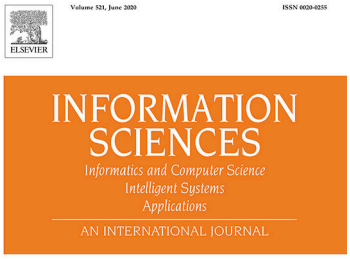


Distance assessment and analysis of high-dimensional samples using variational autoencoders

Marco Inacio, Rafael Izbicki, Bálint Gyires-Tóth

PII: S0020-0255(20)30653-8
DOI: https://doi.org/10.1016/j.ins.2020.06.065
Reference: INS 15628

To appear in: *Information Sciences*

Received Date: 6 December 2019
Revised Date: 1 May 2020
Accepted Date: 29 June 2020

Please cite this article as: M. Inacio, R. Izbicki, B. Gyires-Tóth, Distance assessment and analysis of high-dimensional samples using variational autoencoders, *Information Sciences* (2020), doi: https://doi.org/10.1016/j.ins.2020.06.065

Distance assessment and analysis of high-dimensional samples using variational autoencoders

Marco Inacio[1,2,3], Rafael Izbicki[3], Bálint Gyires-Tóth[4]

Abstract

An important question in many machine learning applications is whether two samples arise from the same generating distribution. Although an old topic in Statistics, simple accept/reject decisions given by most hypothesis tests are often not enough: it is well known that the rejection of the null hypothesis does not imply that differences between the two groups are meaningful from a practical perspective. In this work, we present a novel nonparametric approach to visually assess the dissimilarity between the datasets that goes beyond two-sample testing. The key idea of our approach is to measure the distance between two (possibly) high-dimensional datasets using variational autoencoders. We also show how this framework can be used to create a formal statistical test to test the hypothesis that both samples arise from the same distribution. We evaluate both the distance measurement and hypothesis testing approaches on simulated and real world datasets. The results show that our approach is useful for data exploration (as it, for instance, allows for quantification of the discrepancy/separability between categories of images), which can be particularly helpful in early phases of the a machine learning pipeline.

Keywords: variational autoenconders; two-sample comparison; high-dimensional data; hypothesis testing

[1]Corresponding author: m@marcoinacio.com.
[2]University of São Paulo.
[3]Federal University of São Carlos.
[4]Budapest University of Technology and Economics.

1. Introduction

An important question in many applications of machine learning and Statistics is whether two samples (or datasets) arise from the same data generating probability distribution [**gretton2012kernel**, **holmes2015two**, **soriano2015bayesian**]. Although an old topic in statistics [**Mann47**, **Smirnov48**], simple accept/reject decisions given by most hypothesis tests are often not enough: it is well known that the rejection of the null hypothesis does not mean that the difference between the two groups is meaningful from a practical perspective [**1904.06605**, **Wasserstein2019**]. Thus, tests that go beyond accept/reject decisions are preferred in practice. In particular, tests that provide not only single and interpretable numerical values, but also a visual way of exploring how far apart the datasets are from each other especially useful. This raises the question of how to assess the distance between two groups meaningfully, which is especially challenging in high-dimensional spaces.

In this work, we present a novel nonparametric approach to assess the dissimilarity between two high-dimensional datasets using variational autoencoders (VAE) [**vae**]. We show how our approach can be used to visually assess how far apart datasets are from each other via a boxplot of their distances and additionally, provide a way of interpreting the scale of these distances by using the distance between known distributions as a baseline. We also show how a formal permutation-based hypothesis testing can be derived within our framework.

The remaining of this paper is organized as follows. In sections 1.1 and 1.2, we present a brief description of work related to our proposed method. In Section 2, we present a review of variational inference and VAE, and show that the latter can be interpreted as a density estimation procedure, which is the basis of the proposed method. In Section 3, we show how variational autoencoders can be used as a method of exploring the differences between two samples. In Section 4, we use our method to derive a formal hypothesis testing procedure. Both sections also show applications of the methods to simulated and real-world datasets. Finally, Section 5 concludes the paper with final remarks. Appendix 5

contains details on the configurations of the software and neural networks used, as well as a link to our implementation, which is published open source.

1.1. Related work on two-sample hypothesis testing

Several nonparametric two-sample testing methods have been proposed in the literature; they date back to **Mann47**, **Smirnov48**, **WELCH1947**: three classical two-sample tests (Mann-Whitney rank test, Kolmogorov-Smirnov and Welch's t-test, respectively) which were designed to work for univariate random variables only. On the other hand, **holmes2015two**, **soriano2015bayesian**, **ceregatti2018wiks** investigate Bayesian univariate methods for this task.

More recently, **gretton2012kernel** introduce a two-sample test comparison using reproducing kernel Hilbert space theory that works for high-dimensional data. The test, however, does not provide a way of to visually assess the dissimilarity between the datasets. **two-sample-deep-learning** proposes a method for two-sample hypothesis testing utilizing deep learning, which contrary to a permutation based test, only controls the type-1 error rate asymptotically; **binary-two-sample** proposes a test statistic built using binary classifier in the context of causal inference and causal discovery, also relaying on asymptotic distribution for the test statistic (the distance between the performance of binary classifiers) under the null hypothesis.

Other two-sample tests for high-dimensional data can be found in [**mondal2015high**, **NIPS2016_6209**] and references therein. Although these tests are robust and effective in many settings, they do not provide a visual analysis to assess the distance between the groups. Thus, they do not provide ways of checking if the difference between the datasets is meaningful from a practical perspective, a gap in literature which is filled by this article.

1.2. Related work on two-sample comparison and distance measurement

There has also been some work devoted to two-sample comparison and related tasks: In particular **deAlmeidaIncio2018**, provides a framework for assessing the distance between populations using density estimation methods.

