Generative Adversarial Networks.

Adalberto Claudio Quiros

March 28, 2019

Computing Science Department

Generative Adversarial Networks - Overview

GANs:

- Model.
- Zero-Sum Game and Nash Equilibrium.
- Optimal Point.
- Maximum Likelihood Models vs GANs.
- GAN Problems.
- Evaluation.

Models:

- DCGAN.
- WGAN/WGAN-GP.
- SNGAN.
- ProGAN.
- Conditional GANs.
- SAGAN.
- BigGAN.

Generative Adversarial Networks - Model

Generator:

- Generates fake samples as close as possible to the real ones, p_g approximates real distribution $p_{data}(x)$ through a map from the prior $p_z(z)$.
- $G(z;\theta_g)$ Function that maps a noise vector $\vec{z} = \{z_1, ..., z_m\}$ to a output vector $\vec{x} = \{x_1, ..., x_n\}$.

$$G : \mathfrak{R}^m \rightarrow \mathfrak{R}^n$$

Discriminator:

- Distinguishes between real and fake samples. Probability that $\vec{x}$ comes from the real data rather than p_g
- $D(x;\theta_d)$ Function that maps a input vector $\vec{x} = \{x_1, ..., x_n\}$ to scalar, zero for real samples, and ones for real ones.

$$D : \mathfrak{R}^n \rightarrow \mathfrak{R}$$

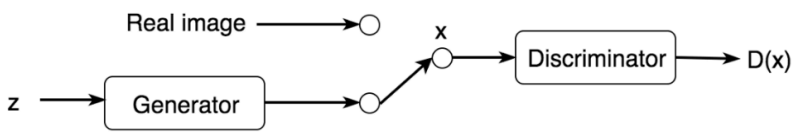


Figure 1: GAN block representation.

Generative Adversarial Networks - Model

Cost Function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[log\ D(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$

- Blue: Discriminator
- Green: Generated data distribution.
- Black: Real data distribution.

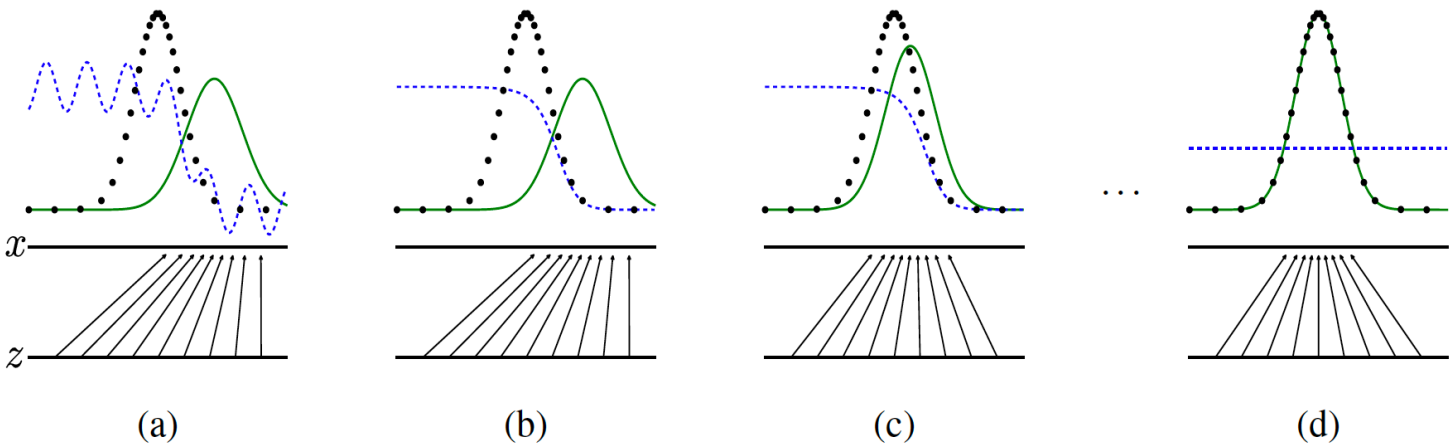


Figure 2: Generated data distribution over training [1].

Training:

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 3: Training Algorithm

Gans As Zero-sum Game And Nash Equilibrium

Zero-Sum Game:

- Conflict game, situation in which participants gains are other players losses.
- The addition of gains and losses of the participant's game are zero.

Nash Equilibrium:

- Proposed solution to a non-cooperative game, conflict game, in which each participant knows the equilibrium strategies and **no player has nothing to gain by changing their own strategy.**

Gans As Zero-sum Game And Nash Equilibrium

Zero-Sum Game: Prisoner's dilemma.

- Two criminals that committed a crime together are apprehended. When they are simultaneously interrogated each have 2 options: 'snitch' and betray each other, or cooperate and keep quiet.
- If both of them cooperate they get a 1 year sentence, if one of them cooperates and the other betrays him, they get 3 year and none respectively, and if both of them 'snitch', both of them get 2 years each.
- **Nash Equilibrium:** Both deflect, no better outcome by changing the decision unilaterally.

Prisoner B / Prisoner A	**Prisoner B stays silent (*cooperates*)**	**Prisoner B betrays (*defects*)**
Prisoner A stays silent (*cooperates*)	Each serves 1 year	Prisoner A: 3 years Prisoner B: goes free
Prisoner A betrays (*defects*)	Prisoner A: goes free Prisoner B: 3 years	Each serves 2 years

Figure 4: Pay-off representation.

Optimal Point

Let's assume a fixed Generator $G(z)$, and find the optimal Discriminator $D^*(x)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[log\ D(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$

If Discriminator $D(x)$ wants to maximize the cost function $V(D,G)$, which Discriminator achieves this maximum:

$$V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[log\ D(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))] =$$

$$\int_x p_{data}(x) log\ D(x) dx + \int_z p_z(z) log(1 - D(G(z))) dz =$$

$$\int_x (p_{data}(x) log\ D(x) dx + p_g(x) log[1 - D(x))]) dx$$

$$\frac{\partial V(D,G)}{\partial D} = 0 \rightarrow p_{data}(x) \frac{1}{D(x)} - p_g(x) \frac{1}{1 - D(x)} = 0$$

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Optimal Point

What's the cost function for the Optimal Discriminator $D^*(x)$?

$$\min_G V(D^*, G) = \int_x (p_{data}(x) log\ D^*(x) dx + p_g(x) log(1 - D^*(x)))) dx =$$

$$\int_x (p_{data}(x) log[\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}] + p_g(x) log[\frac{p_g(x)}{p_{data}(x) + p_g(x)}]) dx =$$

$$\int_x (p_{data}(x)(-log(2)) + p_g(x)(-log(2))) dx +$$

$$\int_x (p_{data}(x) log[\frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x)}{2}}] + p_g(x) log[\frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x)}{2}}]) dx =$$

$$\int_x (p_{data}(x) + p_g(x))(-log(2))) dx +$$

$$2[\frac{1}{2} D_{KL}(p_r(x) || \frac{p_{data}(x) + p_g(x)}{2}) \ + \ \frac{1}{2} D_{KL}(p_g(x) || \frac{p_{data}(x) + p_g(x)}{2})] \rightarrow$$

$$\min_G \mathsf{V}(\mathsf{D}^*, \mathsf{G}) = \mathsf{D}_{\mathsf{JS}}(\mathsf{p}_\mathsf{r}(\mathsf{x}) || \mathsf{p}_\mathsf{g}(\mathsf{x})) - \mathsf{log}(4)$$

Optimal Point

Assuming a fixed $G(z)$, and optimal $D^*(x) \rightarrow$

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

$$\min_G V(D^*, G) = D_{JS}(p_r(x)||p_g(x)) - log(4)$$

So minimizing our GAN cost is the same as minimizing the Jensen-Shannon Distance between $p_r(x)$ and $p_{data}(x)$:

$$D_{JS}(p_r(x)||p_g(x))$$

For our GAN optimal point, our generator $G^*(z)$ matches the real data:
$p_r(x) = p_{data}(x)$

$$V(D^*, G^*) = -log(4)$$

This is the Nash Equilibrium point.

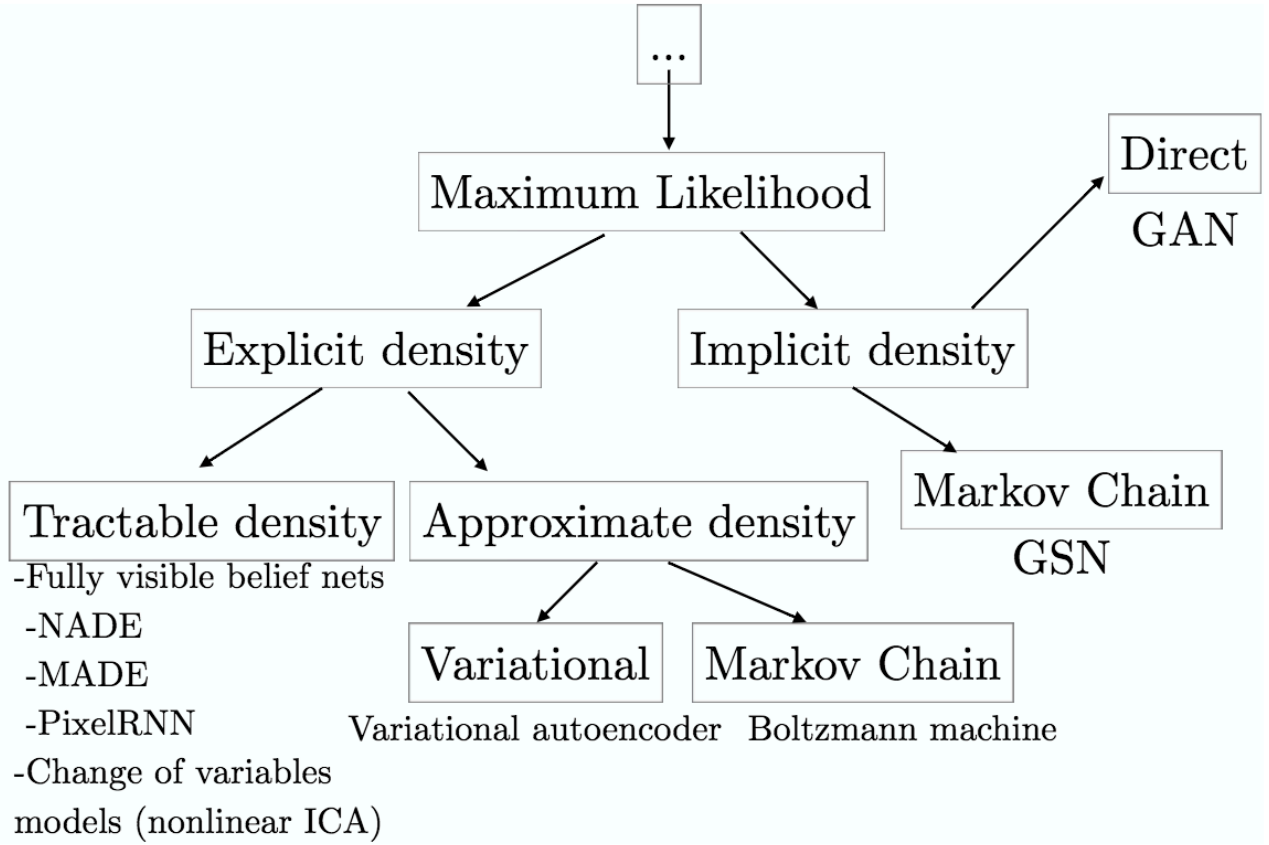


Figure 5: Generative Models

Maximum Likelihood Models Vs Gans

Many generative models maximize the likelihood given a model parameter θ:

$$\theta^* = arg\max_{\theta} \prod_{i=1}^{N} p(x_i/\theta)$$

These can be seen as minimizing the KL Divergence between the data distribution $p(x)$, and the approximation $q(x)$:

$$\lim_{m\to\infty} \max_{\theta} \frac{1}{m} \sum_{n=1}^{m} log(p_\theta(x^{(i)})) = \max_{\theta} - \int_x p_r(x) log(p_\theta(x)) dx =$$

$$\min_{\theta}[\int_x p_r(x) log(p_r(x)) dx - \int_x p_r(x) log(p_\theta(x)) dx] = \min_{\theta} D_{KL}(P_r(x)||P_\theta(x))$$

