Manifold exploration of industrial processes with Variational AutoEncoders

Brendan L'Ollivier[1], Sonia Tabti[1], and Julien Budynek[1]

[1]Fieldbox.ai, Bordeaux, France

June 8, 2021

Abstract

In this article, a computationally efficient manifold learning algorithm combining a variational autoencoder and a nearest neighbor graph is proposed. In fact, using a variational autoencoder to compute an approximation of the underlying data distribution allows our method to tackle some shortcomings of neighbor graph construction methods, namely the ability to deal with noisy and high dimensional data. This method aims to extend the range of application of graph-based manifold learning techniques to the complexity of industrial process data. Once a graph is computed, it provides a condensed representation of the behavior of the process. Also, the graph framework makes it more convenient to incorporate industrial metrics, such as product quality, through weights customization. The final graph can be used to assist the operator in selecting optimal process parameters values. The proposed approach is tested on both synthetic and real data.

Keywords: Manifold exploration, Neighbor Graph, Variational AutoEncoder, Industry.

1 Introduction

Advances in the digitization of the manufacturing industry are resulting in a considerable increase in available data, both in terms of quantity and diversity of sources all along the production cycle. And paradoxically, the high costs of data labelization, typically occurring at the output of a process, lead to a relatively small amount of labeled data. Uncovering the structure of unlabeled data, expressed as the field of representation learning [BCV14], appears to be one of the the next milestones in the application of machine learning in the industry.

Representation learning is about transforming the observable data, commonly expressed in the Euclidean framework, into new representation that captures its underlying structure better. Manifold learning follows this principle with the assumption that the intrinsic dimension of the data is potentially much lower than the dimension of the ambient space, resulting in data points lying around a manifold embedded in the real space. Learning this manifold leads to more insightful representations of the data that can potentially improve the performances of subsequent machine learning tasks, or allow data visualization.

Manifold learning techniques are numerous. Principal Components Analysis (PCA) is a popular linear algorithm that scales very well to large volumes of data. However, its nonlinear variant, Kernel PCA, scales badly to large datasets. Since the early 2000s, several algorithms dealing with nonlinearity have been proposed. For instance, Isometric mapping (or Isomap, which is the nonlinear extension of the Multidimensional Scaling algorithm), Locally Linear Embedding and Laplacian Eigenmaps can be cited. A survey on these algorithms can be found in [Cay05]. They all have in common to be based on the computation of a nearest neighbor graph on which various techniques are applied to find an optimal layout on a low dimensional space. The neighborhood of a sample is either defined by a number k of nearest neighbors or a radius ϵ. These parameters generally require careful, costly tuning when dealing with real world data. In fact, nearest neighbors graphs do not behave as expected in case of noisy data and high variations in the curvature of the manifold and the sampling density of the data. Other graph-based methods have been developed to overcome these drawbacks. They focus on two aspects of the graph construction. On the one hand, some models attempt to focus on the graph architecture at

global scale. For instance, ensemble of Minimum Spanning Trees (MST) [CPZ04] are computed from various versions of the data perturbed with isotropic Gaussian noise and combined together. Another architecture is the Partition-based Graph Abstraction (PAGA) [WHP+18] that connects groups of points at different scales. On the other hand, some graph construction models focus on a better evaluation of local connectivity. As an example, the Uniform Manifold Approximation and Projection (UMAP) algorithm [MHM20], known for its application in data visualization, defines a local Riemannian metric as a fuzzy estimation of the connectivity between points. Another example is adaptive manifold learning [ZWZ12] that dynamically adapts the local neighborhood size k by taking into account the interplay between the curvature and the local density.

Deep learning also addressed the problem of manifold learning. Instead of analyzing local connectivity in the data, it directly learns the latent factors from which the observable features derive. The architecture of autoencoders neural networks is a well known approach [Bal87], but many more elaborated models have been developed since then [BCV14]. A promising deep learning framework for manifold learning is the one given by the Variational AutoEncoder (VAE). The original version [KW14] can be seen as a regularized version of the autoencoder which is able to learn complex distributions and represent the data in a low dimensional space, usually called latent space, that tends to preserve the local structure of the data.

