IFT 6756 - Lecture 9
Generative Adversarial Networks Part 2

This version of the notes has not yet been thoroughly checked. Please report any bugs to the scribes or instructor.

Scribes **Instructor:** Gauthier Gidel
Winter 2021: Francisco Gallegos, Albert Orozco Camacho, Martin Dallaire

1 Summary

In the previous lecture we talked about Generative Adversarial Networks, explained the concepts of supervised and unsupervised learning,log-likelihood and Generative Modeling to finally introduce the idea of Arithmetic in the Latent Space.
In this lecture the idea and intuitions about the Arithmetic in the Latent Space are explained more deeply and the evaluation metrics for Generative Adversarial Networks are elucidated . This lecture is based in the Salimans et al. [6], Sajjadi et al. [5] and Heusel et al. [1] papers.

2 Arithmetic in the latent space

The main motivation of the arithmetic in the latent space is to take an image and be able to change some features of it that are relatively natural. This idea is illustrated in figure 1. In this example, we have two original pictures within the first column. The subsequent columns of the figure, from left to right, correspond to the modified versions of the original picture. I.e., Age, Eyeglasses, Gender and Pose.

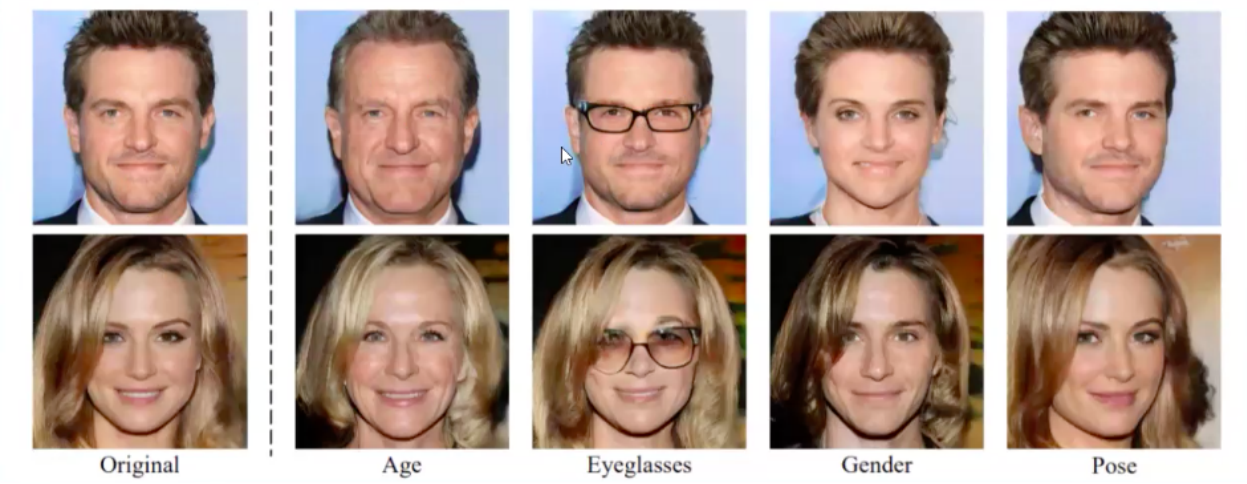


Figure 1: Figure: Latent space example.Source: `https://github.com/genforce/interfacegan`

Whereas this kind of manipulation is hard to do in practice, some people has achieved this task using Generative Adversarial Networks.

2.1 What is the latent space?

To answer this question, we use Plato's Allegory of the Cave, illustrated in figure 2. As observers, we are able to see the observed variable. Meanwhile, we are unable to see the latent variable which generates the observed variable trough a mapping from the latent space to the observed space. We are interested in knowing the latent variable as it contains meaningful information about the observed variable.

Figure 2: Figure: What is the Latent Space?
Source: `https://ourpolitics.net/the-allegory-of-the-cave-textual-analysis/`

If we apply this example to the concept of a generator, the bird represents the latent variable in the latent space which is projected as a shadow in to the observed space trough a mapping from the latent to the observed space. If we go back to figure 1, the latent variables are the features of the image such as the age, the color of the eyes and all high level attributes of the person.

What we are able to do in practice, is to use the reverse mapping of the observed variable to the latent variable, and then do a natural transformation of the latent variable. In the Allegory of the Cave example, if we move the bird from the left or to the right, it will move the observed variable. In this case the shadow of the bird, to the correspondent direction. Generative Adversarial Networks aim is to learn in the Arithmetic in the Latent Space, a way to manipulate the latent space in order to observe the corresponding modification in the image space.

2.2 Arithmetic in the Latent Space Example

The initial idea from Radford et al. [4] is to apply arithmetic in the latent space instead of the pixel space. Instead of looking at the directions in the pixel space, they took the correspondent latent variables generated from images of

people with a given characteristic and average them. Then, applying arithmetic operations to the averaged images, obtained the desired image.This is illustrated in figures 3 and 4. In this example, the direction of the averaged latent variable of man without glasses is subtracted from the direction of the averaged latent variable of man with glasses to finally add the direction of the averaged latent variable of a woman without glasses to obtain a woman with glasses image.

