How to Measure Distance Between High-Dimensional Distributions

Pramook Khungurn

February 5, 2025

This is note is about how to measure distance between probability distributions in high-dimensional spaces such as distributions of images. The main application of this problem is evaluation of generative models. The content of this note is mainly based on an expository article by Bischoff et al. [BDD+24].

- The paper title starts with "A Practical Guide," which means that it is not a survey paper and so is not comprehensive.
- The authors say that the aim is to "provide an accessible entrypoint to understanding popular sampled-based statistical distance."
- From the get-go, the authors give a list of more specialized papers. Let's list them here for reference.
  - Theis et al. *A note on evaluation of generative models.* [TvdOB16].
  - Borji. *Pros and cons of GAN evaluation measures.* [Bor18]
  - Xu et al. *An empirical study on evaluation metrics of generative adversarial networks.* [XHY+18]
  - Yang et al. *Revisiting the Evaluation of Image Synthesis with GANs.* [YYZ+23]
  - Basseville. *Divergence measures for statistical data processing—An annotated bibliography.* [Bas13]
  - Gibbs and Su. *On choosing and bounding probability metrics.* [GS02]

1 Problem Setup

- We have two datasets of i.i.d. samples.
  - $\{x_1, x_2, \ldots, x_n\} \sim p_1(x)$.
  - $\{y_1, y_2, \ldots, y_m\} \sim p_2(y)$.

  Here p_1 and p_2 are probability distributions on the same domain. These two distributions can either be
  - two generative models,
  - a generative model and the training data.
- We typically assume that the data domain is $\mathbb{R}^d$ where d is large.
- We want to quantify how similar p_1 and p_2 to each other.
- However, we don't have access to the distributions. We only have the samples.

2 Sliced-Based Distances

- When d is large, computing distance between distributions become very expensive.
- To deal with this problem, "sliced" distances have been instroduced [KNS+19, NDC+22, GG21].
- Main idea: for many existing statistical distances, we can efficiently evaluate the distance in low-dimensional spaces, especially in 1D.
- A slice is thus typically a projection of the data points onto a line in $\mathbb{R}^d$.
  - Each point is mapped to the nearest point on the line.
- The projection is typically chosen at random.
  - This is typically a point from the unit sphere S^{d-1}.
- Using only one projection is unreliable. So, we do multiple random projections, compute the distance with respect to each projection, and then average the results.
  - More formally, this is the expected value of the distance metric where the projection is chosen at random.
- As long as the distance computed in 1D is a metric, the sliced distance computed this way is a metric as well [NDC+22].
- The most popular sliced distance is the **sliced Wasserstein distance**. It has several desirable properties.
  - It can be computed in a differentiable manner.
  - Computation does not rely heavily on choices of hyperparameters.
  - It is very fast to compute.
- Before we can talk about sliced Wasserstein, we need to talk about Wasserstein distance first.
- **Definition 1.** *Let p_1 and p_2 be probability distributions on a domain Ω. A* **coupling** *between p_1 and p_2 is a distribution γ on $\Omega \times \Omega$ such that the marginal distributions agree with p_1 and p_2. More specifically,*

$$p_1(x_1) = \int_\Omega \gamma(x_1, x_2)\, \mathrm{d}x_2,$$
$$p_2(x_2) = \int_\Omega \gamma(x_1, x_2)\, \mathrm{d}x_1.$$

  We also call a coupling a **transport plan**.
- Let $\Gamma(p_1, p_2)$ denote the set of all couplings between p_1 and p_2.
- **Definition 2.** *Let $q \geq 1$ be a real number. The* **q-norm** *of $x \in \mathbb{R}^d$ is*

$$\|x\|_q = \left(\sum_{i=1}^{d} |x^i|^q \right)^{1/q}$$

  where x^i denotes the ith component of x (i.e., Spivak's notation).