However, the method provided in that work is based on MCMC (Markov Chain Monte Carlo) Bayesian simulations, and therefore it is unable to scale to large datasets (see **betancourt_mcmc_subsampling**, for instance) and high-dimensional spaces. In this work, we overcome these issues by using variational autoencoders to estimate densities, and by introducing a specific metric which has an analytic solution even in high-dimensional spaces).

pmlr-v97-kornblith19a (and references therein) proposes a new method of comparison of neural networks representation. **pmlr-v48-larsen16** propose a variant of variational autoencoders (VAE) that better measure similarities in data space than a vanilla VAE. **an2015variational** uses VAEs for anomaly detection: that is, with the goal of identifying whether a single instance is different from an observed sample. **1280752** evaluates existing similarity measurement methods in the context of image retrieval. These papers however do not use their methods for performing formal hypothesis tests.

Finally, for closely-related problems and applications, see also **pfister2016kernel**, **ramdas2017wasserstein**, **1908.00105** for methods on how to solve the problem of independence testing and **desmistifying-gans** which uses two-sample tests as a tool to evaluate generative adversarial networks.

2. Variational Autoencoders

In this section, we review key aspects of the variational autoencoders framework [**vae**] which are important to our proposed method.

Variable Autoencoders are among the most famous deep neural network architectures. The generative behaviour of VAEs makes these model attractive for many application scenarios. VAEs are often used in computer vision related tasks. Introducing labeled data to the VAE training, attribute vectors, such as smile vector [**yan2016attribute2image**], can be computed; i.e. in **yan2016attribute2image** the smile vector is computed by subtracting the mean latent vector for images of smiling and non-smiling people. In the generation phase, this vector can be altered in the latent space to generate faces

with different smiling attributes. Another work utilizes VAEs to predict the possible movement of objects on images, pixelwise [**walker2016uncertain**]. Videos were also generated from text by combining VAEs with Generative Adversarial Network (GANS) [**li2018video**]. VAEs are also successfully applied in speech technologies. **hsu2017learning** learns latent representations from unlabelled data with VAEs for speech transformation (including phonetic content and speaker identity). In text-to-speech synthesis systems VAEs can be successfully applied for learning attributes and thus, controllable, expressive speech can be generated [**akuzawa2018expressive**]. Other types of sequences, like text, can also be modeled with VAEs. **semeniuta2017hybrid** uses convolutional encoder and deconvolutional decoder components, augmented with a recurrent language model in a variational autoencoder architecture to model text. Furthermore, there have been numerous theoretical research that focuses on or utilizes VAEs, like for second-order gradient estimation [**fan2015fast**], for importance weighting [**burda2015importance**], for anomaly detection [**suh2016echo**] and for novel architectures, like ladder VAE [**sonderby2016ladder**].

2.1. Statistical definition

Consider an i.i.d. random sample $D = (X_1, X_2, ..., X_n)$. Variational autoencoders estimate the density of this sample by encoding the information of each X_i using latent random variables $Z = (Z_1, Z_2, ..., Z_n)$, which are linked to $(X_1, X_2, ..., X_n)$ by a parameter θ. More precisely, the model assumes the structure

$$P_\theta(D = d|Z) = \prod_{i=1}^{n} \mathcal{N}(X_i = x_i; (\mu_i, \sigma_i) = g_\theta(Z_i)),$$

where $Z_i \sim \mathcal{N}(0, 1)$, g_θ is a complex function (which is the output of a neural network) with parameter θ (i.e.: the parameters/weights of a neural network), and μ_i and σ_i are the mean and standard deviation of the Gaussian distribution. Inference on such model is performed by maximizing the evidence $P(D = d; \theta) := P_\theta(D = d)$.

Note that, if g_θ is complex enough, we can actually model any distribution of X_i [**devroye1986sample**]. This is why g_θ is parametrized using an artificial neural network; it leads to flexibility (because of the richness of the space of functions they can represent, **Hornik1989**) as well as scalability.

Unfortunately, maximization of the evidence cannot be directly solved due to the curse of dimensionality [**tutorial_vae**]. The next section shows how variational inference can be used to overcome this.

2.2. Variational inference

The curse of dimensionality can be solved using variational inference, which consists of optimizing

$$\begin{aligned}&\log P_\theta(D=d) - \mathbf{D}_{KL}(Q_\phi^{(Z|D=d)}|P_\theta^{(Z|D=d)})\\&= E_{Q_\phi}\left[\log P_\theta(D|Z)|\, D=d\right]\\&- \mathbf{D}_{KL}(Q_\phi^{(Z|D=d)}|P^{(Z)})\end{aligned}$$

where $\mathbf{D}_{KL}$ refers to the Kullback-Leibler divergence and Q given by:

$$Q_\phi(Z_i|X_i=x_i) = \mathcal{N}(Z_i;(\mu_i,\sigma_i)=h_\phi(x_i))$$

This framework is called variational autoenconder and it solves the curse of dimensionality [**vae**]. The training procedure for variational autoencoders is presented in Figure 1; for more details, see **vae**.

2.3. Generative model

The trained model can be used to generate new instances $\widetilde{X}_j$: this can be done by applying $P_{\hat{\theta}}(\widetilde{X}_j) = \int P_{\hat{\theta}}(\widetilde{X}_j|\widetilde{Z_j}=z)P^{(\widetilde{Z}_j)}(\mathrm{d}z)$, i.e.: sample $Z \sim N(0,1)$, apply it on the neural network g_θ and then sample from a $\mathcal{N}((\mu,\sigma) = g_\theta(z))$. Therefore, variational autoencoders are a Gaussian[5] mixture model (of

[5]Note that variational autoencoders setup can be used with distributions other then Gaussians; e.g.: discrete data with Bernoulli distribution.

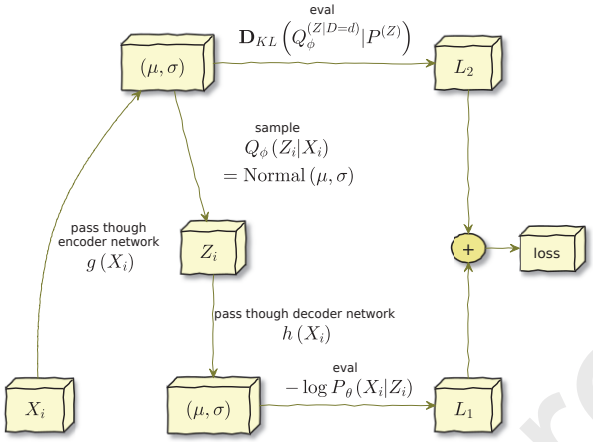


Figure 1: *VAE training procedure.*

an infinite number of Gaussians), and thus a density estimator.

