On scalable and efficient computation of large scale optimal transport (Machine Learning 2020 Course)

Vladimir Dmitriev [1] Alexander Shumilov [1] Veronika Zorina [1] Daria Gazizova [1] Elizaveta Noskova [1]

Abstract

Optimal transport is one of the popular methods used in ML, e.g. for domain adaptation or sample generation. But due to the heavy computing load, this method is difficult to use everywhere. To solve this problem, the SPOT algorithm was described, the operation of which was tested in this project. SPOT (Scalable Push-forward of Optimal Transport) is an implicit generative learning-based framework by which the optimal transport problem is casted into a minimax problem and then is efficiently solved by primal dual stochastic gradient-type algorithms. In this work SPOT was tested on Gaussian distribution toy dataset and used for domain adaptation and image generation.

1. Introduction

Optimal transport is one of the popular methods used in ML. Such ML problems like domain adaptation (Ganin & Lempitsky, 2014) or sample generation (Santambrogio, 2010) could be formulated as optimal transport problem.

1.1. Optimal Transport

Central idea of optimal transport problem is following: assume that there are two distributions defined on spaces $\mathcal{S}^1$ and $\mathcal{S}^2$ and cost functional, which is just a function of two variables defined on samples from this distributions. Then the problem is to find such joint distribution that will minimize expectation of this functional and satisfy following condition: marginal distribution with respect to each variable will coincide with given distributions of this variable. Nature of cost functional affects meaning of resulting distribution. This main concept of feasible classical optimal transport problem was formulated by Kantorovich.

$$\gamma^* = argmin_{\gamma \in P(\mu,\nu)} \mathbb{E}_{(X,Y)\sim\gamma}[c(X,Y)]$$

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Vladimir Dmitriev <v.dmitriev@skoltech.ru>.

Final Projects of the Machine Learning 2020 Course, Skoltech, Moscow, Russian Federation, 2020.

where X and Y are two datasets generated from two different distributions μ and ν respectively, c is cost function and γ^* is optimal joint distribution of X and Y, that minimizes the expectation on c. In the existing literature γ^* is named as optimal transport plan and $\mathcal{W} = \mathbb{E}_{(X,Y)\sim\gamma}$. If c respects to some metric, then expectation of $[c(X,Y)]$ over transport plan can be viewed as Wassernstein distance between initial given distributions.

1.2. SPOT: Scalable Push-forward of Optimal Transport

Optimal Transport could be used to solve such problems as domain adaptation and image generation. But in such complicated tasks arises heavy computational problems: for example, to take a small approximation error for complex distributions in high dimensions the grid size needs to be exponentially large. To solve this issue new implicit generative learning-based framework for solving optimal transport problems was presented by (Xie et al., 2019). They approximate γ^* by a generative model G (for example, neural ordinary differential equation(ODE) or deep neural network), which maps from some latent variable Z from latent distribution ρ to (X,Y):

$$\begin{bmatrix} X \\ Y \end{bmatrix} = G(Z) = \begin{bmatrix} G_X(Z) \\ G_Y(Z) \end{bmatrix} \; with \; Z \sim \rho$$

Accordingly, optimal transport problem could be formulated as:

$$G^* = argmin_G \, \mathbb{E}_{Z\sim\rho}[c(G_X(Z), G_Y(Z))]$$

$$subject \; to \;\; G_X(Z) \sim \mu, G_Y(Z) \sim \nu$$

which can be solve with Lagrangian multiplier methods for minimax optimization problems.Strictly speaking it doesn't solve exactly initial optimal transport problem, it just let one to achieve some mapping between distributions. To solve exactly initial OT problem neural ODE approach could be applied. We are not aimed to discuss this approach here and more references and theoretical explanations of SPOT algorithm could be found in (Xie et al., 2019).

1.3. The main contributions of this report

To successfully complete the project, we were assigned the following tasks:

- Code the GAN-based SPOT algorithm from (Xie et al., 2019)
- Test it in on toy datasets with samples from Gaussian distribution
- Test it on domain adaptation (e.g. MNIST→USPS) and MNIST digits generation (generative optimal transport)
- Test the approach on paired image generation (photo to monet).

