi(\mathbf{z} \mid \mathbf{x})$ to compute the KL divergence loss in 14.

To balance the loss terms in the ARD-VAE objective function, we introduce a hyperparameter β as in [10], and the modified objective function is

$$\max_{\theta,\phi} \mathbb{E}_{p(\mathbf{x})} \Big[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \Big(p_\theta(\mathbf{x} \mid \mathbf{z}) \Big) - \beta \, \mathrm{KL} \Big(q_\phi(\mathbf{z} \mid \mathbf{x}) || p(\mathbf{z} \mid D_\mathbf{z}) \Big) \Big]. \tag{16}$$

We tune β to achieve the desired reconstruction loss on the validation data. The outline of training the ARD-VAE is detailed in the algorithm 2 in the appendix. For practical purposes, $D_\mathbf{z}$ is updated after every epoch ($u_{D_\mathbf{z}} = 1$ in the algorithm 2 in Appendix) with a new subset of training samples, $\mathcal{X}_{\boldsymbol{\alpha}}$. Therefore, the SGD samples, $\mathcal{X}_{sgd} = \mathcal{X}_{train} - \mathcal{X}_{\boldsymbol{\alpha}}$, are also updated in the process. The typical size of $\mid \mathcal{X}_{\boldsymbol{\alpha}} \mid = 10K$ for all the datasets studied in this work (please see the ablation study in the Appendix).

3.4 Determining the Relevant Axes

The ARD-VAE does not use a trainable masking vector as in [4, 26] to determine the relevant axes. Instead, it relies on the precision of the latent axes, $\boldsymbol{\alpha}$, to identify the spurious dimensions after the model is trained to convergence. The estimated variance is computed using the parameters, $\boldsymbol{a}_L$ and $\boldsymbol{b}_L$, of the hyperprior distribution $p(\boldsymbol{\alpha} \mid D_{\mathbf{z}})$,

$$\hat{\boldsymbol{\sigma}}^2 = \boldsymbol{b}_L / \boldsymbol{a}_L. \tag{17}$$

As shown in Fig. 1(a, b), the estimated variance for the collapsed dimensions (no effect on the decoder output) has relatively low but non-zero values. Moreover, we observe variations in the estimates across latent dimensions in Fig. 1(a). Thus, the choice of a threshold or the percentage of variance (used in PCA) for selecting relevant dimensions that works under different scenarios is non-trivial.

For axes that are not relevant to the reconstruction, we have observed that the decoder produces virtually no variability in output in response to deviations along these axes. This is consistent with the fact that these axes are not being used for reconstruction. This motivates us to consider the deviation of the output, $\hat{\mathbf{x}} \in \mathbb{R}^D$ (produced by the decoder $\mathbf{D}_\theta$), with respect to the mean representation, $\boldsymbol{\mu}_{\mathbf{x}} \in \mathbb{R}^L (\boldsymbol{\mu}_{\mathbf{x}} \leftarrow \mathbf{E}_\phi(\mathbf{x}))$, as the measure of relevance. We leverage this idea to find a weight vector $\mathbf{w}_{\hat{\boldsymbol{\sigma}}} \in \mathbb{R}^L$ for better estimation of the relevant axes. The weight vector $\mathbf{w}_{\hat{\boldsymbol{\sigma}}}$ is computed as follows:

$$\mathbf{w}_{\hat{\boldsymbol{\sigma}}} = \sum_{k=1}^{D} \frac{1}{N} \sum_{i=1}^{N} \|\mathbb{J}_i\|^2, \tag{18}$$

$$\text{where } \mathbb{J} = \left[\frac{\partial \hat{\mathbf{x}}}{\partial \mu_1} \cdots \frac{\partial \hat{\mathbf{x}}}{\partial \mu_L} \right] \in \mathbb{R}^{D \times L} \text{ is the Jacobian matrix.}$$

To get a reliable estimate of the weight vector $\mathbf{w}_{\hat{\boldsymbol{\sigma}}}$ using 18, we compute the Jacobian for multiple data samples, which is typically the size of a minibatch (100) in this work. The weighted estimated variance defined as

$$\hat{\boldsymbol{\sigma}}^2_{\mathbf{w}} = \mathbf{w}_{\hat{\boldsymbol{\sigma}}} \odot \hat{\boldsymbol{\sigma}}^2 \tag{19}$$

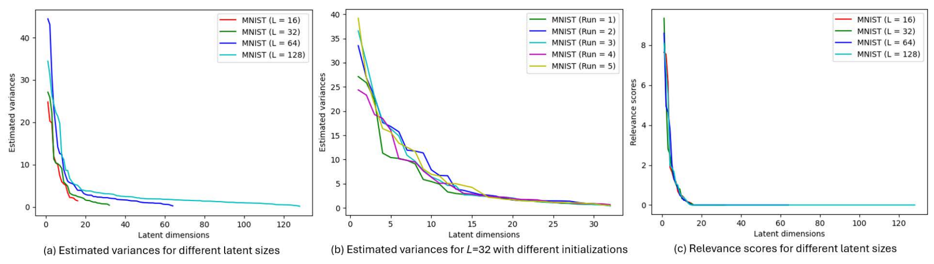

Fig. 1. The estimated variances on the MNIST dataset using 17 (a) for different latent sizes and (b) with different initialization for $L = 32$. (c) The relevance scores computed on the same models using 19.

gives us the *relevance score* that is used to determine the relevant axes of the ARD-VAE. The relevance score squashes the variance for the spurious latent axes (removes noise) and relatively scales the variance of the relevant dimensions, as shown in Fig. 1(c), and the scores for different Ls are almost similar. Thus, $\hat{\boldsymbol{\sigma}}^2_{\mathbf{w}}$ is better suited to find the relevant/necessary dimensions using the percentage-based approach, typically explaining 99% of the variance in $\hat{\boldsymbol{\sigma}}^2_{\mathbf{w}}$ for this work. The importance of latent axes estimated using 18 can also be used to determine the relevant axes of a trained VAE and its variants.

4 Experiments

The motivation of the ARD-VAE is to determine the intrinsic dimensions of a dataset in the VAE. Finding the relevant latent axes helps in improving the modeling of the data distribution. Thus, it is important to compare the learned data distribution with the true distribution. In addition, it would be interesting to examine whether the relevant dimensions identified by the ARD-VAE encode any meaningful information. This is related to the *disentanglement analysis* studied with VAEs [10, 16, 41, 14], where we aim to find whether the relevant latent axes that can explain variability in the observed data.

