Generative Models for images Project - MVA 2024/2025

Dany Mostefai Danymost@protonmail.com
Yoann Loesch yoann.loesch@gmail.com

March 21, 2025

1 Introduction

Most machine learning methods use the optimal transport cost as a loss for generative models. Since 2021 [1], it has been shown that optimal transport plan can be used as a generative model, achieving similar performances.

Scientific works before this paper generally use Wasserstein 2-cost and try to recover nonstochastic Optimal Transport map, which may not always exist [1]. The contribution of this article is to propose an efficient algorithm for computing both stochastic and nonstochastic optimal transport maps with deep neural networks.

In this project, we will present the main results of the paper, carry out experiments on other datasets and see the limitations.

2 Preliminaries about Optimal Transport

Notations : $\Pi(\mathbb{P}, \mathbb{Q})$ is the set of probability distributions on $\mathcal{X} \times \mathcal{Y}$ with marginals $\mathbb{P}$ and $\mathbb{Q}$.

2.1 Overview of OT formulations

For $\mathbb{P} \in \mathcal{P}(\mathcal{X})$, $\mathbb{Q} \in \mathcal{P}(\mathcal{Y})$ and a cost function $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, the **Strong OT formulation** is

$$\text{Cost}(\mathbb{P}, \mathbb{Q}) \overset{\text{def}}{=} \inf_{T_{\#}\mathbb{P}=\mathbb{Q}} \int_{\mathcal{X}} c(x, T(x)) d\mathbb{P}(x) \qquad (1)$$

where $T : \mathcal{X} \to \mathcal{Y}$ is a measurable function that maps $\mathbb{P}$ to $\mathbb{Q}$. The optimal T^* is called the **OT map**. The drawback of this expression is that the cost is not symmetric and that there may not be a push-forward measure $T_{\#}$ such that : $T_{\#}\mathbb{P} = \mathbb{Q}$ (Take for instance $\mathbb{P}$ as one dirac and $\mathbb{Q}$ as two diracs : you can't split the mass of one point with Monge's formulation).

Therefore, Kantorovitch proposed the following relaxation :

$$\text{Cost}(\mathbb{P}, \mathbb{Q}) \overset{\text{def}}{=} \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{Y}} c(x, y) d\pi(x, y) \qquad (2)$$

The optimal π^* is called the **optimal transport plan**. If π^* is of the form $[id, T^*]_{\#}\mathbb{P}$ for some T^*, then the plan is called **deterministic**. If not, it is called **stochastic**.

Let $C : \mathcal{X} \times \mathcal{P}(\mathcal{Y}) \to \mathbb{R}$ be a **weak cost** where $\pi(\cdot|x)$ denotes the conditional distribution. The **Weak OT formulation** is :

$$\text{Cost}(\mathbb{P}, \mathbb{Q}) \overset{\text{def}}{=} \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X}} C(x, \pi(\cdot|x)) d\pi(x), \qquad (3)$$

The advantage of this formulation is to be a generalization of Kantorovich's one (2).

2.2 Example of weak OT cost

For $\mathcal{X} = \mathcal{Y} = \mathbb{R}^D, \gamma \in \mathbb{R}_+$, one can take as a weak OT cost the γ-weak Wasserstein-2 $\mathcal{W}_{2,\gamma}$:

$$C(x, \pi(.|x)) = \underbrace{\int_{\mathcal{Y}} \frac{1}{2} \|x - y\|^2 d\pi(y|x)}_{\text{Dissimilarity}} - \frac{\gamma}{2} \underbrace{\text{Var}(\pi(.|x))}_{\text{Diversity}}$$

The first term of the loss measures the dissimilarity while the second the diversity of the generated points.

2.3 Dual form of the weak OT

For weak costs $C(x, \mu)$ which are lower bounded, convex in μ and jointly lower semicontinuous in a certain manner. These assumptions allow to prove the existence of the minimizer π^* in (3). We can then define the *dual form* of (3) as :

$$\text{Cost}(\mathbb{P}, \mathbb{Q}) = \sup_f \int_{\mathcal{X}} f^C(x) d\mathbb{P}(x) + \int_{\mathcal{Y}} f(y) d\mathbb{Q}(y) \qquad (4)$$

where $f^C(x) \overset{\text{def}}{=} \inf_{\mu \in \mathcal{P}(\mathcal{Y})} \left\{ C(x, \mu) - \int_{\mathcal{Y}} f(y) d\mu(y) \right\}$ where f are continuous functions on $\mathcal{Y}$ with 'not very rapid growth'.