Maximum Likelihood Models Vs Gans

KL Divergence is a non-symmetrical function:

$$D_{KL}(P_r(x)||P_\theta(x)) = \int_x p_r(x) log(\frac{p_r(x)}{p_\theta(x)})dx$$

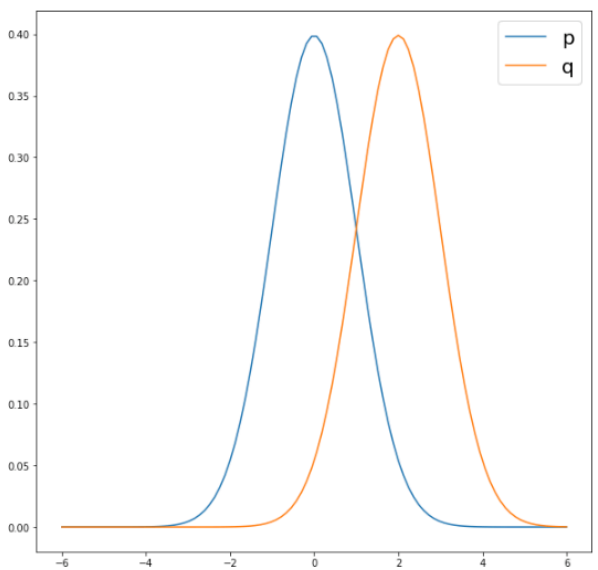
$$D_{KL}(P_r(x)||P_\theta(x)) \neq D_{KL}(P_\theta(x)||P_r(x))$$

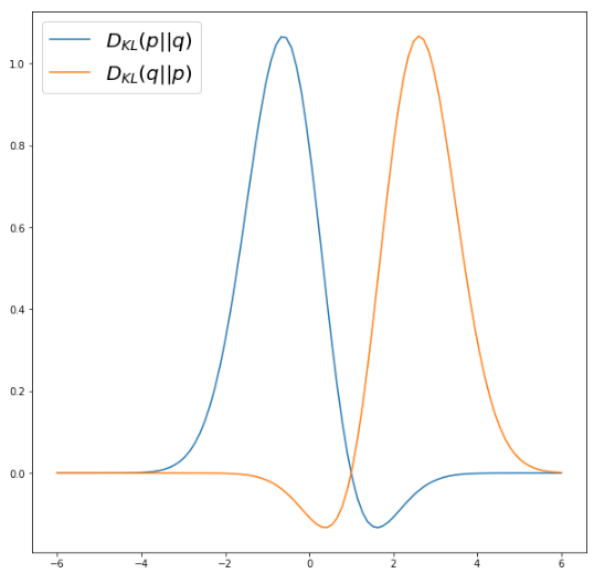
- $D_{KL}(P_r(x)||P_\theta(x))/x \geq 3 \rightarrow$ Penalizes for the lack of modes in $p_\theta(x)$ where $p_r(x) \neq 0$, it doesn't penalize $p_\theta(x)$ having a mode where $p_r(x) = 0$. Once the density is on $P_\theta(x)$, it is not redistributed if $P_r(x) = 0$ there. **Poorer quality but more diverse samples.**
- $D_{KL}(P_\theta(x)||P_r(x))/x \leq -3 \rightarrow$ No penalty for missing modes in $P_\theta(x)$ where $P_r(x) \neq 0$. **Better quality but less diversity in samples.**

Maximum Likelihood Models Vs Gans

$$D_{KL}(P_r(x)||P_\theta(x)) = \int_x p_r(x) log(\frac{p_r(x)}{p_\theta(x)}) dx$$

- $D_{KL}(P_r(x)||P_\theta(x))/x \geq 3 \rightarrow$ Penalizes for the lack of modes in $p_\theta(x)$ where $p_r(x) \neq 0$, it doesn't penalize $p_\theta(x)$ having a mode where $p_r(x) = 0$. Once the density is on $P_\theta(x)$, it is not redistributed if $P_r(x) = 0$ there. **Poorer quality but more diverse samples.**
- $D_{KL}(P_\theta(x)||P_r(x))/x \leq -3 \rightarrow$ No penalty for missing modes in $P_\theta(x)$ where $P_r(x) \neq 0$. **Better quality but less diversity in samples.**





Maximum Likelihood Models Vs Gans

Jensen-Shannon Divergence:

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q||\frac{p+q}{2})$$

- Symmetrical function.
- Penalizes both cases:

$$p(x) > 0, q(x) \to 0, and\ q(x) > 0, p(x) \to 0$$

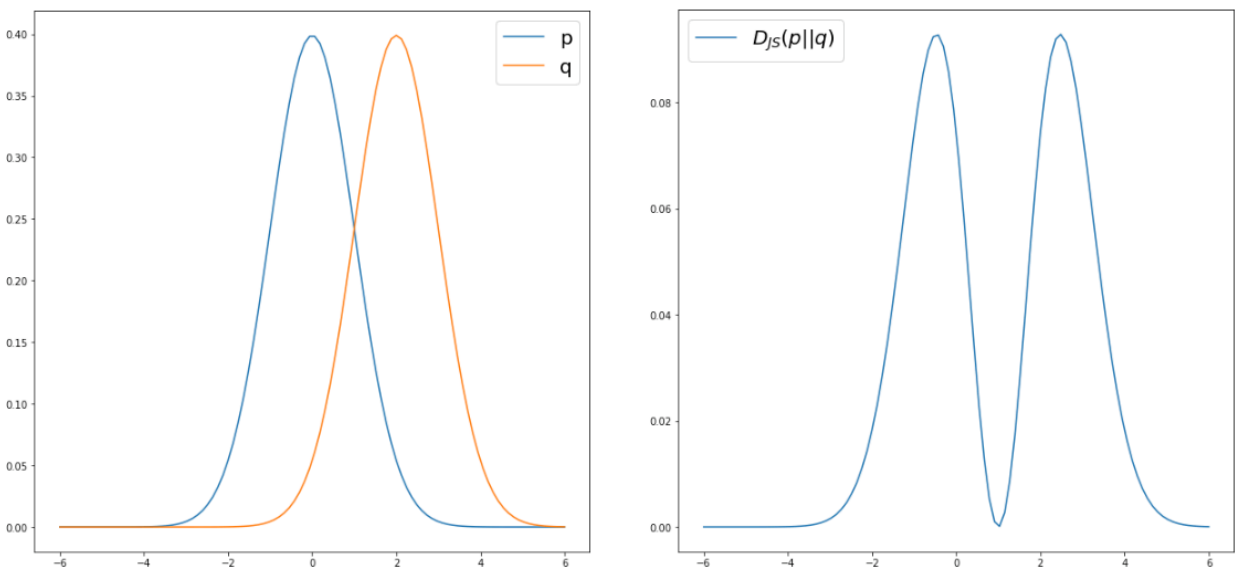


Figure 7: JS Divergence before computing the integral

Gans Problems

- Non-convergence.
- Diminished Gradient
- Unstable Gradients.
- Mode Collapse.

Non-convergence

Salimans et al. (2016) [2] provides an example of a zero-sum game which is not able to converge to the Nash equilibrium point. GANs do not guarantee convergence to Nash equilibrium.

Player 1 controls x and wants to minimize the cost function $c_1(x) = xy$. Player 2 controls y and minimizes the cost $c_2(y) = -xy = -c_1(x)$.

$\frac{\partial c_1}{\partial x} = y$; $\frac{\partial c_1}{\partial y} = x$, and we update x and y through gradient descent with a learning rate $\alpha = .1$, $x := x + \alpha y$ and $y := y + \alpha x$.

After a few iterations we can see that x, y divergence instead of converging to the Nash equilibrium point $x = y = 0$.

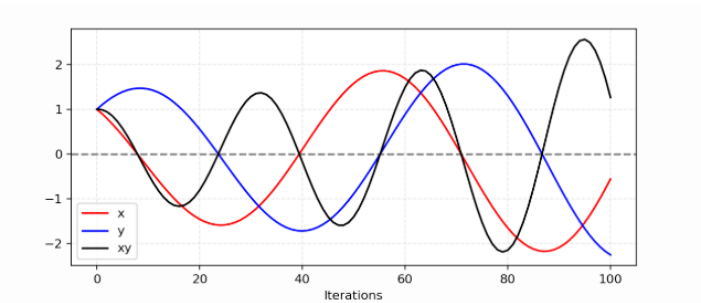


Figure 8: Simulation for x to minimize xy and y to minimize -xy.

Diminished Gradient

At the beginning of training it's easy for the discriminator to distinguish between real and fake images.

The discriminator saturates and provides really small gradients to generator:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[log\ D(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

$$\nabla_{G(z)}(log(1-D(G(z)))); \quad D(x) = \sigma(x;\theta)$$

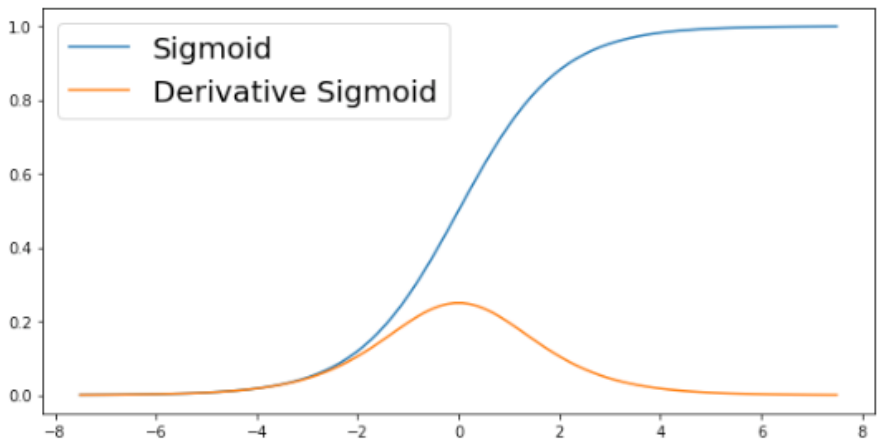


Figure 9: Sigmoid function and gradient.

Diminished Gradient

Arjovsky et al. (2017)[3] show how freezing the Generator and training the GAN for optimal Discriminator fails to provide gradients to the Generator.

