How Well Do WGANs Estimate the Wasserstein Metric? (Machine Learning 2021 Course)

Denis Rollov [1] **Nikolay Goncharov** [1] **Svetlana Gabdullina** [1]

Abstract

A great variety of neural-network-based methods for estimating the Wasserstein-1 distance between probability distributions has been recently proposed in the literature. Many of them are being used in generative models, e.g. Wasserstein GAN. However, the question of how well do these methods actually estimate the Wasserstein-1 distance still remains open. In this project, we study different ways for computing W1 distance between distributions by using neural networks and how well they perform. For this gradient penalty (GP), weight clipping (WC), c- / $(c, 0.1)$- / $(c, 1)$-transform, and Lipschitz penalty (LP) are considered.

Github repo: project github with code
Video presentation: your link here (gdrive, etc.)

1. Introduction

General adversarial networks (GANs) (Goodfellow et al., 2014) are a class of generative models that have recently gained a lot of attention. They are based on the idea of defining a game between two competing neural networks (NNs): a generator and a discriminator. While the discriminator aims at distinguishing generated from real data, the generator tries to generate samples which the discriminator can not distinguish from the ones from the empirical distribution. To efficiently train the discriminator the comparison between distributions is needed. One of the options is optimal transport (OT) which robustly compares structured data, e.g., probability measures (images, point clouds), and empirical distributions (Lai & Zhao, 2014). OT-inspired objectives can be seen as lifting similarity measures between samples to similarity measures between probability measures. When the cost function c is related to a distance function d by $c(x, y) = d^p(x, y)$, $p \geq 1$, the OT formulation defines the so-called Wasserstein metric, which is a distance on the space of probability measures, i.e. a symmetric and positive definite function that satisfies the triangle inequality.

It was proposed (Arjovsky & Bottou, 2017) to train discriminator by minimizing the Wasserstein-1 distance ($p = 1$). Accordingly, this version of GAN was called Wasserstein GAN (WGAN). Using this metric requires the discriminator function to lie in the space of 1-Lipschitz functions. There are different ways to estimating the Wasserstein metric and satisfy this constraint.

In the original paper, this was enforced by clipping the weights of the discriminator to lie inside some small box, which, however, proved to be inefficient. The Gradient penalty WGAN (WGANGP) (Gulrajani et al., 2017) was more successful at this, by enforcing the constraint through a gradient norm penalization. These and three more methods are compared in (Mallasto et al., 2019b). One more way is the Lipschitz penalty (LP) which was presented in (Petzka et al., 2018).

The main contributions of this report are:

- Reproducing the results observed in (Mallasto et al., 2019b) by quantifying how much the estimates differ from accurately computed ground truth values between subsets of commonly used datasets and how stable the approximations are.
- Adding LP-method to the comparison above.

2. Preliminaries

Here we will introduce key definitions and concepts.

2.1. Probabilistic Notions

Let $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ be complete and separable metric spaces. Let $\mathcal{P}(\mathcal{X})$ be the set of probability measures on $\mathcal{X}$. Then, given a probability measure $\mu \in \mathcal{P}(\mathcal{X})$ and $\nu \in \mathcal{P}(\mathcal{Y})$, we define $\mathrm{ADM}(\mu, \nu)$ as a set of elements γ which define a joint probabilities between μ and ν.

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Denis Rollov <Denis.Rollov@skoltech.ru>, Nikolay Goncharov <Nikolay.Goncharov@skoltech.ru>, Svetlana Gabdullina <Svetlana.Gabdullina@skoltech.ru>.

Final Projects of the Machine Learning 2021 Course, Skoltech, Moscow, Russian Federation, 2021.