In this article, the proposed manifold learning algorithm is applied to manifold exploration (or traversal). It consists in exploiting the learned structure for operations such as interpolation between any two points of the data. One of the main interest of manifold exploration is the inference of meaningful intermediary points with smooth transitions between each point of the path. VAEs have already found concrete applications on real world data as a framework for manifold exploration. In biology, they showed great efficiency in learning the structure of molecules and discovering new ones with interpolation in the latent space [SMAH20].

This work is motivated by an analogous application in industry: the interpolation between distinct setpoints of a physical process, while maintaining its stability or any other operational criterion. The proposed approach combine some advantages of the two domains mentioned above: graphs are very efficient for manifold exploration and the VAEs offer a manifold learning framework that scales well to big data. Instead of tweaking the nearest neighbor graph in order to make it less sensitive to noise, the use of VAEs as a preprocessing step for the input data is explored. The VAE learns a smoothed version of the data and a low dimensional latent space. Both can be used to build a cleaner nearest neighbor graph.

After presenting the Variational AutoEncoder in section 2, we introduce the mixed approach between classical graph based algorithms and VAE in section 3. In section 4, we expose the benefits of manifold learning in an industrial context, and present the results of experiments conducted on actual data from three different processes.

2 Learning statistical representation of non linear manifolds with VAEs

In the field of machine learning, available data are usually composed of N observed variables $\boldsymbol{x} = (x_1, x_2, ..., x_N) \in \mathbb{R}^N$ without any prior knowledge on the underlying process with true distribution $p^*(\boldsymbol{x})$. This distribution is approximated with a chosen model $p_\theta(\boldsymbol{x})$, with parameters θ. This fully-observed model is then extended with $L < N$ latent variables $\boldsymbol{z} = (z_1, z_2, ..., z_L) \in \mathbb{R}^L$ from which the observed data is generated. The marginal distribution over the observed variables $p_\theta(\boldsymbol{x})$, is given by:

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} \tag{1}$$

VAEs rely on the variational inference technique for learning the approximated distribution, which has proven to be faster and easier to scale to large data compared to the other dominant technique: Markov Chain Monte Carlo (MCMC) sampling [BKM17]. Variational inference is an optimization strategy for approximating the intractable marginal likelihood in equation (1). It introduces a family of distributions $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ such that:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) \approx p_\theta(\boldsymbol{z}|\boldsymbol{x}) \tag{2}$$

The optimization objective is the minimization of the Kullback-Leibler divergence between the two distributions in (2) : $\mathbb{KL}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})\big)$, which is equivalent to maximizing the Evidence Lower Bound (ELBO) [KW14]:

$$ELBO = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathbb{KL}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})\big) \tag{3}$$

The most common choice for $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is a diagonal Gaussian distribution, the parameters of which $\boldsymbol{\mu_z} = (\mu_{z_1}, ..., \mu_{z_L})$ and $\boldsymbol{\sigma_z} = (\sigma_{z_1}, ..., \sigma_{z_L})$ are the outputs of an encoder network parametrized by ϕ:

$$\begin{aligned}(\boldsymbol{\mu_z}, \boldsymbol{\sigma_z}) &= Encoder_\phi(\boldsymbol{x}) \\ q_\phi(\boldsymbol{z}|\boldsymbol{x}) &= \mathcal{N}\big(\boldsymbol{z}; \boldsymbol{\mu_z}, \boldsymbol{\sigma_z}\big)\end{aligned} \tag{4}$$

When $\boldsymbol{x}$ is continuous, $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is also modeled by a diagonal Gaussian distribution, whose parameters $\boldsymbol{\mu_x} = (\mu_{x_1}, ..., \mu_{x_N})$ and $\boldsymbol{\sigma_x} = (\sigma_{x_1}, ..., \sigma_{x_N})$ are the outputs of a decoder network parametrized by θ:

$$\begin{aligned}(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x}) &= Decoder_\theta(\boldsymbol{z}) \\ p_\theta(\boldsymbol{x}|\boldsymbol{z}) &= \mathcal{N}\big(\boldsymbol{x}; \boldsymbol{\mu_x}, \boldsymbol{\sigma_x}\big)\end{aligned} \tag{5}$$

And the prior $p_\theta(\boldsymbol{z})$ is set to the standard normal distribution $\mathcal{N}(0, 1)$.