- **Definition 3.** *Let p_1 and p_2 be probability distributions on $\mathbb{R}^d$. The* **Wasserstein-q distance** *is defined as*

$$W_q(p_1, p_2) = \inf_{\gamma \sim \Gamma(p_1, p_2)} \left(E_{(x_1, x_2) \sim \gamma} \left[\|x_1 - x_2\|_q^q \right] \right)^{1/q}.$$

- The Wasserstein distance is also called the **earth mover's distance** (EMD).
- We typically estimate $W_q(p_1, p_2)$ by finite samples from p_1 and p_2.
- More concretely, we compute the Wasserstein-q distances between empirical distributions of p_1 and p_2.
- **Definition 4.** *Let p be a probability distribution on Ω. Let N be a positive integer, and let x_1, x_2, $\ldots$, x_N be i.i.d. samples from p. An* **empirical distribution** *$p^{(N)}$ is given by*

$$p^{(N)}(x) = \frac{1}{N} \sum_{i=1}^{N} \delta(x - x_i)$$

where δ denotes the Dirac delta function.

- So, given p_1 and p_2, we can estimate $W_q(p_1, p_2)$ with $W_q(p_1^{(N)}, p_2^{(N)})$ where $p_1^{(N)}$ and $p_2^{(N)}$ are empirical distributions of p_1 and p_2, respectively.
  - According to Papp and Sherlock, this estimate is biased but consistent [PS22]. In other words,

$$\lim_{N \to \infty} W_q(p_1^{(N)}, p_2^{(N)}) = W_q(p_1, p_2).$$

  - However, convergence is exponentially slower as the dimensionality of the space increases.
- Let us now turn to how to compute $W_q(p_1^{(N)}, p_2^{(N)})$.
- This problem of is the special case of the **Kantorovich problem**.
  - We are given two sets of samples $\{x_1, x_2, \ldots, x_N\}$ and $\{y_1, y_2, \ldots, y_N\}$.
  - We are also given a cost matrix C whose entries are pairwise distance between x_i and y_j:

$$C = \begin{bmatrix} \mathfrak{d}(x_1, y_1) & \mathfrak{d}(x_1, y_2) & \cdots & \mathfrak{d}(x_1, y_N) \\ \mathfrak{d}(x_2, y_1) & \mathfrak{d}(x_2, y_2) & \cdots & \mathfrak{d}(x_2, y_N) \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{d}(x_N, y_1) & \mathfrak{d}(x_N, y_2) & \cdots & \mathfrak{d}(x_N, y_N) \end{bmatrix}$$

    Here, $\mathfrak{d}(\cdot, \cdot)$ is a distance metric.
  - We are then given two vectors $\mu \in [0,1]^N$ and $\nu \in [0,1]^N$ such that $\mathbb{1}^T \mu = \mathbb{1}^T \nu = 1$ where $\mathbb{1}$ is the vector whose entries are always ones.
    - These vectors represents discrete probabilities on $\{x_1, x_2, \ldots, x_N\}$ and $\{y_1, y_2, \ldots, y_N\}$, respectively.
  - Let $\mathcal{U}(\mu, \nu) \subset [0,1]^{N \times N}$ denote the set of $N \times N$ matrix such that $U\mathbb{1} = \mu$ and $U^T \mathbb{1} = \nu$.
  - Each matrix $P \in \mathcal{U}(\mu, \nu)$ is a coupling, i.e. a transport plan, of μ and ν.
  - The Kantorovich problem is as follows:

$$\inf_{P \in \mathcal{U}(\mu, \nu)} \{P \otimes C\}$$

    where $\otimes$ denote element-wise multiplication.

- The Kantorovich problem is a linear program. It can also be casted into the problem of min-cost maximum flow.
- Surprisingly, when μ and ν are vectors that are associated with empirical distributions, we have that the solution to the Kantorovich problem has a special form [PC19] (Proposition 2.1).

  Theorem 5. *If $\mu = \nu = \mathbb{1}/N$, then there is an optimal solution of the Kantorovich probem that is a permutation matrix.*

  In other words, it becomes a min-cost bipartite matching problem. Hence, it can be solved by the Hungarian algorithm in $O(N^3)$ time.
- However, when $d = 1$, we can compute the solution to the Kantorovich problem much faster.
  - Sort $\{x_1, x_2, \ldots, x_N\}$ and $\{y_1, y_2, \ldots, y_N\}$ in increasing order.
  - The solution is then to match x_1 with y_1, x_2 with y_2, and so on.
  - As a result,

$$\inf_{P \in \mathcal{U}(\mu,\nu)} \{P \otimes C\} = \sum_{i=1}^{N} \mathfrak{d}(x_i, y_i).$$

  This algorithm takes $O(N \log N)$. This is why sliced Wasserstein distance is much faster to compute than regular Wasserstein distance.
- Note, however, that the sliced Wasserstein distance does not converge to the Wasserstein distance even if we take the number of projections to infinity [NDC+22].
- A drawback to sliced distance is that, if the two distributions differ only along a small subset of directions, then it is hard to hit those directions with random projections.