Evaluation Metrics : Comparison of the model's data distribution and the true but unknown data distribution is challenging, and several methods have been proposed to compare distributions [38, 9]. We use the Fréchet Inception Distance (FID) [9] to quantify the quality of the samples produced by a model. In the disentanglement analysis, we evaluate whether the latent dimensions of a DLVM represent true generative factors of the observed data. In an ideal scenario, each latent axis must correspond to a single generative factor. We consider two popular metrics, (i) the FactorVAE metric [14] and (ii) the mutual information gap (MIG) [41] in our analysis.

Datasets : We use multiple benchmark datasets for the evaluation of different DLVMs using several metrics discussed above. Samples in the MNIST, CelebA, and CIFAR10 datasets are used to model the corresponding data distributions. We follow the standard (train, test) split of the data as reported in the literature. The DSprites [25] and the 3D Shapes [2] datasets are specially curated for the disentanglement analysis, where we know the true generative factors. This helps in quantifying the disentanglement in the learned latent representations. Moreover, the number of relevant axes estimated by the ARD-VAE for the DSprites [25] and 3D Shapes [2] can be compared with the number of known true generative factors of these datasets.

Competing Methods : We consider the GECO-L_0-ARM-VAE [4] and MaskAAE [26] in our analysis, which are developed to find the intrinsic data dimensions in DLVMs. To demonstrate the efficacy of the ARD-VAE over other SOTA DLVMs, we use the baseline VAE [15], β-TCVAE [41], RAE [5], WAE [43], and DIP-VAE [16] in our study.

Implementation Details : Different neural network architectures and model-specific hyperparameters are used for training a DLVM, depending on the complexity of the data. For a fair comparison of the competing methods, we fix the number of latent dimensions, neural network architecture, and optimization settings (such as the learning rate, learning rate scheduler, epochs, and batch size) for a dataset. Details of the neural network architectures and related hyperparameter settings used in training different models on the benchmark datasets are reported in the Appendix. All models are trained with 5 different initialization seeds for every dataset studied in this work.

4.1 Results

Evaluation of the Learned Data Distribution: In this analysis, we train different DLVMs matching the conditional posterior (used in the baseline VAE and its variants) or aggregate posterior (used in WAE) to the prior using the encoder-decoder architectures reported in the Appendix. The RAE method, unlike VAEs, produces a *deterministic* latent encoding with associated regularization (i.e., L_2) on the latent parameters. The method then uses a Gaussian approximation to the distribution of samples in the latent space. Due to its effective regularization strategies and favorable results, we consider the RAE to be important for comparison. In addition, we evaluate the performance of the MaskAAE and GECO-L_0-ARM-VAE, which are designed to find the intrinsic dimension of a dataset using different DLVM frameworks. In this study, we consider the standard encoder-decoder architecture reported in the literature [43, 5] for training models on MNIST, CelebA, and CIFAR10 datasets. The initial latent dimensions used for the MNIST, CelebA, and CIFAR10 datasets are $L = 16$, $L = 64$, and $L = 128$, respectively [5]. The motivation of this analysis is to examine whether the proposed ARD-VAE is able to find the relevant axes required for modeling a dataset using the standard SOTA encoder-decoder architectures and to examine how close the learned data distribution is to the true distribution. The estimated variance in 17 is used to generate samples for the ARD-VAE. Besides, the FID scores of the generated samples, reported as GEN-FID, we report the FID scores of the reconstructed samples as REC-FID. The number of latent dimensions used by a method to reconstruct/generate samples are reported as ACTIVE. Using the importance of the latent axes estimated using 18, we determine the relevant axes for the VAE, β-TCVAE, RAE, and WAE that are used for model evaluation (refer to the appendix for this result).

The average FID scores (along with the standard deviations) of different methods are reported in Table 1, where all the models are trained with 5 different initialization of the trainable parameters. The relevant dimensions estimated by the ARD-VAE are consistently less than the initial (or nominal) dimension L for all the datasets. We have no knowledge of any "ground truth" intrinsic dimension for these datasets. Instead, we rely on comparisons with empirically derived bottle-neck sizes from literature and evaluate whether the subset of learned dimensions is sufficient to represent or reproduce the data. These results show that the FID scores of the ARD-VAE are better than the baseline VAE except for the CelebA dataset (slightly higher). Though the WAE's reconstructed samples' FID score is low for the MNIST and CelebA datasets, the FID score for the *generated samples* is high for all the datasets [5]. The WAE-MMD ignores the higher order moments in matching distributions [6], possibly leading to *holes* in the encoded distribution and resulting in higher FID scores. The RAE performs well across multiple datasets under different scenarios, conceivably due to the use of aggregate latent distribution to generate samples [43, 3]. However, the learned representations cannot be readily used for subsequent statistical analysis, such as the disentanglement analysis and outlier detection [36], due to the lack of structure in the latent representations.

The MaskAAE and GECO-L_0-ARM-VAE, which are designed to find the relevant axes to represent a dataset using DLVMs, have failed, using this encoder-decoder architecture, to learn the data distribution, as indicated by significantly high Gen-FID scores relative to other methods. This result may signal issues about the generalization of those formulations and sensitivity to the hyperparameter tuning relative to different architectures and datasets (as will be further examined in the following analysis). The number of relevant (or /Active) dimensions estimated by the GECO-L_0-ARM-VAE are sufficient to produce reasonably good reconstructed samples, indicated by comparable Rec-FID scores. However, using this encoder-decoder architecture, it failed to map samples to the prior distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I})$, in the latent space. In contrast, the number of relevant dimensions estimated by the MaskAAE collapsed for the complex datasets, such as the CelebA and CIFAR10. We have tuned the hyperparameters of the MaskAAE over a series (i.e., dozens) of experiments, and no better results could be obtained. The number of hyperparameters corresponding to different regularization losses added to the AAE loss, associated with a complex training recipe, makes the tuning of the MaskAAE difficult. Compared to the MaskAAE and GECO-L_0-ARM-VAE, the ARD-VAE could find the relevant dimensions required to model the data distribution, supported by favorable (i.e., the best or nearly best) FID scores. Both the GECO-L_0-ARM-VAE and ARD-VAE have a single hyperparameter. The values of τ and β in GECO-L_0-ARM-VAE and ARD-VAE, respectively, are set such that the reconstruction losses across datasets are comparable to other methods. This experiment demonstrates the ability of the ARD-VAE to find relevant axes and learn the data distribution.

Sensitivity of β to the Initial Bottleneck Size L in the ARD-VAE: In real applications, we would not know the intrinsic dimension of a dataset.

Table 1. The FID scores of the reconstructed (Rec-FID) and generated samples (Gen-FID) along with the number of latent dimensions (Active) used by the competing methods. The **best** FID score is in **bold** and the $\underline{\text{second best}}$ score is $\underline{\text{underlined}}$.

