Neural optimal transport

Reproduced by: Ilya Sudakov

1 Problem Statement and Methodology Description

The Optimal Transport (OT) problem involves finding the most efficient way to transfer mass from one probability distribution to another. Below, we present the mathematical formulations of this problem.

Strong OT formulation

For $\mathbb{P} \in \mathcal{P}(\mathcal{X})$, $\mathbb{Q} \in \mathcal{P}(\mathcal{Y})$ transport cost function $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, Monge OT formulation described as follows:

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

where the minimum is taken over all functions $T : \mathcal{X} \to \mathcal{Y}$ that push forward $\mathbb{P}$ to $\mathbb{Q}$. The optimal function T^* is called the optimal transport map.

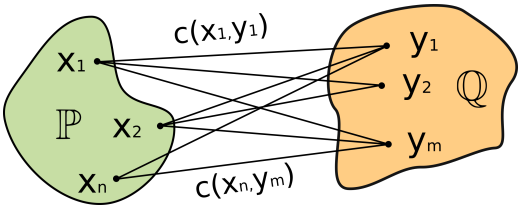

Figure 1: Strong OT.

Note that (1) does not allow a single point to be mapped to multiple points, which means that for some $\mathbb{P}, \mathbb{Q} \in \mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y})$, there may not exist a function T satisfying $T_{\#}\mathbb{P} = \mathbb{Q}$. Therefore, Kantorovich [1] proposed a more general formulation:

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

where the minimum is taken over all transport plans π (Figure 1), i.e., distributions on $\mathcal{X} \times \mathcal{Y}$ whose marginals are $\mathbb{P}$ and $\mathbb{Q}$. The optimal distribution $\pi^* \in \Pi(\mathbb{P}, \mathbb{Q})$ is called the optimal transport plan.

Weak OT formulation

The Kantorovich formulation [1] can be further generalized as follows:

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

where $\pi(\cdot|x)$ denotes the conditional distribution (Figure 2). Note that (3) indeed generalizes (2): if we take $C\big(x, \mu\big) = \int_{\mathcal{Y}} c(x, y) d\mu(y)$, then the weak formulation (3) reduces to the strong formulation (2). An example of a weak OT cost for $\mathcal{X} = \mathcal{Y} = \mathbb{R}^D$ is the γ-weak cost ($\gamma \geq 0$). Wasserstein-2 ($\mathcal{W}_{2,\gamma}$):

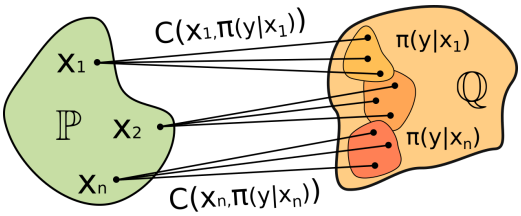

Figure 2: Weak OT.

$$C\big(x, \mu\big) = \int_{\mathcal{Y}} \frac{1}{2}\|x - y\|^2 d\mu(y) - \frac{\gamma}{2}\text{Var}(\mu) \tag{4}$$

Methodology

In the NOT paper [2], the authors, using non-trivial theoretical developments, reformulate the weak OT problem as an unconditional dual problem, for which the solution method is already well understood. Since the weak OT problem is the most general one, we will focus on how it can be solved. The solutions to other OT problems, which it generalizes, will then follow naturally.

Max-Min formulation of the dual problem: The following holds:

$$\text{Cost}(\mathbb{P}, \mathbb{Q}) = \sup_{f} \inf_{T} \mathcal{L}(f, T), \tag{5}$$

where the functional $\mathcal{L}$ is defined as

$$\mathcal{L}(f, T) \stackrel{def}{=} \int_{\mathcal{Y}} f(y) d\mathbb{Q}(y) + \int_{\mathcal{X}} \Big(C\big(x, T(x, \cdot)_{\#}\mathbb{S}\big) - \int_{\mathcal{Z}} f\big(T(x, z)\big) d\mathbb{S}(z) \Big) d\mathbb{P}(x). \tag{6}$$

Practical Algorithms

Algorithm 1

To solve the problem (5) in practice, two neural networks are introduced: $T_\theta : \mathbb{R}^P \times \mathbb{R}^S \to \mathbb{R}^Q$ and $f_\omega : \mathbb{R}^Q \to \mathbb{R}$ to parameterize T and f. Their parameters can be trained using batches of data from $\mathbb{P}, \mathbb{Q}, \mathbb{S}$; see Algorithm 1.