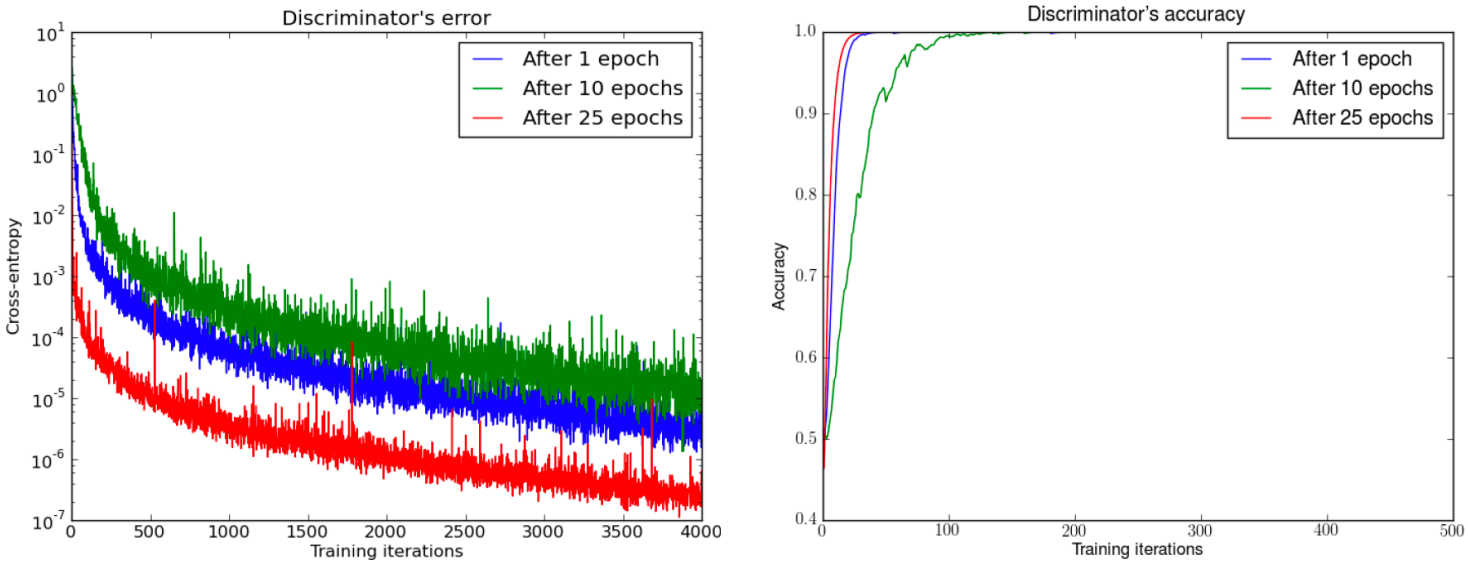


Figure 10: Vanishing gradients on discriminator.[3]

Unstable Gradients

To avoid the diminished gradient due to Discriminator $D(x)$ saturation, new cost function for Generator $G(z)$.

- No longer a zero-sum game.
- Comes with a cost: Unstable gradients as we keep training.

$$L^{(D)} = -\mathbb{E}_{x \sim p_{data}(x)}[log\ D(x)] - \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$
$$L^{(G)} = -L^{(D)} = \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$
$$\downarrow$$
$$L^{(D)} = -\mathbb{E}_{x \sim p_{data}(x)}[log\ D(x)] - \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$
$$L^{(G)} = -\mathbb{E}_{z \sim p_z(z)}[log(D(G(z)))] = \max_{G(z)} \mathbb{E}_{z \sim p_z(z)}[log(D(G(z)))]$$

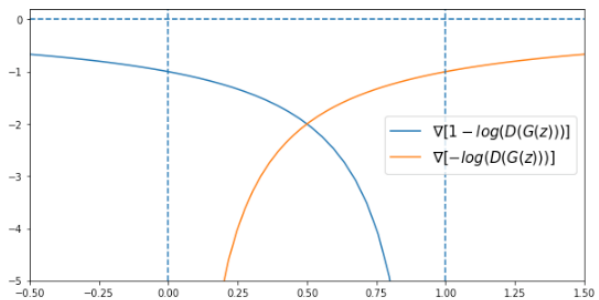


Figure 11: Gradient comparison on alternative cost function for the Generator.

Unstable Gradients

Arjovsky et al. (2017) [3] show in the same paper that this new cost function provides unstable and noisy gradients to the generator as we keep training the GAN.

$$L^{(G)} = -\mathbb{E}_{z \sim p_z(z)}[log(D(G(z)))] = \max_{G(z)} \mathbb{E}_{z \sim p_z(z)}[log(D(G(z)))]$$

New Generator cost function allows the Generator to learn faster in the first iterations but it becomes more unstable as we keep training the GAN.

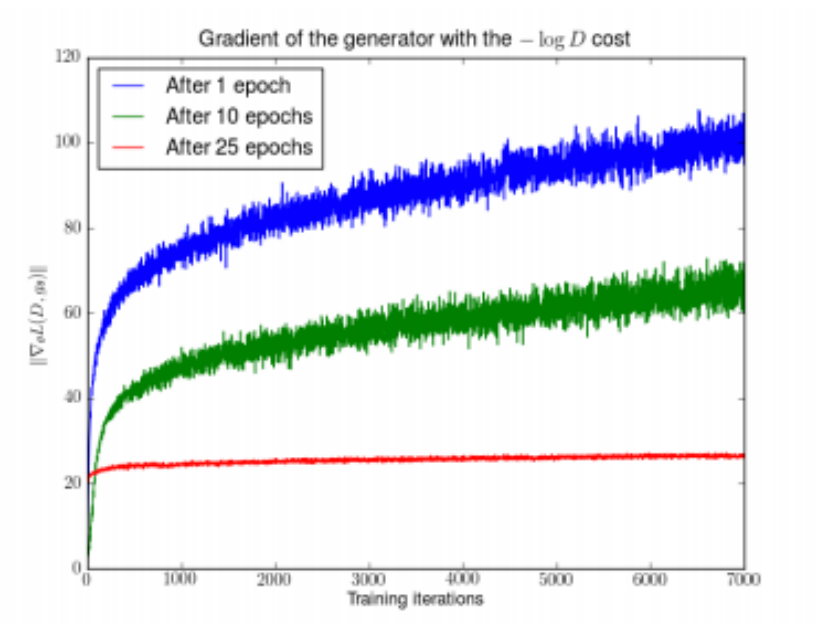


Figure 12: Gradient noise to generator through training.

Mode Collapse

Situation in which our generator only produces a few different sample, ”modes”.

From Arjovsky et a. (2017): [3]

$$\mathbb{E}_{z\sim p_z(z)}[-\nabla_\theta log(D^*(G(z)))] = \nabla_\theta[D_{KL}(p_g(x)||p_{data}(x)) - 2D_{JS}(p_{data}(x)||p_g(x))]$$

- The negative $D_{JS}(p_{data}(x)||p_g(x))$ at the same time pushes for $p_{data}(x)$, and $p_g(x)$ to be different, which seems counter intuitive.
- Inverted KL Divergence $D_{KL}(p_g(x)||p_{data}(x))$ will penalize non realistic images, with no cost to mode dropping.
- This cost pushed then have realistic images but encouraging mode dropping.

Mode Collapse

Situation in which our generator only produces a few different sample, ”modes”.

From Metz et a. (2017): [4]

- Let’s consider the extreme case in which the Generator is trained over a fixed Discriminator.
- It is possible for the Generator to learn a function that fools the Discriminator without taking any information from the latent vector z.
- This is possible to happen at the beginning of training, and the Generator will be pushed to learn rotating in the single modes in the ’mouse-cat’ game.

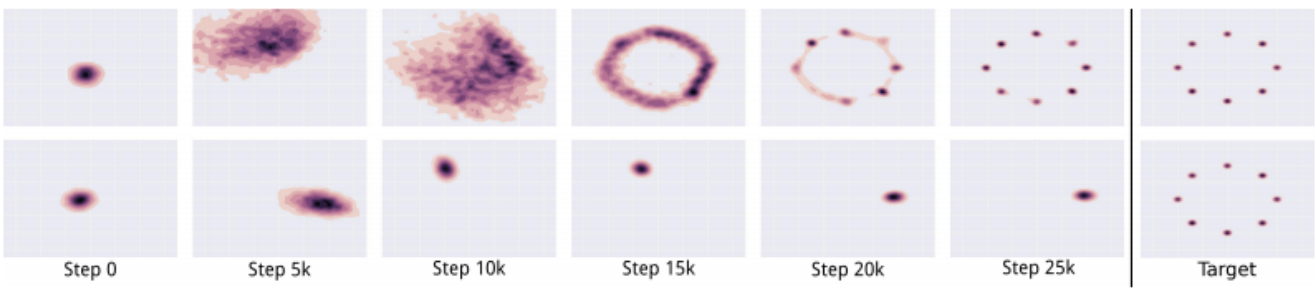


Figure 13: Comparison between a ’healthy’ GAN, and mode collapse.

Mode Collapse

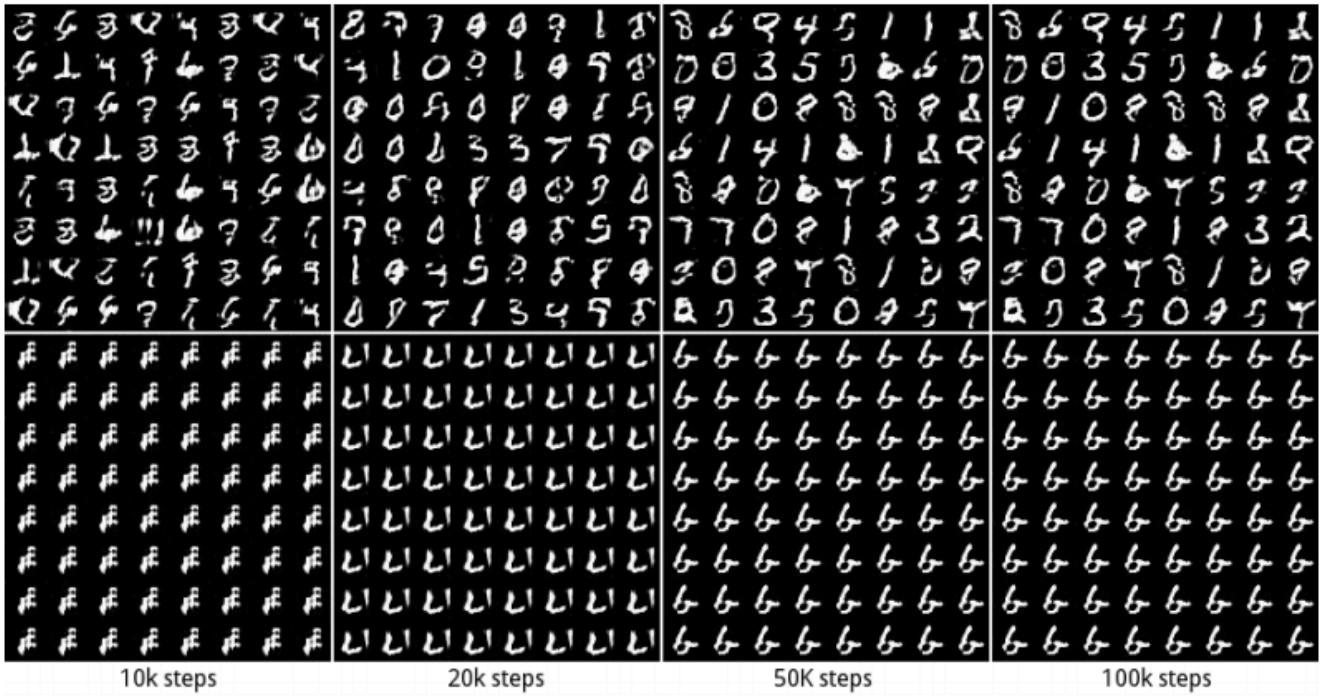


Figure 14: Comparison between a 'healthy' GAN, and mode collapse.

Evaluation

- Inception Score.
- Fréchet Inception Distance.