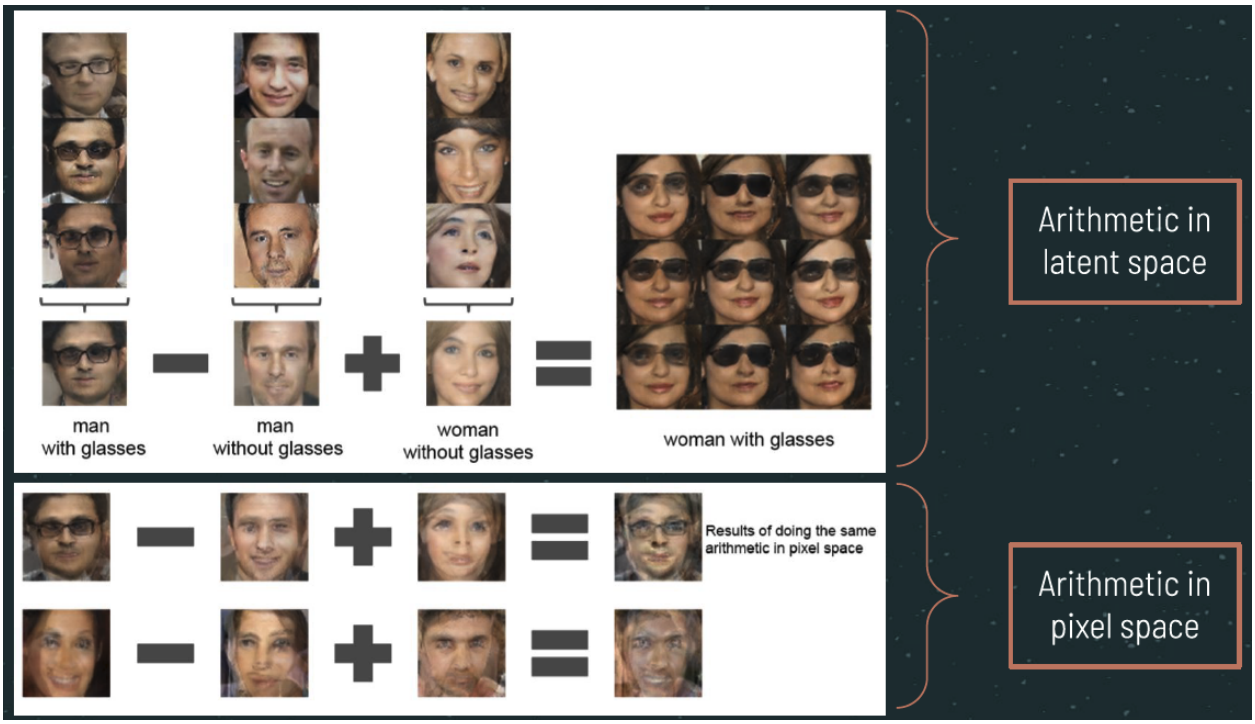


Figure 3: Figure: Initial idea from Radford et al.Source: Radford et al. [4]

In figure 3 the intuition behind this idea, which is to have a separability in the latent space of the averaged learned features, is illustrated. In this figure, the averaged latent variable of man with glasses is in one side of the latent space and the averaged latent variable of man without glasses is in the other side of this space.Then, if we subtract one to the other, we get the glasses direction in the latent space. Finally, if we add this direction to the averaged latent variable of women without glasses, we obtain the woman with glasses image from the latent space.
Thus, when learning the right mapping $g(x)$ which corresponds to the right data distribution, we obtain meaningful features from the latent space.
Other approach is using classic machine learning binary classifiers, such as Linear Simple Vector Machines, and the mapping from the images to latent space, learn the latent directions of these features. I.e., Age, Eyeglasses, Gender and pose. This idea is illustrated in figure 5
In the link Neural Photo Editor, using a simple interface for editing natural photos with generative neural networks, we can play with the latent space in order to create new images consistent with the pictures.

3 Evaluation of generative models

In the previous lecture, we saw (from Theis et al. [7]) that it was possible to have the following two scenarios:

- A model that generates great samples, but has poor loglikelihood;
- A model that generates poor samples, but has great loglikelihood.

As such, we need another method to evaluate sample fidelity and diversity, as log-likelihood is not enough by itself.

