Generative adversarial learning of Sinkhorn algorithm initializations

Jonathan Geuter,[1] Vaios Laschos [2]

submitted: December 2, 2022

[1] Department of Mathematics TU- Berlin
    Straße des 17. Juni 136
    10623 Berlin
    Germany
    E-Mail: jonathan.geuter at gmx.de

[2] Weierstrass Institute
    Mohrenstr. 39
    10117 Berlin
    Germany
    E-Mail: vaios.laschos@wias-berlin.de

Generative adversarial learning of Sinkhorn algorithm initializations

Jonathan Geuter, Vaios Laschos

Abstract

The Sinkhorn algorithm [Cut13] is the state-of-the-art to compute approximations of optimal transport distances between discrete probability distributions, making use of an entropically regularized formulation of the problem. The algorithm is guaranteed to converge, no matter its initialization. This lead to little attention being paid to initializing it, and simple starting vectors like the n-dimensional one-vector are common choices. We train a neural network to compute initializations for the algorithm, which significantly outperform standard initializations. The network predicts a potential of the optimal transport dual problem, where training is conducted in an adversarial fashion using a second, generating network. The network is universal in the sense that it is able to generalize to any pair of distributions of fixed dimension. Furthermore, we show that for certain applications the network can be used independently.

1 Introduction

Optimal Transport [Vil09, PC19] plays an increasing role in various areas. Besides economics [Gal16], it thrives in machine learning applications, and has been used in domain adaptation [CFHR17], single-cell genomics [SST+19], imitation learning [DHGP20], imaging [SHB+18] and signal processing [KPT+17]. The discrete optimal transport problem can be solved as a linear program; however, this approach proves to be prohibitively expensive, particularly in high dimensions. Adding an entropic regularizer to the problem, one can solve it using the well-known Sinkhorn algorithm [Cut13], which is computationally efficient, more robust to outliers, differentiable, and easily parallelizable. Furthermore, it is guaranteed to converge to the solution of the entropic problem, which is unique by strict convexity of the problem. Hence, little attention has been paid to initializing the Sinkhorn algorithm. We will show that good initializations can speed up the algorithm significantly. To this end, we learn initializations to the algorithm with a neural network, which, given two distributions, predicts a potential of the optimal transport dual problem, which is closely linked to the limit point of the Sinkhorn algorithm. In Section 5, we will see that for certain problems, the network can also be used independently as an approximation function of optimal dual potentials. Importantly, this approach preserves all the important advantages of the Sinkhorn algorithm, such as efficiency, differentiability, and parallelizability.

Training will be supervised, where the true potential is computed for each training sample. The crucial question is: What is the best way to generate training samples? The training dataset needs to be rich enough to allow the network to generalize to any dataset during testing. We tackle this issue with a two-network approach, where one network (the *generator*) learns to generate training samples from a Gaussian prior while the other network (the *approximator*) learns to predict dual potentials given the generating network's outputs. The generator, denoted by g_θ with parameters θ, is a one-layer ResNet [HZRS16], and the approximator, denoted by h_ϕ with parameters ϕ, is a three-layer fully connected network. The networks will be trained in an adversarial fashion, similar to a GAN, where the generator's

loss is negative the approximator's loss. The generator and approximator will be trained in alternation, such that the generator consistently aims at producing those samples which the approximator has most problems with. To our knowledge, this is the first universal approach to initializing the Sinkhorn algorithm.

2 Related Work

Initializing Sinkhorn. There exists very little literature on initializing the Sinkhorn algorithm. [TC22] propose using dual vectors recovered from the unregularized 1D optimal transport problem, or from known transport maps in a Gaussian setup, and were able to significantly speed up convergence. In [ACLR22], a learned approach is taken as well. However, the authors restrict themselves to the case of particular datasets such as MNIST, on which both training and testing is performed. They also do not use a generator in their training. Furthermore, the loss function they use for the approximator on a sample of two distributions (μ, ν) is

$$\text{loss}(\mu, \nu) = -(\langle \text{net}(\mu, \nu), \mu \rangle + \langle \text{net}(\mu, \nu)^C, \nu \rangle),$$

where $\text{net}(\mu, \nu)^C$ denotes the C-transform (cmp. [Vil09]), C being the cost matrix; i.e., they try to maximize the optimal transport dual, cf. section 3. This approach is more elegant than ours in the sense that it allows for unsupervised training, as no ground-truth dual potentials are needed in the training data. However, as we will see in section 5, it it significantly worse at learning to approximate potentials than our supervised approach.

Generative Modelling. In generative modelling, the goal is usually to find a parametrized distribution ρ_θ which minimizes, in some metric, the distance to a target distribution ρ_{data}. Typically, this means minimizing some functional $F(\rho_\theta) = d(\rho_\theta, \rho_{\text{data}})$, where d measures the discrepancy between the distributions, and samples from ρ_{data} are usually available. While our generator also tries to learn such a parametrized distribution ρ_θ, our approach differs from this framework in that we do not have access to a given target distribution. Another similarity can be found in our loss function, which resembles the loss function of Generative Adversarial Networks [GPAM+14], or GANs for short. Given samples $\mathbf{z} \sim \rho_{\mathbf{z}}$ from a (Gaussian) prior and samples $\mathbf{x} \sim \rho_{\text{data}}$ from the target distribution, the GAN loss is

$$\min_G \max_D \mathbb{E}_{\mathbf{x} \sim \rho_{\text{data}}} \left[\log D(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z} \sim \rho_{\mathbf{z}}} \left[\log(1 - D(G(\mathbf{z})))\right],$$

where G is the generator and D the so-called *discriminator*, which predicts the probability that a sample came from the target distribution rather than the generator. Similarly, our loss will be of the form

$$\max_\theta \min_\phi \mathbb{E}_{\mathbf{z} \sim \rho_{\mathbf{z}}} \left[\text{MSE}\left(h_\phi(g_\theta(\mathbf{z})), \mathbf{f}_{g(\mathbf{z})}\right)\right],$$

where $\mathbf{f}_{g(\mathbf{z})}$ denotes a dual potential of the sample $(\mu, \nu) = g(\mathbf{z})$, and MSE denotes the mean squared error. For more details, see section 4. Within the area of generative modelling, a fast-growing line of work is that of *Normalizing Flows* [KPB21, PNR+21], which are compositions of parametrized, invertible transformations pushing a (typically Gaussian) probability distribution to a target distribution. Our approach shares some common characteristics; however, we do not need invertibility, which is crucial to Normalizing Flows. This allows us to simply use a (non-invertible) ResNet as a generator instead.

3   Optimal Transport

In this section, we recall some properties of optimal transport in the discrete case. We will write vectors in bold and matrices as capital letters. By $\llbracket n \rrbracket$ we refer to the set $\{1, 2, ..., n\}$. By $1_n \in \mathbb{R}^n$ we denote the vector where all entries are equal to 1. The $n-1$ dimensional simplex in $\mathbb{R}^n$ will be denoted by Δ^{n-1}, and all elements in the simplex with positive entries are denoted by $\Delta^{n-1}_{>0}$. In the following, let μ and ν be two discrete, m- resp. n-dimensional probability measures on some spaces $\mathcal{X} = \{x_1, ..., x_m\}$ and $\mathcal{Y} = \{y_1, ..., y_n\}$ equipped with the discrete topologies. We will oftentimes abuse notation by considering μ and ν to be the vectors $\begin{bmatrix} \mu_1 & \dots & \mu_m \end{bmatrix}^\top \in \Delta^{m-1}$ resp. $\begin{bmatrix} \nu_1 & \dots & \nu_n \end{bmatrix}^\top \in \Delta^{n-1}$.