3 Classifier Two-Sample Test

- The **classifier two-sample test** (C2ST) uses a classsifier that discriminates between samples from the two distributions.
- Classifier performance is the distance between the distributions: the better the performance, the farther the distributions.
- For example, we can train a classifier $c(x)$ to distinguish samples from distributions p and q. Then, we can evaluate C2ST as

$$\frac{1}{2}\Big(E_{x\sim}[\mathbb{1}(c(x) = 0)] + E_{x\sim q}[\mathbb{1}(c(x) = 1)]\Big)$$

  where $\mathbb{1}(\cdot)$ is the indicator function.
- The expression above is classifier accuracy.
  - If the value is 0.5, which the classifer is at chance level, then the distributions are indistinguisable to the classifer.
  - If the value is 1.0, then the two distributions have disjoint support.
  - For any two distributions, there is the maximum accuracy that any classifier can attain.
    - This can be evaluated if we have access to the distribution functions of both distributions, which we generally do not have.

- To make C2ST as accurate as possible, one must train the optimal classifier.
  - By using neural networks, we hope that we can train a classifer that is close to optimal.
- C2ST pros
  - Highly interpretable.
  - Can be used to test statistical significance of the difference between two sets of samples.
- C2ST cons
  - Resource intensive: you need to train a classifier.
  - Very challenging to use a classifier in a training loop.
    - GANs are notoriously hard to train.
  - C2ST values depend on classifer capacity, architecture, and hyperparameters, and training processes. This means you cannot really say anything for sure.
- Common failure modes.
  - C2ST number is low, but the distributions are not similar.
    - Using not enough training examples or underpowered architectures can result in C2ST values that are lower than the maximally possible value.
    - It's easy to fool yourself that you have struck gold when the classifier you train is not an optimal one.
  - C2ST can be surprisingly high for seemingly good generative results.
    - A generative model might align very well with the data for every marginal.
    - The C2ST can still be high if the data is high-dimensinoal.
    - It can be hard to achieve low C2ST values in high-dimensional data.
    - For example, you can try to represent each image in the MNIST dataset with mixture of Gaussians. Even if the Gaussian images look good, a classifier can still tease out the original MNIST dataset from the Gaussian mixture ones.
- Other variants
  - Statistics other than accuracy [Ras14] can be used to define C2ST too.
  - Mean square error of achieved accuracy to target value of 0.5 [KLL19].
  - Likelihood ratio statistics [PBNF24].
  - Average different in logits [CC22].
- Learned classifier can be used to estimate the density ratio $p(x)/q(x)$.
  - This has application in computing the KL divergence [TR19, Hus17] and Pearson divergence [SXGS20].

4 Maximum Mean Discrepancy

- **Maximum mean discrepancy** (MMD) is a popular distance metric [GBR+12].
  - Applicable to a variety of data domains.
  - Has been used to evaluate generative models [STS+21, Bor18, LBG+21].
  - Has the ability to indicate where the model and the true distribution differ.

- It uses a function ϕ, called a **feature map**, to embed samples before computing any statistics.
  - If we choose the feature map, then properties of the underlying distribution can be easily computed.
- We can define MMD as the difference between the means of the embedding of two distributions.

$$\text{MMD}[\phi, p_1, p_2] = \sqrt{\left\| E_{x\sim p_1}[\phi(x)] - E_{x\sim p_2}[\phi(x)] \right\|^2}.$$

- An illuminating set of examples.
  - Say, we want to compare two distributions p_1 and p_2 on the real line $\mathbb{R}$.
  - Let us consider the simplest feature map, the identity function: $\phi^{(1)}(x) = x$.
  - We have that the MMD is just the absolute difference between the means of the distributions.

