arXiv:1906.08144v5 [cs.LG] 6 Feb 2020

Generative Restricted Kernel Machines

Arun Pandey, Joachim Schreurs & Johan A. K. Suykens
Department of Electrical Engineering, ESAT-STADIUS,
KU Leuven. Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
{arun.pandey,joachim.schreurs,johan.suykens}@esat.kuleuven.be

February 7, 2020

Abstract

We introduce a novel framework for generative models based on Restricted Kernel Machines (RKMs) with multi-view generation and uncorrelated feature learning capabilities, called Gen-RKM. To incorporate multi-view generation, this mechanism uses a shared representation of data from various views. The mechanism is flexible to incorporate both kernel-based, (deep) neural network and Convolutional based models within the same setting. To update the parameters of the network, we propose a novel training procedure 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 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 [1], exploration in reinforcement learning [2], collaborative filtering [3], inpainting [4] and many more.

Some examples of graphical models based on a probabilistic framework with latent variables are Variational Auto-Encoders [5] and Restricted Boltzmann Machines (RBMs) [6, 7]. More recently proposed models are based on adversarial training such as Generative Adversarial Networks (GANs) [8] and its many variants. Furthermore, auto-regressive models such as Pixel Recurrent Neural Networks (PixelRNNs) [9] 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 [7]. 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 comprised of different representations of the data, or 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 [10, 11, 12]. Also, it is among the goals of the latent variable modelling to model the description of data in terms of *uncorrelated or independent* components. Some classical examples are Independent Component Analysis; Hidden Markov models [13]; Probabilistic Principal Component Analysis (PCA) [14]; Gaussian-Process Latent variable model [15] 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 separate factors of variation in the data for e.g. lighting conditions, style, colors, etc. The definition of disentanglement in the literature is not precise, however many believe that a representation with statistically independent variables is a good

starting point [16, 17]. Such representations extract information into a compact form which makes it possible to generate samples with specific characteristics [18, 19, 20, 21]. Additionally, these representations have been found to generalize better and be more robust against adversarial attacks [22].

In this work, we propose an alternative generative mechanism based on the framework of Restricted Kernel Machines (RKMs) [23], 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) [24] 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, [25] used this framework to develop tensor-based multi-view classification models and [26] showed how kernel PCA fits into this framework.

Contributions. 1) A novel multi-view generative model based on the RKM framework where multiple views of the data can be generated simultaneously. **2)** Two methods are proposed 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, (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 experiments 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.

This paper is organized as follows. In Section 2, we discuss the Gen-RKM training and generation mechanism when multiple data sources are available. In Section 3, we explain how the model incorporates both kernel methods and neural networks through the use of implicit and explicit feature maps respectively. When the feature maps are defined by neural networks, the Gen-RKM algorithm is explained in Section 4. 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. Additional supplementary materials are given in the Appendix A.

2 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 another.

2.1 Training

Similar to Energy-Based Models (EBMs, see [27] 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, as shown in Figure 1 is similar to Discriminative RBMs [28] 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 $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{y}_i \in \mathbb{R}^p$ comprising 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. Starting from the RKM interpretation of Kernel PCA, which gives an upper bound on the equality constrained Least-Squares Kernel PCA objective function [23], and applying the feature-maps $\phi_1 : \mathbb{R}^d \mapsto \mathbb{R}^{d_f}$ and $\phi_2 : \mathbb{R}^p \mapsto \mathbb{R}^{p_f}$ to the input data points, the training objective function $\mathcal{J}_t$ for generative

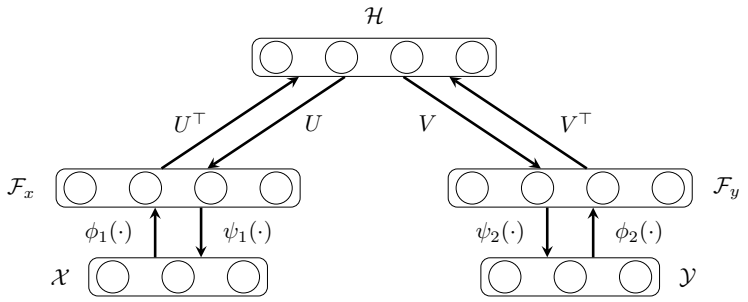


Figure 1: Gen-RKM schematic representation modeling a common subspace $\mathcal{H}$ between two data sources $\mathcal{X}$ and $\mathcal{Y}$. The ϕ_1, ϕ_2 are the feature maps ($\mathcal{F}_x$ and $\mathcal{F}_y$ represent the feature-spaces) corresponding to the two data sources. While ψ_1, ψ_2 represent the pre-image maps. The interconnection matrices $\boldsymbol{U}, \boldsymbol{V}$ model dependencies between latent variables and the mapped data sources.

RKM is given by[1]:

$$\begin{aligned}\mathcal{J}_t = \sum_{i=1}^{N} \left(-\phi_1(\boldsymbol{x}_i)^\top \boldsymbol{U}\boldsymbol{h}_i - \phi_2(\boldsymbol{y}_i)^\top \boldsymbol{V}\boldsymbol{h}_i + \frac{1}{2}\boldsymbol{h}_i^\top \boldsymbol{\Lambda}\boldsymbol{h}_i \right) \\ + \frac{\eta_1}{2}\operatorname{Tr}(\boldsymbol{U}^\top\boldsymbol{U}) + \frac{\eta_2}{2}\operatorname{Tr}(\boldsymbol{V}^\top\boldsymbol{V}),\end{aligned} \tag{1}$$

where $\boldsymbol{U} \in \mathbb{R}^{d_f \times s}$ and $\boldsymbol{V} \in \mathbb{R}^{p_f \times s}$ are the unknown interaction matrices, and $\boldsymbol{h}_i \in \mathbb{R}^s$ are the latent variables modeling a common subspace $\mathcal{H}$ between the two input spaces $\mathcal{X}$ and $\mathcal{Y}$ (see Figure 1). The derivation of this objective function is given in the Appendix A.1.

Given $\eta_1 > 0$ and $\eta_2 > 0$ as regularization parameters, the stationary points of $\mathcal{J}_t$ are given by:

$$\begin{cases} \frac{\partial \mathcal{J}_t}{\partial \boldsymbol{h}_i} = 0 \implies \boldsymbol{\Lambda}\boldsymbol{h}_i = \boldsymbol{U}^\top\phi_1(\boldsymbol{x}_i) + \boldsymbol{V}^\top\phi_2(\boldsymbol{y}_i), \ \ \forall i \\ \frac{\partial \mathcal{J}_t}{\partial \boldsymbol{U}} = 0 \implies \boldsymbol{U} = \frac{1}{\eta_1}\sum_{i=1}^{N}\phi_1(\boldsymbol{x}_i)\boldsymbol{h}_i^\top \\ \frac{\partial \mathcal{J}_t}{\partial \boldsymbol{V}} = 0 \implies \boldsymbol{V} = \frac{1}{\eta_2}\sum_{i=1}^{N}\phi_2(\boldsymbol{y}_i)\boldsymbol{h}_i^\top. \end{cases} \tag{2}$$