3.1   Unregularized Optimal Transport

The discrete optimal transport problem, also referred to as the *Kantorovich problem*, is defined as follows.

Problem 1 (Optimal Transport Problem)**.**

$$L(\mu, \nu) := \min_{\Gamma \in \Pi(\mu, \nu)} \langle C, \Gamma \rangle$$

Here, $\Pi(\mu, \nu)$ denotes the set of all *transport plans* between μ and ν, i.e. matrices $\Gamma \in \mathbb{R}^{m \times n}_{\geq 0}$ s.t. $\Gamma 1_n = \mu$ and $\Gamma^\top 1_m = \nu$. The problem has a dual formulation:

Problem 2 (Dual Optimal Transport Problem)**.**

$$D(\mu, \nu) := \max_{\substack{\mathbf{f} \in \mathbb{R}^m, \ \mathbf{g} \in \mathbb{R}^n \\ \mathbf{f} + \mathbf{g} \leq C}} \langle \mathbf{f}, \mu \rangle + \langle \mathbf{g}, \nu \rangle$$

Here, $\mathbf{f} + \mathbf{g} \leq C$ is to be understood as $\mathbf{f}_i + \mathbf{g}_j \leq C_{ij}$ for all $i \in \llbracket m \rrbracket, j \in \llbracket n \rrbracket$. In the special case where $\mathcal{X} = \mathcal{Y}$ and C corresponds to a metric, i.e. $C_{ij} = d(x_i, y_j)$, the *Wasserstein distance of order p between μ and ν* for $p \in [1, \infty)$ is defined as:

$$W_p(\mu, \nu) = \left(\min_{\gamma \in \Pi(\mu, \nu)} \sum_{i,j} C_{ij}^p \Gamma_{ij} \right)^{\frac{1}{p}}.$$

3.2   Entropic Optimal Transport

A common regularization of the problem consists of adding an entropic regularizer. We define entropy as follows:

Definition 3 (Entropy)**.** *For a matrix $P = [p_{ij}]_{ij} \in \mathbb{R}^{m \times n}$, we define its* entropy $H(P)$ *as*

$$H(P) := -\sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}(\log p_{ij} - 1)$$

if all entries are positive, and $H(P) := -\infty$ if at least one entry is negative. For entries $p_{ij} = 0$, we use the convention $0 \log 0 = 0$, as $x \log x \xrightarrow{x \to 0} 0$.

The entropic optimal transport problem is defined as follows.

Problem 4 (Entropic Optimal Transport Problem)**.** *For $\varepsilon > 0$, the* entropic optimal transport problem *is defined as:*

$$L^{\varepsilon}(\mu, \nu) := \min_{\Gamma_{\varepsilon} \in \Pi(\mu, \nu)} \langle C, \Gamma_{\varepsilon} \rangle - \varepsilon H(\Gamma_{\varepsilon}).$$

The term $-\varepsilon H(\Gamma_{\varepsilon})$ is referred to as the entropic regularizer*, and ε as the* regularizing constant*.*

Note that this is identical to the regular optimal transport problem, except that the regular one does not contain the regularization term $-\varepsilon H(\Gamma)$. As the objective in Problem 4 is ε-strongly convex, the problem admits a unique solution (see [PC19]).

The *Gibbs kernel* is defined as $K = \exp(-C/\varepsilon)$. Then the entropic dual problem reads:

Problem 5 (Entropic Dual Problem)**.** *The entropic dual problem is defined as:*

$$D^{\varepsilon}(\mu, \nu) := \max_{\mathbf{f}_{\varepsilon} \in \mathbb{R}^m, \ \mathbf{g}_{\varepsilon} \in \mathbb{R}^n} \langle \mathbf{f}_{\varepsilon}, \mu \rangle + \langle \mathbf{g}_{\varepsilon}, \nu \rangle - \varepsilon \left\langle e^{\mathbf{f}_{\varepsilon}/\varepsilon}, K e^{\mathbf{g}_{\varepsilon}/\varepsilon} \right\rangle.$$

Again, without the regularization term $-\varepsilon \left\langle e^{\mathbf{f}_{\varepsilon}/\varepsilon}, K e^{\mathbf{g}_{\varepsilon}/\varepsilon} \right\rangle$, this equals the regular optimal transport dual; note, however, that the unregularized dual is subject to the constraint $\mathbf{f} + \mathbf{g} \leq C$. The following proposition holds (see, e.g., [PC19]).

Proposition 6. *The unique solution of Problem 4 is given by*

$$\Gamma_{\varepsilon} = diag(\mathbf{u}) K diag(\mathbf{v})$$

for two positive scaling vectors $\mathbf{u}$ and $\mathbf{v}$ unique up to a scaling constant (i.e. $\lambda \mathbf{u}$, $\frac{1}{\lambda}\mathbf{v}$ for $\lambda > 0$). Furthermore, $(\mathbf{u}, \mathbf{v})$ are linked to the solution $(\mathbf{f}_{\varepsilon}, \mathbf{g}_{\varepsilon})$ from Problem 5 via

$$(\mathbf{u}, \mathbf{v}) = (\exp(\mathbf{f}_{\varepsilon}/\varepsilon), \exp(\mathbf{g}_{\varepsilon}/\varepsilon)).$$

A solution to the entropic dual can be approximated by a solution to the regular dual in the following sense.

Proposition 7. *Let $(\mathbf{f}, \mathbf{g})$ be optimal for the unregularized dual problem and $(\mathbf{f}^{\varepsilon}, \mathbf{g}^{\varepsilon})$ be optimal for the regularized dual problem for some $\varepsilon > 0$. Then $(\mathbf{f}^{\varepsilon}, \mathbf{g}^{\varepsilon})$ is feasible for the unregularized problem, i.e. $\mathbf{f}^{\varepsilon} + \mathbf{g}^{\varepsilon} \leq C$, and*

$$0 \leq D^{\varepsilon}(\mu, \nu) - \left[\langle \mathbf{f}, \mu \rangle + \langle \mathbf{g}, \nu \rangle - \varepsilon \left\langle e^{\mathbf{f}/\varepsilon}, K e^{\mathbf{g}/\varepsilon} \right\rangle \right] \leq mn\varepsilon,$$

i.e. the value the entropic dual takes at $(\mathbf{f}, \mathbf{g})$ differs from the optimal value by at most a factor of $mn\varepsilon$. In particular, if $\varepsilon \to 0$, the optimum of the entropic dual converges to its value at $(\mathbf{f}, \mathbf{g})$.

A proof can be found in the appendix.

Algorithm 1 Sinkhorn Algorithm

1: **in** $C \in \mathbb{R}^{m \times n}, \varepsilon > 0, \mu \in \Delta_{>0}^{m-1}, \nu \in \Delta_{>0}^{n-1}$
2: initialize $\mathbf{v}^0$ (e.g. $\mathbf{v}^0 \leftarrow 1_n$), $l \leftarrow 0$, $K \leftarrow \exp(-C/\varepsilon)$
3: **repeat**
4:      $\mathbf{u}^{l+1} \leftarrow \mu./K\mathbf{v}^l$
5:      $\mathbf{v}^{l+1} \leftarrow \nu./K^\top \mathbf{u}^{l+1}$
6:      $l \leftarrow l + 1$
7: **until** stopping criterion is met
8: $\Gamma \leftarrow \mathrm{diag}(\mathbf{u}^l)K\mathrm{diag}(\mathbf{v}^l)$
9: **out** Γ, $\langle C, \Gamma \rangle$

3.3 Sinkhorn Algorithm Initializations

The Sinkhorn algorithm – see Algorithm 1 – is an iterative procedure based on the original work of Sinkhorn and Knopp [SK67]. It was first applied to the optimal transport setting in the seminal work *Sinkhorn distances: lightspeed computation of optimal transport* [Cut13].

