GAN Precision-Recall Evaluation

Lex Meulenkamp
4721071

Abstract

1. Introduction

The evaluation of generative adversial networks (gan) is a challenging task. For gan evaluation the Fréchet Inception Distance (FID) is a popular metric. The authors of FID [1] have also shown that FID captures different kind of image artifacts. However, since it's a one dimensional score, the distinction between image fidelity and image diversity performance is vague. Does a bad FID score indicates a a gan with good fidelity samples or a gan with good sample diversity? Answers to such questions can only answered by manual inspection of the generated gan images. Also for detecting what kind of failure modes are present, one still needs to detect it by looking at the generated images. To address some of these issues, the authors from [4] created one of the first practical precision-recall (pr) metric for gan evaluation. On a high level, the pr metric captures a gans image fidelity by precision and image diversity by recall. Inspired by the framework of [4] multiple papers about pr-evaluation in the gan setting were published. Just like [4] the method of [5] outputs pr curve.
While [2] and [3] are both methods that output a single pr score.

compared their precision-recall (pr) metric with the metric from [4]. The main difficulty is that the latter outputs pr value pairs to form the pr curve, while the former outputs a two dimensional pr score. To be able to compare those two metrics; the metric output from [4] must also be summarized into a two dimensional pr score. Another option would be to extend the metric from [2] to be able to output pr curve.
Just as in traditional pr evaluation we can summarize the pr curve with (generalized) f-scores. The default value used by [4] is $\beta = 8$, which means that recall is valued 8 times as important as precision. The following formula with the β value is used to calculate f-scores for each pr pair in the pr curve. The maximum of the set of f-scores is taken in order to output a two dimensional pr point.

$F_\beta = (1+\beta^2)\frac{precision*recall}{(\beta^2*precision)+recall}$
Precision $= F_{\frac{1}{\beta}}$
Recall $= F_\beta$

This has also been adapted by [2] for their comparison experiments. Whether the default beta value is a appropriate choice to summarize the evaluation power of the pr curve has not been experimented and/or reported.

1.1. Current Questions

The pr curve in practice is a set of multiple two dimensional pr pairs. Curve evaluation contain a lot of information about a system, but in this setting each gan has its own pr curve. Thus, it can get cluttered if we analyze a large set of gans. So before we do curve analysis we might want to start first analyzing at a higher level. Meaning, summarizing the curves by a single two dimensional point. Thus, we need to decide on one β value for generating f-scores and we need a statistic (e.g. max, median, avg) to summarize those f-scores in order to output the final two dimensional pr score.

- *What kind of statistic should be used to summarize multiple f-scores into one pr score?*
- *What kind of beta value should be used or is it just a matter of preference between recall/precision. (Maybe a range of trustworthy beta scores)*
- *Does the distribution of f-scores generated by multiple beta values give insightful information for choosing a beta value and statistic?*
- *Extra, can we say something interesting about the different metrics?*

A central question which is most likely important for all the questions above:
Can we analyze when certain choices fail/succeed under certain experimental conditions?

2. Methods

For precision and recall metrics in the gan setting, we are comparing the similarity between two distributions, namely the real dataset R and the fake dataset F. Q=F is learned

2.1. Kmeans method

[4] is one of the first papers with both a theoretical and practical algorithm for precision and recall curve analysis for gans. Intuitively, we express both distributions as one component which can be generated by the other distribution, and the other component which can't be generated by the other distribution. Those components are mainly composed by the sets created by the intersection operator.
First the support sets:
$A = supp(R) \cap supp(F)$.
$B = supp(R) - A$.
$C = supp(F) - A$.

Based on the support sets, there are multiple distributions configurations possible for the three probability distributions μ_A, μ_B, and μ_C. Where the underscore denotes the support set of the probability distribution. A precision α and recall β pair, both α and $\beta \in (0, 1]$, are defined by the metric if the following equations are true.

$$R = \beta\mu_A + (1-\beta)\mu_B \tag{1}$$

$$F = \alpha\mu_A + (1-\alpha)\mu_C \tag{2}$$

For instance, perfect recall $\beta = 1$ is only possible if R can be decomposed just by the intersection distribution μ_A.

