

Нейронный оптимальный транспорт

Выполнили: Илья Судаков, Антон Мосин

1 Постановка задачи и описание методологии

Задача оптимального транспорта (ОТ) заключается в нахождении наиболее эффективного способа перенести массу из одного вероятностного распределения в другое. Приведем математические формулировки данной задачи.

Strong OT формулировка

Для $\mathbb{P} \in \mathcal{P}(\mathcal{X})$, $\mathbb{Q} \in \mathcal{P}(\mathcal{Y})$ и функции транспортной цены $c : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, формулировка ОТ Монжа следующая:

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

где минимум взят по всем функциям $T : \mathcal{X} \rightarrow \mathcal{Y}$ которые отображают \mathbb{P} в \mathbb{Q} . Оптимальная функция T^* называется оптимальным транспортным отображением.

Отметим, что (1) не позволяет переводить одну точку в несколько, а значит, для каких-то $\mathbb{P}, \mathbb{Q} \in \mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y})$, может не быть T удовлетворяющего $T \# \mathbb{P} = \mathbb{Q}$. Поэтому, Канторович [1] предложил более обобщенную формулировку:

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

где минимум взят по всем транспортным планам π (Figure 1), то есть таким распределениям на $\mathcal{X} \times \mathcal{Y}$ у которых маргинальные распределения \mathbb{P} и \mathbb{Q} . Оптимальное распределение $\pi^* \in \Pi(\mathbb{P}, \mathbb{Q})$ называется оптимальным транспортным планом.

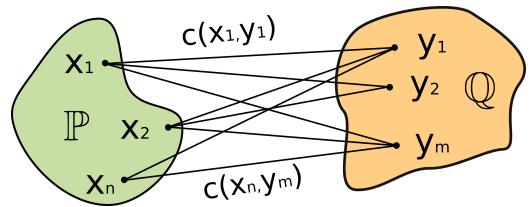


Рис. 1: Strong OT.
При этом Канторович [1] предложил более обобщенную формулировку:

Weak OT формулировка

Формулировку Канторовича [1] можно еще сильнее обобщить следующим образом:

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

где $\pi(\cdot|x)$ обозначает условное распределение (Рисунок 2).

Заметим что (3) действительно обобщает (2), если взять $C(x, \mu) = \int_{\mathcal{Y}} c(x, y) d\mu(y)$, то weak формулировка (3) становится strong (2). Примером weak OT цены для $\mathcal{X} = \mathcal{Y} = \mathbb{R}^D$ является γ -weak ($\gamma \geq 0$) Wasserstein-2 ($\mathcal{W}_{2,\gamma}$):

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

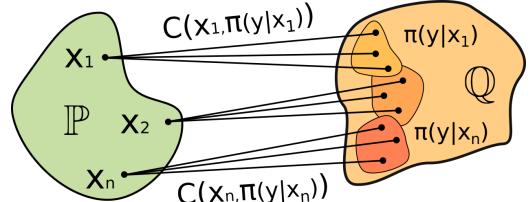


Рис. 2: Weak OT.

Методология

В статье NOT [2], авторы используя нетривиальные теоретические выкладки, переводят задачу weak OT в безусловную задачу двойственную к ней, у которой уже понятно как искать решение. Так как задача weak OT является наиболее общей, то рассмотрим как можно решать ее, а другие задачи, которые она обобщает будут уже понятно как решать.

МаксМин формулировка двойственной задачи: Верно следующее:

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

где функционал \mathcal{L} определен как

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

Практические алгоритмы

Алгоритм 1

Чтобы решить задачу (5) на практике, вводятся две нейросети $T_\theta : \mathbb{R}^P \times \mathbb{R}^S \rightarrow \mathbb{R}^Q$ и $f_\omega : \mathbb{R}^Q \rightarrow \mathbb{R}$ чтобы параметризовать T и f . Учить их параметры можно используя батчи данных из $\mathbb{P}, \mathbb{Q}, \mathbb{S}$, см. алгоритм 1.