In the algorithm, $./$ is to be understood as element-wise division. Note that the algorithm requires both input distributions to be positive everywhere to prevent division by zero. As Sinkhorn and Knopp showed in their original work, the iterates $\mathbf{u}^l$ and $\mathbf{v}^l$ from the algorithm converge to the vectors $\mathbf{u}$ and $\mathbf{v}$ from Proposition 6. By the same proposition, we know that $\mathbf{v} = \exp(\mathbf{g}_\varepsilon/\varepsilon)$ for a solution $\mathbf{g}_\varepsilon$ of the regularized dual, and by Proposition 7 we know that we can approximate $\mathbf{g}_\varepsilon$ by a solution $\mathbf{g}$ of the unregularized dual. Hence, we can make a neural network learn such $\mathbf{g}$ to initialize the Sinkhorn algorithm via $\mathbf{v}^0 = \exp(\mathbf{g}/\varepsilon)$. Furthermore, optimal $(\mathbf{f}, \mathbf{g})$ of the unregularized dual are linked via $\mathbf{g} = \mathbf{f}^C$, where $\mathbf{f}^C$ is the C-*transform* of $\mathbf{f}$, defined via $\mathbf{g}_j = \min_i C_{ij} - \mathbf{f}_i$. As both $\mathbf{f}$ and $\mathbf{g}$ are C-*concave* functions – meaning they are the C-transform of some other function, in this case of each other – whenever they're optimal (see [Vil09]), we can also instead learn $\mathbf{f}$ and compute $\mathbf{g}$ from it via $\mathbf{g} = \mathbf{f}^C$. This has the advantage that in enforces C-concavity of $\mathbf{g}$.[1] In the following section, we will see how this learning is performed.

4 Learning Initializations

We will train and test the algorithm on 28×28-dimensional images, i.e. 784-dimensional distributions. We will always use the squared euclidean distance in the unit square as the cost function for our experiments; however, other cost functions could also be considered. Our choice yields the squared Wasserstein-2 distance as the optimal transport cost. Hyperparameter values can be found in the appendix. All code is available at `https://github.com/j-geuter/SinkhornNNHybrid`.

4.1 Generator

The generator is a one-layer ResNet-like network, where inputs $\mathbf{z}$ come from a 128-dimensional Gaussian prior, $\mathbf{z} \sim \rho_{\mathbf{z}}$. This yields an 8×8 input image for each of the two distributions. The output

---

[1]One could also make the network learn $\mathbf{f}_\varepsilon$ directly. However, then one is faced with the problem of having to choose a regularizing constant ε for training, while during testing, this constant might change; also, as the entropic problem is only solved approximately by the Sinkhorn algorithm, one would have to use ground truth labels for $\mathbf{f}_\varepsilon$ during training that are mere approximations. Empirically, learning $\mathbf{f}$ works slightly better.

of the generator is

$$(\mu, \nu) = g_\theta(\mathbf{z}) = \lambda \text{ReLU}(T(\mathbf{z})) + \text{net}_{g_\theta}(\mathbf{z}) + c,$$

where T transforms the 8×8-dimensional input images to 28×28-dimensional images via interpolation, and λ is a constant controlling the impact of the skip connection. The network net_{g_θ} is a single fully connected linear layer with ReLU activation. The small, positive constant c ensures that all inputs are active during training. Apart from vastly improving learning, this is not even restricting the samples as the Sinkhorn algorithm requires distributions to be positive anyways. In the end, $g_\theta(\mathbf{z})$ will be normalized such that it contains two distributions which both sum to one. In Figure 1, images generated by g after no training, training on 50k unique samples, and training on 100k unique samples are shown.

4.2   Approximator

The approximator is a three-layer fully connected network, where the first layer has $2 \cdot 784$-dimensional in- and $6 \cdot 784$-dimensional output, the second layer has $6 \cdot 784$-dimensional in- and output, and the last layer has $6 \cdot 784$-dimensional in- and 784-dimensional output. The first two layers contain ReLU activations and batch normalizations, while the last layer has neither.

4.3   Training

As we have seen before, our training objective is

$$\max_\theta \min_\phi \mathbb{E}_{\mathbf{z} \sim \rho_\mathbf{z}} \left[\text{MSE} \left(h_\phi(g_\theta(\mathbf{z})), \mathbf{f}_{g(\mathbf{z})} \right) \right],$$

where $\mathbf{f}_{g(\mathbf{z})}$ denotes a dual potential of the sample $(\mu, \nu) = g(\mathbf{z})$. Note that it does not have a θ subscript, because this is the target value and we do not backpropagate through it.

Algorithm 2 Training Algorithm

1: **in** cost $C \in \mathbb{R}^{n \times n}$, prior $\rho_\mathbf{z}$, learning rates α, β, epochs
2: **for** $i = 1, 2, \dots$ until stopping criterion **do**
3:      $\mathbf{z} \leftarrow \text{sample}(\rho_\mathbf{z})$
4:      $(\mu, \nu) \leftarrow g_\theta(\mathbf{z})$
5:      $\mathbf{f} \leftarrow \text{DualPotential}((\mu, \nu), C)$
6:      $\mathbf{f} \leftarrow \mathbf{f} - \frac{\sum_i \mathbf{f}_i}{n}$
7:      **for** $e = 1, 2, \dots,$ epochs **do**
8:          $\mathbf{f}_\star \leftarrow h_\phi((\mu, \nu))$
9:          $\phi \leftarrow \phi - \alpha_i \nabla_\phi \text{MSE}(\mathbf{f}_\star, \mathbf{f})$
10:         $\mathbf{f}_\star^{\mathbf{C}} \leftarrow h_\phi((\nu, \mu))$
11:         $\phi \leftarrow \phi - \alpha_i \nabla_\phi \text{MSE}(\mathbf{f}_\star^C, \mathbf{f}^C)$
12:     **end for**
13:     $\theta \leftarrow \theta + \beta_i \nabla_\theta \text{MSE}(\mathbf{f}_\star, \mathbf{f})$
14: **end for**