Evaluation

Inception Score:

- Proposed by Salimans et al. (2016) [2].
- We use a trained Inception Network v3.
- Image Quality $P(Y/X)$: Ability to identify whats in the image, a lower entropy defines a more confident model.
- Image Diversity $P(Y)$: How are the different classes distributed? Higher entropy means a more randomly distributed classes.

$$P(y) = \int_x p(y/x) p_g(x) dx$$

- It doesn't measure in relation to the real distribution. The more X, and Y have in common the larger score, missing diversity or detail per se.

$$IS(X) = exp\ \mathbb{E}_{x \in Pg}[\ D_{KL}(P(Y/X)\ ||\ P(Y))\ ] \tag{1}$$

$$IS(X) = exp[\ H(Y) - H(Y|X)\ ] = exp[\ I(X;Y)\ ]$$

Fréchet Inception Distance:

- Proposed by Heusel et al. (2017) [5].
- We use Inception Network v3 to extract features from intermediate layer, Inception-v3 pool3 layer.
- Model the distribution of those features as multivariate Gaussian with mean μ and co-variance Σ.
- The more your generated features distribution looks like the real, the lower the distance.
- **FID overcomes a limitation from IS**: it is able to measure diversity within a class, one image per class is enough to get perfect score in IS.

$$FID(X, G) = ||\mu_X - \mu_G||_2^2 + \mathrm{Tr}(\Sigma_X + \Sigma_G - 2(\Sigma_X \Sigma_G)^{1/2}) \tag{2}$$

$$X_r \sim \mathcal{N}(\mu_X, \Sigma_X) \equiv \text{Real Images}$$

$$X_G \sim \mathcal{N}(\mu_G, \Sigma_G) \equiv \text{Generated Images}$$

Fréchet Inception Distance vs Inception Score:

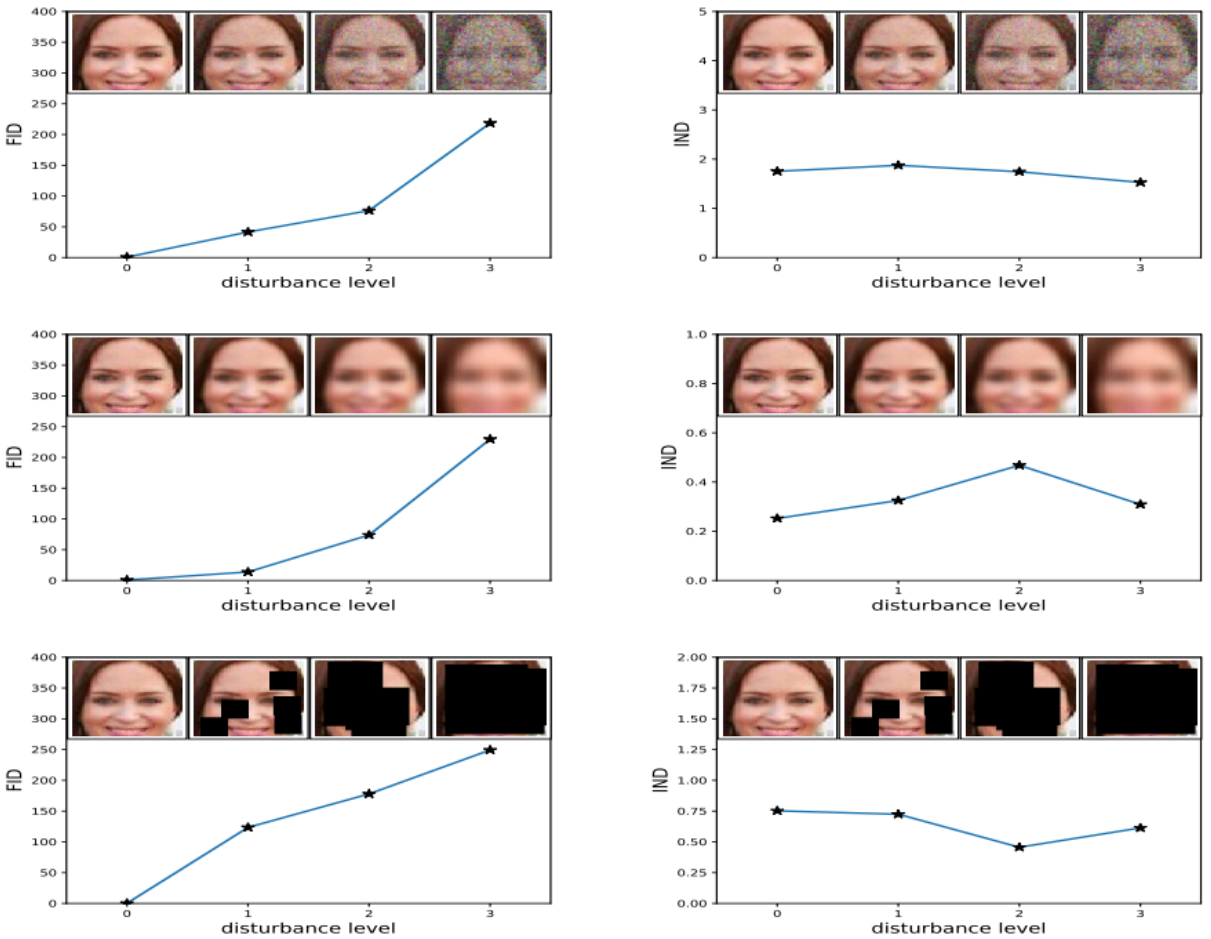


Figure 15: FID vs IS scores.

Fréchet Inception Distance vs Inception Score:

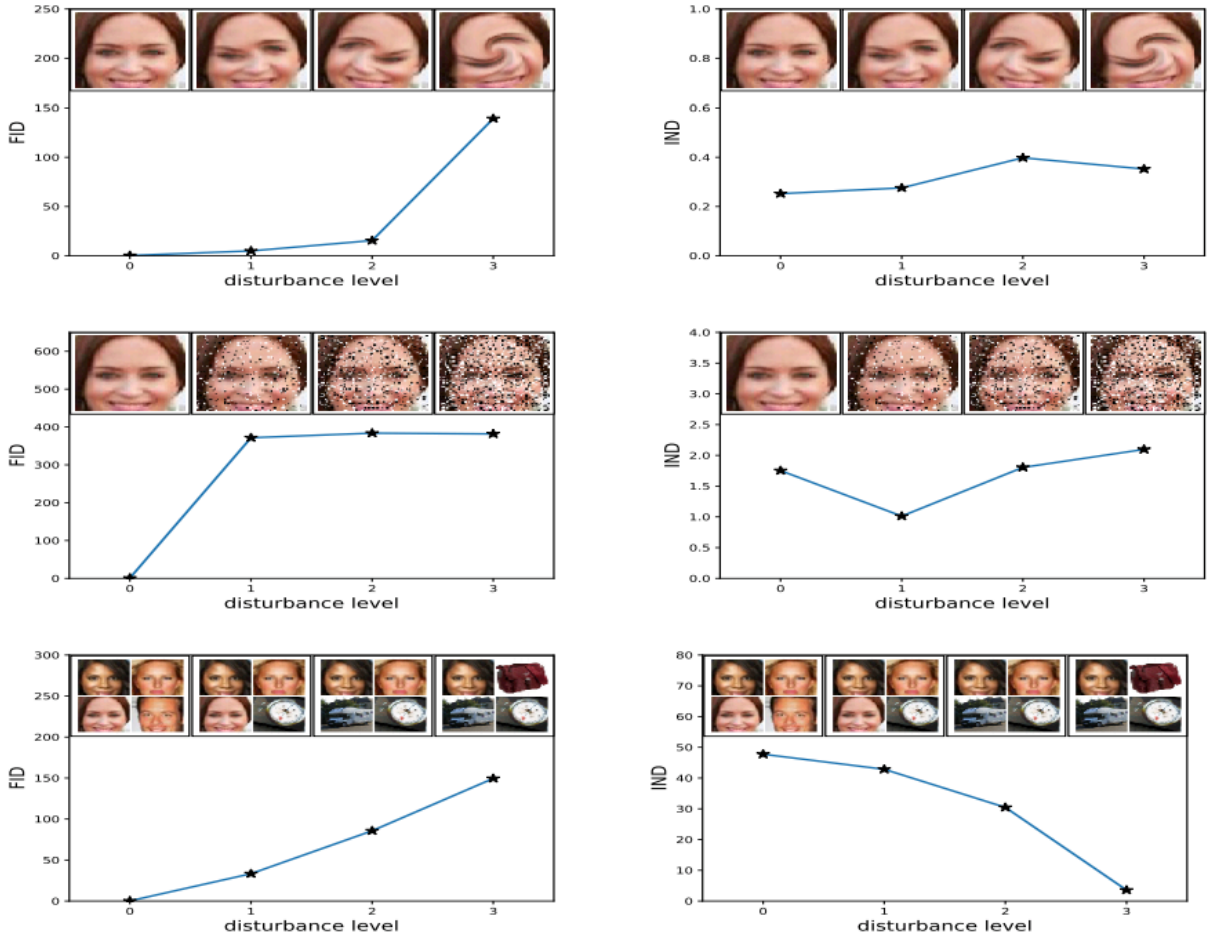


Figure 16: FID vs IS scores.

Models

- DCGAN.
- WGAN/WGAN-GP.
- SNGAN.
- ProGAN.
- Conditional GANs.
- SAGAN.
- BigGAN.

Deep Covolutional GANs: Radford et al. (2015) [6]

- Introduce convolutional layers.
- Replace any pooling layer with strided convolutional (Discriminator), and transposed convolutional layer (Generator) for down-sampling and up-sampling the height and width dimensions.
- Batch Normalization in both Generator (not in last layer), and Discriminator (not in first).
- Fully connected layers for deeper architecture.
- Activations:
  - ReLU - Generator.
  - LeakyReLU - Discriminator. (As a measure to help propagate the gradient to the Generator).
  - Tanh - Output of Generator.

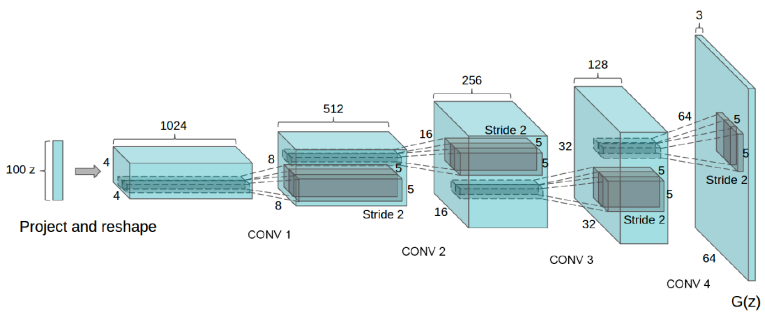


Figure 17: DCGAN Generator Arquitecture

Generator has interesting vector arithmetic properties.

Figure 18: Linear Interpolation in latent space z results in gradual change in images.

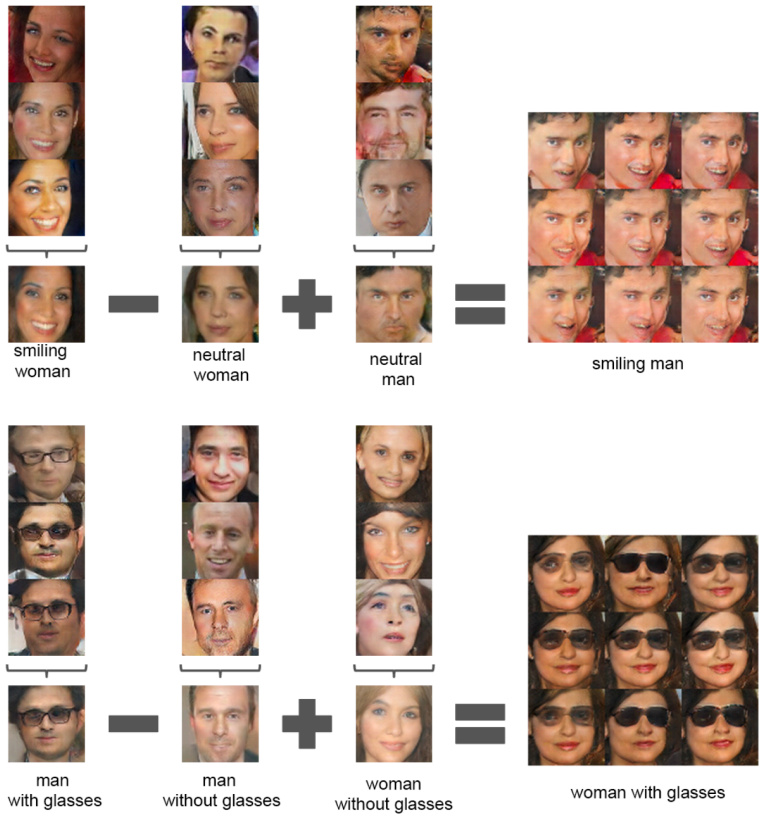


Figure 19: Arithmetic operations in latent vectors Z, result in meaningful changes in images.

