
Article:
Ye, Fei and Bors, Adrian Gheorghe orcid.org/0000-0001-7838-0021 (2022) Deep Mixture Generative Autoencoders. IEEE Transactions on Neural Networks and Learning Systems. pp. 5789-5803. ISSN 2162-237X

https://doi.org/10.1109/TNNLS.2021.3071401

Deep Mixture Generative Autoencoders

Fei Ye and Adrian G. Bors, *Senior Member, IEEE*
Department of Computer Science, University of York, York YO10 5GH, UK
E-mail: {fy689, adrian.bors}@york.ac.uk

***Abstract*—Variational autoencoders (VAEs) are one of the most popular unsupervised generative models which rely on learning latent representations of data. In this paper, we extend the classical concept of Gaussian mixtures into the deep variational framework by proposing a mixture of VAEs (MVAE). Each component in the MVAE model is implemented by a variational encoder and has an associated sub-decoder. The separation between the latent spaces modelled by different encoders is enforced using the d-variable Hilbert-Schmidt Independence Criterion (dHSIC) criterion. Each component would capture different data variational features. We also propose a mechanism for finding the appropriate number of VAE components for a given task, leading to an optimal architecture. The differentiable categorical Gumbel-Softmax distribution is used in order to generate dropout masking parameters within the end-to-end backpropagation training framework. Extensive experiments show that the proposed MAVE model learns a rich latent data representation and is able to discover additional underlying data factors.**

***Index Terms*—Mixtures of Variational Autoencoders, Generative deep learning, Representation learning, Optimal number of components in mixtures.**

I. Introduction

Research in deep learning focused initially on addressing supervised classification problems, where training data are labelled. Meanwhile, unsupervised learning aims to find the intrinsic structure of the data without assuming any *a priori* knowledge. A classic model for data representation, rooted in statistical inference and proving excellent statistical approximation properties is the Gaussian mixture model (GMM). GMMs have been used to define Radial Basis Functions (RBF) networks by adding a second layer of linear processing units for supervised classification [1], [2], [3], [4]. RBF networks have been shown to have universal approximation properties [5], [6], [7]. The variational GMM model addresses the uncertainty in the estimation of the mixture model parameters by defining a lower bound on the marginal log-likelihood, replacing the true posterior distribution with a variational approximation, [8], [9], [10].

Lately, generative deep models have gained an increasing attention from the research community. The Variational Autoencoder (VAE) [11] consists of two convolution neural network (CNN) components: the encoder and the decoder. The decoder is used to approximate the conditional distribution $p(\mathbf{x}|\mathbf{z})$ for reconstructing the data $\mathbf{x}$ from the estimated latent space $\mathbf{z}$. Meanwhile, the latent space is modelled by a variational distribution $q(\mathbf{z}|\mathbf{x})$, estimated by the encoder, which aims to match the prior distribution during the training. In relation to classical clustering methods, VAEs have considered GMMs as prior distributions [12] and they have been used for deep spectral clustering [13], [14]. Another generative model is the Generative Adversarial Network (GAN) [15]. A GAN is also composed of two networks: the generator and the discriminator, playing a min-max game. The generator aims to generate realistic data in order to fool the discriminator whose task is to distinguish the real data from fake. GANs generate better quality images with higher contrast than VAE, but they lack an appropriate inference mechanism. Mixed GAN and VAE architectures have been enabled with representation learning capabilities [16], [17].

Let us consider the generative ability of deep learning structures based on representations inferred from the latent space. One limitation of the VAE is the fact that its latent space is fixed after the training, with the data' posterior probability represented by the parameters characterizing a simple Gaussian distribution. A single VAE model has a low-dimensional latent space and can only capture a few underlying variation factors of the data. For instance, a VAE with a two-dimensional stochastic latent variable vector, can only learn two meaningful representations. Furthermore, the posterior in VAEs is of a rather simple form, and not able to capture complex structures behind data. Other problems when using single VAEs are overfitting and over-regularisation in data representation, [12]. In this work we address these problems by developing a novel Mixture of Variational Autoencoders (MVAE), which enjoys many more advantages than existing models, including memory efficiency and performance. The contributions of this research study are summarized as follows:

1) We propose an efficient network architecture design for the VAE mixture model. Unlike in other mixture models using deep networks for the decoder [12], [18], MVAE implements the decoder of each component as a simple non-linear mapping requiring few parameters and low computational costs. A Dirichlet sampling process is used for assigning mixing parameters for each component. Unlike using a fixed symmetric Dirichlet distribution as in other approaches [19], the proposed sampling process finds automatically the optimal weight of each component.
2) We propose a measure for enforcing the separability of the latent space representation associated with each VAE component, by using the d-variable Hilbert-Schmidt Information criterion (dHSIC) [20]. The dHSIC measure is always positive and can be easily integrated in the objective function used for training MVAE deep learning model. The dHSIC measure can also significantly relieve the over-regularization problem which affects other VAE based methods [11], [12], [18].

3) We propose a new way for selecting the number of components, during the training. A dropout mechanism is enabled by sampling from either a Gumbel-softmax or a Gaussian distribution. We define a loss function which includes the dropout rate estimation, controlling how the number of components is decided and ensuring the end-to-end training for MVAE.
4) We show through extensive experimentation, that the proposed mixture model learns several distinct clusters in the latent space, which enables a rich data representation, benefiting many down-stream tasks. The proposed model provides a competitive performance when compared with the state of the art VAE frameworks.

The rest of paper is organized as follows. Related research is discussed in Section II. The proposed model and its objective function are described in detail in Section III, while in Section IV we present the architecture of the model and its training algorithm. The experimental results are reported Section V and the conclusions are drawn in Section VI.

II. Related work

A. Probabilistic mixture model

The Gaussian mixture model (GMM) is an unsupervised learning model which can be trained using the Expectation Maximization (EM) algorithm, [21]. Meanwhile, a variational model can be used to define a lower bound for the marginal log-likelihood [8], [9]. The Variational Expectation-Maximization (EM) algorithm was used to find a set of hyperparameters for the variational GMM model [10]. Gaussian distributions are used to model the distribution of the means of each Gaussian component of the mixture, Wishard distributions are used for their corresponding covariance matrices, and Dirichlet for the mixing parameters.

GMMs have been embedded into Radial Basis Functions (RBF) networks by adding a second layer of linear processing units [1], [2], [3], [4], [5] for supervised classification. RBF networks have been shown to have universal approximation properties [5], [6], [7] and were trained using backpropagation [1], robust clustering [2], or orthogonal least squares [3].

B. The variational autoencoder (VAE)

A variational autoencoder (VAE) [11] is a probabilistic model which learns a compressed data representation. The VAE model is made up of two complementary networks: encoder and decoder. VAEs have probabilistic inference mechanisms that can capture data' characteristics. Let $\mathbf{x}$ be a data sample vector and $\mathbf{z}$ a vector of stochastic latent variables. While the encoder maximizes $p_\eta(\mathbf{z}|\mathbf{x})$, the decoder in VAE implements a distribution $q_\theta(\mathbf{x}|\mathbf{z})$, called the variational posterior, where η and θ represent the parameters of the Convolution Neural Networks (CNNs) implementing the encoder and decoder, respectively. The evidence lower bound (ELBO) on the marginal log-likelihood in VAEs is defined as :

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim\mathcal{X}}[\log p(\mathbf{x})] \geq & \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}[\mathbb{E}_{q_\theta(\mathbf{z}|\mathbf{x})}[\log p_\eta(\mathbf{x}|\mathbf{z}) \\ & + \log p(\mathbf{z}) - \log q_\theta(\mathbf{z}|\mathbf{x})]]\end{aligned} \tag{1}$$

where $\mathcal{X}$ is the empirical distribution characterizing the data and $p_\eta(\mathbf{x}|\mathbf{z})$ is a generative model implemented by the decoder of parameters η, while $p(\mathbf{z})$ is the prior distribution of the latent space, usually a Gaussian distribution with the identity matrix as its covariance. The re-parameterization trick [22], [23] is used in order to allow for the gradients to be efficiently transferred from the VAE's encoder to decoder when maximizing ELBO from (1). The first term from the right side of (1) can be calculated using the reconstruction error, while the others can be seen as the Kullback-Leibler (KL) divergence between the variational posterior and the prior distributions, encouraging the variational posterior to match the prior distribution.

C. Representation learning in VAEs

VAEs provide an efficient inference mechanism for estimating informative latent variables corresponding to the given data. There are two measures to evaluate the quality of approximate inference in VAEs, [22]. One consists in the ability of the variational posterior to match the true posterior. The second measure represents the capacity of the inference network to yield good variational parameters. One way for improving the quality of the approximate inference consists in increasing the expressiveness of the approximate posteriors. For instance, the normalizing flow [24], [25], [26], [27] was used in VAEs in order to make the approximate posteriors more expressive. Another way is by introducing auxiliary variables, such as the Hierarchical Variational Models [14], [28], the Hamiltonian variational inference [29], or using two stochastic layers [30]. Importance sampling [31], [32] is also used in VAEs for improving the quality of the inference.

One of the problems displayed by VAEs is that of posterior collapse when the variational distribution closely matches the uninformative prior for a subset of latent variables. InfoVAE [33] aims to address the posterior collapse problem by using a mutual information term in the objective function. Ma *et al.* [34], introduced a new regularization term in the VAE objective, called the mutual posterior-divergence, used to measure the diversity of posteriors. Zhang *et al.* [35], proposed a new form of VAE, namely Wasserstein-Wasserstein Auto-Encoders, which replaces the KL divergence term with a new regularization term measuring the squared Wasserstein-2 distance between the prior and the aggregated posterior.

D. Deep mixture models using VAEs

Some recent efforts propose to use mixture models based on VAEs for learning complex structures behind the data. Kurle *et al* [18] introduced a mixture model, called Multi-Source Neural Variational Inference (MSVI) aiming to capture probabilistic characteristics from multiple sources. However, MSVI relies on multiple source domains and would not encourage disentanglement between encoding distributions. Dilokthanakul *et al* [12], [18] develops a model considering a GMM as prior distribution for unsupervised clustering tasks. This model still suffers from over-regularisation. A mixture of VAEs defined in a fixed architecture was proposed in [36].

In summary, existing VAE models do not learn multiple separate representations of data, where each representation could capture rich statistical characteristics in different ways. One advantage for the proposed Mixture of VAEs model over other mixture models [12], [18], is that it can learn multiple disentangled representations by using an efficient network architecture which requires a lower computational complexity and a reduced number of parameters. Furthermore, instead of using a symmetric Dirichlet distribution [12] for sampling the mixing parameter, we model the sampling process by using inference models, leading to optimal configuration for the mixing parameters. We also propose a novel dropout mechanism for the selection of MVAE's components by using dHSIC regularization which overcomes the over-regularization problem characteristic in VAEs [11].

III. The objective function for the Mixture of Variational Autoencoders

A single VAE has a fixed processing capacity defined by its structure which is not able to model probabilistic relationships characterizing complex data. In the following, for comprehensively modelling complex data, we introduce the Mixture of Variational Autoencoders (MVAE) model. MVAE extends the concept of Mixtures of Gaussians [8], [10], into the deep learning framework. MVAE model learns a collection of separate latent spaces, extracted independently by each VAE component of a mixture. In Section III-A we define the basic objective function for the mixture model. In Section III-B we discuss how various VAE components can learn different aspects of the data, while deciding the number of components in the MVAE model is explained in Sections III-C and III-D.