Furthermore, as discussed in Section 3, optimal potentials $(\mathbf{f}, \mathbf{g})$ are C-transforms of one another. Hence, if the network's output of (μ, ν) is $\mathbf{f}$, its output of (ν, μ) should be $\mathbf{f}^C$. This allows us to easily double the available training data. Additionally, it drives the network towards being more "C-symmmetricïn its inputs. Another property of the optimal transport dual is that its value is invariant
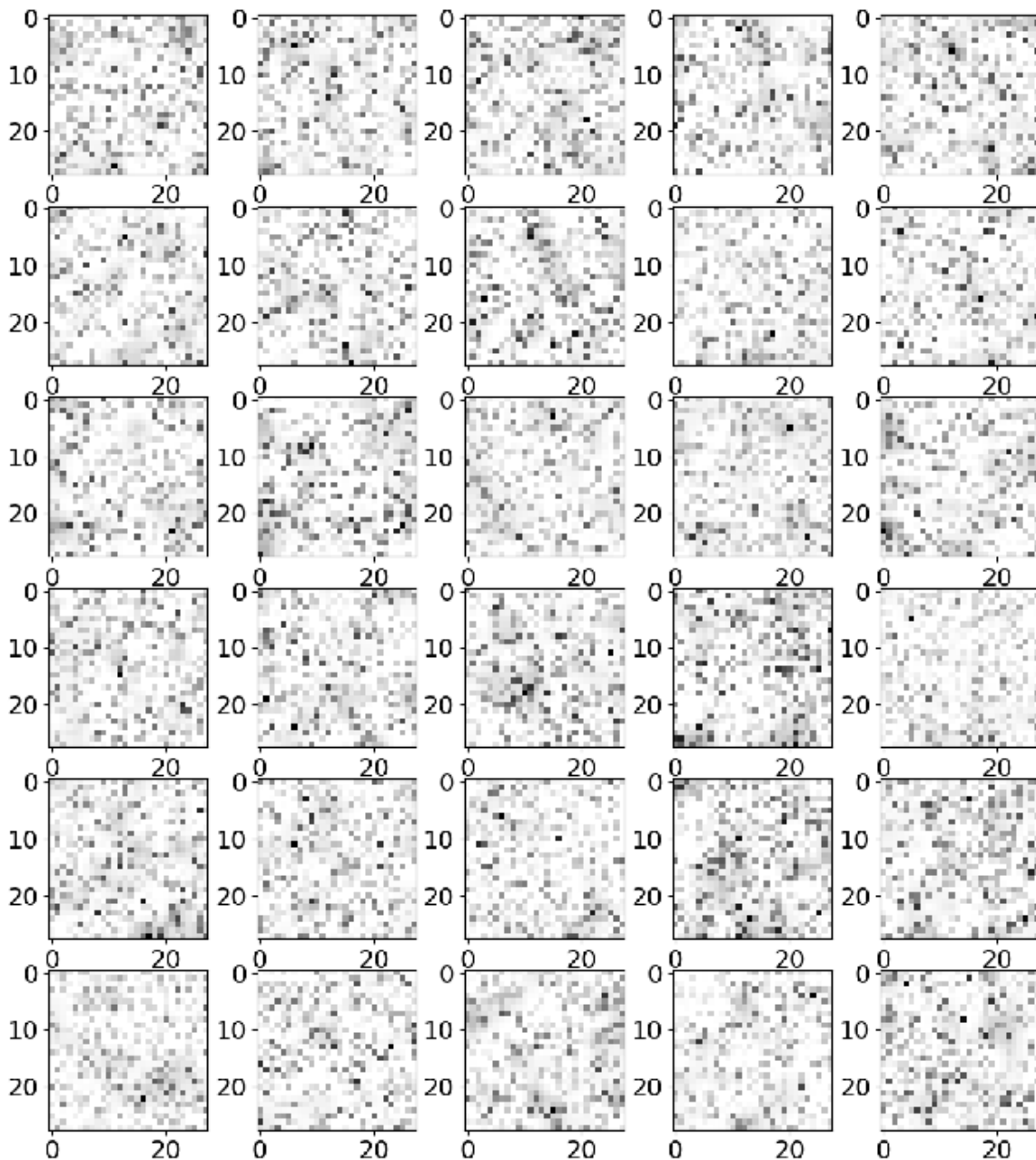
Figure 1: Generated images after no training (top two rows), training on 50k unique samples (middle two rows), and on 100k unique samples (bottom two rows), for 5 epochs resp. For all three, the first row corresponds to μ and the second row to ν.

under adding a constant to $\mathbf{f}$ and subtracting it from $\mathbf{g}$; cf. [PC19]. Hence, to make learning more stable, we will only consider dual potentials that sum to 0. The training algorithm can be seen in Algorithm 2. In practice, we use the Adam optimizer for updating parameters.

De facto, $\mathbf{z}$ in Algorithm 2 will contain a batch of samples. DualPotential denotes any algorithm that computes a dual potential for the unregularized problem; in practice, we use `ot.emd` from the POT package [FCG$^+$21]. Also, as computing these potentials is the most expensive part of the algorithm, we let each batch of samples run through h_ϕ for multiple epochs. During each such epoch, all samples in the batch are shuffled randomly and fed to h_ϕ in minibatches.

5   Experiments

In all plots in this section, we will always plot the 95% confidence intervals alongside the mean values; however, oftentimes, the confidence intervals are too narrow to be visible.

5.1   Test Sets

We test the network on four different test datasets. The first one is generated by assigning each point in the distribution a random value (see the appendix for more details), the second one consists of teddy bears from the Google Quick, Draw! dataset, the third one is MNIST, and the last one a greyscale version of CIFAR10. Figure 2 shows the test datasets.
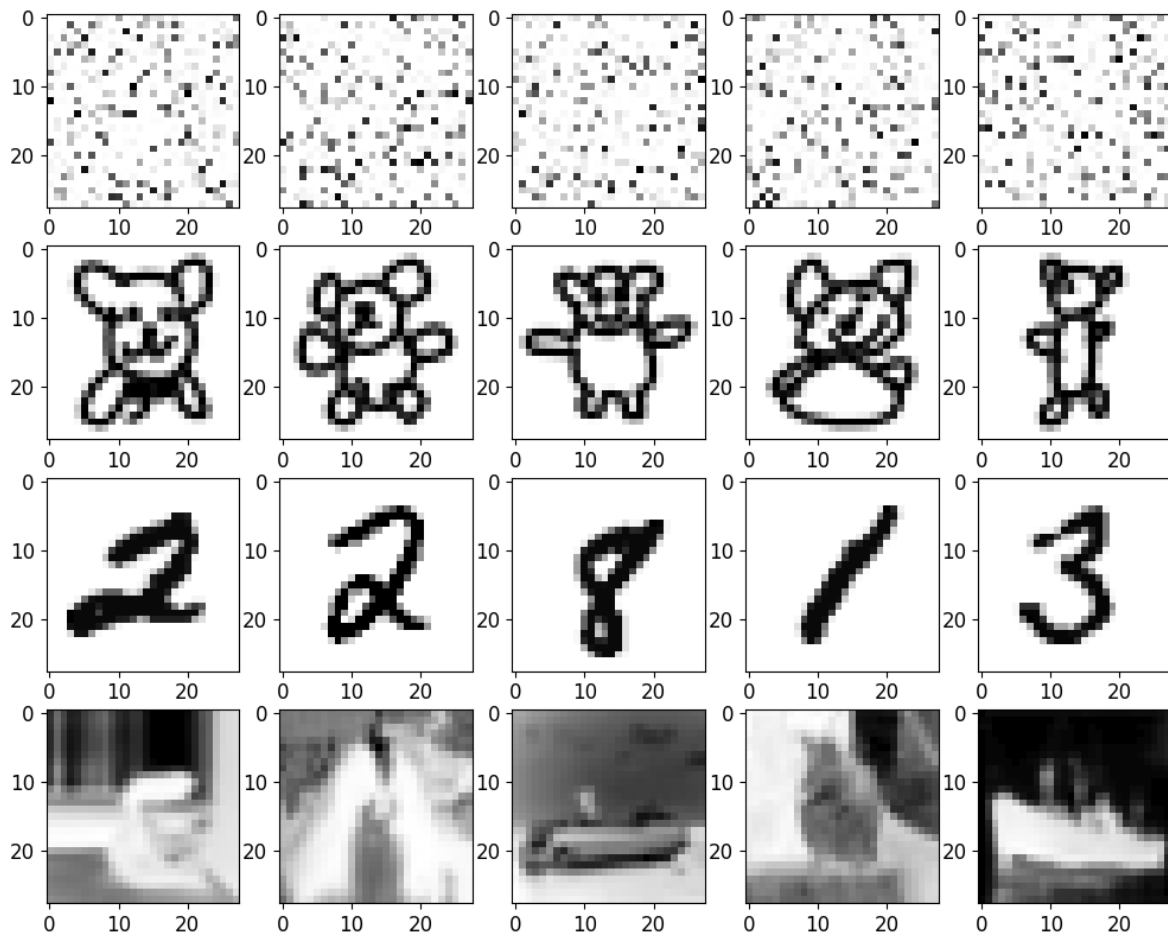

Figure 2: Test datasets 'random', 'teddies', 'MNIST' and 'CIFAR'.