	MNIST ($L = 16$)			CelebA ($L = 64$)			CIFAR10 ($L = 128$)		
	Active	Rec-FID ↓	Gen-FID ↓	Active	Rec-FID ↓	Gen-FID ↓	Active	Rec-FID ↓	Gen-FID ↓
VAE	16	26.45 ± 0.23	28.78 ± 0.48	64	41.98 ± 0.33	$\underline{49.89 \pm 0.57}$	128	136.21 ± 1.18	147.74 ± 0.81
β-TCVAE	16	48.46 ± 0.83	50.62 ± 1.19	64	43.37 ± 0.62	50.14 ± 0.78	128	172.87 ± 1.70	180.94 ± 1.16
RAE	16	$\mathbf{7.98 \pm 0.22}$	$\mathbf{18.79 \pm 0.31}$	64	$\mathbf{39.86 \pm 0.64}$	$\mathbf{48.81 \pm 1.02}$	128	$\mathbf{53.76 \pm 0.59}$	94.34 ± 1.58
WAE	16	9.86 ± 0.19	25.42 ± 1.19	64	$\underline{40.79 \pm 0.16}$	72.01 ± 2.26	128	66.18 ± 1.16	140.49 ± 0.64
GECO-L_0-ARM-VAE	10.00 ± 1.10	$\underline{8.76 \pm 1.13}$	304.75 ± 64.29	35.60 ± 3.07	42.28 ± 0.8	294.97 ± 28.45	68.00 ± 2.97	92.33 ± 2.83	320.75 ± 45.09
MaskAAE	9.80 ± 1.60	143.80 ± 47.73	144.92 ± 16.80	5.4 ± 0.80	343.57 ± 34.00	333.40 ± 10.91	3.80 ± 0.40	394.26 ± 34.27	298.30 ± 8.44
ARD-VAE	12.80 ± 0.40	17.50 ± 0.23	$\underline{22.24 \pm 0.57}$	45.20 ± 0.40	45.65 ± 0.25	52.35 ± 0.52	105.80 ± 1.33	$\underline{59.68 \pm 2.21}$	$\underline{87.56 \pm 1.21}$
ARD-VAE—$2L$[1]	12.60 ± 0.49	17.53 ± 0.10	22.30 ± 0.33	48.60 ± 0.80	46.72 ± 0.34	53.86 ± 0.68	116.40 ± 3.72	59.89 ± 7.58	$\mathbf{86.50 \pm 1.43}$

[1] The initial dimension of the ARD-VAE—$2L$ is two times L (the initial dimension used by all the methods).

Table 2. The number of active dimensions (Active) discovered by the ARD-VAE and the FID scores of the generated samples (Gen-FID). The lowest bottleneck dimension is specified by L and is increased as multiples of L.

	L		$2L$		$4L$		$8L$	
	Active	Gen-FID ↓	Active	Gen-FID ↓	Active	Gen-FID ↓	Active	Gen-FID ↓
MNIST ($L = 16$)	12.80 ± 0.40	22.24 ± 0.57	12.60 ± 0.49	22.30 ± 0.33	12.40 ± 0.49	22.31 ± 0.58	12.20 ± 0.40	22.59 ± 0.26
CIFAR10 ($L = 64$)	60.40 ± 0.49	104.03 ± 1.33	105.80 ± 1.33	87.56 ± 1.21	116.40 ± 3.72	86.50 ± 1.43	117.80 ± 15.04	87.88 ± 1.8

Table 3. The FID scores of the generated samples (Gen-FID) using the active dimensions (Active) identified by the MaskAAE and ARD-VAE when trained using the neural network architectures used in the MaskAAE [26].

	MaskAAE		ARD-VAE	
	Active	Gen-FID ↓	Active	Gen-FID ↓
MNIST($L = 16$)	9.80 ± 2.23	$\mathbf{21.35 \pm 0.70}$	8.0 ± 0.0	21.74 ± 0.49
MNIST($l = 32$)	11.40 ± 2.65	21.74 ± 3.77	7.8 ± 0.4	$\mathbf{20.32 \pm 0.96}$
CIFAR($L = 256$)[1]	NA	NA	60.4 ± 10.89	94.86 ± 2.68

[1] The MaskAAE collapsed all the dimensions for the CIFAR10 dataset after 20 epochs. Thus, we skip the evaluation of the MaskAAE for the CIFAR10 dataset.

Table 4. The number of the active dimensions (Active) and FID scores of the generated samples (Gen-FID) of the GECO-L_0-ARM-VAE and ARD-VAE using the neural network architectures used in the GECO-L_0-ARM-VAE [4].

	GECO-L_0-ARM-VAE		ARD-VAE	
	Active	Gen-FID ↓	Active	Gen-FID ↓
MNIST($L = 16$)	9.00 ± 1.41	64.05 ± 5.31	14.60 ± 0.49	$\mathbf{34.45 \pm 0.82}$
MNIST($L = 32$)	18.2 ± 2.93	93.62 ± 22.49	15.60 ± 0.49	$\mathbf{34.15 \pm 1.15}$

Thus, the interesting use case of this methodology is to start with a relatively high-dimensional latent space and let the algorithm/training find the relevant dimensions required to model a particular dataset. For a given latent space size, the hyperparameter β of the ARD-VAE is regulated to achieve an approximate reconstruction loss. Under this scenario, it would be interesting to examine whether the value of β is dependent on the latent space's initial size. To understand this phenomenon, we devised an experiment where we fix β (for a dataset and an autoencoder architecture) and vary the size of the latent space. We evaluate the ARD-VAE for the number of estimated relevant dimensions and the FID scores of the learned data distribution for the MNIST and CIFAR10 datasets. The results in Table 2 indicate that the hyperparameter β of the ARD-VAE is agnostic to the initial size of the latent space, in general. For the CIFAR10 dataset, when the initial dimension is $L = 64$, the ARD-VAE finds almost all the latent axes relevant (≈ 61). This result shows that the initial bottleneck size of $L = 64$ is insufficient to represent the complex CIFAR10 dataset. The number of relevant dimensions (≈ 110) estimated for $L = \{128, 256, 512\}$ corroborates our hypothesis that L should be > 64 to represent the complex CIFAR10 dataset. The insensitivity of β to L is also observed in the performance of the ARD-VAE—$2L$ in Table 1, where we use the same β as in ARD-VAE for all the datasets.

Performance of the ARD-VAE across Neural Network Architectures: For the encoder-decoder architecture used in Table 1, we have observed that the MaskAAE and GECO-L_0-ARM-VAE could not produce comparable results. This is possibly due to the sensitivity to the neural network architecture and related hyperparameter settings. To understand the dependency of these methods on the architectures and optimization settings used by the authors, we ran these methods to reproduce the results reported in the papers associated

Table 5. Disentanglement scores of competing methods trained with 5 different seeds on multiple datasets (higher is better). ACTIVE indicates the number of the latent dimensions used by different methods. The **best** score is in **bold** and the $\underline{\text{second best}}$ score is $\underline{\text{underlined}}$.

