Flow-based Distributionally Robust Optimization

Chen Xu[1], **Jonghyeok Lee**[1], **Xiuyuan Cheng** [2], **Yao Xie** [1]
[1]School of Industrial and Systems Engineering, Georgia Tech
[2]Department of Mathematics, Duke University

Abstract

Flow-based models establish a continuous-time invertible transport map between a data distribution and a pre-specified target distribution, such as the standard Gaussian in normalizing flow. In this work, we study beyond the constraint of known target distributions. We specifically aim to find the worst-case distribution in distributional robust optimization (DRO), which is an infinite-dimensional problem that becomes particularly challenging in high-dimensional settings. To this end, we introduce a computational tool called FlowDRO. Specifically, we reformulate the difficult task of identifying the worst-case distribution within a Wasserstein-2 uncertainty set into a more manageable form, i.e., training parameters of a corresponding flow-based neural network. Notably, the proposed FlowDRO is applicable to general risk functions and data distributions in DRO. We demonstrate the effectiveness of the proposed approach in various high-dimensional problems that can be viewed as DRO, including adversarial attack and differential privacy.

1 Introduction

Flow-based models are continuous-time models that gradually transform base distributions into desired target distributions. Such models have been popular in the field of normalizing flow [Kobyzev et al., 2020], where the target distribution is pre-specified as the standard multivariate Gaussian. In this context, flow-based models allow accurate density estimation of and efficient sampling from the base distribution. However, in many problems, our objective is not to model the current data distribution but to find a density transport map that fits other goals. In this sense, we need to *extrapolate* the data distribution along certain directions. In this regards, one particular consideration is the problem of distributionally robust optimization (DRO) [Mohajerin Esfahani and Kuhn, 2018], which aims to find robust optimal solutions that minimize certain risk over the worst-case distribution within a pre-specified uncertainty set. An essential aspect of solving DRO is to find the worst-case distribution, which is an infinite-dimensional optimization problem that is particularly challenging in high dimension for general risk functions.

Contributions. In this work, to overcome the challenges in high dimension, we develop a flow-based neural network called FlowDRO to find the worst-case distribution in DRO. Main contributions are: (1) Re-formulate the problem of finding worst-case distributions in DRO into its Wasserstein proximal form and subsequently reduce the infinite-dimensional problem into solving for a transport map; (2) Parametrize the transport map by continuous-time flow neural networks and develop an efficient block-wise training algorithm to train the flow parameters; (3) Demonstrate the effectiveness of FlowDRO on various high-dimensional problems from adversarial attack and differential privacy.

Related works. The problem of DRO aims to find a minimax robust optimal solution that minimizes some expected loss taken over the worst-case distribution within a pre-specified set of distributions (i.e., a uncertainty set). At the core of DRO is how to find the worst-case distribution within an appropriately specified uncertainty set, so as to enable theoretical analyses and computational methods. The celebrated work [Mohajerin Esfahani and Kuhn, 2018] considers the Wasserstein uncertainty set over the base distribution, which is restricted to be the empirical data distribution. It proves a strong

Mathematics of Modern Machine Learning Workshop at NeurIPS 2023.

duality result for the problem and reformulates it as a finite-dimensional convex problem, where a worst-case *discrete* distribution can be found to attain the worst-case expectation in finite sample or asymptotically. Later works considered more general settings: [Gao and Kleywegt, 2023] considers a more general Wasserstein DRO problem, where the base distribution lies in a general Polish space. [Staib and Jegelka, 2019] considered maximum mean discrepancy (MMD) uncertainty sets. [Wang et al., 2021] considered uncertainty sets based on the sinkhorn distance. Despite the existing efforts to find appropriate DRO formulations that allow analytic or approximate solutions, current approaches still have limited scalability in solving high-dimensional problems with general loss functions.

Problem setup. Consider a real-valued *risk function* $\mathcal{R}$ taking as inputs a d-dimensional distribution P with a finite second moment and a measurable test function ϕ. Specifically, we assume there is a pre-specified loss function ℓ so that $\mathcal{R}(P, \phi) = \mathbb{E}_{X\sim P}[\ell(X, \phi)]$. We are interested in solving the following distributionally robust optimization (DRO) problem:

$$\min_{\phi\in\Phi} \max_{P\in\mathcal{P}_r} \mathcal{R}(P, \phi). \tag{1}$$

In (1), Φ denotes the constraint set for ϕ and $\mathcal{P}_r$ is the Wasserstein-2 (W_2) ball around the data distribution P_X. Namely, we let

$$\mathcal{P}_r = \{P : W_2^2(P, P_X) \le r^2\}, \tag{2}$$

where by the Monge formulation, the W_2 distance $W_2^2(P_2, P_1)$ between two d-dimensional distributions P_2, P_1 with finite second moments is written as

$$W_2^2(P_2, P_1) = \min_{T:T_\# P_1=P_2} \mathbb{E}_{X\sim P_1}\|T(X) - X\|_2^2. \tag{3}$$

In (3), $T : \mathbb{R}^d \to \mathbb{R}^d$ denotes the transport map and $T_\#$ denotes the push-forward operation by T, where $(T_\# P)(A) = P(T^{-1}(A))$ for a measureable set A.