Substituting $\boldsymbol{U}$ and $\boldsymbol{V}$ in the first equation above, denoting $\boldsymbol{\Lambda} = \operatorname{diag}\{\lambda_1, \ldots, \lambda_s\} \in \mathbb{R}^{s\times s}$ 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 $\boldsymbol{H} = \begin{bmatrix}\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\end{bmatrix} \in \mathbb{R}^{s\times N}$ 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 [29], positive-definite kernel functions $k_1 : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$, $k_2 : \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$ can be defined such that $k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle\phi_1(\boldsymbol{x}_i), \phi_1(\boldsymbol{x}_j)\rangle$, and $k_2(\boldsymbol{y}_i, \boldsymbol{y}_j) = \langle\phi_2(\boldsymbol{y}_i), \phi_2(\boldsymbol{y}_j)\rangle$, $\forall i, j = 1, \ldots, N$ forms the elements of corresponding kernel matrices. The feature maps ϕ_1 and ϕ_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 [30]. However, one can also define explicit feature maps, still preserving the positive-definiteness of the kernel function by construction [24].

2.2 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

[1]For convenience, it is assumed that all the feature vectors are centered in the feature space $\mathcal{F}$ using $\tilde{\phi}(\boldsymbol{x}) := \phi(\boldsymbol{x}) - \frac{1}{N}\sum_{i=1}^{N}\phi(\boldsymbol{x}_i)$. Otherwise, a centered kernel matrix could be obtained using Eq. 17 (Appendix A.4).

[2]While in the above section we have assumed that only two data sources (namely $\mathcal{X}$ and $\mathcal{Y}$) 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: $\left[\sum_{\ell=1}^{M}\frac{1}{\eta_\ell}\boldsymbol{K}_\ell\right]\boldsymbol{H}^\top = \boldsymbol{H}^\top\boldsymbol{\Lambda}$.

remaining variables that minimizes the energy [27]. Given the learned interconnection matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, and a given latent variable $\boldsymbol{h}^\star$, consider the following objective function:

$$\mathcal{J}_g = -\phi_1(\boldsymbol{x}^\star)^\top \boldsymbol{U}\boldsymbol{h}^\star - \phi_2(\boldsymbol{y}^\star)^\top \boldsymbol{V}\boldsymbol{h}^* + \frac{1}{2}\phi_1(\boldsymbol{x}^\star)^\top \phi_1(\boldsymbol{x}^\star) + \frac{1}{2}\phi_2(\boldsymbol{y}^\star)^\top \phi_2(\boldsymbol{y}^\star), \tag{4}$$

with an additional regularization term on data sources. Here $\mathcal{J}_g$ denotes the objective function for generation. 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:

$$\begin{cases} \frac{\partial \mathcal{J}_g}{\partial \phi_1(\boldsymbol{x}^\star)} = 0 \implies \phi_1(\boldsymbol{x}^\star) = \boldsymbol{U}\boldsymbol{h}^\star, \\ \frac{\partial \mathcal{J}_g}{\partial \phi_2(\boldsymbol{y}^\star)} = 0 \implies \phi_2(\boldsymbol{y}^\star) = \boldsymbol{V}\boldsymbol{h}^\star. \end{cases} \tag{5}$$

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

$$\phi_1(\boldsymbol{x}^\star) = \left(\frac{1}{\eta_1}\sum_{i=1}^{N}\phi_1(\boldsymbol{x}_i)\boldsymbol{h}_i^\top\right)\boldsymbol{h}^\star, \quad \phi_2(\boldsymbol{y}^\star) = \left(\frac{1}{\eta_2}\sum_{i=1}^{N}\phi_2(\boldsymbol{y}_i)\boldsymbol{h}_i^\top\right)\boldsymbol{h}^\star. \tag{6}$$

To obtain the generated data, one now needs to compute the inverse images of the feature maps $\phi_1(\cdot)$ and $\phi_2(\cdot)$ in the respective input spaces, i.e., solve the *pre-image problem*. We seek to find the functions $\psi_1 \colon \mathbb{R}^{d_f} \mapsto \mathbb{R}^d$ and $\psi_2 \colon \mathbb{R}^{p_f} \mapsto \mathbb{R}^p$ corresponding to the two data-sources, such that $(\psi_1 \circ \phi_1)(\boldsymbol{x}^\star) \approx \boldsymbol{x}^\star$ and $(\psi_2 \circ \phi_2)(\boldsymbol{y}^\star) \approx \boldsymbol{y}^\star$, where $\phi_1(\boldsymbol{x}^\star)$ and $\phi_2(\boldsymbol{y}^\star)$ are calculated using Eq. 6.

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 [31]. However, various approximation techniques have been proposed [32, 33, 34, 35] which could be used to obtain the approximate pre-image $\hat{\boldsymbol{x}}$ of $\phi_1(\boldsymbol{x}^\star)$. In section 3.1, we employ one such technique to demonstrate the applicability in our model, and consequently generate the multi-view data. One could also define explicit pre-image maps. In section 3.2, 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.

3 Implicit & Explicit feature map

3.1 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 [26]. Left multiplying Eq. 6 by $\phi_1(\boldsymbol{x}_i)^\top$ and $\phi_2(\boldsymbol{y}_i)^\top$, $\forall i = 1, \ldots, N$, we obtain:

$$\boldsymbol{k}_{\boldsymbol{x}^\star} = \frac{1}{\eta_1}\boldsymbol{K}_1\boldsymbol{H}^\top\boldsymbol{h}^\star, \quad \boldsymbol{k}_{\boldsymbol{y}^\star} = \frac{1}{\eta_2}\boldsymbol{K}_2\boldsymbol{H}^\top\boldsymbol{h}^\star, \tag{7}$$

where, $\boldsymbol{k}_{\boldsymbol{x}^\star} = [k(\boldsymbol{x}_1, \boldsymbol{x}^\star), \ldots, k(\boldsymbol{x}_N, \boldsymbol{x}^\star)]^\top$ represents the *similarities* between $\phi_1(\boldsymbol{x}^\star)$ and training data points in the feature space, and $\boldsymbol{K}_1 \in \mathbb{R}^{N\times N}$ represents the centered kernel matrix of $\mathcal{X}$. Similar conventions follow for $\mathcal{Y}$ respectively. Using the *kernel-smoother* method [36], the pre-images are given by:

$$\hat{\boldsymbol{x}} = \psi_1\left(\phi_1(\boldsymbol{x}^\star)\right) = \frac{\sum_{j=1}^{n_r}\tilde{k}_1(\boldsymbol{x}_j, \boldsymbol{x}^\star)\boldsymbol{x}_j}{\sum_{j=1}^{n_r}\tilde{k}_1(\boldsymbol{x}_j, \boldsymbol{x}^\star)}, \quad \hat{\boldsymbol{y}} = \psi_2\left(\phi_2(\boldsymbol{y}^\star)\right) = \frac{\sum_{j=1}^{n_r}\tilde{k}_2(\boldsymbol{y}_j, \boldsymbol{y}^\star)\boldsymbol{y}_j}{\sum_{j=1}^{n_r}\tilde{k}_2(\boldsymbol{y}_j, \boldsymbol{y}^\star)}, \tag{8}$$

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 Eq. 8) between 0 and 1 and n_r the number of closest points based on the similarity defined by kernels $\tilde{k}_1$ and $\tilde{k}_2$.