Algorithm 1: weak Neural optimal transport

Input : distributions $\mathbb{P}, \mathbb{Q}, \mathbb{S}$ accessible by samples; mapping network $T_\theta : \mathbb{R}^P \times \mathbb{R}^S \rightarrow \mathbb{R}^Q$;
 potential network $f_\omega : \mathbb{R}^Q \rightarrow \mathbb{R}$; number of inner iterations K_T ;
 (weak) cost $C : \mathcal{X} \times \mathcal{P}(\mathcal{Y}) \rightarrow \mathbb{R}$; empirical estimator $\hat{C}(x, T(x, Z))$ 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 $Z_x \sim \mathbb{S}$;
 $\mathcal{L}_f \leftarrow \frac{1}{|X|} \sum_{x \in X} \frac{1}{|Z_x|} \sum_{z \in Z_x} f_\omega(T_\theta(x, z)) - \frac{1}{|Y|} \sum_{y \in Y} f_\omega(y)$;
 Update ω by using $\frac{\partial \mathcal{L}_f}{\partial \omega}$;
for $k_T = 1, 2, \dots, K_T$ **do**
 | Sample batch $X \sim \mathbb{P}$; for each $x \in X$ sample batch $Z_x \sim \mathbb{S}$;
 | $\mathcal{L}_T \leftarrow \frac{1}{|X|} \sum_{x \in X} [\hat{C}(x, T_\theta(x, Z_x)) - \frac{1}{|Z_x|} \sum_{z \in Z_x} f_\omega(T_\theta(x, z))]$;
 | Update θ by using $\frac{\partial \mathcal{L}_T}{\partial \theta}$;
until not converged;

Алгоритм 1 требует эмпирического вычисления \hat{C} для $C(x, T(x, \cdot) \# \mathbb{S})$. Если $C(x, \mu) = \int_Y c(x, y) d\mu(y)$, можно использовать несмешенную Monte-Carlo оценку по батчу $Z \sim \mathbb{S}$:

$$C(x, T(x, \cdot) \# \mathbb{S}) = \int_Z c(x, T(x, z)) d\mathbb{S}(z) \approx \frac{1}{|Z|} \sum_{z \in Z} c(x, T(x, z)) \stackrel{\text{def}}{=} \hat{C}(x, T(x, Z)). \quad (7)$$

Для γ -weak квадратичной цены (4), несмешенной Monte-Carlo оценкой будет:

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

где $\hat{\sigma}^2$ дисперсия оцененная по батчу $\hat{\sigma}^2 = \frac{1}{|Z|-1} \sum_{z \in Z} \|T(x, z) - \frac{1}{|Z|} \sum_{z \in Z} T(x, z)\|^2$. Чтобы оценить γ -weak квадратичную стоимость (8), нужно $|Z| \geq 2$ чтобы оценить $\hat{\sigma}^2$.

Процедура генерации будет выглядеть очень просто:

Algorithm 2: Sampling: weak Neural optimal transport

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

Алгоритм 2

Если мы не хотим уметь семплировать разные картинки из одного входа, то в качестве C можно взять $C(x, \mu) = \int_Y c(x, y) d\mu(y)$, но можно заметить что решение такой задачи будет совпадать с детерминированным решением, а значит модель никак не будет учитывать переменную Z . Тогда алгоритм 1 можно переписать в алгоритм 3:

Аналогично 2, процедура генерации следующая 4:

Алгоритм 3

Приведенные выше методы можно модифицировать, если заставить дискриминатор различать не только чистые картинки, но и зашумленные. То есть обусловить дискриминатор на уровень шума, а к картинке на вход прибавлять соответствующий шум. Это будет стимулировать генератор попадать в распределение настоящих картинок аналогично идее использованной в DMD. Получится следующий алгоритм 5:

Алгоритм генерации не будет отличаться от 4.