2 Proposed FlowDRO

We focus on solving for the worst-case distribution within $\mathcal{P}_r$ (i.e., the maximizer of the inner problem of (1)), assuming ϕ is given. Note that for each $r > 0$, there is an $h > 0$ dependent on r, such that we have the following equivalent unconstrained problem:

$$\min_{P\in\mathcal{P}} -\mathcal{R}(P, \phi) + \frac{1}{2h} W_2^2(P, P_X), \tag{4}$$

where $\mathcal{P}$ denotes the space of all probability distributions on $\mathbb{R}^d$ with a finite second moment. Problem (4) can be viewed as a proximal step to minimize the objective $-\mathcal{R}(P, \phi)$ under the Wasserstein-2 metric in probability space [Jordan et al., 1998, Salim et al., 2020, Lin et al., 2021]. By the Monge formulation of the W_2 distance and the assumption that $\mathcal{R}(P, \phi) = \mathbb{E}_{X\sim P}[\ell(X, \phi)]$ for some loss function ℓ, the problem of solving for the worst-case distribution P in (4) thus reduces to finding the optimal transport map $T : \mathbb{R}^d \to \mathbb{R}^d$ that maps on data $X \sim P_X$:

$$\min_{T:\mathbb{R}^d\to\mathbb{R}^d} \mathbb{E}_{X\sim P_X} \left[-\ell(T(X), \phi) + \frac{1}{2h}\|T(X) - X\|_2^2\right]. \tag{5}$$

To yield a tractable and meaningful solution of T, we parametrize it as the solution map of a NeuralODE model [Chen et al., 2018]. Specifically, consider a density evolution (i.e., flow) $\rho(x, t)$ such that $\rho(x, 0) = P_x$ at $t = 0$, and as t increases, $\rho(x, t)$ approaches $\tilde{P}$, which is the (unknown) maximizer of (4).

We consider when the flow is induced by an ODE of $x(t)$ in $\mathbb{R}^d$: $\dot{x}(t) = v(x(t), t)$, where $x(0) \sim P_X$. We then parametrize $v(x(t), t)$ by a neural network $f(x(t), t; \psi)$ with trainable parameters ψ. As a result, at any time $t > 0$, the ψ-parametrized solution map T_s^t over an arbitrary interval $[s, t)$ can be expressed as

$$T_s^t(x; \psi) = x + \int_s^t f(x(s'), s'; \psi)ds'. \tag{6}$$

Without loss of generality, we assume the entire solution map is the integral of $f(x(t), t; \psi)$ over the unit interval $[0, 1)$. Using (6), the problem of finding T in (5) thus reduces to training ψ in the following problem:

$$\min_{\psi} \mathbb{E}_{X\sim P_X} \left[-\ell(T_0^1(X; \psi), \phi) + \frac{1}{2h}\|T_0^1(X; \psi) - X\|_2^2\right]. \tag{7}$$

We also propose a step-wise training algorithm of minimizing (7) with respect to the network parameters ψ. We build on the block-wise training method in [Xu et al., 2023], which was originally developed for training normalizing flows. Specifically, we would learn K networks block-wise, where the networks are parametrized by $\{\psi_k\}_{k=1}^K$. To do so, we first train ψ_1 using (7) with the penalty factor $h = h_1$. The expectation is taken over $x(0) \sim P_X$, the data distribution. Using the trained parameters $\hat{\psi}_1$, we could thus compute the push-forward distribution $P_1 = T_0^1(\cdot; \hat{\psi}_1)_\# P_X$. This push-forward operation is done empirically by computing $x(t_1) = T_0^1(x(0); \hat{\psi}_1), x(0) \sim P_X$ using the trained flow model. Then, we continue training ψ_2 using (7), where the expectation is instead taken over $x(t_1) \sim P_1$. In general, starting at $P_0 = P_X$, we are able to train the $(k+1)^{th}$ block parameters ψ_{k+1} given previous k blocks, where the expectation is taken over P_k.

Algorithm 1 Block-wise FlowDRO training

Require: Penalty parameters $\{h_k\}_{k=1}^K$, training data $X(0) \sim P_X$.

Ensure: K Trained flow blocks $\{T_0^1(\cdot; \hat{\psi}_k)\}$.

1: **for** $k = 1, \ldots, K$ **do**
2: Optimize $T_0^1(\cdot; \psi_k)$ upon minimizing (7) with $h = h_k$ using mini-batch sample average, given previous $k-1$ blocks $\{T_0^1(\cdot; \hat{\psi}_i)\}_{i=1}^{k-1}$.
3: **end for**

The computational complexity of the block-wise training Algorithm 1 can be analyzed in terms of counting the number of function evaluation of the network $f(x(t), t; \psi)$ when computing the loss (7). Suppose at block k, we break the integral of $f(x(t), t; \psi_k)$ over $[0, 1)$ into $S \geq 1$ smaller pieces $\{[t_i, t_{i+1})\}_{i=0}^{S-1}$. Let the integral on each piece be numerically estimated by the fixed-stage Runge-Kutta fourth-order method. As a result, it takes $O(4S)$ evaluation of $f(x(t), t; \psi_k)$ per X. If we have N training data and a total K blocks is trained, the overall computation is thus on the order of $O(4SKN)$, which is linear in the number of samples and thus scalable to large datasets.

3 Experiments

We consider two sets of experiments to demonstrate the effectiveness of the learned worst-case distribution of FlowDRO using Algorithm 1. Additional details are in Appendix A.

Adversarial attack. In the context of computer vision, adversarial attack perturb input raw images X_{img} into $\tilde{X}_{\text{img}}$, so that the accuracy of a pre-trained classifier ϕ on perturbed images is low [Madry et al., 2018]. Our FlowDRO can be used as a *distributional* attacker, where we find an alternative distribution $\tilde{P}$ on which the risk (average cross-entropy loss) of ϕ is high and accuracy is low.