3.2 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 Eq. 3. In the experiments, we use a set of (convolutional) neural networks as the feature maps $\phi_{\boldsymbol{\theta}}(\cdot)$. Another (transposed convolutional) neural network is used for the pre-image map $\psi_{\boldsymbol{\zeta}}(\cdot)$ [37]. The network parameters $\{\boldsymbol{\theta}, \boldsymbol{\zeta}\}$ are learned by minimizing the reconstruction errors defined by $\mathcal{L}_1(\boldsymbol{x}, \psi_{1_{\boldsymbol{\zeta}_1}}(\phi_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x})))$ and $\mathcal{L}_2(\boldsymbol{y}, \psi_{2_{\boldsymbol{\zeta}_2}}(\phi_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y})))$. In our experiments, we use the mean-squared errors $\mathcal{L}_1(\boldsymbol{x}, \psi_{1_{\boldsymbol{\zeta}_1}}(\phi_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}))) = \frac{1}{N}\sum_{i=1}^{N} \left\|\boldsymbol{x}_i - \psi_{1_{\boldsymbol{\zeta}_1}}(\phi_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}_i))\right\|_2^2$ and $\mathcal{L}_2(\boldsymbol{y}, \psi_{2_{\boldsymbol{\zeta}_2}}(\phi_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}))) = \frac{1}{N}\sum_{i=1}^{N} \left\|\boldsymbol{y}_i - \psi_{2_{\boldsymbol{\zeta}_2}}(\phi_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}_i))\right\|_2^2$, however, in principle, one can use any other loss appropriate to the dataset. Here $\phi_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}_i)$ and $\phi_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}_i)$ are computed from Eq. 6, 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 [23]. 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 A.3 for details). The combined training objective is given by:

$$\min_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\boldsymbol{\zeta}_1,\boldsymbol{\zeta}_2} \mathcal{J}_c \;\; = \mathcal{J}_{stab} + \frac{c_{\text{acc}}}{2N}\left(\sum_{i=1}^{N}\left[\mathcal{L}_1(\boldsymbol{x}_i, \psi_{1_{\boldsymbol{\zeta}_1}}(\phi_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}_i))) + \mathcal{L}_2(\boldsymbol{y}_i, \psi_{2_{\boldsymbol{\zeta}_2}}(\phi_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}_i)))\right]\right), \tag{9}$$

where $c_{\text{acc}} \in \mathbb{R}^+$ 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.

There is also an intuitive connection between Gen-RKM and autoencoders. Namely, the properties of kernel PCA resemble the objectives of the 3 variations of an autoencoder: standard [38], VAE [5] and β-VAE [39]. **1)** Similar to an autoencoder, Gen-RKM minimizes the reconstruction error in the loss function (see Eq. 9), where kernel PCA which acts as a denoiser (the information is compressed in the principal components). **2)** By interpreting kernel PCA within the LS-SVM setting [24], the PCA analysis can take the interpretation of a one-class modeling problem with zero target value around which one maximizes the variance [40]. When choosing a good feature map, one expects the latent variables to be normally distributed around zero. This property resembles the added regularization term in the objective of the VAE [5], which is expressed as the Kullback-Leibler divergence between the encoder's distribution and a unit Gaussian as a prior on the latent variables. **3)** Kernel PCA gives uncorrelated components in feature space. While it was already shown that PCA does not give a good disentangled representation for images [41, 39]. Hence 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 show on the example in Figure 5. The uncorrelated components enhances the interpretation of the model.

4 The Gen-RKM Algorithm

Based on the previous analysis, we propose a novel algorithm, called the Gen-RKM algorithm, combining kernel learning and generative models. We show that this procedure is efficient to train and evaluate. It is also scalable to large datasets when using explicit feature maps. 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 the kernel matrix which is composed of the features from various views (see Eq. 3). The latent variables are given by the eigenvectors, which are then passed via a pre-image map to reconstruct the sample. Figure 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. $d_f + p_f \leq \frac{N}{m}$. While 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 Eq. 10 and Appendix A.2 for detailed derivation), where computing the $\boldsymbol{h}_i$ involves only matrix-multiplication which is readily parallelizable on modern GPUs:

$$\begin{bmatrix} \frac{1}{\eta_1}\Phi_{\boldsymbol{x}}\Phi_{\boldsymbol{x}}^\top & \frac{1}{\eta_1}\Phi_{\boldsymbol{x}}\Phi_{\boldsymbol{y}}^\top \\ \frac{1}{\eta_2}\Phi_{\boldsymbol{y}}\Phi_{\boldsymbol{x}}^\top & \frac{1}{\eta_2}\Phi_{\boldsymbol{y}}\Phi_{\boldsymbol{y}}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix} \boldsymbol{\Lambda}, \quad \begin{aligned} \Phi_{\boldsymbol{x}} &\coloneqq [\phi_1(\boldsymbol{x}_1), \ldots, \phi_1(\boldsymbol{x}_N)]\,, \\ \Phi_{\boldsymbol{y}} &\coloneqq [\phi_2(\boldsymbol{y}_1), \ldots, \phi_2(\boldsymbol{y}_N)]\,. \end{aligned} \tag{10}$$

Algorithm 1 Gen-RKM

Input: $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$, η_1, η_2, feature map $\phi_j(\cdot)$ - explicit *or* implicit via kernels $k_j(\cdot,\cdot)$, for $j \in \{1,2\}$
Output: Generated data $\boldsymbol{x}^\star$, $\boldsymbol{y}^\star$

procedure TRAIN
  if $\phi_j(\cdot)$ = Implicit **then**
    Hyperparameters: kernel specific
    Solve Eq. 3
    Select s principal components
  else if $\phi_j(\cdot)$ = Explicit **then**
    while not converged **do**
      $\{\boldsymbol{x}, \boldsymbol{y}\} \leftarrow$ {Get mini-batch}
      $\phi_1(\boldsymbol{x}) \leftarrow \boldsymbol{x}$; $\phi_2(\boldsymbol{y}) \leftarrow \boldsymbol{y}$
      do steps 4-5
      $\{\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{y})\} \leftarrow h$ (Eq. 6)
      $\{\boldsymbol{x}, \boldsymbol{y}\} \leftarrow \{\psi_1(\phi_1(\boldsymbol{x})), \psi_2(\phi_2(\boldsymbol{y}))\}$
      $\Delta\boldsymbol{\theta}_1 \propto -\nabla_{\boldsymbol{\theta}_1}\mathcal{J}_c$; $\Delta\boldsymbol{\theta}_2 \propto -\nabla_{\boldsymbol{\theta}_2}\mathcal{J}_c$
      $\Delta\boldsymbol{\zeta}_1 \propto -\nabla_{\boldsymbol{\zeta}_1}\mathcal{J}_c$; $\Delta\boldsymbol{\zeta}_2 \propto -\nabla_{\boldsymbol{\zeta}_2}\mathcal{J}_c$
    end while
  end if
end procedure