Intuitively, the potentials f and f^C correspond to a "price".

3 Weak Optimal Transport for generative modeling

3.1 Maximin reformulation of the dual problem

Let $\mathcal{Z} \in \mathbb{R}^S$ with an atomless distribution $\mathbb{S}$. For example, we can take $\mathbb{S} = \mathcal{U}([0, 1])$ or $\mathcal{N}(0, 1)$.

The authors have reformulated the dual problem (4) as a saddle point optimization problem :

$$\text{Cost}(\mathbb{P}, \mathbb{Q}) = \sup_f \inf_T \mathcal{L}(f, T) \qquad (5)$$

where $\mathcal{L}(f, T) \overset{\text{def}}{=} \int_{\mathcal{Y}} f(y) d\mathbb{Q}(y) + \int_{\mathcal{X}} \left(C(x, T(x, \cdot)\#\mathbb{S}) - \int_{\mathcal{Z}} f(T(x, z)) d\mathbb{S}(z) \right) d\mathbb{P}(x)$

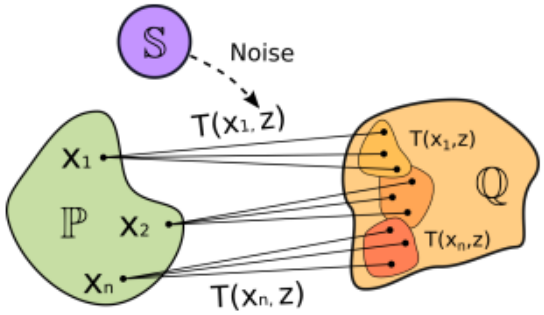


Figure 1: Illustration of the implicit representation of transport maps π by a stochastic function T

If T is independant of z, the map is ***deterministic***. Otherwise, T is a ***stochastic map***.

Let's notice that transport maps $\pi \in \Pi(\mathbb{P}, \mathbb{Q})$ can be represented implicitely through a stochastic function T. This is called **Noise outsourcing**.

3.2 Theoretical results on Maximin reformulation

The authors of the paper showed that :
Lemma 1 : For a stochastic map T^* which represents implicitely an optimal transport plan π^*, $T^* \in \arg\inf_T \mathcal{L}(f^*, T)$ for optimal f^* of (4).
In other words, optimal maps solve the maximin problem (5) for optimal f^*.

Lemma 2 : For strictly convex $\mu \mapsto C(\cdot, \mu)$ in μ, solving $\arg\inf_T \mathcal{L}(f^*, T)$ leads to stochastic OT maps.

However, the γ-weak cost is not striclty convex with respect to μ. Therefore, solving $\arg\inf_T \mathcal{L}(f^*, T)$ may **contain non-optimal OT maps** which is one of the main drawbacks of maximin reformulation. We will study this phenomenon in the section 4.2.

3.3 Neural Optimal Transport (NOT)

The authors proposed to train neural networks $T_\theta : \mathbb{R}^P \times \mathbb{R}^S \to \mathbb{R}^Q$ and $f_\omega : \mathbb{R}^Q \to \mathbb{R}$ to respectively parametrize T and f.

The parameters are trained with the algorithm below 1.

Algorithm 1: Neural optimal transport (NOT)

Input : distributions $\mathbb{P}, \mathbb{Q}, \mathbb{S}$ accessible by samples; mapping network $T_\theta : \mathbb{R}^P \times \mathbb{R}^S \to \mathbb{R}^Q$;
potential network $f_\omega : \mathbb{R}^Q \to \mathbb{R}$; number of inner iterations K_T;
(weak) cost $C : \mathcal{X} \times \mathcal{P}(\mathcal{Y}) \to \mathbb{R}$; empirical estimator $\widehat{C}\big(x, T(x, Z)\big)$ for the cost;