Under these assumptions, the two terms of the ELBO in equation (3) can be computed analytically. The reconstruction term $\log p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is the likelihood function of the Gaussian distribution $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu_x}, \boldsymbol{\sigma_x})$.

$$\log p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \sum_{n=1}^{N} \left(-\frac{1}{2}\log(2\pi\sigma_{x_n}^2) - \frac{1}{2\sigma_{x_n}^2}(x_n - \mu_{x_n})^2 \right) \tag{6}$$

And the regularization term is the KL divergence between two normal distribution $\mathcal{N}(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x})$ and $\mathcal{N}(0, 1)$

$$\mathbb{KL}\big(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big) = \sum_{l=1}^{L} -\frac{1}{2}\left(1 + \log\sigma_{z_l}^2 - \sigma_{z_l}^2 - \mu_{z_l}^2\right) \tag{7}$$

One common simplification is to consider that the output standard deviations $\big(\sigma_{x_n}\big)_{1 \leq n \leq N}$ are constant [OKS20]. The reconstruction in equation (6) is then comparable to a mean squared error between the input and the reconstructed output, similarly to the autoencoder models. The model is focusing on learning the manifold on which an averaged version of the data lies. Passing an input through the model becomes equivalent to projecting it onto the manifold.

3 Combining neighbor graph and VAE

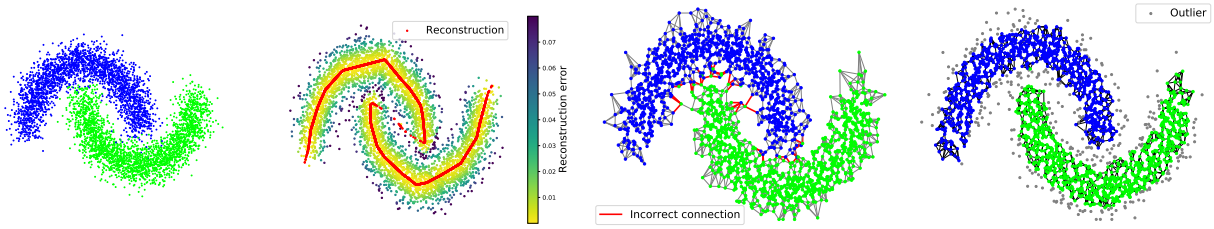
A neighbor graph is a directed graph that aims to capture the structure of the data by evaluating its local connectivity. Each point is connected to its neighbors according to a specified metric. Two common ways for defining the neighborhood of a given point are: the k nearest neighbors or the neighbors within a ball of radius ϵ. In both cases, the parameters k and ϵ represent the size of the neighborhood and therefore control the balance between the preservation of local vs. global data structure. Neighbor graphs are a powerful technique for manifold exploration since, once the graph is computed, operations such as shortest path search are computationally cheap and interpretable. However, they poorly scale up to real world data because of two main limitations. Firstly, the choice of the parameter k or ϵ is very sensitive to noise. Neighbor graphs built on noisy data are likely to incorrectly connects outliers. Secondly, the exact computation of a neighbor graph has an exponential time complexity in the dimension of the data. It has been exposed in section 2 that a variational autoencoder is able to extract a smooth manifold from noisy data and to represent it in a low dimensional latent space. The following subsections details the method for exploiting these outputs in order to improve the quality of the neighbor graph.