Algorithm 3: strong Neural optimal transport

Input : distributions \mathbb{P}, \mathbb{Q} accessible by samples; mapping network $T_\theta : \mathbb{R}^P \rightarrow \mathbb{R}^Q$;
potential network $f_\omega : \mathbb{R}^Q \rightarrow \mathbb{R}$; number of inner iterations K_T ;
(strong) cost $C : \mathcal{X} \times \mathcal{P}(\mathcal{Y}) \rightarrow \mathbb{R}$; empirical estimator $\hat{C}(x, T(x))$ for the cost;
Output: learned stochastic OT map T_θ representing an OT plan between distributions \mathbb{P}, \mathbb{Q} ;
repeat

Sample batches $Y \sim \mathbb{Q}$, $X \sim \mathbb{P}$;
 $\mathcal{L}_f \leftarrow \frac{1}{|X|} \sum_{x \in X} f_\omega(T_\theta(x)) - \frac{1}{|Y|} \sum_{y \in Y} f_\omega(y)$;
Update ω by using $\frac{\partial \mathcal{L}_f}{\partial \omega}$;
for $k_T = 1, 2, \dots, K_T$ **do**
 Sample batch $X \sim \mathbb{P}$;
 $\mathcal{L}_T \leftarrow \frac{1}{|X|} \sum_{x \in X} [\hat{C}(x, T_\theta(x)) - f_\omega(T_\theta(x))]$;
 Update θ by using $\frac{\partial \mathcal{L}_T}{\partial \theta}$;

until not converged;

Algorithm 4: Sampling: strong Neural optimal transport

Input : mapping network $T_\theta : \mathbb{R}^P \times \mathbb{R}^S \rightarrow \mathbb{R}^Q$;
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 $T_\theta : \mathbb{R}^P \rightarrow \mathbb{R}^Q$;
potential network $f_\omega : \mathbb{R}^Q \times \mathbb{R} \rightarrow \mathbb{R}$; number of inner iterations K_T ;
(strong) cost $C : \mathcal{X} \times \mathcal{P}(\mathcal{Y}) \rightarrow \mathbb{R}$; empirical estimator $\hat{C}(x, T(x))$ 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 $Z \sim N(0, I)$, $sigma \sim P_{sigma}$;
 $\mathcal{L}_f \leftarrow \frac{1}{|X|} \sum_{x \in X} f_\omega(T_\theta(x) + Z \cdot sigma, sigma) - \frac{1}{|Y|} \sum_{y \in Y} f_\omega(y + Z \cdot sigma, sigma)$;
Update ω by using $\frac{\partial \mathcal{L}_f}{\partial \omega}$;
for $k_T = 1, 2, \dots, K_T$ **do**
 Sample batch $X \sim \mathbb{P}$; For each X , sample $Z \sim N(0, I)$, $sigma \sim P_{sigma}$;
 $\mathcal{L}_T \leftarrow \frac{1}{|X|} \sum_{x \in X} [\hat{C}(x, T_\theta(x)) - f_\omega(T_\theta(x) + Z \cdot sigma, sigma)]$;
 Update θ by using $\frac{\partial \mathcal{L}_T}{\partial \theta}$;

until not converged;

Интуитивное обоснование

Данные алгоритмы имеют следующее интуитивное обоснование: f оптимизируется так, чтобы выдавать большое значение сгенерированным семплам и меленькое значение настоящим картинкам, таким образом минимизируя этот лосс, будет получаться дискриминатор. T оптимизируется так, чтобы во первых картинка на входе была похожа на картинку на выходе, + (дисперсия выходных картинок была побольше), во вторых чтобы дискриминатор выдавал число побольше сгенерированным объектам, таким образом T будет пытаться сохранять стиль исходной картинки и при этом попадать в нужное распределение настоящих картинок. А это как раз то что нужно в оптимальном транспорте.

2 Техническое описание поставленных экспериментов

Датасеты

Было решено проверять работу представленных методов на переводе из картинок женщин в картинки аниме женщин. Поэтому использовались следующие датасеты:

- celeba-hq-dataset (18K картинок + 1K валидация) доступный по ссылке

- Aligned Anime Faces (500K картинок) доступный по ссылке

Для обоих датасетов предоставлен код для их скачивания, распаковки и препроцессинга. Сам этот процесс занял примерно 7ч (2.5ч + 2ч + 2.5ч), так как датасет с аниме весит 100 гигабайт.

