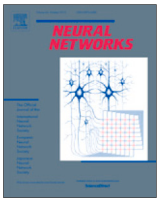
Neural Networks 135 (2021) 177–191

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Generative Restricted Kernel Machines: A framework for multi-view generation and disentangled feature learning

Arun Pandey [*], Joachim Schreurs, Johan A.K. Suykens

Department of Electrical Engineering (ESAT-STADIUS), KU Leuven, 3000 Leuven, Belgium

ARTICLE INFO

Article history:
Received 26 August 2020
Received in revised form 29 November 2020
Accepted 10 December 2020
Available online 23 December 2020

Keywords:
Generative models
Latent variable models
Deep learning
Restricted kernel machines

ABSTRACT

This paper introduces a novel framework for generative models based on Restricted Kernel Machines (RKMs) with joint multi-view generation and uncorrelated feature learning, called Gen-RKM. To enable joint multi-view generation, this mechanism uses a shared representation of data from various views. Furthermore, the model has a primal and dual formulation to incorporate both kernel-based and (deep convolutional) neural network based models within the same setting. When using neural networks as explicit feature-maps, a novel training procedure is proposed, which jointly learns the features and shared subspace representation. The latent variables are given by the eigen-decomposition of the kernel matrix, where the mutual orthogonality of eigenvectors represents the learned *uncorrelated* features. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of generated samples on various standard datasets.

1. Introduction

In the past decade, interest in generative models has grown tremendously, finding applications in multiple fields such as, generated art, on-demand video, image denoising (Vincent, Larochelle, Lajoie, Bengio, & Manzagol, 2010), exploration in reinforcement learning (Florensa, Held, Geng, & Abbeel, 2018), collaborative filtering (Salakhutdinov, Mnih, & Hinton, 2007), in-painting (Yeh, Chen, Yian Lim, Schwing, Hasegawa-Johnson, & Do, 2017) and many more. Some examples of generative models based on a probabilistic framework with latent variables are Variational Auto-Encoders (Kingma & Welling, 2014) and Restricted Boltzmann Machines (RBMs) (Salakhutdinov & Hinton, 2009; Smolensky, 1986). More recently proposed models are based on adversarial training such as Generative Adversarial Networks (GANs) (Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, & Bengio, 2014) and its many variants. Furthermore, auto-regressive models such as Pixel Recurrent Neural Networks (PixelRNNs) (Van Den Oord, Kalchbrenner, & Kavukcuoglu, 2016) model the conditional distribution of every individual pixel given previous pixels. All these approaches have their own advantages and disadvantages. For example, RBMs perform both learning and Bayesian inference in graphical models with latent variables. However, such probabilistic models must be properly normalized, which requires evaluating intractable integrals over the space of all possible variable configurations (Salakhutdinov & Hinton, 2009). Currently GANs are considered as the state-of-the-art for generative modeling tasks, producing high-quality images but are more difficult to train due to unstable training dynamics, unless more sophisticated variants are applied.

Many datasets are composed of different representations of the data, also called views. Views can correspond to different modalities such as sounds, images, videos, sequences of previous frames, etc. Although each view could individually be used for learning tasks, exploiting information from all views together could improve the learning quality (Chen & Denoyer, 2017; Liu & Tuzel, 2016; Pu, Gan, Henao, Yuan, Li, Stevens, & Carin, 2016). Furthermore, it is among the goals of the latent variable modeling to model the description of data in terms of *uncorrelated* or *independent* components. Some classical examples are Independent Component Analysis; Hidden Markov models (Rabiner & Juang, 1986); Probabilistic Principal Component Analysis (PCA) (Tipping & Bishop, 1999); Gaussian-Process latent variable model (Lawrence, 2005) and factor analysis. Hence, when learning a latent space in generative models, it becomes interesting to find a disentangled representation. Disentangled variables are generally considered to contain interpretable information and reflect distinct factors of variation in the data for e.g. lighting conditions, style, colors, etc. This makes disentangled representations especially interesting for the generation of plausible pseudo-data with certain desirable properties, e.g. generating new chair designs with a certain armrest or new cars with a predefined color. The definition of disentanglement in the literature is not precise,

[*] Correspondence to: B01.62, Department of Electrical Engineering (ESAT-STADIUS), KU Leuven, 3000 Leuven, Belgium.
E-mail address: arun.pandey@esat.kuleuven.be (A. Pandey).

https://doi.org/10.1016/j.neunet.2020.12.010
0893-6080/

however many believe that a representation with statistically independent variables is a good starting point (Ridgeway, 2016; Schmidhuber, 1992). Such representations extract information into a compact form which makes it possible to generate samples with specific characteristics (Bouchacourt, Tomioka, & Nowozin, 2018; Chen, Duan, Houthooft, Schulman, Sutskever, & Abbeel, 2016; Chen, Li, Grosse, & Duvenaud, 2018; Tran, Yin, & Liu, 2017). Additionally, these representations have been found to generalize better and be more robust against adversarial attacks (Alemi, Fischer, Dillon, & Murphy, 2017).

In this work, we propose a novel generative mechanism based on the framework of Restricted Kernel Machines (RKMs) (Suykens, 2017), called Generative-RKM (Gen-RKM). RKMs yield a representation of kernel methods with visible and hidden units establishing links between Kernel PCA, Least-Squares Support Vector Machines (LS-SVM) (Suykens, Van Gestel, De Brabanter, De Moor, & Vandewalle, 2002) and RBMs. This framework has a similar energy form as RBMs, though there is a non-probabilistic training procedure where the eigenvalue decomposition plays the role of normalization. Recently, Houthuys and Suykens (2018) used this framework to develop tensor-based multi-view classification models and Schreurs and Suykens (2018) showed how kernel PCA fits into this framework.

Contributions: (1) A novel joint multi-view generative model based on the RKM framework where multiple views of the data can be generated simultaneously. **(2)** Two methods are discussed for computing the pre-image of the feature vectors: with the feature map explicitly known or unknown. We show that the mechanism is flexible to incorporate both kernel-based and (deep convolutional) neural network based models within the same setting. **(3)** When using explicit feature maps, we propose a training algorithm that jointly performs the feature-selection and learns the common-subspace representation in the same procedure. **(4)** Qualitative and quantitative experiments demonstrate that the model is capable of generating good quality images of natural objects. Further illustrations on multi-view datasets exhibit the potential of the model. Thanks to the orthogonality of eigenvectors of the kernel matrix, the learned latent variables are uncorrelated. This resembles a disentangled representation, which makes it possible to generate data with specific characteristics.

2. Related work

Latent space models were studied in several other works, where multiple links with disentanglement are made. VAEs (Kingma & Welling, 2014) have become a popular framework among different generative models as they provide more theoretically well-founded and stable training than GANs (Goodfellow et al., 2014). Learning a VAE amounts to the optimization of an objective balancing the quality of samples that are autoencoded through a stochastic encoder–decoder pair, measured by the reconstruction error, while encouraging the latent space to follow a fixed prior distribution, often the Gaussian distribution. In β-VAEs (Higgins, Matthey, Pal, Burgess, Glorot, Botvinick, Mohamed, & Lerchner, 2017), an adjustable hyperparameter β is introduced that balances quality of samples and latent space constraints with reconstruction accuracy. The choice of parameter $\beta = 1$ corresponds to the original VAE formulation. Further, they show that with $\beta > 1$ (more emphasis on the latent variables to be Gaussian distributed) the model is capable of learning a more disentangled latent representation of the data. In Burgess, Higgins, Pal, Matthey, Watters, Desjardins, and Lerchner (2018), the effect of the β term is analyzed more in depth. It was suggested that the stronger pressure for the posterior to match the factorized unit Gaussian prior puts extra constraints on the implicit capacity of the latent bottleneck (Higgins et al., 2017). Chen et al. (2018) show a decomposition of the variational lower bound that can be used to explain the success of the β-VAE (Higgins et al., 2017) in learning disentangled representations. The authors claim that the total correlation, which forces the model to find statistically independent factors in the data distribution, is the most important term in this decomposition. The role of disentanglement was also studied in GANs, where the InfoGAN (Chen et al., 2016) is one of the most known works.

The most common approach of joint multimodal/multiview learning with deep neural networks is to share the top of hidden layers in modality specific networks. Srivastava and Salakhutdinov (2012) proposed a Deep Boltzmann Machine for learning multimodal data. The multimodal DBM learns a joint density model over the space of multimodal inputs by sharing the hidden units of the last layer. Examples of joint multimodal training for VAEs are Suzuki, Nakayama, and Matsuo (2016), Wu and Goodman (2018). The work of Suzuki et al. (2016) introduced the joint multi-modal VAE, which learns the common distribution using a joint inference network. The authors use an ELBO objective with two additional divergence terms to minimize the distance between the uni-modal and the multi-modal importance distributions. The MVAE of Wu and Goodman (2018) uses a product of experts formulation and sub-sampled training paradigm to solve the multi-modal inference problem.

In contrast to classical VAE architectures, the proposed model introduces an orthogonal interconnection matrix U motivated by the RKM formulation. The model thus finds an 'optimal' linear subspace of the latent space given by the eigendecomposition. In this paper, we argue that this orthogonality leads to better disentanglement and generation quality.

The paper is organized as follows. In Section 3, we discuss the Gen-RKM training and generation mechanism when multiple data sources are available. In Section 4, we explain how the model incorporates both kernel methods and neural networks through the use of implicit and explicit feature maps respectively. In Section 5, we show experimental results of our model applied on various public datasets. Section 6 concludes the paper along with directions towards the future work. Further discussions and derivations are given in the Appendix and the Python code is available at https://www.esat.kuleuven.be/stadius/E/software.php.

3. Generative restricted kernel machines framework

The proposed Gen-RKM framework consists of two phases: a training phase and a generation phase which occurs one after the other.

3.1. Training phase of the RKM

Similar to Energy-Based Models (EBMs, see LeCun, Huang, and Bottou (2004) for details), the RKM objective function captures dependencies between variables by associating a scalar energy to each configuration of the variables. Learning consists of finding an energy function in which the observed configurations of the variables are given lower energies than unobserved ones. Note that the schematic representation of Gen-RKM model, as shown in Fig. 1 is similar to Discriminative RBMs (Larochelle & Bengio, 2008) and the objective function $\mathcal{J}_t$ (defined below) has an energy form similar to RBMs with additional regularization terms. The latent space dimension in the RKM setting has a similar interpretation as the number of hidden units in a Restricted Boltzmann Machine, where in the specific case of the RKM these hidden units are uncorrelated.

We assume a dataset $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ with $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{y}_i \in \mathbb{R}^p$ consisting of N data points. Here $\boldsymbol{y}_i$ may represent an additional view of $\boldsymbol{x}_i$, e.g., an additional image from a different angle, the caption of an image or a class label. We start with the RKM interpretation of Kernel PCA, which gives an upper bound on the equality constrained Least-Squares Kernel PCA objective function (Suykens, 2017). Applying the feature-maps $\boldsymbol{\phi}_1 : \Omega_x \mapsto \mathcal{H}_x$ and $\boldsymbol{\phi}_2 : \Omega_y \mapsto \mathcal{H}_y$ to the input data points, where $\mathcal{H}_x, \mathcal{H}_y$ are the corresponding Reproducing Kernel Hilbert Spaces (RKHS) of the feature-maps respectively; the training objective function $\mathcal{J}_t$ for generative RKM is given by[1]:

$$\mathcal{J}_t = \sum_{i=1}^{N}\left\{-\boldsymbol{\phi}_1(\boldsymbol{x}_i)^\top \boldsymbol{U}\boldsymbol{h}_i - \boldsymbol{\phi}_2(\boldsymbol{y}_i)^\top \boldsymbol{V}\boldsymbol{h}_i + \frac{1}{2}\boldsymbol{h}_i^\top \boldsymbol{\Lambda}\boldsymbol{h}_i\right\} + \frac{\eta_1}{2}(\boldsymbol{U}^\top\boldsymbol{U}) + \frac{\eta_2}{2}(\boldsymbol{V}^\top\boldsymbol{V}), \tag{1}$$

where $\boldsymbol{U} \in \mathbb{R}^{d_f \times s}$, $\boldsymbol{V} \in \mathbb{R}^{p_f \times s}$ are the unknown interconnection matrices, $\boldsymbol{\Lambda} \succ 0$ the unknown diagonal matrix and $\boldsymbol{h}_i \in \mathbb{R}^s$ are the latent variables modeling a common subspace $\mathcal{H} \subseteq \mathcal{H}_x \bigoplus \mathcal{H}_y$ between the two feature spaces (see Fig. 1). To obtain this objective from LS-SVM formulation see Appendix A. Given $\eta_1 > 0$ and $\eta_2 > 0$ as regularization parameters, the stationary points of $\mathcal{J}_t$ are given by:

$$\begin{cases} \frac{\partial \mathcal{J}_t}{\partial \boldsymbol{h}_i} = 0 \implies \boldsymbol{\Lambda}\boldsymbol{h}_i = \boldsymbol{U}^\top \boldsymbol{\phi}_1(\boldsymbol{x}_i) + \boldsymbol{V}^\top \boldsymbol{\phi}_2(\boldsymbol{y}_i), \quad \forall i \\ \frac{\partial \mathcal{J}_t}{\partial \boldsymbol{U}} = 0 \implies \boldsymbol{U} = \frac{1}{\eta_1}\sum_{i=1}^{N} \boldsymbol{\phi}_1(\boldsymbol{x}_i)\boldsymbol{h}_i^\top, \\ \frac{\partial \mathcal{J}_t}{\partial \boldsymbol{V}} = 0 \implies \boldsymbol{V} = \frac{1}{\eta_2}\sum_{i=1}^{N} \boldsymbol{\phi}_2(\boldsymbol{y}_i)\boldsymbol{h}_i^\top. \end{cases} \tag{2}$$