Experiments and problems we faced during the work on this project:

- SPOT on toy datasets with samples from Gaussian distribution on Zhores supercomputer sandbox CPU (Zacharov et al., 2019)
- Domain adaptation SPOT (DASPOT) on MNIST→USPS, USPS→MNISTM and others on Zhores supercomputer sandbox GPU
- SPOT on MNIST digits generation and paired image generation from photo to monet on Zhores supercomputer sandbox CPU. Here we faced some problems related with sources: on Zhores supercomputer sandbox only one GPU is available and the calculation time on it is limited to 3 hours.

2. Related works

2.1. Methods for sample generation

Another application of algorithms, which can solve Optimal Transport problem between different distributions, is new samples generation. With a condition that these samples should be plausibly obtained from given datasets. For instance, using this method, new photographs, similar, but not the same as in the existing distribution, can be generated. So such issues as generation of photographs, images, realistic scenes, faces, cartoon characters, etc. can be solved. As the problem is in a great request, various techniques exists to find a solution (conditional GANs, CycleGANs, SPOT, etc.). But still there are challenges in realization, such as accuracy (example on pokemon dataset), computation time and memory issues(as for large datasets extremely powerful resources are needed). As an example, these are some articles, in which authors found a key to some gripping problems. In the paper (Brock et al., 2018) synthetic images, that are looks almost as real photos, were received (Figure 1). Another example (Karras et al., 2017) shows the photos generation of non-real human faces(Figure 2).

Figure 1. Example of Realistic Synthetic Photographs Generated with BigGAN (Brock et al., 2018)

Figure 2. Examples of Photorealistic GAN-Generated Faces.Taken from Progressive Growing of GANs for Improved Quality, Stability, and Variation, 2017. (Karras et al., 2017)

2.2. CoGAN

Coupled generative adversarial network (CoGAN) is another method for learning a joint distribution of multi-domain images. It doesn't require tuples of corresponding images in different domains in the training set, CoGAN can learn a joint distribution without any tuple of corresponding images. It can learn a joint distribution with just samples drawn from the marginal distributions. This is achieved by enforcing a weight-sharing constraint that limits the network capacity and favors a joint distribution solution over a product of marginal distributions one. CoGAN consists of the couple of (GANs) framework, which has been established as a viable solution for image distribution learning tasks. Each GAN is responsible for one image domain. (Liu & Tuzel, 2016)

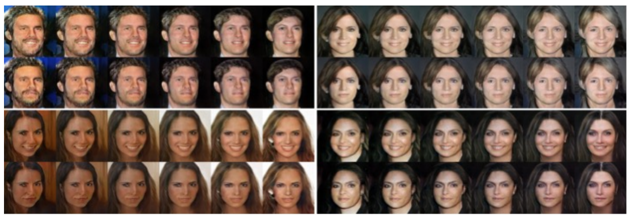

Figure 3. CoGAN learn to generate corresponding smile and non-smile faces (Liu & Tuzel, 2016)

3. Algorithms and Models

3.1. SPOT for domain adaptation: DASPOT

One of the areas of application of optimal transport is domain adaptation. We collect multiple datasets from different domains, then we build model trained on a source dataset, which further we adapt to target datasets. In DA problem we consider two distinct probability distributions X, Y related to a source and a target domains respectively, μ, ν are respective joint continuous distributions.

Here we provide description of algorithm implemented in article (Xie et al., 2019) and in our project. Lets start with the setup: we have two datasets and for one of them we have class labels for each object. For another dataset we don't have any labels for objects. The goal is to train classifier on the first labeled dataset and then to efficiently apply this classifier to objects from another dataset. Authors (Xie et al., 2019) proposed to use their OT -based algorithm to efficiently extract information about unlabeled dataset during learning of the classifier. The idea is the following: we specify latent space and learn transport plan from latent space to known distributions of labeled and unlabeled datasets. This transport plan establish map between labeled and unlabeled datasets and so we have effective labels to learn classifier on second, unlabeled dataset. The idea here is based on creating an effective labeling for unlabeled dataset with help of SPOT algorithm. Additionally we get a framework for generating artificial paired samples from two datasets. As described in article, we constructed two classifiers - one for source dataset, for which we know labels, another for target dataset, for which we do not know labels. We do training step for transport plan and then provide training steps for classifiers : training classifier on pair of objects from datasets, for which we estimate pairing by our transport plan. For labeled dataset we use labels for training, for unlabeled dataset we estimate the closest class of corresponding object from labeled set. We iterate this steps providing solution of minimax problem below:

$$min_{D_x,D_y,G}\ \ max_{\lambda_x,\lambda_y}\mathcal{L}_c(G,\lambda_x,\lambda_y)+$$

$$+\frac{\eta_s}{n}\sum_{i=1}^{n}\mathcal{E}(D_x(x_i),label_{x_i})+$$

$$+E_z\left[\mathcal{E}(D_y(G_y(Z)),argmax(D_x(G_x(Z))))\right]$$

where Lagrangian is defined in next section 3.2.

Algorithm 1 DASPOT

```
**Input:** distribution in latent space ($Z$ coordinates) of dimensionality d, source dataset $X$, target dataset $Y$
Initialize networks: Generator $netG$, discriminator $netLambda$,
classifier $netCl\_emb$ and $net\_class$
**for** $i = 1$ **to** $Numberofepochs$ **do**
    1. Set networks to training mode, zero gradients
    2. Do forward propagation for z sampled from distribution in latent space(used batch size 128) for generator and discriminator networks
    3. Do backward propagation of error for discriminator.Train the discriminator. Do not provide backward propagation for generator on this step
    4. Zero gradients. Do backward propagation of error of discriminator with sign swoop.
    5. Add error of generator to the backward propagation. Do optimization step for generator.
    6. Do forward and backward propagation for real object from source sample. Do optimization step for classifier.
    7. Do forward and backward propagation for fake object from target sample. Use most probable label of corresponding fake object from source sample. Do optimization step for classifier.
**end for**
```

3.2. SPOT for sample generation

Let 's define the metric between a vector X and a distribution μ. For our purposes it will be better to use the standard Wasserstein metric(Xie et al., 2019). So if vector X μ, then the Wasserstein distance equals to zero. Let us denote it as

$$\mathcal{W}_1(X,\mu)=\sup_{\lambda_X\in\mathcal{F}^1}\mathbb{E}_X[\lambda_X(X)]-\mathbb{E}_{U\sim\mu}[\lambda_X(U)]$$

, where, according to the article, $\mathcal{F}^1$ denotes the class of all 1-Lipschitz functions from $\mathbb{R}^d$ to $\mathbb{R}$. So the problem of Optimal Transport we can formulate as following

$$\gamma^*=\operatorname*{argmin}_{\gamma}\mathbb{E}_{(X,Y)\sim\gamma}[c(X,Y)]$$

, subject to

$$\mathcal{W}_1(X,\mu)=0,\ \mathcal{W}_1(Y,\nu)=0$$

As it was before for the push-forward algorithm we introduce a mapping G, which gives us $(X, Y) =$

$(G_X(Z), G_Y(Z))$, where $Z \sim \rho$, ρ is a latent distribution. The problem of finding conditional minimum of the function we can rewrite as Lagrangian:

$$\min_{G} \max_{\lambda_X \in \mathcal{F}_Y^1, \lambda_Y \in \mathcal{F}_X^1, \eta_X, \eta_Y} \mathbb{E}_{Z\sim\rho}[c(G_X(Z), G_Y(Z))] + \eta_X \mathbb{E}_{Z\sim\rho}[\lambda_X(G_X(Z))] - \mathbb{E}_{U\sim\mu}[\lambda_X(U)] + \eta_Y \mathbb{E}_{Z\sim\rho}[\lambda_Y(G_Y(Z))] - \mathbb{E}_{V\sim\nu}[\lambda_Y(V)]$$

For SPOT algorithm for sample generation all mappings, $\lambda_X,\ \lambda_Y,\ G_X, G_Y$ are separate neural networks. Let $\mathcal{G}, \mathcal{F}_X^1, \mathcal{F}_X^1$, which are respectively 1-Lipschitz functions for $\lambda_X,\ \lambda_Y$.

