# Unpaired style-transfer with diffusion models by distilling discrete batched entropy optinal transport (EOT) (Machine Learning 2023 Course)

**Nikita Andreev** [1] **Kseniia Kiseleva** [1] **Vasily Tesalin** [1] **Artem Ostrovsky** [1] **Emiliya Novikova** [1] **Nikita Gushchin** [1]

## Abstract

Unpaired style transfer is a challenging problem in image processing that involves transferring the style of one image to another while preserving the content. Recently, diffusion models have been proposed as a promising approach for this task. In this project, we propose a method for unpaired style transfer with diffusion models by distilling discrete batched entropy optimal transport (EOT).

**Github repo:** link
**Video presentation:** link

## 1. Introduction

The project aims to study an unpaired style transfer, a task that involves generating an image with the style of one image and the content of another. This is a complicated task, especially when the style and content images are not paired, as it requires learning the underlying style and content distributions.

The proposed solution in the project is to use diffusion models, a class of generative models that can learn high-dimensional distributions (Chitwan Saharia, 2022). To improve the performance of the diffusion models, the project uses discrete batched entropy optimal transport (EOT) to distill the models (Cuturi, 2013).

The motivation for the project is to advance the state-of-the-art approaches in unpaired style transfer, which have many practical applications in image editing and synthesis, as well as in artistic expression. By improving the performance of diffusion models through EOT distillation, the project aims to make unpaired style transfer more accessible and efficient, leading to new and exciting possibilities for image creation and manipulation. Ultimately, the project seeks to contribute to the broader field of computer vision and deep learning, advancing our understanding of generative models and their applications.

The main goal of the project is to implement conditional diffusion probabilistic model by using samples from mini-batch entoropic optimal transport as a paired data.

The project involves specific tasks. Here is a list of them:

- Find any ready public code for the conditional diffusion probabilistic model (DPM) and make it work.
- Add mini batch calculation from python optimal transport (OT) inside of a custom dataloader for the main model.
- Choose image datasets for unpaired style transfer.
- Train DPM with mini-batch OT and measure sample quality (SSIM, PSNR).

In this project, we conducted 3 main experiments:

- With colorized MNIST dataset and regularized optimal transport.
- With colorized MNIST dataset and unregularized optimal transport
- With Shoes and Handbags datasets and unregularized optimal transport

## 2. Preliminaries

### 2.1. Diffusion Probabilistic model

Denoising Diffusion Probabilistic models (DDPMs) are a class of deep generative models that use diffusion-based dynamics to model the distribution of high-dimensional data. In a DDPM, the model is defined as a diffusion process that transforms an initial noise distribution into the target data distribution over a series of time steps. The process involves iteratively adding noise to the data and then using a learned denoiser function to remove the added noise. See Fig. 1.

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Kseniia Kiseleva <kseniia.kiseleva@skoltech.ru>.

*Final Projects of the Machine Learning 2023 Course*, Skoltech, Moscow, Russian Federation, 2023.

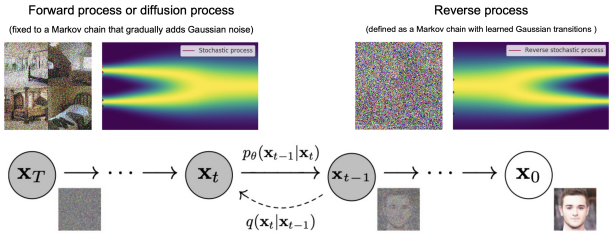


*Figure 1.* Forward and reverse diffusion process, $p$ and $q$ are conditional translations (Song, 2021)

#### 2.1.1. Connection to SDE

Recall that with a finite number of noise scales, we can generate samples by reversing the perturbation process. For infinite noise scales, we can analogously reverse the perturbation process for sample generation by using the reverse Stochastic Differential Equation (SDE).

Solving a reverse SDE yields a score-based generative model. Transforming data to a simple noise distribution can be accomplished with an SDE. It can be reversed to generate samples from noise if we know the score of the distribution at each intermediate time step.

