Exploration with task losses for generative modeling

Adel Nabli [1] **Philippe Marchandise** [1]

Abstract

Although generative modeling is a difficult task to evaluate in high dimension, it could be easier to assess it in a low dimension setting (i.e dimension less than 3). Indeed, in a 3D setting, a generative model produces point clouds which can be compared to the true 3D model if we have an access to it. As comparing 3D point clouds is a task appearing in the computer graphics field, we decided to see whether or not the losses from this field could be used to monitor the training of GANs, to evaluate the quality of the generator and serve as task losses. We report promising results and present some argument advocating for their use in higher dimensional setting.

1. Introduction

The problem of generative modeling could be phrased as follows: we want to find a generator that creates samples $\tilde{x} \sim \mathbb{P}_g$ such that $\mathbb{P}_g$ approximate the true unknown distribution of the real data $x \sim \mathbb{P}_r$. A common assumption used to describe the support $\mathcal{X}$ of $\mathbb{P}_r$ is the manifold hypothesis Tenenbaum et al. (2000) i.e the fact most datasets practitioners are dealing with in machine learning, although with a high dimension p, have in fact an underlying non linear structure with a low number of degrees of freedom $d << p$. Taking this viewpoint, the purpose of generative modeling is to recover this low dimensional structure *(which we will name "manifold" without defining rigorously the term)* and be able to sample from it with a distribution copying $\mathbb{P}_r$. The difficulty being that we don't know the true structure of the manifold and only have access to a point cloud sampled from it, i.e the training set.

Taking the GAN framework Goodfellow et al. (2014), we will only consider a certain type of generator g_θ: the ones parameterized as Multilayer Perceptrons with parameters θ and taking a prior noise variable $z \sim p_z$ as input. We can think of z as the variable parametrizing the generated manifold, i.e we want it to have a dimension equal to the number of degrees of freedom of the structure $\mathcal{X}$. Indeed, Arjovsky & Bottou (2017) showed in their **Lemma 1** that $g_\theta(\mathcal{Z})$ is contained in a countable union of manifolds of intrinsic dimension at most $\dim(\mathcal{Z})$, so we want $\dim(\mathcal{Z})$ to be at least equal to the true intrinsic dimension $\dim(\mathcal{X}) = d$.

Then, let's suppose that our training data is uniformly sampled from the manifold $\mathcal{X}$. Even though real world data is rarely so nicely behaved, not only it is theoretically convenient to do that as done in the Laplacian Eigenmap paper Belkin & Niyogi (2003), but also McInnes et al. (2018) state that we can find a metric such that the training data is uniformly sampled on $\mathcal{X}$ under the assumption of the manifold having a Riemannian metric not inherited from the ambient space.
Under this assumption, the task of assessing the goodness of our generator boils down to compare the supports of $\mathbb{P}_g$ and $\mathbb{P}_r$, i.e to give a similarity score between the constructed manifold $g_\theta(\mathcal{Z})$ and the true one $\mathcal{X}$.

But the task of comparing 2 manifolds is generally non trivial. To do so, let's return to our task: the final purpose is to use the defined loss to assess a generator and monitor and guide its learning. If we re-use the words of Huang et al. (2018), what we want is finding a generator g_θ that achieves 2 distinct goals: generate *realistic* and *diverse* samples. This could be rephrased as: we want to build a manifold $g_\theta(\mathcal{Z})$ that is always close to the true one *(realistic samples)* and that manages to recover as much of the true structure $\mathcal{X}$ as possible *(diverse samples)*. The remaining question is then how can we enforce those two goals in the task loss.

In the next section, we will review some of the losses that are used in this context.

2. Review of some losses

As training and assessing the quality of generative models is both a hard and crucial issue, there is an extensive literature in the matter. Here, we will mainly focus on some losses either related to the paper of Huang et al. (2018) or coming from computer graphics.

[1]Universite de Montreal. Correspondence to: Adel nabli <adel.nabli@umontreal.ca>.

Project report for IFT 6132.