As it was stated in (Xie et al., 2019), $\mathcal{G}, \mathcal{F}_X^1, \mathcal{F}_X^1$ are finited classes, so G cannot represent any continious distributions of (X, Y). In that case constraints cannot not be satisfied. Because of it, the task of finding Lagrangian multipliers for minimization could be impossible. To avoid such obstacle, we tune the parameters $\eta_X = \eta_Y = \eta$. So the Lagrangian for SPOT problem will be:

$$\min_{G\in\mathcal{G}} \max_{\lambda_X \in \mathcal{F}_Y^1, \lambda_Y \in \mathcal{F}_X^1} \mathbb{E}_{Z\sim\rho}[c(G_X(Z), G_Y(Z))] + \eta\Big(\mathbb{E}_{Z\sim\rho}[\lambda_X(G_X(Z))] - \mathbb{E}_{U\sim\mu}[\lambda_X(U)] + \mathbb{E}_{Z\sim\rho}[\lambda_Y(G_Y(Z))] - \mathbb{E}_{V\sim\nu}[\lambda_Y(V)]\Big)$$

.

To solve this issue, following the Algorithm1 from (Xie et al., 2019), we can implement several steps of gradient ascent on λ_X and λ_Y, then one-step gradient descent for G_X and G_Y for fixed other parameters on each time step. During each time iteration, based on Algorithm1 from the paper, Spectral Normalization should be performed to weights of neural network in order to keep up the Lipschitz constant being smaller than one. As even this method is computationally expensive we decided simply to set a limit of each component of weights to make them small enough for calculations. Based on those approaches the next algorithm was used:

Algorithm 2 Algorithm for SPOT

```
Input: Datasets $x_i{}_{i=1}^N \sim \mu, y_i{}_{j=1}^M \sim \nu$; Initialized networks $G_x, G_y, \lambda_X, \lambda_Y$ with parameters $\psi, \phi, \alpha, \beta$ respectively; $lr$ is learning rate; $n_critic$ - the number of gradient ascent for $\lambda_X,\ \lambda_Y$; n - batch size; $n_epochs$ - number of epochs
for epoch = 1, 2, ... $n_{epochs}$ do
  for i = 1,2, ... $n_{critic}$ do
    Sample mini-batch $\{x_i\}_{i=1}^n$ from $\{x_i\}_{i=1}^N$
    Sample mini-batch $\{y_i\}_{i=1}^n$ from $\{y_i\}_{i=1}^M$
    Sample mini-batch $\{z_i\}_{i=1}^n$ from $\rho$
    Add limitations on $\alpha$ and $\beta$
    $g_\alpha \leftarrow \nabla_\alpha(\frac{1}{n}\sum_{k=1}^n \lambda_{X,\alpha}(G_{X,\psi}(z_k)) - \frac{1}{n}\sum_{i=1}^n \lambda_{X,\alpha}(x_i))$
    $g_\beta \leftarrow \nabla_\beta(\frac{1}{n}\sum_{k=1}^n \lambda_{Y,\beta}(G_{Y,\phi}(z_k)) - \frac{1}{n}\sum_{i=1}^n \lambda_{Y,\beta}(y_i))$
    $\alpha \leftarrow \alpha + lr \cdot g_\alpha$
    $\beta \leftarrow \beta + lr \cdot g_\beta$
  end for
  Sample mini-batch $\{z_k\}_{k=1}^n$ from $\rho$
  $g_\psi \leftarrow \nabla_\psi(\frac{1}{n}\sum_{k=1}^n \eta \cdot c(G_{X,\psi}(z_k), G_{Y,\phi}(z_k))) + \frac{1}{n}\sum_{k=1}^n \lambda_{X,\alpha}(G_{X,\psi}(z_k))$
  $g_\phi \leftarrow \nabla_\phi(\frac{1}{n}\sum_{k=1}^n \eta \cdot c(G_{X,\psi}(z_k), G_{Y,\phi}(z_k))) + \frac{1}{n}\sum_{k=1}^n \lambda_{Y,\beta}(G_{Y,\phi}(z_k))$
  $\psi \leftarrow \psi + lr \cdot g_\psi$
  $\phi \leftarrow \phi + lr \cdot g_\phi$
end for
```

4. Experiments and Results

The estimation of SPOT performance was carried out on the basis of several experiments: domain adaptation, paired image generation and generation of new samples based on paired Gaussian distributions. We have also measured the performance of implemented CoGAN algorithm to compare it with SPOT. Develompent of all algorithms was done using our laptops, but usage of personal computers is not ideal for testing such algorithms because of sufficient computational complexity. To obtain data for analysis and comparison we run coded algorithms using GPU on Zhores sandbox(Zacharov et al., 2019). All algorithms were implemented with PyTorch.