In general, SDE can be written in this form:

$$dx = f(x,t)dt + g(t)dw,$$

where $f$ is a vector-valued function called the drift coefficient, $g$ is a real-valued function called the diffusion coefficient and $w$ is a standart Brownian motion. Fig. 2 illustrates that.

#### 2.1.2. Conditional vs unconditional diffusion models

Conditional and unconditional diffusion models are two types of stochastic models used to describe the dynamics of a variable over time. The key difference between these two types of models is the presence or absence of conditioning variables.

In an unconditional diffusion model, the dynamics of the variable of interest are described solely by a stochastic process, typically a continuous-time stochastic process such as a Brownian motion. The model assumes that the variable follows a certain distribution and that its evolution is random and independent of any other variables.

In contrast, a conditional diffusion model takes into account the effect of other variables on the dynamics of the variable of interest. The model specifies a conditional distribution for the variable of interest, given the values of the conditioning variables. In this case we need to diffuse both the input $x$ and condition $y$, and then learn $\nabla logp(x|y)$.

In our work we use conditional model to implement Style Transfer approach, where style image is given as a condition for translation.

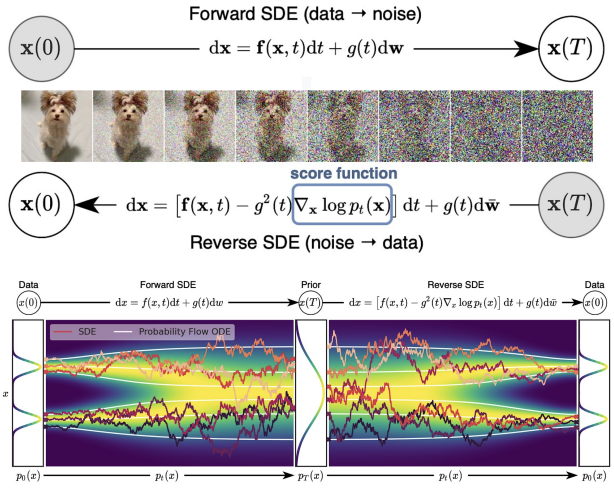


*Figure 2.* Mapping data to a noise distribution (the prior) with an SDE, and reverse this SDE for generative modeling (Song, 2021)

It's also worth to say that parameterized Markov chain trained using variation inference to produce samples matching the data after finite time. Transitions of this chain are learned to reverse a diffusion process, which is a Markov chain that gradually adds noise to the data in the opposite direction of sampling until signal is destroyed.

### 2.2. Optimal Transport

Now let's look at task of transition between distributions. In such a problems our main goal is to compute minimal cost of transportation between the source and the target as it is shown on Fig. 3.

In this work we decided to look at Image-to-image translation, where image from a source domain transported to a target domain. This translation paths can be learned in both Paired (when fit with prepared pairs) or Unpaired manner (when need to create pairs, by computing the distance between them). You can see schematic illustration of this concept on Fig. 4.

In many cases the choice of Unpaired manner is guided by the difficulty of getting Paired type of dataset.

So, in our work we use Discrete Optimal Transport for preparing data pairs from the each batch for the Diffusion model. This allows us to create sufficient data for training encouraging diffusion model from various unpaired datasets

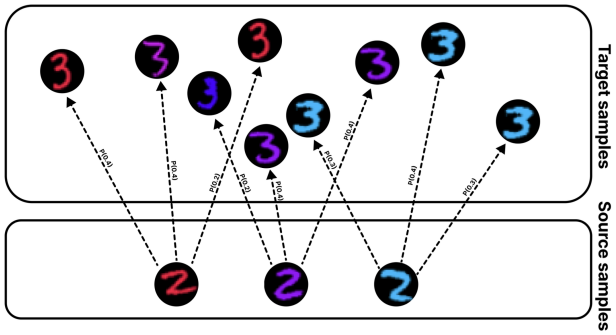


*Figure 3.* Discrete optimal transport scheme with probability weights.

