

# Theory on GANs: Adding Pixel-Wise Losses for Local Stability

by Uira Koshkina Jul 21, 2021

## Introduction

Generative Adversarial Networks are notoriously unstable and hard to train. Researchers have to employ a variety of tricks to make the adversarial training converge, constantly battling issues such as mode collapse and training divergence. Fortunately for us, when it comes to *Generative Compression Networks (GCN)*, the adversarial training seems to be on its best behaviour, causing much less trouble. Many tricks that are absolutely necessary for training GANs are not required in the compression pipeline. For example, we never observed a mode collapse, our GCNs train without gradient penalty, we don't need to add noise to our input samples to the decoder, and previous experiments indicate that pre-training of encoder-decoder is not necessary for getting stable discriminator training. Recently in the visual-loss team, we aim to understand how exactly our pipeline differs from standard GANs, what it means in terms of stability and convergence and why traditional GAN techniques are often not applicable.

One obvious difference is that in GCN, by nature of compression, we always have access to the ground truth image that we aim to generate. That allows us to use pixel-wise distortion losses in generator (encoder-decoder) training. In this blog post, we look at a toy example of GAN, examine the conditions of its convergence and discuss how our specific loss function impacts convergence.

## Convergence of GANs

In terms of game theory, we can think of GAN training as a two-player non-cooperative game. The first player is a generator  $G_\theta(z)$  with parameters  $\theta$  that wants to maximize its payoff  $g(\theta, \psi)$ , the second player is a discriminator  $D_\psi(x)$ , with parameters  $\psi$  that aims to maximize  $d(\theta, \psi)$ . The game is at a Nash equilibrium at  $(\theta^*, \psi^*)$  when neither player can improve its payoff by changing its parameters slightly. When GAN reaches a Nash equilibrium, we can say that it reached local convergence.

One method to train a GAN is to use a Simultaneous Gradient Descent, which can be thought of as a fixed point algorithm that applies an operator  $F(\theta, \psi)$  to the parameters of the generator and discriminator  $(\theta, \psi)$  respectively:

$$F(\theta, \psi) = (\theta, \psi) + hv(\theta, \psi),$$

where  $h$  is a learning rate and  $v(\theta, \psi)$  is the Jacobian of  $L$ , our gradient:

$$v(\theta, \psi) = \begin{bmatrix} -\nabla_\theta L(\theta, \psi) \\ \nabla_\psi L(\theta, \psi) \end{bmatrix}$$

[2] demonstrates that the convergence near an equilibrium point  $(\theta^*, \psi^*)$  can be assessed by looking at the spectrum of Jacobian of our update operator  $F'_h(\theta, \psi)$  at the point  $(\theta^*, \psi^*)$ :

- If **all** eigenvalues have absolute value **less than 1**, the system **converges to**  $(\theta^*, \psi^*)$  with a linear rate (our desired case).
- If there are any eigenvalues with absolute values **greater than 1**, the system **diverges**.
- If all eigenvalues have an absolute value **equal to 1** (lie on the unit circle), it can be convergent, divergent or neither, but if it is convergent, it will generally converge with a sublinear rate.

For detailed proof of why eigenvalues need to lie in a unit circle, see [4] proposition 4.4.1, page 226. This condition directly translates to the eigenvalues of the Jacobian of the gradient  $\nabla$  :

$$F(\theta, \psi) = (\theta, \psi) + hv(\theta, \psi)$$

$$F'(\theta, \psi) = I + hv'(\theta, \psi)$$

and therefore, the eigenvalues of the jacobian of our update operator  $F$  are:

$$\lambda = I + h\mu$$

where  $\mu$  are the eigenvalues of the jacobian of the gradient vector field  $v$ . From here, it is trivial to show that for a positive  $h$ ,  $|\lambda| < 1$  is only the case when  $\mu$  has a negative real part. More specifically:

- If **all** eigenvalues have **negative real-part**, the system **converges** with a linear convergence rate (with a small enough learning rate).
- If there are eigenvalues with **positive real-part**, the system **diverges**.
- If all eigenvalues have **zero real-part**, it can be convergent, divergent or neither, but if it is convergent, it will generally converge with a sublinear rate.

In fact, for a GAN with *positive real parts* to converge, we would require a negative step size, which is impossible since our  $h$  is always above 0.

