A. Proof of Theorem 3.3

Define $V_f(g) \triangleq \mathbb{E}_Q[\langle Y, \nabla g(Y)\rangle - f(\nabla g(Y))]$. The main step of the proof is to show that $\sup_{g\in \mathtt{CVX}(Q)} V_f(g) = \mathbb{E}_Q[f^*(Y)]$. Then the conclusion follows from (4). To prove this, note that for all $g \in \mathtt{CVX}(Q)$, we have

$$\langle y, \nabla g(y)\rangle - f(\nabla g(y)) \le \langle y, \nabla f^*(y)\rangle - f(\nabla f^*(y)) = f^*(y),$$

for all $y \in \mathbb{R}^d$ such that g and f^* are differentiable at y. We now claim that both g and f^* are differentiable Q-almost everywhere (a.e). If the claim is true, upon taking the expectation w.r.t Q:

$$V_f(g) \le V_f(f^*) = \mathbb{E}_Q[f^*(Y)], \quad \forall g \in \mathtt{CVX}(Q)$$

and the inequality is achieved with $g = f^*$. Now we prove the claim as follows: Since $\int g\, \mathrm{d}Q < \infty$, we have $Q(g = \infty) = 0$. Thus $Q(\mathrm{Dom}(g)) = 1$, where $\mathrm{Dom}(g)$ is the domain of the function g. Moreover, $Q(\mathrm{Int}(\mathrm{Dom}(g)) = 1$, where $\mathrm{Int}(\cdot)$ denotes the interior, because the boundary has Q-measure zero (Q has a density). Since g is convex, it is differentiable on $\mathrm{Int}(\mathrm{Dom}(g))$ except at points of Lebesgue measure zero which have Q-measure zero too. Therefore, g is Q-a.e differentiable. Similar arguments hold for f^*.

B. Proof of Theorem 3.6

The proof follows from the bounds

$$\|\nabla g - \nabla f^*\|^2_{L^2(Q)} \le \frac{2}{\alpha}\epsilon_1, \tag{10a}$$

$$\|\nabla f^* - \nabla g_0\|^2_{L^2(Q)} \le \frac{2}{\alpha}\epsilon_2, \tag{10b}$$

and using the triangle inequality. The proof for the first bound is as follows. If f is α-strongly convex, then f^* is $\frac{1}{\alpha}$ smooth. By definition of smoothness,

$$f^*(z) \le f^*(y) + \langle \nabla f^*(y), z - y\rangle + \frac{1}{2\alpha}\|z - y\|^2 \triangleq h_y(z), \quad \forall y, z \in \mathbb{R}^d,$$

where $h_y(z)$ is defined to be the quadratic function of z that appears on the right-hand side of the inequality. From $f^*(z) \le h_y(z)$, it follows that the convex conjugate $f(x) \ge h^*_y(x)$. As a result,

$$f(x) \ge h^*_y(x) = -f^*(y) + \langle y, x\rangle + \frac{\alpha}{2}\|x - \nabla f^*(y)\|^2, \quad \forall x, y \in \mathbb{R}^d. \tag{11}$$

We use this inequality to control the optimality gap $\epsilon_1(f, g)$:

$$\begin{aligned}
\epsilon_1(f, g) &= \mathcal{V}(f, g) - \inf_{\widetilde{g}} \mathcal{V}(f, \widetilde{g}) \\
&= \mathcal{V}(f, g) - \mathcal{V}(f, f^*) \\
&= \mathbb{E}_Q[f^*(Y) - \langle Y, \nabla g(Y)\rangle + f(\nabla g(Y))] \\
&\ge \frac{\alpha}{2}\mathbb{E}_Q[\|\nabla g(Y) - \nabla f^*(Y)\|^2],
\end{aligned}$$

where the last step follows from (**??**), with $x = \nabla g(y)$. This concludes the proof of the bound (**??**). It remains to prove (**??**). To this end, note that the optimality gap $\epsilon_2(f)$ is given by