Table 1 quantitatively compare the risk and accuracy of the pre-trained classifier ϕ on CIFAR10. We notice that under the *same amount* of W_2 perturbation as measured by $\mathbb{E}_{X_{\text{test}} \sim P_{X,\text{test}}} \|X_{\text{test}} - \tilde{X}_{\text{test}}\|^2$, FlowDRO performs much more effective attacks than the PDG baselines under ℓ_2 and ℓ_∞ perturbation [Madry et al., 2018]. Specifically, ϕ on the adversarial distribution found by FlowDRO yields significantly larger risk and lower accuracy. Meanwhile, Figure 1 visualizes the qualitative changes to test images X_{test} by FlowDRO and PGD, where the proposed FlowDRO also induces more meaningful contextual changes to the input image. Lastly, Figure A.1 visualizes the gradual changes of X_{test} over time and blocks by FlowDRO, demonstrating the continuous deformation by our trained flow model on test images X_{test}. In Appendix A.3, we consider additional examples on the MNIST digits, where we provide more insights into the behavior of FlowDRO

Differential privacy. Differential privacy (DP) offers a structured method to measure how well individual privacy is maintained in a statistical database, when collective data insights or statistical

Table 1: Risk and accuracy of a pre-trained VGG-16 classifier ϕ on clean test data and adversarially perturbed data by FlowDRO and by PGD under ℓ_2 and ℓ_∞ perturbation. For a fair comparison, we control the same amount of perturbation by different attacks as measured by the empirical Wasserstein-2 distance from the test distribution.

	Clean data	Attack by FlowDRO	Attack by PGD-ℓ_2	Attack by PGD-ℓ_∞
Risk in (12)	2.03	32.32	6.22	10.51
Accuracy in (13)	87.02	24.22	61.44	41.57

(a) Attacked by FlowDRO (b) Attacked by PGD-ℓ_2

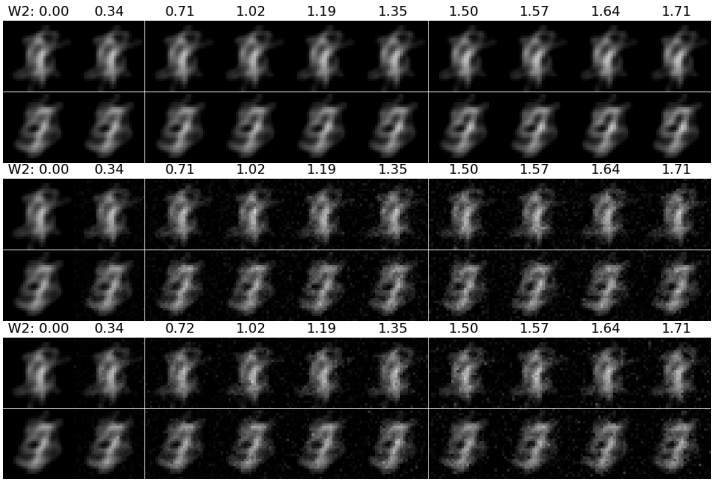
Figure 1: Adversarial samples found by FlowDRO and by PGD-ℓ_2. Captions show prediction by the pre-trained classifier ϕ on input images X_{test} (before attack) and $\tilde{X}_{test}$ (after attack).

summaries are shared as answers to the query. Specifically, given two neighboring datasets S and S' and a query function $q(S)$, DP mechanisms M_r are introduced so that a test function ϕ cannot distinguish between $M_r(q(S))$ and $M_r(q(S'))$. The subscript r refers to the amount of perturbation between $q(S)$ and $M_r(q(S))$, which is the W_2 distance in our examples. Our FlowDRO is thus a *distributional perturbation mechanism* (DPM), which is distinguished from *additional perturbation mechanisms* (APM) that add Gaussian [Dwork et al., 2014] or Laplacian noises [Dwork et al., 2006] to $q(S)$, which we call APM-G and APM-L respectively.

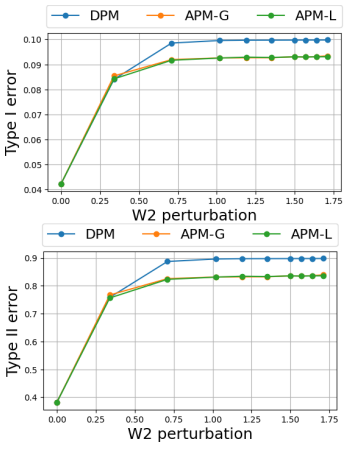
We demonstrate the effectiveness of DPM against APM on an example of MNIST with "one-neighborhood digit missing". Specifically, the dataset $S = \{X_1, \ldots, X_9 : X_i = (X_{\text{img,i}}, Y_i) \sim P_X, Y_i \neq Y_j \text{ if } i \neq j\}$, so it has precisely 9 random image-label pairs, one from each distinct class. The query function $q(S) = \sum_{i=1}^{9} X_{\text{img,i}}/9$, where the sum is taken pixel-wise. The test function ϕ is a pre-trained classifier on $\{q(S), Y_S\}$, where $Y(S)$ denotes the corresponding missing labels to S. Figure 2 shows both qualitative and quantitative comparisons of our proposed DPM against APM-G and APM-L. Qualitatively, we notice more contextual changes by DPM in subfigure (a) than APMs in subfigures (b) and (c). Quantitatively, the higher type-I and type-II errors in subfigure (d) demonstrate the benefit of DPM at protecting privacy against a pre-trained test function ϕ. In Appendix A.4, we perform an additional DP comparison on the raw MNIST digits, where the perturbation by DPM is much more meaningful than APM and protects privacy better.

4 Conclusion

In this work, we developed a flow-based model to solve for the worst-case distribution in high-dimensional distributional robust optimization. In the future, we would provide more rigorous theoretical analyses of the proposed method. We also aim to proposed techniques that solve the original min-max problem (1).



(a) DPM (top), APM-G (middle), and APM-L (bottom)



(b) Type-I and type-II errors by DPM, APM-G, and APM-L.