Algorithm 1: weak Neural optimal transport

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*;

Algorithm 1 requires the empirical computation of $\widehat{C}$ for $C\big(x, T(x,\cdot)_\#\mathbb{S}\big)$. If $C\big(x,\mu\big) = \int_{\mathcal{Y}} c(x,y)\, d\mu(y)$, one can use an unbiased Monte Carlo estimate over a batch $Z \sim \mathbb{S}$:

$$C\big(x, T(x,\cdot)_\#\mathbb{S}\big) = \int_{\mathcal{Z}} c(x, T(x,z)) d\mathbb{S}(z) \approx \frac{1}{|Z|}\sum_{z\in Z} c\big(x, T(x,z)\big) \stackrel{def}{=} \widehat{C}\big(x, T(x,Z)\big). \tag{7}$$

For the γ-weak quadratic cost (4), an unbiased Monte Carlo estimate is given by:

$$\widehat{C}\big(x, T(x,Z)\big) \stackrel{def}{=} \frac{1}{2|Z|}\sum_{z\in Z} \|x - T(x,z)\|^2 - \frac{\gamma}{2}\hat{\sigma}^2, \tag{8}$$

where $\hat{\sigma}^2$ is the variance estimated over the batch, $\hat{\sigma}^2 = \frac{1}{|Z|-1}\sum_{z\in Z}\Big\|T(x,z) - \frac{1}{|Z|}\sum_{z\in Z} T(x,z)\Big\|^2$. To estimate the γ-weak quadratic cost (8), it is required that $|Z| \geq 2$ in order to compute $\hat{\sigma}^2$.

The generation procedure will be very simple:

Algorithm 2: Sampling: weak Neural optimal transport

Input : mapping network $T_\theta : \mathbb{R}^P \times \mathbb{R}^S \to \mathbb{R}^Q$;
Sample $x \sim \mathbb{P}$; sample $z_x \sim \mathbb{S}$;
return $T_\theta(x, z_x)$;

Algorithm 2

If we do not require the ability to sample different images from a single input, then we can choose $C\big(x,\mu\big) = \int_{\mathcal{Y}} c(x,y)\, d\mu(y)$ as the cost. However, it can be observed that the solution to such a problem will coincide with a deterministic solution, meaning that the model will not take the variable Z into account. In this case, Algorithm 1 can be reformulated as Algorithm 3:

Analogously to 2, the generation procedure is given in 4:

Algorithm 3

The methods described above can be modified by training the discriminator to distinguish not only clean images but also noisy ones. In other words, the discriminator is conditioned on the noise level, and the corresponding noise is added to the image input. This encourages the generator to match the distribution of real images, similar to the idea used in DMD. The resulting procedure is given in Algorithm 5:

The generation algorithm does not differ from 4.

Algorithm 3: strong Neural optimal transport

Input : distributions $\mathbb{P}, \mathbb{Q}$ accessible by samples; mapping network $T_\theta : \mathbb{R}^P \to \mathbb{R}^Q$;
potential network $f_\omega : \mathbb{R}^Q \to \mathbb{R}$; number of inner iterations K_T;
(strong) cost $C : \mathcal{X} \times \mathcal{P}(\mathcal{Y}) \to \mathbb{R}$; empirical estimator $\widehat{C}\big(x, T(x)\big)$ for the cost;
Output: learned stochastic OT map T_θ representing an OT plan between distributions $\mathbb{P}, \mathbb{Q}$;
repeat
  Sample batches $Y \sim \mathbb{Q}$, $X \sim \mathbb{P}$;
  $\mathcal{L}_f \leftarrow \frac{1}{|X|}\sum_{x \in X} f_\omega\big(T_\theta(x)\big) - \frac{1}{|Y|}\sum_{y \in Y} f_\omega(y)$;
  Update ω by using $\frac{\partial \mathcal{L}_f}{\partial \omega}$;
  for $k_T = 1, 2, \ldots, K_T$ **do**
    Sample batch $X \sim \mathbb{P}$;
    $\mathcal{L}_T \leftarrow \frac{1}{|X|}\sum_{x \in X}\big[\widehat{C}\big(x, T_\theta(x)\big) - f_\omega\big(T_\theta(x)\big)\big]$;
    Update θ by using $\frac{\partial \mathcal{L}_T}{\partial \theta}$;
until *not converged*;