2.2. Optimal Transport Problem

Given a continuous and lower-bounded cost function $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, the optimal transport problem between probability measures μ and ν is defined as

$$\mathrm{OT}_c(\mu, \nu) := \min_{\gamma \in \mathrm{ADM}(\mu,\nu)} \mathbb{E}_\gamma[c]. \tag{1}$$

Equation (1) defines a constrained linear program, and admits a *dual formulation* that can be approximated using *discriminator* neural networks. Using properties of *Kantorovich potentials*, duality can be written as (Villani, 2008):

$$\mathrm{OT}_c(\mu, \nu) = \max_{(\varphi,\varphi^c)\in \mathrm{ADM}(c)} \left\{\mathbb{E}_\mu[\varphi] + \mathbb{E}_\nu[\varphi^c]\right\}, \tag{2}$$

where φ^c is the *c-transform* of φ:

$$\varphi^c : \mathcal{Y} \to \mathbb{R},\ y \mapsto \inf_{x\in\mathcal{X}} \left\{c(x, y) - \varphi(x)\right\}. \tag{3}$$

2.3. Wasserstein Metric

Consider the case when $\mathcal{X} = \mathcal{Y}$ and $c(x, y) = d^p_{\mathcal{X}}(x, y)$, $p \geq 1$, where we refer to $d_{\mathcal{X}}$ as the ground metric. Then, the optimal transport problem (1) defines the p-Wasserstein metric $W_p(\mu, \nu) := OT_{d^p_{\mathcal{X}}}(\mu, \nu)^{1/p}$ on the space of probability measures with finite p-moments.

3. Related work

The vanilla GAN (Goodfellow et al., 2014) minimizes an approximation to the *Jensen-Shannon (JS) divergence* between the push-forward and target. Later contributions helped to successfully model complex distributions using GANs.

In (Arjovsky & Bottou, 2017) it was proposed to use Wasserstein-1 distance, which has properties superior to the Jensen-Shannon distance in terms of convergence. As it was stated above, the minimization problem with the new metric requires the discriminator function to lie in the space of 1-Lipschitz functions.

In the same paper, the Lipschitz constraint was guaranteed by performing weight clipping, i.e., by constraining the parameters of the discriminator NN to be smaller than a given value in magnitude.

An improved training strategy was proposed in (Gulrajani et al., 2017) based on results from optimal transport theory. Here, instead of clipping weights, the loss gets augmented by a regularization term that penalizes any deviation of the norm of the gradient of the critic function (with respect to its input) from one.

A more direct approach that computes the c-transform over minibatches was proposed in (Mallasto et al., 2019a), avoiding the need to ensure Lipschitzness. In (Mallasto et al., 2019b) also an entropic relaxation approach through (c, ϵ)-transforms over minibatches is considered.

The approach with using a weaker regularization term than in GP is presented in (Petzka et al., 2018). We compare all listed methods in terms of approximation and stability in experimental section.

4. Algorithms and Models

We now discuss how listed different estimates are computed in practice by sampling minibatches of size N from μ and ν, yielding $\{x_i\}_{i=1}^N$ and $\{y_i\}_{i=1}^N$, respectively, at each training iteration. Then, with each method, the distance is estimated by maximizing a model-specific expression that relates to the dual formulation of the 1-Wasserstein distance in (2) over the minibatches. This maximization is carried out via gradient ascent in practice.

4.1. Weight clipping

The vanilla WGAN enforces 1-Lipschitzness of the discriminator at each iteration by forcing the weights W^k of the neural network to lie inside some box $-\varepsilon \leq W^k \leq \varepsilon$, considered coordinate-wise, for some small $\varepsilon > 0$ ($\varepsilon = 0.01$ in the original work). Here k stands for the k^{th} layer in the neural network. The objective for maximization can be written as

$$\max_\omega \left\{\frac{1}{N}\sum_{i=1}^N \varphi_\omega(x_i) - \frac{1}{N}\sum_{i=1}^N \varphi_\omega(y_i)\right\}. \tag{4}$$

4.2. Gradient penalty

Notice that the 1-Lipschitz condition implies that $\|\Delta_x \varphi_\omega(x)\| \leq 1$ holds for x almost surely under μ and ν. This condition can be enforced through the penalization term $\mathbb{E}_{x\sim\chi}[\max(0, 1 - \|\Delta_x\varphi_\omega(x)\|)^2$, where χ is some reference measure, proposed to be the uniform distribution between pairs of points of the minibatches. The objective for this method can be written as

$$\max_\omega \left\{\frac{1}{N}\sum_{i=1}^N \varphi_\omega\left(x_i\right) - \frac{1}{N}\sum_{i=1}^N \varphi_\omega\left(y_i\right) - \frac{\lambda}{M}\sum_{i=1}^M \left(1 - \|\nabla_{z=z_i}\varphi_\omega(z)\|\right)^2\right\} \tag{5}$$

where λ is the magnitude of the penalization, which was chosen to be $\lambda = 10$ in the original paper, and $\{z_i\}_{i=1}^M$ are samples from χ.