2.1. Parametric adversarial divergences

The GAN framework is designed to approximate a solution to the following problem:

$$\inf_{p_{g,\theta}} \sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim p_r,\, \tilde{x} \sim p_{g,\theta}}[f(x, \tilde{x})] \tag{1}$$

If we fix the distribution p_r, then Liu et al. (2017) state that this problem is equivalent to minimizing the adversarial divergence

$$L_{p_r}(\theta) = \sup_{f \in \mathcal{F}} \mathbb{E}_{p_r \otimes p_{g,\theta}}[f] \tag{2}$$

with $\mathcal{F}$ a certain class of parameterized and constrained functions *(arising from both the discriminator's architecture and choice of objective function)*. One of the advantages of using such a task loss is the theoretical guarantees it has: in their **Theorem 10**, Liu et al. (2017) showed that if $L_{p_r}(\theta_n)$ converges to its lower bound, then we have weak convergence of p_{g,θ_n} towards some kind of "optimal" measure *(not necessary p_r as the neural network g_θ we use has not infinite capacity)*. Moreover, Huang et al. (2018) advocate for the use of those kind of losses to train a generator, providing several arguments:

- they manage to integrate the final task in the sense that they can leverage the degrees of freedom they have *(the choice of architecture, the parameters)* to learn a good metric for the task *(whereas non parametric methods have to be provided with the metric to use)*
- they have good sample complexity compared to non parametric methods, which allows them to build a better generator out of a few true samples
- they are robust to transformations in the targeted manifold, meaning that they can provide useful training signals to the generator *(especially if we use a non saturated loss)*

One drawback we can see though is that, in practice, if we want to monitor the training of a generator using these losses to be able to have some feedback on the training, due to the fact that they are upper bounds over a non-trivial set of functions to optimize, we regularly have to pause the learning and spend some time training at convergence a dedicated discriminator, which could be expensive to do.

2.2. Losses from computer graphics

A convenient proxy to directly compare manifolds is using point clouds as the representation of manifolds by samples and comparing the point clouds instead Mémoli & Sapiro (2004). As working with discretized manifolds *(surface meshes or point clouds constructing 3D models)* is the canonical task of computer graphics, there is a rich literature in this field concerning ways to compare two discretized 3D models. We will present here a selection of the existing approaches from this field and discuss the properties of those losses with respect to the two goals presented in the Introduction.

First, let's formalize the setting. In order to compare the two manifolds $\mathcal{X}$ and $g_\theta(\mathcal{Z})$, we make the assumption that we have access to two points clouds R *(the real data)* and G *(the generated data)* that are uniformly sampled from them. Moreover, in order to balance the comparison, we make sure that we have the same number of points m for both point clouds. Then, we consider the following losses:

2.2.1. Hausdorff distance

The Hausdorff distance is a common measure of dissimilarity between two point sets Taha & Hanbury (2015). Intuitively, it computes the biggest "gap" between the two manifolds, using the following formula as definition of "biggest directed gap":

$$\tilde{H}(R, G) = \max_{r \in R} \left(\min_{g \in G} \|r - g\|_2 \right) \tag{3}$$

and then, as $\tilde{H}(R, G) \neq \tilde{H}(G, R)$ in general, we use the $\max$ of both as distance:

$$d_H(G, R) = \max\{\tilde{H}(R, G), \tilde{H}(G, R)\} \tag{4}$$

Thanks to this symmetrization, the penalization of mode collapse is enforced. One drawback of this loss being it is very sensitive to outliers *(they can easily grow the "biggest gap")*.

2.2.2. Dice coefficient

The Dice coefficient is an *overlap based measure*: it tries to quantify how much both manifolds have in common. One way of doing that in computer graphics would be to discretize the ambient space using voxels and count how many voxels are occupied by both point clouds. In order to enforce both precision *(number of voxels in common $v_{G \cap R}$ divided by the number of voxels generated v_G)* and recall *(number of voxels in common $v_{G \cap R}$ divided by the number of voxels to recover v_R, penalize the mode collapse)*, we use the F_1 score, which is called the Dice coefficient in this case Vera et al. (2013):

$$d_D(G, R) = 2 \frac{v_{G \cap R}}{v_G + v_R} \tag{5}$$

Here, it's the fact that we combine both precision and recall that enforce the two goals in the loss.