METHOD	DSPRITES			3D SHAPES		
	FACTORVAE METRIC ↑	MIG ↑	ACTIVE	FACTORVAE METRIC ↑	MIG ↑	ACTIVE
VAE ($L = 6$)	$\underline{64.78 \pm 8.05}$	0.06 ± 0.02	6.00 ± 0.00	55.85 ± 8.89	0.13 ± 0.15	6.00 ± 0.00
β-TCVAE ($L = 6$)	$\mathbf{75.55 \pm 3.52}$	$\underline{0.20 \pm 0.06}$	6.00 ± 0.00	$\underline{75.51 \pm 12.84}$	$\underline{0.40 \pm 0.22}$	6.00 ± 0.00
DIP-VAE-I ($L = 6$)	59.47 ± 2.76	0.05 ± 0.03	6.00 ± 0.00	51.94 ± 1.91	0.06 ± 0.02	6.00 ± 0.00
DIP-VAE-II ($L = 6$)	60.70 ± 10.97	0.08 ± 0.04	6.00 ± 0.00	63.66 ± 11.26	0.24 ± 0.18	6.00 ± 0.00
RAE ($L = 6$)	64.21 ± 6.69	0.04 ± 0.01	6.00 ± 0.00	53.57 ± 13.14	0.03 ± 0.02	6.00 ± 0.00
ARD-VAE ($L = 10$)	63.37 ± 11.81	$\mathbf{0.22 \pm 0.09}$	5.80 ± 0.40	$\mathbf{79.26 \pm 9.31}$	$\mathbf{0.52 \pm 0.25}$	6.40 ± 0.49

with those methods. In table 3, we report the performance of the MaskAAE and ARD-VAE using the network architectures proposed by the authors in [26]. In table 4, the encoder-decoder architecture reported in [4] is used to train the GECO-L_0-ARM-VAE and ARD-VAE.

In table 3, the FID scores of the samples generated by the MaskAAE for the MNIST dataset [1] are close to the scores of other methods in Table 1. These results demonstrate the efficacy of MaskAAE with the proper choice of neural network architecture, hyperparameters, and optimization strategy. Despite that, we could not train the MaskAAE on the CIFAR10 dataset using the architecture and hyper-parameter settings reported by those authors; for the CIFAR10, the MaskAAE collapsed all the latent dimensions. In contrast, we could successfully train the ARD-VAE on both the MNIST and CIFAR10 by finding a reasonable value for the β (without excessive fine-tuning) that produces good reconstruction.

From the results reported in Table 4, we observe that the relevant dimensions estimated by the GECO-L_0-ARM-VAE are sensitive to the size of the initial latent dimension, L, even with a fixed τ (target reconstruction loss used in [4]). However, the estimated ARD-VAE for a fixed β is indifferent to L and estimates almost the same number of relevant dimensions. Moreover, the ARD-VAE does better in learning the data distribution (i.e., lower FID scores). Relatively higher FID scores using this architecture for the MNIST dataset are likely due to the use of non-trainable upsampling layers in the decoder.

Interpretation of the Relevant Axes Discovered by the ARD-VAE: It is desirable that the relevant axes identified by VAEs encode meaningful information and represent, to some degree, relevant generative factors for a particular dataset. To evaluate this property, we refer to disentanglement analysis, an evaluation approach that penalizes a method when the representations along an axis do not correspond to a generative factor. In addition to the baseline VAE and RAE (which produces good FID scores in Table 1), we consider SOTA VAEs [16, 41], which are explicitly designed to produce disentangled latent representations in our analysis. For all the competing methods, except the ARD-VAE, the size of the latent space ($L = 6$) is set to the number of the *known latent factors* of

[1] The FID scores found in this study, using an online version of the authors' code, are higher than the those reported in [26].

the DSprites [25] and Shapes3D [2] dataset — a significant advantage, which would not be available in most applications. We follow the encoder-decoder architectures and training strategies used in [19] for all the methods.

From the analysis reported in Table 5, we observe that the ARD-VAE produces the best disentangled representation for the Shapes3D dataset when evaluated using both metrics. The precise identification of the true generative factors by the ARD-VAE for this dataset helps achieve such scores. For the DSprites dataset, the ARD-VAE accurately estimates the number of hidden explanatory factors. Moreover, the best MIG score indicates that the representations learned by the relevant axes are not entangled relative to other methods. The FactorVAE metric score does not penalize a method if a ground truth factor is represented by multiple latent axes, unlike the MIG. Thus, we observe a low MIG score in many scenarios for methods with a relatively high score for the FactorVAE metric, such as the VAE, DIP-VAE, and RAE. Though the RAE does a reasonable job of learning the data distribution (in Table 1), the learned latent representations are not disentangled, as indicated by low scores. The performance of the DIP-VAE is often poorer than the baseline VAE despite being designed to learn disentangled representations. Among the different variants of the VAE, the β-TCVAE produces results comparable to the ARD-VAE.

5 Conclusion

In this work, we present a Bayesian approach to determine the relevant axes in the VAE using a hierarchical prior without deviating from and adding any regularization loss to the ELBO formulation. Unlike other relevancy detection methods for DLVMs [26, 4], the ARD-VAE is resilient to the choice of neural network architectures. The performance of the ARD-VAE is impervious to the size of the latent space and can scale to high dimensions (≈ 500). The ARD-VAE effectively models the distribution of multiple benchmark datasets, and the identified latent dimensions represent generative factors explaining the variability in the data.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, 8th edn. (2009)
2. Burgess, C., Kim, H.: 3d shapes dataset. https://github.com/deepmind/3d-shapes/ (2018)
3. Dai, B., Wipf, D.: Diagnosing and enhancing vae models. International Conference on Learning Representations (2019)
4. De Boom, C., Wauthier, S., Verbelen, T., Dhoedt, B.: Dynamic narrowing of vae bottlenecks using geco and l0 regularization. In: International Joint Conference on Neural Networks (IJCNN) (2021)
5. Ghosh, P., Sajjadi, M.S.M., Vergari, A., Black, M., Schölkopf, B.: From variational to deterministic autoencoders. In: International Conference on Learning Representations (2020)