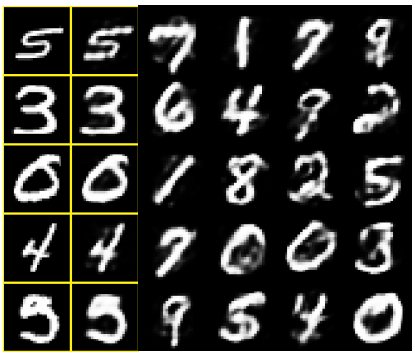
procedure GENERATION
  Select $\boldsymbol{h}^\star$
  if $\phi_j(\cdot)$ = Implicit **then**
    Hyperparameter: n_r
    Compute $\boldsymbol{k}_{\boldsymbol{x}^*}$, $\boldsymbol{k}_{\boldsymbol{y}^*}$ (Eq. 7)
    Get $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$ (Eq. 8)
  else if $\phi_j(\cdot)$ = Explicit **then**
    do steps 11-12
  end if
end procedure

5 Experiments

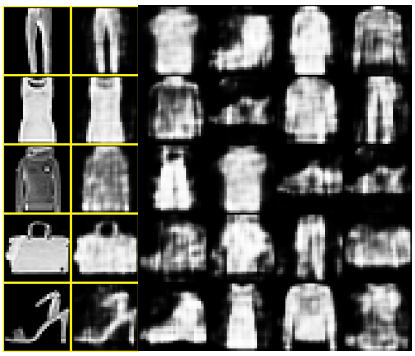
To demonstrate the applicability of the proposed framework and algorithm, we trained the Gen-RKM model on a variety of datasets commonly used to evaluate generative models: MNIST [42], Fashion-MNIST [43], CIFAR-10 [44], CelebA [45], Dsprites [46] and Teapot [41]. The experiments were performed using both the implicit feature map defined by a Gaussian kernel and parametric explicit feature maps defined by deep neural networks, either convolutional or fully connected. As explained in Section 2, in case of kernel methods, training only involves constructing the kernel matrix and solving the eigenvalue problem in Eq. 3. 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 using a kernel smoother. In case of explicit feature maps, we define $\phi_{1_{\boldsymbol{\theta}_1}}$ and $\psi_{1_{\boldsymbol{\zeta}_1}}$ as convolution and transposed-convolution neural networks, respectively [37]; and $\phi_{2_{\boldsymbol{\theta}_2}}$ and $\psi_{1_{\boldsymbol{\zeta}_2}}$ as fully-connected networks. The particular architecture details are outlined in Table 3 in the Appendix. The training procedure in case of explicitly defined maps consists of minimizing $\mathcal{J}_c$ using the Adam optimizer [47] 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, Eq. 3 is solved to update the value of $\mathcal{H}$. Information on the datasets and hyperparameters used for the experiments is given in Table 4 in the Appendix.

Generation:

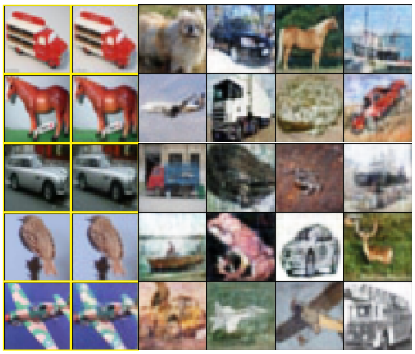
Qualitative examples: Figure 2 shows the generated images using a convolutional neural network and transposed-convolutional neural network as the feature map and pre-image map respectively. The first column in yellow-boxes shows the training samples and the second column on the right shows the reconstructed 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



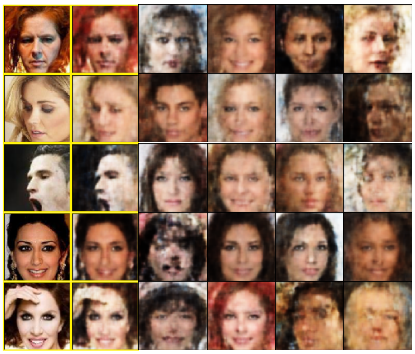
(a) MNIST



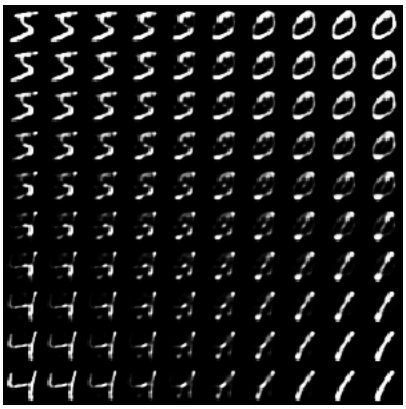
(b) Fashion-MNIST



(c) CIFAR-10



(d) CelebA

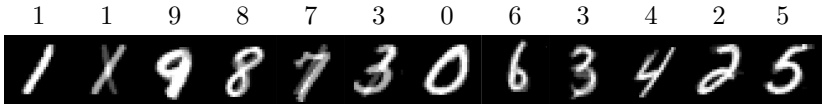


(e) Bilinear interpolation: MNIST

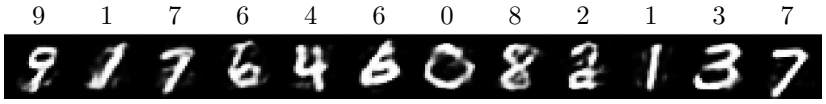
(f) Bilinear interpolation: CelebA

Figure 2: 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) shows the generated images through bilinear interpolations in the latent space.

Figure 3: Multi-view generation on CelebA dataset showing images and attributes.



(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) MNIST: Explicit feature maps and the corresponding pre-image maps are defined by the Convolutional Neural Networks.

(c) CIFAR-10: Explicit feature maps as Convolutional Neural Networks. Pre-images are computed using Transposed CNNs.

Figure 4: Multi-view Generation (images and labels) on various datasets using implicit and explicit feature maps.

generated by random sampling. To elucidate that the model has not merely memorized the training examples, we show the generated images via bilinear-interpolations in the latent space in 2e and 2f.
Comparison: We compare the proposed model with the standard VAE [5]. 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 3. We evaluate the performance qualitatively by comparing reconstruction and random sampling, the results are shown in Figure 8 in the Appendix. In order to quantitatively assess the quality of the randomly generated samples, we use the Fréchet Inception Distance (FID) introduced by [48]. The results are reported in Table 1. Experiments were repeated for different latent-space dimensions (h_{dim}), and we observe empirically that FID scores are better for the Gen-RKM. This is confirmed by the qualitative evaluation in Table 8, where the VAE generates smoother images. 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 eigendecomposition of the kernel matrix whose eigenvalue spectrum decreases rapidly depicting that most information is captured in 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).

Table 1: FID Scores [48] for randomly generated samples (smaller is better).