Figure 2: Differential privacy example of one-neighborhood digit missing. Figures (a) visualize privacy-protected queries $M_r(q(S))$ by DPM, APM-G, and APM-L within different Wasserstein-2 balls with radius r around the distribution of $q(S)$. Figure (b) examines the type-I and type-II errors defined in (17) over different r. We control the value of r by different DP mechanisms to be identical for a fair comparison.

References

Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. Hypothesis testing interpretations and renyi differential privacy. In *International Conference on Artificial Intelligence and Statistics*, pages 2496–2506. PMLR, 2020.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 84(1):3–37, 2022.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

Rui Gao and Anton Kleywegt. Distributionally robust stochastic optimization with wasserstein distance. *Mathematics of Operations Research*, 48(2):603–655, 2023.

Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the Fokker–Planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

Ivan Kobyzev, Simon JD 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, 2020.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Alex Tong Lin, Wuchen Li, Stanley Osher, and Guido Montúfar. Wasserstein proximal of GANs. In *International Conference on Geometric Science of Information*, pages 524–533. Springer, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*, 2018.

Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171(1-2):115–166, 2018.

Adil Salim, Anna Korba, and Giulia Luise. The Wasserstein proximal gradient algorithm. *Advances in Neural Information Processing Systems*, 33:12356–12366, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Matthew Staib and Stefanie Jegelka. Distributionally robust optimization and generalization in kernel methods. *Advances in Neural Information Processing Systems*, 32, 2019.

Jie Wang, Rui Gao, and Yao Xie. Sinkhorn distributionally robust optimization. *arXiv preprint arXiv:2109.11926*, 2021.

Larry Wasserman and Shuheng Zhou. A statistical framework for differential privacy. *Journal of the American Statistical Association*, 105(489):375–389, 2010.

Chen Xu, Xiuyuan Cheng, and Yao Xie. Normalizing flow neural networks by JKO scheme. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

A Experimental details

A.1 DRO in adversarial attack and differential privacy

We explain how adversarial attack and differential privacy can be formulated as (1).

Adversarial training with distributional robustness The problem of adversarial training is to improve the robustness of predictive models so that they can defend against unseen malicious attacks. Specifically, in the context of image classification, the test function ϕ is a K-class classifier mapping raw images $X_{\text{img}} \in \mathbb{R}^{h\times w\times c}$ to a probability distribution over the labels $Y \in \{0, \ldots, K\}$. Thus, the constraint set $\Phi = \{\phi : \mathbb{R}^{h\times w\times c} \to [0,1]^K$ such that $\sum_{i=1}^K \phi(X_{\text{img}})_i = 1\}$. Meanwhile, the loss function ℓ is typically chosen as the cross-entropy loss. Specifically, let $X = (X_{\text{img}}, Y)$, we would have $\ell(X, \phi) = -\log(\phi(X_{\text{img}})_Y)$. Minimizing the cross-entropy loss is thus equivalent to assigning as high a predicted probability to the true label Y given an input image X_{img} as possible. As a result, the risk function $\mathcal{R}(P, \phi) = \mathbb{E}_{X\sim P}[\ell(X, \phi)]$ computes the expected loss over image-label pairs in distribution P. Before perturbing P, we have $P = P_X$, which is the distribution of clean images along with their labels.

To improve the robustness of the classifier ϕ, it is essential to find alternative distributions $\tilde{P}$ on which the risk of ϕ is high. In the language of robust training, we are thus performing an adversarial attack on $X \sim P_X$. Conventionally, this is done via point-wise attack mechanisms as follows. Given a pair $X = (X_{\text{img}}, Y)$, the point-wise attacker fixes Y, and finds δ_{img} as the maximizer of the following problem:

$$\arg \max_{\delta:\|\delta\|_2\le r} -\log(\phi(X_{\text{img}} + \delta), Y). \tag{8}$$

Then, the adversarial image $\tilde{X}_{\text{img}} = X_{\text{img}} + \delta_{img}$. Note that this is called a point-wise attack because the maximizer of (8) only depends on the current pair $X = (X_{\text{img}}, Y)$, without considering the distribution of all $X \sim P_X$. It is also important to note that solving (8) exactly is hard, so that practical procedures often employ the PGD method by iteratively perturbing X_{img} [Madry et al., 2018]. In the context of the DRO problem (1), we thus have the uncertainty set

$$\mathcal{P}_r = \{P : \|\tilde{X}_{\text{img}} - X_{\text{img}}\|_2 \le r, (\tilde{X}_{\text{img}}, Y) \sim P \text{ and } (X_{\text{img}}, Y) \sim P_X\}. \tag{9}$$

In our distributional attack formulation, we have the same definitions of the set of valid classifiers Φ and the risk function $\mathcal{R}$. Rather than considering the point-wise uncertainty set $\mathcal{P}_r$ in (9), we relax it to be the Wasserstein-2 ball around P_X as defined in (2). The resulting perturbation mechanism is thus a distributional attacker of the data distribution P_X. Note that when performing distribution attack, we follow the convention that perturbation is only performed on raw images X_{img}, so that the corresponding label Y is untouched.