2.4. Identifiability of the mixture of Gaussians

As per the structure of variational autoencoders, the distribution of such mixture of Gausssians is not identifiable [**teicher1961identifiability**, **wechsler2013bayesian**]: two different configurations of the parameters, say θ_1 and θ_2, can lead to the exact same distribution of (μ, σ). That is, $g_{\theta_1}(Z) \sim g_{\theta_2}(Z)$ even if $\theta_1 \neq \theta_2$.

In other words: if we train a variational autoencoders framework on a dataset, we will get a "generator" of pairs (μ, σ), say m_1; and if we train a variational autoencoders framework with identical structure on the same dataset, we will get another "generator" of pairs (μ, σ), say m_2. Generators m_1 and m_2 do not necessarily give the same distribution over samples of pairs (μ, σ). Nonetheless, the induced final density (i.e., the Gaussian mixture) should be same analytically (i.e.: ignoring the stochastic variation that estimation methods induce).

3. Two sample comparison: definition of the distance

We name our approach for assessing the similarity between two datasets, D_1 and D_2 as *vaecompare* and describe it as follows. First, we train two varia-

tional autoencoders: one for D_1 and one for D_2. Let g_{θ_1} and g_{θ_2} be the learned functions for each of the autoencoders. g_{θ_1} and g_{θ_2}, together with $Z \sim N(0,1)$, induce two distributions over the parameter space (μ, σ). Let $S_1 = (\mu_1, \sigma_1)$ and $S_2 = (\mu_2, \sigma_2)$ be two samples generated from the enconders g_{θ_1} and g_{θ_2}, respectively. We then measure the distance between S_1 and S_2. Now, recall that each (μ, σ) is used to generate a new sample $X \sim N(\mu, \sigma)$ (Section 2.3). Thus, a meaningful distance between S_1 and S_2 should be in the space of the random variables they generate. The key idea to make the method computationally feasible is to use a symmetric Kullback-Leibler divergence between the distributions induced by S_1 and S_2:

$$\mathbb{D}(S_1, S_2) := \frac{\mathbf{D}_{KL}(P_{S_1}, P_{S_2}) + \mathbf{D}_{KL}(P_{S_2}, P_{S_1})}{2d},$$

where d is the dimension of the feature space, P_{S_i} is a (multivariate) Gaussian distribution with parameters (μ_i, σ_i), and $\mathbf{D}_{KL}$ is the Kullback-Leibler divergence. $\mathbf{D}_{KL}$ has an analytical solution in the Gaussian case:

$$\begin{aligned}
&\mathbf{D}_{KL}(\mathcal{N}(\mu_1, \sigma_1^T I), \mathcal{N}(\mu_2, \sigma_2^T I)) = \\
&\frac{1}{2}\Bigg[2\left(\sum_{i=1}^{d} \log \sigma_{2,i} - \log \sigma_{1,i}\right) - d \\
&+ \left(\sum_{i=1}^{d} \sigma_{1,i}^2 / \sigma_{2,i}^2\right) + \left(\sum_{i=1}^{d} \sigma_{2,i}^2 (\mu_{2,i} - \mu_{1,i})^2\right)\Bigg].
\end{aligned}$$

In case X represents an image, we use the standard approach of using multi-dimensional Bernoulli distributions with dimensions independent from each other (see [**vae**, **tutorial_vae**], for instance). In this case, the Kullback-

Leibler can also be obtained analytically:

$$\begin{aligned}
&\mathbf{D}_{KL}(\text{Bernoulli}(p), \text{Bernoulli}(q)) \\
&+ \mathbf{D}_{KL}(\text{Bernoulli}(q), \text{Bernoulli}(p)) \\
&= \sum_{i=1}^{d} (q_i - p_i)(\log(q_i) - \log(p_i) + \log(1 - p_i) \\
&- \log(1 - q_i))
\end{aligned}$$

Using this approach, we can therefore assess the distance between one sample generated from the first autoencoder and a sample generated from the second autoencoder. In order to assess the divergence between the datasets D_1 and D_2, we can repeat this procedure several times; this will give a sample of the distribution of distances.

Now, in order to overcome the identifiability issue discussed in Section 2.4, we train the variational autoencoders multiple times (we call these "refits") for each dataset (using distinct initialization seeds for the network parameters) and use the new instances pairs (μ, σ) from each of them in equal proportion. The full procedure is summarized in Algorithm 1 and Figure 2.

Note that from the perspective of applying this method to images, it can also be interpreted as a data exploration tool, as it helps exploring the separability and uncertainty of classes of images and the relation between their data generating processes.

3.1. Assessing the magnitude of the distance

In Section 3, we defined a method to measure the distance between two datasets. A yardstick is still required in order to say what is a "low" and "high" distance. In order to create a baseline to interpret such distances, we proceed in similar fashion as **deAlmeidaIncio2018**: we compute the distance between two known distributions.