Substituting $\boldsymbol{U}$ and $\boldsymbol{V}$ in the first equation above, denoting the diagonal matrix $\boldsymbol{\Lambda} = \{\lambda_1, \ldots, \lambda_s\} \in \mathbb{R}^{s\times s}$ with $s \leq N$, yields the following eigenvalue problem:

$$\left[\frac{1}{\eta_1}\boldsymbol{K}_1 + \frac{1}{\eta_2}\boldsymbol{K}_2\right]\boldsymbol{H}^\top = \boldsymbol{H}^\top \boldsymbol{\Lambda}, \tag{3}$$

where $\boldsymbol{H} = \left[\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\right] \in \mathbb{R}^{s\times N}$ with $s \leq N$ is the number of selected principal components and $\boldsymbol{K}_1, \boldsymbol{K}_2 \in \mathbb{R}^{N\times N}$ are the kernel matrices corresponding to data sources.[2] Based on Mercer's theorem (Mercer, 1909), positive-definite kernel functions $k_1 : \Omega_x \times \Omega_x \mapsto \mathbb{R}$, $k_2 : \Omega_y \times \Omega_y \mapsto \mathbb{R}$ can be defined such that $k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\phi}_1(\boldsymbol{x}_i), \boldsymbol{\phi}_1(\boldsymbol{x}_j)\rangle_{\mathcal{H}_x}$, and $k_2(\boldsymbol{y}_i, \boldsymbol{y}_j) = \langle \boldsymbol{\phi}_2(\boldsymbol{y}_i), \boldsymbol{\phi}_2(\boldsymbol{y}_j)\rangle_{\mathcal{H}_y}$, $\forall i, j = 1, \ldots, N$ forms the elements of corresponding kernel matrices. The feature maps $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$, mapping the input data to the high-dimensional feature space (possibly infinite) are implicitly defined by kernel functions. Typical examples of such kernels are given by the Gaussian RBF kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/(2\sigma^2)}$ or the Laplace kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2/\sigma}$ just to name a few (Scholkopf & Smola, 2001). However, one can also define explicit feature maps, still preserving the positive-definiteness of the kernel function by construction (Suykens et al., 2002). Eq. (3) corresponds to a kernel PCA operation. In this spirit, we thus find an orthogonal interconnection matrix U that is the optimal linear subspace of the latent space given by the eigendecomposition.

3.2. Generation

In this section, we derive the equations for the generative mechanism. RKMs resembling energy-based models, the inference consists in clamping the value of observed variables and finding configurations of the remaining variables that minimizes the energy (LeCun et al., 2004). Given the learned interconnection matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, and a given latent variable $\boldsymbol{h}^\star$, consider the following generation objective function $\mathcal{J}_g$:

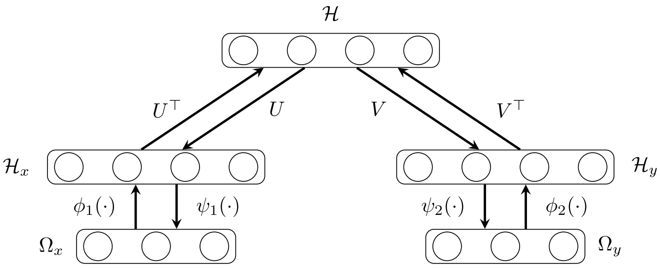


Fig. 1. Gen-RKM schematic representation modeling a common subspace $\mathcal{H} \subseteq \mathcal{H}_x \bigoplus \mathcal{H}_y$ between two data sources Ω_x and Ω_y. The $\boldsymbol{\phi}_1$, $\boldsymbol{\phi}_2$ are the feature maps ($\mathcal{H}_x$ and $\mathcal{H}_y$ represent the RKHS) corresponding to the two data sources. While $\boldsymbol{\psi}_1$, $\boldsymbol{\psi}_2$ represent the pre-image maps. The interconnection matrices $\boldsymbol{U}, \boldsymbol{V}$ capture the dependencies between latent variables and the mapped data sources.

$$\mathcal{J}_g = -\hat{\boldsymbol{\phi}}_1(\boldsymbol{x}^\star)^\top \boldsymbol{U}\boldsymbol{h}^\star - \hat{\boldsymbol{\phi}}_2(\boldsymbol{y}^\star)^\top \boldsymbol{V}\boldsymbol{h}^* + \frac{1}{2}\hat{\boldsymbol{\phi}}_1(\boldsymbol{x}^\star)^\top \hat{\boldsymbol{\phi}}_1(\boldsymbol{x}^\star) + \frac{1}{2}\hat{\boldsymbol{\phi}}_2(\boldsymbol{y}^\star)^\top \hat{\boldsymbol{\phi}}_2(\boldsymbol{y}^\star),$$

with an additional regularization term on data sources. With slight abuse of notation, we denote the generated feature vectors by $\hat{\boldsymbol{\phi}}_1(\boldsymbol{x}^\star)$ and $\hat{\boldsymbol{\phi}}_2(\boldsymbol{y}^\star)$ given the corresponding latent variable $\boldsymbol{h}^\star$, to distinguish from the feature vectors corresponding to training data points (see (1)). The given latent variable $\boldsymbol{h}^\star$ can be the corresponding latent code of a training point, a newly sampled hidden unit or a specifically determined one. Above cases correspond to generating the reconstructed visible unit, generating a random new visible unit or exploring the latent space by carefully selecting hidden units respectively. The stationary points of $\mathcal{J}_g$ are characterized by:

$$\begin{cases} \frac{\partial \mathcal{J}_g}{\partial \hat{\boldsymbol{\phi}}_1(\boldsymbol{x}^\star)} = 0 \implies \hat{\boldsymbol{\phi}}_1(\boldsymbol{x}^\star) = \boldsymbol{U}\boldsymbol{h}^\star, \\ \frac{\partial \mathcal{J}_g}{\partial \hat{\boldsymbol{\phi}}_2(\boldsymbol{y}^\star)} = 0 \implies \hat{\boldsymbol{\phi}}_2(\boldsymbol{y}^\star) = \boldsymbol{V}\boldsymbol{h}^\star. \end{cases} \tag{4}$$

Using $\boldsymbol{U}$ and $\boldsymbol{V}$ from (2), we obtain the generated feature vectors:

$$\hat{\boldsymbol{\phi}}_1(\boldsymbol{x}^\star) = \left(\frac{1}{\eta_1}\sum_{i=1}^{N} \boldsymbol{\phi}_1(\boldsymbol{x}_i)\boldsymbol{h}_i^\top\right)\boldsymbol{h}^\star, \qquad \hat{\boldsymbol{\phi}}_2(\boldsymbol{y}^\star) = \left(\frac{1}{\eta_2}\sum_{i=1}^{N} \boldsymbol{\phi}_2(\boldsymbol{y}_i)\boldsymbol{h}_i^\top\right)\boldsymbol{h}^\star. \tag{5}$$

To obtain the generated data, now one should compute the inverse images of the feature maps $\hat{\boldsymbol{\phi}}_1(\cdot)$ and $\hat{\boldsymbol{\phi}}_2(\cdot)$ in the respective input spaces, i.e., solve the *pre-image problem*. We seek to find the functions $\boldsymbol{\psi}_1 : \mathcal{H} \mapsto \Omega_x$ and $\boldsymbol{\psi}_2 : \mathcal{H} \mapsto \Omega_y$ corresponding to the two data-sources, such that $(\boldsymbol{\psi}_1 \circ \hat{\boldsymbol{\phi}}_1)(\boldsymbol{x}^\star) \approx \boldsymbol{x}^\star$ and $(\boldsymbol{\psi}_2 \circ \hat{\boldsymbol{\phi}}_2)(\boldsymbol{y}^\star) \approx \boldsymbol{y}^\star$, where $\hat{\boldsymbol{\phi}}_1(\boldsymbol{x}^\star)$ and $\hat{\boldsymbol{\phi}}_2(\boldsymbol{y}^\star)$ are given using (5).

When using kernel methods, explicit feature maps are not necessarily known. Commonly used kernels such as the radial-basis function and polynomial kernels map the input data to a very high dimensional feature space. Hence finding the pre-image, in general, is known to be an ill-conditioned problem (Mika, Schölkopf, Smola, Müller, Scholz, & Rätsch, 1999). However, various approximation techniques have been proposed (Bui, Im, Apley, & Runger, 2019; Honeine & Richard, 2011; Kwok & Tsang, 2003; Weston, Schölkopf, & Bakir, 2004) which could be used to obtain the approximate pre-image $\hat{\boldsymbol{x}}$ of $\hat{\boldsymbol{\phi}}_1(\boldsymbol{x}^\star)$. In Section 4.1, we employ one such technique to demonstrate the applicability

[1] For convenience, it is assumed that the feature vectors are centered in the feature space Ω_x, Ω_y using $\tilde{\boldsymbol{\phi}}(\boldsymbol{x}) := \boldsymbol{\phi}(\boldsymbol{x}) - \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\phi}(\boldsymbol{x}_i)$. Otherwise, a centered kernel matrix could be obtained using (C.1) in Appendix C.

[2] While in the above section we have assumed that only two data sources (namely Ω_x and Ω_y) are available for learning, the above procedure could be extended to multiple data-sources. For the M views or data-sources, this yields the training problem: $\left[\sum_{\ell=1}^{M}\frac{1}{\eta_\ell}\boldsymbol{K}_\ell\right]\boldsymbol{H}^\top = \boldsymbol{H}^\top\boldsymbol{\Lambda}$.

in our model, and consequently generate the multi-view data. One could also define explicit pre-image maps. In Section 4.2, we define parametric pre-image maps and learn the parameters by minimizing the appropriately defined objective function. The next section describes the above two pre-image methods for both cases, i.e., when the feature map is explicitly known or unknown, in greater detail.

4. The proposed algorithm with implicit & explicit feature maps

4.1. Implicit feature map

As noted in the previous section, since $\boldsymbol{x}^\star$ may not exist, we find an approximation $\hat{\boldsymbol{x}}$. A possible technique is shown by Schreurs and Suykens (2018). Left multiplying (5) by $\hat{\boldsymbol{\phi}}_1(\boldsymbol{x}_i)^\top$ and $\hat{\boldsymbol{\phi}}_2(\boldsymbol{y}_i)^\top$, $\forall i = 1, \dots, N$, we obtain:

$$\boldsymbol{k}_{\boldsymbol{x}^\star} = \frac{1}{\eta_1}\boldsymbol{K}_1\boldsymbol{H}^\top\boldsymbol{h}^\star, \quad \boldsymbol{k}_{\boldsymbol{y}^\star} = \frac{1}{\eta_2}\boldsymbol{K}_2\boldsymbol{H}^\top\boldsymbol{h}^\star, \tag{6}$$

where, $\boldsymbol{k}_{\boldsymbol{x}^\star} = [k(\boldsymbol{x}_1, \boldsymbol{x}^\star), \dots, k(\boldsymbol{x}_N, \boldsymbol{x}^\star)]^\top$ represents the *similarities* between $\hat{\boldsymbol{\phi}}_1(\boldsymbol{x}^\star)$ and training data points in the feature space, and $\boldsymbol{K}_1 \in \mathbb{R}^{N\times N}$ represents the centered kernel matrix of Ω_x. Similar conventions follow for Ω_y respectively. Using the *kernel-smoother* method (Hastie, Tibshirani, & Friedman, 2001), the pre-images are given by:

$$\begin{aligned} \hat{\boldsymbol{x}} &= (\boldsymbol{\psi}_1 \circ \hat{\boldsymbol{\phi}}_1)(\boldsymbol{x}^\star) = \frac{\sum_{j=1}^{n_r} \tilde{k}_1(\boldsymbol{x}_j, \boldsymbol{x}^\star)\boldsymbol{x}_j}{\sum_{j=1}^{n_r} \tilde{k}_1(\boldsymbol{x}_j, \boldsymbol{x}^\star)}, \\ \hat{\boldsymbol{y}} &= (\boldsymbol{\psi}_2 \circ \hat{\boldsymbol{\phi}}_2)(\boldsymbol{y}^\star) = \frac{\sum_{j=1}^{n_r} \tilde{k}_2(\boldsymbol{y}_j, \boldsymbol{y}^\star)\boldsymbol{y}_j}{\sum_{j=1}^{n_r} \tilde{k}_2(\boldsymbol{y}_j, \boldsymbol{y}^\star)}, \end{aligned} \tag{7}$$

where $\tilde{k}_1(\boldsymbol{x}_i, \boldsymbol{x}^\star)$ and $\tilde{k}_2(\boldsymbol{y}_i, \boldsymbol{y}^\star)$ are the scaled similarities (see (7)) between 0 and 1 and n_r the number of closest points based on the similarity defined by kernels $\tilde{k}_1$ and $\tilde{k}_2$.