Differential privacy beyond additive mechanisms From the viewpoint of an adversary trying to differentiate between neighboring datasets based on the mechanism output, differential privacy can be well understood as a hypothesis testing problem. There have been several attempts to understand and analyze differential privacy as a framework for hypothesis testing [Wasserman and Zhou, 2010, Balle et al., 2020, Dong et al., 2022]. In the terminology of hypothesis testing, we consider

$$H_0 : Y \overset{d}{=} M(S) \sim P \quad vs \quad H_1 : Y \overset{d}{=} M(S') \sim Q$$

given a single observation $Y \in \mathcal{Y}$. The harder this test is, the more difficult it is to distinguish between neighboring datasets, which implies that strong privacy is ensured. Now consider a (randomized) test function ϕ for simple hypothesis testing, and denote its type-I and type-II errors as

$$\alpha_\phi = \mathbb{E}_{Y\sim P}\phi(Y), \quad \beta_\phi = 1 - \mathbb{E}_{Y\sim Q}\phi(Y).$$

The main statistical objective of differentially private randomized mechanisms is to minimize the perturbation (ensuring statistical utility) while obtaining a certain level of indistinguishability between P and Q. Or, equivalently, one can aim to maximize the indistinguishability between P and Q (statistical disclosure limitation) based on a given limited level of perturbation. Consider a risk function $\mathcal{R}((P, Q), \phi)$, typically constructed using α_ϕ and β_ϕ, which represents the ease of the test. To ensure strong privacy with a randomized mechanism, even in the "worst-case scenario" with a very good discriminator, one should make it difficult to distinguish by bringing the two distributions

closely together, thereby reducing the risk function. Hence, constructing a DP mechanism with highest possible statistical utility can be formulated as

$$\inf_{(P,Q)\in\mathcal{P}} \sup_{\phi\in\Phi} \mathcal{R}((P,Q),\phi). \tag{10}$$

Here, the choice of risk function is determined by the definition of DP being used. For instance, in the situation of calibrating a mechanism that satisfies ε-DP, the risk function is given by $\mathcal{R}((P,Q),\phi) = \min\{\frac{1-\beta_\phi}{\alpha_\phi}, \frac{1-\alpha_\phi}{\beta_\phi}\}$. Like this, the goal generally becomes to "maximize" these α_ϕ and β_ϕ in some perspective depending on which definition of DP is used.

The standard and straightforward method to privatize a query function is to apply additive noise. In this case, based on our DRO formulation, the class of perturbed distributions (P,Q) reduces to $\mathcal{P} = \mathcal{P}_{add} = \{(P,Q) : q(S) + \xi \sim P,\ q(S') + \xi \sim Q\}$ with ξ of distribution in a specific family. We will call a mechanism that adds noise of a certain distribution an additive perturbation mechanism (APM). Typical noise distributions used in APM include the Laplace and Gaussian distributions.

However, beyond simple additive noise, our objective is to generalize the randomized mechanism by distributional perturbation with respect to the Wasserstein distance as in (2). By solving

$$\inf_{(P,Q):W_2(P,P_X)\le r, W_2(Q,Q_X)\le r} \sup_{\phi\in\Phi} \mathcal{R}((P,Q),\phi), \tag{11}$$

we aim to provide a more flexible mechanism and consequently ensure indistinguishability with less perturbation than additive mechanisms. We shall refer to the corresponding mechanism as distributional perturbation mechanism (DPM).

A.2 Adversarial attack on CIFAR10

We describe the setup, introduce the comparison metrics, and present the comparative results.

Setup. Given a pre-trained image classifier ϕ and a test image X_{test} with labels Y_{test}, the goal of adversarial attack is to find a perturbed image $\tilde{X}_{test}$ based on X_{test} so that $\phi(\tilde{X}_{test})$ makes an incorrect classification. For this task, instead of performing point-wise attack given individual X_{test}, our FlowDRO finds a continuous flow T that gradually transports the distribution of X_{test} to an adversarial worst-case distribution, on which the classifier ϕ makes incorrect classification and induces high classification loss on average.

Regarding training specifics, we pre-train a VGG-16 classifier [Simonyan and Zisserman, 2015] ϕ with cross-entropy loss on the set of clean CIFAR-10 images, and then train three FlowDRO flow blocks with $h_k = 10$ using Algorithm 1. We train FlowDRO in the latent space of a variational auto-encoder as proposed by [Esser et al., 2021], where the latent space dimension $d = 192$. The architecture of the FlowDRO model on CIFAR10 consists of convolutional layers of `3-128-128-256`, followed by convolutional transpose layers of `256-128-128-3`. The kernel sizes and strides in the CIFAR10 attacker are `3-3-3-3-4-3` and `1-2-1-1-2-1`. We use the softplus activation with $\beta = 20$. Each block is trained for 15 epochs using a batch size of 500, with the Adam optimizer [Kingma and Ba, 2015] with a constant learning rate of 1e-3.

Comparison metric. We evaluate the effectiveness of adversarial attack by FlowDRO and PGD on the pre-trained classifier ϕ. Specifically, given test images X_{test}, we find adversarial samples $\tilde{X}_{test}$ using different attack mechanisms, where we fix *identical* amounts of Wasserstein-2 perturbation measured by $\mathbb{E}_{X_{test}\sim P_{X,test}}\|X_{test} - \tilde{X}_{test}\|^2$ to ensure a fair comparison. Then, given $\tilde{P}_{test}$ defined by the set of perturbed test images $\tilde{X}_{test}$, we evaluate ϕ on $\tilde{P}_{test}$ based on the sample average of

$$\mathcal{R}(\tilde{P}_{test},\phi) = \mathbb{E}_{X\sim\tilde{P}_{test}}[\ell(\phi(X),Y)], \tag{12}$$

$$\text{Accurcy}(\tilde{P}_{test},\phi) = \mathbb{E}_{X\sim\tilde{P}_{test}}[100\cdot\mathbb{1}(Y = \arg\max_j \phi(X)_j)]. \tag{13}$$

Hence, under the same amount of perturbation to find $\tilde{P}_{test}$, a higher risk (12) and a lower accuracy (13) indicate a more effective adversarial attack on ϕ.