5.2   Loss Function Comparison

As discussed in Section 2, [ACLR22] use a loss on the transport distance. In Figure 3, we can see how training compares to our loss function.[2]

---

[2]In this figure, the number of samples on the x-axis refers to the number of unique samples multiplied by the number of epochs.
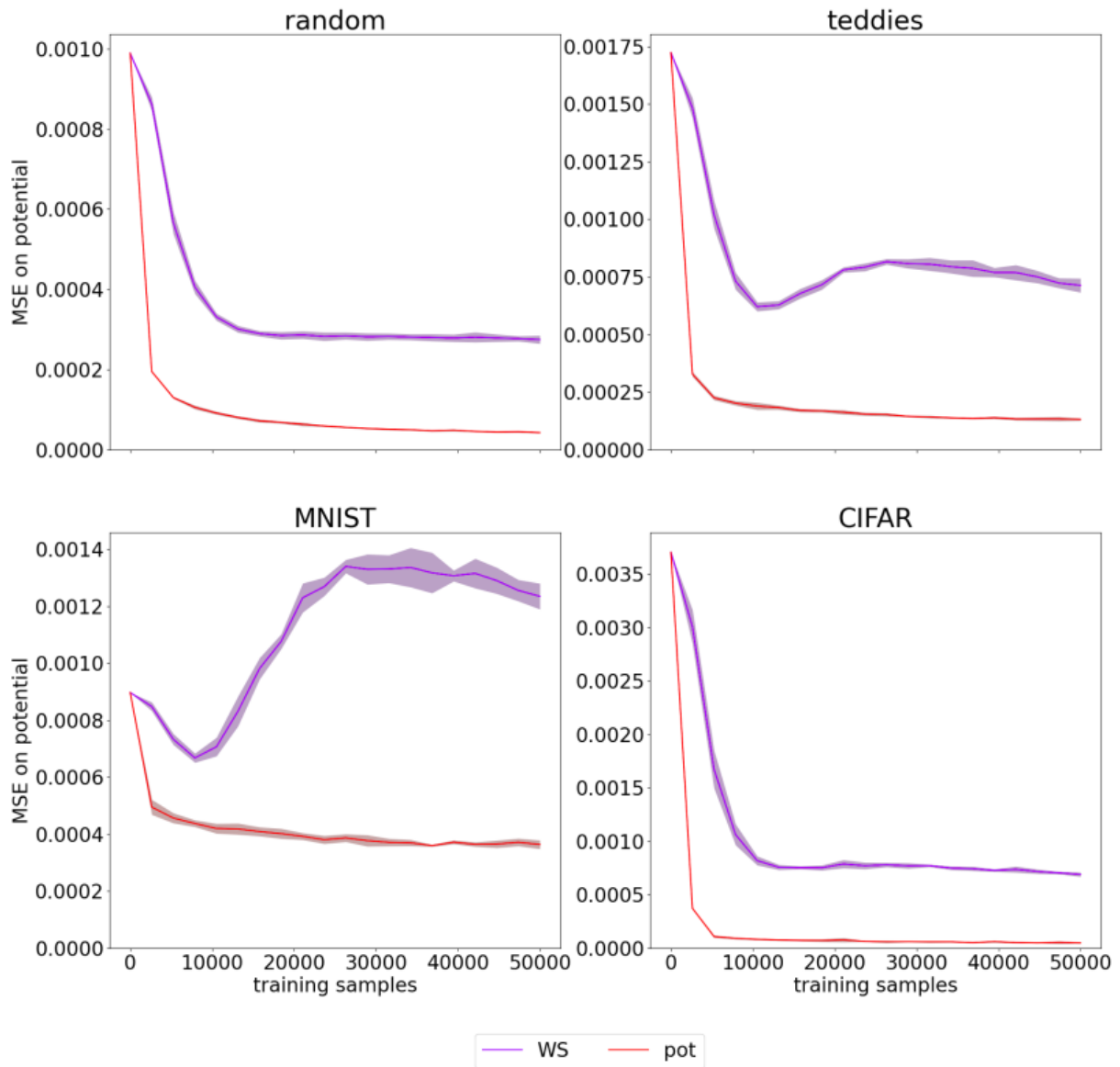
Figure 3: Comparison of loss functions on the transport distance ('WS') vs. on the potential ('pot').

5.3 Initialization Performance

We compare the Sinkhorn algorithm convergence for our learned initialization to the default one (i.e. $1_{28} \in \mathbb{R}^{28}$). We initialize it as outlined in section 3.3; however, to prevent entries from being too small or two large, we bound v^0 from below by $1e\text{-}35$ and from above by $1e35$. This means for input measures μ and ν, we set

$$v^0 \leftarrow \max \left\{ 1e\text{-}35, \min \left\{ 1e35, \exp(h_\phi(\mu, \nu)^C / \varepsilon) \right\} \right\}.$$

The network is trained on 100.000 unique training samples, each of which is trained on epochs $= 5$ times. Training takes just over 3 hours on a NVIDIA Telsa T4 GPU with 16 GB; however, we note that almost all training time is needed to compute ground truth potentials with `ot.emd`. In all experiments, we set $\varepsilon = 0.00025$. Figures 4 and 5 show the relative error on the transport distance, with respect to the number of Sinkhorn iterations and the computation time resp.

As one can see from the figures, the two plots look very similar; this is because the time needed to
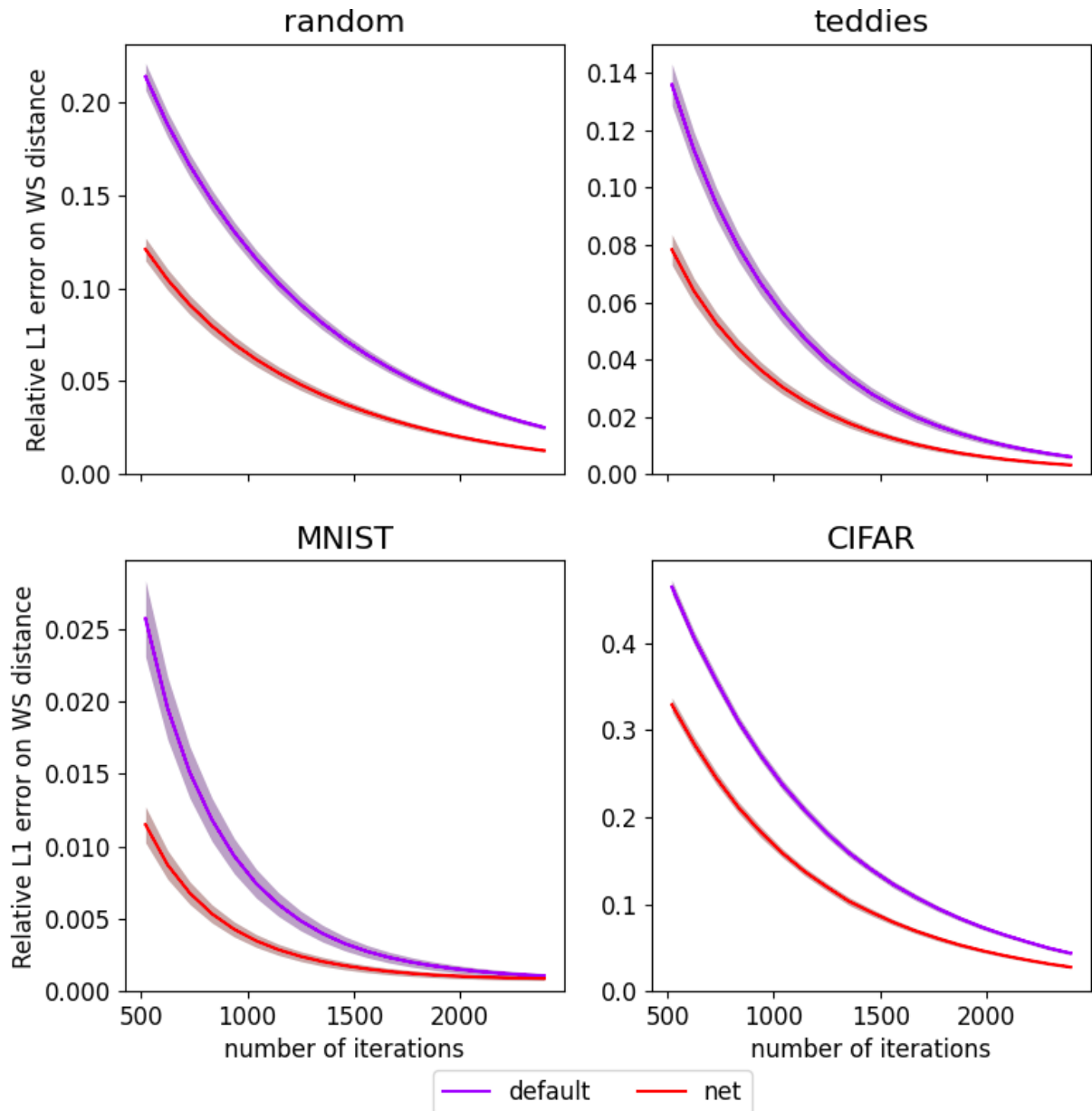
Figure 4: Relative errors on the transport distances w.r.t. the number of Sinkhorn iterations.