Dataset	**Algorithm**	**FID score**		
		$h_{dim} = 10$	$h_{dim} = 30$	$h_{dim} = 50$
MNIST	Gen-RKM	**89.825**	**130.497**	**131.696**
	VAE	250	234.749	205.282
CelebA	Gen-RKM	**103.299**	**84.403**	**85.121**
	VAE	286.039	245.738	225.783

Multi-view Generation: Figures 3 & 4 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. Figure 4a shows the generated images and labels when feature maps are only implicitly known i.e. through a Gaussian kernel. Figures 4b, 4c shows the same when using fully-connected networks as parametric functions to encode and decode labels. We can see that both the generated image and the generated label matches in most cases, albeit not all.

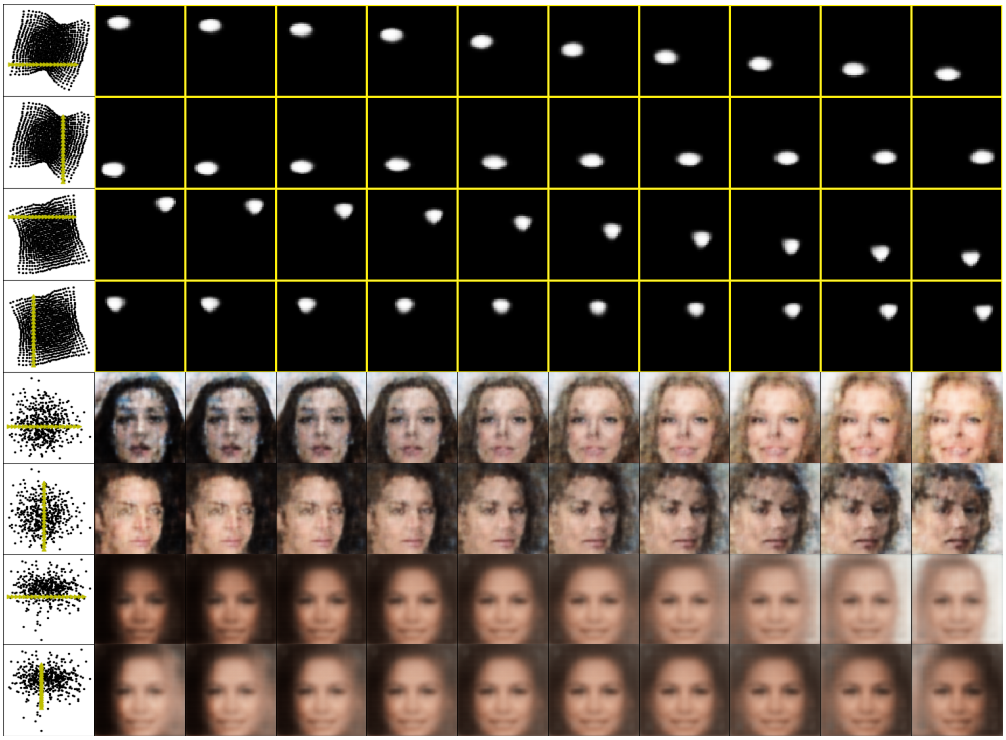


Figure 5: Exploring the learned uncorrelated-features by traversing along the eigenvectors. The first column shows the scatter plot of latent variables using the top two principal components. The green lines within, show the traversal in the latent space and the related rows show the corresponding reconstructed images.

Table 2: Disentanglement Metric on DSprites and Teapot dataset with Lasso and Random Forest regressor [41]. For disentanglement and completeness higher score is better, for informativeness, lower is better.

			Lasso			**Random Forest**		
	h_{dim}	**Algorithm**	**Disent.**	**Comple.**	**Inform.**	**Disent.**	**Comple.**	**Inform.**
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**
	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
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
	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

Disentanglement:
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 on Figure 5, where we explore the uncorrelated features learned by the models on the Dsprites and celebA dataset. 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 findout whether traversing along the eigenvectors, corresponds to traversing the generated image in one particular direction while preserving the

shape of the object. Rows 1 and 2 of Figure 5 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. Rows 5 and 6 shows 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 $\mathcal{H}$.

Comparison: To quantitatively assess disentanglement performance, we compare Gen-RKM with VAE [5] and beta-VAE [39] on the Dsprites and Teapot datasets [41]. The models have the same encoder/decoder architecture, optimization parameters and are trained until convergence, where the details are given in Table 3. The performance is measured using the proposed framework[3] of [41], which gives 3 measures: disentanglement, completeness and informativeness. The results are depicted 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 component are present. The above is confirmed by the Relative Importance Matrix on Figure 6 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 $h_{dim} = 10$. 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.

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 datatsets involving speech, images and texts; further analysis on other feature maps, pre-image methods, loss-functions and uncorrelated feature learning. Finally, this paper has demonstrated the applicability of the Gen-RKM framework, suggesting new research directions to be worth exploring.

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

References

[1] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[2] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic Goal Generation for Reinforcement Learning Agents," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmssan, Stockholm Sweden), pp. 1515–1528, PMLR, 10–15 Jul 2018.

[3] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *ICML '07*, (Corvalis, Oregon), pp. 791–798, ACM Press, 2007.

[4] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[5] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[6] P. Smolensky, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," ch. Information Processing in Dynamical Systems: Foundations of Harmony Theory, pp. 194–281, Cambridge, MA, USA: MIT Press, 1986.

[7] R. Salakhutdinov and G. Hinton, "Deep Boltzmann Machines," *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, vol. Volume 5 of JMLR, 2009.

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2672–2680, 2014.

[9] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel Recurrent Neural Networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1747–1756, JMLR.org, 2016.

[10] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational Autoencoder for Deep Learning of Images, Labels and Captions," NIPS'16, pp. 2360–2368, USA: Curran Associates Inc., 2016.

[11] M.-Y. Liu and O. Tuzel, "Coupled Generative Adversarial Networks," in *Advances in Neural Information Processing Systems 29*, pp. 469–477, Curran Associates, Inc., 2016.

[12] M. Chen and L. Denoyer, "Multi-view generative adversarial networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 175–188, Springer, 2017.

[13] L. R. Rabiner and B.-H. Juang, "An introduction to Hidden Markov models," *IEEE ASSP magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[14] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal Of The Royal Statistical Society, series B*, vol. 61, no. 3, pp. 611–622, 1999.

[15] N. Lawrence, "Probabilistic non-linear principal component analysis with gaussian process latent variable models," *JMLR*, vol. 6, pp. 1783–1816, Dec. 2005.

[16] J. Schmidhuber, "Learning factorial codes by predictability minimization," *Neural Computation*, vol. 4, no. 6, pp. 863–879, 1992.

[17] K. Ridgeway, "A survey of inductive biases for factorial representation-learning," *CoRR*, vol. abs/1612.05299, 2016.

[18] T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," in *Advances in Neural Information Processing Systems*, pp. 2610–2620, 2018.

[19] D. Bouchacourt, R. Tomioka, and S. Nowozin, "Multi-level variational autoencoder: Learning disentangled representations from grouped observations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[20] L. Tran, X. Yin, and X. Liu, "Disentangled representation learning GAN for pose-invariant face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1415–1424, 2017.

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

[22] A. Alemi, I. Fischer, J. Dillon, and K. Murphy, "Deep variational information bottleneck," in *ICLR*, 2017.