In the case of Gaussian VAEs we can work for instance with $\mathbb{D}(\mathcal{N}_0, \mathcal{N}_1)$, where $\mathcal{N}_0$ is a multivariate Gaussian with covariance given by an identity matrix

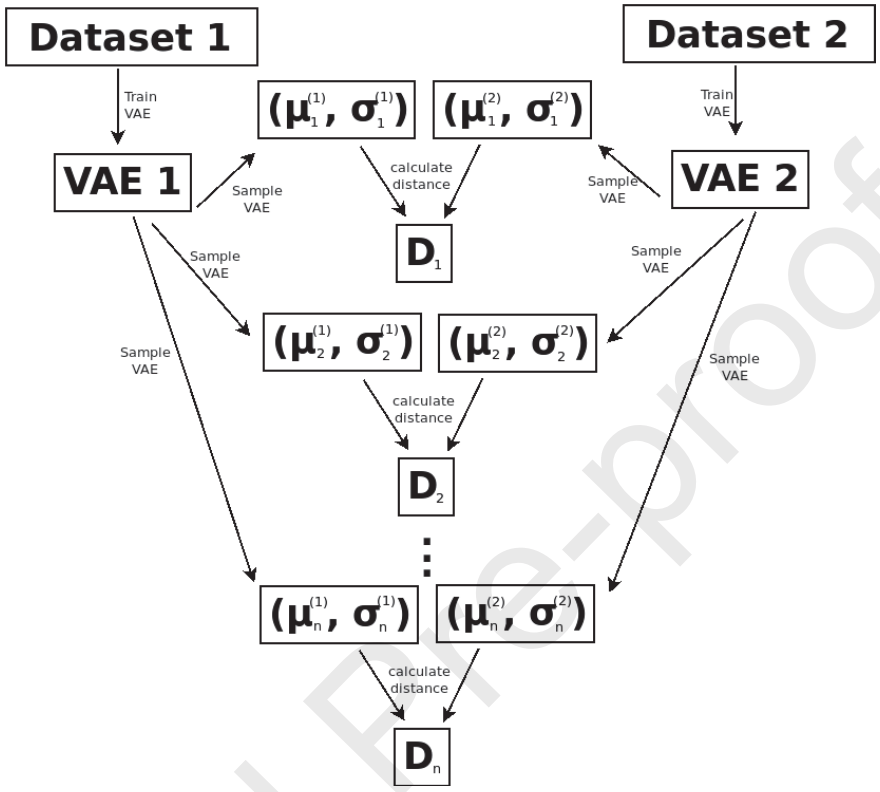


Figure 2: *Schematic representation of the procedure to generate divergence samples for data comparison.*

Algorithm 1 Generating divergence samples using *vaecompare*

Input: dataset D_1, dataset D_2, number of desired samples per refit n, number of desired refits R

Output: divergence samples S.

for $i \in \{1, \ldots, R\}$ **do**
    Train VAE V_1 from D_1
    Train VAE V_2 from D_2
    for $j \in \{1, \ldots, n\}$ **do**
        Generate a sample s_1 from V_1 (e.g.: a pair (μ, σ) for Gaussian VAE).
        Generate a sample s_2 from V_2.
        Calculate $\mathbb{D}(s_1, s_2)$ and store it on S.
    end for
end for

and mean given by a vector of zeros and $\mathcal{N}_1$ is a multivariate Gaussian with covariance given by an identity matrix and mean given by a vector of ones. We have that $\mathbb{D}(\mathcal{N}_0, \mathcal{N}_1) = 1/2$. For binomial VAEs, we use known binomial distributions as the baseline.

3.2. Evaluation (images)

Next, we apply the our method to CIFAR10 data [**cifar10**] using the VAE as a generator of binomial distributions. The dataset consists of images from 10 distinct categories (ranging from 0 to 9), with each category containing 5000 images. To make the comparison fair when comparing a category to itself and when comparing a category to another, we chose to work with half of each category dataset (2500 images) to train each VAE; i.e.: when comparing category 0 to category 1, one VAE is trained with 2500 images from category 0 and the other is trained with 2500 images from category 1; on the other hand, when comparing category 0 to itself, each VAE is trained with half (2500 images) of the category 0 dataset. We worked with 90 VAE refits for each dataset.

In Figure 3, we present the results of such experiment with boxplots of the obtained divergences for all possible category combinations (note that the lower image of each plot is a zoomed-in version of the upper image). The figure also shows the median and mean for each category, as well as the divergence of known Bernoulli distributions (plotted as horizontal lines).

Except for categories 0, 2 and 4, the divergence samples were all concentrated near zero when comparing a category to itself (a desirable behaviour). On the other hand, for these 3 categories, we can observe a considerable amount of divergence samples spread far from zero indicating some uncertainty, but even in this case, there was a considerable amount of them near zero. Note also that the median for these three categories is much closer to zero than the mean, indicating its resilience to outliers.

On the other hand, when making comparisons between distinct categories, there are cases with high uncertainty (i.e.: boxplots with wide extensions; generally with few points close to zero), as well as cases with higher certainty (i.e.:

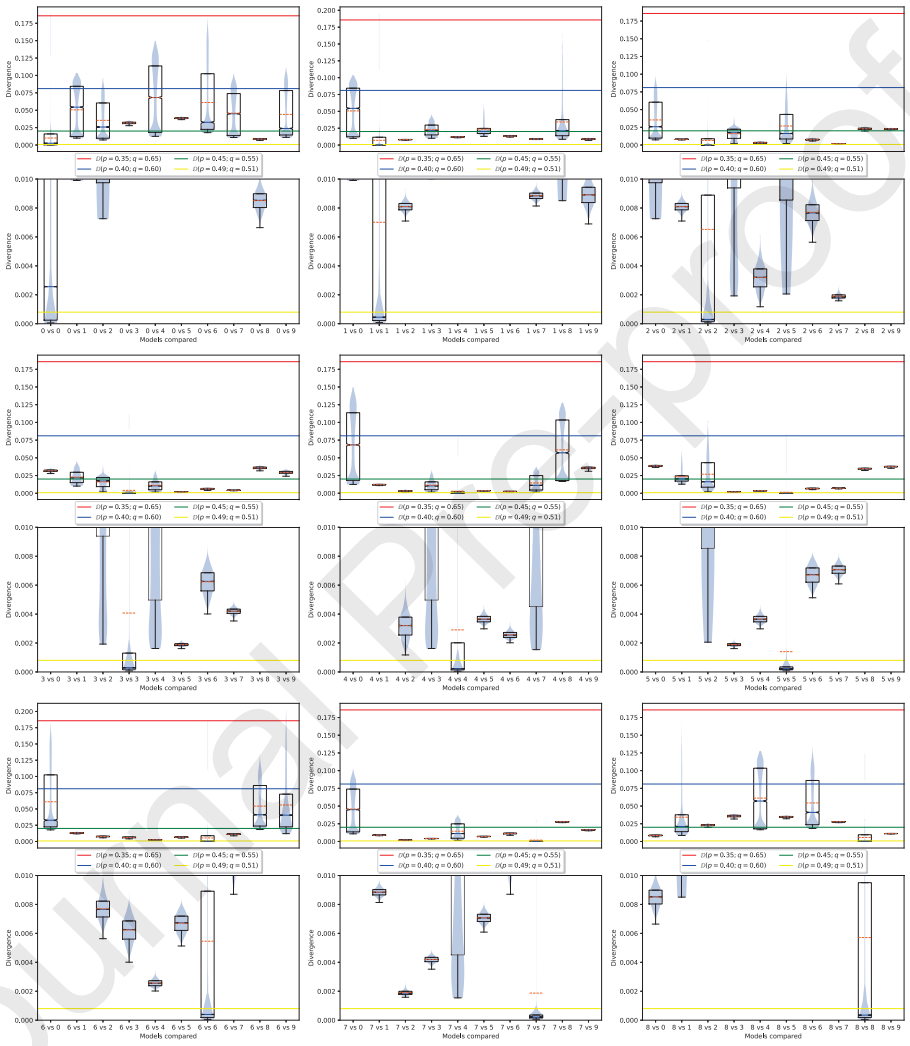


Figure 3: *Box plots of samples from our divergences comparing categories 0 to 8 to all categories. Note that the lower image of each plot is a zoomed-in version of the upper image.*

boxplots with narrow extensions).

We conclude that the method is therefore useful for the purpose of data exploration as it works as expected in a complex space such as images.

4. Hypothesis testing

We can additionally use *vaecompare* to directly test if two samples come from the same population. One way to do this is to find a threshold value (*cutpoint*) from a decision theoretic stand point where we would reject the null hypothesis of the two samples coming from the same population. This is what is done in [**ceregatti2018wiks**] in the case of a Dirichlet process prior, where the threshold is chosen so as to control type I error of the hypothesis test. Unfortunately, this is not possible in general and in general the cutpoint depends on the true data generating function.

Given that, we work instead with a simple permutation test where the datasets are repeatedly permuted against each other (i.e.: their data is mixed), and the average divergence of the samples is used as a test statistic. The p-value is then given by the quantile of the non-permuted dataset among all the statistics[6]. We note that for the hypothesis test to work in the sense of being a proper test (uniform under the null), it is not necessary to do VAE refits, but refits increase the test power as we shall see next. We present the procedure in Algorithm 2.

4.1. Evaluation (simulated data)

In this section, we apply the proposed hypothesis testing method to simulated datasets from a known data generating function and plot the observed p-value distribution. The data generating function for the datasets is defined

[6]For instance, if 43 of the permuted datasets had resulted on a lower divergence statistic than that of non-permuted dataset and on the other hand 57 had resulted on greater divergence, then the p-value would be $43/(43+57) = 0.43$.

Algorithm 2 Obtaning the p-value for hypothesis testing using *vaecompare*

Input: dataset D_1, dataset D_2, number of desired samples per refit n, number of desired refits R, number of permutations t, averaging function M (e.g. mean or median)

Output: p-value ρ.

1: **for** $i \in \{1, \ldots, t\}$ **do**
2: Run Algorithm 1, and store the results in S_i.
3: Calculate $M(S_i)$ and store the result in K_i.
4: Permute the instances of datasets D_1 and D_2.
5: **end for**
6: Obtain the number of points q_1 in $\{K_2, K_3, ..., K_t\}$ which are greater than K_1.
7: Obtain the number of points q_2 in $\{K_2, K_3, ..., K_t\}$ which are greater than or equal to K_1.
8: Set $q = (q_1 + q_2)/2$
9: Store $(q + 1)/(t + 1)$ in ρ.

as:

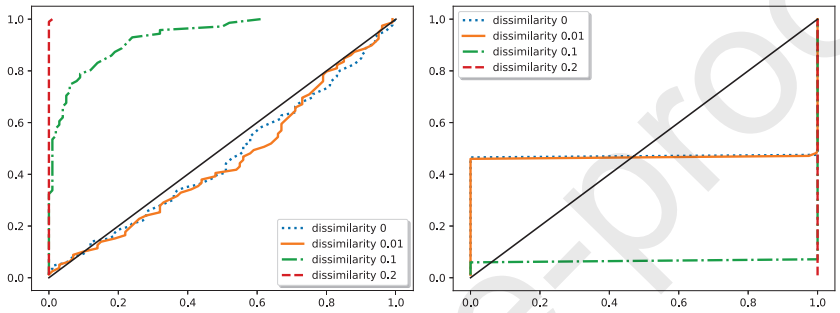
$$\begin{aligned} &\mathrm{lgr}(\mu = log(2), \sigma = \alpha) - \mathrm{lgr}(\mu = log(2), \sigma = 0.5) \\ &+ \mathrm{gr}(\mu = 1, \sigma = 2) + k \end{aligned}$$

where lgr stands for multivariate log Gaussian random number generator, and gr stands for multivariate Gaussian random number generator, both with diagonal covariance matrices. Moreover, $\alpha = \{0.2 + 0.7 * i/9\}_{i=0}^{i=9}$, i.e.: $\alpha_i = 0.2 + 0.7 * i/9$ for $i \in \{0, \mathbf{1}, ..., 9\}$.

For simplicity, we do not use refits here. The value of the vector k is fixed in zero for one of the datasets, and varied for the other. This is done in order to change the dissimilarity between the samples (i.e.: the larger k is, the more dissimilar the sample distributions are) and from that, observe the behaviour of the distribution of the p-value.

In Figure 4a, we present the results of such experiment using the permutation test: the empirical cumulative distribution of the p-values; while in Figure 4b, we do the same simulation study using an Gaussian asymptotic approximate to the permutation test.