You are welcome to find code and additional files on our githab.

4.1. DASPOT

Domain adaptation SPOT algorithm was implemented in python and tested on different pairs of digits recognition datasets including hand-written digits (MNIST, MNISTM, USPS) and real-world image dataset (SVHN). We applied SPOT for following source-target pairs: MNIST-MNISTM, MNIST-USPS, MNIST-SVHN, SVHN-USPS, USPS-MNISTM, USPS-MNIST. As a result we obtained pictures of generated dataset samples (top row is source, other is target). In report were put only two resulting samples - for MNIST-MNISTM (Figure 4), MNIST-USPS (Figure 5). Figure for other pairs can be found in provided github repository.

For all pairs of datasets we used For experiments we took 30000 iterations with learning rate 0.0002. As for optimizer we used Adam (Kingma & Ba, 2014). Below we provide description of architecture of our networks:

- **Generator**. Each of 5 layers consists of three functions: nn.ConvTranspose2d, nn.BatchNorm2d, nn.PReLU. The sequence of kernel sizes, channel numbers, strides and padding for nn.ConvTranspose2d from first layer to last layer is $(4, 1024, 1, 0) \rightarrow (3, 512, 2, 1) \rightarrow (3, 256, 2, 1) \rightarrow (3, 128, 2, 1) \rightarrow (6, output\_size, 1, 1)$. First 4 layers were common for generator of source objects and generator of target objects and only last layer was separate.
- **Discriminator and classifiers**. Discriminator was combined with classifiers. First layer was separate for two inputs - it consists from nn.Conv2d with kernel_size = 5, 20 output channels, stride=1 and padding=0 and from nn.MaxPool2d with kernel_size = 2. Then follows two common layers with convolution parameters of nn.Conv2d: $(5, 50, 1, 0) \rightarrow (4, 500, 1, 0)$ and with nn.MaxPool2d with kernel_size = 2 for first of them and with nn.PReLU() for second of them. Then follows two separate layers of nn.Conv2d with parameters $(1, 100, 1, 0) \rightarrow (1, 1, 1, 0)$ and nn.PReLU() in between of this convolutions. Moreover, there is classification part which mounts to third layer and consist of nn.Conv2d with parameters $(1, 10, 1, 0)$.

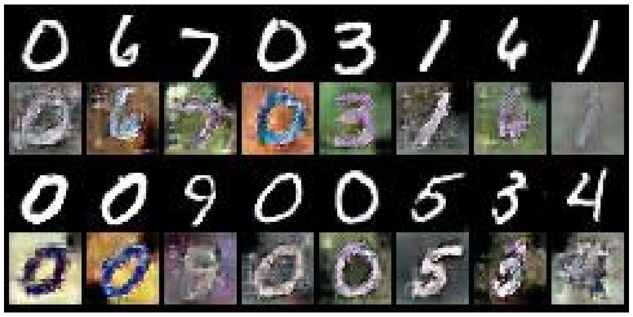

Figure 4. DASPOT: MNIST $\rightarrow$ MNISTM

On (Figure 4) we provide pairs of corresponding numbers generated by DASPOT for given labeled MNIST and unlabeled MNISTM datasets. This pairs were achieved by generated network trained during the process of DASPOT learning. The same, but for unlabeled dataset USPS, is depicted on (Figure 5). Here we have to remind that for needs

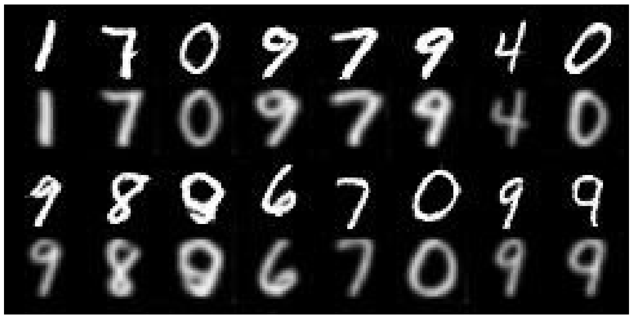

Figure 5. DASPOT: MNIST $\rightarrow$ USPS

of domain adaptation the main outcome of DASPOT is classifier adopted to unlabeled domain, but good generator is additional result. On (Figure **??**) we provide generated pairs of corresponding images achieved by another generative-adversarial networks based algorithm - coGAN. Visual comparison proves better quality of SPOT algorithms. As for the closest to real world dataset, SVHN, results are slightly worse. It definitely related to quality of intial data: it may contain 2-digits numbers what complicates classification. As well training time for colored datasets should be larger. So then it's easy to understand why worst performance of DASPOT algorithm was shown by SVHN-MNISTM.