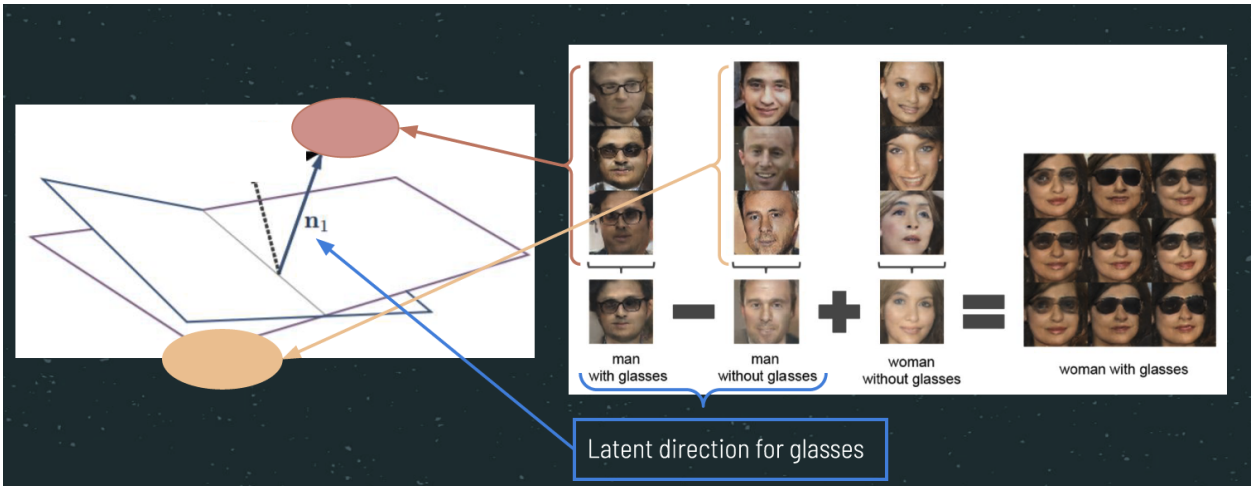


Figure 4: Figure: Representation of the direction of learned meaningful features in the latent space.
Source: Radford et al. [4]

3.1 Eyeballing method

The first suggested method is simply to look at the samples. This has the advantage of being exactly what we want our model to ultimately be good at: generating images that look realistic for a human observer. Furthermore, since we don't have a metric that is fully satisfactory, it will always be necessary to look at the generated images to assess their quality.

Moreover, we have reached a point in the quality of the image we are able to generate, where maybe it doesn't make sense to increase a given metric like inception (which we will discuss further below). As an example, if we look at figure 6, one might wonder if an improvement to the score of a given numerical metric will not simply result into an over-fitting to the metric instead of better images. Altogether, this is an open question which deserves further discussion.
One way to do such an human evaluation is to pay a lot of people to rate whether they think an image is real or not, using such services as Amazon Mechanical Turk.

As discussed above, human evaluation is a good evaluation method, although it has the following drawbacks:

- Too costly (when needing many human evaluators);
- Hard to reproduce;
- Difficult to evaluate the diversity of generated samples
- Does not provide an explicit, numerical metric for comparison purposes.

Figure 7 and 8 below summarize this issue well.