6. Gretton, A., Borgwardt, K.M., Rasch, M.J., Scholkopf, B., Smola, A.: A kernel two-sample test. Journal of Machine Learning Research **13**, 723–773 (2012)
7. Gómez-Bombarelli, R., Wei, J.N., Duvenaud, D., Hernández-Lobato, J.M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T.D., Adams, R.P., Aspuru-Guzik, A.: Automatic chemical design using a data-driven continuous representation of molecules. ACS Central Science **4**(2), 268–276 (2018). https://doi.org/10.1021/acscentsci.7b00572
8. Han, J., Ren, Y., Ding, J., Yan3, K., Xia, G.S.: Few-shot object detection via variational feature aggregation. In: AAAI Conference on Artificial Intelligence (2023)
9. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Conference on Neural Information Processing Systems (2017)
10. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: β-vae: Learning basic visual concepts with a constrained variational framework. In: International Conference on Learning Representations (2017)
11. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems (NeurIPS) (2020)
12. Hoffman, M.D., Johnson, M.J.: Elbo surgery: yet another way to carve up the variational evidence lower bound. In: NIPS Workshop: Advances in Approximate Bayesian Inference (2016)
13. Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., Aila, T.: Alias-free generative adversarial networks. In: Conference on Neural Information Processing Systems (2021)
14. Kim, H., Mnih, A.: Disentangling by factorising. In: International Conference on Machine Learning (2018)
15. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. International Conference on Learning Representations (2014)
16. Kumar, A., Sattigeri, P., Balakrishnan, A.: Variational inference of disentangled latent concepts from unlabeled observations. In: International Conference on Learning Representations (2018)
17. Li, Y., Ji, S.: L0-arm: Network sparsification via stochastic binary optimization. In: European Conference on Machine Learning (2019)
18. Li, Y., Pan, Q., Wang, S., Peng, H., Yang, T., Cambria, E.: Disentangled variational auto-encoder for semi-supervised learning. Information Sciences **482**, 73–85 (2019). https://doi.org/https://doi.org/10.1016/j.ins.2018.12.057
19. Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., Bachem, O.: Challenging common assumptions in the unsupervised learning of disentangled representations. In: International Conference on Machine Learning (2019)
20. Lucasz, J., Tuckery, G., Grossez, R., Norouziy, M.: Don't blame the elbo! a linear vae perspective on posterior collapse. In: Conference on Neural Information Processing Systems (2019)
21. Lucasz, J., Tuckery, G., Grossez, R., Norouziy, M.: Understanding posterior collapse in generative latent variable models. In: International Conference on Learning Representations (2019)
22. M. Bishop, C.: Variational principal components. In: ICANN (1999)
23. Ma, P., Hu, X.: A variational autoencoder with deep embedding model for generalized zero-shot learning. In: AAAI Conference on Artificial Intelligence (2020)
24. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders. In: International Conference on Learning Representations (2016)
25. Matthey, L., Higgins, I., Hassabis, D., Lerchner, A.: dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/ (2017)

26. Mondal, A.K., Chowdhury, S.P., Jayendran, A., Singla, P., Asnani, H., AP, P.: Maskaae: Latent space optimization for adversarial auto-encoders. In: Uncertainty in Artificial Intelligence (UAI) (2020)
27. Murphy, K.P.: Conjugate bayesian analysis of the gaussian distribution (2007)
28. Neal, R.M.: Bayesian learning for neural networks (1996)
29. Ochiai, T., Inukai, T., Akiyama, M., Furui, K., Ohue, M., Matsumori, N., Inuki, S., Uesugi, M., Sunazuka, T., Kikuchi, K., Kakeya, H., Sakakibara, Y.: Variational autoencoder-based chemical latent space for large molecular structures with 3d complexity. Communications Chemistry **6**(249) (2023). https://doi.org/10.1038/s42004-023-01054-6
30. Oord, A.v.d., Vinyals, O., Kavukcuoglu, K.: Neural discrete representation learning. In: Conference on Neural Information Processing Systems (2017)
31. Palma, A., Rybakov, S., Hetzel, L., Theis, F.: Modelling single-cell rna-seq trajectories on a flat statistical manifold. In: NeurIPS AI for Science Workshop (2023)
32. Razavi, A., Oord, A.v.d., Poole, B., Vinyals, O.: Preventing posterior collapse with δ-vaes. In: International Conference on Learning Representations (2019)
33. Rezende, D.J., Viola, F.: Generalized elbo with constrained optimization, geco. In: Third workshop on Bayesian Deep Learning (NeurIPS) (2018)
34. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: International Conference on Machine Learning. pp. 1278–1286 (2014)
35. Rosca, M., Lakshminarayanan, B., Mohamed, S.: Distribution matching in variational inference. arxiv (2018), preprint at https://arxiv.org/abs/1802.06847
36. Saha, S., Elhabian, S., Whitaker, R.: Gens: generative encoding networks. Machine Learning **111**, 4003–4038 (2022)
37. Saha, S., Joshi, S., Whitaker, R.: Matching aggregate posteriors in the variational autoencoder (2023), preprint at https://arxiv.org/pdf/2311.07693.pdf
38. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Conference on Neural Information Processing Systems (2016)
39. Schonfeld, E., Ebrahimi, S., Sinha, S., Darrell, T., Akata, Z.: Generalized zero- and few-shot learning via aligned variational autoencoders. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
40. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: International Conference on Learning Representations (2021)
41. T. Q. Chen, R., Li, X., Grosse, R., Duvenaud, D.: Isolating sources of disentanglement in vaes. In: Conference on Neural Information Processing Systems (2019)
42. Tipping, M.E.: Sparse bayesian learning and the relevance vector machine. Journal of Machine Learning Research **1**, 211–244 (2001)
43. Tolstikhin, I., Bousquet, O., Gelly, S., Schoelköpf, B.: Wasserstein auto-encoders. In: International Conference on Learning Representations (2018)
44. Tonolini, F., Aletras, N., Jiao, Y., Kazai, G.: Nestedvae: Isolating common factors via weak supervision. In: IEEE Conference on Computer Vision and Pattern Recognition (2020)
45. Vowels, M.J., Camgoz, N.C., Bowden, R.: Robust weak supervision with variational auto-encoders. In: International Conference on Machine Learning (2023)

Appendix to the ARD-VAE: A Statistical Formulation to Find the Relevant Latent Dimensions of Variational Autoencoders

Surojit Saha[1], Sarang Joshi[1], and Ross Whitaker[1]

Scientific Computing and Imaging Institute, Kahlert School of Computing, The University of Utah, Salt Lake City, USA
{surojit.saha,sarang.josh,ross.whitaker}@utah.edu

1 Algorithms

Algorithm 1 Produce $D_{\mathbf{z}}$ using the training subset $\mathcal{X}_{\boldsymbol{\alpha}}$

```
Input: $\mathcal{X}_{\boldsymbol{\alpha}}$
Output: $D_{\mathbf{z}}$
Initialize $D_{\mathbf{z}} \leftarrow \phi$.
for $\mathbf{x}' \in \mathcal{X}_{\boldsymbol{\alpha}}$ do
    $\boldsymbol{\mu}_{\mathbf{x}'}, \boldsymbol{\sigma}^2_{\mathbf{x}'} \leftarrow \mathbf{E}_{\phi}(\mathbf{x}')$
    $\mathbf{z}' \leftarrow \boldsymbol{\mu}_{\mathbf{x}'} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}_{\mathbf{x}'}$
    $D_{\mathbf{z}} \leftarrow D_{\mathbf{z}} \cup \mathbf{z}'$
end for
```

Table 1. Encoder and decoder architectures used by all the methods for the MNIST, CelebA and CIFAR10 datasets [9, 3].