DCGAN

Still DCGAN has its problems:

- Very sensitive to hyper-parameter changes. Moving not so far from the values provided in the paper, the GAN is likely to collapse.
- Changes in the architecture would make the GAN not to converge.
- We are still using the same cost function, all instability is still there.

Wasserstein GANs: Arjovky et al. (2017) [7]

- New cost function.
- Instead of working with KL Divergence or JS Divergence, uses the Earth Mover (Wasserstein distance from Optimal Transport.
- How can we efficiently move one probability distribution to another one?

Earth Mover/Wasserstein distance:

- Minimum cost of transporting mass in converting one distribution q into another distribution p.
- Example representation:
  - Moving boxes from left distribution(1,2,3) into right distribution(7,8,9,10)
  - Based on this, we may have different transportation plans.

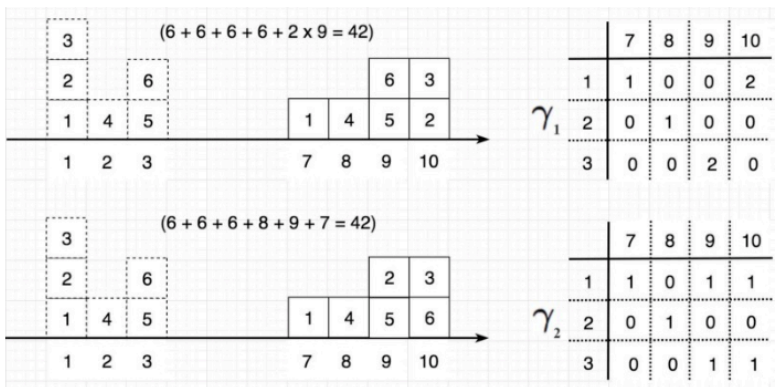


Figure 20: Distribution representation and transportation plans.

Earth Mover/Wasserstein distance:

- The marginal of the transportation plans must be equal to the original and the destination distributions.

$$\sum_x \gamma(x, y) = p(y)$$

$$\sum_y \gamma(x, y) = p(x)$$

Π

y

x	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0

γ_1

x	7	8	9	10
1	1	0	0	2
2	0	0	1	0
3	0	1	1	0

γ_2

Figure 21: Transportation plans.

Earth Mover/Wasserstein distance:

$$W(P_r, P_g) = \inf_{\gamma \in \pi(P_r, P_g)} \int_x \int_y \gamma(x, y)||x - y|| = \inf_{\gamma \in \prod(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}||x - y||$$

- $\prod(P_r, P_g) \equiv$ Set of all possible joint probabilities of P_r, and P_g.
- $\gamma \in \prod(P_r, P_g) \equiv$ One joint distribution that describes one transportation plan.
- Wasserstein distance finds the infimum transportation plan to transform distribution P_r to P_g.

Why Wasserstein distance instead of KL, JS Divergence?
Example from Arjovky et al. (2017) [7]:

$$\forall (x, y) \in P, x = 0 \text{ and } y \sim U(0, 1)$$
$$\forall (x, y) \in Q, x = \theta, 0 \leq \theta \leq 1 \text{ and } y \sim U(0, 1)$$

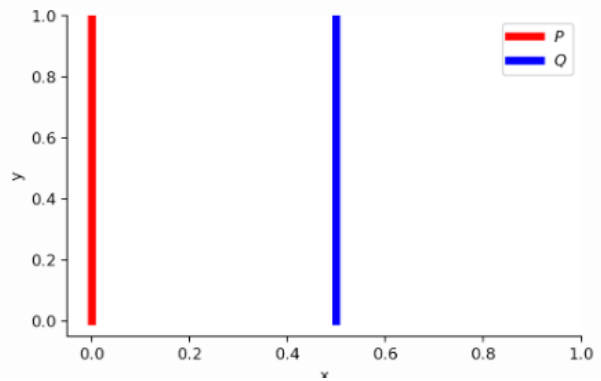


Figure 22: P and Q distributions.

Why Wasserstein distance instead of KL, JS Divergence?

- When $\theta \neq 0$:

$$D_{KL}(P/Q) = \sum_{x=0, y\sim U(0,1)} 1log(\frac{1}{0}) = +\infty$$

$$D_{KL}(Q/P) = \sum_{x=\theta, y\sim U(0,1)} 1log(\frac{1}{0}) = +\infty$$

$$D_{JS}(Q/P) = \frac{1}{2}(\sum_{x=0, y\sim U(0,1)} 1log(\frac{1}{1/2} + \sum_{x=\theta, y\sim U(0,1)} 1log(\frac{1}{1/2})) = +2$$

$$W(Q, P) = |\theta|$$

- When $\theta = 0$:

$$D_{KL}(P/Q) = D_{KL}(Q/P) = D_{JS}(Q/P) = 0$$

$$W(Q, P) = |\theta| = 0$$

- Wasserstein distance provides a smoother measure, specially when the two distributions are disjoint.

Earth Mover/Wasserstein distance:

$$W(P_r, P_g) = \inf_{\gamma \in \pi(P_r, P_g)} \int_x \int_y \gamma(x, y)||x - y|| = \inf_{\gamma \in \prod(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}||x - y||$$

- $P_r, and \prod(P_r, P_g)$ are intractable, we need to find a new way.
- We can use the Karontovich-Rubenstein Duality to transform this distance.
- Long story, short: adds an additional optimization over a function $f : x \to k \in \mathfrak{R}$, to assume that we are working with the optimal plan. To ensure that $||x - y|| - (f(x) - f(y)) = 0$, we need to enforce that $f(x)$ is 1-Lipschitz.

$$W(P_r, P_g) = \sup_{||f||_{L\leq 1}} \mathbb{E}_{s\sim p_r}[f(s)] - \mathbb{E}_{t\sim p_g}[f(t)]$$

Lipschitz Continuity A real-valued function $f : \mathfrak{R} \to \mathfrak{R}$ is called K-Lipschitz continuous if there exists a real constant $K \geq 0$ such that, for all $x_1, x_2 \in \mathfrak{R}$

$$||f(x_1) - f(x_2)|| \leq K||x_1 - x_2||$$

Limits how fast a function can change, the slope of the function has to be bounded for any point x. In this case the slope cannot be larger than 1.

Cost Function:

$$\min_G \max_{D \in 1-Lip} \mathbb{E}_{x \sim p_r}[D(x)] - \mathbb{E}_{z \sim p_g(z)}[D(G(z))]$$

Take aways:

- More stable training, we can train Discriminator in a ratio 5:1.
- Provides linear gradients as discriminator doesn't saturate.
- Weight clipping as a way to enforce 1-Lipschitz continuity. This slows down learning and it's hard to find appropriate clipping value.

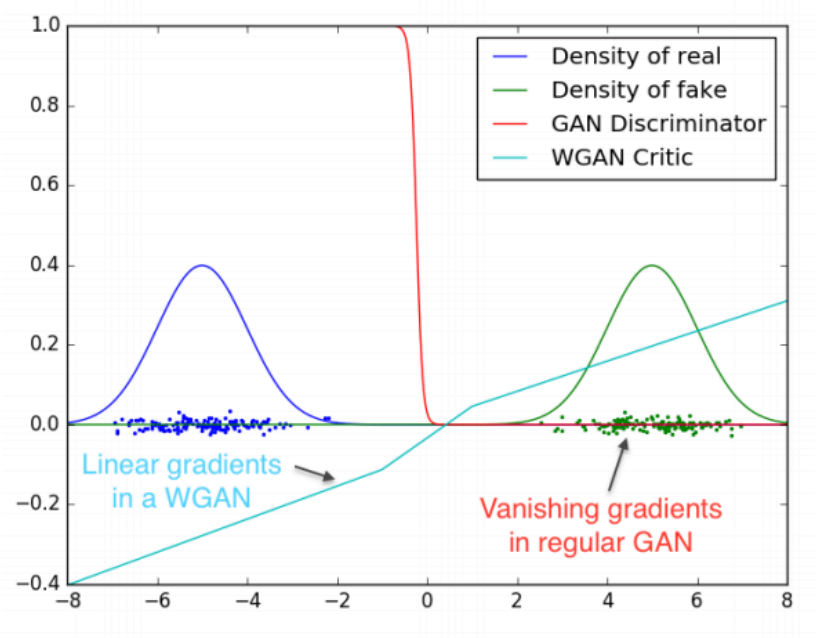


Figure 23: WGAN Gradients.

Wasserstein GAN - Gradient Penalty: Gulrajani et al. (2017) [8]

- Provides an efficient way of enforcing Lipschitness continuity.
- Shows robust training.
- Avoids Batch Normalization in the Critic/Discriminator. It creates correlation between samples in the batch, and impacts the performance of the Gradient penalty.
- Provides high quality images and becomes the standard model at the time.
- Gradient penalty slows down training.

$$\min_{G} \max_{D \in 1-Lip} \mathbb{E}_{x \sim p_r}[D(x)] - \mathbb{E}_{z \sim p_g(z)}[D(G(z))] + \lambda \mathbb{E}_{\hat{x} \sim P_{\hat{(x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2]$$

$$\hat{x} \leftarrow \epsilon x_r + (1 - \epsilon) x_g$$

Robust training.

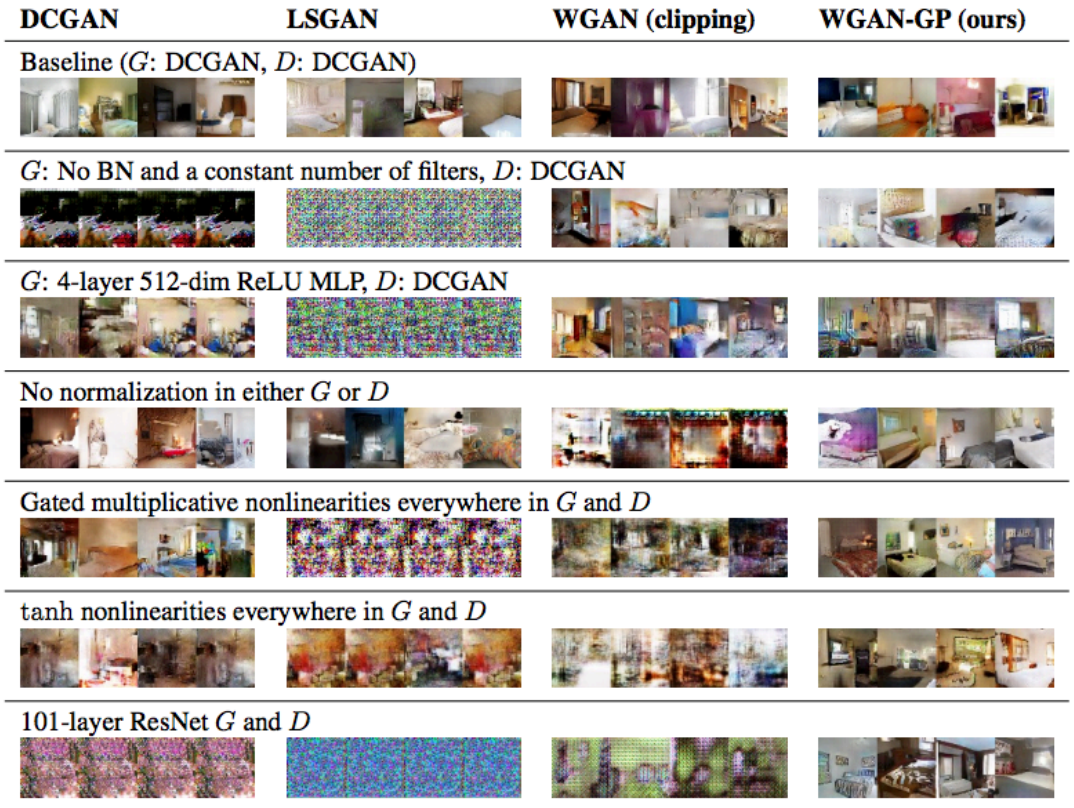


Figure 24: 128x128 Images for different models

High quality images.

