Abstract—Variational autoencoders (VAEs) are one of the most popular unsupervised generative models that rely on learning latent representations of data. In this article, we extend the classical concept of Gaussian mixtures into the deep variational framework by proposing a mixture of VAEs (MVAEs). Each component in the MVAE model is implemented by a variational encoder and has an associated subdecoder. The separation between the latent spaces modeled by different encoders is enforced using the d-variable Hilbert–Schmidt independence criterion (dHSIC). 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 MVAE model can learn a rich latent data representation and is able to discover additional underlying data representation factors.

Index Terms—Generative deep learning, mixtures of variational autoencoders (MVAEs), optimal number of components in mixtures, representation learning.

I. INTRODUCTION

Research in deep learning focused initially on addressing supervised classification problems, where training data are labeled. 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 function (RBF) networks by adding a second layer of linear processing units for supervised classification [1]–[4]. RBF networks have been shown to have universal approximation properties [5]–[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]–[10].

Lately, generative deep models have gained increasing attention from the research community. The variational autoencoder (VAE) [11] consists of two convolution neural networks (CNNs): the encoder and the decoder. The decoder is used to approximate the conditional distribution \( p(x|z) \) for reconstructing the data \( x \) from the estimated latent space \( z \). Meanwhile, the latent space is modeled by a variational distribution \( q(z|x) \), 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 2-D 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-regularization in data representation [12]. In this work, we address these problems by developing a novel mixture of VAEs (MVAEs), 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 nonlinear 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 that 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, enabling a rich data representation, benefiting many downstream tasks. The proposed model provides a competitive performance when compared with the state-of-the-art VAE frameworks.

The rest of this article 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 in Section V, and the conclusions are drawn in Section VI.

II. RELATED WORK

A. Probabilistic Mixture Model

The 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 EM algorithm was used to find a set of hyper-parameters for the variational GMM model [10]. Gaussian distributions are used to model the distribution of the means of each Gaussian component of the mixture, Wishart distributions are used for their corresponding covariance matrices, and Dirichlet is used for the mixing parameters.

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

B. Variational Autoencoder

A 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 \( x \) be a data sample vector and \( z \) be a vector of stochastic latent variables. While the encoder maximizes \( p_{\theta}(z|x) \), the decoder in VAE implements the variational posterior \( q_{\phi}(x|z) \), where \( \eta \) and \( \theta \) represent the parameters of the CNNs implementing the encoder and decoder, respectively. The evidence lower bound (ELBO) on the marginal log-

$$
\mathbb{E}_x \left[ \log p(x) \right] \geq \mathbb{E}_x \left[ \log p(x|z) \right] + \log p(z) - \log q_{\theta}(z|x)
$$

(1)

where \( X \) is the empirical distribution characterizing the data and \( p_{\theta}(x|z) \) is a generative model implemented by the decoder of parameters \( \eta \), while \( p(z) \) is the prior distribution of the latent space, usually a Gaussian distribution with the identity matrix as its covariance. The reparameterization 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-hand 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]–[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. Info-VAE [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 autoencoders, 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 multisource 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] developed a model considering a GMM as prior distribution for unsupervised clustering tasks. This model still suffers from over-regularization. MVAEs 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 MVAEs 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 the dHSIC regularization, which overcomes the over-regularization problem characteristic in VAEs [11].

III. 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 modeling complex data, we introduce the 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 that the number of components in the MVAE model is explained in Sections III-C and III-D.

A. 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 $p_{θ_i}(x|z_i)$ the variational posterior for the $i$th decoder, implemented by a CNN of parameters $θ_i$, where $z_i$ represents the inferred stochastic latent variable vector, for $i = 1, . . . , K$, where $K$ is the number of VAE components.

The data generation process is defined by the following stages. The latent variable $z_i$ is yielded by the $i$th encoder

$$z_i \sim N(μ_{θ_i}, σ_{θ_i})$$  \hspace{1cm} (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 first consider a simple network as a subdecoder, implemented by a nonlinear transformation function denoted as $t_i(·)$, $i = 1, . . . , K$, which outputs a variable $s_i$.