Algorithm 4: Sampling: strong Neural optimal transport

Input : mapping network $T_\theta : \mathbb{R}^P \times \mathbb{R}^S \to \mathbb{R}^Q$;
Sample $x \sim \mathbb{P}$;
return $T_\theta(x)$;

Algorithm 5: noised strong Neural optimal transport

Input : distributions $\mathbb{P}, \mathbb{Q}$ accessible by samples; mapping network $T_\theta : \mathbb{R}^P \to \mathbb{R}^Q$;
potential network $f_\omega : \mathbb{R}^Q \times \mathbb{R} \to \mathbb{R}$; number of inner iterations K_T;
(strong) cost $C : \mathcal{X} \times \mathcal{P}(\mathcal{Y}) \to \mathbb{R}$; empirical estimator $\widehat{C}\big(x, T(x)\big)$ 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, sample $Z \sim N(0, I), sigma \sim P_{sigma}$;
  $\mathcal{L}_f \leftarrow \frac{1}{|X|}\sum_{x \in X} f_\omega\big(T_\theta(x) + Z \cdot sigma, sigma\big) - \frac{1}{|Y|}\sum_{y \in Y} f_\omega(y + Z \cdot sigma, sigma)$;
  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, sample $Z \sim N(0, I), sigma \sim P_{sigma}$;
    $\mathcal{L}_T \leftarrow \frac{1}{|X|}\sum_{x \in X}\big[\widehat{C}\big(x, T_\theta(x)\big) - f_\omega\big(T_\theta(x) + Z \cdot sigma, sigma\big)\big]$;
    Update θ by using $\frac{\partial \mathcal{L}_T}{\partial \theta}$;
until *not converged*;

Intuitive Justification

These algorithms are based on the following intuitive reasoning: the function f is optimized to output high values for generated samples and low values for real images. In this way, minimizing this loss yields a discriminator. The function T is optimized so that, first, the output image is similar to the input image (and the variance of the output images is relatively high), and second, the discriminator outputs higher values for the generated samples. Thus, T aims to preserve the style of the input image while matching the distribution of real images. And this is precisely the goal in optimal transport.

2 Technical Description of the Experiments

Datasets

We chose to evaluate the proposed methods on the task of translating images of real women into anime-style images. The following datasets were used:

- **celeba-hq-dataset** (18K images + 1K for validation), available at this link
- **Aligned Anime Faces** (500K images), available at this link

Code for downloading, unpacking, and preprocessing both datasets was provided. The entire process took approximately 7 hours (2.5h + 2h + 2.5h), as the anime dataset is around 100 GB in size.

The preprocessing of the anime dataset involved cropping images to a size of (256, 256) at a vertical offset of 14 pixels above the center, followed by downscaling to a resolution of (128, 128).

All experiments were conducted on images with a resolution of either (128, 128) or (64, 64), so additional downscaling and normalization were applied during preprocessing to facilitate training.

Architecture

For the discriminator, we used a ResNet architecture from the official NOT repository, and a Unet architecture (also from the same repository) was used to parameterize the generator. Both models were relatively lightweight (92MB for f, 39MB for T). The neural network hyperparameters were also taken directly from the official implementation to avoid reproducibility issues under time constraints.

In Algorithm 3, to condition the discriminator on the noise level, we used the positional embedding implementation from the EDM repository along with the corresponding noise schedule.

To accelerate training, we employed AMP (Automatic Mixed Precision).

Metrics

To monitor training dynamics and model performance, we implemented a function to compute the FID score for generated samples. The FID was computed on 1000 samples (a relatively small number due to the limited dataset size) from the validation split of the celeba-hq dataset. Additionally, LPIPS and L2 metrics were used to measure the similarity between the output and the input images.

Algorithm 1 (Weak OT)

This method was trained with the following hyperparameters:

- T_ITERS = 10
- f_LR, T_LR = 2e-4, 2e-4
- IMG_SIZE = 64
- BATCH_SIZE = 128
- PLOT_INTERVAL = 100
- COST = 'weak_mse'
- CPKT_INTERVAL = 500
- ZC = 1
- Z_STD = 0.1
- Z_SIZE = 8
- GAMMA0, GAMMA1 = 0.0, 0.66
- GAMMA_ITERS = 500

The hyperparameter GAMMA_ITERS = 500 was set significantly lower than in the original paper due to limited training time.

Training was performed on Kaggle using two GPUs in parallel. Since images of size (128, 128) did not fit in memory even with a batch size of 64, we used images of size (64, 64), which allowed for a batch size of 128. Consequently, the learning rate was doubled compared to the original implementation.