Intuitively we can understand this requirement if we think of GAN updates as a fixed point iteration. Then we can think of the updates as the Euler discretization of the first-order ODE of the gradient vector:

$$\frac{dL(\beta)}{d\beta} = v(L(\beta))$$

Where  $\beta$  represents our parameters. The Euler step (with step size  $h$ ) becomes:

$$\beta^{k+1} = \beta^k + hv(L(\beta))$$

This means that our convergence analysis is, in fact, an analysis of the critical points, nodes and saddle points of the ODE. We can then visualise the behaviour of the system by plotting  $v(\theta, \psi)$  as a collection of arrows with a given magnitude and direction, each attached to a point  $(\theta, \psi)$ . Figure 1 visually shows how the behaviour of the system changes depending on the eigenvalues of the Jacobian of  $v(\theta, \psi)$ . Subplot c) demonstrates what it means to have negative real values and zero imaginary values, which would be the ideal case for the convergence of our fixed point.

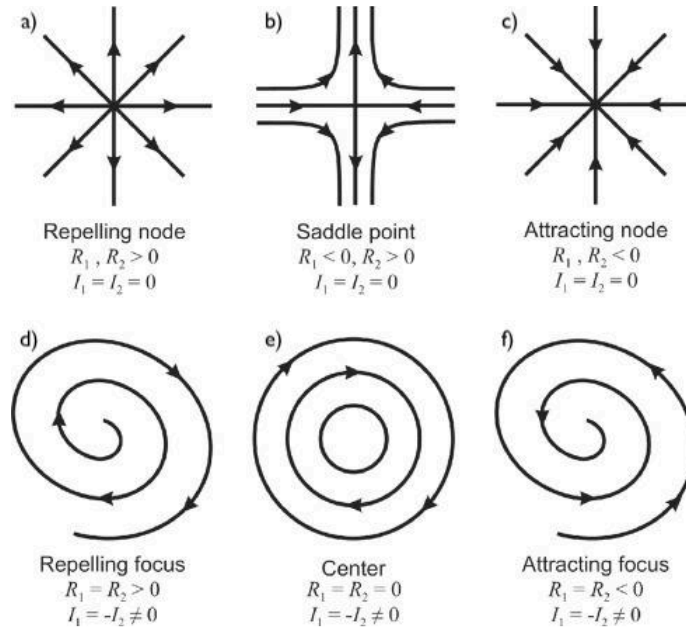


Figure 1. Classification of critical points of a 2D vector field according to eigenvalues.  $R_1, R_2$  denote the real parts of the eigenvalues of the Jacobian matrix and  $I_1, I_2$  - the imaginary parts. The figure was modified from Figure 1 in [5].

Having the criteria for convergence and a tool to visualise the behaviour in a 2D case, we can start the analysis of GAN models. To analyse the behaviour of different training methods and visualise the findings, the authors in [1] propose to look at a toy example called Dirac GAN.

## Dirac-GAN

The Dirac-GAN consists of a (univariate) generator distribution  $p_\theta = \delta_\theta$  and a linear discriminator  $D_\psi(x) = \psi x$ . The true data distribution  $p_D$  is given by a Dirac-distribution concentrated at 0.

Therefore, under this formulation, both the discriminator and the generator has exactly one parameter. This simplicity allows us to easily plot the vector field for the GAN in a 2D space. For a great explanation of vector fields and convergence of GANs, check out this [inFERENCE blog post](#).

The non-saturated GAN generator loss is:

$$\max_{\theta} L(\theta, \psi) = f(-\psi\theta)$$

where  $f(t) = -\log(1 + e^{-t})$ . The discriminator loss is:

$$\max_{\psi} L(\theta, \psi) = f(\psi\theta) - \text{const}$$

The gradient vector is then:

$$v(\theta, \psi) = \begin{bmatrix} -\psi f'(-\psi\theta) \\ \theta f'(\psi\theta) \end{bmatrix}$$

The system has a unique equilibrium point of the training objective, at point  $(\theta, \psi) = (0, 0)$ . Indeed, since  $f(0) = \text{const}$ ,  $L(\theta, 0) = L(0, \psi) = \text{const}$  for all  $\theta, \psi \in \mathbb{R}$ . Therefore  $(\theta^*, \psi^*) = (0, 0)$  is a Nash-equilibrium. Now, assuming that  $f(0) \neq 0$  we have that  $v(\theta, \psi) = 0$  only if  $\theta = \psi = 0$ .