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

$$\{w_1, \ldots, w_K\} \sim \text{Dir}(a)$$  \hspace{1cm} (3)

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

$$s = \sum_{i=1}^{K} w_is_i$$  \hspace{1cm} (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 $s$

$$p(s|x) = \sum_{i=1}^{K} w_it_i\left(\int p(θ_i(z|x))dx\right)$$  \hspace{1cm} (5)

where $θ_i$ represents the parameters of each encoder network $i = 1, . . . , K$. Afterward, in the mixture model, we have a single mix-decoder for reconstructing the data $x'$

$$x' \sim p(x|s).$$  \hspace{1cm} (6)

For the generation process, each $i$th subdecoder 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 $q_θ(z, w|x)$ implemented by an independent encoder. We use Jensen’s inequality to obtain the variational lower bound as follows:

$$\log p(x) = \log E_{q_θ(z,w|x)} \left[ \frac{p(x, z, w)}{q_θ(z, w|x)} \right]$$

$$\geq E_{q_θ(z,w|x)} \left[ \log \frac{p(x, z, w)}{q_θ(z, w|x)} \right].$$  \hspace{1cm} (7)

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

$$L_{\text{MELBO}} = -E_{q_θ(z,w|x)}[\log p_θ(x|z, w)] + E_{q_θ(z,w|x)} \log \left[ \frac{p(z)}{q(z|x)} \right] + E_{q_θ(z,w|x)} \log \left[ \frac{p(w)}{q(w|x)} \right]$$

$$= -E_{q_θ(z,w|x)}[\log p_θ(x|z, w)] + D_{\text{KL}}(q_θ(z|x)||p(z)) + D_{\text{KL}}(q_θ(w|x)||p(w))$$  \hspace{1cm} (8)

where $p(w)$ and $p(z)$ are the priors for the mixing parameters $w$ and for the latent variables $z = \{z_i|i = 1, . . . , K\}$, while $θ = \{θ_i|i = 1, . . . , K\}$, $η = \{η_i|i = 1, . . . , K\}$, represent the parameters of the networks modeling the mixture of encoders and decoders, respectively. After explicitly expanding the basic objective function corresponding to the mixture model,
we have
\[ L_{\text{MELBO}} = -E_{p(s|z)}[\log p_f(x|s)] + D_{KL}(q_0(w|x)||p(w)) + \frac{1}{K} \sum_{i=1}^{K} D_{KL}(q_{\theta_i}(z|x)||p(z)) \] (9)

where \( K \) is the number of components, \( q_0(z|x) \) represents the variational modeled by one of the mixing components, defined by the parameters \( \theta_i \), and \( p_f(x|s) \) is the probability of the mixture modeled by a network defined by the parameters \( \eta_i \), aiming to reconstruct the data \( x' \). Let us consider \( t_{yi}(s_i|z) \) the function of each subdecoder, where \( \{t_{yi}, i = 1, \ldots, K\} \) are the parameters of \( K \) subdecoders. The output of each subdecoder is multiplied by its corresponding mixing parameter, and the results are then summed up like in (5).

### B. Enforcing the Separation in the Latent Space Among the MVAE Components

\( L_{\text{MELBO}} \) from (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 \( r(z) \)

\[ L_{\text{Obj}} = L_{\text{MELBO}} + \beta \cdot r(z) \] (10)

where \( L_{\text{Obj}} \) is the objective function of MVAE and \( \beta \) 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 subdecoder 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 \( r(z) \), 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 and 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

\[ r(z) = \sum_{i=1}^{K} \sum_{j=1, j \neq i}^{K} d(p(z_i), p(z_j)), \quad i \neq j \] (11)

where \( d(\cdot, \cdot) \) represents one of the discriminatory measures mentioned above, evaluated between the probabilistic representations \( p(z_i) \) and \( p(z_j) \), 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 MVAEs is equal to the product of marginal probabilities of the individual variables [38]

\[ q(z_1, z_2, \ldots, z_K) = \prod_{i=1}^{K} q(z_i) \] (12)

where \( q(z_i) \) is the marginal probability of the latent variable \( z_i \), which represents the output of the \( i \)th encoder. Let us consider the cross-covariance operator \( C_{z_i, z_j} \), defined on the encoders output variables \( z_i \) and \( z_j \). The largest eigenvalue of the operator \( C_{z_i, z_j} \) measures the dependence score between the distributions defined by the latent variables \( z_i \) and \( z_j \) 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]:

\[ C_{z_i, z_j} = E_{z_i, z_j}[k(z_i), l(z_j)] - E_{z_i}[k(z_i)]E_{z_j}[l(z_j)] \] (13)

where \( z_i \) and \( z_j \) are the latent variables produced by \( i \)th and \( j \)th encoders, respectively, and \( k(\cdot) \) and \( l(\cdot) \) 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

\[ \text{HSIC}(z_i, z_j) = \|C_{z_i, z_j}\|^2 \] (14)

where \( C_{z_i, z_j} \) is provided in (13).