$$\begin{aligned}
\epsilon_2(f) &= \mathcal{V}(f_0, g_0) - \inf_{\widetilde{g}} \mathcal{V}(f, \widetilde{g}) \\
&= \mathcal{V}(f_0, f_0^*) - \mathcal{V}(f, f^*) \\
&= -(\mathbb{E}_P[f_0(X)] + \mathbb{E}_Q[f_0^*(Y)]) + (\mathbb{E}_P[f(X)] + \mathbb{E}_Q[f^*(Y)]) \\
&= -\mathbb{E}_Q[f_0(\nabla f_0^*(Y)) + f_0^*(Y)] + \mathbb{E}_Q[f(\nabla f_0^*(Y)) + f^*(Y)] \\
&= -\mathbb{E}_Q[\langle Y, \nabla f_0^*(Y)\rangle] + \mathbb{E}_Q[f(\nabla f_0^*(Y)) + f^*(Y)]
\end{aligned}$$

Using the inequality (**??**) with $x = \nabla f_0^*(y)$ yields:

$$\epsilon_2(f) \ge \frac{\alpha}{2}\mathbb{E}_Q[|\nabla f_0^*(Y) - \nabla f^*(Y)|^2]$$

concluding (**??**) noting that $f_0^* = g_0$.

C. Experimental set-up

C.1. Two-dimensional experiments

Datasets. We use the following synthetic datasets: (i) Checkerboard, and (ii) Mixture of eight Gaussians. For the Checkerboard dataset, the source distribution Q is the law of the random variable $Y = X + Z$, where $X \sim \mathrm{Unif}(\{(0,0),(1,1),(1,-1),(-1,1),(-1,-1)\})$ and $Z \sim \mathrm{Unif}([-0.5,0.5] \times [-0.5,0.5])$. Similarly, P is the distribution of random variable $Y = X + Z$, where $X \sim \mathrm{Unif}(\{(0,1),(0,-1),(1,0),(-1,0)\})$ and $Z \sim \mathrm{Unif}([-0.5,0.5] \times [-0.5,0.5])$. Note that $\mathrm{Unif}(B)$ denotes the uniform distribution over any set B. For the mixture of eight Gaussians dataset, we have $Q = \mathcal{N}(0, I_2)$ and P is the law of random variable Y, where $Y = X + Z$ with $X \sim \mathrm{Unif}(\{(1,0),(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})\},(0,1),(\frac{-1}{\sqrt{2}},\frac{1}{\sqrt{2}}),(-1,0),(\frac{-1}{\sqrt{2}},\frac{-1}{\sqrt{2}}),(0,-1),(\frac{1}{\sqrt{2}},\frac{-1}{\sqrt{2}})\})$ and $Z \sim \mathcal{N}(0, 0.5I_2)$.

Architecture details. For our Algorithm 1, we parametrize both the convex functions f and g by ICNNs. Both these ICNN networks have equal number of nodes for all the hidden layers followed by a final output layer. We choose a square of leaky ReLU function, i.e $\sigma_0(x) = (\max(\beta x, x))^2$ with a small positive constant β as the *convex* activation function for the first layer σ_0. For the remaining layers, we use the leaky ReLU function, i.e $\sigma_l(x) = \max(\beta x, x)$ for $l = 1, \ldots, L-1$, as the *monotonically non-decreasing and convex* activation function. Note that the assumptions (ii)-(iii) of the ICNN are satisfied. In all of our experiments, we set the parameter $\beta = 0.2$. In some of the experiments as explained below, we chose the SELU activation function which also obeys the convexity assumptions.

For the three baselines, Barycentric-OT, W1-LP, and W2GAN, we use the implementations of Leygonie et al. (2019), made publicly available at `https://github.com/jshe/wasserstein-2`. For all these methods, we use the default settings of hyperparameters which were fixed to be the best values from the respective papers. Further, for a fair comparison we allow the number of parameters in each of these baselines to be larger than ours; in fact, for W2GAN and Barycentric-OT, the default number of neural network parameters is much larger than ours.