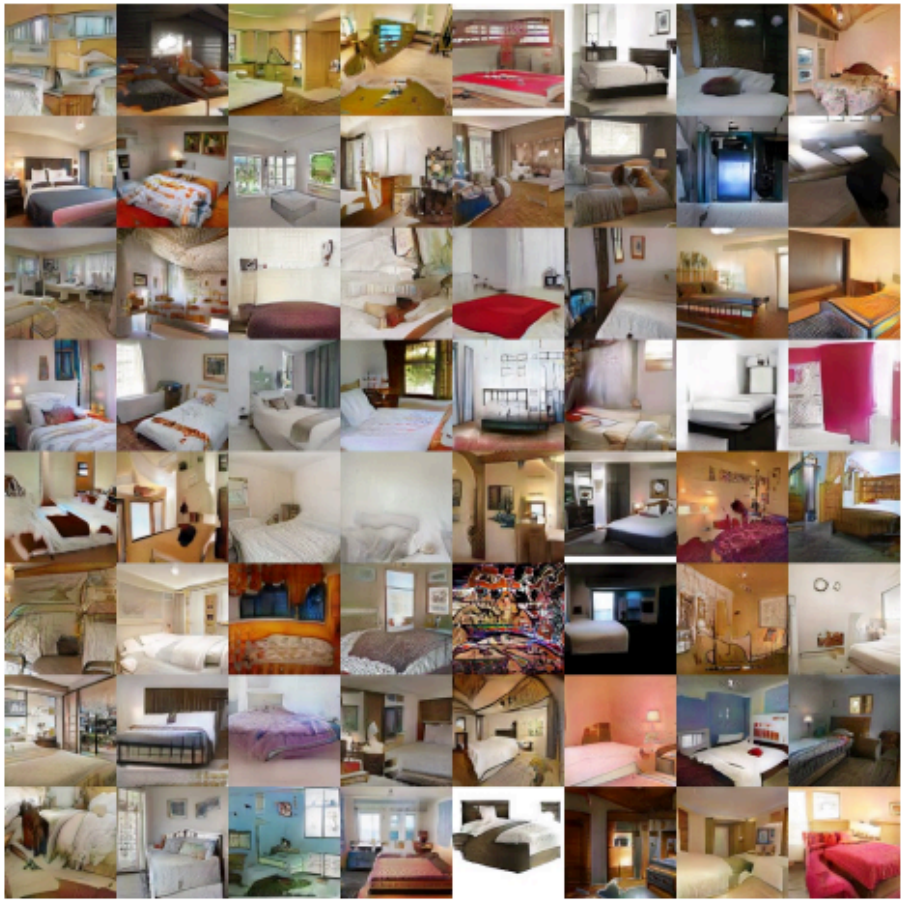


Figure 25: 128x128 LSUN generated images.png

ProGAN

ProGAN: Karras et al. (2017) [9]

- First GAN to achieve high resolutions 1024x1024.
- Training is done through increasing resolution and architecture of the GAN.
- When a layer is added, there's a transitional fading period.

ProGAN

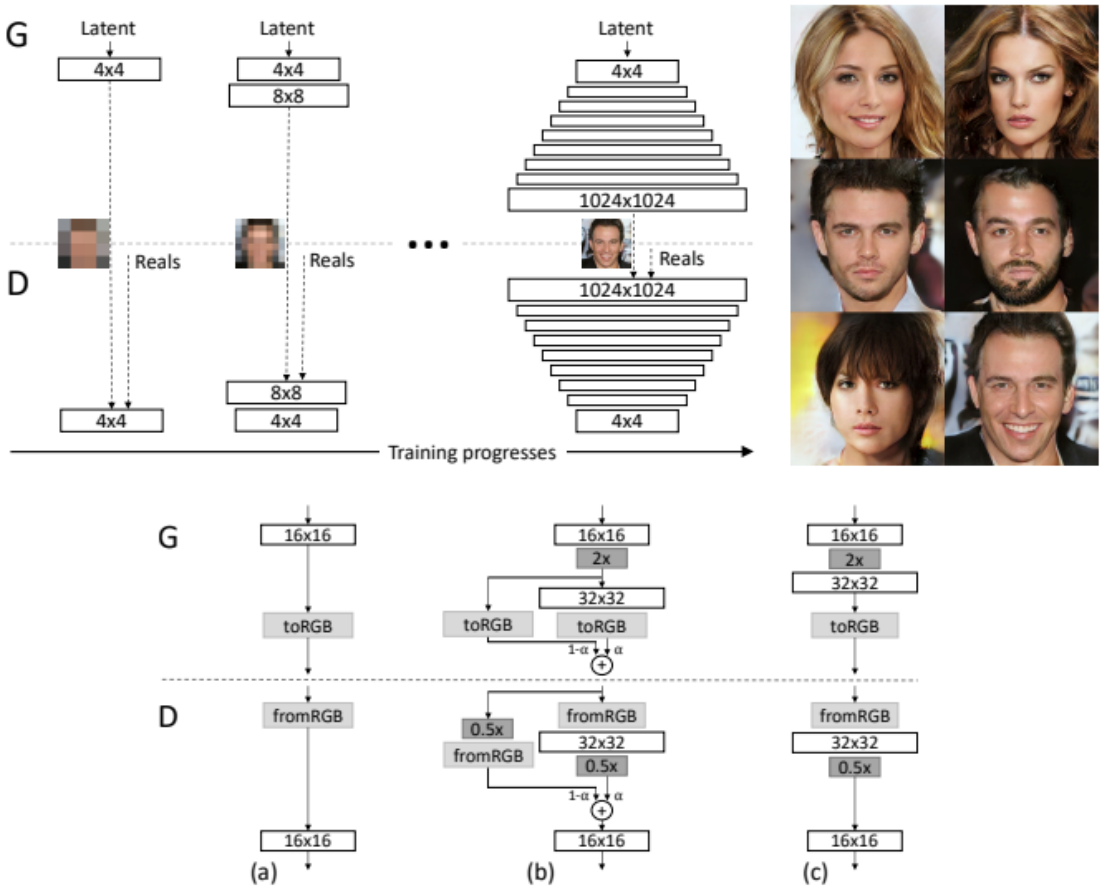


Figure 26: Architecture through training

CelebA Generated Images

Figure 27: CelebA Generated Images

LSUN Generated Images

Figure 28: LSUN Generated Images.

SNGAN

Spectral Normalization for Generative Adversarial Networks: Miyato et al. (2018) [10]

- Spectral Normalization $\rightarrow$
  - Lipschitz continuity on discriminator layers, this bounds the gradient in the network.
  - Stabilizes training having smoother updates.
  - It has an impact on the diversity of the images, high diversity of images even within a class.
  - Applied on Discriminator.
- Conditional Batch Normalization.
- Hinge/Wasserstein loss.

Lipschitz Continuity A real-valued function $f : \mathfrak{R} \to \mathfrak{R}$ is called K-Lipschitz continuous if there exists a real constant $K \geq 0$ such that, for all $x_1, x_2 \in \mathfrak{R}$

$$||f(x_1) - f(x_2)|| \leq K||x_1 - x_2||$$

Limits how fast a function can change, the slope of the function has to be bounded for any point x:

$$f : \mathfrak{R}^n \to \mathfrak{R}^m$$

$$K \geq sup_{x \in \mathfrak{R}^n} ||J_f(x)||$$

$$||f||_{Lip} = ||J_f(x)||_2 = sup\{||J_f(x)x|| : x \in \mathfrak{R}^n; ||x|| = 1\} = \sigma_{max}(f)$$

Lipschitz constant of Linear Map

$$A : \mathfrak{R}^n \rightarrow \mathfrak{R}^m$$

Since A is linear, we can set out point of reference x_2 to zero. If A is K-Lipschitz at zero, it is K-Lipschitz everywhere.

$$||Ax|| \leq K||x|| \ \rightarrow < Ax, Ax > \leq K^2 < x, x > \rightarrow < (A^T A - K^2)x, x > \leq 0$$

Expressing x in the orthonormal basis of the eigenvectors of $A^T A \rightarrow \sum_i x_i v_i$

$$< (A^T A - K^2) \sum_i x_i v_i, \sum_j x_j v_j > = \sum_i \sum_j x_i x_j < v_i, v_j > = \sum_i (\lambda_i - K^2) x_i^2 \leq 0$$

$$K \geq \sqrt{\lambda_i} = \sigma_i(A)$$

$A^T A$ is positive semi definite, λ_i eigenvalues are non negative.
So we can use the largest singular value to σ_i to ensure Lipschitz continuity.

Lipschitz on composition of functions

$$f : \mathfrak{R}^n \to \mathfrak{R}^m, g : \mathfrak{R}^m \to \mathfrak{R}^l$$

$$\nabla(g \circ f)(x) = \nabla(g(f(x)))\nabla f(x)$$

$$\sigma(\nabla f(x)) = sup_v ||\nabla f(x)v||$$

$$\sigma(\nabla(g \circ f)(x)) = sup_v ||\nabla(g(f(x)))\nabla f(x)v||$$

We can bound the composition by:

$$sup_v ||\nabla(g(f(x)))\nabla f(x)v|| \leq sup_u ||\nabla(g(f(x)))u||\ sup_v ||\nabla f(x)v||$$

$$||g \circ f||_{Lip} \leq ||g||_{Lip} ||f||_{Lip}$$

We can define the Discriminator as:

$$f(x;\theta) = W^{L+1} a_L(W^L(a_{L-1}(W^{L-1}(...\ a_1(W^1 x)\ ...))));$$

$$D(x,\theta) = A(f(x,\theta))$$

$$||f||_{Lip} = ||W^{L+1}||_{Lip} ||a_L||_{Lip} ||W^L||_{Lip} \ ... \ ||W^1||_{Lip} = \prod_{i=1}^{L+1} \sigma(W^L); ||a_l||_{Lip} = 1$$

- Spectral Normalization applied to all weights in Discriminator
- Approximate largest singular value of the weight matrix through Power Iteration $\sigma_0(W)$
- Normalize $\hat{W} = W/\sigma_0(W) \rightarrow \sigma_0(\hat{W}) = 1$.
- Update weights on Stochastic Gradient Descent.

Algorithm 1 SGD with spectral normalization

- Initialize $\tilde{\boldsymbol{u}}_l \in \mathcal{R}^{d_l}$ for $l = 1, \ldots, L$ with a random vector (sampled from isotropic distribution).
- For each update and each layer l:
  1. Apply power iteration method to a unnormalized weight W^l:

$$\tilde{\boldsymbol{v}}_l \leftarrow (W^l)^{\mathrm{T}} \tilde{\boldsymbol{u}}_l / \|(W^l)^{\mathrm{T}} \tilde{\boldsymbol{u}}_l\|_2 \tag{20}$$

$$\tilde{\boldsymbol{u}}_l \leftarrow W^l \tilde{\boldsymbol{v}}_l / \|W^l \tilde{\boldsymbol{v}}_l\|_2 \tag{21}$$

  2. Calculate $\bar{W}_{\mathrm{SN}}$ with the spectral norm:

$$\bar{W}^l_{\mathrm{SN}}(W^l) = W^l/\sigma(W^l), \text{ where } \sigma(W^l) = \tilde{\boldsymbol{u}}_l^{\mathrm{T}} W^l \tilde{\boldsymbol{v}}_l \tag{22}$$

  3. Update W^l with SGD on mini-batch dataset $\mathcal{D}_M$ with a learning rate α:

$$W^l \leftarrow W^l - \alpha \nabla_{W^l} \ell(\bar{W}^l_{\mathrm{SN}}(W^l), \mathcal{D}_M) \tag{23}$$

Figure 29: Spectral Normalization Algorithm.