Output : learned stochastic OT map T_θ representing an OT plan between distributions $\mathbb{P}, \mathbb{Q}$;

repeat

  Sample batches $Y \sim \mathbb{Q}$, $X \sim \mathbb{P}$; for each $x \in X$ sample batch $Z_x \sim \mathbb{S}$;

  $\mathcal{L}_f \leftarrow \frac{1}{|X|}\sum_{x\in X} \frac{1}{|Z_x|}\sum_{z\in Z_x} f_\omega\big(T_\theta(x,z)\big) - \frac{1}{|Y|}\sum_{y\in Y} f_\omega(y)$;

  Update ω by using $\frac{\partial \mathcal{L}_f}{\partial \omega}$;

  for $k_T = 1, 2, \ldots, K_T$ **do**

    Sample batch $X \sim \mathbb{P}$; for each $x \in X$ sample batch $Z_x \sim \mathbb{S}$;

    $\mathcal{L}_T \leftarrow \frac{1}{|X|}\sum_{x\in X}\Big[\widehat{C}\big(x, T_\theta(x, Z_x)\big) - \frac{1}{|Z_x|}\sum_{z\in Z_x} f_\omega\big(T_\theta(x,z)\big)\Big]$;

    Update θ by using $\frac{\partial \mathcal{L}_T}{\partial \theta}$;

until *not converged*;

Figure 2: NOT algorithm

This algorithm has a theoretical background because the article demonstrates that neural networks are universal approximators of stochastic transport maps. Once the models are trained, we can use the transport map to generate images like shown in Figure 2.

3.4 Links to WGAN

WGAN computes the optimal transport cost and uses it as a loss function in order to train the generator whereas the NOT algorithm aims to learn stochastic optimal transport plans as generative model. Moreover, the NOT algorithm does not have any constraints on f contrary to the 1-Lipschitz continuity of the Wgan's discriminator.

Adversarial training in WGAN can be seen as a minimax whereas it corresponds to a maximin for NOT (see (5)) Indeed, for a transport map T (generator) and a discriminator f, we have: $\inf_{T\in\mathcal{G}} \sup_{f\in\mathcal{D}} \mathbb{E}[f(X)] - \mathbb{E}[f(T(Z))] = \inf_T \sup_f \mathcal{L}(T, f)$

We can understand this difference by noting that in NOT the generator T is adversarial to the discriminator f unlike in standard WGAN.

4 Experiments

4.1 Methodology

We will carry out first experiments on 2D points synthetic and see how NOT algorithm performs. Then, we will start try to colorize images by learning an OT map that maps $\mathbb{P}$ =Grayscale images to $\mathbb{Q}$ = 'RGB' images.

4.2 2D Synthetic Dataset

All the details about the models and so on, can be found in Appendix A.

First experiment : We start from $\mathbb{P} = \mathcal{N}(0, \mathbf{I}_2)$ and wanted to learn a stochastic map $\hat{T}$ that maps to a distribution that forms the letter 'O' and 'T' (see figure 3.)

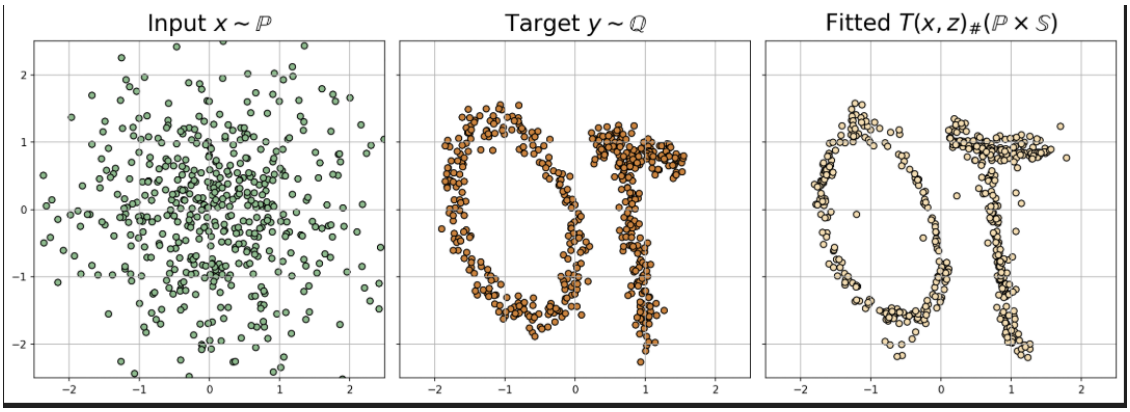


Figure 3: Stochastic maps $\hat{T}$ between $\mathbb{P}$ and $\mathbb{Q}$ with $\gamma = 1$.