Препроцессинг для датасета аниме заключался в обрезании до размера (256, 256) с уровня на 14 пикселей выше центра, а затем картинки сжимались до разрешения (128, 128).

Далее эксперименты ставились на картинках в разрешении либо (128, 128) либо (64, 64), поэтому также для препроцессинга картинки еще сжимались и нормализовались для лучшего обучения.

Архитектура

Для параметризации дискриминатора использовалась архитектура ResNet взятая из репозитория к статье NOT, оттуда же была взата архитектура Unet для параметризации генератора. Обе архитектуры весили не так много (92Мб для f, 39Мб для T). Гиперпараметры нейросетей были так же взяты как в официальной имплементации, чтобы не возникало проблем с воспроизведением за ограниченное время.

В алгоритме 3, чтобы обусловить дискриминатор на уровень шума, была взята имплементация позиционного эмбеддинга из репозитория EDM, а так же расписание уровня шума.

Так же для более быстрого обучения, использовалась AMP

Метрики

Для наблюдения за динамикой обучения и качеством работы модели, были имплементированы функция для подсчета FID сгенерированных семплов, который вычислялся по 1000 (не так много, так как сам датасет не очень большой) картинкам из валидационного множества датасета celeba-hq. Так же для наблюдения за похожестью выхода на вход были рассчитаны LPIPS и L2 метрики.

Алгоритм 1 (Weak OT)

Данный метод обучался со следующими гиперпараметрами:

- 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

Гиперпараметр GAMMA_ITERS = 500 был выбран сильно меньше чем у авторов, из-за меньшего времени обучения. Обучение делалось на Kaggle, используя 2 видеокарты параллельно, так как картинки (128, 128) не помещались даже в батче = 64, то были использованы картинки (64, 64), при этом они влезли по памяти с батчем 128, поэтому лернинг рейт был увеличен в 2 раза по сравнению с авторской имплементацией. Так как Kaggle позволяет делать вычисления только 12 часов без перерыва, то это делалось в 2 захода, используя сохраненные чекпоинты моделей и оптимайзеров с первого обучения. С учетом того, что датасет подгружался примерно 1.5 часа и последний чекпоинт первого обучения был сделан не в самом конце, то суммарно модель обучалась 22 часа, на двух видеокартах Tesla T4. За это время прошло 3400 итераций обучения. Так же приведем динамику лоссов генератора 3 и дискриминатора 4 в процессе обучения.

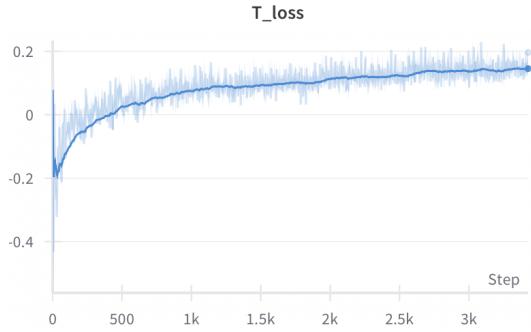


Рис. 3: weak OT T loss

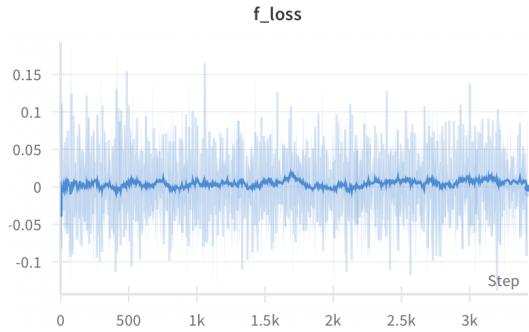


Рис. 4: weak OT f loss

Алгоритм 2 (Strong OT)

Данный метод обучался со следующими гиперпараметрами:

- 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

Обучение модели проводилось на одной RTX 3060. BATCH_SIZE = 128 был подобран как максимальный вмещающий картинки 128 на 128, лернинг рейт был так же увеличен в 2 раза, так как взяли побольше батч. После примерно 24 часов такого обучения, лосс моделей начал взрываться и модель начала обучаться будто заново (судя по сгенерированным картинкам), поэтому обучение было завершено на 5600 итерациях. Так же приведем динамику лоссов генератора 5 и дискриминатора 6 в процессе обучения.