The permutation test fulfilled the required properties of a frequentist hypothesis test, as it has approximately (sub)uniform distribution under the null hypothesis (as expected) and the test power increases as the divergence increases. Notice that, for simplicity, we do not use refits here; if we do, we expect the power to increase as is the case in the next section. The asymptotic test, on the other hand, performed poorly.



(a) *Permutation.* **(b)** *Asymptotic.*

Figure 4: *Empirical cumulative distribution function of the p-values for distinct dissimilarity values (when the dissimilarity is zero, the null hypothesis is true) using a permutation test and asymptotic (approximate to permutation test).*

4.2. Evaluation (images)

Here, we also applied the hypothesis testing method to the CIFAR10 dataset, using the same 2500 images for each category as described in 3.2. In Tables 1, 2, 3 and 4 we present the p-values obtained in the test while in Tables 5, 6, 7 and 8 we present the combinations that gave the correct results and type 2 error for a significance level of 5%. We applied the tests both without VAE refits and with 5 refits; we also tried the median as an alternative to the mean with the intuition that this might help remove the weight of outlier distance points.

In Table 9, we present a summary of the results. The method performed well under the null for both the mean and median metrics. Moreover, the method has shown to have a significant increase in test power when used with VAE refits.

In case of metrics performance comparison, it can be seem that the mean had incurred in less type I errors while the median incurred in larger but an admissible number given the critical rate of 5%. On the other hand the performance of median metric was considerably better regarding type II errors, this might be related to its robustness to outliers which have shown to be a frequent problem in the Figure 3.

Table 1: *P-values for hypothesis testing for each category* ***without*** *refits and averaging using the* ***median****.*

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	0.99	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.40	0.01
c1	-	0.42	0.56	0.01	0.01	0.01	0.05	0.24	0.03	0.37
c2	-	-	0.74	0.50	0.40	0.04	0.58	0.04	0.01	0.01
c3	-	-	-	0.78	0.31	0.26	0.49	0.22	0.01	0.01
c4	-	-	-	-	0.39	0.21	0.29	0.23	0.01	0.01
c5	-	-	-	-	-	0.02	0.01	0.01	0.01	0.01
c6	-	-	-	-	-	-	0.96	0.32	0.01	0.01
c7	-	-	-	-	-	-	-	0.78	0.01	0.01
c8	-	-	-	-	-	-	-	-	0.06	0.01
c9	-	-	-	-	-	-	-	-	-	0.54

Table 2: *P-values for hypothesis testing for each category* ***with*** *refits and averaging using the* ***median****.*

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	0.20	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01
c1	-	0.26	0.03	0.01	0.01	0.01	0.01	0.01	0.01	0.02
c2	-	-	0.36	0.20	0.05	0.04	0.07	0.01	0.01	0.01
c3	-	-	-	0.41	0.06	0.14	0.25	0.03	0.01	0.01
c4	-	-	-	-	0.90	0.02	0.01	0.02	0.01	0.01
c5	-	-	-	-	-	0.94	0.02	0.01	0.01	0.01
c6	-	-	-	-	-	-	0.02	0.01	0.01	0.01
c7	-	-	-	-	-	-	-	1.00	0.01	0.01
c8	-	-	-	-	-	-	-	-	0.36	0.01
c9	-	-	-	-	-	-	-	-	-	0.03

Table 3: *P-values for hypothesis testing for each category* ***without*** *refits and averaging using the* ***mean****.*

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	0.25	0.12	0.01	0.01	0.01	0.01	0.01	0.01	0.48	0.05
c1	-	0.82	0.50	0.23	0.03	0.05	0.07	0.15	0.01	0.05
c2	-	-	0.19	0.49	0.48	0.44	0.09	0.02	0.10	0.01
c3	-	-	-	0.22	0.11	0.36	0.15	0.27	0.01	0.01
c4	-	-	-	-	0.87	0.20	0.40	0.01	0.01	0.01
c5	-	-	-	-	-	0.19	0.05	0.23	0.01	0.01
c6	-	-	-	-	-	-	0.73	0.01	0.01	0.01
c7	-	-	-	-	-	-	-	0.48	0.01	0.02
c8	-	-	-	-	-	-	-	-	0.45	0.06
c9	-	-	-	-	-	-	-	-	-	0.48

Table 4: *P-values for hypothesis testing for each category* ***with*** *refits and averaging using the* ***mean****.*

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	0.16	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.01
c1	-	0.30	0.07	0.01	0.01	0.01	0.01	0.01	0.01	0.02
c2	-	-	0.10	0.06	0.12	0.37	0.04	0.01	0.01	0.01
c3	-	-	-	0.83	0.08	0.45	0.15	0.02	0.01	0.01
c4	-	-	-	-	0.53	0.05	0.10	0.20	0.01	0.01
c5	-	-	-	-	-	0.50	0.01	0.01	0.01	0.01
c6	-	-	-	-	-	-	0.12	0.05	0.01	0.01
c7	-	-	-	-	-	-	-	0.48	0.01	0.01
c8	-	-	-	-	-	-	-	-	0.18	0.01
c9	-	-	-	-	-	-	-	-	-	0.71

Table 5: *Results of the hypothesis testing when applying a critical rate of 5%* ***without*** *refits and averaging using the* ***median****. Here G stands for "good" and E2 for type 2 error.*

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	G	G	G	G	G	G	G	G	E2	G
c1	-	G	E2	G	G	G	G	E2	G	E2
c2	-	-	G	E2	E2	G	E2	G	G	G
c3	-	-	-	G	E2	E2	E2	E2	G	G
c4	-	-	-	-	G	E2	E2	E2	G	G
c5	-	-	-	-	-	E1	G	G	G	G
c6	-	-	-	-	-	-	G	E2	G	G
c7	-	-	-	-	-	-	-	G	G	G
c8	-	-	-	-	-	-	-	-	G	G
c9	-	-	-	-	-	-	-	-	-	G