The training was quite stable and we can observe that $\hat{T}_{\#}(\mathbb{P} \times \mathbb{S}) = \mathbb{Q}$ on this particular example.

Second experiment : We decided to start from a $\mathbb{P}$ which is composed of 8 gaussian blobs that we moved to 8 differents places on the map. And we took $\mathbb{Q}$ as a square with random points in it.

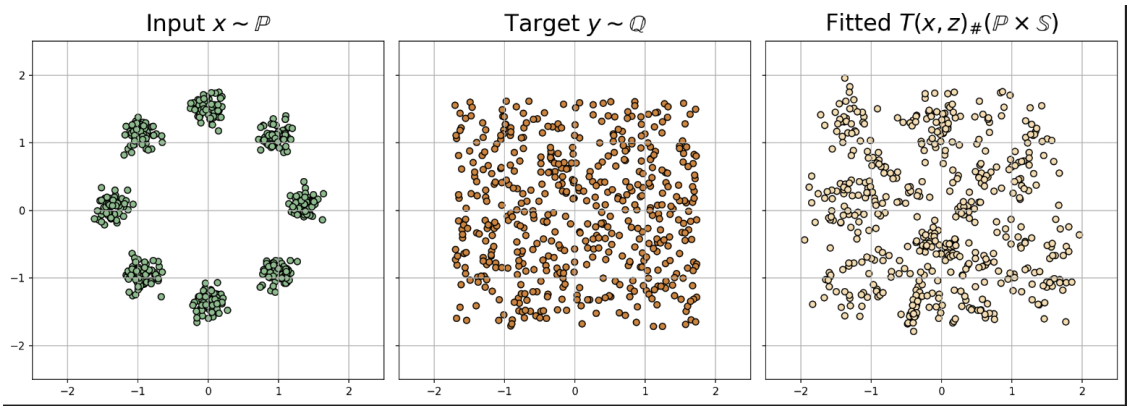


Figure 4: Stochastic maps $\hat{T}$ between $\mathbb{P}$ and $\mathbb{Q}$ with $\gamma = 1$ for 10k iterations

From 4 with $\gamma = 1$, we can observe that the model has not converged and it seems that the learned $\hat{T}$ is not a stochastic OT map.

To evaluate the impact of the γ parameter on this drawback, we also carry out experiments on $\gamma = 0.45$ and $\gamma = 0.1$.

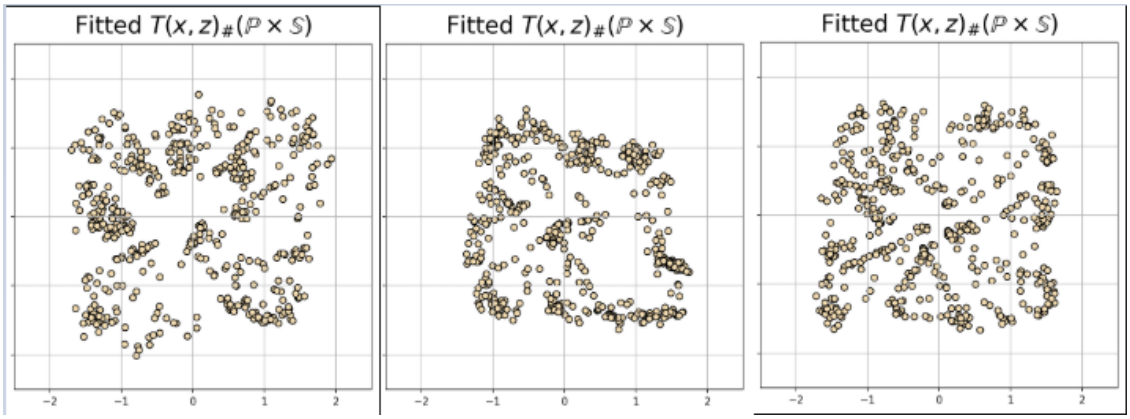


Figure 5: Stochastic maps $\hat{T}$ between $\mathbb{P}$ and $\mathbb{Q}$ with $\gamma = 0.4$ for respectively 9800, 10k and 10k200 iterations

The figure 5 shows that NOT algorithm is unstable and does not converge at all. This is one of the drawback stated by **Lemma 2** causd by the strictly non-convexity of our γ weak cost.