To provide visualisation of evolution of generated distribution we saved t-SNE dimensionality reduction results. As we need this only for visualisation, we did t-SNE only to two-dimensional space. On (Figure 6) t-SNE results for step 29500 are shown. As one may expect, generated distribution mimics given one, this is one more visual evidence of good convergence of SPOT performance.

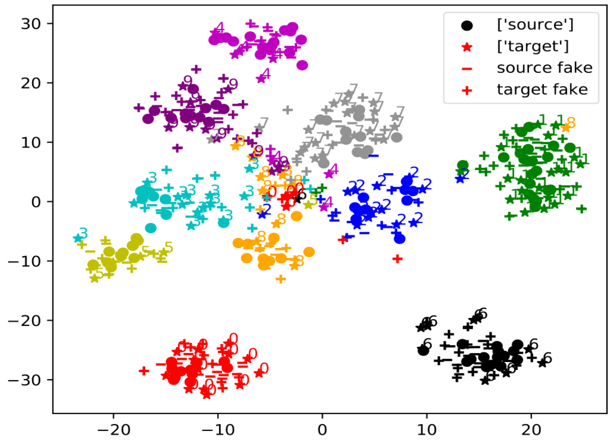

Figure 6. DASPOT: Low-dimensional visualisation of initial and generated distributions

4.2. CoGAN

We decided to compare our realization of SPOT with CoGAN like it was made in original article (Xie et al., 2019). Unfortunately, we faced some problems with the CoGAN pytorch-implementation repository, which was cited in original article (Xie et al., 2019), but we have found (another source) and use it for comparison. We have made 31 epohs and got the following result:

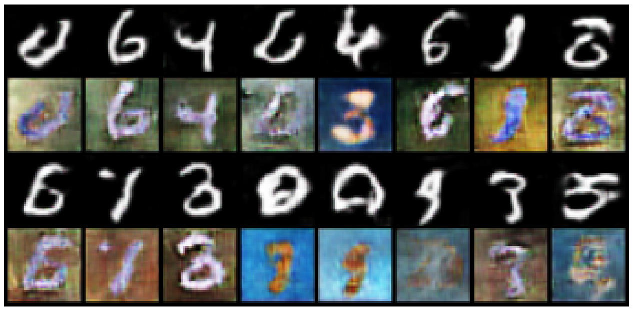

Figure 7. CoGAN: MNIST $\to$ MNISTM

It is clear from the pictures above (Figure 7), that SPOT performs quite better, than CoGAN. We also launched training of an CoGAN Caffe-implementation from the repo, used in (Xie et al., 2019). But we did not have enough laptop-power and time to wait for any results from it.

4.3. SPOT

By replication of (Xie et al., 2019) algorithms, we can show that with solving SPOT problems, we can generate paired samples from unbound but similar distributions. $((X, Y) \to (G(X), G(Y)))$