3.1 Neighbor graph on averaged data

The figure 1 illustrates the thresholding of the reconstruction error on synthetic data: the two moons dataset. A VAE with one latent dimension is trained in order to learn the data distribution. Figure 1b shows the result of the training. The averaged data, μ_x, learned by the model appears in red, and all the points samples from the original data have been colored according to their reconstruction error. The model encodes successfully the curvy structure of the data and is used to filter out points too far from their reconstruction. This operation allows a more accurate evaluation of the 5-nearest neighbor graph, as shown in figures 1c and 1d, where the graph built directly on the original data present faulty connections, defined as edges that connect two points from distinct clusters.

3.2 Neighbor graph on latent space

The issue of the dimensionality of the input data can be tackled by computing the neighbor graph on the latent space formed by the latent variables $\boldsymbol{z} \in \mathbb{R}^L$. Usually $L \ll N$, which leads to a tremendous gain in computational time. The downside of this method compared to the one described in section 3.1 is that, even if the latent space preserves the overall structure of the data, it doesn't guarantee to preserve the distances [AHH18], and in particular, shortest paths in the latent space do not necessarily correspond to geodesics in the real space. However, in an industrial context, finding the geodesics is not a relevant problem. As discussed in



(a) Noisy two moons (b) 1D manifold (c) Graph on original data (d) Graph on smoothed data

Figure 1: From left to right: fig.1a shows the original two moons data colored by cluster. Fig.1b presents the data colored by the reconstruction error when projecting a point on the averaged manifold given by μ_x, showed in red, learned by a VAE with one dimensional latent space. Fig.1c and 1d compare two different 5-nearest neighbor graphs. The one on fig.1c is built on the original two moons data with $\sigma^2_{Noise} = 0.0225$. The one in fig.1d is built on the data filtered by the value the reconstruction loss. All points with a reconstruction error inferior to σ^2_{Noise} have been considered as outliers. Smoothing the data with this threshold, removes the outliers from the scope of the graph computation, and thus reduces the likelihood of connecting the two clusters.

Section 4, business value comes from the customization of the edge weights in order to incorporate any type of industrial metrics, such as product quality, in the path construction.

For any input point $\boldsymbol{x} \in \mathbb{R}^N$, the VAE computes its averaged version $\boldsymbol{\mu_x} = VAE(\boldsymbol{x})$. And the quadratic error $\mathcal{E}(\boldsymbol{x})$ between $\boldsymbol{x}$ and $\boldsymbol{\mu_x}$ informs about the likelihood of $\boldsymbol{x}$.

$$\begin{aligned} \mathcal{E}(\boldsymbol{x}) &= \sum_{n=1}^{N} \frac{1}{2}\left(x_n - \mu_{x_n}\right)^2 \\ \mathcal{L}(\boldsymbol{x}) &\propto -\exp(\mathcal{E}(\boldsymbol{x})) \end{aligned} \tag{8}$$

Thresholding the likelihood $\mathcal{L}$ is already a natural approach to anomaly detection as explored with good results in [XCZ+18] [SBLvdS16].
Also the likelihood can be used the other way, to detect the most dense regions. That way outliers can be removed and the algorithm is focusing on learning the structure of the most representative part of the data.

Let L_e^- be the sublevel set of $\mathcal{E}$ with level e :

$$L_e^- = \left\{\boldsymbol{x} \in \mathbf{R}^N,\, \mathcal{E}(\boldsymbol{x}) \leq e\right\} \tag{9}$$

And a neighbor graph is computed on this set. The parameter e controls the amount of residual noise and can be easily tuned empirically. Note that, since $\mathcal{E}(\boldsymbol{x}) = 0$ for $\boldsymbol{x} = \boldsymbol{\mu_x}$, the case $e = 0$ is valid and corresponds to the special case of building the graph on the averaged data.