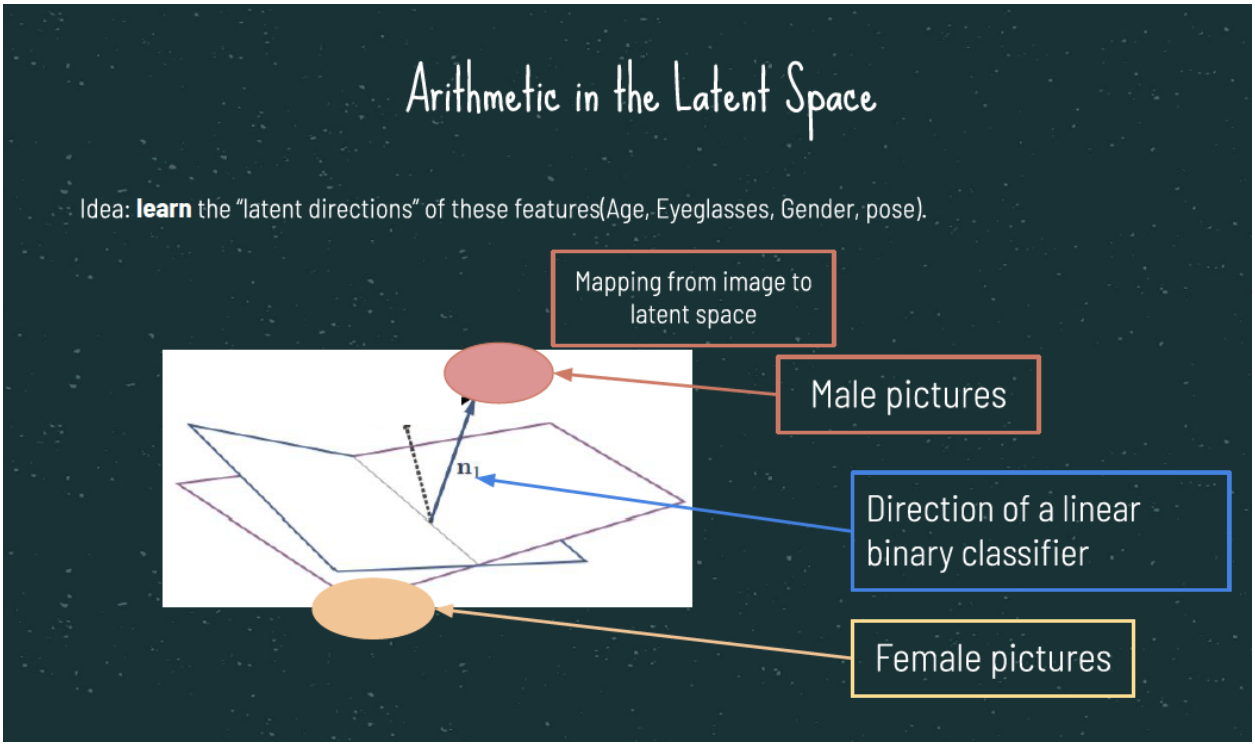


Figure 5: Figure: Representation of the direction of learned meaningful features in the latent space.
Source: `https://genforce.github.io/interfacegan/`

(notes for next section: david comments "we are using neural networks to evaluate neural networks, thus potentially reinforcing the bias of these networks". gauthier comments: "overfitting the metric is a risk, which is why it's important to also use human observers to assess the models")

3.2 Inception score

As mentioned in the previous section, humans cannot look at the full diversity of generated samples. We thus need to come up with an automated way of evaluating generative models.

One common technique is to use a pretrained classifier on these images. This is not without flaws, as using a neural network to evaluate a neural network risks reinforcing the bias of such models. However, this is the best method we have found so far, as it usually correlates reasonably well with human judgement. Finding a better method of automatically assessing model quality is currently an active area of research, however.

Let us first discuss the Inception score method. First introduced by Salimans et al. [2016], it involves using the Inception model (a standard pretained classifier). Note that the choice of the Inception model was mostly arbitrary. It was one of the models which, at the time, was amongst the best-scoring models on ImageNet. Still, it is still widely used today, as there is a very need for reproducibility in terms of metrics.

The Inception score works as follow:
It is first trained on the train set.
We then estimate the label distribution, given a specific generated image input, which corresponds to the conditional probability below.

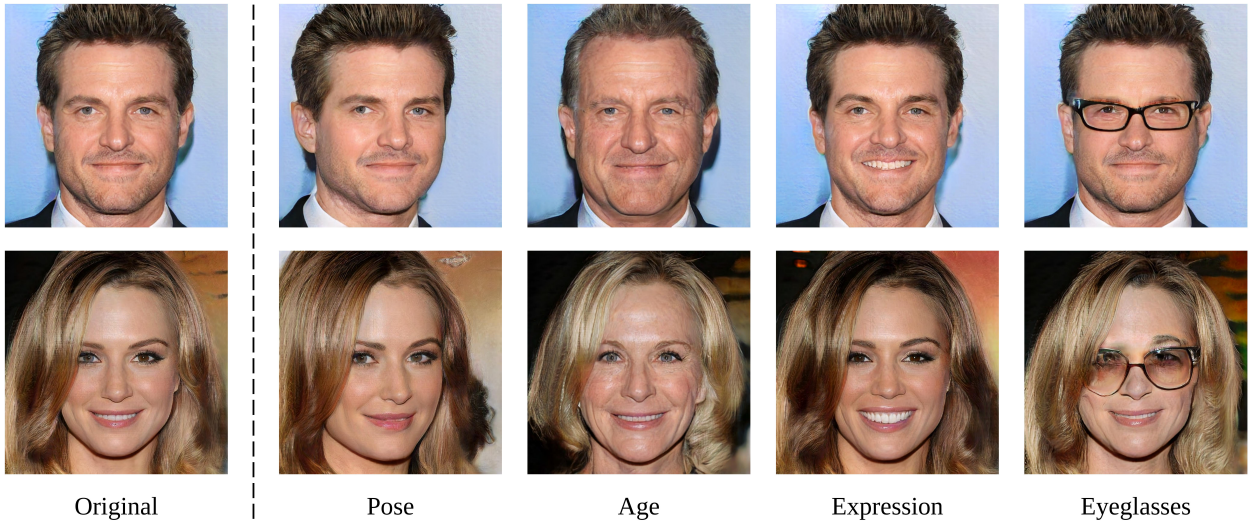


Figure 6: Figure: High-quality facial attributes editing results with InterFaceGAN.
Source: https://github.com/genforce/interfacegan

Figure 7: Comparing generated samples of two GANs trained on MNIST and achieving the same Fréchet inception distance of 49. It is difficult to numerically determine which is better. One is much more diverse, while the other produces better images.
Source article: Assessing Generative Models via Precision and Recall [5]

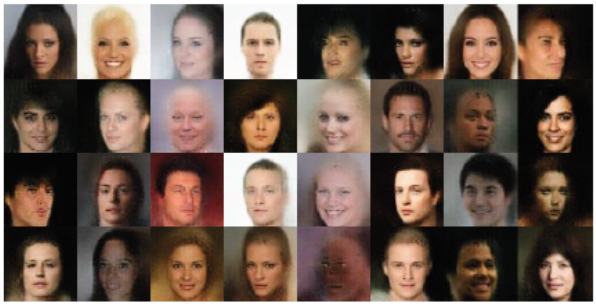
$$p_g(y|x) \approx f_\theta(x)$$

The function f above (and below) is simply our Inception model, with parameters θ, trained on our train set.