4.2. Explicit feature map

While using an explicit feature map, Mercer's theorem is still applicable due to the positive semi-definiteness of the kernel function by construction, thereby allowing the derivation of (3). In the experiments, we use a set of (convolutional) neural networks as the parametric feature maps $\boldsymbol{\phi}_{\boldsymbol{\theta}}(\cdot)$. Another (transposed convolutional) neural network is used for the pre-image map $\boldsymbol{\psi}_{\boldsymbol{\zeta}}(\cdot)$ (Dumoulin & Visin, 2016). The network parameters $\{\boldsymbol{\theta}, \boldsymbol{\zeta}\}$ are learned by minimizing the reconstruction errors $\mathcal{L}_1(\boldsymbol{x}, \boldsymbol{\psi}_{1_{\zeta_1}}(\hat{\boldsymbol{\phi}}_{1_{\theta_1}}(\boldsymbol{x}))) = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{\psi}_{1_{\zeta_1}}(\hat{\boldsymbol{\phi}}_{1_{\theta_1}}(\boldsymbol{x}_i))\|_2^2$ for the first view and $\mathcal{L}_2(\boldsymbol{y}, \boldsymbol{\psi}_{2_{\zeta_2}}(\hat{\boldsymbol{\phi}}_{2_{\theta_2}}(\boldsymbol{y}))) = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{y}_i - \boldsymbol{\psi}_{2_{\zeta_2}}(\hat{\boldsymbol{\phi}}_{2_{\theta_2}}(\boldsymbol{y}_i))\|_2^2$ for the second view, however, in principle, one can use any other loss appropriate to the dataset. Here $\hat{\boldsymbol{\phi}}_{1_{\theta_1}}(\boldsymbol{x}_i)$ and $\hat{\boldsymbol{\phi}}_{2_{\theta_2}}(\boldsymbol{y}_i)$ are given by (5), i.e., the generated points in feature space from the subspace $\mathcal{H}$. Adding the loss function directly into the objective function $\mathcal{J}_t$ is not suitable for minimization. Instead, we use the stabilized objective function defined as $\mathcal{J}_{stab} = \mathcal{J}_t + \frac{c_{\text{stab}}}{2}\mathcal{J}_t^2$, where $c_{stab} \in \mathbb{R}^+$ is the regularization constant (Suykens, 2017). This tends to push the objective function $\mathcal{J}_t$ towards zero, which is also the case when substituting the solutions $\lambda_i, \boldsymbol{h}_i$ back into $\mathcal{J}_t$ (see Appendix B for details). The combined training objective is given by:

$$\begin{aligned} \min_{\theta_1,\theta_2,\zeta_1,\zeta_2} \mathcal{J}_c \ &= \mathcal{J}_{stab} + \frac{\gamma}{2N}\left(\sum_{i=1}^{N}\left[\mathcal{L}_1(\boldsymbol{x}_i, \boldsymbol{\psi}_{1_{\zeta_1}}(\hat{\boldsymbol{\phi}}_{1_{\theta_1}}(\boldsymbol{x}_i)))\right.\right. \\ &+ \ \left.\left.\mathcal{L}_2(\boldsymbol{y}_i, \boldsymbol{\psi}_{2_{\zeta_2}}(\hat{\boldsymbol{\phi}}_{2_{\theta_2}}(\boldsymbol{y}_i)))\right]\right), \end{aligned}$$

Algorithm 1 Gen-RKM

```
Input: ${\boldsymbol{x}_i, \boldsymbol{y}_i}_{i=1}^N$, $\eta_1$, $\eta_2$, feature map $\boldsymbol{\phi}_j(\cdot)$ - explicit or implicit
  via kernels $k_j(\cdot,\cdot)$, for $j \in \{1, 2\}$
Output: Generated data $\boldsymbol{x}^\star$, $\boldsymbol{y}^\star$
 1: procedure TRAIN
 2:     if $\boldsymbol{\phi}_j(\cdot)$ = Implicit then
 3:         Solve the eigendecomposition in (3)
 4:         Select the $s$ first principal components
 5:     else if $\boldsymbol{\phi}_j(\cdot)$ = Explicit then
 6:         while not converged do
 7:             $\{\boldsymbol{x}, \boldsymbol{y}\} \leftarrow$ {Get mini-batch}
 8:             $\boldsymbol{\phi}_1(\boldsymbol{x}) \leftarrow \boldsymbol{x}$; $\boldsymbol{\phi}_2(\boldsymbol{y}) \leftarrow \boldsymbol{y}$            ▷ Get embeddings
 9:             do steps 3-4
10:             $\{\hat{\boldsymbol{\phi}}_1(\boldsymbol{x}), \hat{\boldsymbol{\phi}}_2(\boldsymbol{y})\} \leftarrow \boldsymbol{h}$ ((5))
11:             $\{\boldsymbol{x}, \boldsymbol{y}\} \leftarrow \{\boldsymbol{\psi}_1(\hat{\boldsymbol{\phi}}_1(\boldsymbol{x})), \boldsymbol{\psi}_2(\hat{\boldsymbol{\phi}}_2(\boldsymbol{y}))\}$   ▷ Pre-image map
12:             $\Delta\{\boldsymbol{\theta}, \boldsymbol{\zeta}\} \propto -\nabla_{\{\boldsymbol{\theta},\boldsymbol{\zeta}\}}\mathcal{J}_c$            ▷ Update parameters
13:         end while
14:     end if
15: end procedure
 1: procedure GENERATION
 2:     Select $\boldsymbol{h}^\star$
 3:     if $\boldsymbol{\phi}_j(\cdot)$ = Implicit then
 4:         Set hyperparameter: $n_r$
 5:         Compute $\boldsymbol{k}_{\boldsymbol{x}^*}$, $\boldsymbol{k}_{\boldsymbol{y}^*}$ ((6))
 6:         Get $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$ ((7))
 7:     else if $\boldsymbol{\phi}_j(\cdot)$ = Explicit then
 8:         do steps 10-11
 9:     end if
10: end procedure
```

where $\gamma \in \mathbb{R}^+$ is a regularization constant to control the stability with reconstruction accuracy. In this way, we combine feature-selection and subspace learning within the same training procedure.

In the objective of the VAE (Kingma & Welling, 2014), an extra term in the form of the Kullback–Leibler divergence between the encoder's distribution and a unit Gaussian is added as a prior on the latent variables. This ensures the latent space is smooth and without discontinuities, which is essential for good generation. By interpreting kernel PCA within the LS-SVM setting (Suykens et al., 2002), the PCA analysis can take the interpretation of a one-class modeling problem with zero target value around which one maximizes the variance (Suykens, Van Gestel, Vandewalle, & De Moor, 2003). When choosing a good feature map, one expects the latent variables to be normally distributed around zero. As a result, the latent space of the Gen-RKM is continuous, allowing easy random sampling and interpolation (see Fig. 2). Kernel PCA gives uncorrelated components in feature space (Bishop, 2006). While the standard PCA does not give a good disentangled representation for images (Eastwood & Williams, 2018; Higgins et al., 2017). By designing a good kernel (through appropriate feature-maps) and doing kernel PCA, it is possible to get a disentangled representation for images as we demonstrate in Fig. 7.

4.3. The Gen-RKM algorithm

Based on the previous discussion, we propose a novel procedure, called the Gen-RKM algorithm, combining kernel learning and generative models. We show that this procedure is efficient to train and evaluate. The training procedure simultaneously involves feature selection, common-subspace learning and pre-image map learning. This is achieved via an optimization procedure where one iteration involves an eigen-decomposition of

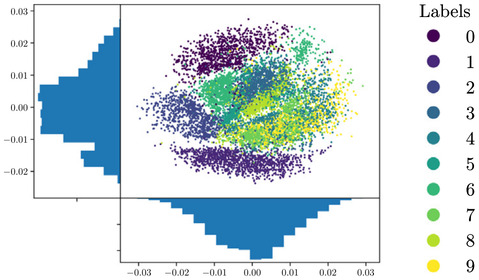


Fig. 2. MNIST: Scatter plot of latent variable distribution when trained on 10000 images ($s = 2$). Training was unsupervised (i.e. one-view) and labels are only used to color the plot. The latent space resembles a Gaussian distribution centered around 0, where the various digits are clustered together. Generated samples from a uniform grid over this space are shown in Fig. G.13.

the kernel matrix which is composed of the features from various views (see (3)). The latent variables are given by the eigenvectors, from which a pre-image map reconstructs the generated sample. Fig. 1 shows a schematic representation of the algorithm when two data sources are available.

Thanks to training in m mini-batches, this procedure is scalable to large datasets (sample size N) with training time scaling super-linearly with $T_m = c\frac{N^\gamma}{m^{\gamma-1}}$, instead of $T_k = cN^\gamma$, where $\gamma \approx 3$ for algorithms based on decomposition methods, with some proportionality constant c. The training time could be further reduced by computing the covariance matrix (size $(d_f + p_f) \times (d_f + p_f)$) instead of a kernel matrix (size $\frac{N}{m} \times \frac{N}{m}$), when the sum of the dimensions of the feature-spaces is less than the samples in mini-batch i.e. $d_f + p_f \leq \frac{N}{m}$. When using neural networks as feature maps, d_f and p_f correspond to the number of neurons in the output layer, which are chosen as hyperparameters by the practitioner. Eigendecomposition of this smaller covariance matrix would yield $\boldsymbol{U}$ and $\boldsymbol{V}$ as eigenvectors (see (8) and Appendix A.1 for detailed derivation), where computing the $\boldsymbol{h}_i$ involves only matrix-multiplication which is readily parallelizable on modern GPUs:

$$\begin{bmatrix} \frac{1}{\eta_1}\boldsymbol{\Phi}_{\boldsymbol{x}}\boldsymbol{\Phi}_{\boldsymbol{x}}^\top & \frac{1}{\eta_1}\boldsymbol{\Phi}_{\boldsymbol{x}}\boldsymbol{\Phi}_{\boldsymbol{y}}^\top \\ \frac{1}{\eta_2}\boldsymbol{\Phi}_{\boldsymbol{y}}\boldsymbol{\Phi}_{\boldsymbol{x}}^\top & \frac{1}{\eta_2}\boldsymbol{\Phi}_{\boldsymbol{y}}\boldsymbol{\Phi}_{\boldsymbol{y}}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix} \boldsymbol{\Lambda}, \tag{8}$$

where:

$$\boldsymbol{\Phi}_{\boldsymbol{x}} := \left[\boldsymbol{\phi}_1(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}_1(\boldsymbol{x}_N)\right],$$
$$\boldsymbol{\Phi}_{\boldsymbol{y}} := \left[\boldsymbol{\phi}_2(\boldsymbol{y}_1), \ldots, \boldsymbol{\phi}_2(\boldsymbol{y}_N)\right].$$

5. Experiments

To demonstrate the applicability of the proposed framework and algorithm, we trained the Gen-RKM model on a variety of datasets commonly used to evaluate generative models: MNIST (LeCun & Cortes, 2010), Fashion-MNIST (Xiao, Rasul, & Vollgraf, 2017), CIFAR-10 (Krizhevsky, 2009), CelebA (Liu, Luo, Wang, & Tang, 2015), Sketchy (Sangkloy, Burnell, Ham, & Hays, 2016a), Dsprites (Matthey, Higgins, Hassabis, & Lerchner, 2017) and Teapot (Eastwood & Williams, 2018). The proposed method adheres to both a primal and dual formulation to incorporate both kernel based methods as well as neural networks based models in the same setting. The convolutional neural networks are used as explicit feature maps which are known to outperform kernel based feature maps on the image datasets. Moreover, by using explicit feature maps we demonstrate the capability of the algorithm to jointly learn the feature map and shared subspace representation. For completeness, we also give an illustration when using implicit feature maps through the Gaussian kernel in Fig. 6.