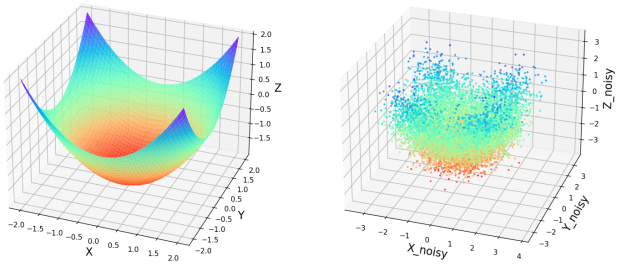
The figure 2 presents the result of an experiment conducted on a synthetic 3D parabolic distribution. Isotropic Gaussian noise has been added to the data with a standard deviation value set to 0.5. This relatively high noise aims to emphasize the limitation of a nearest neighbor graph on a shortest path search between two diametrically opposite samples. When comparing the three paths: the one computed on the latent space, the one computed on the averaged manifold and the one computed on the original noisy data, it first appears that the graph built on noisy data fails to capture the structure of the parabola since the path crosses the center of the parabola instead of following its curvature. On the contrary, the paths computed on the data preprocessed by the VAE are much more realistic as they follow the most dense regions among the historical samples. With the noticeable but expected difference that the path computed on the latent space doesn't necessary follow a geodesic.

In summary, two new ways of evaluating a neighbor graph, while avoiding the detrimental effect of noise, are described. Each one having its benefit and its downside. Building the graph on the averaged data, is a better candidate for finding the geodesics but scales poorly to very high dimensional data. While building the graph on the latent space is computationally efficient but doesn't guarantee to capture all the complexity of the geometrical structure.

4 Application in industry

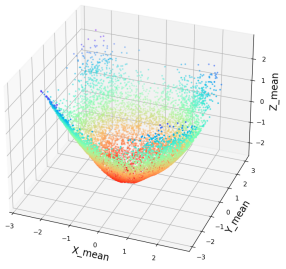
4.1 Exploring the manifold

Most of the business value offered by machine learning in the industry is currently made with predictive mod-

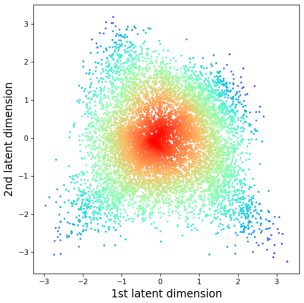


(a) True manifold

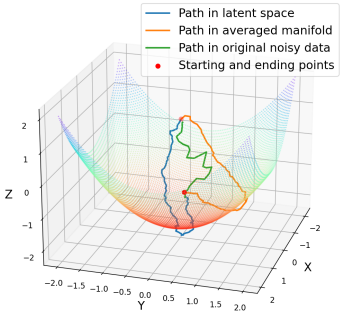
(b) Random samples



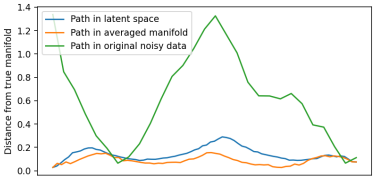
(c) Learned manifold



(d) 2D latent space



(e) Comparison of trajectories embedded in the real space.



(f) Comparison of distances between each point of the trajectories and the nearest point from the true manifold.

Figure 2: Application of the two graph construction methods on a synthetic parabolic distribution. The true manifold follows the equation $z = \frac{1}{2}(x^2 + y^2) - 2$. The figures 2a and 2b shows respectively the true manifold and a noisy version, with isotropic Gaussian noise ($\sigma = 0.5$) that plays the role of the observable data. A VAE reconstructs the manifold and learns a 2D latent space as shown in figures 2c and 2d. After building a 5-nearest neighbor graph on these two outputs in addition to a 5-nearest neighbor graph on the original noisy data, a manifold traversal between two diametrically opposite samples has been experimented. The figures 2e and 2f that the paths built on top of the VAE's outputs are much more accurate in terms of respect of the true curvature of the underlying manifold. It is an explicit illustration of the limited capacity of a neighbor graph to correctly capture the structure of noisy data.

els that can replace expensive and time consuming physical measures [WLS+19]. However, this paradigm is limited by the small amount of labeled data, and thus looses potentially valuable information held by the unlabeled data recorded by sensors along a production line. This article is motivated by the assumption that the industry could also benefit from the shift currently occurring in the artificial intelligence research: there is a growing interest in algorithms that build insightful representations of the data in an unsupervised manner.