Getting the precision α and recall β values from equation 1 and 2 is difficult because we have to check whether there are suitable distributions μ_A, μ_B, and μ_C for all possible precision α and β values. The practical approach by [4] get the precision-recall pairs by using the kmeans method on the union feature space by a cnn from datasets R and F.
In more detail, by fixing $\alpha = \lambda\beta$ we can iterate over multiple λ values to obtain precision and recall pairs for a certain weighting factor λ. Equations 3 and 4 are used to calculate precision and recall pairs which are intersection operation equations for discrete histograms. Those histograms are approximated by the kmeans algorithm on the union feature space. For each dataset, we obtain its histogram by counting the amount of samples per cluster, thus its histogram is a one dimensional vector with its length equal to the amount of k clusters from kmeans. Those count histograms can then be normalized into a density histogram which essentially is an approximated distribution.

$$\text{Precision}(\lambda \in \Lambda) = \sum_{\omega \in \Omega} min\big(\lambda R(w), G(\omega)\big) \tag{3}$$

$$\text{Recall}(\lambda \in \Lambda) = \sum_{\omega \in \Omega} min\big(R(\omega), \frac{G(\omega)}{\lambda}\big) \tag{4}$$

$$\Lambda = \left\{ \tan\left(\frac{i}{(m+1)} . \frac{\pi}{2} \right) \mid i = 1, 2, \ldots, m \right\} \tag{5}$$

$$\textit{Curve} = \left\{ \Big(\text{Precision}(\lambda), \text{Recall}(\lambda) \Big) \mid \lambda \in \Lambda \right\} \tag{6}$$

All the $\lambda \in \Lambda$ form an equiangular grid of values which are used to obtain the precision and recall pairs from equation 3 and 4. Where $m \in \mathbb{Z}^+$ is the radial resolution which determiners the amount of precision-recall pairs we sample for the curve. Also looking at it differently, $\alpha = \lambda\beta$ is essentially a line equation such as $x = \lambda y$ where the line intersection with the distribution is the precision recall pair we need.

2.2. Classifier method

The authors from [5] mention that the histogram approach by [4] is restricted to discrete probability distributions. They remove this restriction by creating precision and recall curves from arbitrary probability distributions [5]. We have a mixture dataset from real and generated data Y, labeled generated with $Y = 0$ and real with $Y = 1$. A classifier $\hat{Y}$ which is trained on Y. Just as the kmeans method we use $\lambda \in \Lambda$ equation 5 values to get different precision-recall pairs. The weighting of the λ values is now used to weight the type I and type II error instead of the histograms.

$$\text{Type I error} = \alpha = P(\hat{Y} = 0 \mid Y = 1)$$

$$\text{Type II error} = \beta = P(\hat{Y} = 1 \mid Y = 0)$$

$$\text{Precision} = \lambda\alpha + \beta$$

$$\text{Recall} = \alpha + \frac{\beta}{\lambda}$$

2.3. KNN

The pr metric by [2] is a two dimensional pr score. Where the k from the knn algorithm is a flexible hyperparameter plus the feature extractor. Both the manifolds from dataset R and F are approximated by the knn algorithm. These manifolds are used to estimate the pr score. For instance, a fake sample can be either rejected or accepted based on it's distance from the approximated real dataset manifold. More specifically, the fake sample must have at least one real sample where its distance to this samples is smaller or equal to the real samples kth (real sample) neighbour. Different distance metrics could be used, but [2] uses equation 7.