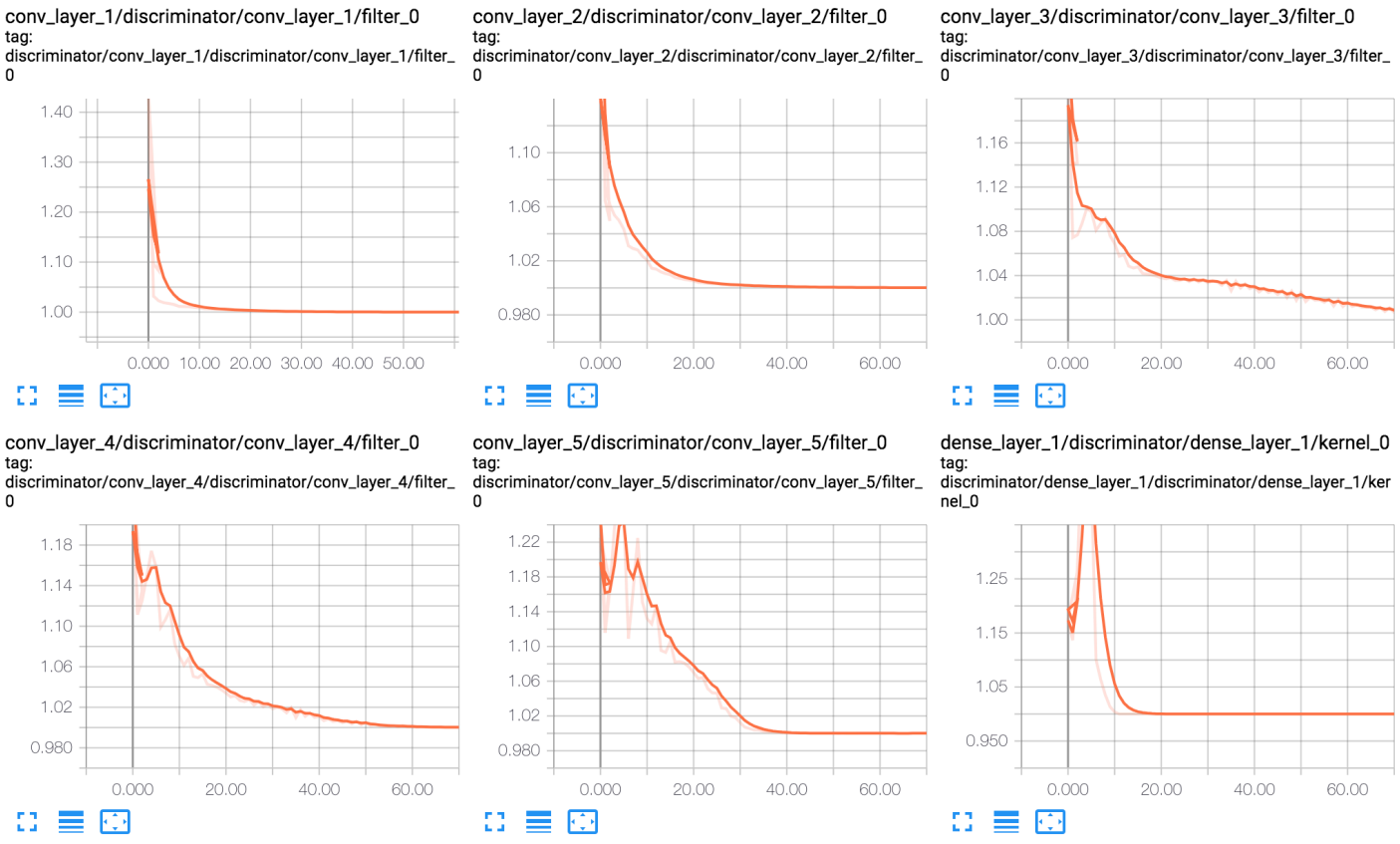


Figure 30: Max. singular Values for Conv. Layer Weights over training.

SNGAN

Hinge loss.

$$V_D(G, D) = \max \ \mathbb{E}_{x \sim p_r(x)}[min(0, -1 + D(x))] + \mathbb{E}_{z \sim p(z)}[min(0, -1 - D(G(z)))]$$

$$V_G(G, D) = \min \ -\mathbb{E}_{z \sim p(z)}[D(G(z))]$$

- Optimizing these is equivalent to minimizing the reversed KL Divergence $D_{KL}(p_g||p_r)$
- We ensure that we cover all 'modes'.
- Interesting that images still have good quality.

- Removes Batch Normalization from Discriminator.
- Introduces Conditional Batch Normalization from Style Transfer Networks, Dumoulin et al. (2017) [11].
- Mean and variance is calculated over the Batch, Height, and Width.
- γ and β are calculated using a MLP whose input is the label.

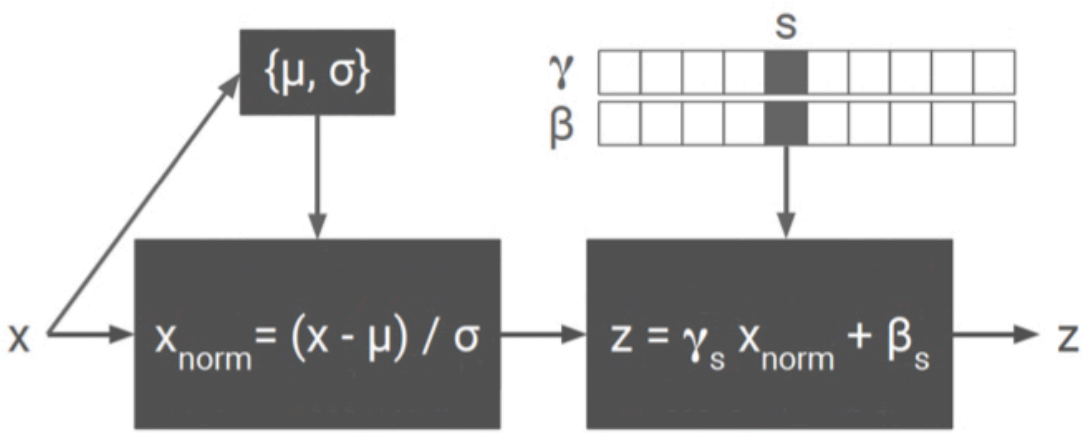


Figure 31: Conditional Batch Normalization.

SNGAN

- Achieves high score on IS, and low on FID.
- High diversity of samples within a class.
- First conditional GAN to work on full ImageNet, 14 Million Images and 20K classes.

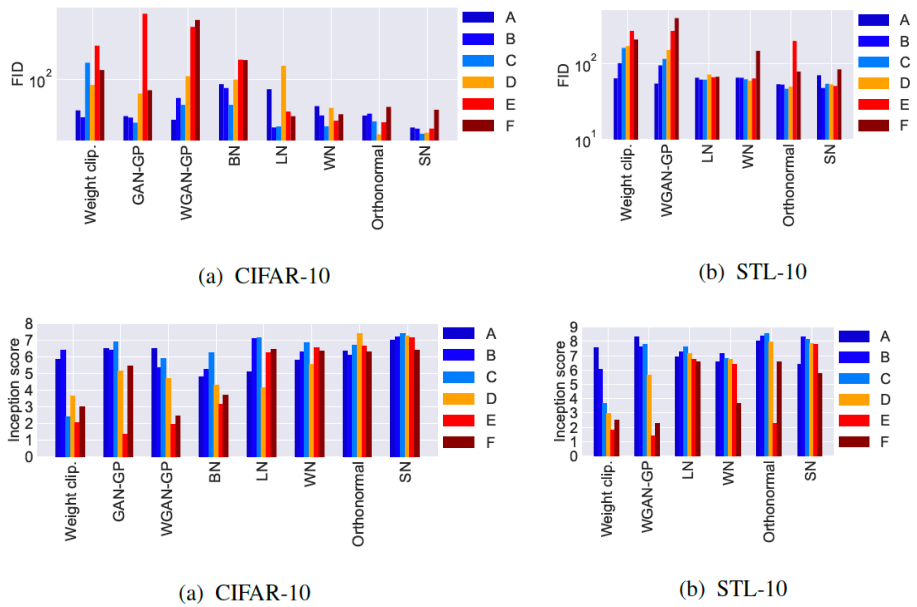


Figure 32: FID and IS scores.

Figure 33: Diversity of images in one class.

Conditional GANs

Generator: Label is fed directly $P_g(x/y)$.
Discriminator:

- **Conditional GAN:** Label is fed to the Discriminator. Mirza et al. (2014) [12]
- **Auxiliary Classifier GAN:** Discriminator predicts the label, Adversarial loss + Classification loss. Odena et al. (2014) [13]
- **Projection Discriminator:** Discriminator as log likelihood ratio between true distribution and Generator distribution. Miyato et al. (2014) [14]

$$f(x, y; \theta) = log(\frac{q(y/x)}{p(y/x)}) + log(\frac{q(x)}{p(x)}) = r(y/x) + r(x) = y^T V \phi(x; \theta_\phi) + \Gamma(\phi(x; \Psi_\phi); \theta_\Psi)$$

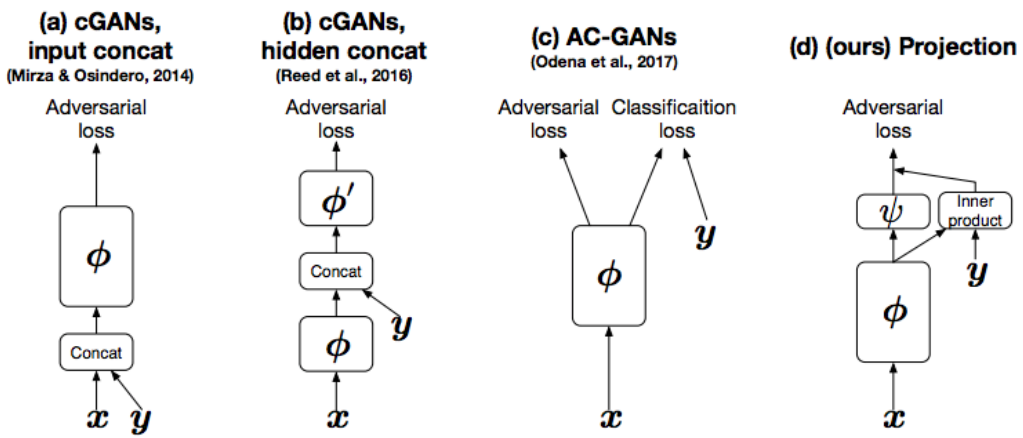


Figure 34: Conditional GAN models.

SAGAN

Self-Attention Generative Adversarial Networks: Zhang et al. (2018) [15]

- Introduced an attention layer in both Discriminator and Generator.
- Spectral Normalization in both Discriminator and Generator.
- Hinge loss.
- Conditional SAGAN:
  - Conditional Batch Normalization for Generator.
  - Projection discriminator for D.
- Improvement from SNGAN, produces ImageNet samples without conditioning.

Attention Mechanism

- First we find the relationship between each pixel in the input image.
- With the softmax we find the probability for each pixel and the importance to the relationship with the others.
- Finally to synthesize O, we use the previous output to define the relationship between different regions of the image.