After the theoretical considerations presented in the sections 2 and 3, this section introduces an example of business value brought by the approximation of the true distribution of the process variables, learned from unlabeled historical data. Learning the density function of the process leads to a better understanding of its overall behavior, and reduces industrial risk when changing the process parameters. Typical outcomes to be avoided when shifting the process from one set-point to another, are quality degradation and breakdowns. The approach presented in section 3 addressed this issue. Experiments have been conducted on three industrial Kaggle datasets: one issued from a roasting machine [noab] and two provided by Bosch, taken from different stations of a production line [noaa]. See table 1 for more details.

4.2 Influence of the graph construction technique

The first experiment consists in comparing the different graph computation methods on the three dataset. For each dataset, a VAE is trained on the raw data and three nearest neighbor graphs are computed:

Dataset	N	M	L
Roasting Machine	17	2M	4
Bosch Production Line station L0S0	11	670K	6
Bosch Production Line station L1S24	169	180K	16

Table 1: The three industrial datasets from Kaggle used in experiments. N is the number of features, M the number of samples. The value of L is the dimension of the latent space and has been selected empirically.

- 10-nearest neighbor graph built on raw data,
- 10-nearest neighbor graph built on averaged data computed by the VAE,
- 10-nearest neighbor graph built on latent space.

The choice of $k = 10$ showed empirically to be satisfactory and has not been subject to extensive optimization. For the experiment, two nodes are randomly selected, and the path between them is computed with the Dijkstra algorithm [D$^+$59], which consists in finding the path that minimizes the sum of the weights of the visited edges. In this first experiment the weights are set to the Euclidean distances between the two nodes of an edge. This process can be seen as a geodesic estimation between the starting and ending setpoints. Computing the geodesics doesn't have a real industrial application but illustrates the differences in the way each graph captures the structure of the underlying data. Figure 3 provides visual comparisons between the three methods, for each of the datasets, and shows that graph construction influences the obtained paths. In conclusion, the choice of the graph construction technique should be taken into account while applying this method to real world problems.

4.3 Graph weights customization

The problem of manifold exploration can be extended with the incorporation of business metrics through customization of the graph weights. The objective is still to find a path made of viable intermediary states between two setpoints, but with an additional constraint on the exploration dictated by the business metric. To illustrate this customization, we used 30K labeled samples (over a total of 2M historical samples) from the roasting machine dataset, namely samples for which the quality of the final product has been measured and expressed with a numerical variable. This quality variable is incorporated to the weights of the graph in order to avoid regions with low quality. To make it relative, the raw quality variable is replaced by the relative deviation from a reference value. For any node i of the graph: $\text{target}_i = |\frac{\text{quality}_i - \text{quality}_i^{\text{ref}}}{\text{quality}_i^{\text{ref}}}|$. Let j be a node connected to i. The Euclidean distance between them is noted $d_{i,j}$. The new weight of the edge connecting theses nodes: $w_{i,j}$ is given by the following equation:

$$w_{i,j} = d_{i,j} \times \left(\frac{\text{target}_i + \text{target}_j}{2} \right)^n \tag{10}$$

Where $n \geq 1$ is a coefficient that controls the intensity of the penalization. The experiment shows the result for $n = 1$ and $n = 2$, referenced as linear and quadratic weighting in the figure 4. It also shows that smoothing the distribution of the target with a predictive model (here a multi layer perceptron) regularizes the path. In conclusion, with very few additional development it is possible to compute paths that respect both the overall behavior of the process and any external industrial constraints.