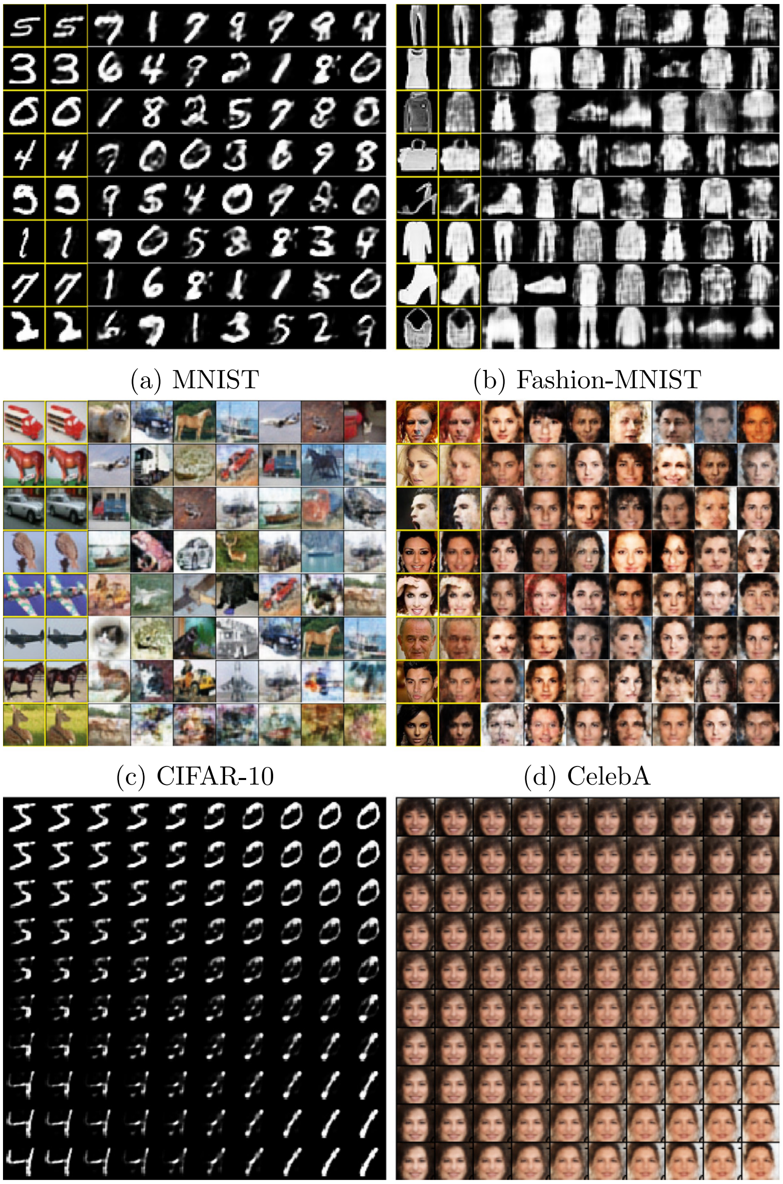
Table 1
FID Scores (Heusel, Ramsauer, Unterthiner, Nessler, & Hochreiter, 2017) for randomly generated samples (smaller is better).

Dataset	Algorithm	FID score (↓)		
		$s = 10$	$s = 30$	$s = 50$
MNIST	Gen-RKM	**89.825**	**130.497**	**131.696**
	VAE	250	234.749	205.282
	β-TCVAE	221.45	182.93	158.31
	InfoGAN	238.75	204.63	179.43
CelebA	Gen-RKM	**103.299**	**84.403**	**85.121**
	VAE	286.039	245.738	225.783
	β-TCVAE	248.47	226.75	173.21
	InfoGAN	264.79	228.31	185.93
fMNIST	Gen-RKM	**93.437**	**127.893**	146.643
	VAE	239.492	211.482	196.794
	β-TCVAE	206.784	187.221	**136.466**
	InfoGAN	247.853	215.683	199.337
CIFAR10	Gen-RKM	**122.475**	**138.467**	**158.871**
	VAE	295.382	259.557	231.475
	β-TCVAE	283.58	214.681	168.483
	InfoGAN	295.321	258.471	220.482

In our experiments, we fit a Gaussian mixture model (GMM) with l components to the latent variables of the training set, and randomly sample a new point $\boldsymbol{h}^\star$ for generating views. In case of explicit feature maps, we define $\boldsymbol{\phi}_{1_{\theta_1}}$ and $\boldsymbol{\psi}_{1_{\zeta_1}}$ as convolution and transposed-convolution neural networks, respectively (Dumoulin & Visin, 2016); and $\boldsymbol{\phi}_{2_{\theta_2}}$ and $\boldsymbol{\psi}_{1_{\zeta_2}}$ as fully-connected networks. The particular architecture details are outlined in Table D.4 in Appendix A. The training procedure in case of explicitly defined maps consists of minimizing $\mathcal{J}_c$ using the Adam optimizer (Kingma & Ba, 2014) to update the weights and biases. To speed-up learning, we subdivided the datasets into m mini-batches, and within each iteration of the optimizer, (3) is solved to model the subspace $\mathcal{H}$. Information on the datasets and hyperparameters used for the experiments is given in Table D.3 in Appendix A. A comparison of the average training time is given in Table G.5 in Appendix A.

Random generation. (1) Qualitative examples: Fig. 3 shows the generated images using a convolutional neural network and transposed-convolutional neural network as the feature map and pre-image map respectively. The first column in yellow-boxes shows the training samples and the second column on the right shows the reconstructed samples. The other images shown are generated by random sampling from a GMM over the learned latent variables. Notice that the reconstructed samples are of better quality visually than the other images generated by random sampling. To demonstrate that the model has not merely memorized the training examples, we show the generated images via bilinear-interpolations in the latent space in Fig. 3(e) and Fig. 3(f).

(2) Quantitative comparison: We compare the proposed model with the standard VAE (Kingma & Welling, 2014), β-VAE (Kingma & Welling, 2014), β-TCVAE (Chen et al., 2018) and Info-GAN (Chen et al., 2016). For the Info-GAN, batch normalization is added for training stability. As suggested by the authors, we keep $\alpha = \gamma = 1$ and only modify the hyperparameter β for the β-TCVAE model. Determination of the β hyperparameter is done by starting from values in the range of the parameters suggested in the authors' reference implementation. After trying various values we noticed that $\beta = 3$ seemed to work good across all datasets that we considered. For a fair comparison, the models have the same encoder/decoder architecture, optimization parameters and are trained until convergence, where the details are given in Table D.4. We evaluate the performance qualitatively by comparing reconstruction and random sampling, the results are shown in



(a) MNIST

(b) Fashion-MNIST

(c) CIFAR-10

(d) CelebA

(e) Bilinear interpolation

(f) Bilinear interpolation

Fig. 3. Generated samples from the model using CNN as explicit feature map in the kernel function. In (a), (b), (c), (d) the yellow boxes in the first column show training examples and the adjacent boxes show the reconstructed samples. The other images (columns 3–6) are generated by random sampling from the fitted distribution over the learned latent variables. (e) and (f) show the generated images through bilinear interpolations in the latent space.

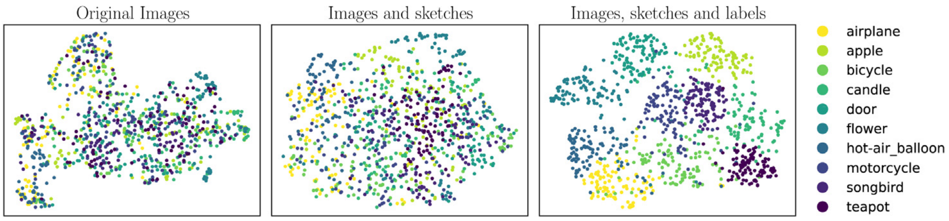


Fig. 4. Learned latent space visualization of the Sketchy dataset in 1, 2 and 3-view Gen-RKM setting by using an UMAP embedding (McInnes, Healy, Saul, & Großberger, 2018).

Fig. G.12 in Appendix A. In order to quantitatively assess the quality of the randomly generated samples, we use the Fréchet Inception Distance (FID) introduced by Heusel et al. (2017). The results are reported in Table 1. Experiments were repeated for different latent-space dimensions (s), and we observe empirically that FID scores are better for the Gen-RKM. This is confirmed by the qualitative evaluation in Fig. G.12. An interesting trend could be noted that as the dimension of latent-space is increased, VAE gets better at generating images whereas the performance of Gen-RKM decreases slightly. This is attributed to the eigen-decomposition of the kernel matrix whose eigenvalue spectrum decreases rapidly depicting that most information is captured in

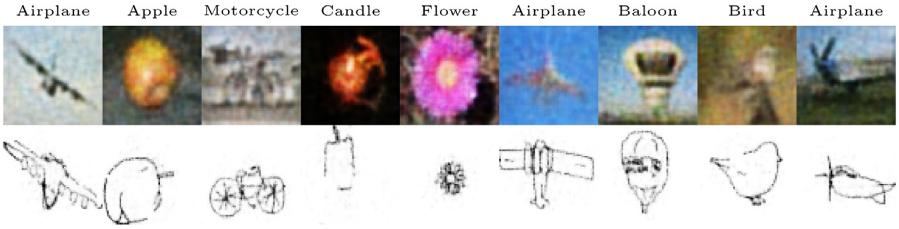


Fig. 5. Multi-view generation on Sketchy dataset showing labels, images and sketches generated together from the common subspace.

few principal components, while the rest is noise. The presence of noise hinders the convergence of the model. It is therefore important to select the number of latent variables proportionally to the size of the mini-batch and the corresponding spectrum of the kernel matrix (the diversity within a mini-batch affects the eigenvalue spectrum of the kernel matrix).

Multi-view generation. Figs. 5 & 6 demonstrate the multi-view generative capabilities of the model. In these datasets, labels or attributes are seen as another view of the image that provides extra information. One-hot encoding of the labels was used to train the model. Fig. 5(a) shows the generated images and labels when feature maps are only implicitly known i.e. through a Gaussian kernel. Figs. 5(b) and 5(c) show the same when using fully-connected networks as parametric functions to encode and decode labels. Next we show an illustration of multi-view generation on the Sketchy database (Sangkloy, Burnell, Ham, & Hays, 2016b). The dataset is a collection of sketch-photo pairs resulting in 3 views: images, sketches and labels. The dataset includes 125 object categories with 12,500 natural object images and 75,471 hand-drawn sketches for each class. The following pre-processing is done before training the GEN-RKM model: for the sketchy dataset, we selected 10 classes for training: airplane, apple, bicycle, candle, door, flower, hot-air-balloon, motorcycle, songbird and teapot. After that, the images are resized to 64 × 64 × 3. Further details on the used model architectures and hyperparameters are given in the Appendix. The learned latent space is visualized in Fig. 4. One can clearly observe that the joint learning of the different views results in a better separation of the classes. Joint random generations are given in Fig. 5.

Disentanglement. (1) Qualitative examples: The latent variables are uncorrelated, which gives an indication that the model could resemble a disentangled representation. This is confirmed by the empirical evidence in Fig. 7, where we explore the uncorrelated features learned by the models on the Dsprites and celebA datasets. In our experiments, the Dsprites training dataset comprised of 32 × 32 positions of oval and heart-shaped objects. The number of principal components chosen were 2 and the goal was to find out whether traversing in the direction of principal components, corresponds to traversing the generated images in one particular direction while preserving the shape of the object. Rows 1 and 2 of Fig. 7 show the reconstructed images of an oval while moving along first and second principal component respectively. Notice that the first and second components correspond to the y and x positions respectively. Rows 3 and 4 show the same for hearts. On the celebA dataset, we train the Gen-RKM with 15 components on a subset. Rows 5 and 6 show the reconstructed images while traversing along the principal components. When moving along the first component from left-to-right, the hair-color of the women changes, while preserving the face structure. Whereas traversal along the second component, transforms a man to woman while preserving the orientation. When the number of principal components were 2 while training, the brightness and background light-source corresponds to the two largest variances in the dataset. Also notice that, the reconstructed images are more blurry due to the selection of less number of components to model $\mathcal{H}$.