The definition of the cross-covariance operator can be extended for assessing the independence of \( d \) variables, similar to the expressions from (13) and (14), representing the latent space \( \{z_1, z_2, \ldots, z_d\} \) 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 \( z_i, i = 1, \ldots, K \) 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 \( r(z) \) in the optimization function \( L_{\text{Obj}} \) from (10), as follows:

\[ L_{\text{Obj}} = L_{\text{MELBO}} + \beta \cdot \text{dHSIC}(z_1, z_2, \ldots, z_K) \] (15)

where \( L_{\text{MELBO}} \) defines the inference process for MVAE and \( \beta \) 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 variational EM 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 $K$ subdecoders. 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

\[
b_i = \frac{m_i w_i}{\sum_{j=1}^{K} m_j w_j}.
\]

(16)

Then, the output of each subdecoder is multiplied by the corresponding mixing parameter and the probability of the $m$th variable 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 modeling the masking. However, this is nondifferentiable 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 $\tau = p/(1 - p)$, with $p$ 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 considered in the objective function $L_{\text{Obj}}$ 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:

\[
L_{\text{MVAE-Gau}} = -\mathbb{E}_{p(\mathbf{s}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{s})] + D_{KL}(p_{\theta}(\mathbf{w}|\mathbf{x})||p(\mathbf{w})) + \frac{1}{K} \sum_{i=1}^{K} D_{KL}(q_{\theta}(\mathbf{z}_i|\mathbf{x})||p(\mathbf{z})) + \beta d\text{HSIC}(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K) + D_{KL}(q(\mathbf{m})||p(\mathbf{m})).
\]

(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 by using the following polynomial expression:

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

(19)

where $\tau$ defines the variance of the Gaussian distribution used for sampling the dropout, and the constants used in the polynomial approximation while $c$, $c_1$, $c_2$, and $c_3$ 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

\[
\text{one_hot}\left(\arg \max_i (\log a_i + g_i)\right)
\]

(20)

where $g_i$ is a sample drawn from the Gumbel(0, 1) distribution. The Gumbel-softmax trick [44], [45] adopts the softmax function as a continuous, differentiable approximation to the arg max expression as [43]

\[
m_i = \frac{\exp((\log(a_i) + g_i)/T)}{\sum_{j=1}^{K} \exp((\log(a_j) + g_j)/T)}
\]

(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:

\[
m_i = \frac{\exp((\log(1 - p) + g_i)/T)}{\exp((\log(1 - p) + g_1)/T) + \exp((\log(1 - p) + g_2)/T)}
\]

(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 $i$. $g_1, g_2 \sim \text{Gumbel}(0, 1)$, [46], and $p$ 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 (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 MVAEs with the 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. [47] considered the dropout regularization term as the entropy of a Bernoulli random variable

\[
H(p) = -p \log p - (1 - p) \log(1 - p)
\]

(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

$$L_{\text{MVAE-GS}} = -E_{p(x|z)}[\log p(x|z)] + D_{KL}(p(\mathbf{w}|x)||p(\mathbf{w})) + \frac{1}{K} \sum_{i=1}^{K} D_{KL}(q_{\theta_i}(z|x)||p(z)) + \beta \text{dHSIC}(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K) - H(p)$$

(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. MVAE Model Structure

The proposed mixture model is based on a collection of encoders and subdecoders 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 subdecoder 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 modeled by the corresponding hyperparameters. In order to allow the gradients to be estimated from the encoder to subdecoder, we use the reparameterization trick [11]

$$\mathbf{z}_i = \mu_i + \sigma_i \odot \epsilon$$

(25)

where $\epsilon \sim \mathcal{N}(0, I)$ is a random variable sampled from the Gaussian distribution of mean 0, 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 $\text{Dir}(a_1, a_2, \ldots, a_K)$, with each parameter produced by one of the encoders

$$\left( \frac{\omega_1}{\sum_{j=1}^{K} \omega_j}, \ldots, \frac{\omega_K}{\sum_{j=1}^{K} \omega_j} \right) \sim \text{Dir}(a_1, \ldots, a_K)$$