Hyperparameters. For reproducibility, we provide the details of the numerical experiments for each of the figures. For the Checkerboard dataset in Figure 3 (same as Figure 1), we run Algorithm 1 with the following parameters: For both the ICNNs f and g, we set the hidden size $m = 64$, number of layers $L = 4$, regularization constant $\lambda = 1.0$, Leaky ReLU activation and for training we use batch size $M = 1024$, learning rate 10^{-4}, generator iterations $K = 10$, total number of iterations $T = 10^5$, and the Adam optimizer with $\beta_1 = 0.5$, and $\beta_2 = 0.9$. For each of the baselines, the following are the values of the parameters: (a) Barycentric-OT: 3 (1 corresponding to the dual stage and the rest for the map step) neural networks each with $m = 128, L = 3, M = 512, T = 2 \times 10^5$ and l_2-entropy penalty, (b) W1-LP: Both the discriminator and the generator neural networks with $m = 128, L = 3, K = 5$ and $M = 512, T = 2 \times 10^5$, and (c) W2GAN: 3 neural networks (1 corresponding to the generator whereas the remaining are for two functions in the dual formulation (3)) each with $m = 128, L = 3, K = 5, M = 512, T = 2 \times 10^5$. W2GAN also uses six additional regularization terms which set to default values as provided in the code. Also, all these baselines use ReLU activation and Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.990$ and the learning rate for generator parameters being 0.0001 and 0.0005 for the rest. For the mixture of eight Gaussians dataset, we use the same parameters except batch-size $M = 256$, whereas all the baselines use the same parameters as the above setting. Also, for the multiple trials in Figure 4 for W1-LP and W2GAN, we use the above parameters but with a different random initialization of the neural network weights and biases.

Remark C.1. *Algorithm performance is not very sensitive to regularization constant, but the total iterations (*$50k$*-*$100k$*), batch-size(*128*, *256*), and inner loop iterations are crucial.*

C.2. High dimensional experiments

Gaussian to Gaussian. Source distribution $Q = \mathcal{N}(0, I_d)$ and target distribution $P = \mathcal{N}(\mu, I_d)$, for some fixed $\mu \in \mathbb{R}^d$ and $d = 784$. The mean vector $\mu = \alpha(1, \ldots, 1)^\top$ with $\alpha \in \{1, 5, 10\}$. For both the ICNNs f and g, we have $d = 784, m = 1024, L = 3$, Leaky ReLU activation, batch size $M = 60$, $K = 16$, $\lambda = 0.1$, $T = 40,000$, Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.99$, learning rate decay by a factor of 0.5 for every $2,000$ iterations. Note that in Figure 5a, 1 epoch corresponds to 1000 iterations.

High-dim. Gaussian to low-dim. mixture. Source distribution $Q = \mathcal{N}(0, I_d)$ with $d = 784$. The target distribution is a mixture of four Gaussians $P = \sum_{i=1}^{4} \frac{1}{4}\mathcal{N}(\mu_i, \Sigma)$, where $\mu_i = (\pm 1.4, \pm 1.4, 0, \ldots, 0) \in \mathbb{R}^{784}$ and $\Sigma = \mathrm{diag}(0.2, 0.2, 0, \ldots, 0)$. For both the ICNNs f and g, we have $d = 784, m = 1024, L = 3$, Leaky ReLU activation, batch size $M = 60$, $K = 25$, $\lambda = 0.01$, Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.99$, learning rate decay by a factor of 0.5 for every two epochs. The algorithm is simulated for 30 epochs, where each epoch corresponds to 1000 iterations.

MNIST $\{0, 1, 2, 3, 4\}$ to MNIST $\{5, 6, 7, 8, 9\}$. To obtain the latent embeddings of the MNIST dataset, we first train a VAE with both the encoder and decoder having 3 hidden layers with 256 neurons and the size of latent vector being 16 dimensional. We then use ICNNs f and g to learn the optimal transport between the embeddings of digits $\{0, 1, 2, 3, 4\}$ to that of $\{5, 6, 7, 8, 9\}$. For both these ICNNs we have $d = 16, m = 1024, L = 3$, CELU activation, batch size = 128, $K = 16$, $\lambda = 1$, $T = 100,000$, Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$, learning rate decay by a factor of 0.5 for every $4,000$ iterations.

Gaussian to MNIST. To obtain the latent embeddings for the MNIST, we use the same pre-trained VAE models as above. Also we use the same hyperparameter settings as that of the "**MNIST $\{0, 1, 2, 3, 4\}$ to MNIST $\{5, 6, 7, 8, 9\}$**" experiment with the only change of batch size being 64.