A. The underlying framework and objective function

The learning goal of the proposed mixture model is to model a collection of separate latent spaces that capture different aspects of data. Let us denote by $p_{\eta_i}(\mathbf{x}|\mathbf{z}_i)$ the variational posterior for the i-th decoder, implemented by a Convolution Neural Network (CNN) of parameters η_i, where $\mathbf{z}_i$ represents the inferred stochastic latent variable vector, for $i = 1, \ldots, K$, where K is the number of VAE components.

The data generation process is defined by the following stages. The latent variable $\mathbf{z}_i$ is yielded by the i-th encoder :

$$\mathbf{z}_i \sim \mathcal{N}(\mu_{\theta_i}(\mathbf{x}), \sigma_{\theta_i}(\mathbf{x})), \tag{2}$$

where each encoder, is defined as a CNN of parameters θ_i, and models a multivariate Gaussian function, [36].

We do not directly recover the data by using a decoder, as in the classical VAE [11], due to the complexity of the mixture model. Instead, we firstly consider a simple network as a sub-decoder, implemented by a nonlinear transformation function denoted as $t_i(\cdot)$, $i = 1, \ldots, K$ which outputs a variable $\mathbf{s}_i$.

In the following, the mixing weights are sampled from a Dirichlet distribution, $Dir(\mathbf{a})$ [10], as :

$$\{w_1, \ldots, w_K\} \sim Dir(\mathbf{a}), \tag{3}$$

where $\mathbf{a} = \{a_1, a_2, \ldots, a_K\}$ represents its parameters, with each entry provided by one of the encoders. The transformed latent representations are then combined, considering the mixing parameters, to form a single representation:

$$\mathbf{s} = \sum_{i=1}^{K} w_i \mathbf{s}_i, \tag{4}$$

characterized by the constraint $\sum_{i=1}^{K} w_i = 1$, and considering the inference in the latent space we have the probability for the output variable $\mathbf{s}$:

$$p(\mathbf{s}|\mathbf{z}) = \sum_{i=1}^{K} w_i t_i \left(\int p_{\theta_i}(\mathbf{z}_i|\mathbf{x}) d\mathbf{x} \right) \tag{5}$$

where θ_i represents the parameters of each encoder network $i = 1, \ldots, K$. Afterwards, in the mixture model, we have a single mix-decoder for reconstructing the data $\mathbf{x}'$:

$$\mathbf{x}' \sim p(\mathbf{x}|\mathbf{s}). \tag{6}$$

For the generation process, each i-th sub-decoder corresponding to an encoder, is seen as a component of the mixture model. Each component has its own independent inference mechanism, which is beneficial for representation learning when representing complex latent spaces. Let us consider an approximate posterior $q_\theta(\mathbf{z}, \mathbf{w}|\mathbf{x})$ implemented by an independent encoder. We use the Jensen's inequality to obtain the variational lower bound as follows :

$$\begin{aligned} \log p(\mathbf{x}) &= \log \mathbb{E}_{q_\theta(\mathbf{z},\mathbf{w}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{z}, \mathbf{w})}{q_\theta(\mathbf{z}, \mathbf{w}|\mathbf{x})} \right] \\ &\geq \mathbb{E}_{q_\theta(\mathbf{z},\mathbf{w}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z}, \mathbf{w})}{q_\theta(\mathbf{z}, \mathbf{w}|\mathbf{x})} \right]. \end{aligned} \tag{7}$$

In the following we consider the independence of the latent variables in the joint log-likelihood $p(\mathbf{x}, \mathbf{z}, \mathbf{w}) = p(\mathbf{x}|\mathbf{z}, \mathbf{w})p(\mathbf{z})p(\mathbf{w})$ and also $q(\mathbf{z}, \mathbf{w}|\mathbf{x}) = q(\mathbf{z}|\mathbf{x})q(\mathbf{w}|\mathbf{x})$ and we replace these in (7). Then we have the upper bound on $-\log p_\eta(\mathbf{x})$ representing the mixture of the lower error bound (MELBO) basic objective function for MVAE, where the loss :

$$\begin{aligned} L_{MELBO} &= -\mathbb{E}_{q_\theta(\mathbf{z},\mathbf{w}|\mathbf{x})}[\log p_\eta(\mathbf{x}|\mathbf{z}, \mathbf{w})] \\ &\quad + \mathbb{E}_{q_\theta(\mathbf{z},\mathbf{w}|\mathbf{x})} \log \left[\frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right] + \mathbb{E}_{q_\theta(\mathbf{z},\mathbf{w}|\mathbf{x})} \log \left[\frac{p(\mathbf{w})}{q(\mathbf{w}|\mathbf{x})} \right] \\ &= -\mathbb{E}_{q_\theta(\mathbf{z},\mathbf{w}|\mathbf{x})}[\log p_\eta(\mathbf{x}|\mathbf{z}, \mathbf{w})] \\ &\quad + D_{KL}(q_\theta(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) + D_{KL}(q_\theta(\mathbf{w}|\mathbf{x})||p(\mathbf{w})) \end{aligned} \tag{8}$$

where $p(\mathbf{w})$ and $p(\mathbf{z})$ are the priors for the mixing parameters $\mathbf{w}$ and for the latent variables $\mathbf{z} = \{\mathbf{z}_i | i = 1, \ldots, K\}$, while $\theta = \{\theta_i | i = 1, \ldots, K\}$, $\eta = \{\eta_i | i = 1, \ldots, K\}$, represent the parameters of the networks modelling the mixture of encoders and decoders, respectively. After explicitly expanding the basic objective function corresponding to the mixture model we have:

$$\begin{aligned} L_{MELBO} &= -\mathbb{E}_{p(\mathbf{s}|\mathbf{z})}[\log p_\eta(\mathbf{x}|\mathbf{s})] + D_{KL}(q_\theta(\mathbf{w}|\mathbf{x})||p(\mathbf{w})) \\ &\quad + \frac{1}{K} \sum_{i=1}^{K} D_{KL}(q_{\theta_i}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \end{aligned} \tag{9}$$

where K is the number of components, $q_{\theta_i}(\mathbf{z}|\mathbf{x})$ represents the variational modelled by one of the mixing components, defined

by the parameters θ_i, and $p_\eta(\mathbf{x}|\mathbf{s})$ is the probability of the mix-decoder, modelled by a network defined by the parameters η, aiming to reconstruct the data $\mathbf{x}'$. Let us consider $t_{\gamma_i}(\mathbf{s}_i|\mathbf{z})$ the function of each sub-decoder, where $\{\gamma_i|i=1,\ldots,K\}$ are the parameters of K sub-decoders. The output of each sub-decoder is multiplied by its corresponding mixing parameter and the results are then summed up like in equation (5).

B. Enforcing the separation in the latent space among the MVAE components

L_{MELBO} from equation (9) is the basic objective function for the mixture model, where each encoder defines its own latent space. We enforce that various components learn distinct aspects of the data by employing a regularization term $r(\mathbf{z})$:

$$L_{Obj} = L_{MELBO} + \beta\, r(\mathbf{z}) \tag{10}$$

where L_{Obj} is the objective function of MVAE, and β is a hyperparameter controlling the relative strength of the regularization.

In the mixture model, each encoder has its own independent inference mechanism. Nevertheless, without a regularization mechanism, the encoders might all learn the same features, resulting in overlaps of their characteristic latent spaces. This happens because each associated encoder and sub-decoder shares the same network architecture. Consequently, we should increase the distinction between the latent representations of various MVAE components in order to encourage them to learn different aspects of the data.

Various measures for the regularization function $r(\mathbf{z})$, from (10), can be used for enforcing each component to learn a distinct latent space from the others. For example, the L2 norm between the latent variables of two different encoders, or statistical distances such as KL divergence or its symmetric correspondent, Jensen-Shannon (JS) divergence [37] between the probabilistic representation of the latent spaces for each pair of encoders, can be used. In any of these measures we calculate the differences between the latent space representations for all possible pairs of encoders from the mixture:

$$r(\mathbf{z}) = \sum_{i=1}^{K}\sum_{j=1}^{K} d(p(\mathbf{z}_i), p(\mathbf{z}_j)), \quad i \neq j \tag{11}$$

where $d(\cdot,\cdot)$ represents one of the discriminatory measures mentioned above, evaluated between the probabilistic representations $p(\mathbf{z}_i)$ and $p(\mathbf{z}_j)$, characterizing the latent space of a pair of encoders $\{i,j\}_{i,j=1,\ldots,K}$.

All measures listed above are always positive, and when considered in (10) they would be differentiated during the Stochastic Gradient Descent (SGD) training which can lead to derailing the convergence during the training. In order to avoid this situation we consider a new measure of independence by assuming that the joint probability of the latent space for the whole mixture of VAEs is equal to the product of marginal probabilities of the individual variables, [38]:

$$q(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K) = \prod_{i=1}^{K} q(\mathbf{z}_i) \tag{12}$$

where $q(\mathbf{z}_i)$ is the marginal probability of the latent variable $\mathbf{z}_i$, which represents the output of the i-th encoder. Let us consider the cross-covariance operator $C_{\mathbf{z}_i,\mathbf{z}_j}$ defined on the encoders output variables $\mathbf{z}_i$ and $\mathbf{z}_j$. The largest eigenvalue of the operator $C_{\mathbf{z}_i,\mathbf{z}_j}$ measures the dependence score between the distributions defined by the latent variables $\mathbf{z}_i$ and $\mathbf{z}_j$ and this should be zero for ensuring the independence of the two latent spaces. The covariance operator is defined based on the squared Hilbert-Schmidt norm as follows, [20] :

$$C_{\mathbf{z}_i,\mathbf{z}_j} = E_{\mathbf{z}_i,\mathbf{z}_j}[k(\mathbf{z}_i), l(\mathbf{z}_j)] - E_{\mathbf{z}_i}[k(\mathbf{z}_i)]E_{\mathbf{z}_j}[l(\mathbf{z}_j)] \tag{13}$$

where $\mathbf{z}_i$ and $\mathbf{z}_j$ are the latent variables produced by i-th and j-th encoders and $k(\cdot)$, $l(\cdot)$ represent kernel functions in the latent space, usually defined as Gaussian. The Hilbert-Schmidt independence criterion (HSIC) [20] represents the generalization of Frobenius norm and is defined as :

$$HSIC(\mathbf{z}_i, \mathbf{z}_j) = \|C_{\mathbf{z}_i,\mathbf{z}_j}\|^2 \tag{14}$$

where $C_{\mathbf{z}_i,\mathbf{z}_j}$ is provided in (13).

The definition of the cross-covariance operator can be extended for assessing the independence of d variables, similarly to the expressions from (13) and (14), representing the latent space $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_d\}$ defined by d VAE components [20], [38]. This criterion is called dHSIC and evaluates the independence among all K encoder components. dHSIC is null if and only if all components $\mathbf{z}_i$, $i=1,\ldots,K$ are mutually independent. The K components are mutually independent if their joint distribution is equal to the product of their marginal distributions, [39]. dHSIC can be easily integrated as the regularization factor $r(\mathbf{z})$ in the optimization function L_{Obj}, from (10), as follows:

$$L_{Obj} = L_{MELBO} + \beta\, \mathrm{dHSIC}\left(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_\mathrm{K}\right), \tag{15}$$

where L_{MELBO} defines the inference process for MVAE and β is the contribution of the constraint associated with dHSIC.