(2) Quantitative comparisons: To quantitatively assess disentanglement performance, we compare Gen-RKM with VAE (Kingma & Welling, 2014) and β-VAE (Higgins et al., 2017) on the Dsprites and Teapot datasets (Eastwood & Williams, 2018). The models have the same encoder/decoder architecture, optimization parameters and are trained until convergence, where the details are given in Table D.4. The performance is measured using the proposed framework[3] of Eastwood and Williams (2018), which gives 3 measures: disentanglement, completeness and informativeness. The results are shown in Table 2. Gen-RKM has good performance on the Dsprites dataset when the latent space dimension is equal to 2. This is expected as the number of disentangled generating factors in the dataset is also equal to 2, hence there are no noisy components in the kernel PCA hindering the convergence. The opposite happens in the case $h_{dim} = 10$, where noisy components are present. The above is confirmed by the Relative Importance Matrix in Fig. F.8 in the Appendix, where the 2 generating factors are well separated in the latent space of the Gen-RKM. For the Teapot dataset, Gen-RKM has good performance when $s = 10$. More components are needed to capture all variations in the dataset, where the number of generating factors is now equal to 5. In the other cases, Gen-RKM has a performance comparable to the others. Note that the model selection was done a-priori, that is, the hyperparameters of classifiers were selected before evaluating the disentanglement metric. This may explain the poor scores for Gen-RKM with Random Forest classifier in Teapot dataset ($s = 5$).

6. Conclusion and future work

The paper proposes a novel framework, called Gen-RKM, for generative models based on RKMs with extensions to multi-view generation and learning uncorrelated representations. This allows for a mechanism where the feature map can be implicitly defined using kernel functions or explicitly by (deep) neural network based methods. When using kernel functions, the training consists of only solving an eigenvalue problem. In the case of a (convolutional) neural network based explicit feature map, we used (transposed) networks as the pre-image functions. Consequently, a training procedure was proposed which involves joint feature-selection and subspace learning. Thanks to training in mini-batches and capability of working with covariance matrices, the training is scalable to large datasets. Experiments on benchmark datasets illustrate the merit of the proposed framework for generation quality as well as disentanglement. Extensions of this work consists of adapting the model to more advanced multi-view datasets involving speech, images and texts; further analysis on other feature maps, pre-image methods, loss-functions and

[3] Code and dataset available at https://github.com/cianeastwood/qedr.

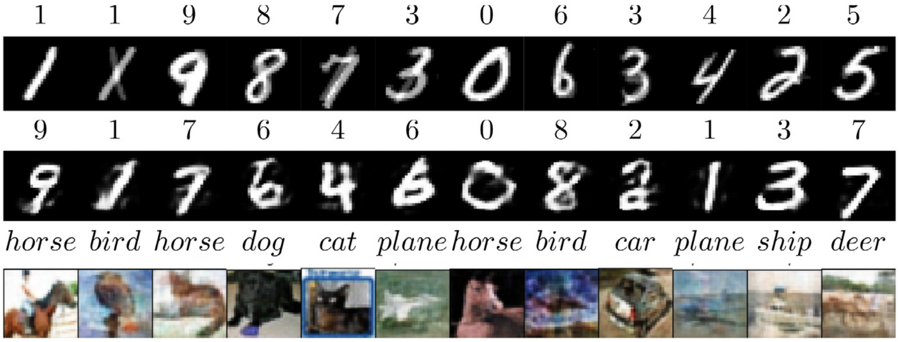

1 1 9 8 7 3 0 6 3 4 2 5
9 1 7 6 4 6 0 8 2 1 3 7
horse bird horse dog cat plane horse bird car plane ship deer

Fig. 6. Multi-view Generation (images and labels) on various datasets using implicit and explicit feature maps. (a) MNIST: Implicit feature maps with Gaussian kernel are used during training. For generation, the pre-images are computed using the kernel-smoother method. (b, c) MNIST and CIFAR-10: Explicit feature maps and the corresponding pre-image maps are defined by the Convolutional Neural Networks and Transposed CNNs respectively.

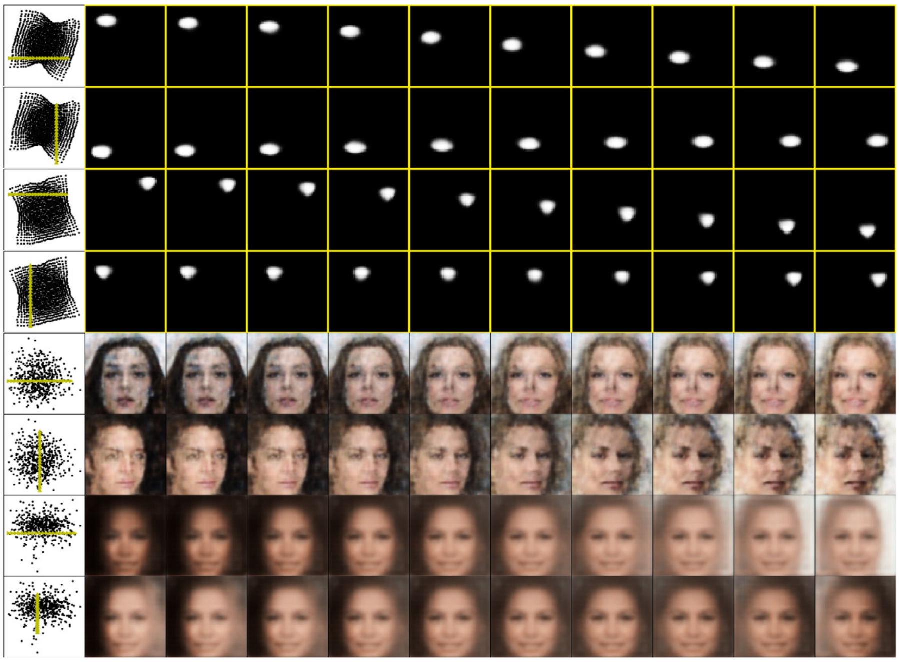


Fig. 7. Exploring the learned uncorrelated-features by traversing along the eigenvectors. The first column shows the scatter plot of latent variables using the top two principal components. The green lines within, show the traversal in the latent space and the related rows show the corresponding reconstructed images.

uncorrelated feature learning. Finally, this paper has demonstrated the applicability of the Gen-RKM framework, suggesting new research directions to be worth exploring.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

EU: The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program/ERC Advanced Grant E-DUALITY (787960). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information. Research Council KUL: Optimization frameworks for deep kernel machines C14/18/068 Flemish Government: FWO: projects: GOA4917N (Deep Restricted Kernel Machines: Methods and Foundations), Ph.D./Postdoc grant Impulsfonds AI: VR 2019 2203 DOC.0318/1QUATER Kenniscentrum Data en Maatschappij Ford KU Leuven Research Alliance Project KUL0076 (Stability analysis and performance improvement of deep reinforcement learning algorithms), ICT 48 TAILOR, Leuven.AI Institute

Appendix A. Derivation of Gen-RKM objective function

Given $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{y}_i \in \mathbb{R}^p$ and feature-map $\boldsymbol{\phi}_1 : \Omega_x \mapsto \mathcal{H}_x$ and $\boldsymbol{\phi}_2 : \Omega_y \mapsto \mathcal{H}_y$, the Least-Squares Support Vector Machine (LS-SVM) formulation of Kernel PCA (Suykens

Table 2
Disentanglement Metric on DSprites and Teapot dataset with Lasso and Random Forest regressor (Eastwood & Williams, 2018). For disentanglement and completeness higher score is better, for informativeness, lower is better.

	s	Algorithm	Lasso			Random Forest		
			Dis. (↑)	Com. (↑)	Inf. (↓)	Dis. (↑)	Com. (↑)	Inf. (↓)
DSprites	10	Gen-RKM	0.30	0.10	0.87	0.12	0.10	0.28
		VAE	0.11	0.09	**0.17**	**0.73**	**0.54**	**0.06**
		β-VAE ($\beta = 3$)	0.53	**0.18**	0.18	0.58	0.36	**0.06**
		β-TCVAE ($\beta = 3$)	**0.55**	0.17	0.18	0.72	**0.54**	0.11
		Info-GAN	0.37	0.13	0.22	0.61	0.35	0.15
	2	Gen-RKM	**0.72**	**0.71**	**0.64**	**0.05**	0.19	**0.03**
		VAE	0.04	0.01	0.87	0.01	0.13	0.11
		β-VAE ($\beta = 3$)	0.13	0.40	0.71	0.00	**0.26**	0.09
		β-TCVAE ($\beta = 3$)	0.51	0.15	0.67	0.03	0.17	0.14
		Info-GAN	0.46	0.14	0.66	0.04	0.17	0.21
Teapot	10	Gen-RKM	0.28	0.23	0.39	**0.48**	**0.39**	**0.19**
		VAE	0.28	0.21	**0.36**	0.30	0.27	0.21
		β-VAE ($\beta = 3$)	0.33	**0.25**	**0.36**	0.31	0.24	0.20
		β-TCVAE ($\beta = 3$)	**0.35**	0.24	0.39	0.35	0.25	0.31
		Info-GAN	0.23	0.2	0.41	0.32	0.21	0.22
	5	Gen-RKM	0.22	0.23	0.74	0.08	0.09	**0.27**
		VAE	0.16	0.14	**0.66**	0.11	0.14	0.28
		β-VAE ($\beta = 3$)	0.31	0.25	0.68	**0.13**	0.15	0.29
		β-TCVAE ($\beta = 3$)	**0.33**	**0.26**	0.69	0.12	**0.16**	0.29
		Info-GAN	0.21	0.19	0.71	0.11	0.14	0.28

Table D.3
Datasets and hyperparameters used for the experiments. The bandwidth of the Gaussian kernel for generation corresponds to the bandwidth that gave the best performance determined by cross-validation on the MNIST classification problem.

Dataset	N	d	N_{subset}	s	m	σ	n_r	l
MNIST	60000	28 × 28	10000	500	50	1.3	4	10
Fashion-MNIST	60000	28 × 28	500	100	5	/	/	10
CIFAR-10	60000	32 × 32 × 3	500	500	5	/	/	10
CelebA	202599	128 × 128 × 3	3000	15	5	/	/	20
Dsprites	737280	64 × 64	1024	2/10	5	/	/	/
Teapot	200000	64 × 64 × 3	1000	5/10	100	/	/	/
Sketchy	75471	64 × 64 × 3	1000	30	100	/	/	/

et al., 2002) for the two data sources can be written as:

$$\min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{e}_i} \frac{\eta_1}{2}(\boldsymbol{U}^\top \boldsymbol{U}) + \frac{\eta_2}{2}(\boldsymbol{V}^\top \boldsymbol{V}) - \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{e}_i^\top \boldsymbol{\Lambda}^{-1}\boldsymbol{e}_i \\ \text{s.t. } \boldsymbol{e}_i = \boldsymbol{U}^\top \boldsymbol{\phi}_1(\boldsymbol{x}_i) + \boldsymbol{V}^\top \boldsymbol{\phi}_2(\boldsymbol{y}_i) \quad \forall i = 1,\dots,N, \tag{A.1}$$

where $\boldsymbol{U} \in \mathbb{R}^{d\times s}$ and $\boldsymbol{V} \in \mathbb{R}^{p\times s}$ are the interconnection matrices.