2.2.3. CHAMFER DISTANCE

An other commonly used metric to compare point clouds is the Chamfer distance Achlioptas et al. (2018):

$$d_{CH}(G,R) = \sum_{g\in G} \min_{r\in R} \|g-r\|_2^2 + \sum_{r\in R} \min_{g\in G} \|r-g\|_2^2 \quad (6)$$

We can think of it as an averaged Hausdorff: instead of taking into account only the biggest gap, we sum every gap, which makes it less sensitive to outliers.

2.2.4. NEAREST NEIGHBOR DISTANCE

A simpler approach would be to only take into account the second term of the Chamfer distance:

$$d_{NN}(G,R) = \sum_{r\in R} \min_{g\in G} \|g-r\|_2^2 \quad (7)$$

Then, minimizing this distance could be interpreted as maximizing the likelihood of the true data knowing the generated one, which forces the generator to create diverse samples in order to make the true ones likely, preventing mode collapse. This approach was attempted in the context of generative modeling in Li & Malik (2018).

2.3. L_2 loss using Nearest Neighbor in latent space as target

An other point of view we can adopt to define a task loss is seeing the task of the generator as a regression one. Indeed, intuitively what we want to do is mapping a point z in the coordinate system of the latent space to a point x on the manifold embedded in the ambient space. The problem being that we don't have formed pairs (z, x), only some z we can sample and some examples of true x without obvious connection between the two. To remedy that, for each possible z we could sample, we could try to associate one x from the training set using some manifold learning techniques and train g_θ using the L_2 loss for regression on those artificial pairs. Let's formalize a bit the method.

If we think of z as a variable parameterizing the true manifold, we can see $\mathcal{X}$ as a parameterized curve in the ambient space, allowing us to define $\gamma : \mathcal{Z} \to \mathcal{X}$. For convenience, we will assume that γ is bijective. As $\gamma(\mathcal{Z}) = \mathcal{X}$ is unknown, what we want is estimate γ using a parameterized estimator g_θ. A natural task loss would then be:

$$L_\gamma(\theta) = \mathbb{E}_{z\sim p_z}[l(g_\theta(z), \gamma(z))] \quad (8)$$

with l a classical regression loss *(e.g regularized L_2)* and p_z the uniform distribution on $\mathcal{Z}$ *(it is well defined as our training set is finite, we can restrain ourselves to compact supports for our distributions)*. With this task loss, we would indeed reach the two goals of the Introduction: as $\gamma(\mathcal{Z})$ covers all of $\mathcal{X}$, we penalize the problem of mode collapse if $g_\theta(\mathcal{Z})$ only recovers parts of $\mathcal{X}$, and l naturally enforce the closeness of the two manifolds.

Algorithm 1 Regression loss using NN in latent space

Input: training set $\mathbf{X}$, latent dimension d, set of input noises $\mathbf{Z}$, generator g_θ
$\mathbf{Z}_{train}$ = NonLinearDimRed($\mathbf{X}$, d)
$\mathbf{Z}_{train}$ = StandardScale($\mathbf{Z}_{train}$)
$\mathbf{Id}_{train}$ = NearestNeighbor($\mathbf{Z}$, $\mathbf{Z}_{train}$)
targets = $\mathbf{X}[\mathbf{Id}_{train}]$
output $L_2(g_\theta(\mathbf{Z}), \text{targets})$

What prevents us from directly using (8) is the fact that γ is unknown, so we can't have access to $\gamma(z)$. But, using the Nearest Neighbor algorithm in the latent space *(as the latent space dimension d is allegedly much lower than the ambient one p, running this algorithm there should be feasible)* we could estimate it with a $\gamma(z')$ such that:

$$z' = \arg\min_{\gamma^{-1}(x)} \|z - \gamma^{-1}(x)\|_2 \quad (9)$$

with x in the training set. The use of an euclidean distance being justified is we suppose the latent space correctly "straightened". Now, to estimate $\gamma^{-1}(x)$ for every x of the training set, we could use some non-linear dimension reduction techniques from the manifold learning field *(e.g Laplacian Eigenmap, UMAP, Auto-Encoders)*.
We summarize the procedure in **Algorithm 1**.