4.3. *c*-transform

The $\mathrm{ADM}(c)$ constraint can be taken into account directly, by computing the c-transform over the minibatches as

$$\varphi_\omega^c(y_i) \approx \widehat{\varphi_\omega^c}(y_i) = \min_j \{c(x_j, y_i) - \varphi_\omega(x_j)\}, \quad (6)$$

where $c = d_2$ in the 1-Wasserstein case. The original paper proposes to include penalization terms to enforce the discriminator constraints, however, this is unnecessary as the c-transform enforces the constraints. Therefore, the objective can be written as

$$\max_\omega \left\{ \frac{1}{N}\sum_{i=1}^N \varphi_\omega(x_i) + \frac{1}{N}\sum_{i=1}^N \widehat{\varphi_\omega^c}(y_i) \right\}. \quad (7)$$

4.4. (*c*, ϵ)-transform

Entropic relaxation (Cuturi, 2013) applied to W_1 results in the (1,ϵ)-Sinkhorn divergence S_1^ϵ (Genevay et al., 2017), which satisfies $S_1^\epsilon \to W_1$ as $\epsilon \to 0$. Then, S_1^ϵ can be viewed as a smooth approximation to W_1. The benefits of this approach are that φ_ω is not required to satisfy the $\mathrm{ADM}(c)$ constraint, and the resulting transport plan is smoother, providing robustness towards noisy samples.

$$\begin{aligned} S_1^\epsilon(\mu,\nu) = \mathrm{OT}_c^\epsilon(\mu,\nu) - \\ \frac{1}{2}\left(\mathrm{OT}_c^\epsilon(\mu,\mu) + \mathrm{OT}_c^\epsilon(\nu,\nu)\right). \end{aligned} \quad (8)$$

The terms $\mathrm{OT}_c^\epsilon(\mu,\mu)$ and $\mathrm{OT}_c^\epsilon(\nu,\nu)$ are computed with Sinkhorn-Knopp algorithm (Feydy et al., 2018). (c,ϵ)-transform is computed as

$$\begin{aligned} &\varphi_\omega^c(y_i) \approx \widehat{\varphi_\omega^{(c,\epsilon)}}(y_i) = \\ &-\epsilon \log\left(\frac{1}{N}\sum_{j=1}^N \exp\left(-\frac{1}{\epsilon}\left(c(x_j,y_i) - \varphi_\omega(x_j)\right)\right)\right), \end{aligned} \quad (9)$$

and the objective for the discriminator is

$$\max_\omega \left\{ \frac{1}{N}\sum_{i=1}^N \varphi_\omega(x_i) + \frac{1}{N}\sum_{j=1}^N \widehat{\varphi_\omega^{(c,\epsilon)}}(y_j) \right\}. \quad (10)$$

4.5. Lipschitz penalty

An alternative to Gradient penalty method is Lipschitz penalty method (Petzka et al., 2018). Since the NN generates (almost everywhere) differentiable functions, we can penalize whenever gradient norms are strictly larger than one. For these reasons authors suggest to add the regularization term $\max\{0, \|\Delta_{z=z_i}\varphi_\omega(z)\| - 1\}^2$, and the final formula looks as follows:

$$\begin{aligned} \max_\omega \Bigg\{ &\frac{1}{N}\sum_{i=1}^N \varphi_\omega(x_i) - \frac{1}{N}\sum_{i=1}^N \varphi_\omega(y_i) - \\ &\frac{\lambda}{M}\sum_{i=1}^M \max\{0, \|\nabla_{z=z_i}\varphi_\omega(z)\| - 1\}^2 \Bigg\}. \end{aligned} \quad (11)$$

5. Experiments and Results

All the experiments can be found on github.