We then estimate the diversity of labels by looking at the overall distribution of generated images, according to our Inception model:

$$p_g(y) \approx \mathbb{E}_{x \sim p_g}[f_\theta(x)]$$

Ideally, the estimated conditional probability vectors should be very ”picky”, almost always giving high probability to a single class. To understand why, we need to consider that the Inception classifier is trained on the train set, and is presumably good at determining the ”class” of an image. A good generator should thus produce images that are also



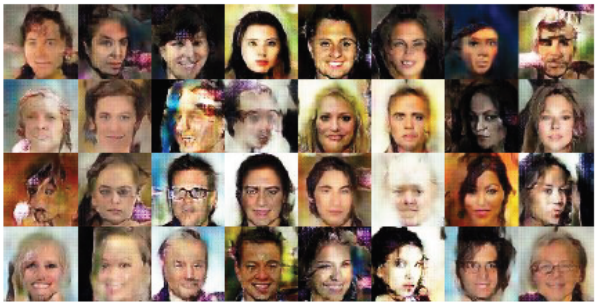


Figure 8: Comparing generated samples of two GANs trained on CelebA dataset and achieving Fréchet inception distance scores of 65 and 62, respectively. Note that a lower score is better and that the SOTA is below 10. Images on the left do not have artefacts, and look subjectively better, whereas images on the right have visual artefacts, but are more varied. We can see, from the better score attributed to the image on the right, that the FID metric prioritizes different things than a human would.
Source article: Assessing Generative Models via Precision and Recall

classified correctly. This corresponds to the **fidelity** of the generated image distribution.

On the other hand, on average, each "choices" should be evenly distributed, assuming that the train set was uniformly distributed. Thus, we want our Inception model to roughly equally predict each classes, given a set of generated images. This corresponds to the **diversity** of the generated image distribution.

Figure 9:
Top: A given generated image should be classified with high probability.
Bottom: A set of generated images should be uniformly classified into different bins
Source: https://medium.com/octavian-ai/a-simple-explanation-of-the-inception-score-372dff6a8c7a

Given the ideas above, in a situation where the classes are unbalanced, we will need to re-weight the probabilities accordingly.
Now remains the question of how to combine those two distributions into one metric. The choice made my the creators of the Inception score is to use the KL divergence of the two estimated distributions. Indeed, as we can see in figure 10, the KL divergence of two very different distributions is high, and this corresponds to our ideal scenario. On the other hand, if the two distributions are very similar, we will have a low KL divergence, which indicates that our generator is perhaps not performing as well as we would want.

Note: for the Inception score, high scores are good; this is in contrast with the FID score we will see later, for which low scores are good

Further note: an even higher KL divergence is possible, if the two distributions had peaks in different places entirely. However, such a situation should not arise in practice, on average, as it would require a peak for the marginal distri-

bution that is not the same place as the peak for the label distribution. One exception to this, potentially, is if one were to use the Inception score itself to train their model. Then, this situation could theoretically arise. For this reason (as well as others), one should never use the Inception score metric to train their model. In general, it is never acceptable to train on the evaluation metric we wish to use to assess a given model.

Final note: Given the framework above, we see that we need a labeled data set to use this metric.