Using the notion of *conjugate feature duality* introduced in Suykens (2017), the error variables $\boldsymbol{e}_i$ are conjugated to latent variables $\boldsymbol{h}_i$ using:

$$\frac{1}{2}\boldsymbol{e}^\top \boldsymbol{\Lambda}^{-1}\boldsymbol{e} + \frac{1}{2}\boldsymbol{h}^\top \boldsymbol{\Lambda}\boldsymbol{h} \geq \boldsymbol{e}^\top \boldsymbol{h}, \qquad \forall \boldsymbol{e}, \boldsymbol{h} \in \mathbb{R}^s \tag{A.2}$$

which is also known as the Fenchel–Young inequality for the case of quadratic functions (Rockafellar, 1974). By eliminating the variables $\boldsymbol{e}_i$ from (A.1) and using (A.2), we obtain the Gen-RKM training objective function:

$$\mathcal{J}_t = \sum_{i=1}^{N}\left(-\boldsymbol{\phi}_1(\boldsymbol{x}_i)^\top \boldsymbol{U}\boldsymbol{h}_i - \boldsymbol{\phi}_2(\boldsymbol{y}_i)^\top \boldsymbol{V}\boldsymbol{h}_i + \frac{1}{2}\boldsymbol{h}_i^\top \boldsymbol{\Lambda}\boldsymbol{h}_i\right) \\ + \frac{\eta_1}{2}(\boldsymbol{U}^\top \boldsymbol{U}) + \frac{\eta_2}{2}(\boldsymbol{V}^\top \boldsymbol{V}).$$

A.1. Computing latent variables using covariance matrix

From (2), eliminating the variables $\boldsymbol{h}_i$ yields the following:

$$\frac{1}{\eta_1}\left[\sum_{i=1}^{N} \boldsymbol{\phi}_1(\boldsymbol{x}_i)\boldsymbol{\phi}_1(\boldsymbol{x}_i)^\top \boldsymbol{U} + \sum_{i=1}^{N} \boldsymbol{\phi}_1(\boldsymbol{x}_i)\boldsymbol{\phi}_2(\boldsymbol{y}_i)^\top \boldsymbol{V}\right] = \boldsymbol{\Lambda}\boldsymbol{U},$$

$$\frac{1}{\eta_2}\left[\sum_{i=1}^{N} \boldsymbol{\phi}_2(\boldsymbol{y}_i)\boldsymbol{\phi}_1(\boldsymbol{x}_i)^\top \boldsymbol{U} + \sum_{i=1}^{N} \boldsymbol{\phi}_2(\boldsymbol{y}_i)\boldsymbol{\phi}_2(\boldsymbol{y}_i)^\top \boldsymbol{V}\right] = \boldsymbol{\Lambda}\boldsymbol{V}.$$

Denote $\boldsymbol{\Phi_x} := \left[\boldsymbol{\phi}_1(\boldsymbol{x}_1),\dots,\boldsymbol{\phi}_1(\boldsymbol{x}_N)\right]$, $\boldsymbol{\Phi_y} := \left[\boldsymbol{\phi}_2(\boldsymbol{y}_1),\dots,\boldsymbol{\phi}_2(\boldsymbol{y}_N)\right]$ and the diagonal matrix $\boldsymbol{\Lambda} = \{\lambda_1,\dots,\lambda_s\} \in \mathbb{R}^{s\times s}$ with $s \leq N$. Now, composing the above equations in matrix form, we get the following eigen-decomposition problem:

$$\begin{bmatrix} \frac{1}{\eta_1}\boldsymbol{\Phi_x}\boldsymbol{\Phi_x}^\top & \frac{1}{\eta_1}\boldsymbol{\Phi_x}\boldsymbol{\Phi_y}^\top \\ \frac{1}{\eta_2}\boldsymbol{\Phi_y}\boldsymbol{\Phi_x}^\top & \frac{1}{\eta_2}\boldsymbol{\Phi_y}\boldsymbol{\Phi_y}^\top \end{bmatrix}\begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix}\boldsymbol{\Lambda}.$$

Here the size of the covariance matrix is $(d_f + p_f) \times (d_f + p_f)$. The latent variables $\boldsymbol{h}_i$ can be computed using (2), which simply involves matrix multiplications.

Appendix B. Stabilizing the objective function

Proposition 1. *All stationary solutions for* $\boldsymbol{H}$, $\boldsymbol{\Lambda}$ *in* (3) *of* $\mathcal{J}_t$ *lead to* $\mathcal{J}_t = 0$.

Table D.4
Details of model architectures used in the paper. All convolutions and transposed-convolutions are with stride 2 and padding 1. Unless stated otherwise, the layers have Parametric-RELU ($\alpha = 0.2$) activation function, except the output layers of the pre-image maps which has sigmoid activation function.

Dataset	Optimizer		Architecture	
	(Adam)		$\mathcal{X}$	$\mathcal{Y}$
MNIST/fMNIST	1e−3	Input	28 × 28 × 1	10 (One-hot encoding)

		Feature-map (fm)	Conv 32 × 4 × 4;
			Conv 64 × 4 × 4;
			FC 128 (Linear)
				FC 15, 20 (Linear)

		Pre-image map	reverse of fm	reverse of fm
		Latent space dim.	500/100	
CIFAR-10	1e−3	Input	32 × 32 × 3	10 (One-hot encoding)

		Feature-map (fm)	Conv 64 × 4 × 4;
			Conv 128 × 4 × 4;
			FC 128 (Linear)
				FC 15, 20

		Pre-image map	reverse of fm	reverse of fm
		Latent space dim.	500	
CelebA	1e−4	Input	64 × 64 × 3	–

		Feature-map (fm)	Conv 32 × 4 × 4;
			Conv 64 × 4 × 4;
			Conv 128 × 4 × 4;
			Conv 256 × 4 × 4 ;
			FC 128 (Linear)
				–

		Pre-image map	reverse of fm	–
		Latent space dim.	15	
Dsprites	1e−4	Input	64 × 64 × 1	–

		Feature-map (fm)	Conv 20 × 4 × 4;
			Conv 40 × 4 × 4;
			Conv 80 × 4 × 4;
			FC 128 (Linear)
				–

		Pre-image map	reverse of fm	–
		Latent space dim.	2/10	
Teapot	1e−4	Input	64 × 64 × 3	–

		Feature-map (fm)	Conv 30 × 4 × 4;
			Conv 60 × 4 × 4;
			Conv 90 × 4 × 4;
			FC 128 (Linear)
				–

		Pre-image map	reverse of fm	–
		Latent space dim.	5/10	
Sketchy	1e−4	Input	64 × 64 × 3	–

		Feature-map (fm)	Conv 40 × 4 × 4;
			Conv 80 × 4 × 4;
			Conv 160 × 4 × 4;
			FC 128 (Linear)
				–

		Pre-image map	reverse of fm	–
		Latent space dim.	30	

Proof. Let $\lambda_i, \boldsymbol{h}_i$ are given by (3). Using (2) to substitute $\boldsymbol{V}$ and $\boldsymbol{U}$ in (1) yields:

$$\mathcal{J}_t(\boldsymbol{V},\boldsymbol{U},\Lambda,\boldsymbol{H}) = \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{\eta_1}{2}\left(\frac{1}{\eta_1^2}\sum_{i=1}^{N}\boldsymbol{h}_i\boldsymbol{\phi}_1(\boldsymbol{x}_i)^\top \sum_{j=1}^{N}\boldsymbol{\phi}_1(\boldsymbol{x}_j)\boldsymbol{h}_j^\top\right)$$

$$+\frac{\eta_2}{2}\left(\frac{1}{\eta_2^2}\sum_{i=1}^{N}\boldsymbol{h}_i\boldsymbol{\phi}_2(\boldsymbol{y}_i)^\top \sum_{j=1}^{N}\boldsymbol{\phi}_2(\boldsymbol{y}_j)\boldsymbol{h}_j^\top\right)$$

$$=\sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{\eta_1}{2}\left(\frac{1}{\eta_1^2}\boldsymbol{H}\boldsymbol{K}_1\boldsymbol{H}^\top\right) + \frac{\eta_2}{2}\left(\frac{1}{\eta_2^2}\boldsymbol{H}\boldsymbol{K}_2\boldsymbol{H}^\top\right)$$

$$=\sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{1}{2}\left(\boldsymbol{H}\left[\frac{1}{\eta_1}\boldsymbol{K}_1 + \frac{1}{\eta_2}\boldsymbol{K}_2\right]\boldsymbol{H}^\top\right).$$

From (3), we get:

$$\mathcal{J}_t(\boldsymbol{V},\boldsymbol{U},\Lambda,\boldsymbol{H}) = \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{1}{2}\left(\boldsymbol{H}\boldsymbol{H}^\top\Lambda\right)$$

$$= \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{1}{2}\sum_{i=1}^{N}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i = 0.$$

Proposition 2. *Let $J(\boldsymbol{x}) : \mathbb{R}^N \to \mathbb{R}$ be a smooth function, for all $\boldsymbol{x} \in \mathbb{R}^N$ and for $c \in \mathbb{R}_{>0}$, define $\bar{J}(\boldsymbol{x}) := J(\boldsymbol{x}) + \frac{c}{2}J(\boldsymbol{x})^2$. Assuming $(1 + cJ(\boldsymbol{x})) \neq 0$, then $\boldsymbol{x}^\star$ is the stationary point of $\bar{J}(\boldsymbol{x})$ iff $\boldsymbol{x}^\star$ is the stationary point for $J(\boldsymbol{x})$.*

Proof. Let $\boldsymbol{x}^\star$ be a stationary point of $J(\boldsymbol{x})$, meaning that $\nabla J(\boldsymbol{x}^\star) = 0$. The stationary points for $\bar{J}(\boldsymbol{x})$ can be obtained from:

$$\frac{d\bar{J}}{d\boldsymbol{x}} = (\nabla J(\boldsymbol{x}) + cJ(\boldsymbol{x})\nabla J(\boldsymbol{x})) = (1 + cJ(\boldsymbol{x}))\,\nabla J(\boldsymbol{x}). \tag{B.1}$$

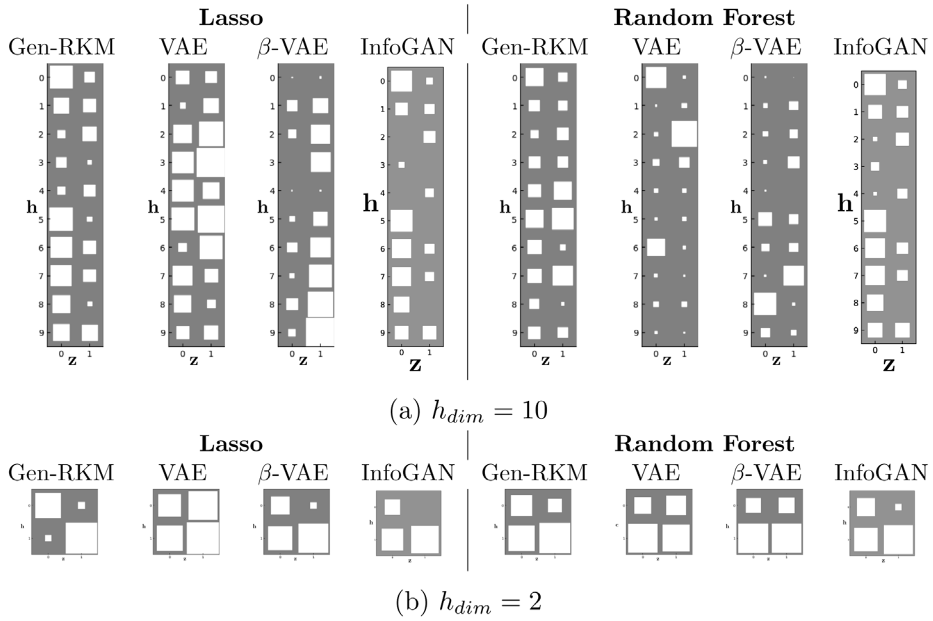
It is easy to see from 2 that if $\boldsymbol{x} = \boldsymbol{x}^*$, $\nabla J(\boldsymbol{x}^*) = 0$, we have that $\left.\frac{d\bar{J}}{d\boldsymbol{x}}\right|_{\boldsymbol{x}^*} = 0$, meaning that all the stationary points of $J(\boldsymbol{x})$ are stationary points of $\bar{J}(\boldsymbol{x})$.

To show the other way, let $\boldsymbol{x}^\star$ be stationary point of $\bar{J}(\boldsymbol{x})$ i.e. $\nabla\bar{J}(\boldsymbol{x}^\star) = 0$. Assuming $(1 + cJ(\boldsymbol{x}^\star)) \neq 0$, then from (B.1) for all $c \in \mathbb{R}_{>0}$, we have

$$\left(1 + cJ(\boldsymbol{x}^\star)\right)\nabla J(\boldsymbol{x}^\star) = 0,$$

implying that $\nabla J(\boldsymbol{x}^\star) = 0$.