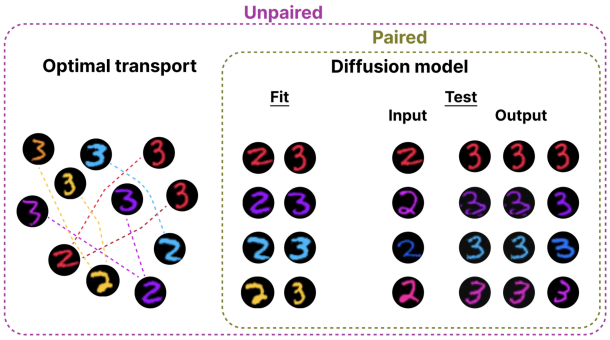


*Figure 4.* Common intercourse between Diffusion model and Optimal Transport.

which are quite common and easy accessible.

## 3. Related Work

### 3.1. Palette architecture

In this paper we use Palette (Chitwan Saharia, 2022), a simple framework for image-to-image translation based on conditional diffusion models.

It was tested and give good results in different tasks: inpainting, colorization and uncropping.

This model uses a U-Net architecture (Jonathan Ho & Abbeel, 2020) with some modifications. The network architecture is based on the 256×256 class-conditional U-Net model of (Jonathan Ho & Abbeel, 2021). The two main differences between this architecture and Palette are :

(i) absence of class-conditioning

(ii) additional conditioning of the source image via concatenation, following (Chitwan Saharia & Norouzi, 2021).

Unlike many Generative Adversarial Network models, Palette produces diverse and high fidelity outputs as you can see on examples on Fig. 5.


*Figure 5.* Palette diversity for inpainting, colorization, and uncropping tasks (Chitwan Saharia, 2022).

### 3.2. Optimal transport and Sinkhorn distance

For the optimal transportation problem where we were given a $d \times d$ cost matrix $M$ ($d$ is a dimention of simplex elements over $r$ and $c$). The cost of mapping $r$ to $c$ using a transportation matrix (or joint probability) $P$ can be quantified as:

$$d_M(r, c) = min_{P \in U(r,c)} <P, M>$$

in which we denote $U(r, c)$ the transportation polytope of r and c.

It is shown in (Cuturi, 2013) that regularizing with an intuitive entropic penalty (Sinkhorn distance is parameterized by a regularization weight) gives us a good solution for this problem with many applications at the intersection of optimal transportation theory and machine learning.

So, parameterized distance will have such a form:

$$d_{M,\alpha}(r, c) = min_{P \in U_\alpha(r,c)} <P, M>$$

Schematic view of the transportation polytope and the Kullback-Leibler (KL) ball of level $\alpha$ that surrounds the independence table, as an illustration of this idea is provided in Fig. 6.

In our work we used distributed under the MIT licence optimal transport toolbox written in Python which was presented and explained here (Flamary et al., 2021).

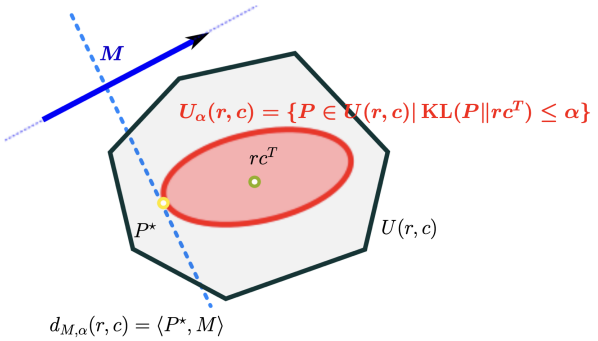


*Figure 6.* The Sinkhorn distance is the dot product of M with the optimal transportation table in that ball (Cuturi, 2013).

## 4. Datasets

It is important to first discuss the datasets used in this work before delving into the details of the implemented models and algorithms. In this project, three datasets were utilized: MNIST(Deng, 2012), UT Zappos50K, and Handbags dataset. Further UT Zappos50K dataset will be called Shoes dataset.

### 4.1. MNIST

For the needs of training MNIST dataset was modernised. First, images were set to size 32x32 and pixels were normalized to be in the range [-1,1]. Next, digits were colored in one of 6 random colors. For the sake of simplicity only twos and threes were picked for model training and evaluation. Below is the description of how dataset was colorized.