2.4. A note on Optimal transport distances and Maximum Mean Discrepancies

As optimal transport could be described as the problem of minimizing the cost of transporting a source probability distribution to a target one, the optimal transport distance seems an appropriate task loss here. We could formalize it as:

$$L_{p_r}(\theta) = \min_\pi \mathbb{E}_{(x,\tilde{x})\sim\pi}[c(x,\tilde{x})] \quad (10)$$

with π a coupling measure between p_r and $p_{g,\theta}$, and c a cost function representing the work needed to move a unit of mass x to $\tilde{x}$. Thanks to the emergence of faster algorithm to compute optimal transport distances *(Benamou & Brenier (2000) with a numerical partial differential equation approach, Cuturi (2013) with the addition of a maximum entropy penalty to speed up the computation)*, there have been more and more use-cases for these distances in both machine learning and computer graphics. This kind of loss has been used as a mean to quantify the distance between two discrete 3D models Lavenant et al. (2018) and in the

context of generative modeling for high dimensional data Cuturi (2013).

An other way to tackle the issue would be considering the comparison of p_r and $p_{g,\theta}$ as a two sample problem, i.e provided with samples from both distribution, managing to say whether $p_r = p_{g,\theta}$ or not. In order to tackle this problem, Fortet & Mourier (1953) proposed a statistic to measure the discrepancy between the two measures that only takes into account their means, statistic which was later called the *Maximum Mean Discrepancies* by Gretton et al. (2007):

$$L_{p_r}(\theta) = \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{\tilde{x} \sim p_{g,\theta}}[f(\tilde{x})] \right) \quad (11)$$

$$= \int k(x, \tilde{x}) d\xi(x) d\xi(\tilde{x}) \quad (12)$$

with $\xi = p_r - p_{g,\theta}$, $\mathcal{F}$ a unit ball in a universal RKHS with associated kernel $k(.,.)$ such that $\forall x,\ f(x) = \langle k(x,.), f \rangle$. Under such conditions, Gretton et al. (2007) showed in their **Theorem 3** that $L_{p_r}(\theta) = 0$ iff $p_r = p_{g,\theta}$.

Feydy et al. (2018) state that for both losses, $L_{p_r}(\theta) = 0$ is equivalent to have weak convergence of p_{g,θ_n} towards p_r. As both losses have their own advantages *(appealing geometric properties for the OT distance and cheap computation for the MMD)*, Feydy et al. (2018) proposed an interpolation between them under the name of *Sinkhorn divergences*, introducing a new mixing parameter ϵ and managed to prove interesting theoretical properties for these new losses *(positivity, convexity, scalability to large dataset)*.

3. Experiments

As we were interested in assessing the properties of the losses from the computer graphics field in comparison with other more classical losses, we considered a simpler problem than usual in generative modeling in machine learning: instead of trying to generate highly dimensional points *(e.g images)* belonging to an unknown manifold, we only considered 3D points lying on a known 2D manifold. The training data was points generated from the surface of a 3D model *(we used the Stanford bunny and dragon (Sta))*. Then, our purpose was to build a generator that would generate point clouds recovering the manifold *(in our case, the bunny)*. This setting allowed us to visualize the geometry of the generated point cloud in comparison to the true geometry from the model *(which allows to visually assess both the quality and diversity of the data generated)*, which is usually not possible to do in the high dimensional setting *(in the high dimensional case, we can usually only visualize alone "points" and judge by ourselves whether they seem to belong to the manifold, i.e if they seem realistic, but more "global" point of view are hard to obtain)*.

We assessed the different losses on two tasks:

- Do they allow us to monitor the training of the generator and constitute good structured loss to assess the quality of the trained model ?
- Can they constitute good task losses to train the model with ?

As it seemed non-trivial to implement the Dice coefficient in a differentiable way, we only considered it as a way to monitor the training and assess the quality of the generator.

3.1. Experimental setup

We attempted two types of experiments: to generate points from the bunny and to generate points from two disconnected manifolds *(the bunny next to the dragon)*. In order to train our generator, we used as a training set a sample of 10000 points from the bunny and of 20000 points from the union of the bunny and the dragon.