As Kaggle limits continuous computation to 12 hours, training was done in two sessions using saved checkpoints of the models and optimizers from the first run. Considering that dataset loading took about 1.5 hours and the last checkpoint from the first session was not taken at the very end, the model was trained for a total of 22 hours on two Tesla T4 GPUs.

Over this period, 3400 training iterations were completed. We also provide the training loss dynamics for the generator (Figure 3) and the discriminator (Figure 4).

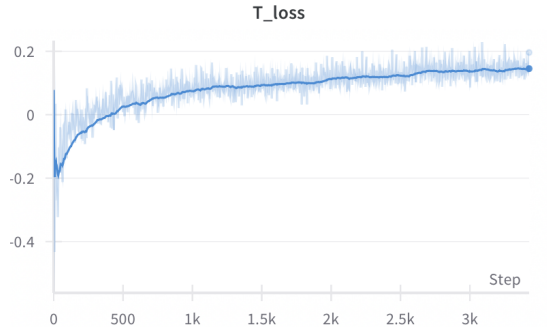

Figure 3: weak OT T loss

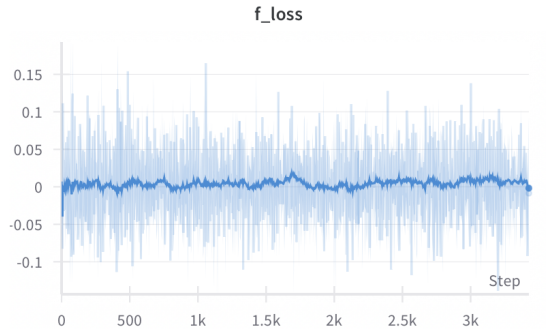

Figure 4: weak OT f loss

Algorithm 2 (Strong OT)

This method was trained with the following hyperparameters:

- T_ITERS = 10
- f_LR, T_LR = 2e-4, 2e-4
- IMG_SIZE = 128
- BATCH_SIZE = 128
- PLOT_INTERVAL = 100
- COST = 'mse'
- CPKT_INTERVAL = 1000

The model was trained on a single RTX 3060 GPU. The BATCH_SIZE = 128 was chosen as the maximum that could fit images of size (128, 128) in memory. The learning rate was also doubled, as a larger batch size was used.

After approximately 24 hours of training, the model losses began to explode, and the generator started behaving as if it was training from scratch (as observed in the generated images). Therefore, training was stopped at 5600 iterations.

We also provide the training loss dynamics for the generator (Figure 5) and the discriminator (Figure 6).

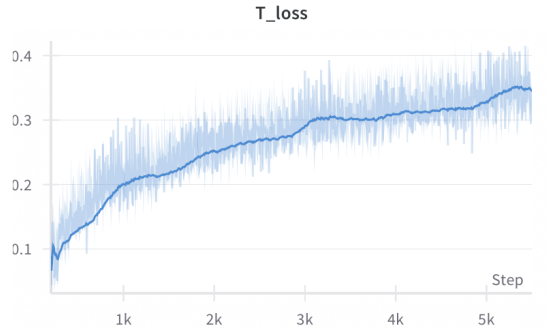

Figure 5: strong OT T loss

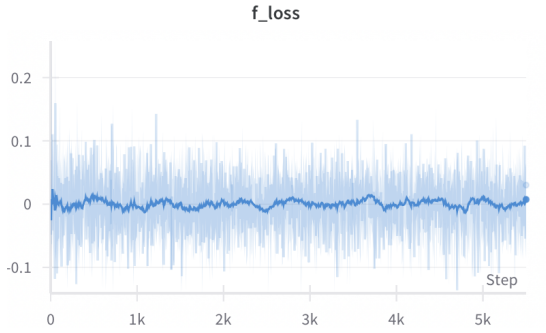

Figure 6: strong OT f loss

We also present in this section the dynamics of the evaluation metrics during training: FID (Figure 7), LPIPS (Figure 8), and L2 (Figure 9).

From these graphs, it can be observed that if training were continued, the FID score could potentially decrease further. The LPIPS values appear reasonable: at the beginning of training, the generator simply outputs the input image, and later it starts modifying it, which leads to an increase in LPIPS.

The L2 distance plot between input and output seems somewhat unusual. The value of 16384 appears to be the theoretical maximum L2 distance if the images lie in the compact set $[0, 1]$, but I was not able to identify the source of the issue so far.