[23] J. A. K. Suykens, "Deep Restricted Kernel Machines using Conjugate Feature Duality," *Neural Computation*, vol. 29, pp. 2123–2163, Aug. 2017.

[24] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. River Edge, NJ: World Scientific, Jan. 2002.

[25] L. Houthuys and J. A. K. Suykens, "Tensor learning in multi-view kernel PCA ," in *27th International Conference on Artificial Neural Networks ICANN, Rhodes, Greece*, vol. 11140, pp. 205–215, 2018.

[26] J. Schreurs and J. A. K. Suykens, "Generative Kernel PCA," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* , pp. 129–134, 2018.

[27] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. II–97–104 Vol.2, 2004.

[28] H. Larochelle and Y. Bengio, "Classification using discriminative restricted Boltzmann machines," in *Proceedings of the 25th International Conference on Machine Learning - ICML '08*, (Helsinki, Finland), pp. 536–543, ACM Press, 2008.

[29] J. Mercer, "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*, vol. 209, pp. 415–446, Jan. 1909.

[30] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[31] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "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, 1999.

[32] A. T. Bui, J.-K. Im, D. W. Apley, and G. C. Runger, "Projection-Free Kernel Principal Component Analysis for Denoising," *Neurocomputing*, 2019.

[33] J. T. Kwok and I. W.-H. Tsang, "The pre-image problem in kernel methods," *IEEE Transactions on Neural Networks*, vol. 15, pp. 1517–1525, 2003.

[34] P. Honeine and C. Richard, "Preimage Problem in Kernel-Based Machine Learning," *IEEE Signal Processing Magazine*, vol. 28, pp. 77–88, Mar. 2011.

[35] J. Weston, B. Schölkopf, and G. H. Bakir, "Learning to Find Pre-Images," in *NIPS 16*, pp. 449–456, 2004.

[36] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer New York Inc., 2001.

[37] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[38] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[39] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "Beta-VAE: Learning basic visual concepts with a constrained variational framework.," *ICLR*, vol. 2, no. 5, p. 6, 2017.

[40] J. A. K. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor, "A support vector machine formulation to PCA analysis and its kernel version," *IEEE Transactions on neural networks*, vol. 14, no. 2, pp. 447–450, 2003.

[41] C. Eastwood and C. K. I. Williams, "A framework for the quantitative evaluation of disentangled representations," in *International Conference on Learning Representations*, 2018.

[42] Y. LeCun and C. Cortes, "MNIST handwritten digit database." http://yann.lecun.com/exdb/mnist/, 2010.

[43] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," 2017.

[44] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.

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

[46] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner, "dsprites: Disentanglement testing sprites dataset." https://github.com/deepmind/dsprites-dataset/, 2017.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[48] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (USA), pp. 6629–6640, Curran Associates Inc., 2017.

[49] R. T. Rockafellar, *Conjugate Duality and Optimization*. SIAM, 1974.

A Appendix

A.1 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 $\phi_1 : \mathbb{R}^d \mapsto \mathbb{R}^{d_f}$ and $\phi_2 : \mathbb{R}^p \mapsto \mathbb{R}^{p_f}$, the Least-Squares Support Vector Machine (LS-SVM) formulation of Kernel PCA [24] for the two data sources can be written as:

$$\begin{aligned} \min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{e}_i} \ & \frac{\eta_1}{2}\operatorname{Tr}(\boldsymbol{U}^\top\boldsymbol{U}) + \frac{\eta_2}{2}\operatorname{Tr}(\boldsymbol{V}^\top\boldsymbol{V}) - \frac{1}{2}\sum_{i=1}^N \boldsymbol{e}_i^\top\boldsymbol{\Lambda}^{-1}\boldsymbol{e}_i \\ & \text{s.t. } \boldsymbol{e}_i = \boldsymbol{U}^\top\phi_1(\boldsymbol{x}_i) + \boldsymbol{V}^\top\phi_2(\boldsymbol{y}_i) \quad \forall i = 1,\ldots,N, \end{aligned} \tag{11}$$

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 [23], the error variables $\boldsymbol{e}_i$ are conjugated to latent variables $\boldsymbol{h}_i$ using:

$$\frac{1}{2}\boldsymbol{e}^\top\boldsymbol{\Lambda}^{-1}\boldsymbol{e} + \frac{1}{2}\boldsymbol{h}^\top\boldsymbol{\Lambda}\boldsymbol{h} \geq \boldsymbol{e}^\top\boldsymbol{h}, \qquad \forall \boldsymbol{e},\boldsymbol{h} \in \mathbb{R}^s \tag{12}$$

which is also known as the Fenchel-Young inequality for the case of quadratic functions [49]. By eliminating the variables $\boldsymbol{e}_i$ from Eq. 11 and using Eq. 12, we obtain the Gen-RKM training objective function:

$$\mathcal{J}_t = \sum_{i=1}^N \left(-\phi_1(\boldsymbol{x}_i)^\top\boldsymbol{U}\boldsymbol{h}_i - \phi_2(\boldsymbol{y}_i)^\top\boldsymbol{V}\boldsymbol{h}_i + \frac{1}{2}\boldsymbol{h}_i^\top\boldsymbol{\Lambda}\boldsymbol{h}_i\right) + \frac{\eta_1}{2}\operatorname{Tr}(\boldsymbol{U}^\top\boldsymbol{U}) + \frac{\eta_2}{2}\operatorname{Tr}(\boldsymbol{V}^\top\boldsymbol{V}). \tag{13}$$

A.2 Kernel PCA in the primal

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

$$\begin{aligned} \frac{1}{\eta_1}\left[\sum_{i=1}^N \phi_1(\boldsymbol{x}_i)\phi_1(\boldsymbol{x}_i)^\top\boldsymbol{U} + \sum_{i=1}^N \phi_1(\boldsymbol{x}_i)\phi_2(\boldsymbol{y}_i)^\top\boldsymbol{V}\right] &= \boldsymbol{\Lambda}\boldsymbol{U}, \\ \frac{1}{\eta_2}\left[\sum_{i=1}^N \phi_2(\boldsymbol{y}_i)\phi_1(\boldsymbol{x}_i)^\top\boldsymbol{U} + \sum_{i=1}^N \phi_2(\boldsymbol{y}_i)\phi_2(\boldsymbol{y}_i)^\top\boldsymbol{V}\right] &= \boldsymbol{\Lambda}\boldsymbol{V}. \end{aligned} \tag{14}$$

Denote $\Phi_{\boldsymbol{x}} \coloneqq [\phi_1(\boldsymbol{x}_1),\ldots,\phi_1(\boldsymbol{x}_N)]$, $\Phi_{\boldsymbol{y}} \coloneqq [\phi_2(\boldsymbol{y}_1),\ldots,\phi_2(\boldsymbol{y}_N)]$ and $\boldsymbol{\Lambda} = \operatorname{diag}\{\lambda_1,\ldots,\lambda_s\} \in \mathbb{R}^{s\times s}$ with $s \leq N$. Now, composing the above equations in matrix form, we get the following eigen-decomposition problem:

$$\begin{bmatrix} \frac{1}{\eta_1}\Phi_{\boldsymbol{x}}\Phi_{\boldsymbol{x}}^\top & \frac{1}{\eta_1}\Phi_{\boldsymbol{x}}\Phi_{\boldsymbol{y}}^\top \\ \frac{1}{\eta_2}\Phi_{\boldsymbol{y}}\Phi_{\boldsymbol{x}}^\top & \frac{1}{\eta_2}\Phi_{\boldsymbol{y}}\Phi_{\boldsymbol{y}}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix} \boldsymbol{\Lambda}. \tag{15}$$

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 Eq. 2, which simply involves matrix multiplications.