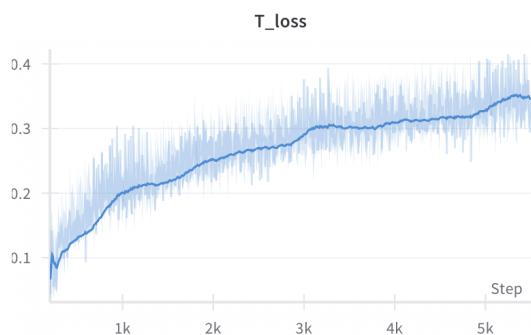


Рис. 5: strong OT T loss

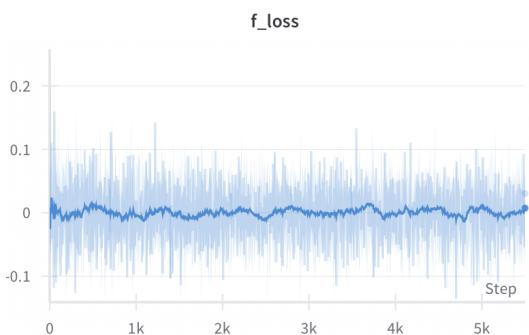


Рис. 6: strong OT f loss

Так же приведем в этой секции динамику метрик 7, 8, 9 во время обучения. По этим графикам видно, что если еще поучить, то FID может еще упасть немножко. Значение LPIPS кажутся правильными, в начале обучения генератор просто выдает вход, а затем уже как-то его модифицирует и LPIPS растет. График L2 разности входа и выхода кажется немного странным, ведь значение 16384 это по идее максимальное значение L2 если картинки лежат в компакте [0, 1], но ошибку я не смог найти пока.

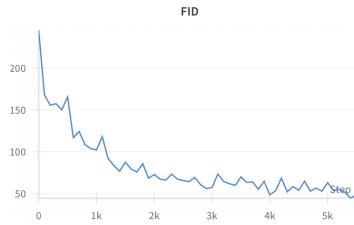


Рис. 7: strong OT FID



Рис. 8: strong OT LPIPS

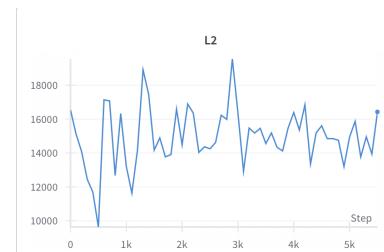


Рис. 9: strong OT L2

Алгоритм 3 (Noised Strong OT)

Данный метод обучался с гиперпараметрами:

- 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

Выбор гиперпараметров σ_{\min} и σ_{\max} такой же как в репозитории EDM. Остальные гиперпараметры выбирались из той же логики что и в других экспериментах.

Алгоритм 3 обучался 36 часов, 6100 итераций. Про его обучение можно сказать, что он завелся только с 3 раза у меня, в начале я взял не совсем правильное расписание уровня шума, затем я попробовал выкинуть множитель, который каждому уровню шума дает вес, пообучав часов 5 каждую такую попытку, FID почти не падал. Затем я кажется нашел ошибку в коде и обучение заработало, поэтому можно сказать что процедура обучения шумного дискриминатора чувствительна в гиперпараметрам. Так же по сравнению с другими методами, которые достаточно быстро приходят к какому-то нормальному качеству, этот метод сходился очень долго (повезло что я в него верил и не выключил).