5 Conclusion and perspectives

This work introduces a framework that makes nearest neighbor graphs suitable for the task of manifold exploration in the context of noisy and high dimensional data distributions encountered in industry. A Variational AutoEncoder is used to learn the complex interactions between the variables describing an industrial process. It produces three valuable outputs that upgrade the performance of the nearest neighbor graph: a compressed representation of the data, called latent space, that reduces the computational cost of computing the nearest neighbor graph, an averaged version of the data that drastically reduces the noise and thus avoid faulty connections in the graph, and an estimation of the likelihood of any point, that can be thresholded to control the amount of noise in the data. When applied on industrial data, the resulting graph is a great to tool for summarizing the global behavior of a process and can be used to reduce risks inherent to the control of the process parameters. Also, edge weights can be customised according to a business metric in order to avoid some regions of the space.
The experiments conducted during this study were a successful proof of concept for the combination of deep generative models and graphs and open the way to future works. Here are some promising lines of research. Exploiting the learned variance, considered as constant in this study could be an interesting metric for dynamically adapting the connectivity criterion in the graph.

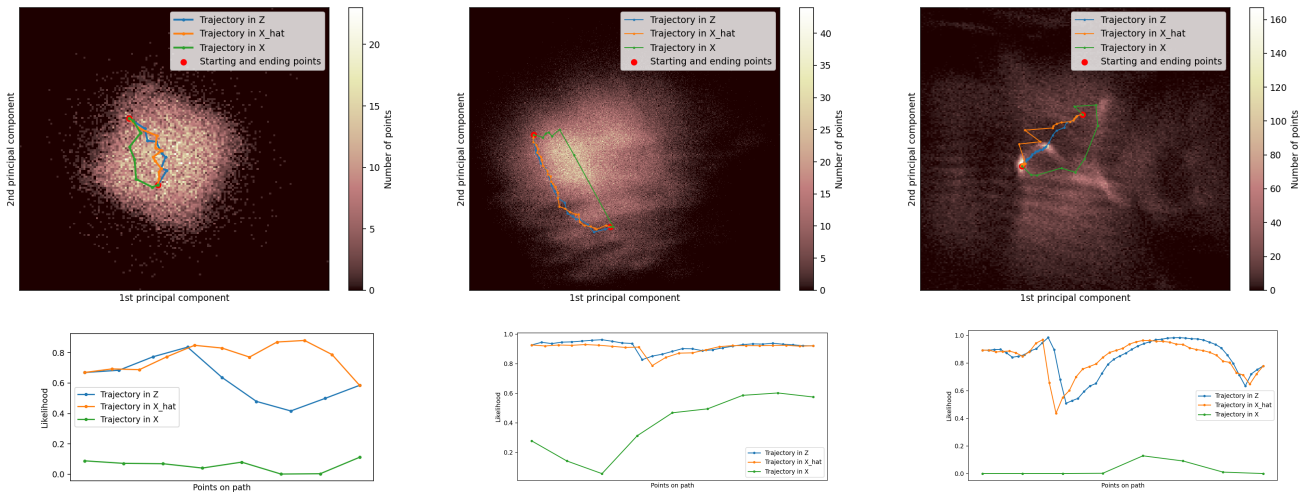
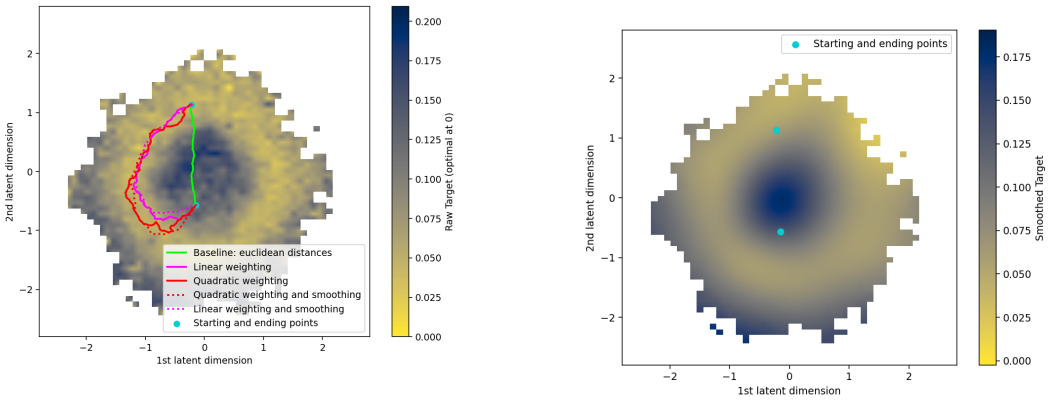