4.3.1. Synthetic Data

The implementation is the following. We initialize four neural networks without sharing parameters, 2 generators and 2 discriminators - G_X, G_Y, λ_X, λ_Y, respectively. Leaky-ReLU activation (Maas, 2013) was used for activational layers. Network for G_X and G_Y has 2 hidden linear layers, meanwhile, network for λ_X and λ_Y has 1 hidden layer. All layers have 30 units per each layer. The latent variable was generated from standard Gaussian distribution ($Z \sim \mathcal{N}(0, I) \in \mathbb{R}^2$). Batch size is equal to 100. We used clamping parameter for weights in λ_X and λ_Y equaled to 0.1.
For synthetic data $c(x, y) = ||x - y||^2$ was taken as a cost function. Learning rate was 10^{-3}. We have two marginal distributions to check. We generated $n = 10^4$ samples of each distribution for training neural networks. First experiment was to generate samples from distributions μ and ν with following parameters. The means are $(-2.5, 0)^T$ and $(2.5, 0)^T$ respectively and covariance $0.5I$, shown in the upper-left corner of Figure 8. Then with performing SPOT algorithm we got the result, which is shown in the left-upper corner of Figure 8. The second experiment was performed using Gaussian distribution as μ with mean $(-2.5, 0)^T$ and covariance $0.5I$, as ν was taken $(sin(Y_1) + Y_2,\ 2Y_1 - 3)^T$, where Y_1 is a uniform distribution on $[0, 3]$ and Y_2 follows a Gaussian distribution $\mathcal{N}(2, 0.1)$. The initial and generated be SPOT distributions are located at the lower row of the Figure 8. As it was shown in (Xie et al., 2019), all of the generated samples approximately follow the distributions μ and ν.

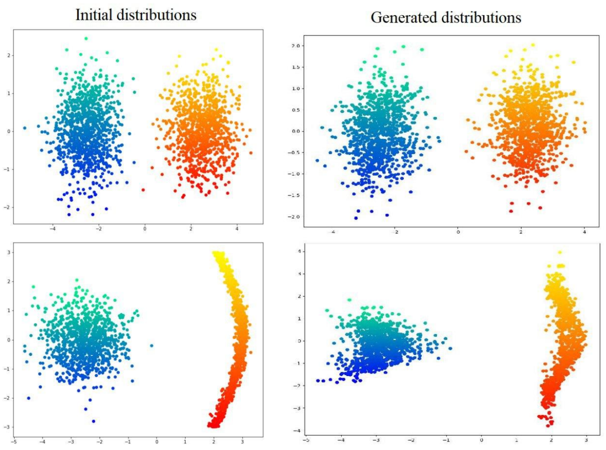

Figure 8. SPOT: Initial and generated distributions.

4.3.2. Real Data

Our goal was to show as in (Xie et al., 2019), that SPOT algorithm can generate high quality samples of paired images such as MNIST $\to$ MNISTM and Photo $\to$ Monet. But due to the high computational cost and limited resources we checked the algorithm, which will be identical for both datasets, only on MNIST-MNISTM samples. At this experiments convolutional and deconvolutional neural networks was used for describing G_X, G_Y and λ_X, λ_Y. To solve OT problem for such paired dataset authors (Xie et al., 2019) proposed to apply Sobel filter to images to stress contours of numbers as common feature. The cost function then was chosen as Frobenius norm of difference between convolutions of Sobel filter with images from source and target datasets. When this norm is small, difference in contours is also small and one expect to see the same number on both images. Formally it will look like:

$$Cost(x, y) = \sum_{channels} \sum_{Sobels} ||(C_i \otimes x_j - C_i \otimes y_j)||_F$$

For the first pair of networks, as it was stated in Appendix.A1((Xie et al., 2019)), 4 hidden layers was implemented. Batch size was equal to 32.

- For generator network the first layer consists from nn.ConvTranspose2d function with kernel_size, output channels, stride and padding equal respectively (3,512,1,0). Then follows 4 blocks of functions, each block consist from nn.BatchNorm2d, nn.ReLU and nn.ConvTranspose2d. For nn.ConvTranspose2d the sequence of parameters is following:

$$(3, 256, 2, 1) \rightarrow (3, 128, 2, 1) \rightarrow$$

$$(3, 64, 2, 1) \rightarrow (2, 3, 2, 1)$$

  Then activation function was applied: nn.Tanh.

- For discriminator network we have the same architecture but with nn.LeakyReLU(0.2) instead of nn.ReLU and with nn.Conv2d instead of nn.ConvTranspose2d. The sequence of parameters for convolutions is following:

$$(4, 64, 1, 0) \rightarrow (2, 128, 2, 1) \rightarrow$$

$$(4, 512, 2, 1) \rightarrow (3, 1, 1, 0)$$

As it was mentioned in the given article, we should run this code approximately $2 \cdot 10^5$ times for obtaining stable results. In our case we performed 1000 time iterations and it took lots of computation time. In our work all results we obtained are illustrated on Figure 9 and Figure 10