Finally, for a smaller $\gamma = 0.1$, NOT converged very quickly to a stochastic OT map around 3000 iterations approximately. We can observe the result in the figure 6 below.

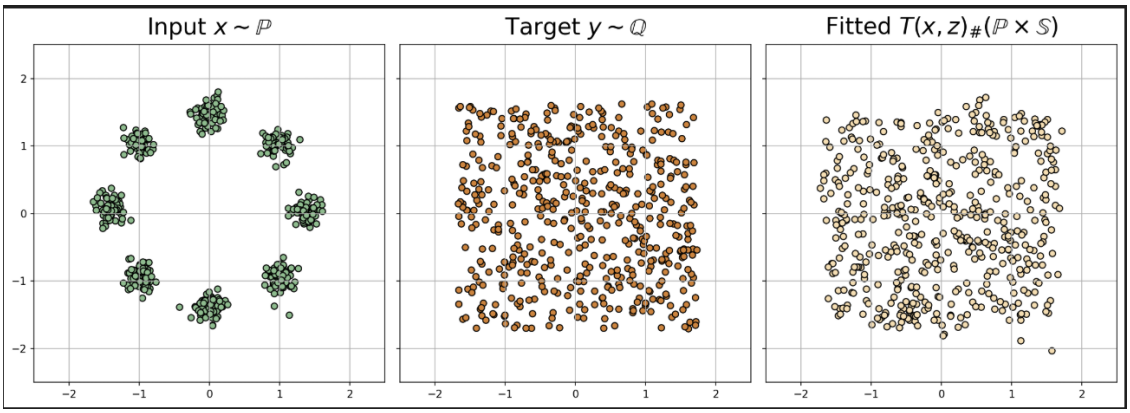


Figure 6: Stochastic maps $\hat{T}$ between $\mathbb{P}$ and $\mathbb{Q}$ with $\gamma = 0.1$ for 3800 iterations

4.3 Colorization of images

We wanted to tackle a problem that is still challenging : image colorization.

Given a grayscale image, we want to get a colorized version of it : this problem can naturally be stated as a OT problem: Given a set of grayscales images $\mathbb{R}^{H\times W}$ and a set of colorized images $\mathbb{R}^{H\times W\times 3}$ with respective probability distributions $\mathbb{P}$ and $\mathbb{Q}$, we want to minimize $Cost(\mathbb{P},\mathbb{Q})$.

4.3.1 Colorization of images

To facilitate the convergence, we've considered a very narrow subset of images: a dataset containing images of deserts coming from the dataset "Landscape Recognition | Image Dataset | 12k Images" . This choice was very important, since our algorithm has converged in approximatively

2000 steps, after an unstable beginning, thanks to the predictable set of colors in the dataset (Figure 7).