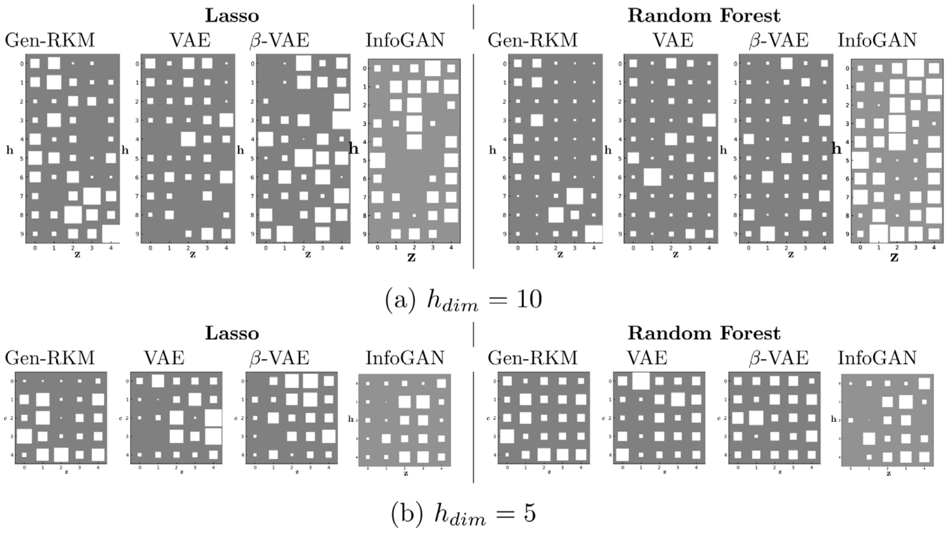
Based on the above propositions, we stabilize our original objective function (1) to keep it bounded and hence is suitable



(a) $h_{dim} = 10$

(b) $h_{dim} = 2$

Fig. F.8. Relative importance matrix as computed by Lasso and Random Forest regressors on DSprites dataset for $h_{dim} \in \{10, 2\}$ against the underlying data generating factors $z_{dim} = 2$ corresponding to x, y positions of object.



(a) $h_{dim} = 10$

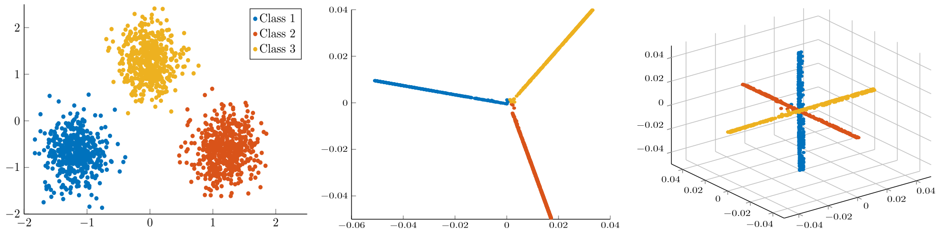
(b) $h_{dim} = 5$

Fig. F.9. Relative importance matrix as computed by Lasso and Random Forest regressors on Teapot dataset for $h_{dim} \in \{10, 5\}$ against the underlying data generating factors $z_{dim} = 5$ corresponding to azimuth, elevation and colors red, green and blue of the teapot object.

Table G.5

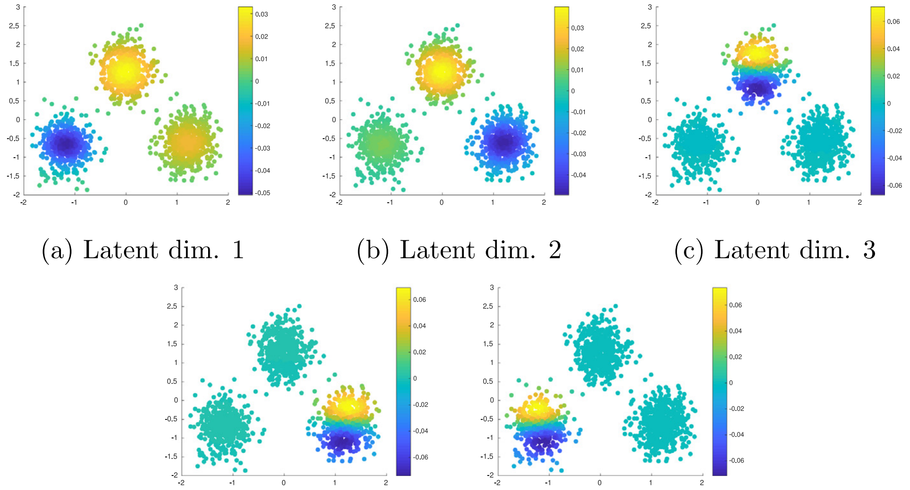
Training time per epoch comparisons (in seconds with standard deviation over 10 epochs) on MNIST and CelebA datasets. The architecture is the same as shown in Table D.4 with mini-batch size 100 and batch-size 2000. In both the cases Info-GAN is the most computationally expensive due to the additional auxiliary network and two backward passes per iteration. β-TCVAE has the second worst computation times due to relatively more complicated ELBO objective. VAE is marginally better in case of MNIST whereas the Gen-RKM outperforms in case of CelebA. This could be due to significantly large number of parameters for CelebA architecture which increases the computational burden of VAE. However, due to the fixed computational cost of eigendecomposition (for fixed mini-batch size), the latent variables in Gen-RKM are computed with this fixed cost.

Dataset	Gen-RKM	VAE	Info-GAN	β-TCVAE
MNIST	0.275 (±0.042)	**0.223** (±0.013)	0.372 (±0.044)	0.318 (±0.031)
CelebA	**4.274** (±0.147)	4.308 (±0.112)	5.815 (±0.131)	5.201 (±0.152)



(a) Input data (b) PCs 1 & 2 (c) PCs 3,4 & 5

Fig. G.10. Visualization of the toy dataset together with the first 5 Principal Components (PCs) of the latent space of the Gen-RKM model.



(d) Latent dim. 4 (e) Latent dim. 5

Fig. G.11. Visualization of the traversals along the Principal components. Here the color corresponds to the value of the datapoint in latent space.

for minimization with Gradient-descent methods. Without the reconstruction errors, the stabilized objective function is

$$\min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{h}_i} \mathcal{J}_t + \frac{c}{2}\mathcal{J}_t^2.$$

Denote $\bar{J} = \mathcal{J}_t + \frac{c_{stab}}{2}\mathcal{J}_t^2$. Since the derivatives of $\mathcal{J}_t$ are given by (2), the stationary points of $\bar{J}$ are:

$$\begin{cases} \frac{\partial \bar{J}}{\partial \boldsymbol{V}} = (1 + c_{stab}\mathcal{J}_t)\left(-\sum_{i=1}^{N}\boldsymbol{\phi}_1(\boldsymbol{x}_i)\boldsymbol{h}_i^\top + \eta_1\boldsymbol{V}\right) = 0, \\ \frac{\partial \bar{J}}{\partial \boldsymbol{U}} = (1 + c_{stab}\mathcal{J}_t)\left(-\sum_{i=1}^{N}\boldsymbol{\phi}_2(\boldsymbol{y}_i)\boldsymbol{h}_i^\top + \eta_2\boldsymbol{U}\right) = 0, \\ \frac{\partial \bar{J}}{\partial \boldsymbol{h}_i} = (1 + c_{stab}\mathcal{J}_t)\left(-\boldsymbol{V}^\top\boldsymbol{\phi}_1(\boldsymbol{x}_i) - \boldsymbol{U}^\top\boldsymbol{\phi}_2(\boldsymbol{y}_i) + \lambda\boldsymbol{h}_i\right) = 0, \end{cases}$$

which gives the following solution:

$$\begin{cases} \boldsymbol{V} = \frac{1}{\eta_1}\sum_{i=1}^{N}\boldsymbol{\phi}_1(\boldsymbol{x}_i)\boldsymbol{h}_i^\top, \\ \boldsymbol{U} = \frac{1}{\eta_2}\sum_{i=1}^{N}\boldsymbol{\phi}_2(\boldsymbol{y}_i)\boldsymbol{h}_i^\top, \\ \lambda\boldsymbol{h}_i = \boldsymbol{V}^\top\boldsymbol{\phi}_1(\boldsymbol{x}_i) + \boldsymbol{U}^\top\boldsymbol{\phi}_2(\boldsymbol{y}_i), \end{cases}$$

assuming $1 + c_{stab}\mathcal{J}_t \neq 0$. Elimination of $\boldsymbol{V}$ and $\boldsymbol{U}$ yields the following eigenvalue problem $\left[\frac{1}{\eta_1}\boldsymbol{K}_1 + \frac{1}{\eta_2}\boldsymbol{K}_2\right]\boldsymbol{H}^\top = \boldsymbol{H}^\top\boldsymbol{\Lambda}$, which is indeed the same solution for $c_{stab} = 0$ in (1) and (3).

Appendix C. Centering of kernel matrix

Centering of the kernel matrix is done by the following equation:

$$\boldsymbol{K}_c = \boldsymbol{K} - N^{-1}\mathbf{1}\mathbf{1}^\top\boldsymbol{K} - N^{-1}\boldsymbol{K}\mathbf{1}\mathbf{1}^\top + N^{-2}\mathbf{1}\mathbf{1}^\top\boldsymbol{K}\mathbf{1}\mathbf{1}^\top, \tag{C.1}$$

where $\mathbf{1}$ denotes an N-dimensional vector of ones and $\boldsymbol{K}$ is either $\boldsymbol{K}_1$ or $\boldsymbol{K}_2$.

Appendix D. Architecture details

See Tables D.3 and D.4 for details on model architectures, datasets and hyperparameters used in this paper and double precision is used for training the Gen-RKM model. The PyTorch library in Python was used as the programming language with a 8GB NVIDIA QUADRO P4000 GPU.

Appendix E. Bilinear interpolation

Given four vectors $\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3$ and $\boldsymbol{h}_4$ (reconstructed images from these vectors are shown at the corners of Figs. 3(e), 3(f)), the interpolated vector $\boldsymbol{h}^\star$ is given by:

$$\boldsymbol{h}^\star = (1-\alpha)(1-\gamma)\boldsymbol{h}_1 + \alpha(1-\gamma)\boldsymbol{h}_2 + \gamma(1-\alpha)\boldsymbol{h}_3 + \gamma\alpha\boldsymbol{h}_4,$$

with $0 \leq \alpha, \gamma \leq 1$. This $\boldsymbol{h}^\star$ is then used in step 8 of the generation procedure of Gen-RKM algorithm (see Algorithm 1) to compute $\boldsymbol{x}^\star$.

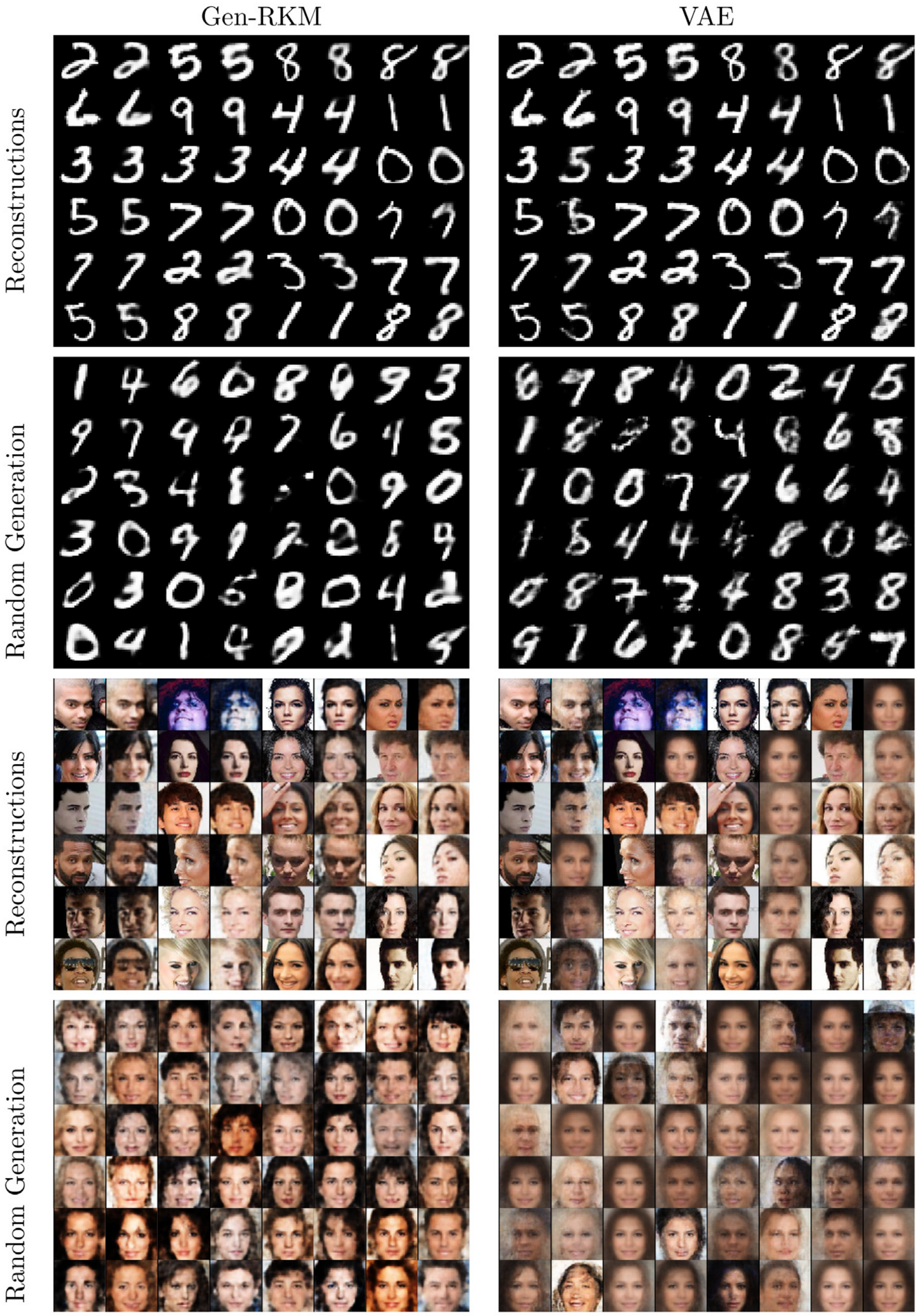


Fig. G.12. Comparing Gen-RKM and standard VAE for reconstruction and generation quality. In reconstruction MNIST and reconstruction CelebA, uneven columns correspond to the original image, even columns to the reconstructed image.