For colorization procedure we used *get_random_colored_images* function that could be found in training-palette-with-optimal-transport/notebooks/mnistregularized-ot.ipynb in our GitHub repository. This function takes in a set of images and a seed value as input. The function first sets the NumPy random seed to the given seed value. It then applies a normalization step to the input images by scaling them to a range of [-0.5, 0.5]. Next, the function computes the number of images in the input set and initializes an empty list to store the colored images.

The function then generates a random hue value for each image between 0 and 360 degrees. It uses the hue value to convert each input image to a colored image with a randomly generated color. This is done by converting the input image from the RGB color space to the HSV color space, changing the hue value while keeping the saturation and value channels constant, and then converting back to the RGB color space.

The conversion is implemented using a loop that iterates through each image and hue value. For each image, it extracts its value channel and calculates the corresponding hue channel based on the randomly generated hue value. It then uses the hue channel to compute the red, green, and blue channels of the colored image based on the hue value. Finally, the function appends each colored image to the list of colored images.

After processing all the images, the function converts the list of colored images to a tensor and applies a normalization step to scale them to the range of [-1, 1]. The function returns the colored images as a tensor. The PyTorch library is used for manipulating the tensors.

### 4.2. Shoes

Shoes dataset is a large-scale dataset of shoe images created by the University of Texas at Austin and Zappos.com. It consists of 50,000 images of shoes, with each image being of size 128 x 128 pixels. For the project needs images were set to size 32x32 and pixels were normalized to be in the range [-1,1]. The images are divided into two categories: men's shoes and women's shoes. The shoes are roughly centered, but not well aligned, and roughly facing left, with frontal to side view.

Each image in the dataset is labeled with a text description of the shoe, including information such as brand, color, category, and style. The dataset also includes additional information, such as the price of the shoe and whether it is currently in stock on the Zappos website.

The dataset contains a total of 10 categorical features, including brand, category, and subcategory, as well as 4 real-valued features, such as price and rating. The categorical features are represented as text strings, while the real-valued features are represented as floating-point numbers.

One of the key features of the dataset is the ability to use the text descriptions of the shoes to perform natural language processing (NLP) tasks, such as text classification and sentiment analysis. The dataset also provides a valuable resource for computer vision tasks, such as image classification and object detection, due to the large number of high-quality shoe images.

The dataset was downloaded from iGAN project GitHub repository(Zhu)

### 4.3. Handbags

Handbags dataset is a large-scale dataset consisting of 137,000 images of handbags downloaded from Amazon. For the project needs images were set to size 32x32 and pixels were normalized to be in the range [-1,1]. The downloaded handbags are roughly centered but no further alignment has

been performed.

This dataset was obtained from Amazon.com's product images. The authors of the paper(Zhu et al., 2016) downloaded the images using Amazon's Product Advertising API. Specifically, they searched for "handbags" and filtered the results to include only images of handbags.

It's worth noting that while the dataset was obtained from Amazon, it was not created or curated by Amazon. Instead, it was assembled by the authors of the paper for their research purposes. Other specifics are not stated in the original paper.

The dataset was downloaded from iGAN project GitHub repository(Zhu)

## 5. Model

Initially, we began by utilizing an existing GitHub repository containing a diffusion model (LouisRouss) and refined it for the paired colorization task of MNIST digits, transitioning them from black and white to colored images. To achieve this, we extensively modified the code from the aforementioned repository, incorporated logs to wandb and established a customized dataloader for the colored MNIST dataset. The validation process was the primary issue with the existing code, as inappropriate values for alphas and betas were selected, resulting in poor generation. To address this problem, new values have been determined: 1000 betas were generated between the range of $10^{-6}$ and 0.01, and alphas were calculated as 1 minus the value of each beta.

To create pairs for training, we utilized optimal transport to construct the collate function using the pythonot library (Flamary et al., 2021). For this purpose both regularized and unregularized versions of optimal transport were implemented.