Our task is to study how efficiently the presented methods estimate the 1-Wasserstein metric. The experiments use the MNIST and CIFAR-10 datasets[1]. On these datasets, we look at two characteristics:

- how well the minibatch-wise distance between two measures can be computed,
- how well these minibatch-wise distances relate to the 1-Wasserstein distance between the two full measures.

The discriminator is modeled as either

- a simple multilayer perceptron (MLP) with two hidden layers of width 128 and ReLU activations, and a linear output,

or

- a convolutional neural network architecture (CNN) used in DCGAN[2] (Radford et al., 2016),

and trained by optimizing the objective function using batch gradient ascent.

- For the gradient penalty we use the Adam optimizer with following parameters. Beta values are $(0, 0.9)$. In MLP case for both datasets learning rate is 5×10^{-3}. In CNN case – 8×10^{-3} and 10^{-2} for MNIST and CIFAR-10 respectively.
- For the Lipschitz penalty methods, we use the Adam optimizer with beta values $(0, 0.9)$ and learning rate for MLP for both datasets 10^{-3} and for CNN – 10^{-4} and 10^{-3} for MNIST and CIFAR-10 respectively.

[1] The original paper (Mallasto et al., 2019b) also uses CelebA, but the dataset was not available at the server when the project was being done.

[2] MNIST dataset is resized to (32, 32) to fit the model.

- For weight clipping we use RMSprop with learning rate 5×10^{-5}.
- For the c-transform we use RMSprop with learning rate 10^{-3}.
- For the (c, ϵ)-transform we use RMSprop with learning rate 10^{-4}.

5.1. Approximation

We normalize the datasets with parameters ((0.1307,), (0.3081,)) and divide them into two parts which contain samples from different classes, forming the two measures μ and ν, between which the distance is approximated, and train the discriminators on minibatches $\mu_k \subset \mu$ and $\nu_k \subset \nu$, $k = 1, ..., 500$, of size 64 running 20 epochs. Then, without training the discriminators, we sample another 100 evaluation minibatches and use the discriminators to approximate the minibatchwise distance between each μ_l and ν_l. The ground-truth minibatch-wise distance for comparing our approximations with is computed by POT. The errors are presented with 95%-confidence intervals.

The parameters of each model are listed below. The results are shown in the Tables 1, 2 and on the plots in Appendix.

5.1.1. Weight clipping

For the MLP model we chose $\epsilon = 0.05$ for MNIST dataset and $\epsilon = 0.08$ – for CIFAR-10 dataset. These parameters showed the lowest error. For the CNN model $\epsilon = 0.03$ and $\epsilon = 0.2$ in MNIST and CIFAR-10 cases respectively.

5.1.2. Gradient penalty and Lipschitz penalty

For all models and datasets we chose $\lambda = 10$.

5.1.3. (c, ϵ)-transform

For MNIST dataset we chose $\epsilon = 12$, for CIFAR-10 – $\epsilon = 1$. The choice of parameters is explained by numerical errors in numbers' representation. Smooth minimum in the formula can return nan in some cases.

Table 1. Approximation errors for MNIST.

MNIST	MLP	CNN
WC	0.407 ± 0.018	0.184 ± 0.011
GP	1.641 ± 0.009	2.612 ± 0.049
LP	1.491 ± 0.002	61.65 ± 1.659
c-T	0.059 ± 0.001	0.065 ± 0.001
(c, ϵ)-T	0.189 ± 0.001	0.186 ± 0.001

Table 2. Approximation errors for CIFAR-10.

CIFAR-10	MLP	CNN
WC	4.854 ± 0.157	3.056 ± 0.109
GP	7.819 ± 0.017	9.926 ± 0.063
LP	7.642 ± 0.011	9.761 ± 0.18
c-T	0.448 ± 0.005	0.344 ± 0.006
(c, ϵ)-T	1.696 ± 0.008	1.807 ± 0.007

5.2. Stability

For the stability check only MLP is used due to computational efficiency. We train the discriminators for 500 iterations on small datasets of size 512, that form subsets of the datasets mentioned above. We train with two minibatch sizes $N = 64$ and $N = 512$. We then compare how the resulting discriminators estimate the minibatch-wise and total distances (minibatches of size $M = 64$ and $M = 512$). Results are presented on the Tables 3 and 4.

Table 3. Stability check. Estimations on MNIST dataset.