(26)

where the sum of the mixing parameters is 1. The mixing parameters are then multiplied by the dropout parameters that 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 subdecoders, we show that it can be easily trained by using the SGD algorithm [49] when considering either the objective function from (18) for MVAE-Gau or that from (24) for MVAE-GS, depending on what component dropout procedure is adopted for the mixing model. Similar 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 $D_{KL}(q_{\theta_i}(z|x)||p(z))$, between the prior and posterior for each $i$th encoders, is calculated considering Gaussian pdfs in the latent space as

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

(27)
where $S_i$ is the dimension of the latent variable space $z$ 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 $D_{KL}(q_\theta(w|x)||p(w))$ 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 $a = \{a_1, \ldots, a_K\}$ and $b = \{b_1, \ldots, b_K\}$ as

$$D_{KL}(q_\theta(a|x)||p(b)) = \log \Gamma(a_0) - \sum^K_{i=1} \log \Gamma(a_i)$$

$$- \log \Gamma(b_0) + \sum^K_{i=1} \log \Gamma(b_i)$$

$$+ \sum^K_{i=1} (a_i - b_i)(\psi(a_i) - \psi(a_0))$$  \hspace{1cm} (28)

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

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

$$L_{Rec} = \frac{1}{2} \sum^N_{i=1} \|x'_i - x_i\|_F^2$$ \hspace{1cm} (29)

where $x'$ 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 (18), derived in Section III-C, is given by

$$\nabla_{\Omega} L_{MVAE-Gau} \left[ L_{Rec} + \sum^K_{i=1} D_{KL}(q_\theta(z|x)||p(z)) \right.$$ \hspace{1cm} (30)

$$\left. + D_{KL}(q_\theta(a|x)||p(b)) + \beta \text{dHSIC}(z_1, \ldots, z_K) \right]$$

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

$$\nabla_{\Omega} L_{MVAE-GS} \left[ L_{Rec} + \sum^K_{i=1} D_{KL}(q_\theta(z|x)||p(z)) \right.$$ \hspace{1cm} (31)

$$\left. + D_{KL}(q_\theta(a|x)||p(b)) + \beta \text{dHSIC}(z_1, \ldots, z_K) - H(p) \right]$$

with the last term provided in (23).

The parameters being updated in the MVAE model are

$$\Omega = \{\theta, \gamma, \nu, \eta\}$$ \hspace{1cm} (32)

where $\theta$ and $\gamma$ are the parameters of the mixture of $K$ encoders $q_\theta(z|x)$ and subdecoders $p_\gamma(s|z)$, respectively, and $\nu$ and $\eta$ are the parameters for the mix-decoder network, while $\nu$ 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 $X' = \{x^1, x^2, \ldots, x^N\}$ from training dataset
2: While epoch $< epoch_{\text{max}}$ do
3: While batch $< batch_{\text{max}}$ do minibatch procedure
4: $x_{\text{batch}} = \text{Select (epoch, } X')$ batch samples
5: $(\mu_1, \sigma_1, a_1) \ldots (\mu_k, \sigma_k, a_k) \leftarrow E_1(x_{\text{batch}}), \ldots, E_k(x_{\text{batch}})$
6: $\{z_1, \ldots, z_k\} \leftarrow \text{latent variables from all encoders}$
7: $(a_1, \ldots, a_k) \leftarrow \text{Dirichlet parameters from all encoders}$
8: $(w_1, \ldots, w_k) \leftarrow \text{Dir} (a_1, \ldots, a_k)$ Sampling mixing weights
9: $\{s_1, \ldots, s_k\} \leftarrow t_1 (z_1) \ldots t_k (z_k)$ Transform by sub-decoders
10: $GS(p,k) \sim (m_1, \ldots, m_k)$ Dropout from Gumbel-Softmax
11: $b_i = \sum_{j=1}^{m_i} s_i/b_i$ Dropout many components
12: $s = \sum_{j=1}^{m_i} s_i/b_i$ Integral outputs of all components
13: $x' = p(x|s)$ reconstruct data by mix-decoder
14: Calculate reconstruction error using (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)
19: End
20: End
21: End

V. EXPERIMENTAL RESULTS

In this section, we assess the results provided by the proposed MVAEs model on a variety of data sets. Each component of the MVAE model is represented by an encoder and a subdecoder. The probabilistic models are implemented by using fully connected and convolutional networks, depending
on the complexity of the data set. 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 Data Set**