compute the initialization vector is negligible compared to the time needed by the Sinkhorn algorithm.

However, in practice, one does not have access to the true transport distances, hence a loss on the marginal constraint violation is oftentimes used as a stopping criterion for the Sinkhorn algorithm. The marginal constraint violations measure how far the plan $\Gamma_{(l)}^{\varepsilon}$ computed by the Sinkhorn algorithm after l iterations is from fulfilling the two marginal constraints $1_m^{\top}\Gamma_{(l)}^{\varepsilon} = \nu^{\top}$ and $\Gamma_{(l)}^{\varepsilon}1_n = \mu$. Different flavours to measure this violation exist; we use

$$\frac{\left(\left\|1_m^{\top}\Gamma_{(l)}^{\varepsilon} - \nu^{\top}\right\|_1\right) + \left(\left\|\Gamma_{(l)}^{\varepsilon}1_n - \mu\right\|_1\right)}{2}.$$

The average marginal constraint violations with respect to the number of Sinkhorn iterations can be
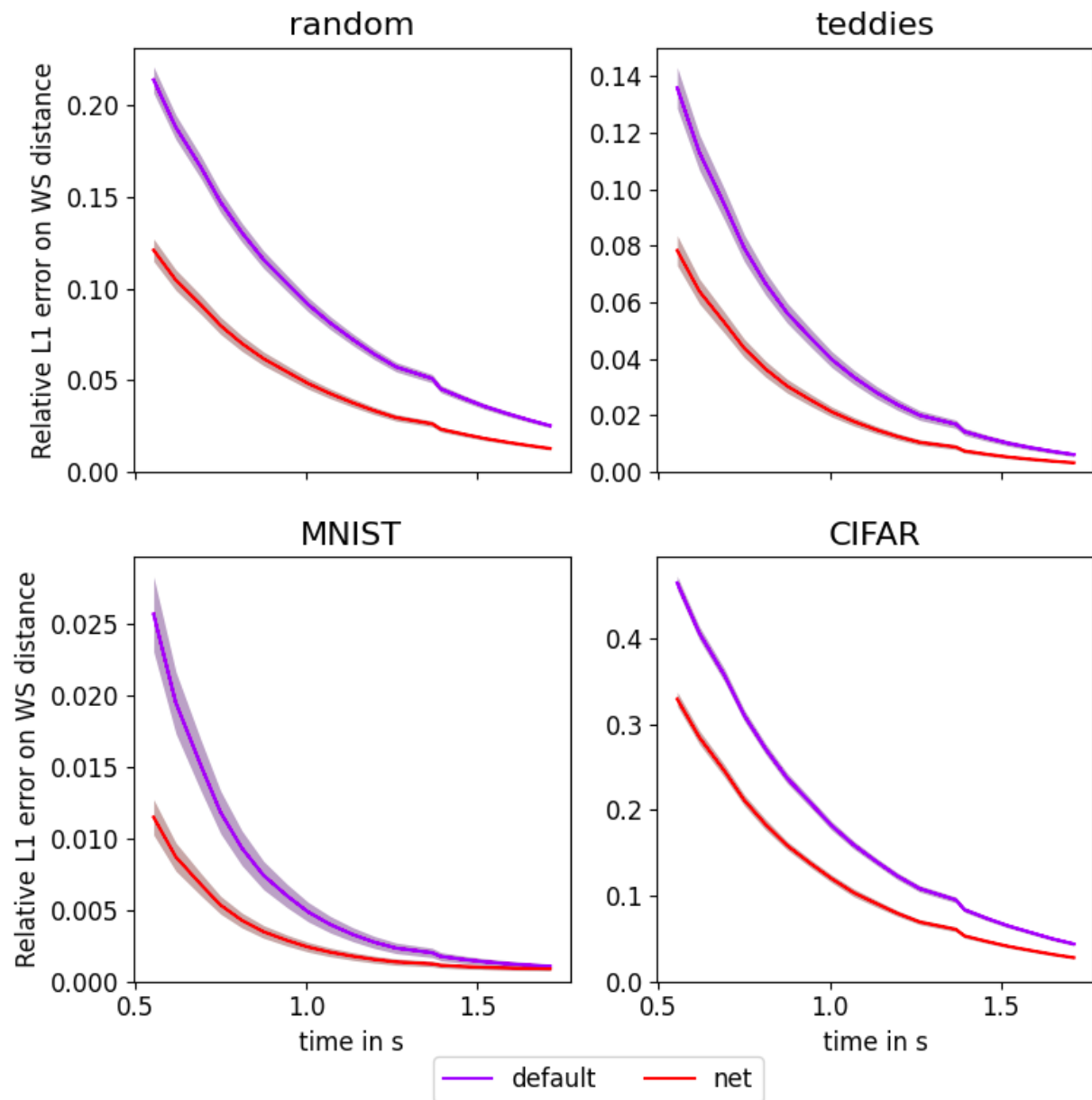
Figure 5: Relative errors on the transport distances w.r.t. computation time.

seen in Figure 6.