MNIST	N	64	64	512	512
	M	64	512	64	512
WC		0.08	0.06	0.0	0.0
GP		-0.19	-0.66	-0.38	-0.18
LP		0.01	0.02	0.01	0.02
c-T		1.45	1.3	1.44	1.3
(c, ϵ)-T		1.69	1.64	1.65	1.59
ground		1.36	1.36	1.36	1.36

Table 4. Stability check. Estimations on CIFAR-10 dataset.

CIFAR-10	N	64	64	512	512
	M	64	512	64	512
WC		0.05	0.04	0.0	0.01
GP		-0.74	-0.14	-0.6	-2.9
LP		0.02	-0.04	0.01	0.01
c-T		7.21	6.49	6.98	6.3
(c, ϵ)-T		9.22	9.19	9.17	9.13
ground		6.89	6.89	6.89	6.89

6. Conclusion

Based on our experiments (Tables 1, 2), weight clipping, (c, ϵ)-transform and c-transform are more accurate at computing the minibatch distance and estimating the batch distance than the gradient penalty and Lipschitz penalty methods.

Tables 3 and 4 demonstrates that (c, ϵ)-transform and c-transform converge rapidly compared to weight clipping

method.

Gradient and Lipschitz penalty methods have complex implementation and behave unexpectedly. In our experiments both of these methods rather go to the minimum than to the maximum and even return negative distance, which of course is far from the truth.

c-transform and (c, ϵ)-transform showed the best results in both tests, but the interesting fact is that in our experiments c-transform estimation was always lower than ground-truth, while (c, ϵ)-transform transform was always higher. Further work with these methods can be related to the computational hacks of the smooth minimum, to allow smaller ϵ, which should make the error even smaller.

References

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks, 2017.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transportation distances, 2013.

Feydy, J., Séjourné, T., Vialard, F.-X., ichi Amari, S., Trouvé, A., and Peyré, G. Interpolating between optimal transport and mmd using sinkhorn divergences, 2018.

Genevay, A., Peyré, G., and Cuturi, M. Learning generative models with sinkhorn divergences, 2017.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved training of wasserstein gans, 2017.

Lai, R. and Zhao, H. Multi-scale non-rigid point cloud registration using robust sliced-wasserstein distance via laplace-beltrami eigenmap, 2014.

Mallasto, A., Frellsen, J., Boomsma, W., and Feragen, A. (q,p)-wasserstein gans: Comparing ground metrics for wasserstein gans, 2019a.

Mallasto, A., Montúfar, G., and Gerolin, A. How well do wgans estimate the wasserstein metric?, 2019b.

Petzka, H., Fischer, A., and Lukovnicov, D. On the regularization of wasserstein gans, 2018.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

Villani, C. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008. ISBN 9783540710509. URL https://books.google.ru/books?id=hV8o5R7_5tkC.

A. Plots for the approximation part

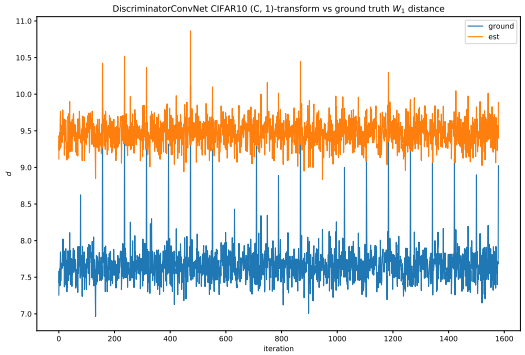

Figure 1. CNN, CIFAR-10, (c, ϵ)-transform.

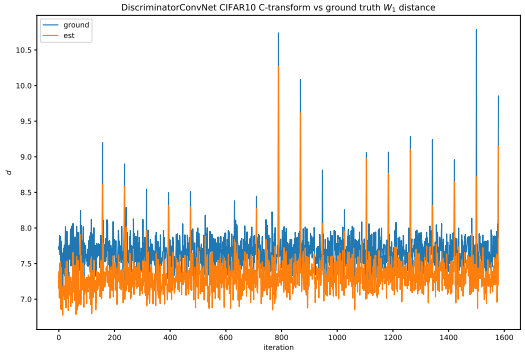

Figure 2. CNN, CIFAR-10, c-transform.