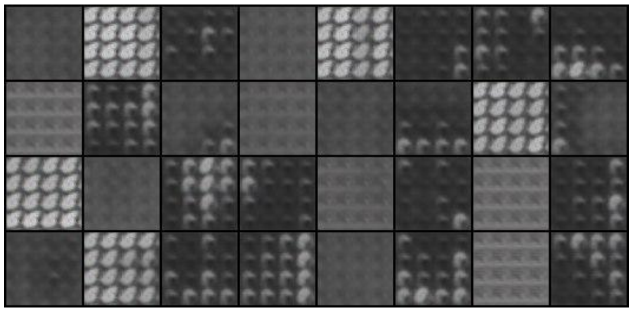
Figure 9. Generated sample for MNIST dataset.

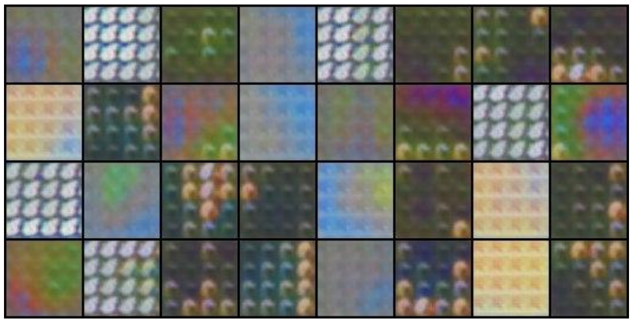
Figure 10. Generated sample for MNISTM dataset.

5. Conclusion

In this project we obtained the following results:

- Study OT problem and its application to sample generation and domain adaptation. In particular, we test SPOT algorithm on problem of paired data generation for cases, when we need to get pairs of corresponding images and to have an ability to generate labeled data.
- Provide test of SPOT algorithm on different distributions and achieved good agreement with results of (Xie et al., 2019).
- Provide tests on handwritten numbers images paired datasets. Good results was not achieved due to the inefficient code and lack of resources.

References

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis, 2018.

Ganin, Y. and Lempitsky, V. Unsupervised domain adaptation by backpropagation, 2014.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014.

Liu, M.-Y. and Tuzel, O. Coupled generative adversarial networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 469–477. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6544-coupled-generative-adversarial-networks.pdf`.

Maas, A. L. Rectifier nonlinearities improve neural network acoustic models. 2013.

Santambrogio, F. Models and applications of optimal transport in economics, traffic and urban planning, 2010.

Xie, Y., Chen, M., Jiang, H., Zhao, T., and Zha, H. On scalable and efficient computation of large scale optimal transport. *CoRR*, abs/1905.00158, 2019. URL http://arxiv.org/abs/1905.00158.

Zacharov, I., Arslanov, R., Gunin, M., Stefonishin, D., Pavlov, S., Panarin, O., Maliutin, A., Rykovanov, S. G., and Fedorov, M. 'zhores' - petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in skolkovo institute of science and technology. *CoRR*, abs/1902.07490, 2019. URL http://arxiv.org/abs/1902.07490.

A. Team member's contributions

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

Vladimir Dmitriev (20% of work)

- Reviewing literature on the topic (2 papers)
- Coding DASPOT algorithm
- Preparing the Section 1, 2.1 of this report

Alexander Shumilov (20% of work)

- Reviewing literature on the topic (2 papers, 1 video)
- Coding DASPOT algorithm
- Preparing the GitHub Repo
- Preparing the Section 3 of this report

Veronika Zorina (20% of work)

- Reviewing literature on the topic (2 papers, 1 video)
- Coding SPOT algorithm for sample generation
- Experimenting SPOT
- Preparing the Sections 2.2, 4.2 of this report

Daria Gazizova (20% of work)

- Reviewing literature on the topic (1 paper, 1 video)
- Experimenting DASPOT and SPOT on Zhores supercomputer sandbox GPU
- Preparing the Section 1 of this report
- Preparing and record presentation

Elizaveta Noskova (20% of work)

- Reviewing literature on the topic (2 papers)
- Reproducing CoGAN algorithm
- Experimenting CoGAN for domain adaptation on Zhores supercomputer sandbox GPU
- Preparing the Section 3.2, 4.1 of this report

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

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.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: CoGAN code

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: None

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

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

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

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

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: None

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

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment: None

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: None

9. The exact number of evaluation runs is included.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

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: None