Table 6: *Results of the hypothesis testing when applying a critical rate of 5%* ***with*** *refits and averaging using the* ***median****. Here G stands for "good" and E2 for type 2 error.*

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	G	G	G	G	G	G	G	G	G	G
c1	-	G	G	G	G	G	G	G	G	G
c2	-	-	G	E2	G	G	E2	G	G	G
c3	-	-	-	G	E2	E2	E2	G	G	G
c4	-	-	-	-	G	G	G	G	G	G
c5	-	-	-	-	-	G	G	G	G	G
c6	-	-	-	-	-	-	E1	G	G	G
c7	-	-	-	-	-	-	-	G	G	G
c8	-	-	-	-	-	-	-	-	G	G
c9	-	-	-	-	-	-	-	-	-	E1

Table 7: *Results of the hypothesis testing when applying a critical rate of 5% **without** refits and averaging using the **mean**. Here G stands for "good" and E2 for type 2 error.*

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	G	E2	G	G	G	G	G	G	E2	G
c1	-	G	E2	E2	G	G	E2	E2	G	G
c2	-	-	G	E2	E2	E2	E2	G	E2	G
c3	-	-	-	G	E2	E2	E2	E2	G	G
c4	-	-	-	-	G	E2	E2	G	G	G
c5	-	-	-	-	-	G	G	E2	G	G
c6	-	-	-	-	-	-	G	G	G	G
c7	-	-	-	-	-	-	-	G	G	G
c8	-	-	-	-	-	-	-	-	G	E2
c9	-	-	-	-	-	-	-	-	-	G

Table 8: *Results of the hypothesis testing when applying a critical rate of 5% **with** refits and averaging using the **mean**. Here G stands for "good" and E2 for type 2 error.*

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	G	G	G	G	G	G	G	G	G	G
c1	-	G	E2	G	G	G	G	G	G	G
c2	-	-	G	E2	E2	E2	G	G	G	G
c3	-	-	-	G	E2	E2	E2	G	G	G
c4	-	-	-	-	G	G	E2	E2	G	G
c5	-	-	-	-	-	G	G	G	G	G
c6	-	-	-	-	-	-	G	G	G	G
c7	-	-	-	-	-	-	-	G	G	G
c8	-	-	-	-	-	-	-	-	G	G
c9	-	-	-	-	-	-	-	-	-	G

Table 9: *Summary of the results of the hypothesis testing when applying a critical rate of 5%.*

Averaging	Refits	Number Type I errors	Number Type II errors
mean	with	0	18
mean	without	0	38
median	with	2	10
median	without	1	30

4.3. Evaluation (comparison with other methods)

Next, we apply our proposed hypothesis testing method (using the median as the averaging function and 10 refits) to simulated datasets from a known data generating function and compare it with other well established two-sample comparison methods: Mann-Whitney rank test[**Mann47**], Kolmogorov-Smirnov[**Smirnov48**] and Welch's t-test[**WELCH1947**].

Given that such methods work only with univariate datasets, we choose the true distribution of the generating data to be a mixture of 3 equiprobable Gaussian distributions with means -2, 0 and 2 and standard deviation 1. For testing the alternative hypothesis, one of the datasets had a 0.1 disturbance added (i.e., the dataset is generated from a mixture of 3 equiprobable Gaussian distributions with means -2.1, 0.1 and 2.1). Each dataset being compared is composed of $n = 1000$ instances generated independently from the true distribution. Due to computational limitations, the power function was estimated using 500 simulations for vaecompare, while we used 10000 simulations for the other tests.

Figure 5 shows the mean test power of each test with a confidence band of 2 times the standard error (i.e., an approximately 95% confidence). In order to make the visualization of the results easier, we also present a smoothed version of this figure in Figure 6 with smoothing done by simple point interpolation[7].

The figures indicate that *vaecompare* had competitive performance when compared to the other methods.[8] Additionally, it is the only method that can be used exploratory data analysis (on two-sample comparison) instead of just hypothesis testing and moreover, as shown in Section 4.2, the test power could potentially increase if an additional number of refits, which were set to be a small number because of the computational restrictions of the simulation study.

[7]Such smoothed version could also be obtained by increasing the number of permutations for each test, this has not been done due to computational constraints of such increase.

[8]Note in particular, that, contrary to the problems of classification and regression, is not possible to easily data split the dataset to choose the best hypothesis testing method before applying it the whole dataset (at least not without causing further problems such as bias multiple comparisons).

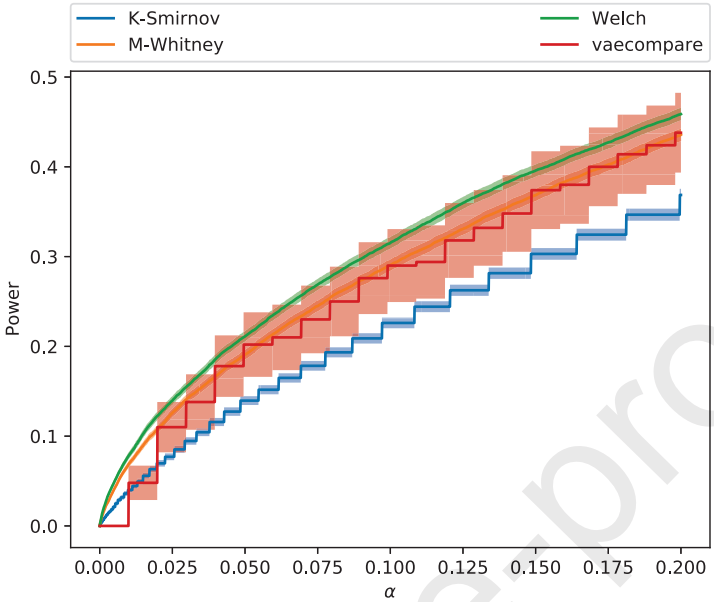


Figure 5: *Comparison of vaecompare with other hypothesis testing methods. Our procedure shows good power.*