(a) 10000 points sampled from the 3D bunny

(b) 20000 points sampled from the union of the bunny and the dragon

Figure 1. Datasets used for the experiments

Throughout the experiments, we took the same architecture for the generators g_θ: an input z of dimension 2 *(we want to generate a surface)* following $z \sim \mathcal{N}(0,1)$ *(or* $z \sim \mathcal{U}(-0.5, 0.5)$ *for the regression experiment)*, a MLP with 2 hidden layers of hidden dimension 3000, a Leaky-RELU non linearity with $\alpha = 0.01$, and an output dimension of 3 *(the ambient space is 3 dimensional)*. We trained our generators with the Adam optimizer and using 6 different task losses:

- Using a **parametric adversarial divergence** approximating the Jensen Shannon Divergence (JSD) at convergence *(Vanilla GAN)*.
- Using the **Hausdorff distance**
- Using the **Chamfer distance**
- Using the **Nearest Neighbor distance**
- Using the L_2 **loss** with target corresponding to the Nearest Neighbor in latent space

- Using the **Sinkhorn Divergence**

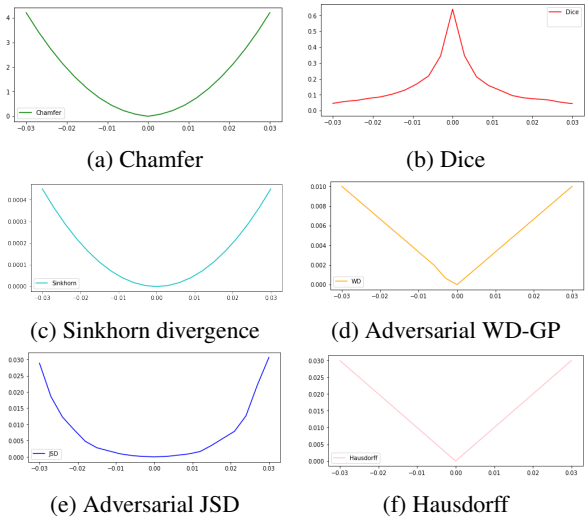
We monitored the learning and assessed the final model using: the adversarial JSD and Wasserstein Distance with gradient penalty, the Hausdorff distance, the Chamfer Distance and the Dice coefficient.

As the Dice coefficient is sensible to the scale of the voxels we use *(by using a voxel size too small, the number of voxels shared between the point cloud generated and the one sampled from the manifold would be small if the number of points sampled is not large enough, and conversely, by using too large voxels, any point sampled from the generator would lie inside a voxel from the true manifold)*, we tried to make this measure less sensitive to this variable by tying the voxel size used to the number of point constituting the point cloud with the following formula:

$$length_{voxel} = \sqrt{\frac{surface_{manifold}}{n_{point\ cloud}}} \tag{13}$$

With this formula, a point cloud generated from the true manifold would have a Dice coefficient of 1 in expectation.

Finally, we also tried to reproduce a similar experiment than in Huang et al. (2018) to assess the robustness of the losses to small transformations of the manifold. For that, we compared the divergences of a point cloud sampled from the buny and one sampled from the buny translated of ϕ according to the x-axis. We report those results in Figure 2.



(a) Chamfer (b) Dice

(c) Sinkhorn divergence (d) Adversarial WD-GP

(e) Adversarial JSD (f) Hausdorff

Figure 2. Evolution of divergences with a small change ϕ

3.2. Results

In this section, we report the results of our experiments. We begin in Table 1 and Table 2 with the end values of our metrics for the several generator for both experiments.