C. Deciding the number of VAE components using dropout probabilities

The number of components in classical mixture models, such as GMMs or RBF networks, was selected in various ways. New processing units were added in [1] as long as decreasing the classification error. Bayesian Information Criterion [40], which is equivalent to the Minimum Description Length [41], was used in the context of Variation Expectation-Maximization algorithm in [10] for deciding the number of mixing components. In this research study, we propose to extend the idea of probabilistic dropout for selecting the appropriate number of VAE components during the training of MVAE while ensuring an efficient end-to-end training mechanism. Each mixing component is seen as a probabilistic node which contributes to the mixture output in the network, according to a dropout probability.

We consider two different approaches for the dropout procedure when selecting the number of components in MVAE. First we introduce a traditional dropout method, by sampling a set of variables, which are either 0 or 1, according to the dropout rate. This models a vector whose entries represent

masking parameters for the outputs of each of the K sub-decoders. We denote a masking vector, sampled from the Bernoulli distribution, by $\mathbf{m} = \{m_1, m_2, \ldots, m_K\}$, whose entries are used as the weights for the mixing parameters :

$$b_i = \frac{m_i w_i}{\sum_{j=1}^{K} m_j w_j}. \tag{16}$$

Then, the output of each sub-decoder is multiplied by the corresponding mixing parameter and the probability of the variable $\mathbf{s}$, for the whole MVAE model (5), becomes:

$$p(\mathbf{s}|\mathbf{z}) = \sum_{i=1}^{K} b_i t_i \left(\int p_{\theta_i}(\mathbf{z}_i|\mathbf{x}) d\mathbf{x} \right). \tag{17}$$

In this way, only the components contributing significantly to the inference will be considered for the generation process in MVAE, according to the dropout masking parameters from (16). The variable $\mathbf{s}$ is characteristic of the mixed latent space, which is then fed into the mix-decoder, yielding as outputs the reconstructed data as in (6). The Bernoulli distribution can be used for modelling the masking. However, this is non-differentiable and cannot be used directly in the context of the end-to-end backpropagation training.

We could also consider the variational Gaussian dropout for selecting the VAE components (MVAE-Gau) as in [42]. In this case, a dropout vector $\mathbf{m}$ for VAE components is sampled from a Gaussian distribution $\mathcal{N}(1, \tau)$, where $\tau = p/(1-p)$, and p is the dropout rate. The sampled masking vector $\mathbf{m}$ is then used directly on the mixing parameters $\mathbf{w}$ as in (16). The dropout loss is taken into account in the objective function L_{Obj} from (15) by adding the KL-divergence penalty associated with the dropout $D_{KL}(q(\mathbf{m})||p(\mathbf{m}))$, measuring the KL divergence between the variational distribution for the masking parameters $q(\mathbf{m})$ and its corresponding posterior $p(\mathbf{m})$. After considering the general objective function for the mixture model (9), the dHSIC regularization from (15), and considering the loss due to each VAE component dropout, we obtain the following objective function for the MVAE-Gau model :

$$\begin{aligned} L_{MVAE-Gau} = & - \mathbb{E}_{p(\mathbf{s}|\mathbf{z})}[\log p_\eta(\mathbf{x}|\mathbf{s})] + D_{KL}(p_\theta(\mathbf{w}|\mathbf{x})||p(\mathbf{w})) \\ & + \frac{1}{K} \sum_{i=1}^{K} D_{KL}(q_{\theta_i}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ & + \beta \, \mathrm{dHSIC}\left(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_\mathrm{K}\right) \\ & + D_{KL}(q(\mathbf{m})||p(\mathbf{m})). \end{aligned} \tag{18}$$

The last term $D_{KL}(q(\mathbf{m})||p(\mathbf{m}))$, represents the contribution of the dropout loss and is not analytically tractable when considering a Gaussian distribution dropout, but it can be approximated using the following polynomial expression :

$$D_{KL}(q(\mathbf{m})||p(\mathbf{m})) \approx c - 0.5\log(\tau) - c_1\tau - c_2\tau^2 - c_3\tau^3 \tag{19}$$

where τ defines the variance of the Gaussian distribution used for sampling the dropout, and the constants used in the polynomial approximation c, c_1, c_2, c_3 are provided in [42].

D. Sampling the Gumbel-softmax distribution for selecting the number of components

In this section we consider the Gumbel-softmax distribution [43], representing a categorical distribution which is also differentiable, for selecting the number K of VAE components. A sample vector is drawn from a categorical distribution with probabilities $\{a_i | i = 1, \ldots, K\}$ for each encoder of MVAE by using the Gumbel-softmax trick [44], [45] :

$$one\_hot\left(\arg\max_i(\log a_i + g_i)\right) \tag{20}$$

where g_i is a sample drawn from the $\mathrm{Gumbel}(0, 1)$ distribution. The Gumbel-softmax trick adopts the softmax function as a continuous, differentiable approximation to the $\arg\max$ expression as, [43] :

$$m_i = \frac{\exp[(\log(a_i) + g_i)/T]}{\sum_{i=1}^{K} \exp[(\log(a_i) + g_i)/T]} \tag{21}$$

where $\mathbf{m} = \{m_1, m_2, \ldots, m_K\}$ is a K-dimensional masking vector and T is the temperature parameter. When the temperature T is increasing, the samples inferred from the Gumbel-softmax become uniformly distributed and they are no longer selective. In contrast, if the temperature T approaches zero, the Gumbel-Softmax distribution becomes the one_hot selective categorical distribution, picking up one component of the mixture over the others.

In the following, we estimate the dropout masking parameters by using the following sampling process:

$$m_i = \frac{\exp((\log(1-p) + g_1)/T)}{\exp((\log(1-p) + g_1)/T) + \exp((\log(1-p) + g_2)/T)} \tag{22}$$

where m_i is a sampled masking value, which can be closer to either 1 or 0, corresponding to keeping or dropping the mixing component i, $g_1, g_2 \sim Gumbel(0, 1)$, [46], where p is a learnable dropout rate, while we set the temperature as $T = 0.1$. The dropout masking parameters m_i, depending on a single dropout rate p, are then combined with the mixing parameters, sampled from the Dirichlet distribution, as in equation (16), and then used for calculating the mixed latent variable, as in (17). Eventually, the mix-decoder yields the reconstructed data, as in (6). This approach is called the Mixture of VAEs with Gumbel-softmax dropout (MVAE-GS).

The Gumbel-softmax trick from (21) approximates a Bernoulli distribution by generating samples close to either 0 or 1, while it is also differentiable. Gal *et al.* in [47] considered the dropout regularization term as the entropy of a Bernoulli random variable :

$$H(p) = -p\log p - (1-p)\log(1-p) \tag{23}$$

where p is the dropout probability in (22). We can see that this regularization term depends only on the dropout rate p and if we fix the dropout rate during the training, this term can be omitted. However, if we optimize the dropout rate, this term plays an important role in the MVAE mixture components selection. For instance, the dropout rate becomes close to 0.5, when maximizing $H(p)$ in (23).

After considering the entropy of the dropout regularization from (23) for the Gambel-softmax dropout penalty, instead of the last term from (18) used for MVAE-Gau, the objective function for MVAE-GS method becomes :

$$\begin{aligned} L_{MVAE-GS} = &- E_{p(\mathbf{s}|\mathbf{z})}[\log p_\eta(\mathbf{x}|\mathbf{s})] + D_{KL}(p_\theta(\mathbf{w}|\mathbf{x})||p(\mathbf{w})) \\ &+ \frac{1}{K}\sum_{i=1}^{K} D_{KL}(q_{\theta_i}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ &+ \beta\ \mathrm{dHSIC}\left(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_\mathrm{K}\right) - \mathrm{H(p)}, \end{aligned} \tag{24}$$

where $H(p)$ is provided in (23).

IV. MVAE STRUCTURE AND TRAINING

In the following we discuss the architecture and the training algorithm for the MVAE model.

A. The MVAE model structure

The proposed mixture model is based on a collection of encoders and sub-decoders processing the information in parallel. Their outputs are weighted according to their contribution to the data representation. Meanwhile, this structure is enabled with a dropout mechanism aiming to define a minimal processing architecture. The structure of the MVAE model is shown in the diagram from Fig. 1. Each encoder with the associated sub-decoder can be seen as a component in the MVAE mixture model. The encoder outputs the hyperparameters $\{\mu_i, \sigma_i\}$ of a Gaussian distribution, and one parameter a_i of the Dirichlet distribution, for $i = 1, \ldots, K$, considering initially K mixing components. Then, the code is sampled from the distribution modelled by the corresponding hyperparameters. In order to allow the gradients to be estimated from the encoder to sub-decoder, we use the reparametrization trick, [11] :

$$\mathbf{z}_i = \mu_i + \sigma_i \odot \varepsilon, \tag{25}$$

where $\varepsilon \sim \mathcal{N}(0, I)$ is a random variable sampled from the Gaussian distribution of mean 0 and having the identity matrix I as the covariance.

For the mixing parameters, we adopt implicit reparameterization gradients [48]. The mixing parameters are sampled from the Dirichlet distribution $Dir(a_1, a_2, \ldots, a_K)$, with each parameter produced by one of the encoders:

$$\left(\frac{w_1}{\sum_{j=1}^{K} w_j}, \ldots, \frac{w_K}{\sum_{j=1}^{K} w_j}\right) \sim Dir(a_1, \ldots, a_K), \tag{26}$$

where the sum of the mixing parameters is 1. The mixing parameters are then multiplied by the dropout parameters which are either 0 or 1, defined as in Section III-C for MVAE-Gau, or as in Section III-D for MVAE-GS. The contribution of each mixing component is calculated according to (16).