	MNIST	CelebA	CIFAR10
Encoder:	$\mathbf{x} \in \mathbb{R}^{32\times32\times1}$	$\mathbf{x} \in \mathbb{R}^{64\times64\times3}$	$\mathbf{x} \in \mathbb{R}^{32\times32\times3}$
	Conv64 → BN → ReLU	Conv64 → BN → ReLU	Conv128 → BN → ReLU
	Conv128 → BN → ReLU	Conv128 → BN → ReLU	Conv256 → BN → ReLU
	Conv256 → BN → ReLU	Conv256 → BN → ReLU	Conv512 → BN → ReLU
	Conv512 → BN → ReLU	Conv512 → BN → ReLU	Conv1024 → BN → ReLU
	Flatten$_{2\times2\times512}$ → FC$_{k\times16}$ → None	Flatten$_{4\times4\times512}$ → FC$_{k\times64}$ → None	Flatten$_{2\times2\times1024}$ → FC$_{k\times128}$ → None
Decoder:	$\mathbf{z} \in \mathbb{R}^{16}$ → FC$_{2\times2\times512}$	$\mathbf{z} \in \mathbb{R}^{64}$ → FC$_{8\times8\times512}$	$\mathbf{z} \in \mathbb{R}^{128}$ → FC$_{8\times8\times1024}$
	TransConv256 → BN → ReLU	TransConv256 → BN → ReLU	TransConv512 → BN → ReLU
	TransConv128 → BN → ReLU	TransConv128 → BN → ReLU	TransConv256 → BN → ReLU
	TransConv64 → BN → ReLU	TransConv64 → BN → ReLU	TransConv3 → Sigmoid
	TransConv1 → Sigmoid	TransConv3 → Tanh	

2 Experimental Settings

In the neural network architectures reported in Table 1, 2, Conv$n$ and TransConv$n$ define convolution and transpose convolution operation, respectively, with n filters in the output. We have used 4×4 filters for all the datasets. The transpose

Algorithm 2 : ARD-VAE training

```
Input: Training samples $\mathcal{X}$, Number of epochs the $D_{\mathbf{z}}$ should lag $u_{D_{\mathbf{z}}}$, Scaling factor $\beta$
Output: Encoder and decoder parameters, $\phi$ and $\theta$
Split $\mathcal{X}$ into training, $\mathcal{X}_{train}$, and validation data, $\mathcal{X}_{val}$
Choose a subset $\mathcal{X}_{\boldsymbol{\alpha}}$ at random from $\mathcal{X}_{train}$
Initialize SGD samples $\mathcal{X}_{sgd} = \mathcal{X}_{train} - \mathcal{X}_{\boldsymbol{\alpha}}$
Initialize $\phi$ and $\theta$
Initialize epoch index $e \leftarrow 0$
for number of epochs do
  if $e \mod u_{D_{\mathbf{z}}}$ then
    Choose a new subset of the training data $\mathcal{X}_{\boldsymbol{\alpha}}$
    Update SGD samples, $\mathcal{X}_{sgd} = \mathcal{X}_{train} - \mathcal{X}_{\boldsymbol{\alpha}}$
    Update $D_{\mathbf{z}}$ with the samples produced by the algorithm 1 using the update $\mathcal{X}_{\boldsymbol{\alpha}}$
    Update $\boldsymbol{b}_L$, using the latest $D_{\mathbf{z}}$
  end if
  for number of minibatch updates do
    Sample a minibatch from $\mathcal{X}_{sgd}$
    Update $\phi$ and $\theta$ by optimizing the ARD-VAE objective function
  end for
  $e \leftarrow e + 1$
  Shuffle $\mathcal{X}_{sgd}$
end for
```

convolution filters use a stride size of 2 except for the last layer of the decoders used in the CelebA and CIFAR10 datasets. We represent the fully connected layers as $\text{FC}_{k\times n}$ with $k \times n$ nodes, where $k = 1$ for all the methods, except the VAE and β-TCVAE that use $k = 2$. Activation functions used in the networks are ReLU (ReLU), Leaky ReLU (LReLU), sigmoid (Sigmoid), and hyperbolic tangent (Tanh). Input is in the range $[0, 1]$ for all the datasets except CelebA, for which the input is mapped to the range $[-1, 1]$. The encoder-decoder architectures of the MaskAAE [8] and GECO-L_0-ARM-VAE [2] are obtained from the respective papers.

We use the Adam optimizer in all experiments (learning rate set to $5e-04$) with a learning rate scheduler (ReduceLROnPlateau) that reduces the learning rate by 0.5 if the validation loss does not improve for a maximum of 10 epochs except the MaskAAE and GECO-L_0-ARM-VAE due to their sensitivity in optimization of the trainable parameters. Moreover, the MaskAAE uses a specific training recipe. All the methods are trained for 50, 50, and 100 epochs for the MNIST, CelebA, and CIFAR10 datasets, respectively, with a few exceptions for the MNIST dataset. The VAE, β-TCVAE, GECO-L_0-ARM-VAE, and MaskAEE are trained for 100 epochs for the MNIST dataset as it improved the model performance. In the disentanglement analysis, all the methods are trained for 35 and 60 epochs [6] for the DSprites and 3D Shapes datasets, respectively.

We use a batch size of 100 for training all the methods, except the MaskAAE, which is trained using a batch size of 64 (to maintain consistency with their

Table 2. Encoder and decoder architectures used by all the methods for the DSprites and 3D Shapes datasets [4, 6].