TASK LOSS	HAUSD	DICE	CHAMF	JSD	WD-GP
CHAMFER	**0.016**	**0.275**	**0.152**	$7e-4$	$1e-3$
L_2	0.109	0.002	34.48	-	-
HAUSD	0.017	0.108	0.683	-	-
SINKH	0.073	0.071	1.56	-	-
NN	0.099	0.171	0.656	-	-
GAN	0.023	0.179	0.311	$7e-4$	$9e-4$

Table 1. Metric values after training the generator with several task losses on the bunny experiment

TASK LOSS	HAUSD	DICE	CHAMF
CHAMFER	**0.026**	**0.083**	**0.92**
L_2	0.190	0.003	183.1
HAUSD	0.055	0.031	5.95
SINKH	0.076	0.038	3.28
NN	0.149	0.074	1.355
GAN	0.132	0.018	40.17

Table 2. Metric values after training the generator with several task losses on the bunny and dragon experiment

Then, we also visualized what looked like a point cloud generated from the several generators for both experiments, which can be seen in Figure 3 and Figure 4.

Finally, we also report the monitoring of several indicators during the training procedure in Figure 5.

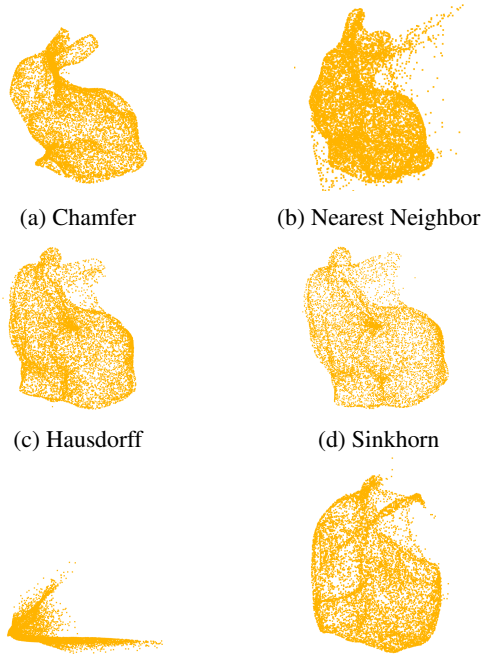
3.3. Discussion

What immediately comes when looking at the results is that the generator trained on the Chamfer distance consistently gives the best results on both experiments quantitatively speaking, but has also seemingly managed to learn the true manifold structure the best on both experiments *(the point clouds it generates resembles the most the true 3D models)*.

We report also good results for the generators trained on the Hausdorff distance, the Nearest Neighbor distance and the Sinkhorn Divergence for both tasks.

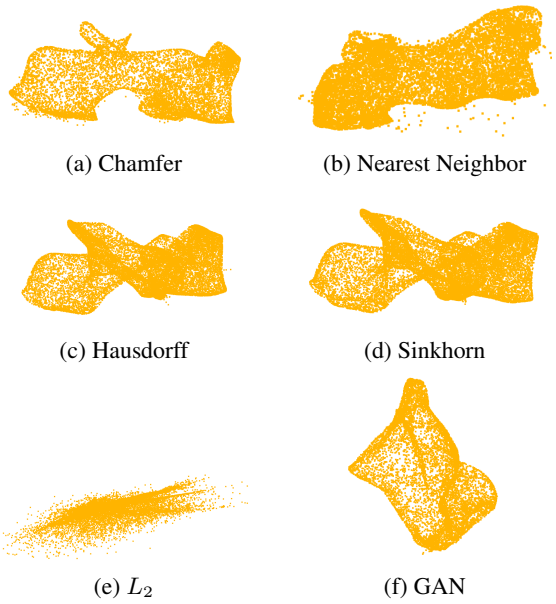
On the other hand, the GAN has been hard to train and we didn't manage to find a good way to train it on the second experiment.

As for the generator trained as a regressor, we didn't manage to produce any good looking results. We don't



(a) Chamfer (b) Nearest Neighbor (c) Hausdorff (d) Sinkhorn (e) L_2 (f) GAN

Figure 3. Generated point clouds for the bunny experiment



(a) Chamfer (b) Nearest Neighbor (c) Hausdorff (d) Sinkhorn (e) L_2 (f) GAN

Figure 4. Generated point clouds for the bunny-dragon experiment

know if it's because our approach is intrinsically flawed or rather if the method has to be finely tuned to produce any sound result.