B. Training the Mixture of VAEs model

Although the proposed mixture model is based on a collection of encoders and sub-decoders, we show that it can be easily trained by using the SGD algorithm [49], when considering either the objective function from equation (18) for MVAE-Gau or that from (24) for MVAE-GS, depending on what component dropout procedure is adopted for the mixing model. Similarly to the classical VAE [11], we consider the isotropic multivariate Gaussian $\mathcal{N}(0, I)$ as the prior distribution for each decoder over their latent variable space representations. The KL divergence $D_{KL}(q_{\theta_i}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$, between the prior and posterior for each i-th encoders, is calculated considering Gaussian pdfs in the latent space, as:

$$D_{KL}(q_{\theta_i}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{S_i}(1 + \log\left(\sigma_{i,j}^2\right) - \mu_{i,j}^2 - \sigma_{i,j}^2), \tag{27}$$

where S_i is the dimension of the latent variable space $\mathbf{z}$ for the i-th encoder and the characteristic latent space variables $\mu_{i,j}$ and $\sigma_{i,j}$ are evaluated from the training data. The total KL divergence, is calculated considering all K components, where each VAE component contributes with the expression from (27). We also calculate the KL divergence $D_{KL}(q_\theta(\mathbf{w}|\mathbf{x})||p(\mathbf{w}))$ corresponding to the mixing parameters, representing the second term from either objective function, (18) or (24). The KL divergence corresponding to the mixing parameters is evaluated analytically between two Dirichlet distributions of parameters $\mathbf{a} = \{a_1, \ldots, a_K\}$ and $\mathbf{b} = \{b_i, \ldots, b_K\}$ as:

$$\begin{aligned} D_{KL}(q_\theta(\mathbf{a}|\mathbf{x})||p(\mathbf{b})) = &\log\Gamma(a_0) - \sum_{i=1}^{K}\log\Gamma\left(a_i\right) \\ &- \log\Gamma(b_0) + \sum_{i=1}^{K}\log\Gamma(b_i) \\ &+ \sum_{i=1}^{K}(a_i - b_i)(\psi(a_i) - \psi(a_0)) \end{aligned} \tag{28}$$

where $a_0 = \sum_{i=1}^{K} a_i$, $b_0 = \sum_{i=1}^{K} b_i$, $\Gamma(\cdot)$ is the Gamma function and $\Psi(\cdot)$ is the Digamma function. In practice, $\mathbf{a}$ are the parameters of the Dirichlet distribution estimated by all encoders, following the training, while $\mathbf{b}$ are the parameters of an empirical Dirichlet distribution, $p(\mathbf{w})$.

For the reconstruction error, when considering N training images, we use the mean squared error (MSE) criterion, representing the first term from (9) and in the expressions from (18) and (24), as :

$$L_{Rec} = \frac{1}{2}\sum_{i=1}^{N}\|\mathbf{x}_i' - \mathbf{x}_i\|_F^2 \tag{29}$$

where $\mathbf{x}'$ represents the reconstructed image result by the mix-decoder while $\|\cdot\|_F$ denotes the Frobenius norm.

The gradient used for SGD optimization [49], when considering the objective function $L_{MVAE-Gau}$ from equation (18), derived in Section III-C, is given by :

$$\begin{aligned} &\nabla_\Omega^{MVAE-Gau}[L_{Rec} + \sum_{i=1}^{K} D_{KL}(q_{\theta_i}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ &+ D_{KL}(q_\theta(\mathbf{a}|\mathbf{x})||p(\mathbf{b})) + \beta\ dHSIC(\mathbf{z}_1, \ldots, \mathbf{z}_K) \\ &+ D_{KL}(q(\mathbf{m})||p(\mathbf{m}))], \end{aligned} \tag{30}$$

where the first three terms represent the image reconstruction (29), the KL divergence for all MVAE's components, where

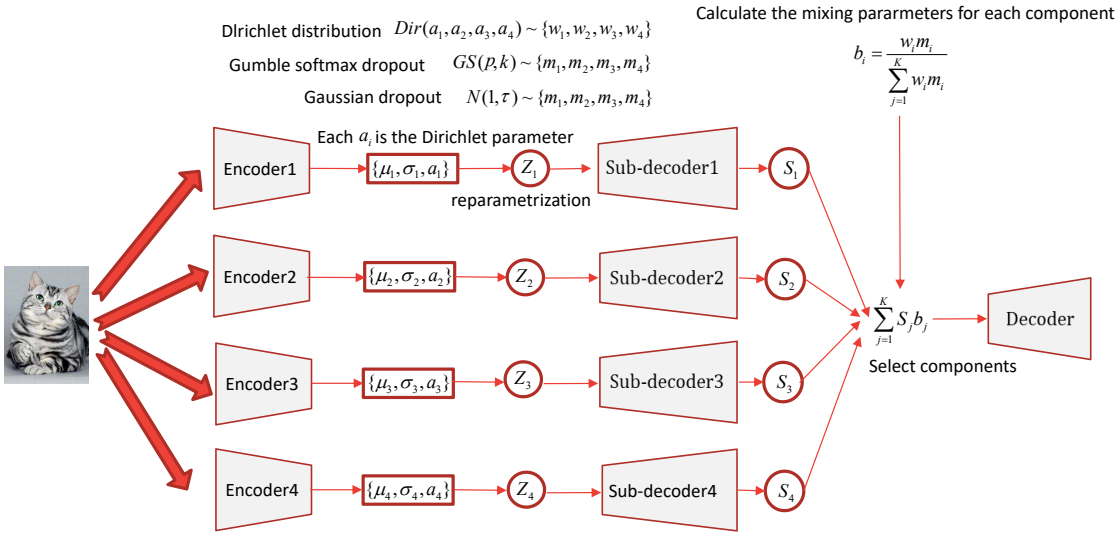


Fig. 1. The structure of the proposed MVAE model, when considering $K = 4$ encoders with associated sud-decoders.

each component contributes with the expression from (27) and their mixing weights (28), respectively, while the last two components represent the objective functions for enforcing the independence of the mixing components (15) and for evaluating their dropout probabilities.

When considering the MVAE-GS approach, described in Section III-D, we have the following updating gradient for the SGD based training:

$$\nabla_{\Omega}^{MVAE-GS}[L_{Rec} + \sum_{i=1}^{K} D_{KL}(q_{\theta_i}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) + D_{KL}(q_{\theta}(\mathbf{a}|\mathbf{x})||p(\mathbf{b})) + \beta\, dHSIC(\mathbf{z}_1,\ldots,\mathbf{z}_K) - H(p) \tag{31}$$

with the last term provided in equation (23).

The parameters being updated in the MVAE model are :

$$\Omega = \{\theta, \gamma, \nu, \eta\} \tag{32}$$

where θ and γ are the parameters of the mixture of K encoders $q_\theta(\mathbf{z}|\mathbf{x})$ and sub-decoders $p_\gamma(\mathbf{s}|\mathbf{z})$, and η are the parameters for the mix-decoder network, while ν represent the parameters of the network inferring the dropout parameters for the components.

The pseudocode of the MVAE-GS training algorithm is provided in Algorithm 1.

Algorithm 1: MVAE-GS training algorithm.

Algorithm : Training procedure with Bernoulli dropout
1:Sample $X^r = \{x^1, x^2, \ldots, x^N\}$ from training dataset
2:**While** $epoch < epoch^{\max}$ **do**
3: **While** $batch < batch^{\max}$ **do** minibatch procedure
4: $x_{batch} = Select(epoch, X^r)$ batch samples
5: $(\mu_1,\sigma_1,a_1)..,(\mu_k,\sigma_k,a_k) \leftarrow E_1(x_{batch})..,E_k(x_{batch})$
6: $\{z_1,\ldots,z_k\} \leftarrow$ latent variables from all encoders
7: $(a_1,\ldots,a_k) \leftarrow$ Dirichlet parameters from all encoders
8: $(w_1,\ldots,w_k) \leftarrow \mathrm{Dir}(a_1,\ldots,a_k)$ Sampling mixing weights
9: $\{s_1,\ldots,s_k\} \leftarrow t_1(z_1)..,t_k(z_k)$ Transform by sub-decoders
10: $GS(a,k) \sim (m_1,\ldots,m_k)$ Dropout from Gumbel-Softmax
11: $b_i = \frac{w_i m_i}{\sum_k w_k m_k}$ Dropout many components
12: $s = \sum_{j=1}^{k} s_j b_j$ Integral outputs of all components
13: $x' = p(x \mid s)$ reconstruct data by mix-decoder
14: Calculate reconstruction error using equation (29)
15: Calculate kl divergence on guassian using equation (28)
16: Calculate kl divergence on dirichlet using equation (27)
17: Calculate dHSIC using equation (14)
18: Update parameters of model according to equation (31)
20: **End**
21:**End**

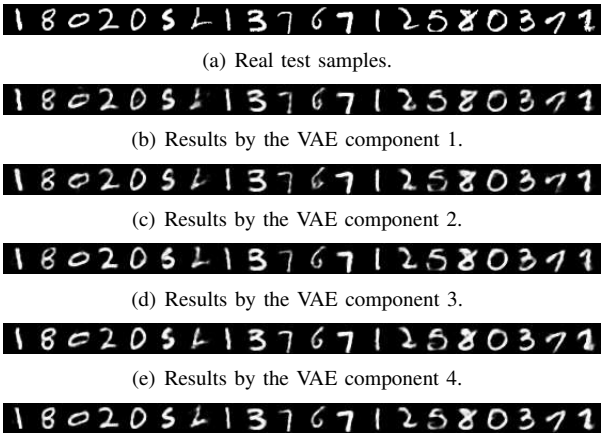
V. Experimental results

In this section we assess the results provided by the proposed Mixture of VAEs (MVAE) model on a variety of datasets. Each component of the MVAE model is represented by an encoder and a sub-decoder. The probabilistic models are implemented by using fully connected and convolutional networks, depending on the complexity of the dataset. The prior is the standard Gaussian distribution $\mathcal{N}(0, I)$ and the outputs for each encoder are the hyperparameters of the Gaussian distribution defining its latent space. We set $\beta = 1$ in the objective function from (15), representing the weight for the dHSIC term, defining the independence among the mixing components. For the implementation we use Python language and the deep learning Tensorflow platform.

A. MNIST dataset

The MNIST dataset [50], consists of images of handwritten digits of size 28×28 pixels. We train MVAE with $K = 4$ components when using the MVAE-Gau loss, defined in equation (18), considering 60,000 and 100,000 images for training and testing, respectively. A set of original MNIST images are shown in Fig. 2a, and their reconstruction by MVAE is provided in Fig. 2f, while the reconstructions by each

of the four components are shown in Figures 2b-e. We observe that the mixture model gives a better reconstruction than each individual component. In the following, we investigate the uniqueness of the latent space representation by each component considering a different sub-decoder than its corresponding encoder during the tests. For the images of digits from Fig. 3a we consider reconstructions by mismatching the sub-encoders and encoders for the same images. The results of such mismatches are shown in Figures 3b-e and we observe that these images are not properly reconstructed. This happens because each sub-decoder is uniquely fitted to the associated encoder and not to any of the others which would yield different latent space representations.



(a) Real test samples.

(b) Results by the VAE component 1.

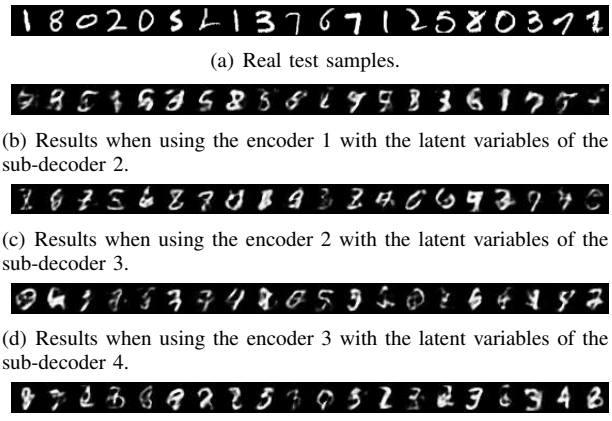
(c) Results by the VAE component 2.

(d) Results by the VAE component 3.

(e) Results by the VAE component 4.

(f) Results by the MVAE model.

Fig. 2. Reconstructed images from the MNIST dataset by MVAE and its individual VAE components.



(a) Real test samples.

(b) Results when using the encoder 1 with the latent variables of the sub-decoder 2.