In Tables 1 and 2, the average number of iterations needed to achieve $1e\text{-}2$ and $1e\text{-}3$ marginal constraint errors is reported.[3]

---

[3]Measured with an accuracy of 25 Sinkhorn iterations.
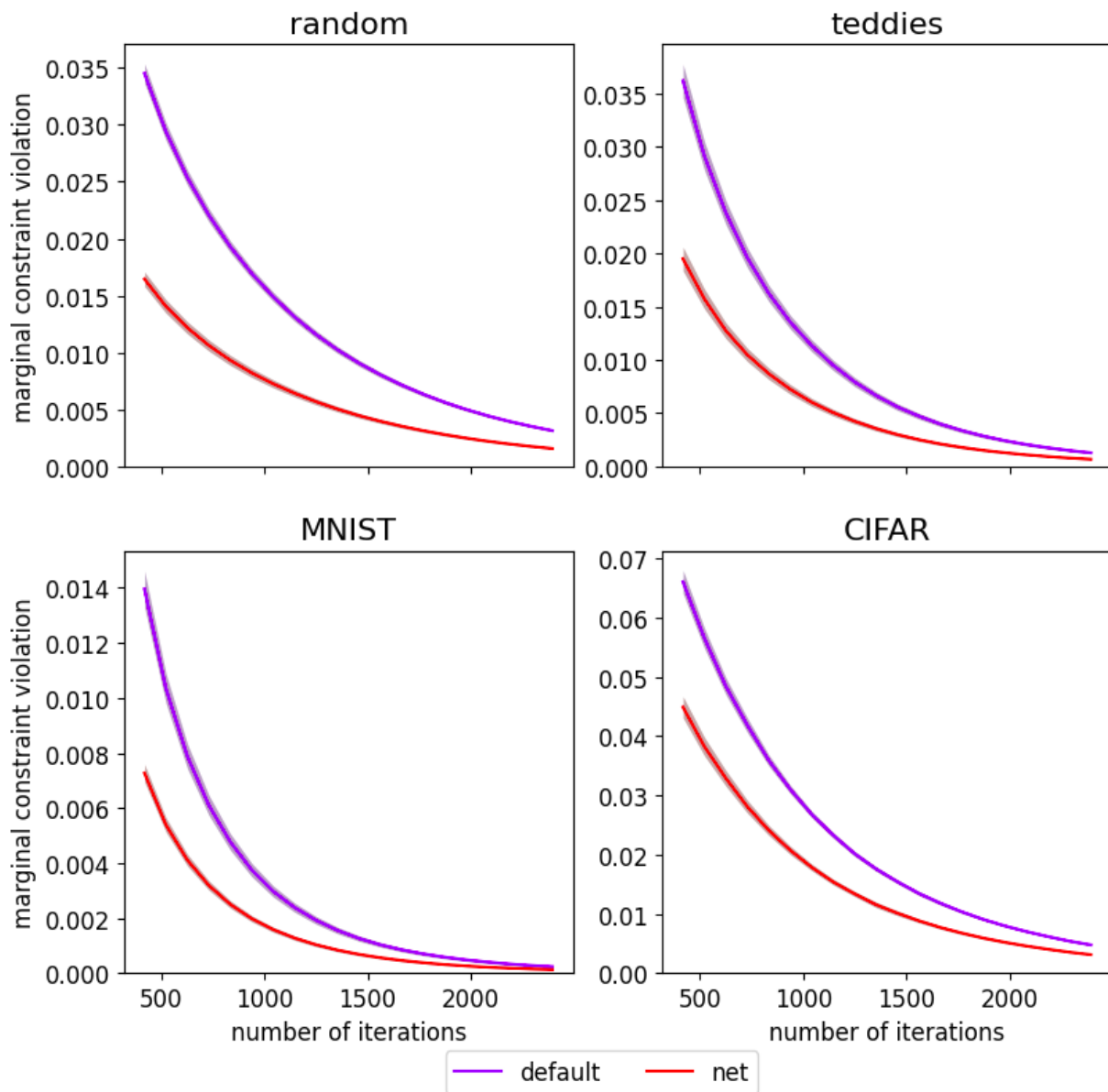
Figure 6: Average marginal constraint violations.

	default	net
random	3355 ± 38	2725 ± 52
teddies	2473 ± 48	2067 ± 49
MNIST	1528 ± 32	1225 ± 23
CIFAR	3555 ± 36	3195 ± 34

Table 1: Average number of iterations needed to achieve a $1e\text{-}3$ marginal constraint error.

	default	net
random	1330 ± 29	755 ± 41
teddies	1083 ± 30	735 ± 39
MNIST	528 ± 17	322 ± 10
CIFAR	1738 ± 20	1422 ± 29

Table 2: Average number of iterations needed to achieve a $1e\text{-}2$ marginal constraint error.

5.4 Wasserstein Barycenters

Wasserstein barycenters are barycenters with respect to the Wasserstein distance. Namely, a Wasserstein barycenter of measures $\{\nu_1, ..., \nu_n\}$ is any measure μ such that

$$\mu = \arg\min_{\mu'} \sum_{i=1}^{n} W_p^p(\mu', \nu_i).$$

By duality, we can compute this as

$$\mu = \arg\min_{\mu'} \sum_{i=1}^{n} \langle \mathbf{f}_i, \mu' \rangle + \langle \mathbf{g}_i, \nu_i \rangle$$

for solutions $(\mathbf{f}_i, \mathbf{g}_i)$ of the dual problem of μ' and ν_i with cost function d^p. In this section, we have a look at how Wasserstein barycenters can be computed using only the approximator. Usually, the dual potential approximations of the approximator are not accurate enough to use them on their own, hence we used them to initialize the Sinkhorn algorithm. However, in applications such as barycenter computations, where the potentials are integrated with respect to the measures, some of the approximation errors tend to cancel out: Assume the true potential is $\mathbf{f}$ and the network's approximation of it is $\mathbf{f} + \sigma$, where $\sigma_i \sim Z_i$ with $\mathbb{E}[Z_i] = 0$, then

$$\mathbb{E}_\sigma[\langle \mathbf{f} + \sigma, \mu \rangle] = \langle \mathbf{f}, \mu \rangle + \mathbb{E}_\sigma \left[\sum_i \sigma_i \mu_i \right] = \langle \mathbf{f}, \mu \rangle.$$

In Figure 7, we can see the barycenters computed with the network only of 20 samples for each of the digits 0, 2, 5, and 7 from the MNIST dataset. They were computed using a simple gradient descent with respect to μ' on the sum

$$\sum_{i=1}^{n} \langle \mathbf{f}_i, \mu' \rangle + \langle \mathbf{g}_i, \nu_i \rangle.$$

6 Summary

We showed that it is possible to learn universal initializations to the Sinkhorn algorithm via the optimal transport dual problem. We used a two-network approach, where one network is used to generate training samples for the second network. Both networks are trained in an adversarial manner similar to GANs. When comparing the convergence speed of the Sinkhorn algorithm for its default initialization vs. the network initialization, the network initialization significantly outperforms the default initialization on all test datasets. This is true both when considering relative errors on the transport distance, as well
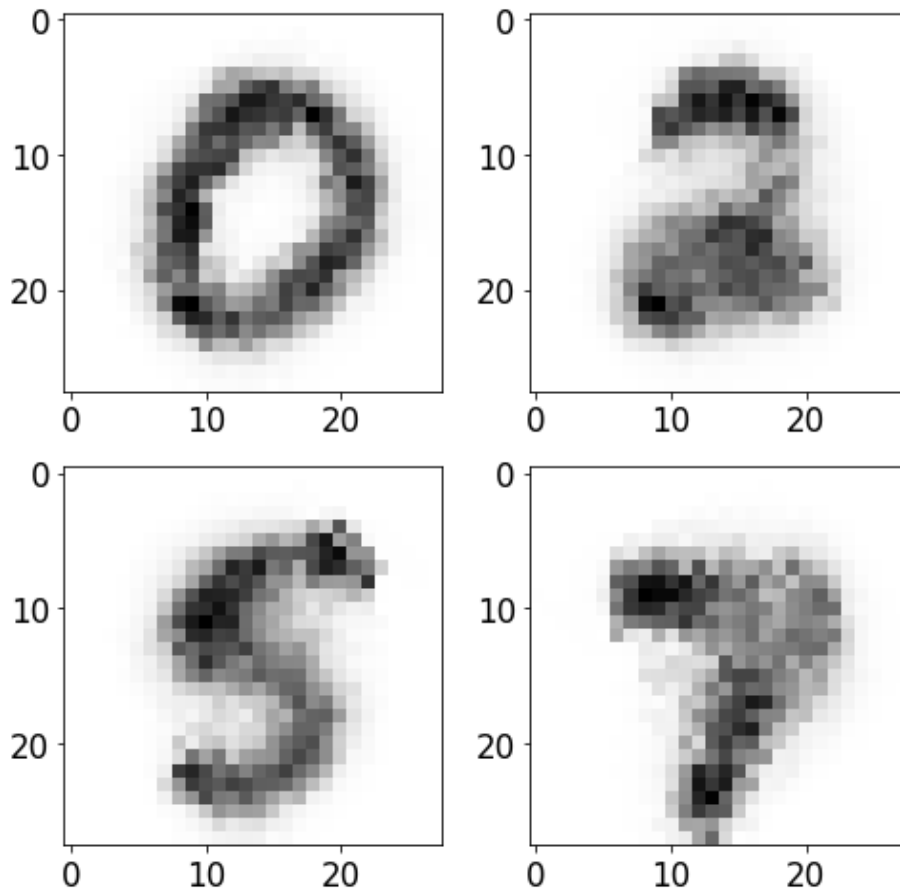
Figure 7: Barycenters of 20 MNIST samples for the digits 0, 2, 5, and 7.