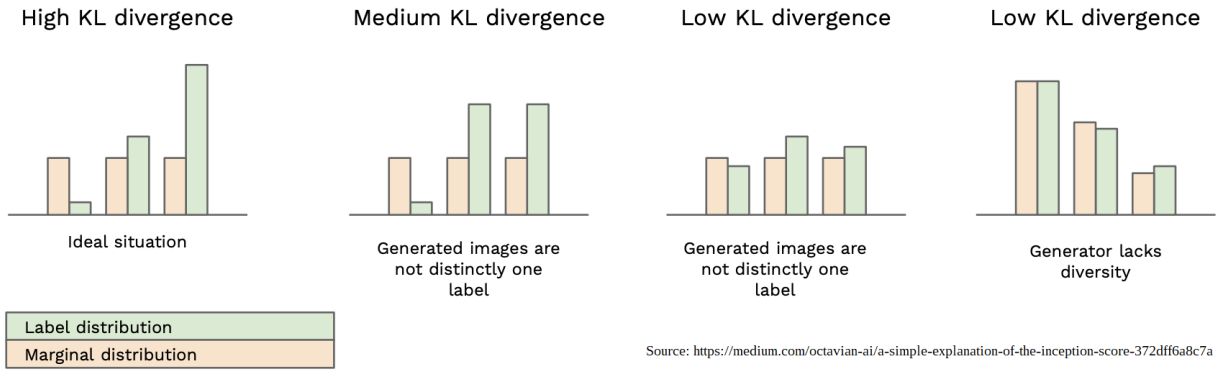


Figure 10: The first graph corresponds to a situation where the marginal distribution is even, and the label distribution is high. This is exactly what we want. A given image's estimated probability vector should heavily favor one label, while the overall set of images should have a high diversity of images. The second image corresponds to a slightly less ideal version, where more than one classes have high label probability for a given image. The third image corresponds to one where there is high diversity of images, but it becomes impossible for the Inception classifier to confidently classify any given generated image, thus indicating that the generated images are too far from the train set's. The fourth image corresponds to one where there is low diversity of images.

With all this in mind, the formula for the Inception score is, intuitively, as follow:

$$IS(G) := \exp(\mathbb{E}_{x\sim p_g}[KL(p(y|x)||p(y))])$$

The exponential function is used for re-scaling purposes, as the KL divergence uses logs.

3.3 Problems with the Inception score

We see that the Inception score correlates well with performance. That is why it has been used a lot for measuring GAN performances. However, the problem with the Inception score is threefold.

- Depends on the weights θ, which will be different for PyTorch and Tensorflow (the Inception model implementations for these two frameworks have different weights);
- Has no way to report overfitting; Copying the train set would instead lead to a great Inception score;
- Only cares about label diversity, and not diversity with a label class; If your generator outputs one image per label, consistently, and these images are easily classified by the Inception model, it will score very well even though it is not very diverse.

Interestingly, one can check the Inception score on the train set directly to get a sort of upper bound on the best score achievable.

3.4 Fréchet Inception Distance (FID)

We now present the idea of combining a strong assumption on our data (*Gaussian*) distribution with the previous Inception score.

Definition 1 (Fréchet Inception Distance)**.** *Assume the data distributions follow* $x_{data} \sim \mathbb{N}(\mu_1, \Sigma_1)$ *and* $x_{fake} \sim \mathbb{N}(\mu_2, \Sigma_2)$*, then we define a* distance *between them as:*

$$d(p_{data}, p_g) \equiv ||\mu_1 - \mu_2||_2^2 + Tr\left(\Sigma_1 + \Sigma_2 - 2\left(\Sigma_1\Sigma_2\right)^{1/2}\right)$$

where the first added term indicates the distance between the means of the chosen distributions, while the second term depicts their *covariance* distance. Observe that in Definition 1 the distance between Gaussian distributions is completely characterized, as such distributions are uniquely characterized by their mean and covariance matrices, respectively. Theorem 2 explains why is Definition 1 well defined as a (Euclidean) distance.

Theorem 2. *For any two covariance matrices* Σ_1*,* Σ_2 *of two arbitrary Gaussian distributions,*

$$d(\Sigma_1^{1/2}, \Sigma_2^{1/2}) = Tr\left(\Sigma_1 + \Sigma_2 - 2\left(\Sigma_1\Sigma_2\right)^{1/2}\right)$$

where d *corresponds to the Euclidean norm between matrices.*

Proof. Observe that for any two arbitrary squared $(n \times n)$ matrices A and B, $\langle A, B\rangle = Tr(AB^T)$ since by definition $\langle A, B\rangle = \sum_{1\leq i,j\leq n} A_{ij}B_{ij}$.
For any S_1, S_2 squared, positive symmetric matrices we have

$$\begin{aligned}
d(S_1, S_2) =& || S_1 - S_2 ||^2 && \textit{by definition}\\
=& \langle S_1 - S_2, S_1 - S_2\rangle\\
=& Tr\left((S_1 - S_2)(S_1 - S_2)^T\right) && \textit{since } S_k^T = S_k \textit{ for } k \in \{1, 2\}\\
=& T\left(S_1^2 - S_2S_1 - S_1S_2 + S_2^2\right) && \textit{since } Tr(AB) = Tr(BA) \textit{ for any } A, B \in R^{n\times n}\\
=& Tr\left(S_1^2 + S_2^2 - 2S_1S_2\right).
\end{aligned}$$

Setting $S_1 = \Sigma_1^{1/2}$ and $S_2 = \Sigma_2^{1/2}$, yields the desired result. Observe that this last claim is possible since the *square root operation* is uniquely defined over *positive squared matrices*, which is the case for the covariance matrix of a Gaussian distribution. □