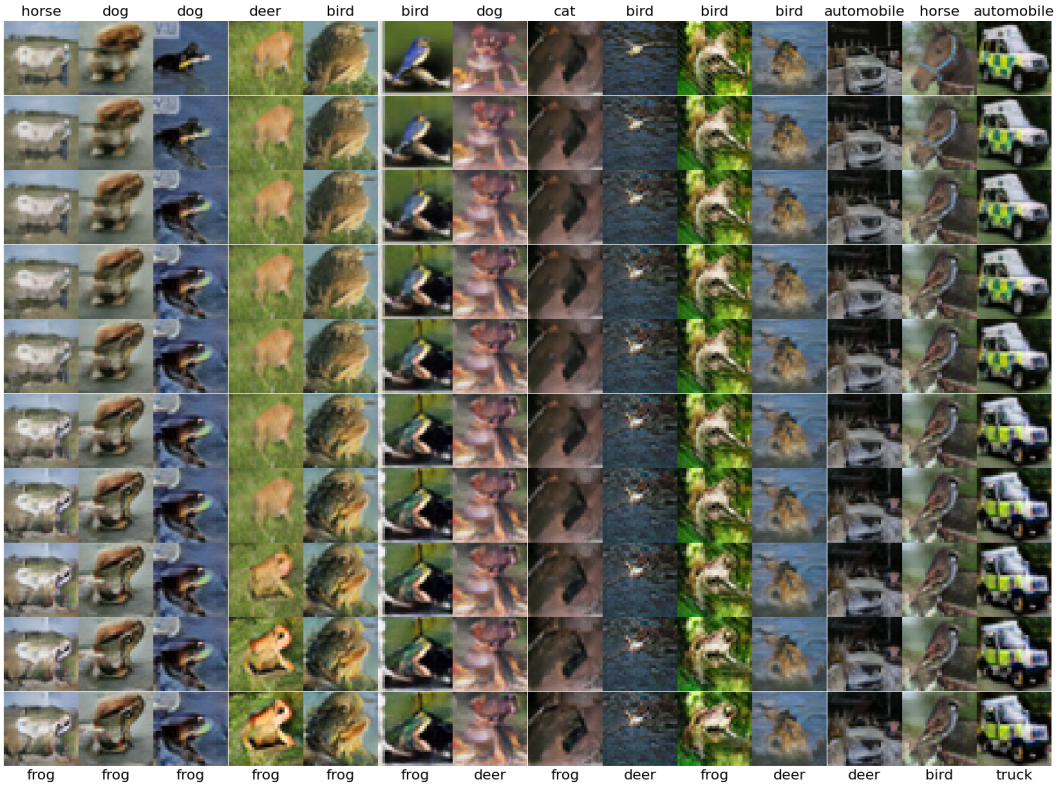


Figure A.1: Trajectory of FlowDRO adversarial attacks on different X_{test} (columns) to $\tilde{X}_{test}$. We visualize the changes as rows over three FlowDRO blocks, each of which breaks $[0, 1)$ into three evenly spaced sub-intervals, resulting in nine perturbation steps. Captions on the top and bottom indicate predictions by the pre-trained ϕ on raw X_{test} and perturbed $\tilde{X}_{test}$.

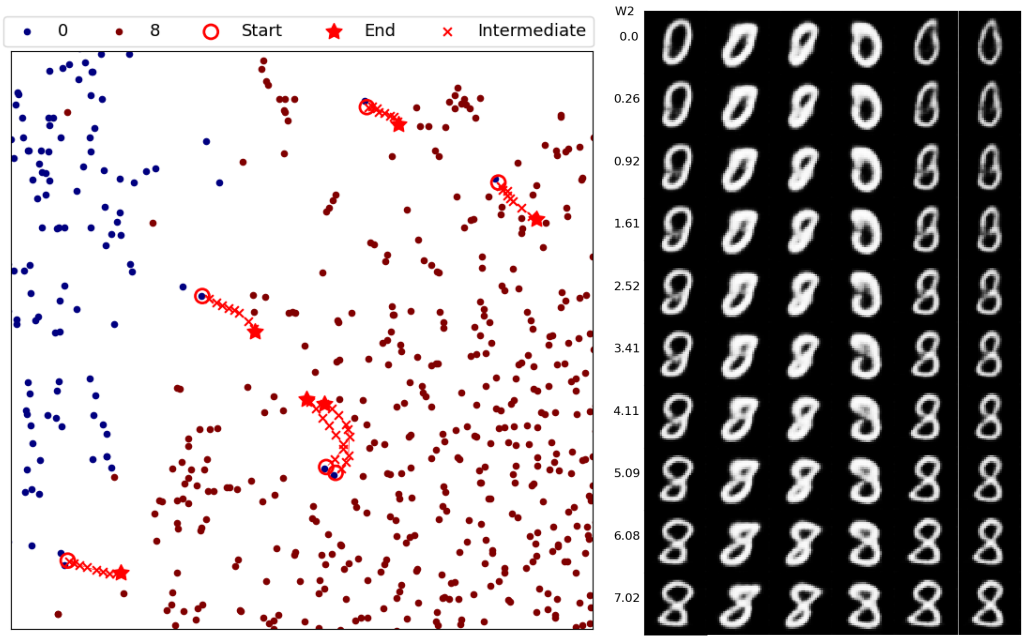
A.3 Adversarial attack on MNIST

We now apply FlowDRO on finding the worst-case distribution, given a pre-trained LeNet classifier [LeCun et al., 1998] ϕ. On this example, we focus on providing more insights into the behavior of FlowDRO, without comparing it against other baselines. We train FlowDRO using Algorithm 1 on the latent space of an auto-encoder with latent dimension $d = 16$. The architecture of the flow model consists of fully-connected layers of `d-256-256-d` with softplus activation.