Subsequently, we trained the model for generating handbags based on input shoe images using Shoes and Handbags datasets. In this phase, we also introduced a custom collate function and dataloader.

All parameters we utilized were the same as in the original repository, as altering them could risk improper learning. The only exception was setting the batch size to the maximum available with our resources, 64. All the parameters are specified in the config of our code which could be accessed through the link to project's GitHub repo.

## 6. Experiments and results

It is important to note that all experiments were performed on Kaggle's GPU Tesla P100 before discussing the obtained results. The experiments were divided into two parts: the first was conducted using colorized MNIST dataset, the second using Shoes and Handbags datasets. All logs (including losses and more frequent visualization) are available on **WandB**: link.

### 6.1. MNIST experiment

Firstly, the model was tested on the colorized MNIST dataset discussed in 4.1 to ensure proper functioning. The objective was to train a model that could generate a three digit based on the input two digit. Additionally, this generation had to have style-transfer so that the color of the generated three digit would match the color of the input two digit. During training at each batch optimal transport was employed to surjectively match each two with a three, and diffusion model was trained on obtained digits pairs. The training and test sets were split using the built-in torch split for the MNIST dataset.

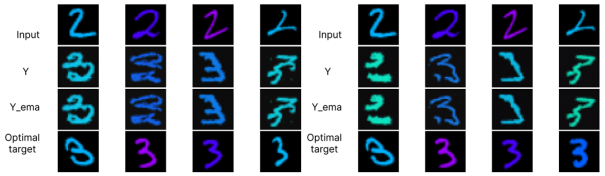


*Figure 7.* Results obtained on the colorized MNIST dataset after 7000 step, utilizing two variations of optimal transport - unregularized (left picture) and regularized (right picture).

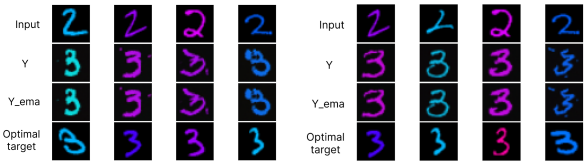


*Figure 8.* Results obtained on the colorized MNIST dataset after 15000 step, utilizing two variations of optimal transport - unregularized (left picture) and regularized (right picture).

In the Fig. 7, you can observe the results obtained after 7000 steps on our data, utilizing two variations of optimal transport - regularized and unregularized. For each input image we 3 outputs. First is generation of model (Y), second is generation with exponential averaging (Y_ema) and third is the optimal pair from the batch according to the predicates of optimal transport (optimal target). For generation with exponential averaging we averaged the weights of several models from different steps. The results are not much better than the basic generation, but this is a popular approach which was tested in the project.

Although some three-digit numbers are distinguishable, they

do not meet desired level of accuracy. However, after doubling the number of steps, we achieved better results, as demonstrated in the Fig. 8. In most cases, style transfer was evident, with the color of the generated image matching that of the input in both cases. Additionally, there was minimal variation between the regularized and unregularized versions.

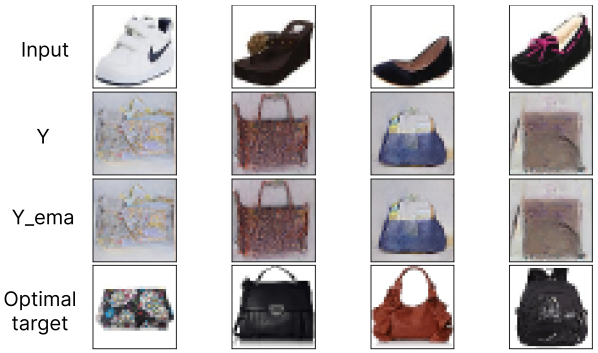


*Figure 9.* Results obtained after 15000 step, utilizing unregularized variation of optimal transport.