(c) Results when using the encoder 2 with the latent variables of the sub-decoder 3.

(d) Results when using the encoder 3 with the latent variables of the sub-decoder 4.

(e) Results when using the encoder 4 with the latent variables of the sub-decoder 3.

Fig. 3. Results when mis-matching the encoders and sub-decoders when reconstructing images from the MNIST dataset.

B. Representing complex images

In the section, we evaluate the performance of the proposed MVAE mixture model on some databases which contain more complex images, such as the human face dataset entitled Celebrities Faces Attributes (CelebA) [51], and ImageNet database [52]. CelebA dataset contains almost 200,000 human face images with 10,177 identities. Each encoder of the mixture model is implemented by a deep convolution net consisting of 5 convolution layers while the fully connected layers are used to output the 256-dimensional hyperparameters of the Gaussian and Dirichlet distributions defining the latent space for MVAE. Each sub-decoder is implemented by a simple architecture with only a single layer of $8 \times 8 \times 256$. The output of the sub-decoder is then transformed into a 3D tensor after multiplying with the mixing weights and dropout parameters, as shown in Fig. 1. The mix-decoder is a deep deconvolution net consisting of 7 layers, which receive the tensor as the input and generates images as the output. We set the kernel size as 3×3 for all convolution processing units. The mixture model is initially built using $K = 6$ mixing components considering the dropout rate optimized during the training using the Gaussian dropout (MVAE-Gau), as described in Section III-C. We train the mixture model using the Adam optimization algorithm for 10 epochs with a learning rate of 0.001. A set of face images from CelebA dataset is shown in Fig. 4a, while their reconstructions by MVAE-Gau are provided in Fig. 4b. We also show the reconstruction results in Fig. 5c for MVAE-Gau for the subset of images from ImageNet database [52], shown in Fig. 5a. For comparison the same images are reconstructed by MSVI [18] in Fig. 5b. It can be observed that both human face and natural images are reconstructed well by the MVAE-Gau model.

We also explore the manifold continuity by performing interpolation experiments in the latent space. For a single VAE, we can directly perform interpolations on the latent variables inferred by that VAE. Nevertheless, in the mixture model, we have multiple variational encoders, each defining its own latent space, which allows us to perform interpolations in new regions of the latent space, located in between the latent spaces modeled by different VAE components. Initially, a pair of images $\{\mathbf{x}_1, \mathbf{x}_2\}$ is randomly selected and we map these into the latent representation by using the trained encoders. We consider $K = 6$ components, and the interpolation $\mathbf{d}_i$ is performed on the output of each encoder as :

$$\mathbf{d}_i = (1 - a)E_i(\mathbf{x}_1) + aE_i(\mathbf{x}_2), \tag{33}$$

where $E_i(\cdot)$ is the output i-th variational encoder, $i = 1, \ldots, 6$ and a is a hyperparameter controlling the interpolation in the latent space. Then, the latent space interpolations are transformed through the sub-decoders $SubDec_i(\cdot)$ into :

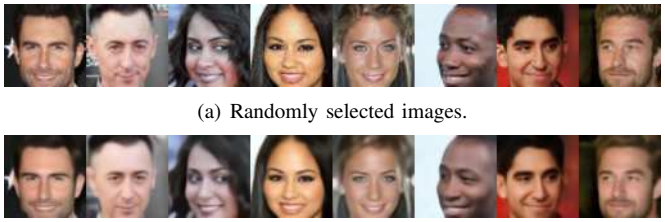
$$\mathbf{c} = \sum_{i=1}^{6} w_i \cdot SubDec_i(\mathbf{d}_i), \tag{34}$$

where w_i is the weight modeling the contribution of each sub-decoder $i = 1, \ldots, K$ to the result of the interpolation $\mathbf{c}$. Then, the mix-decoder $MixDec(\cdot)$ is used to generate the image $\mathbf{x}'$:

$$\mathbf{x}' = MixDec(\mathbf{c}). \tag{35}$$

Interpolation results on CelebA dataset are shown in Fig. 6, where $\{\mathbf{x}_1, \mathbf{x}_2\}$ are displayed as the extreme left and right images from each row of images. The interpolated images, when varying a in (33) are shown in between the reconstructions of the original images. From these results we observe smooth and

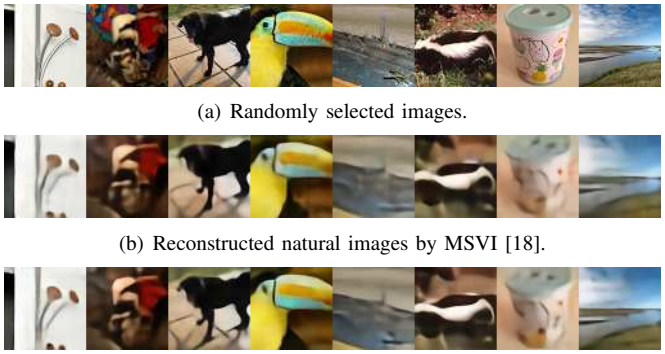
realistic transitions which model various changes in the human face appearance, such as changing the illumination in the image, varying the hair style, modifying the age appearance, and so on. By exploring the manifold continuity, these results show the enriched data representations which can be achieved in the latent space of the MVAE model.



(a) Randomly selected images.

(b) Reconstructed face images by MVAE-Gau.

Fig. 4. Reconstruction results for face images from CelebA dataset, [51].



(a) Randomly selected images.

(b) Reconstructed natural images by MSVI [18].

(c) Reconstructed natural images by the MVAE-Gau.

Fig. 5. Reconstruction results for images from ImageNet dataset.

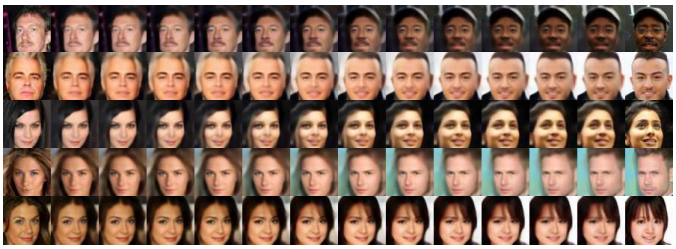


Fig. 6. Interpolation results in the latent space, where the extreme left and right are real images from CelebA, while the images in between are the reconstruction results by MVAE model.

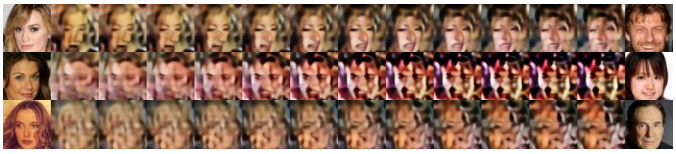
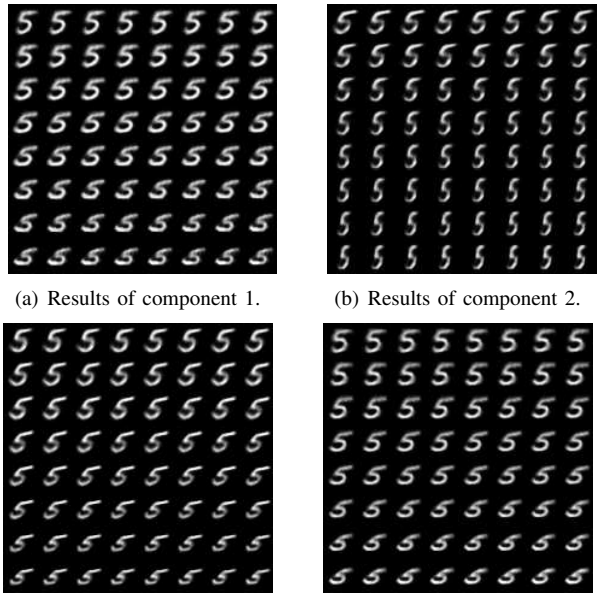


Fig. 7. Interpolation results, when the sub-decoder uses the latent space corresponding to a different encoder as its input, according to equation (36), where the extreme left and right images are real images from CelebA.

We also investigate the separation in the latent space between the information encoded by different encoders. For this experiment, we choose randomly a pair of images and extract their latent variables using the encoders. Then we perform interpolations for each sub-decoder, where instead of using the corresponding inputs as in the previous experiment, we would use the outputs of a different encoder as the input to the given sub-decoder, replacing (34) with :

$$\hat{\mathbf{c}} = \sum_{i=1}^{6} w_i \cdot SubDec_i(\mathbf{d}_j) \tag{36}$$

where $j \neq i$. For example, the first sub-decoder is fed with the interpolation results in the latent space of the second encoder. The reconstruction results are shown in Fig. 7, where the real images $\{\mathbf{x}_1, \mathbf{x}_2\}$ are shown as the first and last on each row, while the images in between are the interpolation results using (36). We observe that in these situations, the mixture model does not generate reasonable results. The main reason is that each component encoder learns a unique latent space, which is distinct from all other latent spaces modelled by the other encoders. This result is the consequence of enforcing the learning of distinct latent spaces for each VAE, by using the dHSIC measure, as described in Section III-B.



(a) Results of component 1.

(b) Results of component 2.

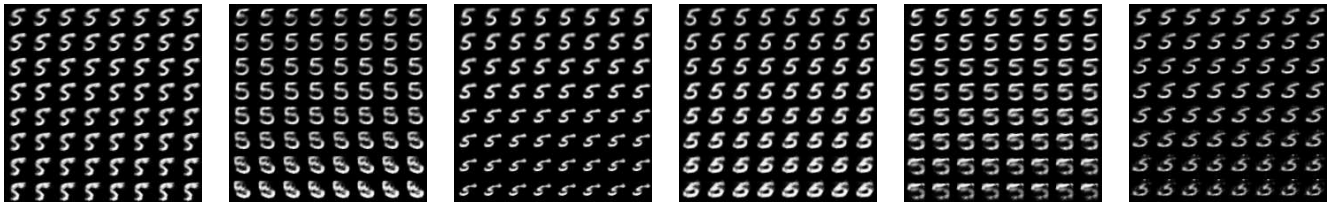
(c) Results of component 3.

(d) Results of component 4.

Fig. 8. Generated images of digits by different VAE components when fixing the class label and changing the first latent variable z_1 from 0 to 3.

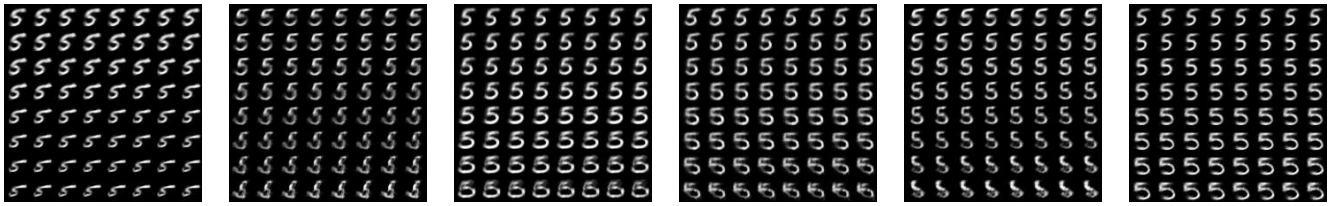
C. Latent space analysis

In the following we explore the latent space representation for the MVAE model. Each encoder and associated sub-decoder consider jointly the data and their corresponding class labels $\{\mathbf{x}, \mathbf{y}\}$, where the class information $\mathbf{y}$ is represented as a one-hot vector. We train the MVAE-Gau model, with Gaussian defined dropout, for $K = 4$ components, and considering only two latent variables, z_1 and z_2, on images from the MNIST dataset. Then we fix the class label while changing only one of the latent variables from 0 to 3. The generated results with images of the digit '5' are shown in the Figures 8a-d for each of the four components considered. Meanwhile, the generated images of the handwritten digit '5', for $K = 6$



(a) Component 1. (b) Component 2. (c) Component 3. (d) Component 4. (e) Component 5. (f) Component 6.