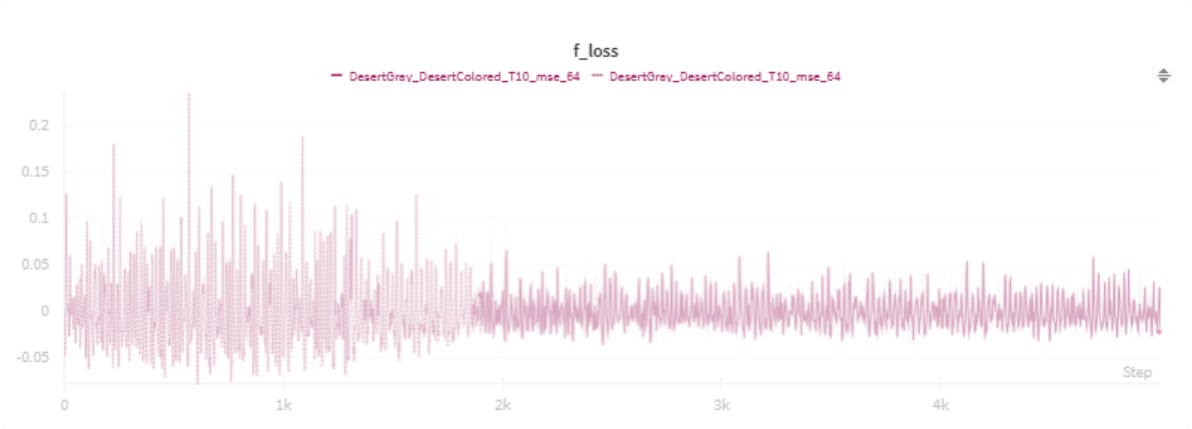


Figure 7: Evolution of the loss across 5000 steps

Figure 11 (shown in appendix B) show colorizations of images at different steps. Notice how the colorized images in figure 11a are decent, but lacks diversity. This is due to the fact we have a moving hyperparameter γ that goes from 0 at the beginning, to favour good transport, to 0.66 at the end to enable diversity in our generated images. The model can, however, be biased towards the most probable colors in the dataset (see the third row of the third column of Figure 11c where it colorized the upper part of the image in blue, but this issue the diversity brought.)

Figure 8 shows a comparison between our colorization and the real image. Since we do not have any metric that can quantify the quality of colorization, we have asked people to rate our colorization (in the same spirit as Mean opinion score), and they found that our colorized image is not only really qualitative, but seems also more natural than the ground truth ! But they also found that the colors bleed, notably the orange color of the desert that is spreading in a small area where there is originally the sky.

Figure 8: Test of our model on a desert taken from Unesco website

4.3.2 Stanford Dogs dataset

For this experiment, we wanted to colorize more complex images. We used the Stanford Dog Dataset and the Dog vs Cat Kaggle Dataset.

Details about the experiment can be found in Appendix. The results are displayed on Figure 9.

We observe that NOT has not converged, and that the algorithm have modified some parts of the

image like in the region above the dog in the third column.

This phenomenon aligns with our remark that solving the maximin problem doesn't necessarily provide optimal tranport maps. Furthermore, we noticed that the training was quite unstable which sometimes provides us very strange outputs. The algorithm sometimes changes parts of the initial dog image (See Appendix B.2). It may be due to the fact that we have allowed a too high γ during the training.

Figure 9: Colorization after 14k steps

5 Conclusion

We have seen that the neural optimal transport algorithm is an efficient tool for learning stochastic transport map which can be useful for image colorization.

However, NOT algorithm suffers from different issues such as its training instability and the choice of its parameters which can lead to stochastic non optimal transport maps.

The use of γ weak cost which is not strictly convex doesn't guarantee that solving the maximin formulation will yield a stochastic OT map. By lack of time due to the long training duration, we would have likek to use strict convexity costs to evaluate the obtained optimal stochastic transport maps.

References

[1] Grady Daniels, Tyler Maunu, and Paul Hand. Score-based generative neural networks for large-scale optimal transport. *Advances in Neural Information Processing Systems*, 34, 2021.

A Experiments on 2D Toy Dataset

You can find the notebook where we did our experiments in the code associated to this report. It is named : NOT_toy_2D.ipynb.

The details of the architecture can be found in figure below 10 where DIM is the dimension of the inputs $ZD = DIM$ is the dimension of the gaussian noise ($\sim \mathcal{N}(0, 0.1^2)$) that we add to the inputs. H is equal to 100 here.

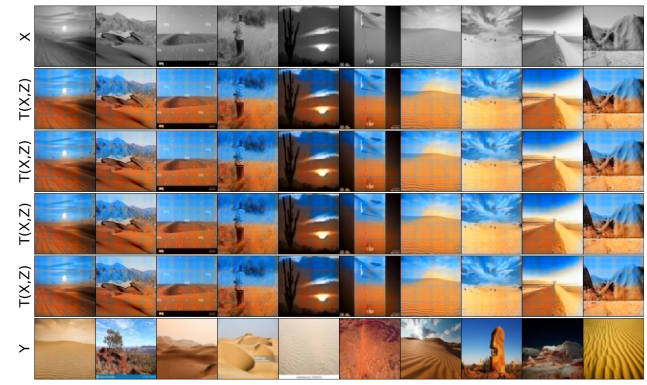
```
T = nn.Sequential(
    nn.Linear(DIM+ZD, H),
    nn.ReLU(True),
    nn.Linear(H, H),
    nn.ReLU(True),
    nn.Linear(H, H),
    nn.ReLU(True),
    nn.Linear(H, DIM)
).cuda()

f = nn.Sequential(
    nn.Linear(DIM, H),
    nn.ReLU(True),
    nn.Linear(H, H),
    nn.ReLU(True),
    nn.Linear(H, H),
    nn.ReLU(True),
    nn.Linear(H, 1)
).cuda()
```