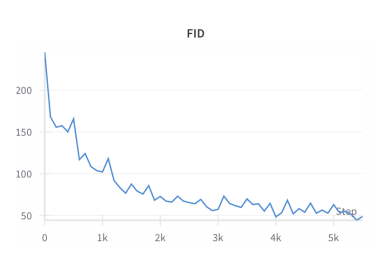

Figure 7: strong OT FID

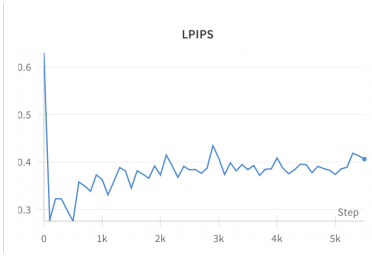

Figure 8: strong OT LPIPS

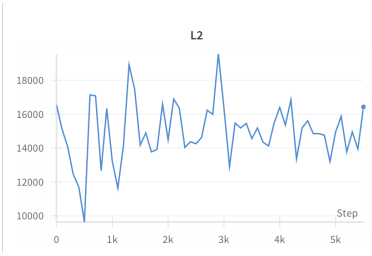

Figure 9: strong OT L2

Algorithm 3 (Noised Strong OT)

This method was trained with the following hyperparameters:

- T_ITERS = 10
- f_LR, T_LR = 2e-4, 2e-4
- IMG_SIZE = 128
- BATCH_SIZE = 128
- PLOT_INTERVAL = 50
- COST = 'mse'
- CPKT_INTERVAL = 1000
- sigma_min=0.02
- sigma_max=100

The choice of hyperparameters `sigma_min` and `sigma_max` follows the settings used in the EDM repository. The remaining hyperparameters were selected based on the same logic as in the previous experiments.

Algorithm 3 was trained for 36 hours, completing 6100 iterations. Regarding the training process, it only started working properly on the third attempt. Initially, I used an incorrect noise schedule. Then I tried removing the weighting factor that assigns importance to each noise level. In both cases, I trained for about 5 hours each, but the FID score barely decreased.

Eventually, I believe I found a bug in the code, after which training began to make progress. This suggests that training the noise-conditioned discriminator is quite sensitive to hyperparameters. Compared to the other methods, which quickly reached reasonable quality, this method converged much more slowly (luckily, I believed in it and didn't stop the training).

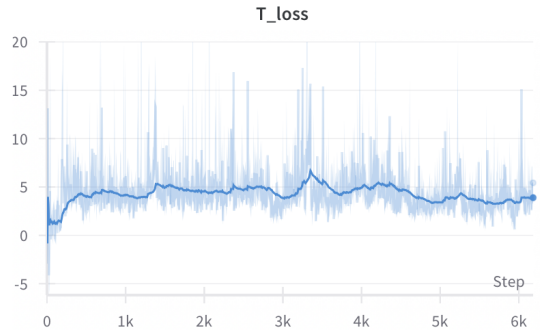

Figure 10: noised strong OT T loss

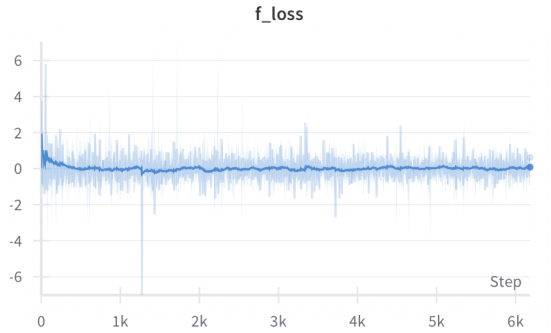

Figure 11: noised strong OT f loss

The generator loss dynamics differ noticeably from the previous case: there, the loss increased over time, while here it remains approximately constant.

The dynamics of the evaluation metrics during training are shown above in Figures 12, 13, and 14. These plots demonstrate that the model is indeed learning (as the FID decreases and LPIPS does not increase significantly), but at the same time, convergence is much slower compared to the previous method. In the earlier case, the final FID score was around 50, whereas here it fluctuates heavily around 100.