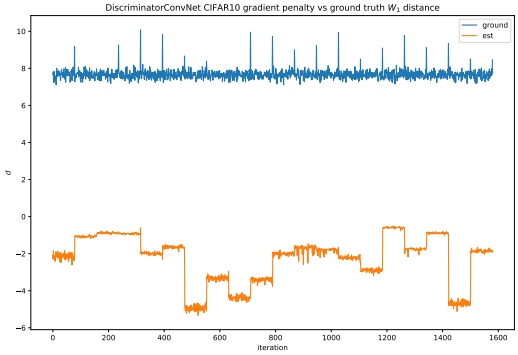

Figure 3. CNN, CIFAR-10, gradient penalty.

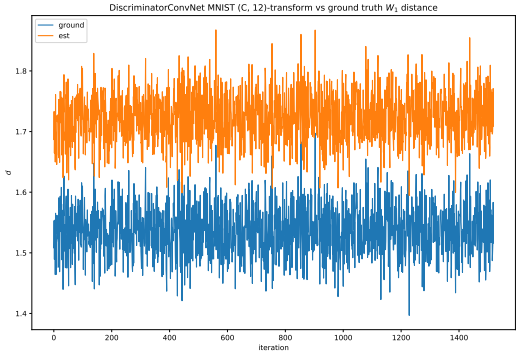

Figure 6. CNN, MNIST, (c, ϵ)-transform.

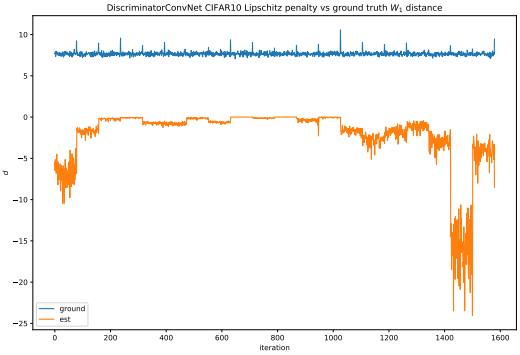

Figure 4. CNN, CIFAR-10, Lipschitz penalty.

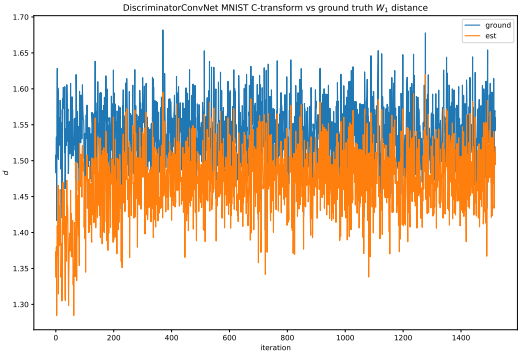

Figure 7. CNN, MNIST, c-transform.

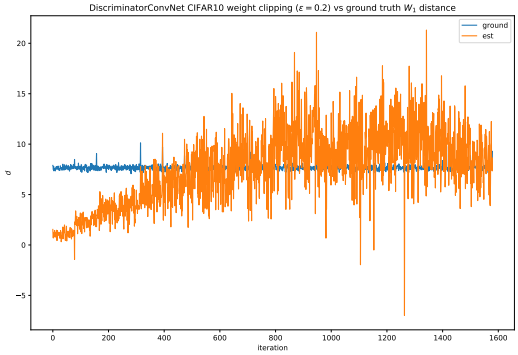

Figure 5. CNN, CIFAR-10, weight clipping.

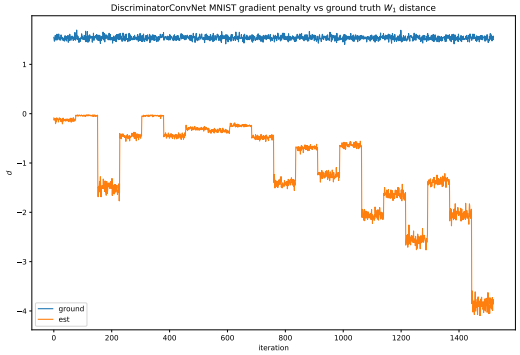

Figure 8. CNN, MNIST, gradient penalty.