Figure 10: Architecture details on toy dataset

B Colorization experiments

B.1 Deserts



(a) Step 300

(b) Step 2000

(c) Step 4800

Figure 11: Different colorization of the images across the steps

B.2 Dogs

For this, we have trained for 3 days on a single RTX 4060.

We run the code with a batch size of 32, a learning rate of 10^{-4} for both optimizers and with 20 iterations for T and 1 for f. Here, f is a ResNet and T is a UNet.

In total, we ran for 18000 epochs with a γ that linearly increases from 0 to 0.66 during the first 12500 epochs.

B.2.1 Unstable Training and weird outputs

Sometimes, the generated images are very strange as shown in the 7th and 8th columns of Figure 12 where we can't recognize dogs anymore or outputs with extravagant colors.

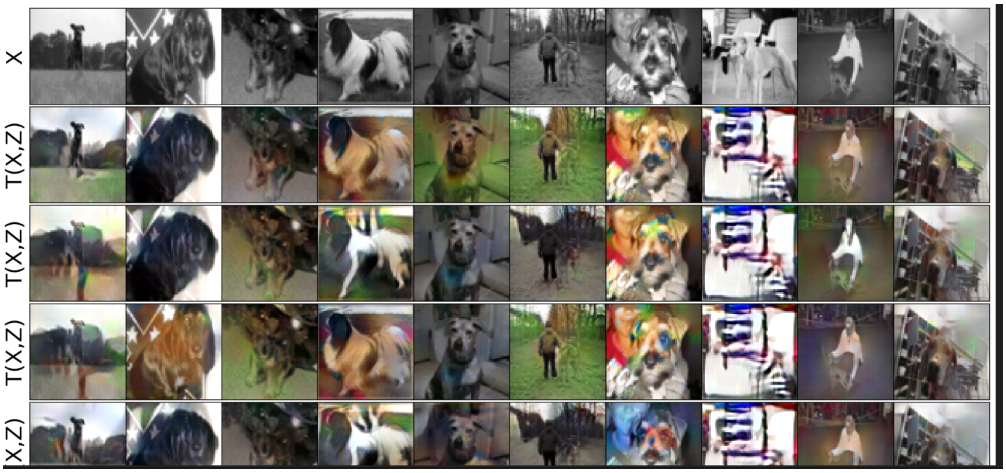


Figure 12: Weird outputs on the 16k steps approximately

B.3 Google Draw

We experiment NOT on the google draw dataset. The objective is to translate real cat images to cat images drawn by users of Google draw.

We have taken the cat dataset of google draw, and Cat dataset, we have trained a ResNet for the potential and a Unet for the transport. We have trained in the same setup than Desert experiment.

Figure 13 shows the results, the model gives very reasonnable outputs without preprocessing our dataset.

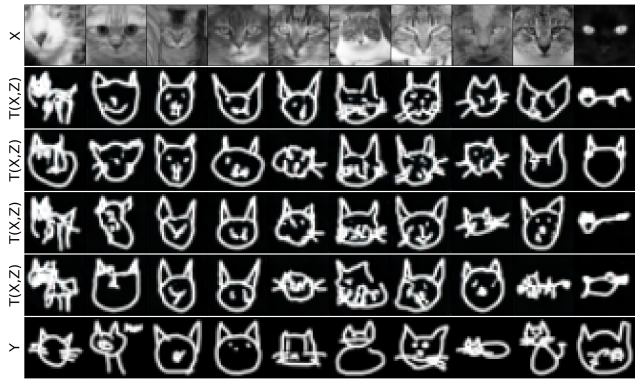


Figure 13: Generated images at step 4800 from random images