Even though FID is well defined only for Gaussian distributions, it is not rigorous to extend it beyond. However, we can bring in the intuition of mapping data distributions to Gaussians, as occurs within other machine learning domains. From this point of view, we can imagine that in an Inception model's latent space, there should exist Gaussian distributions which can be compared using FID. Interestingly, this important point is made clear in the Appendix of [1], where the authors explain their computations over a *feature* space (last layer of an Inception model before classification) rather than directly on data.
Suppose there is a model that generates only one image per class. In principle, such model could have a perfect IS score; nevertheless, the Gaussian assumption on feature space would imply a bad FID score, as covariance matrices encode information about class diversity, which is assessed by their distances. This phenomenon is called *intra-class mode dropping*. Empirically, a good example of mode dropping would happen when a GAN is good at generating some class of image, yet always generates the same output, e.g., the same stylized number on the MNIST data set.
Of relative importance is to note the high complexity issues, when it comes to computing the FID mean and covariance matrices. A typical workaround would be to evaluate only on a small sample of data, and further improve evaluation via parallel processing. Another caveat worth to note is that it is still impossible to detect *overfitting* of a GAN model. Furthermore, FID is only well defined as distance for Gaussian distributions.

4 Summary of evaluation of generative models

So far we can summarize our analyses of generative model evaluation as follows.

1. **Human evaluation is very important**. The lack of a perfect metric, and even further, we don't know if a metric will lead to an accurate evaluation. Moreover, this problem is all about learning how to *"evaluate evaluators"*.

- **PROS** Since, most of the time, *pretty* pictures are presented on a GAN paper, human evaluation becomes really convenient. If the "final task" is, for example, to generate real-looking images, then there is no better way than putting a human to judge them.
- **CONS** There is no such thing as an *explicit evaluation value*; thus, making it hard to compare models that are close. Also, it is hard for humans to make sense about the diversity of the data set classes, while it is easy to fool them by *cherry-pick* the best outputs!

2. **Evaluation with a classifier**, such as IS or FID.

- **PROS** These approaches will give a *reproducible* metric that yields comparable scores.
- **CONS** There are more than one (robust) implementations of the Inception model, thus making any score dependent on their parameters. Moreover, these approaches do not take into account *generalization* (often under-explored on the GAN community), where we would like to ensure a model is not (*at least approximately*) reproducing the training set.
- There is, also, not a single number that accounts for fidelity vs diversity, although the idea is explored in [5].

Lastly, it is important to note that the scientist should **NOT** optimize over the presented metrics, as they are for evaluation purposes only, and thus, this will be considered as cheating!

5 Problems for evaluation

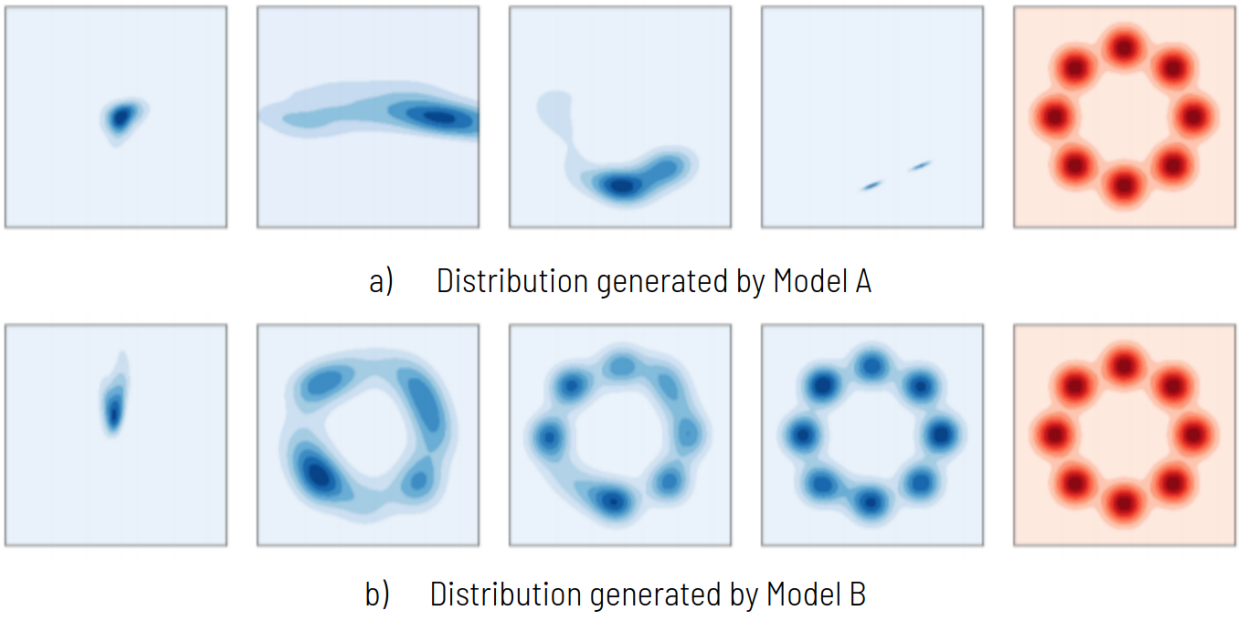


Figure 11: These are plots of the density of two mixed Gaussians in 2D space. The red, right-most plots, reflect 8 different modes for 8 local maxima of the true distributions. The other (blue) plots correspond to each model's learned densities. Taken from [3].