Figure A.2 visualizes the gradual and smooth perturbation of test images X_{test} by FlowDRO. We specifically notice the cost-effectiveness and interpretability of FlowDRO. First, the T-SNE embedding in (a) shows that FlowDRO prefers to perturb digits around the boundary of certain digit clouds to that of other digit clouds, as such changes take the least amount of transport cost but can likely induce great increase of the classification loss by ϕ. Second, changes in the pixel space in (b) shows that more perturbation is applied to the foreground of the image (i.e., actual digits) than to the background (i.e., black regions), as the foreground tends to have a higher impact on the decision making of ϕ.

A.4 Differential privacy on raw MNIST digit recognition

We first describe the precise DP setup, comparison metrics, and then show the results against the baselines. This example directly follows from the adversarial attack example on MNIST in Section A.3. Specifically, the test function ϕ is a pre-trained convolutional neural network classifier on raw MNIST digits with 10 classes, and we train three continuous flow blocks on the class of all digits using Algorithm 1.



(a) Digit deformation trajectories under 2D T-SNE embedding (b) Gradual change of X_{test} in pixel space

Figure A.2: FlowDRO perturbation of MNIST digits over blocks and time integration. Figure (a) visualizes the perturbation trajectories from digits 0 to 8 under 2D T-SNE embedding. Figure (b) shows the trajectory in pixel space, along with the corresponding W_2 perturbation between original and perturbed images over time.

DP setup. We describe the following necessary components in a DP problem: (1) the definition of neighboring datasets S and S', where the two datasets differ in exactly one record; (2) the choice of the query function q taking the datasets as inputs; (3) the privacy-protection randomized mechanism M_r applied to $q(S)$, under the constraint that $P \in \mathcal{P}_r$ for $\mathcal{P}_r$ defined in (2); (4) the hypothesis testing problem with the test function ϕ to distinguish between S and S'. Notation-wise, we assume $X \sim P_X$ is a pair $X = (X_{\text{img}}, Y)$, where X_{img} is the raw image and $Y \in \{0, \ldots, 9\}$ is the corresponding label.

For (1), we let each dataset S contain one image-label pair $X \sim P_X$, so that two datasets S and S' are naturally neighbors in terms of X. In other words, S and S' contain digits either from the same class or from different classes. For (2), given $S = \{X\}$, we let the query function $q(S) = X_{\text{img}}$, so that it returns the raw image of the image-label pair X. For (3), the privacy-protection randomized mechanism M_r either applies our trained FlowDRO model to $q(S)$ or adds random Gaussian or Laplacian noises to $q(S)$, under a pre-specified amount of perturbation. For (4), given the true label Y of image X_{img}, we consider the following sets of hypotheses depending on labels $k \in \{0, \ldots, 9\}$:

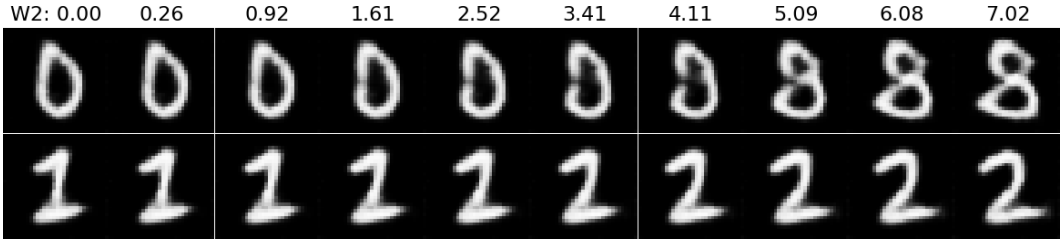
$$H_0(k) : Y \neq k \text{ and } H_1(k) : Y = k. \tag{14}$$

Hence, the goal of a DP mechanism M_r in this case is to not let the classifier ϕ correctly classify the true class of a privacy-protected test image $M_r(X_{test,img})$.

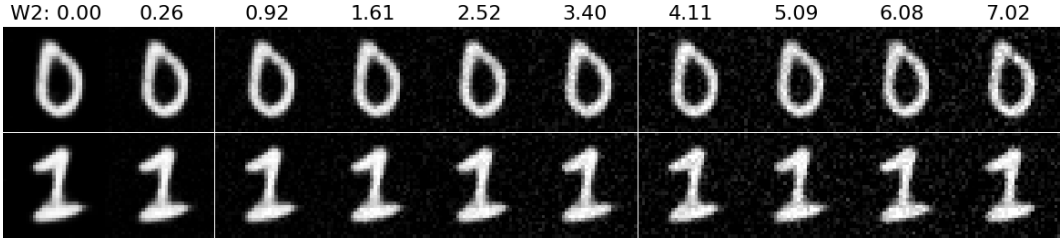
Comparison metrics. We measure the performance of different privacy-protecting randomized mechanisms M_r at radius r by the type-I and type-II errors of the classifier ϕ on testing (14) over different classes k. Recall the classifier ϕ maps an arbitrary input image to a probability distribution over the 10 classes. Thus, given a test dataset $S_{test} = \{X_{\text{test}}\}$ for $X_{\text{test}} \sim P_{X,\text{test}}$, we let $\hat{Y}(M_r) = \arg\max_{j=0,\ldots,9} \phi(M_r(q(S_{test})))_j$ be the predicted class of $M_r(q(S_{test}))$ by ϕ. Then, the type-I error $\alpha(k, M_r)$ and type-II error $\beta(k, M_r)$ are computed as

$$\alpha(k, M_r) = \mathbb{P}(\hat{Y}(M_r) = k | Y \neq k) \tag{15}$$

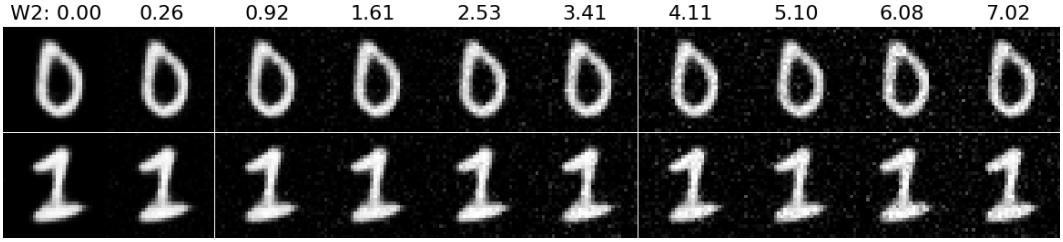
$$\beta(k, M_r) = \mathbb{P}(\hat{Y}(M_r) \neq k | Y = k), \tag{16}$$