A.3 Stabilizing the objective function

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

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

$$\begin{aligned}
\mathcal{J}_t(\boldsymbol{V},\boldsymbol{U},\boldsymbol{\Lambda},\boldsymbol{H}) =& \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \boldsymbol{\Lambda}\boldsymbol{h}_i + \frac{\eta_1}{2}\operatorname{Tr}\left(\frac{1}{\eta_1^2}\sum_{i=1}^{N}\boldsymbol{h_i}\phi_1(\boldsymbol{x}_i)^\top \sum_{j=1}^{N}\phi_1(\boldsymbol{x}_j)\boldsymbol{h_j}^\top\right) \\
&+ \frac{\eta_2}{2}\operatorname{Tr}\left(\frac{1}{\eta_2^2}\sum_{i=1}^{N}\boldsymbol{h_i}\phi_2(\boldsymbol{y}_i)^\top \sum_{j=1}^{N}\phi_2(\boldsymbol{y}_j)\boldsymbol{h_j}^\top\right) \\
=& \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \boldsymbol{\Lambda}\boldsymbol{h}_i + \frac{\eta_1}{2}\operatorname{Tr}\left(\frac{1}{\eta_1^2}\boldsymbol{H}\boldsymbol{K}_1\boldsymbol{H}^\top\right) + \frac{\eta_2}{2}\operatorname{Tr}\left(\frac{1}{\eta_2^2}\boldsymbol{H}\boldsymbol{K}_2\boldsymbol{H}^\top\right) \\
=& \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \boldsymbol{\Lambda}\boldsymbol{h}_i + \frac{1}{2}\operatorname{Tr}\left(\boldsymbol{H}\left[\frac{1}{\eta_1}\boldsymbol{K}_1 + \frac{1}{\eta_2}\boldsymbol{K}_2\right]\boldsymbol{H}^\top\right).
\end{aligned}$$

From Eq. 3, we get:

$$\begin{aligned}
\mathcal{J}_t(\boldsymbol{V},\boldsymbol{U},\boldsymbol{\Lambda},\boldsymbol{H}) &= \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \boldsymbol{\Lambda}\boldsymbol{h}_i + \frac{1}{2}\operatorname{Tr}\left(\boldsymbol{H}\boldsymbol{H}^\top\boldsymbol{\Lambda}\right) \\
&= \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \boldsymbol{\Lambda}\boldsymbol{h}_i + \frac{1}{2}\sum_{i=1}^{N}\boldsymbol{h}_i^\top\boldsymbol{\Lambda}\boldsymbol{h}_i = 0.
\end{aligned}$$

□

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 points 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:

$$\frac{d\bar{J}}{d\boldsymbol{x}} = \left(\nabla J(\boldsymbol{x}) + cJ(\boldsymbol{x})\nabla J(\boldsymbol{x})\right) = \left(1 + cJ(\boldsymbol{x})\right)\nabla J(\boldsymbol{x}). \tag{16}$$

It is easy to see from Eq. 2 that if $\boldsymbol{x} = \boldsymbol{x}^*$, $\nabla J(\boldsymbol{x}^*) = 0$, we have that $\frac{d\bar{J}}{d\boldsymbol{x}}\Big|_{\boldsymbol{x}^*} = 0$, 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 Eq. 16 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 Eq. 1 to keep it bounded and hence is suitable for minimization with Gradient-descent methods. Without the reconstruction errors, the stabilized objective function is

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

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

$$\begin{cases}
\frac{\partial\bar{J}}{\partial\boldsymbol{V}} = (1 + c_{stab}\mathcal{J}_t)\left(-\sum_{i=1}^{N}\phi_1(\boldsymbol{x}_i)\boldsymbol{h}_i^\top + \eta_1\boldsymbol{V}\right) = 0 & \implies \boldsymbol{V} = \frac{1}{\eta_1}\sum_{i=1}^{N}\phi_1(\boldsymbol{x}_i)\boldsymbol{h}_i^\top, \\
\frac{\partial\bar{J}}{\partial\boldsymbol{U}} = (1 + c_{stab}\mathcal{J}_t)\left(-\sum_{i=1}^{N}\phi_2(\boldsymbol{y}_i)\boldsymbol{h}_i^\top + \eta_2\boldsymbol{U}\right) = 0 & \implies \boldsymbol{U} = \frac{1}{\eta_2}\sum_{i=1}^{N}\phi_2(\boldsymbol{y}_i)\boldsymbol{h}_i^\top, \\
\frac{\partial\bar{J}}{\partial\boldsymbol{h}_i} = (1 + c_{stab}\mathcal{J}_t)\left(-\boldsymbol{V}^\top\phi_1(\boldsymbol{x}_i) - \boldsymbol{U}^\top\phi_2(\boldsymbol{y}_i) + \lambda\boldsymbol{h}_i\right) = 0 & \implies \lambda\boldsymbol{h}_i = \boldsymbol{V}^\top\phi_1(\boldsymbol{x}_i) \\
& \qquad\quad + \boldsymbol{U}^\top\phi_2(\boldsymbol{y}_i),
\end{cases}$$

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

A.4 Centering of kernel matrix

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

$$\boldsymbol{K}_c = \boldsymbol{K} - N^{-1}\mathbf{1}\mathbf{1}^\top\boldsymbol{K} - N^{-1}\boldsymbol{K}\mathbf{1}\mathbf{1}^\top + N^{-2}\mathbf{1}\mathbf{1}^\top\boldsymbol{K}\mathbf{1}\mathbf{1}^\top, \tag{17}$$

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

A.5 Architecture details

See Table 3 and 4 for details on model architectures, datasets and hyperparameters used in this paper. The PyTorch library in Python was used as the programming language with a 8GB NVIDIA QUADRO P4000 GPU.