Figure 3: Results of the experiments conducted on three industrial dataset taken from Kaggle. Principal Component Analysis has been performed on resulting latent spaces to allow their visualizations. The first row shows the trajectories followed by each path projected on the first two principal components of the latent space. The paths can also be compared in terms of their likelihood computed by the VAE, as shown in the second row.



(a) Shortest path in a neighbor graph with custom weights

(b) Smoothed target

Figure 4: Using the graph framework to explore industrial manifolds makes the incorporation of business metrics very convenient. Figure 4a shows an example of weight customization with a quality related metric. This metric is preprocessed to build a target that is optimal at 0. This target is used to compute weights that penalize lower quality regions (see section 3 for more details on the weights computation). The dotted paths have been computed with a smoothed target showed in 4b. The smoothed target is the output of a multi layer perceptron.

Also, extending the approach to categorical variables would expand the scope of industrial applications. Finally, more sophisticated weighting formulas can be experimented.

References

[AHH18] G. Arvanitidis, L. Hansen, and S. Hauberg. Latent Space Oddity: on the Curvature of Deep Generative Models. *arXiv:1710.11379 [stat]*, January 2018. arXiv: 1710.11379.

[Bal87] Dana H Ballard. Modular learning in neural networks. In *AAAI*, pages 279–284, 1987.

[BCV14] Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *arXiv:1206.5538 [cs]*, April 2014. arXiv: 1206.5538.

[BKM17] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, Apr 2017.

[Cay05] L. Cayton. Algorithms for manifold learning. July 2005.

[CPZ04] M. Á. Carreira-Perpiñán and R. S. Zemel. Proximity graphs for clustering and manifold learning. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 225–232, Cambridge, MA, USA, December 2004. MIT Press.

[D+59] E. W. Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[KW14] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. arXiv: 1312.6114.

[MHM20] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, September 2020. arXiv: 1802.03426.

[noaa] Bosch production line performance dataset. `https://kaggle.com/c/bosch-production-line-performance`.

[noab] Production quality dataset. `https://kaggle.com/podsyp/production-quality`.

[OKS20] Rybkin O., Daniilidis K., and Levine S. Simple and effective vae training with calibrated decoders, 2020.

[SBLvdS16] M. Soelch, J. Bayer, M. Ludersdorfer, and P. van der Smagt. Variational Inference for On-line Anomaly Detection in High-Dimensional Time Series. *arXiv:1602.07109 [cs, stat]*, June 2016. arXiv: 1602.07109.

[SMAH20] H. Singh, N. McCarthy, Q. Ain, and J. Hayes. ChemoVerse: Manifold traversal of latent spaces for novel molecule discovery. *arXiv:2009.13946 [cs, q-bio]*, September 2020. arXiv: 2009.13946.

[WHP+18] F. Wolf, F. Hamey, M. Plass, J. Solana, J.S. Dahlin, B. Göttgens, N. Rajewsky, L. Simon, and F. J. Theis. Graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. *bioRxiv*, page 208819, November 2018. Publisher: Cold Spring Harbor Laboratory Section: New Results.

[WLS+19] D. Weichert, P. Link, A. Stoll, S. Rüping, S. Ihlenfeldt, and S. Wrobel. A review of machine learning for the optimization of production processes. *The International Journal of Advanced Manufacturing Technology*, 104(5):1889–1902, October 2019.

[XCZ+18] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. page 187–196, 2018.

[ZWZ12] Z. Zhang, J. Wang, and H. Zha. Adaptive Manifold Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):253–265, February 2012. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.