Figure 9. CNN, MNIST, Lipschitz penalty.

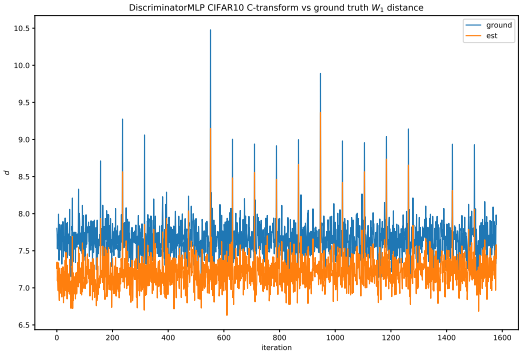

Figure 12. MLP, CIFAR-10, c-transform.

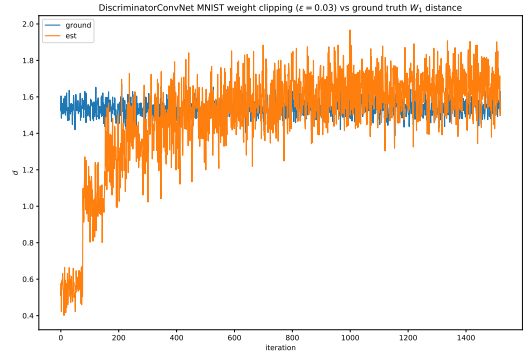

Figure 10. CNN, MNIST, weight clipping.

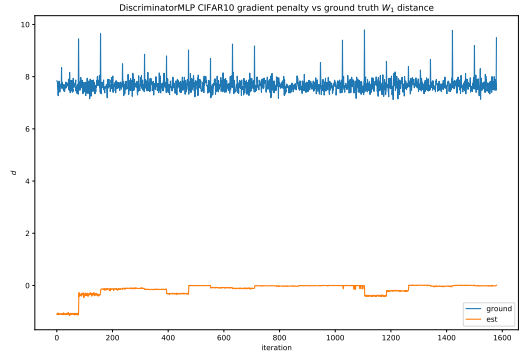

Figure 13. MLP, CIFAR-10, gradient penalty.

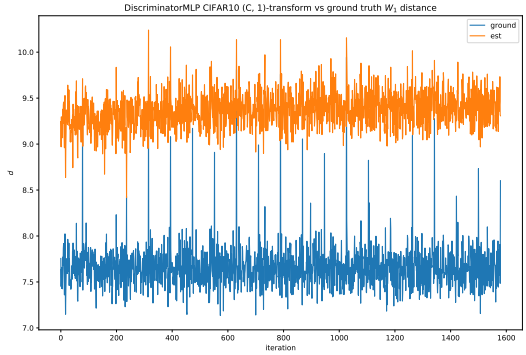

Figure 11. MLP, CIFAR-10, (c, ϵ)-transform.

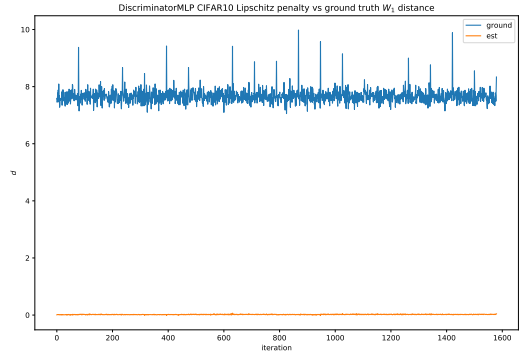

Figure 14. MLP, CIFAR-10, Lipschitz penalty.

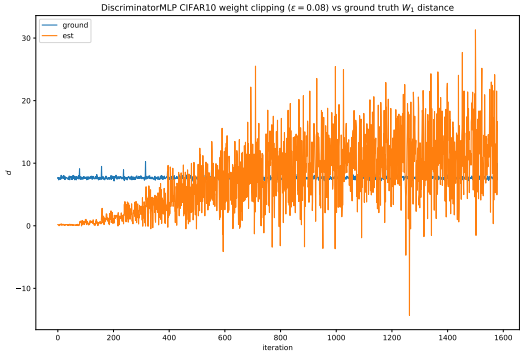

Figure 15. MLP, CIFAR-10, weight clipping.