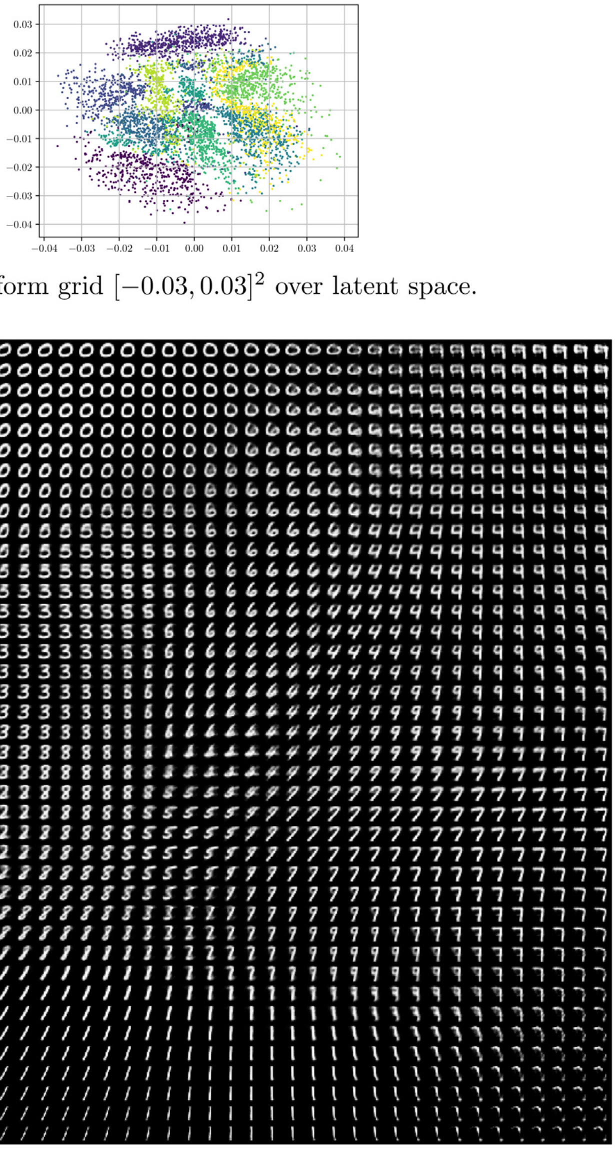
Appendix F. Visualizing the disentanglement metric

In this section we show the Hinton plots to visualize the disentanglement scores as shown in Table 2. Following the conventions of Eastwood and Williams (2018), $\boldsymbol{z}$ represents the ground-truth data generating factors. Figs. F.8 and F.9 show the Hinton plots on DSprites and Teapot datasets using Lasso and Random Forest regressors for various algorithms. Here the white square size indicates the magnitude of the *relative importance* of the latent code $\boldsymbol{h}_i$ in predicting $\boldsymbol{z}_i$.

Appendix G. Further empirical results

G.1. Illustration on toy example using a Gaussian kernel

Here we demonstrate the application of Gen-RKM/Kernel PCA using a Gaussian kernel with $\sigma = 0.5$ on a 3 mode Gaussian dataset. The dataset is shown in Fig. G.10 together with the first 5 Principal Components (PCs) of the latent space. The method looks for PCs that explain the most variance. One can see that moving along first component in latent space correspond to changing classes 3 → 2 → 1, whereas moving along the second component corresponds to changing classes 2 → 3 → 1. For PCs 3 to 5, the model shows disentanglement of the 3 classes, i.e. each Gaussian cluster is mapped to a specific component. Moving along one of



(a) Uniform grid $[-0.03, 0.03]^2$ over latent space.

(b) Generated images from latent vectors obtained from the uniform grid in the latent space depicting smoothness of latent space.

Fig. G.13. MNIST: Latent space exploration.

these components only changes the *within class* variation. This behavior is further confirmed by the experiment in Fig. G.11. Here we visualize again the dataset where now the color corresponds to the value of the datapoint in latent space.

References

Alemi, A., Fischer, I., Dillon, J., & Murphy, K. (2017). Deep variational information bottleneck. In *5th international conference on learning representations, ICLR*.

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. Berlin, Heidelberg: Springer-Verlag.

Bouchacourt, D., Tomioka, R., & Nowozin, S. (2018). Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Thirty-second AAAI conference on artificial intelligence*.

Bui, A. T., Im, J.-K., Apley, D. W., & Runger, G. C. (2019). Projection-free kernel principal component analysis for denoising. *Neurocomputing*.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., & Lerchner, A. (2018). Understanding disentangling in β-VAE. arXiv preprint arXiv:1804.03599.

Chen, M., & Denoyer, L. (2017). Multi-view generative adversarial networks. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 175–188). Springer.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems* (pp. 2172–2180).

Chen, T. Q., Li, X., Grosse, R. B., & Duvenaud, D. K. (2018). Isolating sources of disentanglement in variational autoencoders. In *Advances in neural information processing systems* (pp. 2610–2620).

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285.

Eastwood, C., & Williams, C. K. I. (2018). A framework for the quantitative evaluation of disentangled representations. In *International conference on learning representations*. https://openreview.net/forum?id=By-7dz-AZ.

Florensa, C., Held, D., Geng, X., & Abbeel, P. (2018). Automatic goal generation for reinforcement learning agents. In *Proceedings of machine learning research: Vol. 80, Proceedings of the 35th international conference on machine learning* (pp. 1515–1528). Stockholmsmassan, Stockholm Sweden: PMLR.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., & Bengio, Y. (2014). Generative adversarial nets. In *Advances* *in neural information processing systems 27: annual conference on neural information processing systems 2014 (pp. 2672–2680)*.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York, NY, USA: Springer New York Inc..

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANS trained by a two time-scale update rule converge to a local Nash equilibrium. In *NIPS'17*, *Proceedings of the 31st international conference on neural information processing systems* (pp. 6629–6640). USA: Curran Associates Inc., http://dl.acm.org/citation.cfm?id=3295222.3295408.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., & Lerchner, A. (2017). Beta-VAE: Learning basic visual concepts with a constrained variational framework.. *5th international conference on learning representations, ICLR*, *2*(5), 6.

Honeine, P., & Richard, C. (2011). Preimage problem in kernel-based machine learning. *IEEE Signal Processing Magazine*, *28*(2), 77–88.

Houthuys, L., & Suykens, J. A. K. (2018). Tensor learning in multi-view kernel PCA. In *27th international conference on artificial neural networks ICANN: Vol. 11140* (pp. 205–215).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *2nd international conference on learning representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, conference track proceedings*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *Tech. rep.*, University of Toronto.

Kwok, J. T., & Tsang, I. W.-H. (2003). The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, *15*, 1517–1525.

Larochelle, H., & Bengio, Y. (2008). Classification using discriminative restricted Boltzmann machines. In *Proceedings of the 25th international conference on machine learning* (pp. 536–543). Helsinki, Finland: ACM Press.

Lawrence, N. (2005). Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *JMLR*, *6*, 1783–1816, http://dl.acm.org/citation.cfm?id=1046920.1194904.

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/.

LeCun, Y., Huang, F. J., & Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Computer vision and pattern recognition, 2004: Vol. 2* (pp. II–97–104).

Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of international conference on computer vision*.

Liu, M.-Y., & Tuzel, O. (2016). Coupled generative adversarial networks. In *Advances in neural information processing systems: Vol. 29* (pp. 469–477). Curran Associates, Inc..

Matthey, L., Higgins, I., Hassabis, D., & Lerchner, A. (2017). Dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/.

McInnes, L., Healy, J., Saul, N., & Großberger, L. (2018). UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, *3*(29), 861. http://dx.doi.org/10.21105/joss.00861.

Mercer, J. (1909). Functions of positive and negative type, and their connection the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, *209*(441–458), 415–446.

Mika, S., Schölkopf, B., Smola, A., Müller, K.-R., Scholz, M., & Rätsch, G. (1999). Kernel PCA and de-noising in feature spaces. In *Proceedings of the 1998 conference on advances in neural information processing systems II* (pp. 536–542). MIT Press.

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., & Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. In *NIPS'16*, (pp. 2360–2368). USA: Curran Associates Inc..

Rabiner, L. R., & Juang, B.-H. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, *3*(1), 4–16.

Ridgeway, K. (2016). A survey of inductive biases for factorial representation-learning. *CoRR*, *abs/1612.05299*, arXiv:1612.05299.

Rockafellar, R. T. (1974). *Conjugate duality and optimization*. SIAM.

Salakhutdinov, R., & Hinton, G. (2009). Deep Boltzmann machines. In *Proceedings of the 12th international conference on artificial intelligence and statistics Volume 5 of JMLR*.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *ICML '07* (pp. 791–798). Corvalis, Oregon: ACM Press.

Sangkloy, P., Burnell, N., Ham, C., & Hays, J. (2016a). The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (proceedings of SIGGRAPH)*.

Sangkloy, P., Burnell, N., Ham, C., & Hays, J. (2016b). The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics*, *35*(4), 1–12.

Schmidhuber, J. (1992). Learning factorial codes by predictability minimization. *Neural Computation*, *4*(6), 863–879.

Scholkopf, B., & Smola, A. J. (2001). *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Cambridge, MA, USA: MIT Press.

Schreurs, J., & Suykens, J. A. K. (2018). Generative Kernel PCA. In *European symposium on artificial neural networks, computational intelligence and machine learning* (pp. 129–134).

Smolensky, P. (1986). In D. E. Rumelhart, J. L. McClelland, & C. PDP Research Group (Eds.), *Parallel distributed processing: explorations in the microstructure of cognition, Vol. 1* (pp. 194–281). Cambridge, MA, USA: MIT Press.

Srivastava, N., & Salakhutdinov, R. R. (2012). Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems* (pp. 2222–2230).

Suykens, J. A. K. (2017). Deep restricted kernel machines using conjugate feature duality. *Neural Computation*, *29*(8), 2123–2163.

Suykens, J. A. K., Van Gestel, T., De Brabanter, J., De Moor, B., & Vandewalle, J. (2002). *Least squares support vector machines*. River Edge, NJ: World Scientific.

Suykens, J. A. K., Van Gestel, T., Vandewalle, J., & De Moor, B. (2003). A support vector machine formulation to PCA analysis and its kernel version. *IEEE Transactions on Neural Networks*, *14*(2), 447–450.

Suzuki, M., Nakayama, K., & Matsuo, Y. (2016). Joint multimodal learning with deep generative models. arXiv preprint arXiv:1611.01891.

Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal Of The Royal Statistical Society, series B*, *61*(3), 611–622.

Tran, L., Yin, X., & Liu, X. (2017). Disentangled representation learning GAN for pose-invariant face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1415–1424).

Van Den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *ICML'16*, *Proceedings of the 33rd international conference on international conference on machine learning: Vol. 48* (pp. 1747–1756). JMLR.org.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11*, 3371–3408.

Weston, J., Schölkopf, B., & Bakir, G. H. (2004). Learning to find pre-images. In *NIPS 16* (pp. 449–456).

Wu, M., & Goodman, N. (2018). Multimodal generative models for scalable weakly-supervised learning. In *Advances in neural information processing systems* (pp. 5575–5585).

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv:cs.LG/1708.07747.

Yeh, R. A., Chen, C., Yian Lim, T., Schwing, A. G., Hasegawa-Johnson, M., & Do, M. N. (2017). Semantic image inpainting with deep generative models. In *The IEEE conference on computer vision and pattern recognition*.