Recall the *intra-class mode dropping* problem, briefly presented in Section 3.4. Figure 11 presents an instance of such problem by mapping the 8 local maxima that a given Gaussian model is able to learn. Visually, this phenomenon occurs for *Model A*, as it is roughly generating data for a single mode, which is made evident on three of the presented blue plots.
Even though, empirically, Model B of Figure 11 seems to be the best one, we should be careful to conclude such statement. In particular, this example could have been *cherry-picked* especially to reflect mode dropping only. Hence,

we should clarify that this was the case for *a specific choice of hyperparameters*, including a random seed. Furthermore, in [2], the authors study the problem of comparing GANs in a systematic way: in particular, with sufficient hyperparameter search and computational budget, most models reach similar scores.

5.1 Challenges of a fair comparison

- **Which metric to use?** It is more convenient to use different metrics when dealing with high-dimensional data.
- **Which hyperparameters to use?** Consider a scenario where one wants to tune the hyperparameters of a "new" model, and compare it with a given baseline. A good approach for this would be to do *cross validation* on the new model, although this might cause spending roughly the same time for this process as it is for actually tuning the baseline, which may improve to match the new model.
- **Which random seed to use?** Random seeds shall not be cherry-picked; hence, must not be an objective of optimization. Also, it is important to make a several set of runs to ensure *reproducibility*.
- **Which data set to use** Try several ones!
- **Which amount of budget to put in different models?** Ideally, budget should be put equally for all considered models, but this is not easy, in general.
- **Which optimizer to use?** It is convenient to leverage particular optimizers for a specific model, which can improve performance.
- **Which NN architecture to use?** Same case as what happens with optimizers.

To summarize these challenges, we stress the importance of carefully evaluating GANs. Figure 12 depicts a set of performance results (FID) for different models, each trained on different data sets. Due to the high sensitivity of hyperparameters, it is very important to *always do several experiment runs* and average their performance scores.

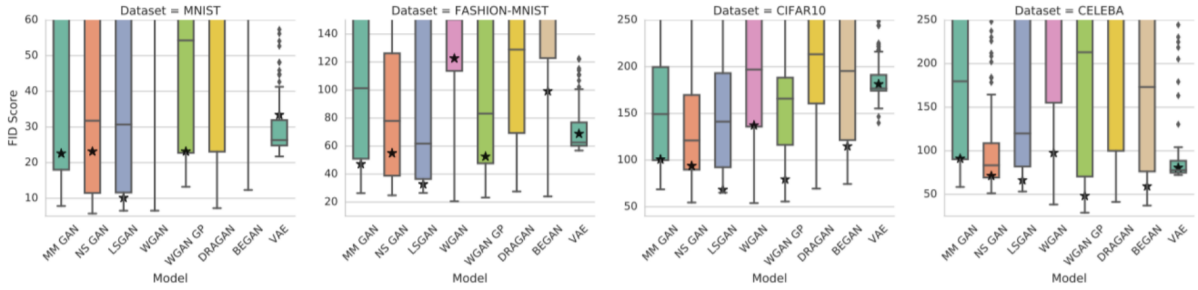


Figure 12: A *wide range* hyperparameter search (100 hyperparameter samples per model). Black stars indicate the performance of suggested hyperparameter settings. We observe that GAN training is extremely sensitive to hyperparameter settings and there is no model which is significantly more stable than others. Taken from [2].

6 Take Away

First of all, since *pretty pictures* often appear for any work where GANs are present, one must be careful about empirically evaluating the presented models, by objectively analyse their objective evaluation methods.
Furthermore, any experiment should *present results after performing several runs*, with different random seed. As we saw before, sensitive hyperparameters and cherry-picking of results may hide any inconvenient result that might break one's hypotheses. Also, this practice ensures statistical significance of results.
Finally, an *ablation study* would be an important tool to justify why a model works as a whole. Often, these kind of studies are performed by removing parts of a model (such as a block of neural layers) and reporting the respective results, which in an ideal case, should be worse than the proposed model. Moreover, these last two practices are

encouraged for any machine learning project. On this line, it is recommended to watch this video[1] for more context on the importance of statistical significance in scientific empirical work.

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, 2018.

[2] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are gans created equal? a large-scale study, 2018.

[3] L. M. Mescheder, S. Nowozin, and A. Geiger. The numerics of gans. *CoRR*, abs/1705.10461, 2017. URL http://arxiv.org/abs/1705.10461.

[4] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[5] M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall, 2018.

[6] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans, 2016.

[7] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models, 2016.

[1] https://youtu.be/42QuXLucH3Q