D. Further discussion of related work

The idea of solving the semi-dual optimization problem (4) is classically considered in (Chartrand et al., 2009), where the authors derive a formula for the functional derivative of the objective function with respect to f and propose to solve the optimization problem with the gradient descent method. Their approach is based on the discretization of the space and knowledge of the explicit form of the probability density functions, that is not applicable to real-world high dimensional problems.

More recently, the authors in (Lei et al., 2017; Guo et al., 2019) propose to learn the function f in a semi-discrete setting, where one of the marginals is assumed to be a discrete distribution supported on a set of N points $\{y_1, \ldots, y_N\} \subset \mathbb{R}^d$, and the other marginal is assumed to have a continuous density with compact convex support $\Omega \subset \mathbb{R}^d$. They show that the problem of learning the function f is similar to the variational formulation of the Alexandrov problem: constructing a convex polytope with prescribed face normals and volumes. Moreover, they show that, in the semi-distrete setting, the optimal f is of the form $f(x) = \max_{1 \leq i \leq 1}\{\langle x, y_i\rangle + b_i\}$ and simplify the problem of learning f to the problem learning N real numbers $b_i \in \mathbb{R}$. However, the objective function involves computing polygonal partition of Ω into N convex cells, induced by the function f, which is computationally challenging. Moreover, the learned optimal transport map ∇f, transports the probability distribution from each convex cell to a single point y_i, which results in generalization issues. Additionally, the proposed approach is semi-discrete, and as a result, does not scale with the number of samples.

Statistical analysis of learning the optimal transport map through the semi-dual optimization problem (4) is studied in (Hütter & Rigollet, 2019; Rigollet & Weed, 2018), where the authors establish a minimax convergence rate with respect to number of samples for certain classes of regular probability distributions. They also propose a procedure that achieves the optimal convergence rate, that involves representing the function f with span of wavelet basis functions up to a certain order, and also requiring the function f to be convex. However, they do not provide a computational algorithm to implement the procedure.

There are also other alternative approaches to approximate the optimal transport map that are not based on solving the semi-dual optimization problem (4). In (Leygonie et al., 2019), the authors propose to approximate the optimal transport map, through an adversarial computational procedure, by considering the dual optimization problem (3), and replacing the constraint with a quadratic penalty term. However, in contrast to the other regularization-based approaches such as (Seguy et al., 2017), they consider a GAN architecture, and propose to take the generator, after the training is finished, as the optimal transport map. They also provide a theoretical justification for their proposal, however the theoretical justification is valid in an ideal setting where the generator has infinite capacity, the discriminator is optimal at each update step, and the cost is equal to the exact Wasserstein distance. These ideal conditions are far from being true in a practical setting.

Two important related works are (Chen et al., 2019; Liu et al., 2018). Both aim to train a generative model to learn a distribution. They only use Wasserstein distance as a cost function to train the generator and do not aim to learn the optimal transport map. Moreover, they do not guarantee that the learned generator is the optimal transport map and they both involve intermediate steps (solving a LP/semi-discrete OT) that scales super-linearly in the number of samples.

Another approach, proposed in (Xie et al., 2019), is to learn the optimal coupling from primal formulation (2), instead of solving the dual problem (3). The approach involves representing the coupling with two generators that map a Gaussian random variable to $\mathbb{R}^d$, and two discriminators to ensure the coupling satisfies the marginal constraints. Although, the proposed approach is attractive when an optimal transport map does not exists, it is computationally expensive because it involves learning four deep neural networks. Finally, a procedure is recently proposed to approximate the optimal transport map that is optimal only on a subspace projection instead of the entire space (Muzellec & Cuturi, 2019). This approach is inspired by the sliced Wasserstein distance method to approximate the Wasserstein distance (Rabin et al., 2011; Deshpande et al., 2018). However, selection of the subspace to project on is a non-trivial task, and optimally selecting the projection is an optimization over the Grassmann manifold which is computationally challenging.

In a recent work, Korotin et al. (2019) too model the convex conjugate function f^* with an ICNN, denoted here by g, and a penalty term of the form $\|\nabla f(\nabla g(y)) - y\|^2$ is added to the semi-dual optimization (4). The penalty term serves to ensure that ∇g is inverse of ∇f and hence $g = f^*$. The additional penalty term makes the problem non-convex, even in the infinite capacity case, where the function representation is not restricted.