Now we can analyse the convergence of the Dirac-GAN around the equilibrium point  $(\theta^*, \psi^*) = (0, 0)$  by calculating the Jacobian of the gradient vector field at the equilibrium point:

$$v'(\theta, \psi) = \begin{bmatrix} f''(-\theta\psi)\psi^2 & -f'(-\theta\psi) + f''(-\theta\psi)\theta\psi \\ f'(\theta\psi) + f''(\theta\psi)\theta\psi & f''(\theta\psi)\theta^2 \end{bmatrix},$$

$$v'(0, 0) = \begin{bmatrix} 0 & -f'(0) \\ f'(0) & 0 \end{bmatrix},$$

which has two eigenvalues,  $\pm f'(0)i$ , which are both on the imaginary axis. This means that the model is not convergent with a linear rate, as it has zero real parts and we expect to see a circular vector field.

We can visualise the training behaviour of Dirac-GAN by plotting the vector fields for the gradient  $v$  and the current state of the system. Figure 2 demonstrates the training behaviour of Dirac-GAN with non-saturating GAN loss in the first 500 steps. The left panes show the gradient vector field of Dirac-GAN (arrows) and the current state of the parameters in blue. An error starting at a given point  $(\theta, \psi)$  shows the value of the gradient  $v(\theta, \psi)$  at that point. The right panel shows the ground-truth value of the generator parameter  $\theta^* = 0$  in green and the current value of the learned  $\theta$  in blue. The orange line represents the angle of the gradient. Dirac GAN with non-saturating loss converges very slowly, with a non-linear rate.

Figure 2. Training behaviours of Dirac-GAN with the non-saturating loss. Left: Vector field and the training state of the system visualise for 500 steps. The starting position is marked in red. Right: Current value of the generator parameter  $\theta$ . The green line indicated the value of the gt parameter  $\theta^* = 0$ , and the blue line shows the estimated  $\theta$ . The orange line shows the current angle of the gradient.

## Adding MSE

Above we look at the non-saturating GAN loss, used in standard generative modelling. Now, let us look at what happens once we adapt this to GCN loss, by adding MSE to the loss for the generator parameters  $\theta$ . The discriminator loss stays the same, but the generator loss becomes:

$$\max_{\theta} L(\theta, \psi) = f(-\psi\theta) - (0 - \theta)^2$$

The gradient vector field now has an additional  $-2\theta$  term is the  $\theta$  partial derivative:

$$v(\theta, \psi) = \begin{bmatrix} -\psi f'(-\psi\theta) - 2\theta \\ \theta f'(-\psi\theta) \end{bmatrix}$$

The Jacobian of the vector field becomes:

$$v'(0, 0) = \begin{bmatrix} -2 & -f'(0) \\ f'(0) & 0 \end{bmatrix},$$

And eigenvalues of the Jacobian of the gradient vector field at the equilibrium point is  $-1 \pm \sqrt{1 - f'(0)}$ , both of which have negative real parts and 0 imaginary parts, which means that the system should be locally convergent around  $(0, 0)$ . The lack of imaginary parts means that there should not be any circular non-convergent behaviour in the gradient vector field. If we now run the Dirac GAN training with this updated generator loss function, we should see convergence and attracting gradient field if the theory matches the practical results. We demonstrate the results in Figure 3.

Figure 3. Training behaviours of Dirac-GAN with the non-saturating loss **and added MSE**. Left: Vector field and the training state of the system visualise for 500 steps. The starting position is marked in red. Right: Current value of the generator parameter  $\theta$ . The green line indicated the value of the gt parameter  $\theta^* = 0$ , and the blue line shows the estimated  $\theta$ . The orange line shows the current angle of the gradient.

### Vanilla GAN loss

We can repeat the analysis for the vanilla GAN loss function. The gradient vector is then:

$$v(\theta, \psi) = \begin{bmatrix} -\psi f'(\psi\theta) \\ \theta f'(\psi\theta) \end{bmatrix}.$$

As before, the system has a unique equilibrium point of the training objective at the point  $(\theta^*, \psi^*) = (0, 0)$ .

The Jacobian of the gradient vector field at the equilibrium point:

$$v'(\theta, \psi) = \begin{bmatrix} -f''(\theta\psi)\psi^2 & -f'(\theta\psi) - f''(\theta\psi)\theta\psi \\ f'(\theta\psi) + f''(\theta\psi)\theta\psi & f''(\theta\psi)\theta^2 \end{bmatrix},$$

$$v'(0, 0) = \begin{bmatrix} 0 & -f'(0) \\ f'(0) & 0 \end{bmatrix},$$

which has two eigenvalues,  $\pm f'(0)i$ , which are both on the imaginary axis. This means that the model is not convergent as it has zero real parts and we expect to see a circular vector field. Figure 4 demonstrates that Dirac-GAN with vanilla GAN loss is divergent around the equilibrium  $(0, 0)$ .