First, the image samples are projected into a more meaningful space for the distance metric. For images, pretained cnn's as feature extractor are often used as starting point, but this aspect of the algorithm is domain dependant. Acceptation or rejection of a samples by a dataset is done by by equation 7, which uses the feature vector of the new sample ϕ, the set of existing feature vectors from a dataset Φ, and the knn algorithm. Where $NN_k(\acute{\phi}, \Phi)$ is the distance between sample feature vector $\acute{\phi}$ and its kth neighbour from the set of feature vectors Φ.

$$f(\phi, \Phi) = \begin{cases} 1, & \text{if } ||\phi - \acute{\phi}||_2 \leq ||\acute{\phi} - NN_k(\acute{\phi}, \Phi)||_2, \\ .. & \text{for at least one } \acute{\phi} \in \Phi \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Precision counts a fake sample as positive if it is contained in at least one real neighbourhood sphere. Based on the fake sample acceptation-rejection rate by the real manifold (knn approximated spheres) we can calculate precision as:
Precision(Φ_r, Φ_f) =

$$\frac{1}{|\Phi_f|} \sum_{\phi_f \in \Phi_f} f(\phi_f, \Phi_r) \tag{8}$$

The same goes for recall but our estimation is reversed, thus recall counts a real sample as positive if it is contained in at least one fake neighbourhood sphere. Again but now for the real samples, the real sample acceptation-rejection rate determines the recall score:
Recall(Φ_r, Φ_f) =

$$\frac{1}{|\Phi_r|} \sum_{\phi_r \in \Phi_r} f(\phi_r, \Phi_f) \tag{9}$$

2.4. Density and Coverage

The density and coverage metric by [3] is also a knn based metric which outputs a single two dimension pr score, which is largely based on the pr metric from [2]. This method also uses the distance metric from equation 7 with a (pretrained-cnn) feature extractor for more meaningful distances between image samples. They argue that the fake manifold which is used by [2] to estimate recall is often unreliable. "Since models tend to generate many unrealistic yet diverse samples, the fake manifold is often an overestimation of the actual fake distribution". [3] One could argue if the word "tend" is appropriate in the context of gan algorithms. The general behaviour of gans is highly volatile to hyperparameters, architecture, loss functions, gan tricks etc. One of the main problems of gans is a lack of diversity while the image fidelity performs better than most generative algorithms. However, if a gan generates diverse (unrealistic noisy) samples than the recall estimation approach by [2] becomes problematic. In such a case, the radii of multiple spheres from the fake manifold are quite large, thus increasing the chance that a real sample will be accepted even though its smallest distance to a fake sample in the feature space might be large. Instead of using recall [3] uses coverage, where coverage counts the fraction of real samples who has at least one fake sample contained in their neighbourhood sphere.

Precision metric by [2] counts a fake sample positive if it is contained in any real sample neighbourhood. This way of counting makes it more susceptible for outlier samples. Therefore, [3] proposes to use the density metric instead, where it counts how many real-sample neighbourhood spheres contain fake samples. Therefore, the weight of an outlier sample is reduced. Important to note, this way of counting makes it possible for the density measure upper bound to be larger than one.

With Φ_f as the set of fake feature vectors, Φ_r as the set of real feature vectors, and $k \in \mathbb{Z}^+$ from the knn algorithm, we have the following metrics:
Density, which counts how many real sample neighbourhood spheres contain fake samples.
(Φ_r, Φ_g) =

$$\frac{1}{k|\Phi_f|} \sum_{\phi_f \in \Phi_f} \sum_{i=1}^{|\Phi_r|} f(\phi_f, \Phi_{r_i}) \tag{10}$$

Coverage, which is the ratio of the real samples that are covered by the fake sample, defined as

$$\text{Coverage}(\Phi_r) =$$

$$\frac{1}{|\Phi_r|}\sum_{\phi_r \in \Phi_r} 1 \text{ if } \exists j \text{ s.t. } \phi_j \in \Phi_f \text{ is in neighbourhood of } \phi_r \tag{11}$$