as for the marginal constraint violations. To achieve a marginal constraint violation error of $1e\text{-}3$, the network initialization needs 16% less iterations averaged over all test datasets; to achieve an error of $1e\text{-}2$, it needs 33% less iterations on average, and even up to 43% less depending on the dataset, almost doubling convergence speed. The network is easy to train and universal as it successfully generalizes to arbitrary datasets, and the computation time needed to compute the initialization is negligible compared to the time needed by the Sinkhorn algorithm. Hence, we propose initializing the Sinkhorn algorithm using such a network.

Acknowledgements

References

[ACLR22]   Brandon Amos, Samuel Cohen, Giulia Luise, and Ievgen Redko. Meta optimal transport, 2022.

[CFHR17]   Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *Advances in Neural Information*

Processing Systems, volume 30, 2017.

[Cut13]   Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[DHGP20]   Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning, 2020.

[FCG$^+$21]   Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

[Gal16]   Alfred Galichon. *Optimal Transport Methods in Economics*. Princeton University Press, 2016.

[GPAM$^+$14]   Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[HZRS16]   Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[KPB21]   Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, 2021.

[KPT$^+$17]   Soheil Kolouri, Se Rim Park, Matthew Thorpe, Dejan Slepcev, and Gustavo K. Rohde. Optimal mass transport: Signal processing and machine-learning applications. *IEEE Signal Processing Magazine*, 34(4):43–59, 2017.

[PC19]   Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[PNR$^+$21]   George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[SHB$^+$18]   Morgan A. Schmitz, Matthieu Heitz, Nicolas Bonneel, Fred Ngolè, David Coeurjolly, Marco Cuturi, Gabriel Peyré, and Jean-Luc Starck. Wasserstein dictionary learning: Optimal transport-based unsupervised nonlinear dictionary learning. *SIAM Journal on Imaging Sciences*, 11(1):643–678, jan 2018.

[SK67]   Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2), 1967.

[SST$^+$19]   Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, Brian Cleary, Vidya Subramanian, Aryeh Solomon, Joshua Gould, Siyan Liu, Stacie Lin, Peter Berube, Lia Lee, Jenny Chen, Justin Brumbaugh, Philippe Rigollet, Konrad Hochedlinger, Rudolf Jaenisch, Aviv Regev, , and Eric S. Lander. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943, 2019.

[TC22]   James Thornton and Marco Cuturi. Rethinking initialization of the sinkhorn algorithm, 2022.

[Vil09]   Cédric Villani. *Optimal Transport Old and New*. Springer, 2009.

Appendix A Theory Background

Proof of Proposition 7. In this proof we will write ε as superscripts. Let γ be the solution of the entropic primal problem. As we have

$$1 \geq \Gamma_{ij} = e^{(\mathbf{f}_i^{\varepsilon} + \mathbf{g}_j^{\varepsilon} - C_{ij})/\varepsilon} \text{ for all } i \in [\![m]\!], \ j \in [\![n]\!],$$

it follows that $\mathbf{f}_i^{\varepsilon} + \mathbf{g}_j^{\varepsilon} - C_{ij} \leq 0$ for all i and j, i.e. $\mathbf{f}^{\varepsilon} + \mathbf{g}^{\varepsilon} \leq C$. This makes $(\mathbf{f}^{\varepsilon}, \mathbf{g}^{\varepsilon})$ feasible for the unregularized dual. From optimality of $(\mathbf{f}, \mathbf{g})$ we get

$$\langle \mathbf{f}, \mu \rangle + \langle \mathbf{g}, \nu \rangle \geq \langle \mathbf{f}^{\varepsilon}, \mu \rangle + \langle \mathbf{g}^{\varepsilon}, \nu \rangle.$$

This gives us

$$
\begin{aligned}
&D^{\varepsilon}(\mu, \nu) - \left[\langle \mathbf{f}, \mu \rangle + \langle \mathbf{g}, \nu \rangle - \varepsilon \langle e^{\mathbf{f}/\varepsilon}, K e^{\mathbf{g}/\varepsilon} \rangle \right] \\
=& \langle \mathbf{f}^{\varepsilon}, \mu \rangle + \langle \mathbf{g}^{\varepsilon}, \nu \rangle - \varepsilon \langle e^{\mathbf{f}^{\varepsilon}/\varepsilon}, K e^{\mathbf{g}^{\varepsilon}/\varepsilon} \rangle - \left[\langle \mathbf{f}, \mu \rangle + \langle \mathbf{g}, \nu \rangle - \varepsilon \langle e^{\mathbf{f}/\varepsilon}, K e^{\mathbf{g}/\varepsilon} \rangle \right] \\
\leq& \varepsilon \left[\langle e^{\mathbf{f}/\varepsilon}, K e^{\mathbf{g}/\varepsilon} \rangle - \langle e^{\mathbf{f}^{\varepsilon}/\varepsilon}, K e^{\mathbf{g}^{\varepsilon}/\varepsilon} \rangle \right] \\
\leq& \varepsilon \sum_{i,j} e^{(\mathbf{f}_i + \mathbf{g}_j - C_{ij})/\varepsilon} \leq mn\varepsilon,
\end{aligned}
$$

where in the last step we used the fact that $\mathbf{f} + \mathbf{g} \leq C$. Also note that the starting expression is always greater or equal to 0 by optimality of $(\mathbf{f}^{\varepsilon}, \mathbf{g}^{\varepsilon})$. $\qquad\square$

Appendix B Training Details

Hyperparameters. In the generator, we set $\lambda = 0.3$ and $c = 1e - 2$. The learning rate for the generator is set to 0.2352 initially, while the approximator's learning rate starts at 2.352. Both are updated via

$$
\begin{aligned}
\alpha_i &\leftarrow 0.99 \cdot \alpha_{i-1} \\
\beta_i &\leftarrow 0.99 \cdot \beta_{i-1}.
\end{aligned}
$$

The batches are of size 500, and each minibatch of size 100. The number of epochs is set to 5.

Test datasets. For the 'random' test dataset, each pixel was assigned a value r^3, where r is a random number between 0 and 1, before the distributions where normalized. Additionally, for each sample in all test datasets, a small constant was added to the distribution before normalization such that they did not contain any zeros.

Code. All code used for experiments is available at `https://github.com/j-geuter/SinkhornNNHybrid`.