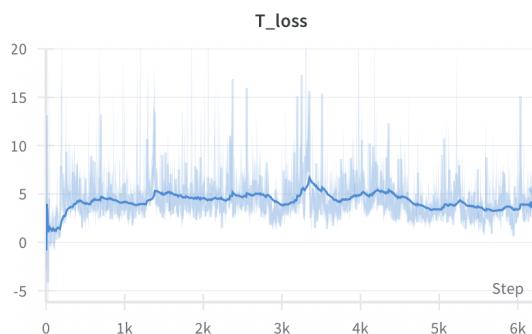


Рис. 10: noised strong OT T loss

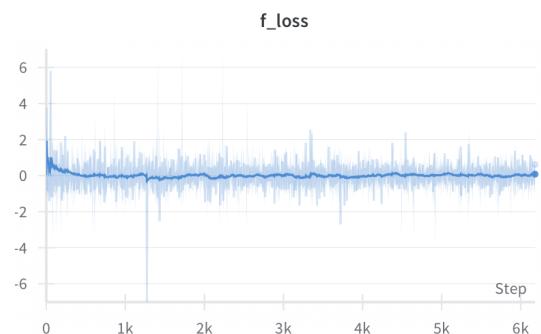


Рис. 11: noised strong OT f loss

Динамика лосса генератора уже заметно отличается от предыдущей, там лосс росс со временем, а тут остается примерно постоянным.

Выше приведена динамика метрик 12, 13, 14 во время обучения. По ней можно как раз проанализировать что модель права учится (так как FID падает и LPIPS не сильно растет), но при этом это обучение сходится сильно медленнее предыдущего (там в конце FID порядка 50, а тут колеблется сильно у 100).

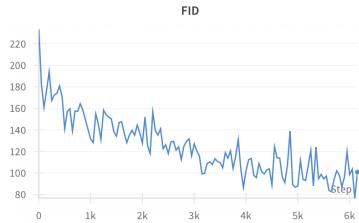


Рис. 12: noised strong OT FID



Рис. 13: noised strong OT LPIPS

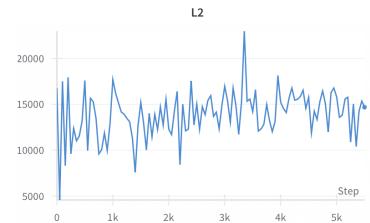


Рис. 14: noised strong OT L2

3 Полученные результаты

Алгоритм 1

За небольшое время обучения, получились следующие картинки 15. К сожалению разрешение картинок не очень высокое, но качественно можно понять, что генерируются достаточно похожие на аниме семплы, при этом для одной и той же картинки получаются разные семплы: чаще всего у картинок различаются глаза, мимика, прическа и ее цвет. Так как в других экспериментах генерация была выполнена на разрешении 128 на 128, то сравнить качество генерации не получится.