3. Experiments

3.1. Variance drop

Sample diversity is often a problem for gans. The gan loss-function does not directly incentives for more sample diversity. Just a few very convincing generator samples can get the gan stuck in a local-minimum. **(Find quotes to support given statements)** A related diversity problem in gan, namely mode collapse is also often studied, where parts of the real distribution can't be generated by the gan. Gans often end up in a state where the generated samples form a (small) subspace of the real distribution.**(Prefer a specific quote to make define mode collapse more rigidly)**. Therefore, experiments which test the metrics sensitivity for diversity differences between distributions is important.
For the experiment, the real and fake distributions are both sampled from a multivariate Gaussian:

$$\text{Real data} \sim N\left[\begin{pmatrix}0\\0\\\vdots\\0\end{pmatrix} \in R^D, \quad \mathbb{I} \in R^{DxD}\right]$$

$$\text{Fake data} \sim N\left[\begin{pmatrix}0\\0\\\vdots\\0\end{pmatrix} \in R^D, \quad A = diag(x) \in R^{DxD}\right]$$

$$x = \left[x_0, \ldots, x_D\right]$$

$$x_i \sim \text{Ber}(0.5) \quad D \in \mathbb{N}^+$$

The density of the fake distribution becomes more (compact) dense as the experiment x vector contains more zero's. However, the mean vector from the fake distribution remains the same as the real distribution. So most samples from the fake distribution should still be generated by the real distribution.**(Test how true this statement is? or word it differently based on experiments)**
Thus, the expectation for the metrics in general is that recall will at least decrease significantly faster than precision. That means that the beta values lower than 1 (precision more weight) have high f scores and beta values higher than 1 (recall more weight) have low f scores.

Among the four metrics, k-means shows an inverse evaluation compared to the rest. Meaning that precision decreases harder than recall. This example also shows why taking the maximum is not necessarily a good approach for all evaluation settings. For traditional pr-evaluation we might want to take the classifier that gives the maximum f-score, but its important to keep in mind that in this evaluation setting, one

pr-curve and all the score derivations from it corresponds just to one single system (gan). For some settings a different (e.g. median) approach might capture the general evaluation consensus better among most pr-points. However, taking the maximum from a set f-scores is not necessarily better/worse than the median. For most cases, we can only analyze which approach is appropriate given that we have (almost) complete knowledge of both distribution. The variance between the approaches can be significant. For instance, if we take default $\beta = 8$ value then by maximum approach we get:
$Recall = Maximum(F_8) \approx 0.8$ and $Precision = Maximum(F_{\frac{1}{8}}) \approx 0.16$. While the median gives:
$Recall = Median(F_8) \approx 0.2$ and $Precision = Median(F_{\frac{1}{8}}) \approx 0.16$

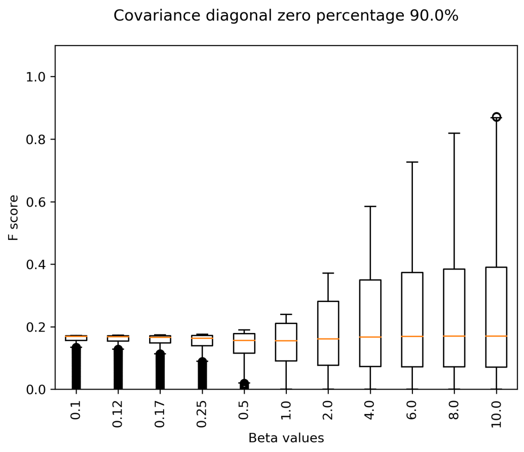

Figure 1: K-means method (k=20)

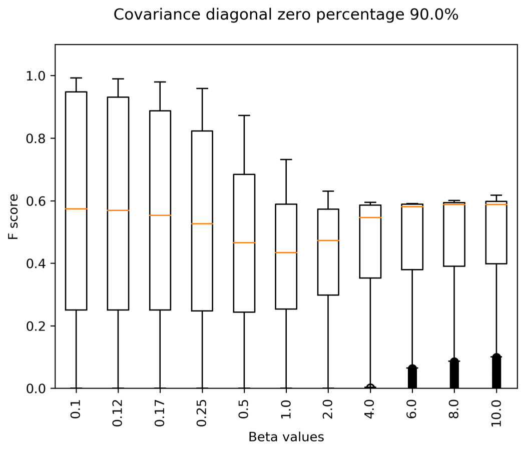

Figure 2: Classifier method (logistic model)

TODO classifier methods results The classifier plots of the distribution are more left-skewed. Also, if we take a classifier which has an easy time to distinguish the samples the error rate are low, thus also all the other linear combination, resulting in a degenerate pr curve (0,0). In such a case, we might argue that the classifier (adaboost) is too strict.