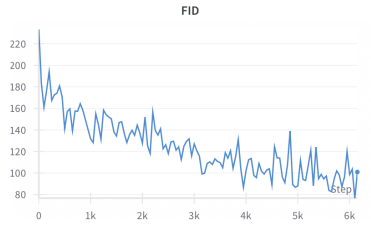

Figure 12: noised strong OT FID

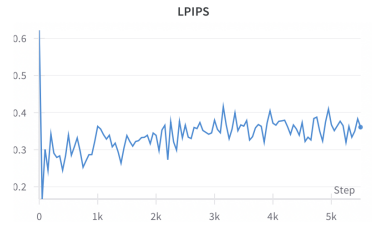

Figure 13: noised strong OT LPIPS

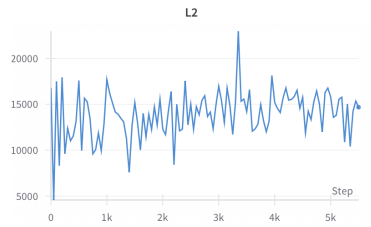

Figure 14: noised strong OT L2

3 Results

Algorithm 1

Despite the relatively short training time, the generated samples in Figure 15 show promising results. Unfortunately, the image resolution is not very high, but visually we can observe that the samples resemble anime-style outputs. Moreover, different outputs are produced for the same input image—most frequently varying in eyes, facial expressions, hairstyles, and hair colors. Since the generation in other experiments was performed at a resolution of 128x128, it is not possible to directly compare the generation quality here.

Figure 15: Weak OT samples

Algorithm 2

In the experiment with the deterministic generator, the resulting images (Figure 16) are, in my opinion, already quite good. While they are generally less detailed than reference images, the style is very similar. It should be noted that the examples shown here were randomly selected before training in order to observe how they change during training. Thus, if the goal had been to find the most aesthetically pleasing generations, better examples could have been selected.

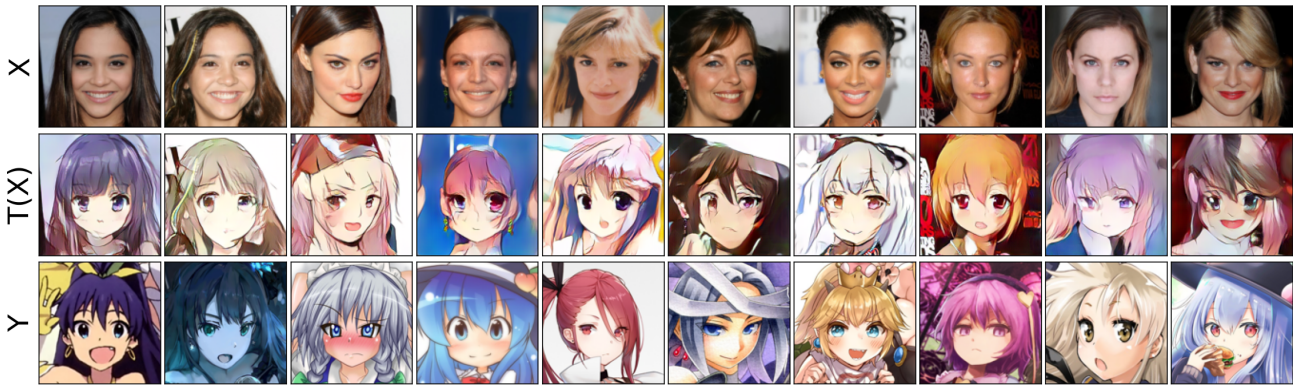

Figure 16: Strong OT samples

Algorithm 3

Again, the generator produced good samples (Figure 17). Upon closer inspection, the hair color in these images appears to match the input image more closely than in the non-noised version, which tended to struggle with that aspect. Additionally, the hair appears more detailed. In other aspects, it's hard to say which model produces better images, though the FID score computed on 1000 samples surprisingly favors the non-noised method.

Figure 17: Noised Strong OT samples

4 Method Applicability: Pros and Cons

Throughout the experiments, it was found that the method solving the strong OT problem worked immediately, required relatively little memory for effective training, and ultimately produced high-quality samples. Therefore, it can be considered the first method to try when tackling a style transfer task.

However, if one aims to solve a one-to-many generation problem, technical difficulties begin to arise. In particular, the batch size must be scaled proportionally to the number of samples generated from the conditional distribution, thus requiring significantly more memory.

The third method, based on NOT, only started working on the third attempt and still required a very long training time. Therefore, its practical utility is currently questionable, especially when the regular strong OT method is already available and performs well.

https://wandb.ai/sudakov/diffusion-NOT?nw=nwuserilyasudakov
https://github.com/iasudakov/NOT

References

[1] Leonid Kantorovitch. On the translocation of masses. *Management Science*, 5(1):1–4, 1958.

[2] Alexander Korotin, Daniil Selikhanovych, and Evgeny Burnaev. Neural optimal transport, 2023.