### 6.2. Shoes-Handbags experiment

Subsequently, we conducted experiments on more challenging data, namely the Shoes and Handbags datasets. The objective was to train a model that could generate a handbag based on the input shoe. Additionally, this generation should have style-transfer so that the style of the generated handbag would match the style of the input shoe. During training at each batch optimal transport was employed to surjectively match each shoe with a handbag, and diffusion model was trained on obtained shoe-handbag pairs. Due to resource constraints, we randomly selected 20% of the data for training and 1000 random samples for testing. After 15000 steps, which correspond to roughly 12 hours of training, we obtained the results shown in the Fig. 9. Although the generation quality is not as desired, and style transfer is not evident, it should be noted that these datasets are more complex than MNIST, and further GPU hours are required to enhance our model.

In total, we conducted three extensive training runs, each taking approximately 10-12 hours, along with multiple smaller ones for debugging and model validation. At every experiment the model was trained from scratch without using any pre-trained checkpoints. All experiments were conducted on Kaggle, and we utilized wandb to examine logs and regularly save the model's checkpoints. These logs and checkpoints can be obtained by the link.

In summary, we obtained decent results on the MNIST dataset, explored various optimal transport versions and achieved good generation results on shoe-handbags dataset without style transfer.

## 7. Metrics

To evaluate the final performance and model comparison we use several common metrics - SSIM and PSNR in table 1.

SSIM metric (et al, 2010) is used for measuring the similarity between two images using formula:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

where $\mu$ and $\sigma$ are the pixel sample mean and the variance of corresponding images respectively, and $c_1, c_2$ are some constants.

The PSNR metric (Alain Horé, 2010) is a widely used method for measuring the reconstruction quality of images, which involves calculating the Mean Squared Error ($MSE$) between two images using following formula:

$$PSNR = 10 \cdot log_{10}(\frac{MAX_I^2}{MSE}),$$

where $MAX_I$ is the maximum possible pixel value among the image. å

A higher SSIM value indicates a greater similarity while lower PSNR means better reconstruction.

Using these metrics allows us to compare various models. For instance, when analyzing the results of regularized OT on MNIST, we observe only a minor difference in the fourth digit, indicating that the results are comparable. In addition, we note that using EMA does not lead to improvements. Meanwhile, for another dataset, we observe that better results are achieved, suggesting superior generation quality. It's important to note that while these metrics are useful for measuring certain aspects of image generation, they do not enable us to evaluate style-transfer quality and must be assessed visually.

## 8. Conclusion

This paper demonstrates the potential of unpaired style-transfer with diffusion models, achieved by distilling discrete batched EOT, in the field of image processing. The model effectively performs style transfer when working with the colorized MNIST dataset. Furthermore, the model was trained multiple times using the Shoes and Handbags datasets, showing good results without style transfer. However, learning on this datasets requires more GPU hours as

| Data | OT version | model | SSIM (↑) | PSNR (↓) |
|---|---|---|---|---|
| MNIST | regularized | regular | **0.1035** | **13.3252** |
| MNIST | regularized | EMA | **0.1035** | **13.3252** |
| MNIST | unregularized | regular | 0.1009 | 13.4127 |
| MNIST | unregularized | EMA | 0.1009 | 13.4126 |
| Shoes-Bags | unregularized | regular | 0.1430 | 9.5900 |
| Shoes-Bags | unregularized | EMA | 0.1430 | **9.5899** |

*Table 1.* Metrics for main experiments.

they are generally more complex than the MNIST dataset. Lastly, the colorized MNIST dataset contains repeated images with identical colors for certain numbers, which is not the case for the Shoes and Handbags datasets.

Here are some of the approaches we are considering for the future development of the project:

- Continue training of the model using Shoes and Handbags datasets choosing another set of parameters.
- Improve the architecture and explore different style-transfer techniques.
- Attempt to train the model on a dataset that includes both high and low resolution images of people's faces.
- Submit extended version of this report to a machine learning conference.

## References


Alain Horé, D. Z. Image quality metrics: Psnr vs. ssim. 2010. URL Imagequalitymetrics:PSNRvs.SSIM.

Chitwan Saharia, William Chan, H. C. C. A. L. J. H. T. S. D. J. F. M. N. Palette: Image-to-image diffusion models. *arXiv preprint arXiv:2111.05826v2*, 2022.