Figure 4. Training behaviours of Dirac-GAN. Left: Vector field and the training state of the system visualise for 500 steps. The starting position is marked in red. Right: Current value of the generator parameter  $\theta$ . The green line indicated the value of the gt parameter  $\theta^* = 0$ , and the blue line shows the estimated  $\theta$ . The orange line shows the current angle of the gradient.

However, for similar reasons as above, the behaviour completely changes once we add the MSE component to the generator loss. The gradient vector field now has an additional  $-2\theta$  term in the  $\theta$  partial derivative:

$$v(\theta, \psi) = \begin{bmatrix} -\psi f'(\psi\theta) - 2\theta \\ \theta f'(\psi\theta) \end{bmatrix}$$

The Jacobian of the vector field becomes:

$$v'(0, 0) = \begin{bmatrix} -2 & -f'(0) \\ f'(0) & 0 \end{bmatrix},$$

And, just like in the case of non-saturating loss, eigenvalues of the Jacobian of the gradient vector field at the equilibrium point are  $-1 \pm \sqrt{1 - f'(0)}$ .

Figure 5. Training behaviours of Dirac-GAN **with added MSE**. Left: Vector field and the training state of the system visualise for 500 steps. The starting position is marked in red. Right: Current value of the generator parameter  $\theta$ . The green line indicated the value of the gt parameter  $\theta^* = 0$ , and the blue line shows the estimated  $\theta$ .

The orange line shows the current angle of the gradient.

This analysis shows that adding MSE has the same impact as gradient regularisation and instance noise, which also remove the circular behaviours in the gradient field and force the negative real part in the eigenvalues. This would explain why these solutions were not impactful when applied to GCNs.


## Conclusion

Adding MSE loss for the generator parameter completely changes the training behaviour of the system, making it converge and do it fast. Meaning that MSE loss acts as a regulariser. This can potentially explain why our GCNs, which are always trained with MSE components in the loss, show much better convergence than standard GANs.

## References

- [1] Mescheder, Lars, Andreas Geiger, and Sebastian Nowozin. "Which training methods for GANs do actually converge?." *International conference on machine learning*. PMLR, 2018. <https://arxiv.org/pdf/1801.04406.pdf>

[2] Mescheder, Lars, Sebastian Nowozin, and Andreas Geiger. "The numerics of gans." *arXiv preprint arXiv:1705.10461* (2017). <https://arxiv.org/pdf/1705.10461.pdf>

[3] Huszár Ferenc "GANs are Broken in More than One Way: The Numerics of GANs."  [GANs are Broken in More than One Way: The Numerics of GANs](#)

[4] Bertsekas, Dimitri P. "Nonlinear programming." (1999). <https://nms.kcl.ac.uk/osvaldo.simeone/bert.pdf>

[5] Theisel, Holger, and Tino Weinkauf. "Vector field metrics based on distance measures of first order critical points." (2002). [http://wscg.zcu.cz/wscg2002/Papers\\_2002/D49.pdf](http://wscg.zcu.cz/wscg2002/Papers_2002/D49.pdf)

## Comments



**Chris Finlay** *Research Scientist*

Jul 22, 2021 (10:53)  Edit   2  0

Very well written article! That analysis is really nice.

I think, further to our discussion in yesterday's group meeting, you could get away with replacing MSE  $\|x - y\|^2$  with any coercive function  $g(x, y)$ . Coercive just means that you can always bound the function below by a quadratic "bowl":

$$g(x, y) \geq M\|x - y\|^2$$

for some  $M > 0$ . This is kinda the extension of positive definite-ness from linear algebra to functions.

Then, you can always bound the eigenvalues of your ODE system by the eigenvalues of the MSE system (where now you're adding in  $M\|x - y\|^2$  instead of the usual  $\|x - y\|^2$ ). So the general case, with a coercive function, boils down to the MSE analysis you've already done 😊

To keep the eigenvalues real, you might also need to have an upper bound  $g(x, y) \leq N\|x - y\|^2$ , but I'm not sure exactly how that'll play out.

Some functions that are coercive:

-- any metric (I think)

-- bilinear forms  $\langle x - y, A(x - y) \rangle$  where  $A$  is positive definite. So eg distortion metrics like distance in Fourier space or Wavelet space are coercive.

But LPIPS I think is not, because I think there are scenarios where  $\text{LPIPS}(x, y) = 0$  even when  $x \neq y$  (I think, could be wrong on this)