Fig. 9. Generated images of digits for six different VAE components when fixing the class label and changing the first latent variable z_1 from 0 to 6.0.



(a) Component 1. (b) Component 2. (c) Component 3. (d) Component 4. (e) Component 5. (f) Component 6.

Fig. 10. Generated images of digits for six different VAE components when fixing the class label and changing the second latent variable z_2 from 0 to 6.0.

components, when changing either the variable z_1 or z_2 from the latent space are shown in Figures 9a-f and 10a-f, respectively, for each VAE component. We observe that each component provides a different output when changing a single variable in the latent space, which shows the ability of the latent space to model various data attributes. We observe that we achieve a better disentanglement in the data representation when increasing the number of components from 4 to 6. By considering two different latent variables for each component of the mixture we can model additional underlying factors, while each component defines a different writing style for the generated images.

We also train MVAE-Gau with $K = 4$ components, under the unsupervised learning setting, assuming that the class labels are not known. The latent variables corresponding to the images from the MNIST database, during testing, are plotted in Fig. 11, where the colours represent different class labels. These results indicate that MVAE provides a rich data representation, where the information is distributed among different regions of the latent space for each component.

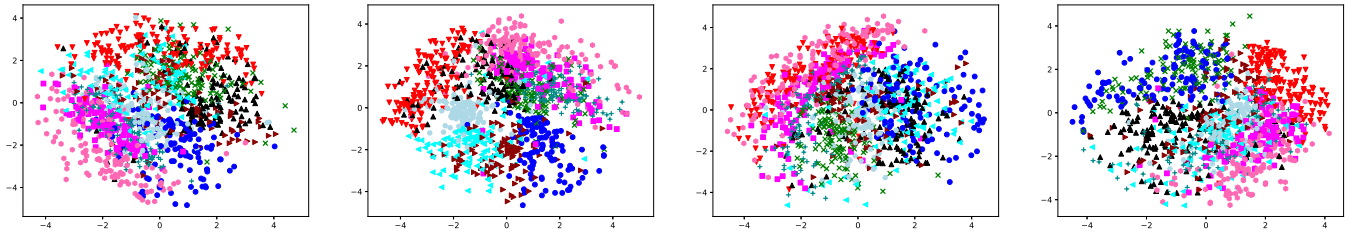
D. Enforcing the separation between the latent space representations of MVAE's components

The dHSIC measure is used to enforce the independence between the latent spaces of the encoders, as explained in Section III-B. In this section we evaluate the separation between the latent spaces modelled by the encoders, through experiments following the training of MVAE-GS on the MNIST dataset. We consider $K = 6$ VAE components and a latent space consisting of two variables, z_1 and z_2. First, we estimate the dHSIC measure between each pair of encoders' latent space distributions. The results, when assuming a penalty of $\beta = 1$ and $\beta = 10$ in the objective function from (24) are provided in Figures 12a and 12b, respectively. We observe that when using a larger β we achieve better independence between the encoding distributions.

After training MVAE-GS, considering the cost function from (24) with $K = 6$ components, we randomly select a batch of images belonging to the same class, and then estimate their corresponding latent vectors by using various mixing components. We map the latent representation results in Figures 13a and 13b, for $\beta = 10$ and $\beta = 0$, respectively, where the colors represent the latent space representations produced by different components. We observe that the latent space represented in Fig. 13a contains distinct clusters of latent vectors for the MVAE components, while when not considering the dHSIC term, *i.e.* $\beta = 0$, each component tends to embed data into the same region of the latent space, as shown in Fig. 13b. These results show that the dHSIC measure, used in the objective function from (24), plays an important role in encouraging each component to embed data in different regions of the latent space.

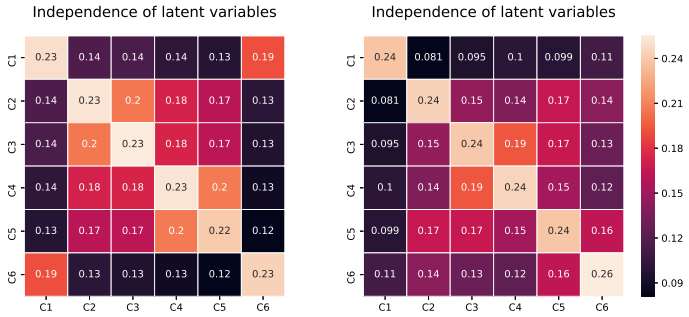
E. Visual quality evaluation

In this section, we evaluate the reconstruction and representation learning ability of the proposed MVAE model and compare with the results achieved by the state of the art. The following models are considered for comparison: (1) Multiple Source Variational Inference (MSVI) [18] model, which is a mixture of experts where each expert is implemented by a VAE. We implement MSVI by using the same network architecture used for our model. However, MSVI has more parameters than the proposed model, since we use a sub-decoder implemented by a single layer instead of an entire deep convolution net. (2) InfoVAE [33] is the current state of art VAE framework, which is able to balance accurate inference with the reconstruction quality. We implement InfoVAE by using a large network architecture. (3) β-VAE is a variant of the VAE framework which aims to learn disentangled representations. (4) We also compare with a single VAE [11] implemented by a large network architecture. (5) Finally, we consider for comparison several other recent VAE frameworks, such as those proposed in [34], [35], [53].



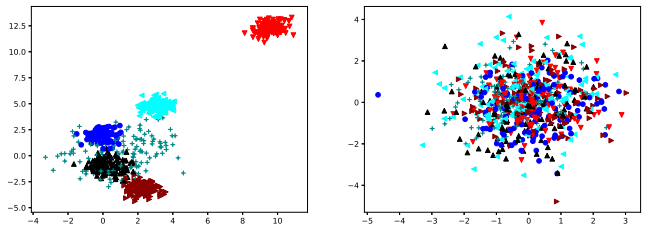
(a) Component 1. (b) Component 2. (c) Component 3. (d) Component 4.

Fig. 11. The representation of the latent space for the MVAE model for the MNIST dataset.



(a) $\beta = 1$ (b) $\beta = 10$

Fig. 12. Analysing the independence of the latent spaces for the mixture components in MVAE-GS, considering the cost function from (24).



(a) $\beta = 10$ (b) $\beta = 0$

Fig. 13. The representation of the latent spaces for MVAE for the images showing the handwritten digit '1' from the MNIST dataset. Different colours represent the latent vector projections of different components.

TABLE I
RMSE AND INCEPTION SCORE ON CIFAR10 DATABASE.

Model	RMSE	*IS*
DCGAN* [54] in [16]	-	6.16
ALI* [55] in [16], [56]	-	5.34
PixelCNN++* [57] in [32]	3.289	5.51
BEGAN [58]	-	5.62
MVAE-Gau	**2.97**	**6.38**
MVAE-Gau fixed K	3.52	6.09
MVAE-GS	3.36	6.26
MSVI [18]	3.46	5.84
InfoVAE [33]	3.24	6.17
β-VAE [59]	9.12	4.92
VAE	4.64	5.04
Wasserstein-Wasserstein Auto-Encoders [35]	3.49	6.05
Continuous Bernoulli VAE* [53]	-	4.55
MAE [34]	4.11	5.49

TABLE II
RMSE AND INCEPTION SCORE ON CIFAR100 DATABASE.

Model	RMSE	*IS*
MVAE-Gau	**3.10**	**5.64**
MVAE-Gau fixed K	3.68	5.44
MVAE-GS	3.11	5.60
MSVI [18]	5.30	4.72
InfoVAE [33]	4.29	5.06
β-VAE [59]	9.17	3.31
VAE	6.84	4.07
Wasserstein-Wasserstein Auto-Encoders [35]	5.83	4.97
MAE [34]	4.11	5.20

We evaluate the reconstruction ability of the proposed approach on CIFAR10 dataset [60], which contains 60,000 images grouped into 10 classes. We train the proposed MVAE model using 50,000 images, with $K = 6$ components initially, using the Gambel-softmax dropout during the training, while setting the maximum number of training epochs to 100. In order to evaluate both generative ability and the likelihood of model collapse, we consider the Inception Score (*IS*), [61]:

$$IS = \exp(\mathbb{E}_x[D_{KL}(p(\mathbf{y}|\mathbf{x})||p^*(\mathbf{y}))]) \quad (37)$$

where D_{KL} represents the Kullback-Leibler divergence between the distributions of the labels of the recovered images and those from the database; $\mathbf{x}$ represents the image and $p(\mathbf{y}|\mathbf{x})$ is the probability of the softmax output for the trained classifier; $p^*(\mathbf{y})$ represents the labels statistics for the given images. The reconstruction Root Mean Square Error (RMSE) as well as the Inception Score for CIFAR10 and CIFAR100 databases are provided in Tables I and II, where "MVAE-Gau fixed K" denotes that the model considers K fixed and '*' represents that we cite results reported at the indicated reference. We also indicate the number of components found when considering the Gambel-softmax dropout (MVAE-GS), and the average, calculated from several runs, for this database is $K = 4.11$. From Table I we find that MVAE-Gau provides the best result. We can also observe that selecting the number of components definitely improves the performance, as shown when comparing the results of MVAE-Gau with those provided by MVAE-Gau using a fixed K.

We also evaluate the performance on the more challenging dataset, ImageNet [52]. The results are reported in Table III, where it can be observed that the proposed MVAE based models outperform all other VAE based methods considered

for comparison when applied on ImageNet.

TABLE III
RMSE AND INCEPTION SCORE (*IS*) ON IMAGENET DATABASE.

Model	RMSE	*IS*
MVAE-Gau	**19.44**	**6.84**
MVAE-Gau fixed K	20.87	6.30
MVAE-GS	20.45	6.52
MSVI [18]	22.29	6.12
InfoVAE [33]	22.73	6.14
β-VAE [59]	31.47	5.05
VAE	28.44	5.46
Wasserstein-Wasserstein Auto-Encoders [35]	25.63	5.79
MAE [34]	23.25	5.87

F. Evaluation of the representation learning

The representation learning ability is a very important property for deep learning models. We assume that the proposed mixture model is able to provide a rich representation of data, which would help to avoid overfitting because it embeds data into different regions of the latent space, as shown in Fig. 11. In order to measure the representation learning ability, we consider using a simple classifier on the latent representations extracted by various models. For the mixture model, we firstly extract features from each component and then concatenate these features into a single vector. The classifier is trained on the latent representations of the training data and then evaluated on a different dataset. We consider initially a simple network consisting of two layers as a basic classifier. The classification results, after training on the latent representations, are provided in Table IV, where we only compare with the best models, InfoVAE and MSVI, according to the results from the previous section. The results show that the proposed approach outperforms other methods by a large margin on CIFAR10 database. We achieve better results than the mixture model MSVI [18] which actually uses more parameters. This shows that the proposed model is able to provide a rich data representation. We also compare with the recently proposed GUIDE model [62].