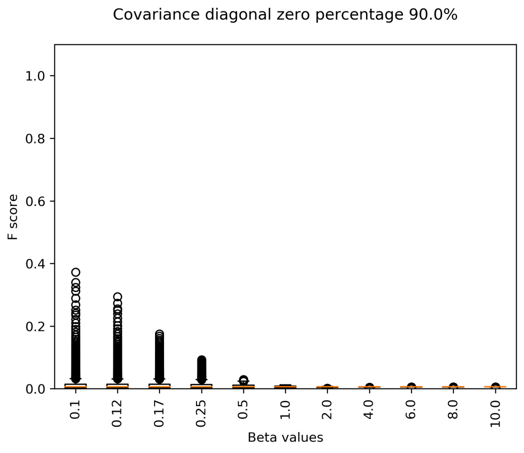

Figure 3: Classifier method (ada boost)

3.2. Mean shifts

First we start with two identical multivariate Gaussian distributions real and fake. The mean shifts experiments shifts the Gaussian mean by a fixed translation vector.

3.3. Mode dropping

[3] did an Gaussian mixture experiment where the real dataset and fake dataset are identical at first. The fake dataset gradually drops all mode data simultaneous except for the first mode. The following experiment is the same but with some slight variations.
The experiments are generated by a Gaussian mixture $\in R^D$ with 10 modes. Each mode uses the same identity matrix I^{DxD} and the mean vectors u^D are sampled uniformly by $U(-1,1)$, where $D \in \{2,4,6,8,16,32,64\}$.

TODO: shorter or gone
Note that:
As the dimensions scale up, there is more room for the mode centers to be sampled further apart by $U(-1,1)$. Missing samples due to simultaneous mode dropping should be easier to spot by th e metrics in high dimensions, which should results generally in lower precision-recall pairs.
In the Appendix section other dimensions are displayed.

Remarks
The Gaussian clusters have a lot of overlap in the problem where $D = 2$. This makes it for every metric more difficult to notice the drop changes. A better classifier (random forest) helps in such a case as it has more complex decision boundaries compared to logistic regression. The cluster method has in general more troubles to recognize the gradual drop. For most experiments it is less sensitive to the gradual dropping of fake samples.

Remarks about experiment setup:
Instead of dropping data for fake, try to resample the fake dataset until both sets are balanced. Resampling fake then happens with adjusted mode weights until the fake dataset is as large as the real dataset.

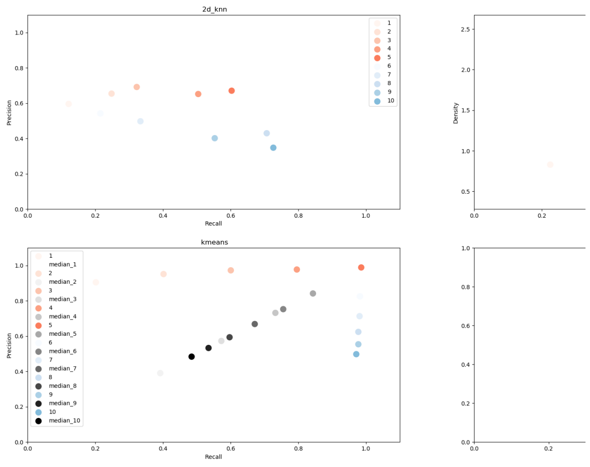

Figure 4: K-means method (k=20)