The MNIST data set [50] consists of images of handwritten digits of size 28 \times 28 pixels. We train MVAE with \( K = 4 \) components when using the MVAE-Gau loss, defined in (18), considering 60000 and 100000 images for training and testing, respectively. A set of original MNIST images are shown in Fig. 2(a), and their reconstruction by MVAE is shown in Fig. 2(f), while the reconstructions by each of the four components are shown in Fig. 2(b)–(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 subdecoder than its corresponding encoder during the tests. For the images of digits from Fig. 3(a), we consider reconstructions by mismatching the subencoders and encoders for the same images. The results of such mismatches are shown in Fig. 3(b)–(e), and we observe that these images are not properly reconstructed. This happens because each subdecoder is uniquely fitted to the associated encoder and not to any of the others, which would yield different latent space representations.

**B. Representing Complex Images**

In the section, we evaluate the performance of the proposed MVAE mixture model on some databases that contain more complex images, such as the human face data set entitled Celebrities Faces Attributes (CelebA) [51] and ImageNet database [52]. The CelebA data set contains almost 200000 human face images with 10177 identities. Each encoder of the mixture model is implemented by a deep convolution net consisting of five 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 subdecoder is implemented by a simple architecture with only a single layer of \( 8 \times 8 \times 256 \). The output of the subdecoder is then transformed into a 3-D 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 seven layers, which receives the tensor as the input and generates images as the output. We set the kernel size as \( 3 \times 3 \) 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 ten epochs with a learning rate of 0.001. A set of face images from the CelebA data set is shown in Fig. 4(a), while their reconstructions by MVAE-Gau are shown in Fig. 4(b). We also show the reconstruction results in Fig. 5(c) for MVAE-Gau for the subset of images from ImageNet database [52], as shown in Fig. 5(a). For comparison, the same images are reconstructed by MSVI [18] in Fig. 5(b). It can be observed that both human face and natural images are reconstructed well by the MVAE-Gau model.

---

![Fig. 2. Reconstructed images from the MNIST data set by MVAE and its individual VAE components. (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.](image1)

![Fig. 3. Results when mismatching the encoders and subdecoders when reconstructing images from the MNIST data set. (a) Real test samples. (b) Results when using the encoder 1 with the latent variables of subdecoder 2. (c) Results when using the encoder 2 with the latent variables of subdecoder 3. (d) Results when using the encoder 3 with the latent variables of subdecoder 4. (e) Results when using the encoder 4 with the latent variables of subdecoder 3.](image2)

![Fig. 4. Reconstruction results for face images from the CelebA data set [51]. (a) Randomly selected images. (b) Reconstructed face images by MVAE-Gau.](image3)
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 \( \{x_1, x_2\} \) is randomly selected and we map these into the latent representation by using the trained encoders. We consider \( K = 6 \) components, and the interpolation \( d_i \) is performed on the output of each encoder as

\[
d_i = (1 - a) E_i(x_1) + a E_i(x_2) \tag{33}
\]

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 subdecoders \( \text{SubDec}_i(\cdot) \) into

\[
c = \sum_{i=1}^{6} w_i \cdot \text{SubDec}_i(d_i) \tag{34}
\]

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

\[
x' = \text{MixDec}(c). \tag{35}
\]

Interpolation results on the CelebA data set are shown in Fig. 6, where \( \{x_1, x_2\} \) 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 that 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.

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 subdecoder, 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 subdecoder, replacing (34) with

\[
\hat{c} = \sum_{i=1}^{6} w_i \cdot \text{SubDec}_i(d_j) \tag{36}
\]

where \( j \neq i \). For example, the first subdecoder 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 \( \{x_1, x_2\} \) are shown as the first and last on each row, while the images in between are the interpolation results.
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. (a) Component 1. (b) Component 2. (c) Component 3. (d) Component 4. (e) Component 5. (f) Component 6.

Fig. 11. Representation of the latent space for the MV AE model for the MNIST data set. (a) Component 1. (b) Component 2. (c) Component 3. (d) Component 4.

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 modeled 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.