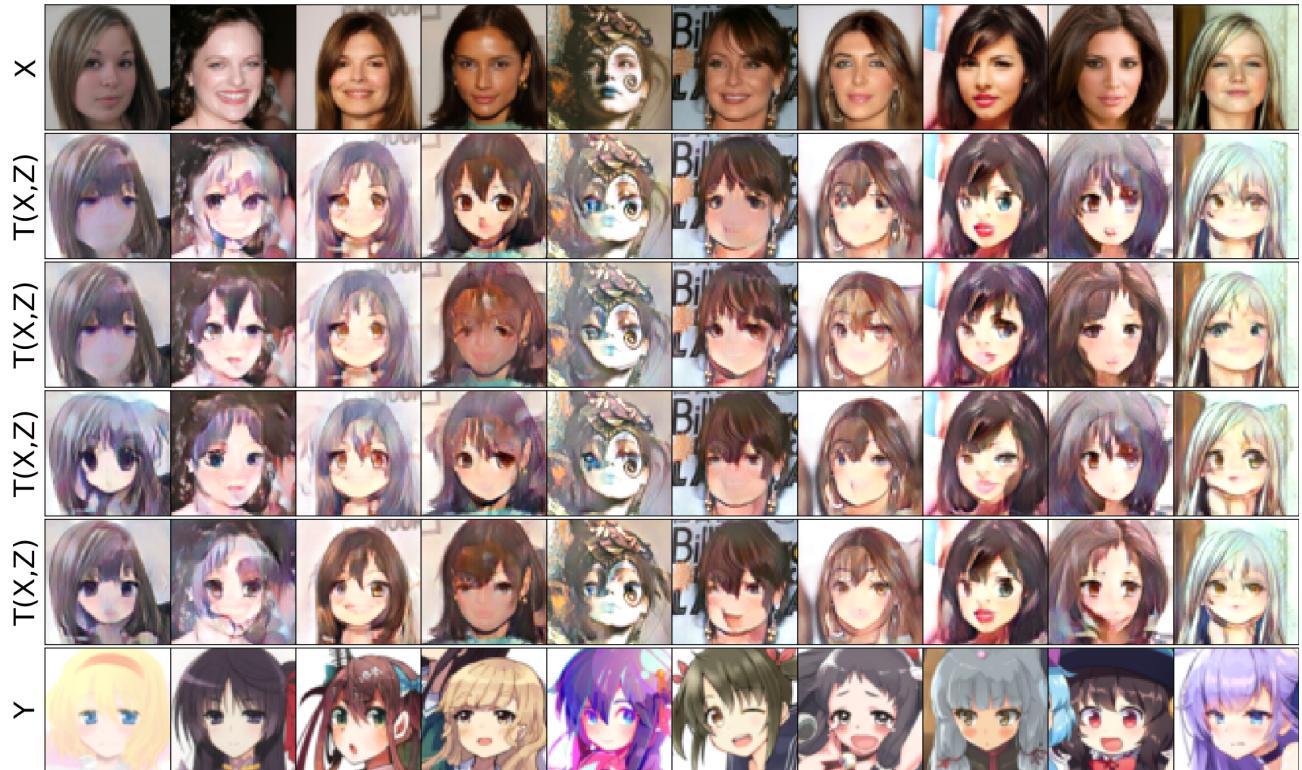


Рис. 15: weak OT samples

Алгоритм 2

В эксперименте с детерминированным генератором, уже получились вполне хорошие на мой взгляд картинки 16. Они скорее просто не такие детализированные как референсные, но по стилю очень похожи. Отмету что тут приведены произвольные примеры генерации, которые были выбраны до обучения, чтобы по ним смотреть как они будут изменяться, поэтому если бы была цель найти красивые генерации, то нашлись бы и лучше.



Рис. 16: strong OT samples

Алгоритм 3

Опять же получились хорошие семплы из генератора 17. Если приглядеться то на этих картинках цвет волос более похож на цвет из исходной картинки, чем у не шумного алгоритма с этим немного проблемы. Так же волосы получились более детализированы. В остальном сложно сказать у кого картинки лучше, но FID оцененный по 1000 семплам считает что лучше у не шумного метода почему-то.



Рис. 17: noised strong OT samples

4 Применимость метода, его плюсы и минусы

В ходе экспериментов, выяснилось что метод решающий задачу strong OT, сразу завелся, достаточно мало требует памяти чтобы эффективно его обучать и в итоге получил хорошие семплы. Поэтому можно считать что это первое что стоит запускать если хочется решать задачу переноса стиля.

Если же хочется решать задачу one2many, то тут уже возникают технические трудности, так как размер батча кратно умножается на число картинок сгенерированных из условного распределения, поэтому необходимо уже больше памяти.

Третий метод основанный на NOT завелся только с третьего раза, при этом учится все равно очень долго. Поэтому смысл его применения пока не сильно оправдан, когда есть обычный метод strong OT.

5 Распределение задач

Илья Судаков:

- имплементация методов strong и noised strong OT
- имплементация функций для подсчета LPIPS, L2, FID
- проведение экспериментов над strong и noised strong OT
- написание отчета

Антон Мосин:

- загрузка и подготовка датасетов
- имплементация метода weak ОТ
- проведение эксперимента над weak ОТ

Полезные ссылки

<https://wandb.ai/sudakov/diffusion-NOT?nw=nwuserilyasudakov>

<https://github.com/sudakovcom/NOT>

Список литературы

- [1] Leonid Kantorovitch. On the translocation of masses. *Management Science*, 5(1):1–4, 1958.
- [2] Alexander Korotin, Daniil Selikhanovich, and Evgeny Burnaev. Neural optimal transport, 2023.