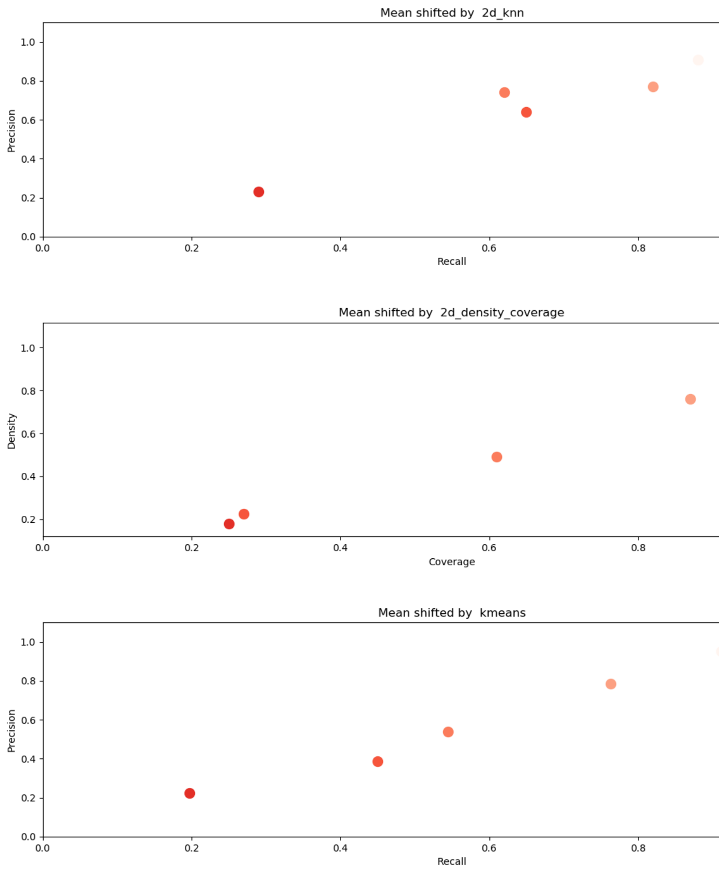

Figure 6: Classifier method (logistic model)

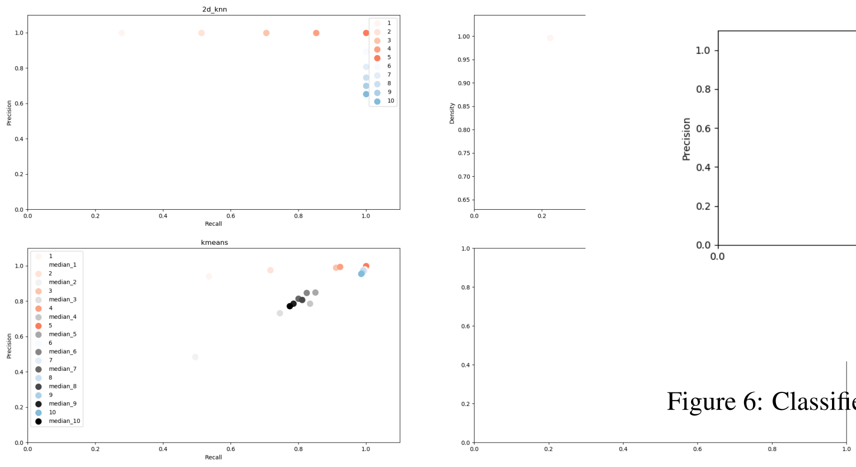

Figure 5: Classifier method (logistic model)

Experiments Included Synthetic multivariate Gaussian data.

- Decrease covariance matrix;
  to simulate fake data in a subspace of the real data.
- Shifting of mean vector
- Robustness to noise
- Gradual mode dropping
- Outlier test

4. Appendix

References

[1] 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. *arXiv preprint arXiv:1706.08500*, 2017.

[2] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems*, pages 3927–3936, 2019.

[3] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo. Reliable fidelity and diversity metrics for generative models. *arXiv preprint arXiv:2002.09797*, 2020.

[4] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems*,

pages 5228–5237, 2018.

[5] L. Simon, R. Webster, and J. Rabin. Revisiting precision and recall definition for generative model evaluation. *arXiv preprint arXiv:1905.05441*, 2019.