	DSprites	3D Shapes
Encoder:	$\mathbf{x} \in \mathbb{R}^{64\times64\times1}$	$\mathbf{x} \in \mathbb{R}^{64\times64\times3}$
	Conv32 $\to$ ReLU	Conv32 $\to$ ReLU
	Conv32 $\to$ ReLU	Conv32 $\to$ ReLU
	Conv64 $\to$ ReLU	Conv64 $\to$ ReLU
	Conv64 $\to$ ReLU	Conv64 $\to$ ReLU
	Flatten$_{4\times4\times64}$ $\to$ FC$_{k\times6}$ $\to$ None	Flatten$_{4\times4\times64}$ $\to$ FC$_{k\times6}$ $\to$ None
Decoder:	$\mathbf{z} \in \mathbb{R}^{6} \to$ FC$_{4\times4\times64}$	$\mathbf{z} \in \mathbb{R}^{6} \to$ FC$_{4\times4\times64}$
	TransConv64 $\to$ ReLU	TransConv64 $\to$ ReLU
	TransConv32 $\to$ ReLU	TransConv32 $\to$ ReLU
	TransConv32 $\to$ ReLU	TransConv32 $\to$ ReLU
	TransConv1 $\to$ None	TransConv3 $\to$ Sigmoid

implementation). All the hyperparameters of the MaskAAE are set according to the GitHub repo in https://github.com/arnabkmondal/MaskAAE for all the datasets. We have tuned the hyperparameters of the MaskAAE over a series (i.e., dozens) of experiments, and no better results could be obtained. For the GECO-L_0-ARM-VAE, we have used the code shared by the authors [2]. Only the target reconstruction loss (τ) in GECO-L_0-ARM-VAE was the hyperparameter in our analysis. For the DIP-VAE-I and DIP-VAE-II, we have followed the suggested hyperparameters in the paper [5]. For the β-TCVAE, we have empirically determined the strength of the regularization loss for different data sets as we could not find them in the literature. We set $\beta = 2$ for the MNIST, CelebA, and CIFAR10 data sets as higher β resulted in poor reconstruction. Table 3 reports the specific hyperparameters for different methods. The $u_{D_\mathbf{z}}$ in the algorithm 2 is set to 1 and $\mid \mathcal{X}_{\boldsymbol{\alpha}} \mid = 10K$ for all the datasets studied in this work.

3 Results

Ablation Study on the Size of $\mathcal{X}_{\boldsymbol{\alpha}}$: In this analysis, we study the effect of the number of samples in $\mathcal{X}_{\boldsymbol{\alpha}}$ on the performance of the ARD-VAE in

Table 3. Optimization settings for different methods.

Method	Parameters	MNIST	CelebA	CIFAR10	DSprites	3D Shapes
β-TCVAE	β:	2	2	2	5	5
DIP-VAE-I	$(\lambda_{od}, \lambda_d)$:	NA	NA	NA	$(10, 100)$	$(10, 100)$
DIP-VAE-II	$(\lambda_{od}, \lambda_d)$:	NA	NA	NA	$(10, 10)$	$(10, 10)$
WAE	Recons-scalar:	0.05	0.05	0.05	NA	NA
WAE	β:	10	100	100	NA	NA
RAE	β:	$1e-04$	$1e-04$	$1e-03$	$1e-04$	$1e-04$
RAE	Dec-L2-reg:	$1e-07$	$1e-07$	$1e-06$	$1e-06$	$1e-06$
GECO-L_0-ARM-VAE	τ:	10.0	300.0	25.0	NA	NA
ARD-VAE	β:	0.5	3.0	0.05	5.0	5.0

Table 4. The number of active dimensions (ACTIVE) discovered by the ARD-VAE and the FID scores of the generated samples (GEN-FID).

	$\mid \mathcal{X}_{\boldsymbol{\alpha}} \mid = 2K$		$\mid \mathcal{X}_{\boldsymbol{\alpha}} \mid = 5K$		$\mid \mathcal{X}_{\boldsymbol{\alpha}} \mid = 10K$		$\mid \mathcal{X}_{\boldsymbol{\alpha}} \mid = 20K$	
	ACTIVE	GEN-FID ↓	ACTIVE	GEN-FID ↓	ACTIVE	GEN-FID ↓	ACTIVE	GEN-FID ↓
MNIST ($L = 32$)	13	21.83	12	22.04	13	22.13	13	24.12
CIFAR10 ($L = 256$)	116	85.15	114	85.65	112	85.99	112	86.83

terms of the number of relevant/active dimensions identified and FID score of the generated samples. Different settings of $\mid \mathcal{X}_{\boldsymbol{\alpha}} \mid$ considered in this study are $\mid \mathcal{X}_{\boldsymbol{\alpha}} \mid = \{2K, 5K, 10K, 20K\}$. The ARD-VAE is trained on the MNIST and CIFAR10 datasets with $L = 32$ and $L = 256$, respectively. We have used the neural network architecture in Table 1 and followed the optimization strategies discussed in the section 2. We use the hyperparameter β as reported in Table 3. Compared to other results reported in the results section, the network parameters are initialized using a single seed, as we did not observe much variation in the performance of the ARD-VAE with different random seeds.

From the results reported in Table 4, we observe negligible variation in the number of relevant dimensions estimated by the ARD-VAE with the size of $\mathcal{X}_{\boldsymbol{\alpha}}$ and there is almost no variation in the FID scores of the generated samples for both datasets. Thus, the performance of the ARD-VAE is not affected by the number of samples in $\mathcal{X}_{\boldsymbol{\alpha}}$.

The ARD-VAE is trained in an unsupervised framework having access to a large amount of data, and it is good to have a representative set that captures the variations in the dataset. Thus, we choose $\mid \mathcal{X}_{\boldsymbol{\alpha}} \mid = 10K$ as a general setting for any random dataset that uses the ARD-VAE.

Relevant axes for the VAE using the Jacobiab: In this experiment we estimate the importance of the latent axes of a trained VAE using 1.

$$\mathbf{w}_{\hat{\boldsymbol{\sigma}}} = \sum_{k=1}^{D} \frac{1}{N} \sum_{i=1}^{N} \|\mathbb{J}_i\|^2, \tag{1}$$

$$\text{where } \mathbb{J} = \left[\frac{\partial \hat{\mathbf{x}}}{\partial \mu_1} \cdots \frac{\partial \hat{\mathbf{x}}}{\partial \mu_L} \right] \in \mathbb{R}^{D \times L} \text{ is the Jacobian matrix.}$$

Table 5. The FID scores of the reconstructed (REC-FID) and generated samples (GEN-FID) along with the number of latent dimensions (ACTIVE) used by the competing methods. The **best** FID score is in **bold** and the <u>second best</u> score is <u>underlined</u>.