Finally, we also notice on the second experiment that every generator have learnt a "connected" version of the setup, they didn't manage to deal with disconnected manifold and have built "bridges" connecting them. This could have been expected as the input space $\mathcal{Z}$ is connected and g_θ is continuous and we know that continuous functions keep intact the connectedness of spaces.

4. To go further: related work

The idea of fitting a generative model to lean the structure of a 3D model is not new, Eckart et al. (2016) used the EM algorithm to fit a Gaussian Mixture Model on the Stanford bunny. The purpose was to compress the model using less parameters in the generator than number of points from the complete 3D model. We did not study the compression properties of our generator and did not attempted to study the effect of the capacity of the generator on the quality of the point cloud it could sample, it would be an interesting question to dig.

In Achlioptas et al. (2018) and Li et al. (2018), the authors used a GAN as a 3D point cloud generator. Their work is different in nature though, as the task they tackled is highly dimensional: instead of focusing on one particular 3D model as we did, they considered point clouds from lots of different 3D models, each of those point clouds being one training example/a high dimensional point in their dataset.

As the losses we used from computer graphics mainly lean on the Nearest Neighbor algorithm, it could be challenging to directly extend the work done in three dimensions to high dimension settings *(e.g on image synthesis)* as this algorithm usually suffers from the curse of dimensionality. But Li & Malik (2017) described a fast way to do exact k-nn search which they used later in Li & Malik (2018) to take the nearest neighbor distance (7) as a task loss for a generative model for image synthesis. Following work by Hoshen & Malik (2018) improved upon these results by working in a latent space for the search of a nearest neighbor. As we have seen in our case that the Chamfer distance gives better results than the Nearest Neighbor distance, it would be interesting to see whether replacing the loss they used with the Chamfer one would lead to improvements in the setting they studied.

On an other subject, the use of a Dice-like coefficient to quantify the shared support for the two distributions *(the true one and the generated one)* has been used recently in Sajjadi et al. (2018) where the authors developed an algorithm that estimate the *precision* and *recall* from samples to quantify the quality of generative models (GANs and VAEs).

As for the disconnected manifold problem, Khayatkhoei et al. (2018) proposed a method using a collection

of generators *(one for each connected component)* coupled to a prior distribution to navigate between the generators to overcome the issue, which would have been interesting to test here.

5. Conclusion

We explored several task losses for generative models trying to generate points lying on a structured 2D manifold embedded in a 3D ambient space. We report promising results for losses coming from the computer graphics an optimal transport field, which are not only easier to train than a Vanilla GAN, but also display competitive results. As new algorithms allow for the fast search for Nearest Neighbors in high dimensional space, we would be curious to see how those results would translate in higher dimensional settings. We would also be interested in following the evolution of the recent python package we used for the computation of the sinkhorn divergence (GeomLoss) to see whether it could constitute a good way to train generative models in high dimension.

References

The stanford 3d scanning repository. http://graphics.stanford.edu/data/3Dscanrep/.

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. Learning representations and generative models for 3D point clouds. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 40–49, 2018.

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, June 2003. ISSN 0899-7667.

Benamou, J.-D. and Brenier, Y. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, Jan 2000.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2292–2300. 2013.

Eckart, B., Kim, K., Troccoli, A., Kelly, A., and Kautz, J. Accelerated generative models for 3d point cloud data. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5497–5505, 2016.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouvé, A., and Peyré, G. Interpolating between Optimal Transport and MMD using Sinkhorn Divergences. *arXiv e-prints*, art. arXiv:1810.08278, 2018.

Fortet, R. and Mourier, E. Convergence de la répartition empirique vers la répartition théorique. *Annales scientifiques de l'École Normale Supérieure*, 3e série, 70(3): 267–285, 1953.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. 2014.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. J. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems 19*, pp. 513–520. 2007.

Hoshen, Y. and Malik, J. Non-adversarial image synthesis with generative latent nearest neighbors. 2018.

Huang, G., Berard, H., Touati, A., Gidel, G., Vincent, P., and Lacoste-Julien, S. Parametric adversarial divergences are good task losses for generative modeling, 2018.