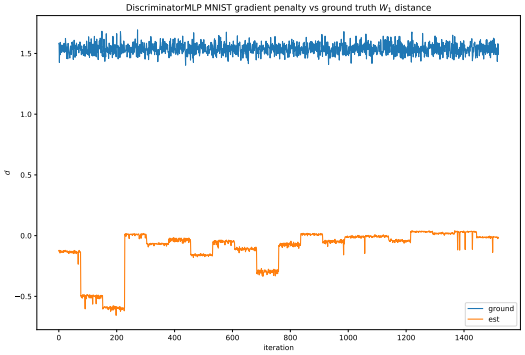

Figure 18. MLP, MNIST, gradient penalty.

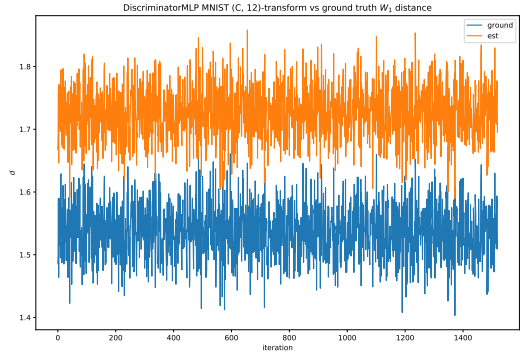

Figure 16. MLP, MNIST, (c, ϵ)-transform.

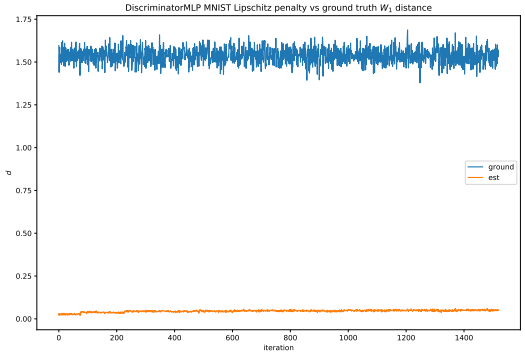

Figure 19. MLP, MNIST, Lipschitz penalty.

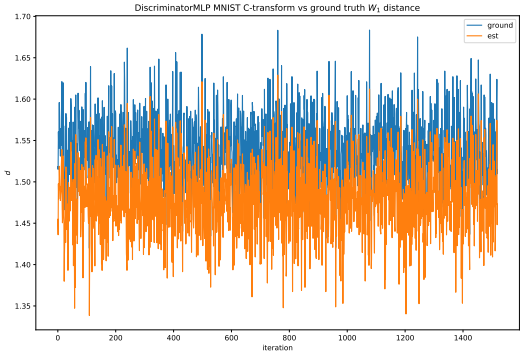

Figure 17. MLP, MNIST, c-transform.

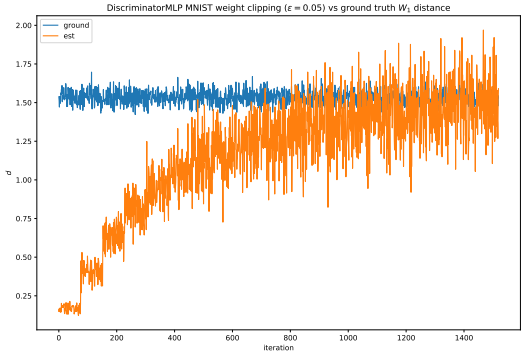

Figure 20. MLP, MNIST, weight clipping.

B. Team member's contributions

Contributions of each team member to the final project.

Denis Rollov ($33.(3)$**% of work)**

- Research on the topic and related works.
- Building NN's architecture.
- Code architecture building and debugging.
- Text and presentation.

Nikolay Goncharov ($33.(3)$**% of work)**

- Research on the topic and related works.
- Ground truth estimation implementing.
- Losses implementation
- Computing the errors and their confidence intervals.

Svetlana Gabdullina ($33.(3)$**% of work)**

- Research on the topic and related works.
- Losses implementation
- Report preparation.
- Text and presentation.

C. 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: None

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: Only well known datasets were used.

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