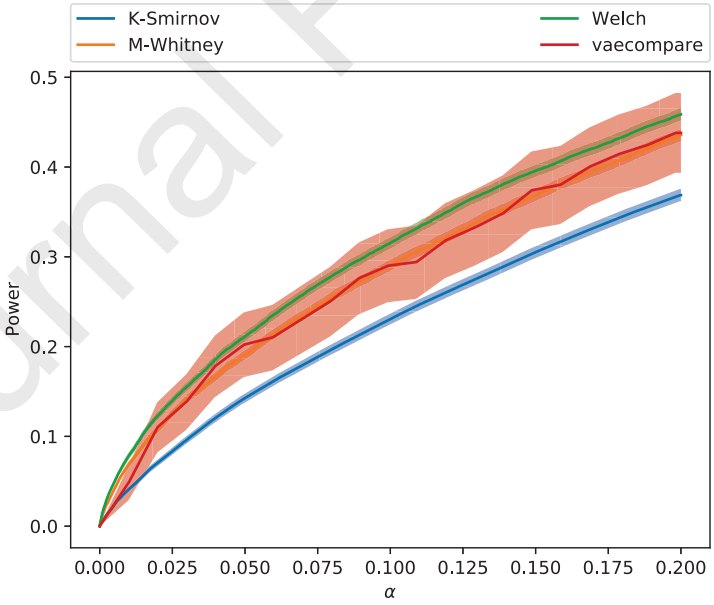


Figure 6: *Comparison of vaecompare with other hypothesis testing methods. Points outside of the grid are smoothed by interpolation. Our procedure shows good power.*

5. Discussion and Conclusions

In this work, we proposed and applied a novel method of two sample distance measurement and hypothesis testing to simulated and real-world datasets. We conclude that both two sample distance measurement and hypothesis testing were able to satisfactorily perform the intended tasks on the tested simulated and real world datasets.

The proposed methods could be used for various tasks in the machine learning pipeline, including:

- Distribution shift detection and measurement: a dataset from a experiment done in one month (e.g.: opinions of customers on a product on a specific month) might diverge in distribution from a dataset collected in another month. With our method it is possible to measure and test this diverge.
- Dataset split: to address overfitting, the data is usually split into train, validation and/or test parts. To be able to develop robust models, these parts should be similar, but should also differ enough to ensure generalization. With the proposed methods the dataset split can be done in a controlled manner, an important speed on state-of-the-art predictive methods (e.g.: see **Breiman1996StackedR**, **Coscrato2020** and references therein).
- Self-supervised clustering: based on the distance, by fine-tuning the threshold (cutpoint), binary or multi-class clustering could be performed.
- Anomaly detection: applying the proposed method to processes where anomaly may occur (e.g. malicious attack, malfunction, etc.). In this case, the distance measurement can give a direct feedback of how much the actual behaviour differs from the normal one.
- To test the quality of data generated from GANs and similar approaches (e.g.: see **binary-two-sample**).

Acknowledgments

Marco Inácio is grateful for the financial support of CAPES (this study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001). Marco Inácio is also grateful for the financial support of the Erasmus Plus programme. Rafael Izbicki is grateful for the financial support of FAPESP (grants 2017/03363-8 and 2019/11321-9) and CNPq (grant 306943/2017-4). Bálint Gyires-Tóth and Marco Inácio are grateful for the financial support of the BME-Artificial Intelligence FIKP grant of Ministry of Human Resources (BME FIKP-MI/SC). Moreover, Bálint Gyires-Tóth is also greateful for the financial support of Doctoral Research Scholarship of Ministry of Human Resources (ÚNKP-19-4-BME-189) in the scope of New National Excellence Program, by János Bolyai Research Scholarship of the Hungarian Academy of Sciences. The authors are also grateful for the suggestions given by Rafael Bassi Stern and by anonymous referees.

Appendix: Neural networks configuration, software and package

We work with a dense neural network of 10 layers with 100 neurons on each layer (totaling 195060 parameters), for both encoder and decoder networks, and the following additional specification:

- **Optimizer**: we work with the Adamax optimizer [**adam-optim**] with initial learning rate of 0.01 and decrease its learning rate by half if improvement is seen on the validation loss for a considerable number of epochs.
- **Initialization**: we used the initialization method proposed by [**nn-initialization**].
- **Layer activation**: we chose ELU [**elu**] as activation functions.
- **Stop criterion**: a 90%/10% split early stopping for small datasets and a higher split factor for larger datasets (increasing the proportion of training instances) and a patience of 50 epochs without improvement on the validation set.

- **Normalization and number of hidden layers**: batch normalization, as proposed by [**batch-normalization**], is used in this work in order to speed-up the training process.
- **Dropout**: here we also make use of dropout which as proposed by [**dropout**] (with dropout rate of 0.5).
- **Software**: we have PyTorch[**NEURIPS2019_9015**] as framework of choice which works with automatic differentiation and the sstudy Python package[**2004.14479**] for organizing the simulation studies and comparisons. Moreover, the software implementation of this work is available at https://github.com/randommm/vaecompare.

Additionally, we present in Figures 3 and 4 the algorithm to evaluate the encoder and decoder neural networks, respectively.

Algorithm 3 Algorithm to evaluate encoder network g(.) presented in Figure 1

Input: x,
Output: μ, σ.

val = x
for $i \in \{1, 10\}$ **do**
    val = linear(val) (with output size 100).
    val = ELU(val).
    val = batch_norm(val)
    val = dropout(val)
end for
μ = linear(val)
σ = exp{linear(val)}.

Algorithm 4 Algorithm to evaluate decoder network h(.) presented in Figure 1

Input: z, distribution
Output: (μ, σ) or p.

val = z
for $i \in \{1, 10\}$ **do**
    val = linear(val) (with output size 100).
    val = ELU(val).
    val = batch_norm(val)
    val = dropout(val)
end for
if distribution is "gaussian" (i.e. continuous data) **then**
    μ = linear(val)
    σ = exp{linear(val)}.
else if distribution is "bernoulli" **then**
    p = sigmoid{linear(val)}.
end if