    $$\text{MMD}[\phi^{(1)}, p_1, p_2] = |\mu_1 - \mu_2|$$

    where $\mu_1 = E_{x\sim p_1}[x]$ and $\mu_2 = E_{x\sim p_2}[x]$.
  - So, $\text{MMD}[\phi^{(1)}, \cdot, \cdot]$ can only distinguish between distributions with different means.
  - However, we can make the feature map more complex to distinguish other features. For example, we can add x^2 term.

    $$\phi^{(2)}(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$$

  - Because $E[x^2] = \mu^2 + E[(x-\mu)^2] = \mu^2 + \sigma^2$ where σ is the standard deviation of the random variable x. We have that

    $$\begin{aligned} \Big(\text{MMD}[\phi^{(2)}, p_1, p_2]\Big)^2 &= (E_{x\sim p_1}[x] - E_{x\sim p_2}[x])^2 + (E_{x\sim p_1}[x^2] - E_{x\sim p_2}[x^2])^2 \\ &= (\mu_1 - \mu_2)^2 + (\mu_1^2 + \sigma_1^2 - \mu_2^2 - \sigma_2^2)^2. \end{aligned}$$

    Here σ_1 and σ_2 are standard deviation of p_1 and p_2, respectively. So, with $\phi^{(2)}$, we can distinguish distributions with different variance as well.
- Following the examples above, we can keep adding features to distinguish higher moments if we want to distinguish more distributions. Howver, this is not feasible because, to distinguish all distributions, we need infinitely many monents and so infinitely many features.
- The above limitation can be circumvent by the **kernel trick**.
  - First, rewrite MMD using inner products of features.

    $$\begin{aligned} &\big(\text{MMD}[\phi, p_1, p_2]\big)^2 \\ &= \left\| E_{x\sim p_1}[\phi(x)] - E_{x\sim p_2}[\phi(x)] \right\|^2 \\ &= \big\langle E_{x\sim p_1}[\phi(x)] - E_{x\sim p_2}[\phi(x)], E_{x\sim p_1}[\phi(x)] - E_{x\sim p_2}[\phi(x)] \big\rangle \\ &= \big\langle E_{x\sim p_1}[\phi(x)], E_{x\sim p_1}[\phi(x)] \big\rangle + \big\langle E_{x\sim p_2}[\phi(x)], E_{x\sim p_2}[\phi(x)] \big\rangle - 2\big\langle E_{x\sim p_1}[\phi(x)], E_{x\sim p_2}[\phi(x)] \big\rangle \\ &= E_{x\sim p_1, x'\sim p_1}\big[\langle \phi(x), \phi(x') \rangle\big] + E_{x\sim p_2, x'\sim p_2}\big[\langle \phi(x), \phi(x') \rangle\big] + 2E_{x\sim p_1, x'\sim p_2}\big[\langle \phi(x), \phi(x') \rangle\big]. \end{aligned}$$

  - Next, write each inner product as a **kernel function**

    $$k(x, y) = \langle \phi(x), \phi(y) \rangle.$$

– So, the MMD becomes

$$\big(\text{MMD}[k, p_1, p_2]\big)^2 = E_{x\sim p_1, x'\sim p_1}\big[k(x, x')\big] + E_{x\sim p_2, x'\sim p_2}\big[k(x, x')\big] + 2E_{x\sim p_1, x'\sim p_2}\big[k(x, x')\big].$$

- If we can find a kernel that captures all the infinintely many moments, then MMD would be zero if and only if the two distributions are exactly the same. This means that MMD becomes a metric. Such as kernel is called a **characteristic kernel**.
- An example of a characteristic kernel is the Gaussian kernel

$$k_G(x, x') = \exp\bigg(-\frac{\|x - x'\|^2}{2\sigma^2}\bigg).$$

  Other kernels can be found in [FGL+09].
- The following is an unbiased estimate of the MMD.

$$\text{MMD}^2 = \frac{1}{m(m-1)}\sum_{i\neq j} k(x_i, x_j) + \frac{1}{n(n-1)}\sum_{i\neq j} k(y_i, y_j) - \frac{2}{mn}\sum_{i,j} k(x_i, y_j)$$

  where $\{x_1, x_2, \dots x_m\}$ are i.i.d. samples from p_1, and $\{y_1, y_2, \dots, y_n\}$ are i.i.d. samples from p_2.
- The selection of the right kernel and its parameters is crucial.
- In case of the Gaussian kernel, the parameter to choose is the standard deviation σ.
  - As $\lim \sigma \to 0$, we have that $k_G(x, x') \to \mathbb{1}(x = x')$. So, the kernel is 0 most of the time.
  - On the other hand σ is to high, then $k_G(x, x')$ would be close to 1 almost all the time.
- How to pick σ.
  - The **median heuristics** sets σ to the media distance between points in the aggregate sample.
  - Use data splitting. This involves dividing datasets into parts.
    * One part is used for kernel selection.
    * One part is used for evaluating MMD.