TABLE IV
CLASSIFICATION RESULTS OF VARIOUS METHODS.

Dataset	InfoVAE	MSVI	MVAE-GS	MVAE-Gua	GUIDE
CIFAR10	42.92	49.48	**52.13**	51.78	-
MNIST	96.73	97.15	98.04	97.59	**98.15**

We also investigate the performance of three classic classifiers: Multilayer Perceptron (MLP), Linear Support Vector Machines (SVM) and K-nearest neighbours (KNN), which are trained on the latent representations extracted by each VAE component of the MVAE model. The results are reported in Table V, where C1 denotes that the classifier is trained on the representation extracted by the first component of MVAE-GS. This result demonstrates that the proposed model embeds data into several distinct latent subspaces, which enhances the performance in classification tasks.

TABLE V
THE RESULTS BY THREE CLASSIFIERS TRAINED ON THE LATENT VARIABLES INFERRED BY THE ENCODER WITH GAUSSIAN DROPOUT.

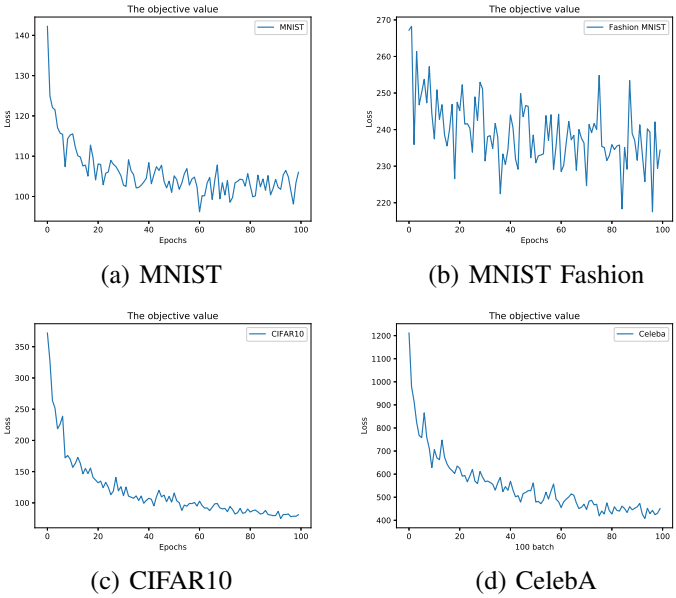
Classifier	Mixture	C1	C2	C3	C4	VAE
MLP	**98.04**	96.94	96.87	96.85	96.97	97.21
Linear SVM	**95.62**	93.65	93.81	93.83	93.67	93.81
KNN	**97.25**	96.91	96.98	96.85	96.15	96.51

G. Selecting the number of VAE components

The use of component dropout, defining the appropriate number of MVAE components, as discussed in Sections III-C and III-D, reduces the overfitting to the training set while also easing the requirements on computation and architecture complexity. The dropout masking vector $\mathbf{m}$, depends on the dropout rate p, which is learned during the training stage. We train the MVAE mixture model changing the initial number of VAE components by adopting different dropout mechanisms. The reconstruction error results for the MNIST dataset, when considering the selection of VAE components using the Gaussian dropout (MVAE-Gau) from (18), and the Gumbel-Softmax dropout (MVAE-GS) using equation (24) as the objective function, are provided in Table VI. The "Selected K" column represents the average number of VAE components required by the model calculated over all trials. When using the Gumbel-softmax trick from (21) to generate the dropout masking parameters m_i, $i = 1, \ldots, K$ for MVAE-GS we consider a threshold of 0.1, while removing all components with lower m_i's. When considering the MVAE-Gau algorithm, we set the threshold as 0.8 on the dropout masking parameter m_i, given that the sampling takes place from a Gaussian distribution with the mean of 1. We observe that the mixture model with all its components provides a lower MSE error than the MVAE model with dropout, except for the MVAE-Gau model. This is due to the fact that the proposed dropout method reduces overfitting. Furthermore, we also consider a mixture model with a single component, $K = 1$ as well as using a single VAE sharing the same network architecture and hyperparameters with the MVAE mixture model. It can be observed from Table VI that the performance of the mixture model with a single component is worse than all other architectures considered.

TABLE VI
THE RECONSTRUCTION ERROR ON THE MNIST DATASET WHEN USING COMPONENT DROPOUT.

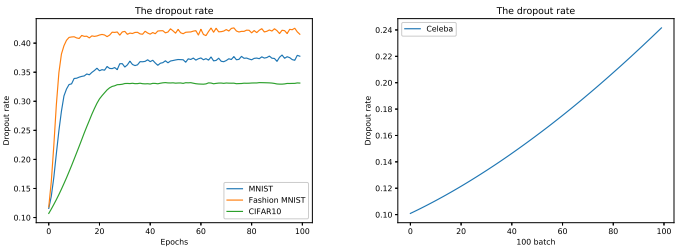
Model	Initial No of Components	Selected K	MSE
MVAE-GS	4	3.07	7.35
	4	4	7.42
	6	4.05	**7.30**
	6	6	7.41
	1	1	10.35
MVAE-Gau	4	4	9.52
	6	3.45	**7.53**
	6	6	7.89
	1	1	12.62
Single VAE	1	1	9.07



(a) MNIST (b) MNIST Fashion

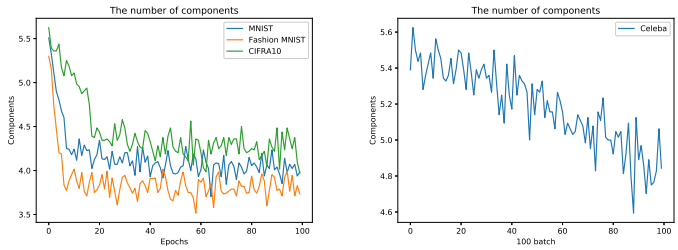
(c) CIFAR10 (d) CelebA

Fig. 14. The variation of the objective function $L_{MVAE-GS}$ during the training.



(a) MNIST, MNIST Fashion and CIFAR10 databases (b) CelebaA database.

Fig. 15. The variation of the dropout rate p during the training for MVAE-GS.



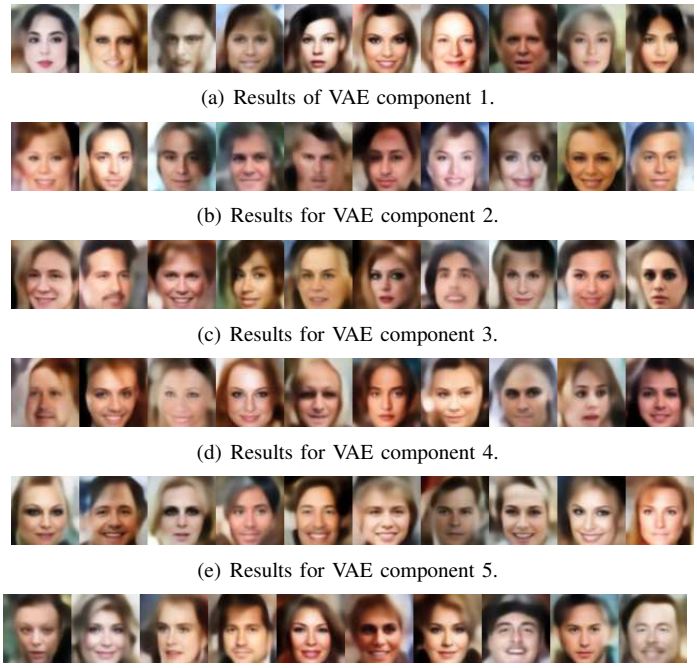
(a) MNIST, MNIST Fashion and CIFAR10 databases (b) CelebaA.

Fig. 16. The variation in the number of VAE components through updating the dropout parameter p as in (22), when initially considering $K = 6$ components, when using Gambel-softmax.

In the following we train a mixture model by setting initially $K = 6$ components and considering the VAE component dropout implemented using the Gumbel-Softmax approach (MVAE-GS), on four datasets: MNIST, MNIST-Fashion, CIFAR10 and CelebA. We consider 100 training epochs for the first three datasets. For the CelebA dataset, we measure the dropout rate for every 100 batches during one epoch. The variation of the objective function $L_{MVAE-GS}$ from equation (24), during the training for these databases, is shown in Figures 14a-d for each of the four databases: MNIST, MNIST Fashion, CIFAR10 and CelebA, respectively. The variation of the dropout p for MVAE-GS algorithm is provided in Fig. 15a for MNIST, Fashion and CIFAR10 databases. We observe that p initially increases quickly, while afterwards it becomes rather stable during the training for each of the first 3 databases. The result for CelebA database is shown in Fig. 15b, where a single epoch is considered for training, because this dataset is larger and more complex than the others. The dropout masking parameters m_i, $i = 1, \ldots, K$ depend on the dropout p, according to equation (22) for MVAE-GS. We consider a threshold of 0.1, on the estimated dropout parameter p, for removing a VAE component. The results for MNIST, Fashion and CIFAR10 are shown in Fig. 16a while for CelebA database are provided in Fig. 16b. It can be observed that MVAE requires more components for modelling and generating the images corresponding to the CIFAR10 dataset, when compared to the other datasets because this dataset contains more diverse images, displaying complex information, compared to the MNIST and MNIST Fashion databases.

H. Data generation diversity by each VAE component in MVAE

In order to show that each component learns different characteristics of the data, we train the MVAE model with $K = 6$ components on the CelebA dataset. Then we sample



(a) Results of VAE component 1.

(b) Results for VAE component 2.

(c) Results for VAE component 3.

(d) Results for VAE component 4.

(e) Results for VAE component 5.

(f) Results for VAE component 6.

Fig. 17. Generation results by each individual VAE component for the CelebA dataset for a set of random inputs.

a random vector from a Gaussian distribution, which is used as the input for each sub-decoder. The output of each sub-decoder is then fed into the final decoder, which outputs the generated images $\mathbf{x}'$. In Figures 17a-f, we show on each row the face images generated by each of the VAE components $i = 1, \ldots, 6$, where the images from each column correspond to the same input random vector, sampled from a Gaussian distribution. From the results from Figures 17 we can observe that each component outputs different human faces or different appearances for the same face thus showing the ability for MVAE to generate diverse images.

VI. Conclusions

In this research study we propose a mixing deep learning model using collections of variational encoders and sub-decoders, called the Mixture of Variational Autoencoders (MVAE). The latent space of each VAE component captures specific characteristics of data in different ways providing rich latent representations benefiting many tasks. These properties result in enhanced abilities for data generation by MVAE model. Each sub-decoder has a simple design consisting of a single layer network benefiting from quick training, while the mix-decoder is implemented by a deeper CNN. The separability between the latent spaces, corresponding to each VAE, is enforced by using the d-variable Hilbert-Schmidt independence (dHSIC) criterion. Each component of MVAE models a distinct latent space, avoiding the overfitting which occurs in other models. We also consider a component dropout mechanism in order to select the appropriate number of VAE components in MVAE. The training of MVAE involves the estimation of the parameters for the encoders, sub-decoders, implementing the dHSIC criterion, the Dirichlet sampling for the mixing weights, the component dropout procedure and the mix-decoder parameters into an end-to-end training procedure using stochastic gradient descent (SGD). A variety of data manipulations, including interpolations in the joint latent spaces of the VAE components, show the capabilities of the proposed MVAE model.