Chitwan Saharia, Jonathan Ho, W. C. T. S. D. J. F. and Norouzi, M. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transportation distances. *arXiv preprint arXiv:1306.0895v1*, 2013.

Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

et al, P. N. Ssim image quality metric for denoised images. 2010. URL https://www.researchgate.net/publication/2621575666.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

Jonathan Ho, A. J. and Abbeel, P. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

Jonathan Ho, A. J. and Abbeel, P. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021.

LouisRouss. Diffusion-based-model-for-colorization. https://github.com/LouisRouss/Diffusion-Based-Model-for-Colorization.

Song, Y. Generative modeling by estimating gradients of the data distribution. 2021. URL https://yang-song.net/blog/2021/score/.

Zhu, J.-Y. igan: Interactive image generation via generative adversarial networks. https://github.com/junyanz/iGAN/blob/master/train_dcgan/README.md.

Zhu, J.-Y., Krähenbühl, P., Shechtman, E., and Efros, A. A. Generative visual manipulation on the natural image manifold. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pp. 597–613. Springer, 2016.

## A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

**Nikita Andreev (30% of work)**

- Coding the main algorithm
- Adding logging for wandb
- Preparing the GitHub Repo
- Reviewing the report

**Kseniia Kiseleva (25% of work)**

- Reviewing literate on the topic (3 papers)
- Team Management
- Presentation preparation
- Preparing the Abstract, Introduction and Conclusion sections of this report.

**Vasily Tesalin (15% of work)**

- Responsible for the report
- Reviewing literate on the topic (3 papers)
- Preparing the Datasets; Model; Experiments and results sections of this report.

**Artem Ostrovsky (15% of work)**

- Preparing the Preliminaries and Related Work sections of this report.
- Reviewing literate on the topic (5 papers)
- Preparing the GitHub Repo

**Emiliya Novikova (15% of work)**

- Reviewing literate on the topic (2 papers)
- Presentation preparation

## B. Reproducibility checklist

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** Used ready-made code for diffusion (but it was with errors, so we spent a lot of time fixing it first) + used ready-made optimal transport from the pythonot library.

   What we did ourselves: debugged the code for diffusion, added a bunch of all kinds of logs to wandb, loaded a colored dataset, wrote custom datasets for a colored mnist and a custom collate-fn (collate function) using optimal transport (different types - with and without regularization) so that in the dataloader then pairs were created for training with its help, then we changed collate-fn and a custom dataset for another dataset (shoes and bags).

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** **Github repo:** link

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** We took the MNIST (size 32x32), deuces and triples. Then they were randomly painted in different colors, after that they were divided into pairs using the optimal transport (that is, for each deuce in the batch, looked for the optimal triple from this batch) and such a pairs go to the diffusion model.

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** **Github repo:** link

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** Take MNIST (which is reduced to a size of $32 \times 32$ and normalize the pixels to be in the range $[-1, 1]$) and leave only deuces and triples from it, then randomly color them.

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** There is a standard split into a train and a test for the MNIST in the PyTorch - we loaded everything like that.

   For bags, we took a random $20\%$ of the data for training, since we already do not have enough computational resources, and another 1k random samples for the test.

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   **Students' comment:** The parameters were taken as in the original diffusion repository (because otherwise there is a risk that everything will not be learned as it should), only the batch size was set to the maximum possible with our resources - 64, all parameters are in the config in our code.

9. The exact number of evaluation runs is included.

   ☑ Yes.
   ☐ No.

- ☐ Not applicable.

**Students' comment:** Total 3 large training runs each about 10-12 hours and many more small ones for model checks and debugging.

10. A description of how experiments have been conducted is included.
    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    **Students' comment:** None

11. A clear definition of the specific measure or statistics used to report results is included in the report.
    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    **Students' comment:** None

12. Clearly defined error bars are included in the report.
    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    **Students' comment:** None

13. A description of the computing infrastructure used is included in the report.
    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    **Students' comment:** Experiments were done using Kaggle GPU TeslaP100.

## C. Train results


*Figure 10.* Noise loss results on train.

Even after we have learned another 15k steps, it is still possible to retrain because the noise loss is steadily decreasing on both datasets.