Khayatkhoei, M., Singh, M. K., and Elgammal, A. Disconnected manifold learning for generative adversarial networks. In *Advances in Neural Information Processing Systems 31*, pp. 7343–7353. 2018.

Lavenant, H., Claici, S., Chien, E., and Solomon, J. Dynamical optimal transport on discrete surfaces. *ACM Trans. Graph.*, 37(6):250:1–250:16, December 2018. ISSN 0730-0301.

Li, C., Zaheer, M., Zhang, Y., Póczos, B., and Salakhutdinov, R. Point cloud GAN. 2018.

Li, K. and Malik, J. Fast k-nearest neighbour search via prioritized DCI. 2017.

Li, K. and Malik, J. Implicit maximum likelihood estimation. 2018.

Liu, S., Bousquet, O., and Chaudhuri, K. Approximation and convergence properties of generative adversarial learning. In *Advances in Neural Information Processing Systems 30*, pp. 5545–5553. 2017.

McInnes, L., Healy, J., and Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, 2018.

Mémoli, F. and Sapiro, G. Comparing point clouds. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 32–40, 2004. ISBN 3-905673-13-4.

Sajjadi, M. S. M., Bachem, O., Lučić, M., Bousquet, O., and Gelly, S. Assessing Generative Models via Precision and Recall. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Taha, A. A. and Hanbury, A. An efficient algorithm for calculating the exact hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11): 2153–2163, 2015.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.

Vera, S., Gil, D., Borràs, A., Linguraru, M. G., and González Ballester, M. A. Geometric steerable medial maps. *Machine Vision and Applications*, 24(6):1255–1266, Aug 2013.

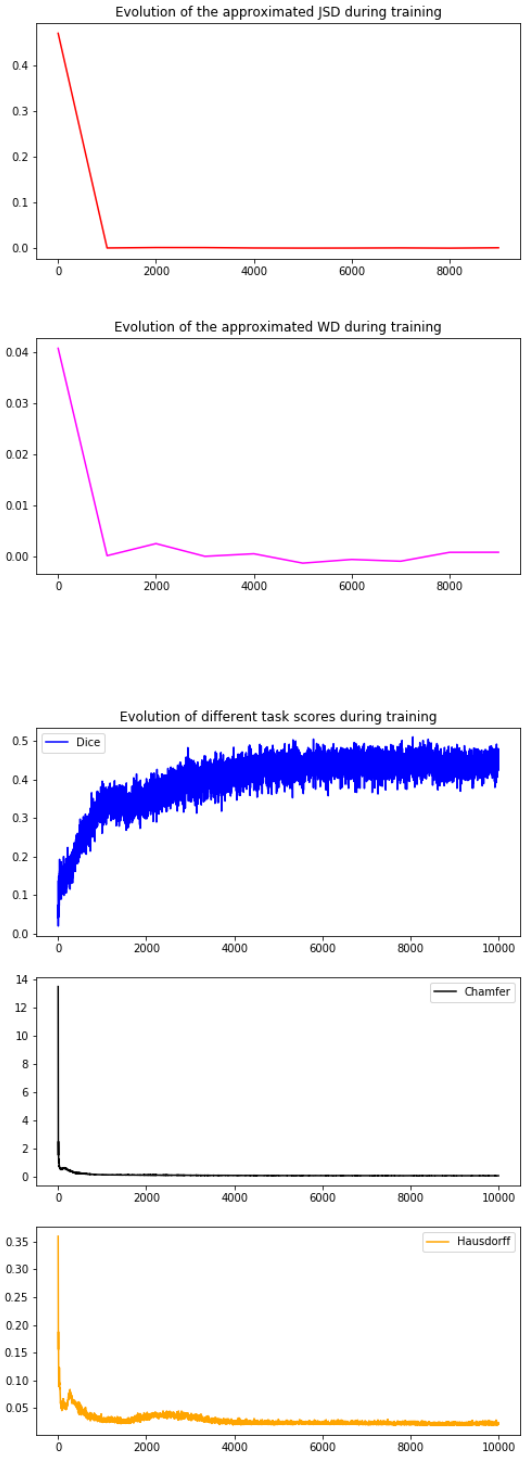


Figure 5. Typical evolution of different losses during training