Table 3: 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** (Adam)		**Architecture** $\mathcal{X}$	$\mathcal{Y}$
MNIST	1e-3	Input	28x28x1	10 (One-hot encoding)
		Feature-map (fm)	Conv 32x4x4; Conv 64x4x4; FC 128 (Linear)	FC 15, 20 (Linear)
		Pre-image map	reverse of fm	reverse of fm
		Latent space dim.		500
Fashion -MNIST	1e-3	Input	28x28x1	10 (One-hot encoding)
		Feature-map	Conv 32x4x4; 64x4x4; FC 128 (Linear)	FC 15, 20
		Pre-image map (fm)	reverse of fm	reverse of fm
		Latent space dim.		100
CIFAR-10	1e-3	Input	32x32x3	10 (One-hot encoding)
		Feature-map (fm)	Conv 64x4x4; Conv 128x4x4; FC 128 (Linear)	FC 15, 20
		Pre-image map	reverse of fm	reverse of fm
		Latent space dim.		500
CelebA	1e-4	Input	64x64x3	-
		Feature-map (fm)	Conv 32x4x4; Conv 64x4x4; Conv 128x4x4; Conv 256x4x4 ; FC 128 (Linear)	-
		Pre-image map	reverse of fm	-
		Latent space dim.		15
Dsprites	1e-4	Input	64x64x1	-
		Feature-map (fm)	Conv 20x4x4; Conv 40x4x4; Conv 80x4x4; FC 128 (Linear)	-
		Pre-image map	reverse of fm	-
		Latent space dim.		2/10
Teapot	1e-4	Input	64x64x3	-
		Feature-map (fm)	Conv 30x4x4; Conv 60x4x4; Conv 90x4x4; FC 128 (Linear)	-
		Pre-image map	reverse of fm	-
		Latent space dim.		5/10

A.6 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 edges of Figs. 2e, 2f), the interpolated vector $\boldsymbol{h}^\star$ is given by:

$$\boldsymbol{h}^\star = (1-\alpha)(1-\gamma)\boldsymbol{h}_1 + \alpha(1-\gamma)\boldsymbol{h}_2 + \gamma(1-\alpha)\boldsymbol{h}_3 + \gamma\alpha\boldsymbol{h}_4, \quad 0 \leq \alpha, \gamma \leq 1.$$

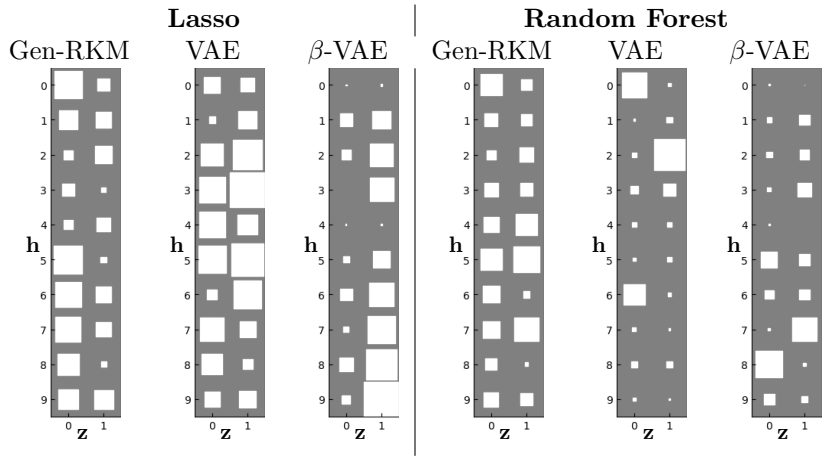
Table 4: Datasets and hyperparameters used for the experiments. The bandwidth of the Gaussian kernel for generation corresponds to the bandwidth that gave the best performance determined by cross-validation on the MNIST classification problem.

Dataset	N	d	N_{subset}	s	m	σ	n_r	l
MNIST	60000	28×28	5000	500	50	1.3	4	10
Fashion-MNIST	60000	28×28	500	100	5	/	/	10
CIFAR-10	60000	$32 \times 32 \times 3$	500	500	5	/	/	10
CelebA	202599	$128 \times 128 \times 3$	500	15	5	/	/	20
Dsprites	737280	64×64	1024	2/10	5	/	/	/
Teapot	200000	$64 \times 64 \times 3$	1000	5/10	100	/	/	/

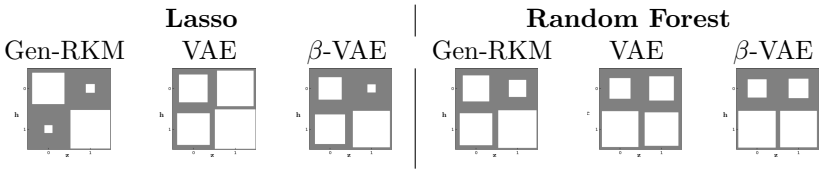
This $\boldsymbol{h}^\star$ is then used in step 8 of the generation procedure of Gen-RKM algorithm (see Algorithm 1) to compute $\boldsymbol{x}^\star$.

A.7 Visualizing the disentanglement metric

In this section we show the Hinton plots to visualize the disentaglement scores as shown in Table 2. Following the conventions of [41], $\boldsymbol{z}$ represents the ground-truth data generating factors. Figs. 6 & 7 shows 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 h_i in predicting z_i.

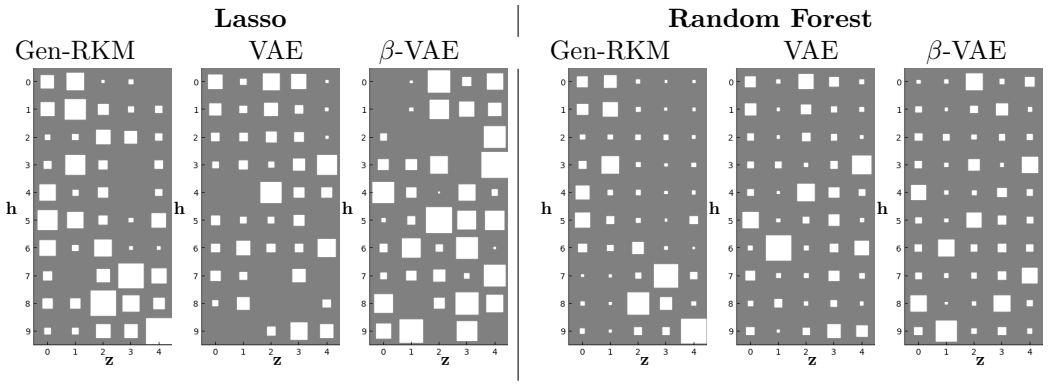


(a) $h_{dim} = 10$

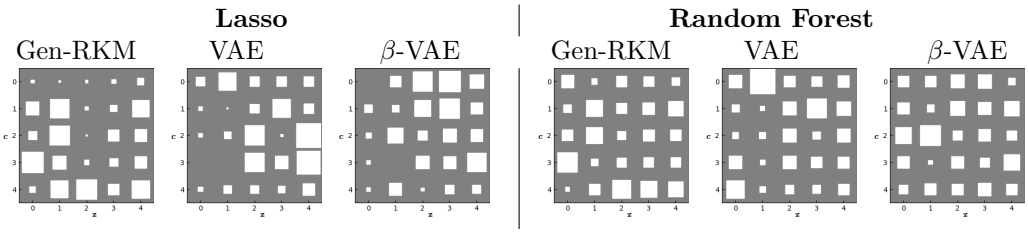


(b) $h_{dim} = 2$

Figure 6: Relative importance matrix as computed by Lasso and Random Forest regaressors 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) $h_{dim} = 5$

Figure 7: Relative importance matrix as computed by Lasso and Random Forest regaressors 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.

A.8 Further empirical results

Figure 8: 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.