### C. Latent Space Analysis

In the following, we explore the latent space representation for the MVAE model. Each encoder and associated subdecoder considers jointly the data and their corresponding class labels $\{x, y\}$, where the class information $y$ 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 data set. 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 Fig. 8(a)–(d) for each of the four components considered. Meanwhile, the generated images of the handwritten digit “5,” for $K = 6$ components, when changing either the variable $z_1$ or $z_2$ from the latent space are shown in Figs. 9(a)–(f) and 10(a)–(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 colors 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 modeled by the encoders, through experiments following the training of MVAE-GS on the
MNIST data set. 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 shown in Fig 12(a) and (b), respectively. We observe that when using a larger $\beta$, 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 Fig. 13(a) and (b), 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. 13(a) 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. 13(b). 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.

First, 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 subdecoder implemented by a single layer instead of an entire deep convolution net. Second, InfoVAE [33] is the current state-of-the-art VAE framework, which is able to balance accurate inference with the reconstruction quality. We implement InfoVAE by using a large network architecture. Third, $\beta$-VAE is a variant of the VAE framework which aims to learn disentangled representations. Fourth, we also compare with a single VAE [11] implemented by a large network architecture. Finally, we consider for comparison several other recent VAE frameworks, such as those proposed in [34], [35], and [53].

We evaluate the reconstruction ability of the proposed approach on CIFAR10 data set [60], which contains 60,000 images grouped into ten 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]

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

where $D_{KL}$ represents the KL divergence between the distributions of the labels of the recovered images and those from the database, $x$ represents the image, $p(y|x)$ is the probability of the softmax output for the trained classifier, and $p^*(y)$ represents the labels statistics for the given images. The reconstruction root-mean-square error (RMSE) as well as the IS for CIFAR10 and CIFAR100 databases are shown 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 $\bar{K} = 4.11$. 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 data set, 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.

**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 first 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 data set. We initially consider a simple network consisting of two layers as a basic classifier. The classification results, after training on the latent representations, are shown 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 the 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].

We also investigate the performance of three classic classifiers: multilayer perceptron (MLP), linear support vector machines (SVM), and K-nearest neighbors (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.

**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 $m$ 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 data set when considering the selection of VAE components using the Gaussian dropout (MVAE-Gau) from (18) and the Gumbel-Softmax dropout (MVAE-GS) using (24) as the objective function are shown 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 $m_i$ values. 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.

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 data sets: MNIST, MNIST-Fashion, CIFAR10, and CelebA. We consider 100 training epochs for the first three data sets. For the CelebA data set, we measure the dropout rate for every 100 batches during one epoch. The variation of the objective function $L_{MVAE-GS}$ from (24) during the training for these databases is shown in Fig. 14(a)–(d) for each of the four databases: MNIST, MNIST Fashion, and CIFAR10 databases. The variation of the dropout $p$ for the MVAE-GS algorithm is shown in Fig. 15(a) for MNIST, Fashion, and CIFAR10 databases. We observe that $p$ initially increases quickly, while afterward, it becomes rather stable during the training for each of the first three databases. The result for CelebA database is shown in Fig. 15(b), where a single epoch is considered for training, because this data set is larger and more complex than the others. The variation of the dropout $p$ for the MVAE-GS algorithm is shown in Fig. 15(a) for MNIST, Fashion, and CIFAR10 databases. We observe that $p$ initially increases quickly, while afterward, it becomes rather stable during the training for each of the first three databases. The result for CelebA database is shown in Fig. 15(b), where a single epoch is considered for training, because this data set 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 (22) for MVAE-GS. We consider a threshold of 0.1, on the estimated dropout $p$, for removing a VAE component. The results for MNIST, Fashion, and CIFAR10 are shown in Fig. 16(a), while the results for CelebA database are shown in Fig. 16(b). It can be observed that MVAE requires more components for modeling and generating the images corresponding to the
CIFAR10 data set, when compared to the other data sets because this data set 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 data set. Then, we sample a random vector from a Gaussian distribution, which is used as the input for each subdecoder. The output of each subdecoder is then fed into the final decoder, which outputs the generated images $\mathbf{x}$. In Fig. 17(a)-(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 Fig. 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. CONCLUSION

In this research study, we propose a mixing deep learning model using collections of variational encoders and subdecoders, as well as a mix-decoder, called the 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 the MVAE model. Each subdecoder 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 $d$-variable Hilbert-Schmidt independence (dHSIC) criterion. Each component of MVAE models a distinct latent space, avoiding the overfitting that 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, subdecoders, 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 SGD. We are currently working towards using mixture models, and developing appropriate training algorithms, for lifelong learning.

REFERENCES