	MNIST ($L = 16$)			CelebA ($L = 64$)			CIFAR10 ($L = 128$)		
	ACTIVE	REC-FID ↓	GEN-FID ↓	ACTIVE	REC-FID ↓	GEN-FID ↓	ACTIVE	REC-FID ↓	GEN-FID ↓
VAE	10.20 ± 0.40	27.38 ± 0.26	29.33 ± 0.41	55.20 ± 0.40	44.10 ± 0.37	$\underline{50.79 \pm 0.64}$	26.40 ± 0.49	143.83 ± 2.54	155.13 ± 1.45
β-TCVAE	7.40 ± 0.49	49.21 ± 0.40	51.35 ± 0.95	51.40 ± 0.49	46.21 ± 0.51	51.61 ± 0.76	16.20 ± 0.40	178.47 ± 3.10	186.03 ± 1.82
RAE	16.00 ± 0.00	$\mathbf{7.98 \pm 0.22}$	$\mathbf{18.85 \pm 0.43}$	63.00 ± 0.00	$\mathbf{40.06 \pm 0.68}$	$\mathbf{49.06 \pm 1.16}$	125.00 ± 0.00	$\mathbf{54.71 \pm 0.58}$	92.66 ± 1.61
WAE	16.00 ± 0.00	9.86 ± 0.19	24.78 ± 0.77	63.00 ± 0.00	$\underline{41.25 \pm 0.22}$	61.18 ± 1.60	127.00 ± 0.00	67.00 ± 1.07	136.79 ± 1.35
GECO-L_0-ARM-VAE	10.00 ± 1.10	$\underline{8.76 \pm 1.13}$	304.75 ± 64.29	35.60 ± 3.07	42.28 ± 0.8	294.97 ± 28.45	68.00 ± 2.97	92.33 ± 2.83	320.75 ± 45.09
MaskAAE	9.80 ± 1.60	143.80 ± 47.73	144.92 ± 16.80	5.4 ± 0.80	343.57 ± 34.00	333.40 ± 10.91	3.80 ± 0.40	394.26 ± 34.27	298.30 ± 8.44
ARD-VAE	12.80 ± 0.40	17.50 ± 0.23	$\underline{22.24 \pm 0.57}$	45.20 ± 0.40	45.65 ± 0.25	52.35 ± 0.52	105.80 ± 1.33	$\underline{59.68 \pm 2.21}$	$\underline{87.56 \pm 1.21}$
ARD-VAE—$2L$[1]	12.60 ± 0.49	17.53 ± 0.10	22.30 ± 0.33	48.60 ± 0.80	46.72 ± 0.34	53.86 ± 0.68	116.40 ± 3.72	59.89 ± 7.58	$\mathbf{86.50 \pm 1.43}$

[1] The initial dimension of the ARD-VAE—$2L$ is two times L (the initial dimension used by all the methods).

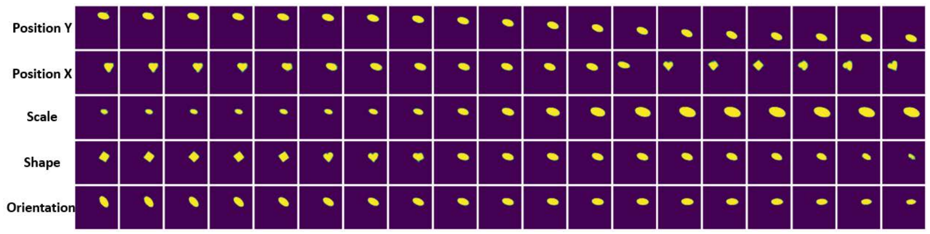

Fig. 1. Latent traversal of the DSprites data set [7] in the range $[-3\sigma, 3\sigma]$ using the relevant axes discovered by the ARD-VAE. The latent factors are mentioned in the left column. All latent factors are represented by independent latent axes with slight entanglement of **Shape** and **Position Y**. The MIG score for this model is **0.35**

To get a reliable estimate of the weight vector $\mathbf{w}_{\hat{\boldsymbol{\sigma}}}$ using 1, we compute the Jacobian for multiple data samples, which is typically the size of a minibatch (100) in this work.

This estimate is used to determine the relevant axes for the VAE, β-TCVAE, RAE, and WAE and the selected axes are used for model evaluations as shown in Table 5. The relevant dimensions of the VAE and β-TCVAE are less than the initial dimension L for the MNIST and CelebA dataset with comparable estimates of the FID scores. However, similar to the GECO-L_0-ARM-VAE and MaskAAE, majority of the dimensions collapses for the complex CIFAR dataset. This experiment demonstrates the robustness of the proposed ARD-VAE across different evaluation scenarios. There is no change in the number of active latent dimensions from L for the RAE and WAE as both methods match aggregate posterior distributions, unlike VAEs.

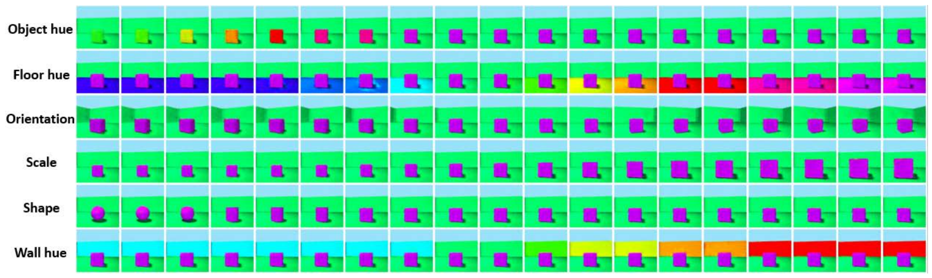

Fig. 2. Latent traversal of the 3D Shapes data set [1] in the range $[-3\sigma, 3\sigma]$ using the relevant axes discovered by the ARD-VAE. The latent factors are mentioned in the left column. All latent factors are represented by independent latent axes with almost no overlap between them. The MIG score for this model is **0.84**

References

1. Burgess, C., Kim, H.: 3d shapes dataset. https://github.com/deepmind/3d-shapes/ (2018)
2. De Boom, C., Wauthier, S., Verbelen, T., Dhoedt, B.: Dynamic narrowing of vae bottlenecks using geco and l0 regularization. In: International Joint Conference on Neural Networks (IJCNN) (2021)
3. Ghosh, P., Sajjadi, M.S.M., Vergari, A., Black, M., Scholköpf, B.: From variational to deterministic autoencoders. In: International Conference on Learning Representations (2020)
4. Kim, H., Mnih, A.: Disentangling by factorising. In: International Conference on Machine Learning (2018)
5. Kumar, A., Sattigeri, P., Balakrishnan, A.: Variational inference of disentangled latent concepts from unlabeled observations. In: International Conference on Learning Representations (2018)
6. Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., Bachem, O.: Challenging common assumptions in the unsupervised learning of disentangled representations. In: International Conference on Machine Learning (2019)
7. Matthey, L., Higgins, I., Hassabis, D., Lerchner, A.: dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/ (2017)
8. Mondal, A.K., Chowdhury, S.P., Jayendran, A., Singla, P., Asnani, H., AP, P.: Maskaae: Latent space optimization for adversarial auto-encoders. In: Uncertainty in Artificial Intelligence (UAI) (2020)
9. Tolstikhin, I., Bousquet, O., Gelly, S., Schoelköpf, B.: Wasserstein auto-encoders. In: International Conference on Learning Representations (2018)