References

[1] A. G. Bors and M. Gabbouj, "Minimal topology for a radial basis functions neural network for pattern classification," *Digital Signal Processing*, vol. 4, no. 3, pp. 173–188, 1994.

[2] A. G. Bors and I. Pitas, "Median radial basis function neural network," *IEEE Trans. on Neural Networks*, vol. 7, no. 6, pp. 1351–1364, 1996.

[3] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. on Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.

[4] L. Weruaga and J. Via, "Sparse multivariate Gaussian mixture regression," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1098–1108, 2015.

[5] E. J. Hartman, K. J. D., and M. Kowalski, J, "Layered neural networks with Gaussian hidden units as universal approximations," *Neural Computation*, vol. 2, no. 2, pp. 210–215, 1990.

[6] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[7] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. the IEEE*, vol. 9, no. 78, pp. 1481–1497, 1990.

[8] H. Attias, "A variational Bayesian framework for graphical models," in *Advances in Neural Inf. Proc. Systems (NIPS)*, 2000, pp. 209–215.

[9] Z. Ghahramani and M. Beal, "Variational inference for Bayesian mixtures of factor analysers," in *Advances in Neural Inf. Proc. Systems (NIPS)*, 2000, pp. 449–455.

[10] N. Nasios and A. G. Bors, "Variational learning for Gaussian mixture models," *IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 4, pp. 849–862, 2006.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013. [Online]. Available: https://arxiv.org/abs/1312.6114

[12] N. Dilokthanakul, P. Mediano, M. Garnelo, M. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with Gaussian mixture variational autoencoders," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1611.02648

[13] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep spectral clustering using dual autoencoder network," in *Proc. IEEE Conf. on Pattern Recognition and Computer Vision (CVPR)*, 2019, pp. 4066–4075.

[14] A. Klushyn, N. Chen, R. Kurle, B. Cseke, and P. van der Smagt, "Learning hierarchical priors in VAEs," in *Advances in Neural Inf. Proc. Systems (NeurIPS)*, 2019, pp. 2870–2879.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Inf. Proc. Systems (NIPS)*, 2014, pp. 2672–2680.

[16] L. Chen, S. Dai, Y. Pu, C. Li, Q. Su, and L. Carin, "Symmetric variational autoencoder and connections to adversarial learning," in *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS), vol. PMLR 84*, 2018, pp. 661–669.

[17] Y. Fei and A. G. Bors, "Learning joint latent representations based on information maximization," *Information Sciences*, 2021.

[18] R. Kurle, S. Günnemann, and P. van der Smagt, "Multi-source neural variational inference," in *Proc. of AAAI Conf. on Artificial Intelligence*, vol. 33, 2019, pp. 4114–4121.

[19] E. Abbasnejad, M. Dick, and A. van der Hengel, "Infinite variational autoencoder for semi-supervised learning," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5888–5897.

[20] Z. Szabó and B. K. Sriperumbudur, "Characteristic and universal tensor product kernels," *Journal of Machine Learning Research*, vol. 18, no. 233, pp. 1–29, 2018.

[21] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[22] C. Cremer, X. Li, and D. Duvenaud, "Inference suboptimality in variational autoencoders," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1801.03558

[23] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proc. Int. Conf. on Machine Learning (ICML), vol. PMLR 32(2)*, 2014, pp. 1278–1286.

[24] R. van den Berg, L. Hasenclever, J. M. Tomczak, and M. Welling, "Sylvester normalizing flows for variational inference," in *Proc. Uncertainity in Artificial Intelligence (UAI)*, 2018, pp. 393–402.

[25] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improving variational inference with inverse autoregressive flow," in *Advances in Neural Inf. Proc. Systems (NIPS)*, 2016, pp. 4743–4751.

[26] J. M. Tomczak and M. Welling, "Improving variational auto-encoders using householder flow," 2016. [Online]. Available: https://arxiv.org/abs/1611.09630

[27] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. Int. Conf. on Machine Learning (ICML), vol. PMLR 37*, 2015, pp. 1530–1538.

[28] R. Ranganath, D. Tran, and D. Blei, "Hierarchical variational models," in *Proc. Int. Conf. on Machine Learning (ICML), vol. PMLR 48*, 2016, pp. 324–333.

[29] T. Salimans, D. P. Kingma, and M. Welling, "Markov chain Monte Carlo and variational inference: Bridging the gap," in *Proc. Int. Conf. on Machine Learning (ICML), vol. PMLR 37*, 2015, pp. 1218–1226.

[30] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, "Auxiliary deep generative models," in *Proc. Int. Conf. on Machine Learning (ICML) vol. PMLR 48*, 2016, pp. 1445–1453.

[31] J. Domke and D. R. Sheldon, "Importance weighting and variational inference," in *Advances in Neural Inf. Proc. Systems (NeurIPS)*, 2018, pp. 4470–4479.

[32] Y. Pu, W. Wang, R. Henao, C. L., Z. Gan, C. Li, and L. Carin, "Adversarial symmetric variational autoencoder," in *Advances in Neural Inf. Proc. Systems (NIPS)*, 2017, pp. 4333–4342.

[33] S. Zhao, J. Song, and S. Ermon, "InfoVAE: Balancing learning and inference in variational autoencoders," in *Proc. AAAI Conf. on Artif. Intel.*, vol. 33, 2019, pp. 5885–5892.

[34] X. Ma, C. Zhou, and E. Hovy, "MAE: Mutual posterior-divergence regularization for variational autoencoders," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2019. [Online]. Available: https://arxiv.org/abs/1901.01498

[35] S. Zhang, Y. Gao, Y. Jiao, J. Liu, Y. Wang, and C. Yang, "Wasserstein-Wasserstein auto-encoders," 2019. [Online]. Available: https://arxiv.org/abs/1902.09323

[36] Y. Fei and A. G. Bors, "Mixtures of variational autoencoders," in *Proc. Int. Conf. on Image Processing Theory, Tools and Applic. (IPTA)*, 2020.

[37] D. M. Endres and J. E. Schindelin, "A new metric for probability distributions," *IEEE Trans. on Information Theory*, vol. 49, no. 3, pp. 1858–1860, 2003.

[38] N. Pfister, P. Bühlmann, B. Schölkopf, and J. Peters, "Kernel-based tests for joint independence," *Jour. of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 80, no. 1, pp. 5–31, 2018.

[39] R. Lopez, J. Regier, M. I. Jordan, and N. Yosef, "Information constraints on auto-encoding variational Bayes," in *Advances in Neural Inf. Proc. Systems (NeurIPS)*, 2018, pp. 6117–6128.

[40] W. He, H. Huang, and S. Ge, "Estimating the dimension of a model," *Annals of Statistics*, vol. 7, no. 2, pp. 461–464, 1978.

[41] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*. World Scientific, 1989.

[42] D. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in Neural Inf. Proc. Systems (NIPS)*, 2015, pp. 2575–2583.

[43] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2017. [Online]. Available: https://arxiv.org/abs/1611.01144

[44] B. E. J. Gumbel, *Statistical theory of extreme values and some practical applications:a series of lectures*, 1954.

[45] C. Maddison, D. Tarlow, and T. Minka, "A* sampling," *Advances in Neural Inf. Processing Systems (NIPS)*, pp. 3086–3094, 2014.

[46] E. J. Gumbel, "Bivariate logistic distributions," *Journal American Statistical Association*, vol. 56, pp. 335–349, 1961.

[47] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in Neural Inf. Processing Systems (NIPS)*, 2017, pp. 3581–3590.

[48] M. Figurnov, S. Mohamed, and A. Mnih, "Implicit reparameterization gradients," in *Advances in Neural Inf. Proc. Systems (NeurIPS)*, 2018, pp. 439–450.

[49] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. [Online]. Available: https://arxiv.org/abs/1609.04747

[50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[51] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, 2015, pp. 3730–3738.

[52] A. van Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proc. Int. Conf. on Machine Learning (ICML), vol. PMLR 48*, 2016, pp. 1747–1756.

[53] G. Loaiza-Ganem and J. P. Cunningham, "The continuous Bernoulli: fixing a pervasive error in variational autoencoders," in *Advances in Neural Inf. Proc. Systems (NeurIPS)*, 2019, pp. 13 266–13 276.

[54] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2015. [Online]. Available: https://arxiv.org/abs/1511.06434

[55] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially learned inference," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2017. [Online]. Available: https://arxiv.org/abs/1606.00704

[56] D. Warde-Farley and Y. Bengio, "Improving generative adversarial networks with denoising feature matching," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2017.

[57] T. Salimans, A. Karpathy, X. Chen, and D. Kingma, "PixelCNN++: Improving the pixelCNN with discretized logistic mixture likelihood and other modifications," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2017. [Online]. Available: https://arxiv.org/abs/1701.05517

[58] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary equilibrium generative adversarial networks," 2017. [Online]. Available: https://arxiv.org/abs/1703.10717

[59] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. Inter. Conf. on Learning Representations (ICLR)*, 2017.

[60] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. of Toronto, Tech. Rep., 2009.

[61] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Inf. Proc. Systems (NIPS)*, 2016, pp. 2234–2242.

[62] Z. Ding, Y. Xu, W. Xu, G. Parmar, Y. Yang, M. Welling, and Z. Tu, "Guided variational autoencoder for disentanglement learning," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 7920–7929.

Fei Ye is a currently third-year PHD student in computer science from University of York. He received the bachelor degree from Chengdu University of Technology, China, in 2014 and the master degree in computer science and technology from Southwest Jiaotong University, China, in 2018. His research topic includes a deep generative image model, lifelong learning and mixture models.

Adrian G. Bors received the MSc degree in Electronics Engineering from the Polytechnic Univ. of Bucharest, Romania, in 1992, and the Ph.D. degree in Informatics from the Univ. of Thessaloniki, Greece in 1999. In 1999 he joined the Department of Computer Science, Univ. of York, U.K., where he is currently a lecturer. Dr. Bors held temporary positions at Tampere Univ. of Technology, Finland, Univ. of California at San Diego (UCSD), and at the Univ. of Montpellier, France. Dr. Bors has authored and co-authored more than 130 research papers including 30 in journals. His research interests include computer vision, computational intelligence and image processing. Dr. Bors was an associate editor of IEEE Trans. Image Processing between 2010 and 2014 and of IEEE Trans. Neural Networks from 2001 to 2009. He was a co-guest editor for a special issue on Machine Vision for the International Journal for Computer Vision in 2018, and for the Journal of Pattern Recognition in 2015. Dr. Bors was a member of the organisation committees for IEEE WIFS 2021, IPTA 2020, IEEE ICIP 2018, BMVC 2016, IPTA 2014, CAIP 2013 and IEEE ICIP 2001. He is a Senior Member of the IEEE.