    This naturally reduces the number of data points for estimating MMD.
  - There are ways to pick σ without data splitting [BSG23, SKA+23, KJSM22, KSB+23].

5 Embedding-Space Measures

- We use a neural network called an **embedding network** $f : \mathcal{X} \to \mathbb{R}^d$ to map a data item to a feature vector.
  - Most of the time, d is much lower than the dimensionality of the data space $\mathcal{X}$.
- Given two sets of samples, we compute two sets of embeddings. Then, the embedding can be compared with apprpriate distance.
- It is common to approximate the embedded distribution with a Guassian distribution by estimating their means μ and covariance matrices Σ. Under Gaussian approximation, the squared Wasserstein distance (also called the Fréchet distance) can be computed as:

$$W^2((\mu_1, \Sigma_1), (\mu_2, \Sigma_2)) = \|\mu_1 - \mu_2\|^2 + \text{tr}\big(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{1/2}\big)$$

  However, any other metric can be used.

- The prime example of this type of metric is the **Fréchet inception distance** (FID) [HRU+18].
  - The embedding network is InceptionV3 [SVI+15] trained on ImageNet classification with the classification head removed.
  - FID is biased in finite samples.
- **Kernel inception distance** (KID) uses MMD as a metric in the embedding space [BSAG21].
  - Pros
    - Unlike FID, KID has a simple, unbised estimator with better sample efficiency.
    - Does not assume any parametric form of the distributions.
  - Cons
    - Must carefuly select hyperparameter for the kernel function.
- **CLIP-MMD** uses CLIP and an embedding network and compute the distance with MMD using a Gaussian kernel [JRV+24].
- Limitation of the approach is that we need a good embedding network.
  - People are exploring other embedding networks such as CLIP and vision language models.

References

[Bas13] M. Basseville, *Divergence measures for statistical data processing–an annotated bibliography*, Signal Processing **93** (2013), no. 4, 621–633.

[BDD+24] S. Bischoff, A. Darcher, M. Deistler, R. Gao, F. Gerken, M. Gloeckler, L. Haxel, J. Kapoor, J. K. Lappalainen, J. H. Macke, G. Moss, M. Pals, F. Pei, R. Rapp, A. E. Sağtekin, C. Schröder, A. Schulz, Z. Stefanidi, S. Toyota, L. Ulmer, and J. Vetter, *A practical guide to sample-based statistical distances for evaluating generative models in science*, 2024, `arXiv:2403.12636 [cs.LG]`.

[Bor18] A. Borji, *Pros and cons of gan evaluation measures*, 2018, `arXiv:1802.03446 [cs.CV]`.

[BSAG21] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, *Demystifying mmd gans*, 2021, `arXiv:1801.01401 [stat.ML]`.

[BSG23] F. Biggs, A. Schrab, and A. Gretton, *Mmd-fuse: Learning and combining kernels for two-sample testing without data splitting*, 2023, `arXiv:2306.08777 [stat.ML]`.

[CC22] X. Cheng and A. Cloninger, *Classification logit two-sample testing by neural networks for differentiating near manifold densities*, IEEE Transactions on Information Theory **68** (2022), no. 10, 6631–6662.

[FGL+09] K. Fukumizu, A. Gretton, G. Lanckriet, B. Schölkopf, and B. K. Sriperumbudur, *Kernel choice and classifiability for rkhs embeddings of probability distributions*, Advances in Neural Information Processing Systems (Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, eds.), vol. 22, Curran Associates, Inc., 2009.

[GBR+12] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, *A kernel two-sample test*, Journal of Machine Learning Research **13** (2012), no. 25, 723–773.

[GG21] Z. Goldfeld and K. Greenewald, *Sliced mutual information: A scalable measure of statistical dependence*, 2021, `arXiv:2110.05279 [cs.IT]`.