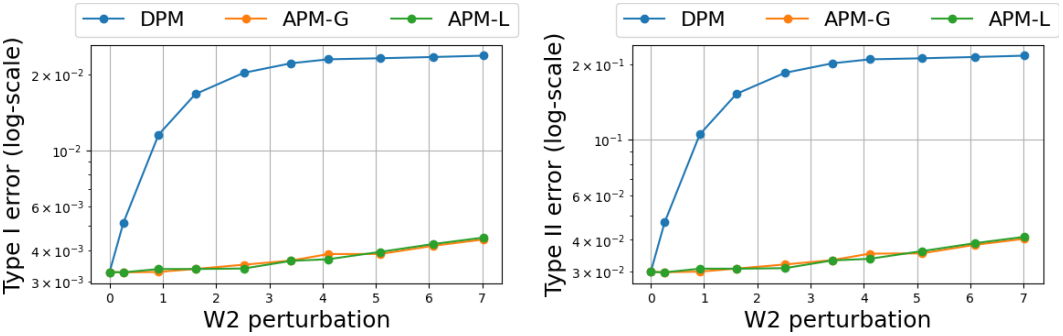
(a) DPM by FlowDRO



(b) APM-G (additive Gaussian)



(c) APM-L (additive Laplacian)



(d) Type-I and type-II errors by DPM, APM-G, and APM-L.

Figure A.3: Differential privacy example of raw MNIST digit recognition. We present similar sets of figures as in Figure 2, where the main difference lies in the definition of dataset S and query function $q(S)$.

where the probability is taken over test image-label pairs $X_{\text{test}} = (X_{test,img}, Y_{test})$ for $X_{\text{test}} \sim P_{X,\text{test}}$. We then measure the performance of M_r by taking the average of (15) and (16) over k:

$$\alpha(M_r) = \sum_{k=0}^{9} \alpha(k, M_r)/10, \ \beta(M_r) = \sum_{k=0}^{9} \beta(k, M_r)/10. \tag{17}$$

If the mechanism M_r provides strong privacy, we would then expect high values of $\alpha(M_r)$ and $\beta(M_r)$, as the classifier ϕ makes high errors on privacy-protected images $M_r(X_{test,img})$.

Results. Figure 2 shows the comparative results by the proposed FlowDRO DPM against the APM-G and APM-L baselines. Qualitatively, we notice in (a)-(c) that under the same amount of perturbation r, DPM induces meaningful contextual changes to the queries $q(S)$ (i.e., changing a digit 0 to a digit 8), whereas the additive mechanisms only blur the queries slightly. Quantitatively, as shown in (d), such

difference helps protect privacy against the test function ϕ: the type-I and type-II errors of ϕ under our proposed DPM are much higher than those of ϕ under the additive perturbation mechanisms. As a result, our DPM is an empirically more effective privacy-protecting mechanism under the same amount of average perturbation as measured in r.

A.5 Differential privacy on one-neighborhood digit missing

We follow the notations in Section A.4 when describing the setup and metrics.

DP setup and comparison metric. We define (1)–(4) in this new setting. For (1), we define a dataset $S = \{X_1, \ldots, X_9 : X_i = (X_{\text{img,i}}, Y_i) \sim P_X, Y_i \neq Y_j \text{ if } i \neq j\}$. Thus, the dataset S has precisely 9 random image-label pairs, one from each distinct class. Given two datasets S and S', they are neighbors in the sense that the set of labels $\{Y_i\}$ in S and S' differ by at most one value. For (2), the query function $q(S) = \sum_{i=1}^{9} X_{\text{img,i}}/9$, where the sum is taken pixel-wise, so that $q(S)$ returns an average image of the same dimension as each X_{img}. For (3), the privacy-protection mechanism M_r either applies our trained FlowDRO model to the average image $q(S)$ or adds random noises to it. For (4), given the true missing label $Y(S)$ of the dataset S, we then consider the following sets of hypotheses depending on the label $k \in \{0, \ldots, 9\}$:

$$H_0(k) : Y(S) \neq k \text{ and } H_1(k) : Y(S) = k. \tag{18}$$

In this new setup, we still evaluate the effectiveness of a DP mechanism M_r using (17), where the probabilities of type-I and type-II errors are taken over test datasets S_{test}, each of which contains nine random test image-label pairs $X_{test} \sim P_{X,test}$.

We also explain how we train the classifier ϕ and the flow model T in this new setting. The architecture of ϕ is still based on convolutional layers, where the training data of ϕ consists of $\{q(S), Y(S)\}$, which are the set of average images $q(S)$ and corresponding missing labels $Y(S)$. We then train ϕ using empirical risk minimization under the cross-entropy loss by sampling mini-batches of datasets S. The classifier ϕ is thus trained to determine the missing label $Y(S)$ given the average image. To train the flow model T using Algorithm 1, we adopt the identical network architecture as in the previous MNIST examples and train three blocks given the classifier ϕ to maximize the expected cross-entropy loss of ϕ with W_2 regularization.