[GS02] A. L. Gibbs and F. E. Su, *On choosing and bounding probability metrics*, 2002, arXiv:math/0209021 [math.PR].

[HRU+18] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, *Gans trained by a two time-scale update rule converge to a local nash equilibrium*, 2018, arXiv:1706.08500 [cs.LG].

[Hus17] F. Huszár, *Variational inference using implicit distributions*, 2017, arXiv:1702.08235 [stat.ML].

[JRV+24] S. Jayasumana, S. Ramalingam, A. Veit, D. Glasner, A. Chakrabarti, and S. Kumar, *Rethinking fid: Towards a better evaluation metric for image generation*, 2024, arXiv:2401.09603 [cs.CV].

[KJSM22] J. M. Kübler, W. Jitkrittum, B. Schölkopf, and K. Muandet, *A witness two-sample test*, 2022, arXiv:2102.05573 [cs.LG].

[KLL19] I. Kim, A. B. Lee, and J. Lei, *Global and local two-sample tests via regression*, 2019, arXiv:1812.08927 [stat.ME].

[KNS+19] S. Kolouri, K. Nadjahi, U. Simsekli, R. Badeau, and G. K. Rohde, *Generalized sliced wasserstein distances*, 2019, arXiv:1902.00434 [cs.LG].

[KSB+23] J. M. Kübler, V. Stimper, S. Buchholz, K. Muandet, and B. Schölkopf, *Automl two-sample test*, 2023, arXiv:2206.08843 [cs.LG].

[LBG+21] J.-M. Lueckmann, J. Boelts, D. S. Greenberg, P. J. Gonçalves, and J. H. Macke, *Benchmarking simulation-based inference*, 2021, arXiv:2101.04653 [stat.ML].

[NDC+22] K. Nadjahi, A. Durmus, L. Chizat, S. Kolouri, S. Shahrampour, and U. Şimşekli, *Statistical and topological properties of sliced probability divergences*, 2022, arXiv:2003.05783 [stat.ML].

[PBNF24] T. Pandeva, T. Bakker, C. A. Naesseth, and P. Forré, *E-valuating classifier two-sample tests*, 2024, arXiv:2210.13027 [stat.ME].

[PC19] G. Peyré and M. Cuturi, *Computational optimal transport: With applications to data science*, 2019.

[PS22] T. Papp and C. Sherlock, *Bounds on wasserstein distances between continuous distributions using independent samples*, 2022, arXiv:2203.11627 [stat.ML].

[Ras14] S. Raschka, *An overview of general performance metrics of binary classifier systems*, 2014, arXiv:1410.5330 [cs.LG].

[SKA+23] A. Schrab, I. Kim, M. Albert, B. Laurent, B. Guedj, and A. Gretton, *Mmd aggregated two-sample test*, 2023, arXiv:2110.15073 [stat.ML].

[STS+21] D. J. Sutherland, H.-Y. Tung, H. Strathmann, S. De, A. Ramdas, A. Smola, and A. Gretton, *Generative models and model criticism via optimized maximum mean discrepancy*, 2021, arXiv:1611.04488 [stat.ML].

[SVI+15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, 2015, arXiv:1512.00567 [cs.CV].

[SXGS20] A. Srivastava, K. Xu, M. U. Gutmann, and C. Sutton, *Generative ratio matching networks*, 2020, arXiv:1806.00101 [stat.ML].

[TR19] M. K. Titsias and F. J. R. Ruiz, *Unbiased implicit variational inference*, 2019, arXiv:1808.02078 [stat.ML].

[TvdOB16] L. Theis, A. van den Oord, and M. Bethge, *A note on the evaluation of generative models*, 2016, arXiv:1511.01844 [stat.ML].

[XHY+18] Q. Xu, G. Huang, Y. Yuan, C. Guo, Y. Sun, F. Wu, and K. Weinberger, *An empirical study on evaluation metrics of generative adversarial networks*, 2018, arXiv:1806.07755 [cs.LG].

[YYZ+23] M. Yang, C. Yang, Y. Zhang, Q. Bai, Y. Shen, and B. Dai, *Revisiting the evaluation of image synthesis with gans*, 2023, arXiv:2304.01999 [cs.CV].