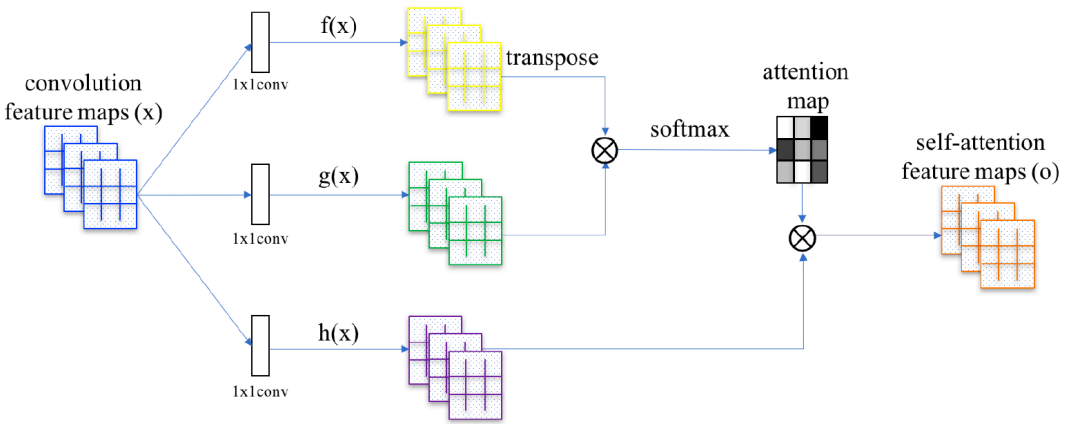


Figure 35: Attention Mechanism.

Attention Mechanism

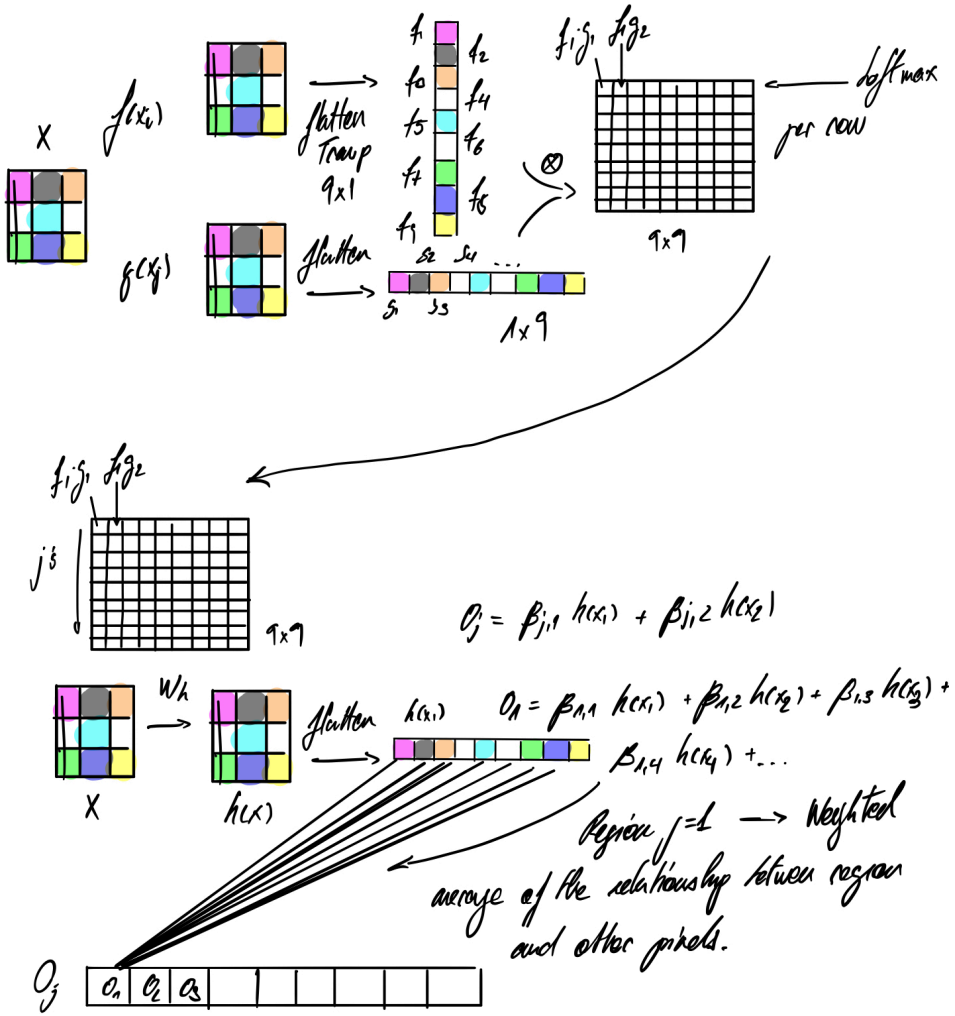


Figure 36: Attention Mechanism.

Image example of the attention mechanism

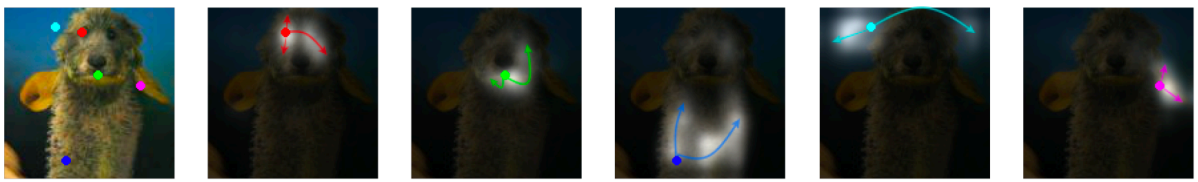


Figure 37: Attention Mechanism.

SAGAN improved performance over SNGAN.

Model	Inception Score	FID
AC-GAN [31]	28.5	/
SNGAN-projection [17]	36.8	27.62*
SAGAN	**52.52**	**18.65**

Figure 38: Score comparison.

Image diversity within a class.

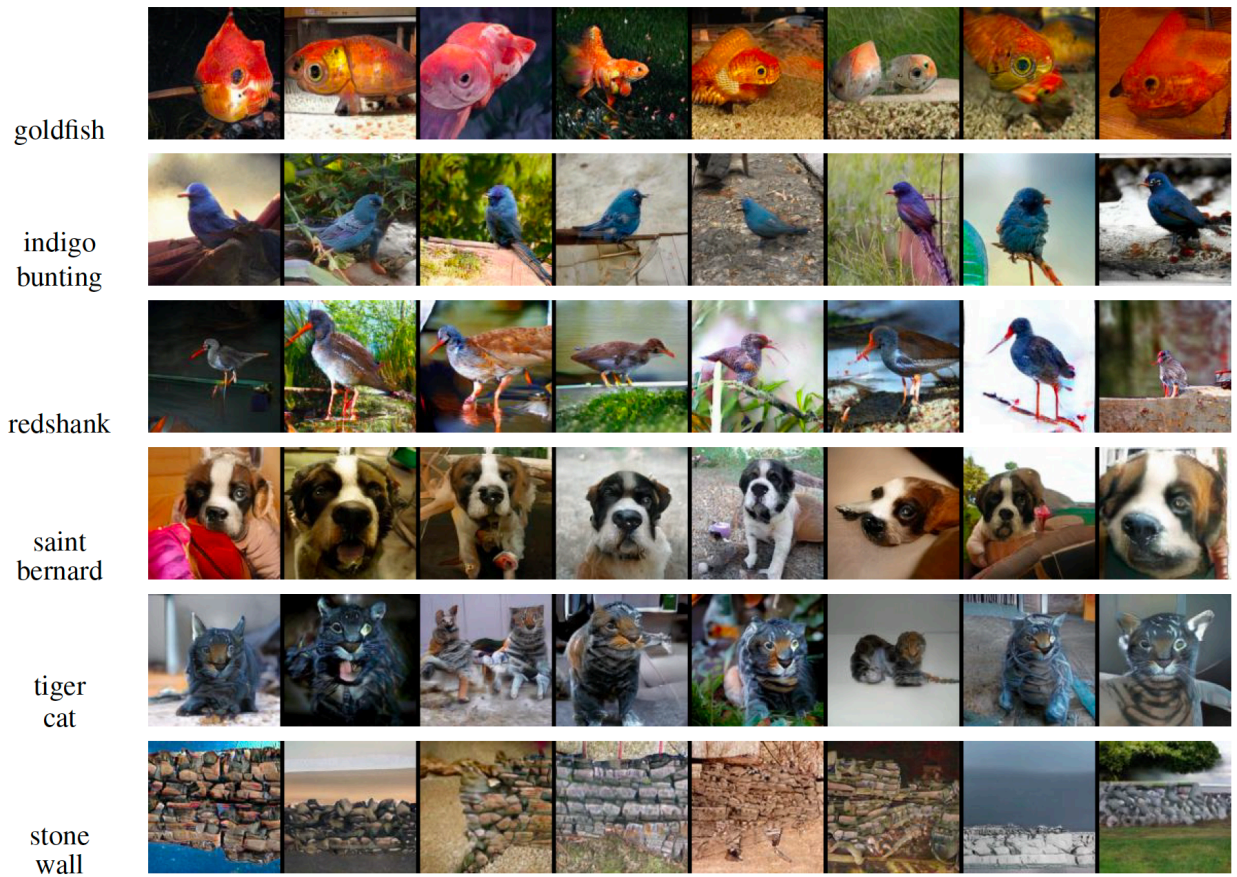


Figure 39: Image Diversity.

BigGAN

BigGAN: Brock et al. (2018) [16]

- Achieves high resolution in images 512x512.
- Introduces the truncation trick.
- Studies instability on Discriminator and Generator.
- Baseline Model:
  - Self-Attention GAN.
  - Hinge loss.
  - Orthogonal initialization and regularization.
  - Class Conditional Batch Normalization on Generator.
  - Class Projection discriminator on Discriminator.
  - Hierarchical Latent Space: Latent vector z is fed into multiple layers of the Generator.

Orthogonal Initialization:

- Orthogonal matrices have all $|\lambda_i| = 1, |W| = 1$, prevents exploding or vanishing values through the network. (First introduced for RNNs, Saxe et al. [17]).
- Orthogonal rows for the network layer to have output uncorrelated with each other, they react to different features.

Orthogonal Initialization:

- Penalizes the non-orthogonality of the weight matrices.
- W orthogonal $\rightarrow W^{-1} = W^T$

$$L_{ortho} = \sum_W (||WW^T - I||)$$

Truncation trick:

- Best results achieves using different latent distribution in training and testing.
- Re-sample values outside the truncation value.
- This translates in a trade-off between quality and diversity in images. (a)
- IS doesn't penalty the lack of diversity within a class, this score increases unconditionally with the trick.
- FID initially increases but sharply drops.

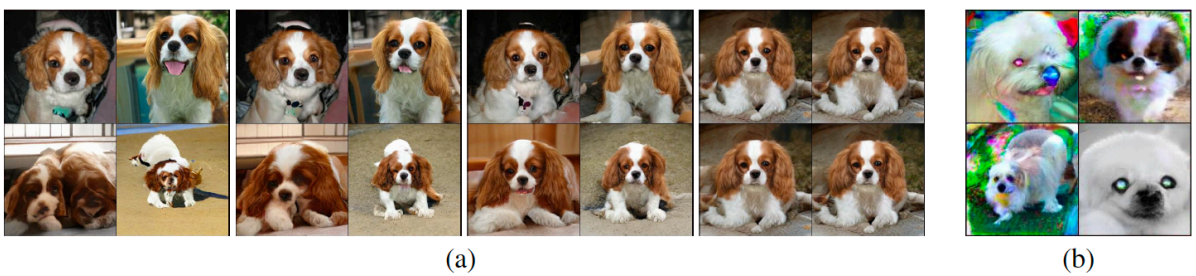


Figure 40: Truncation trick.

Truncation trick:

- Larger models don't work well with the truncation trick, to enforce it orthogonal regulation is applied. (b)
- Best variant of regularization: 16% of the models worked initially $\rightarrow$ 60%

$$R_\beta(W) = \beta||WW^T \odot (1 - I)||_F^2$$

Figure 41: Truncation trick.

Characterization of instability:

- 3 top singular values of each W are the most informative.
- First dense layer of both Generator and Discriminator are ill-behaved.
- On collapse singular values spike, this occurs when the Discriminator overfits the training data.

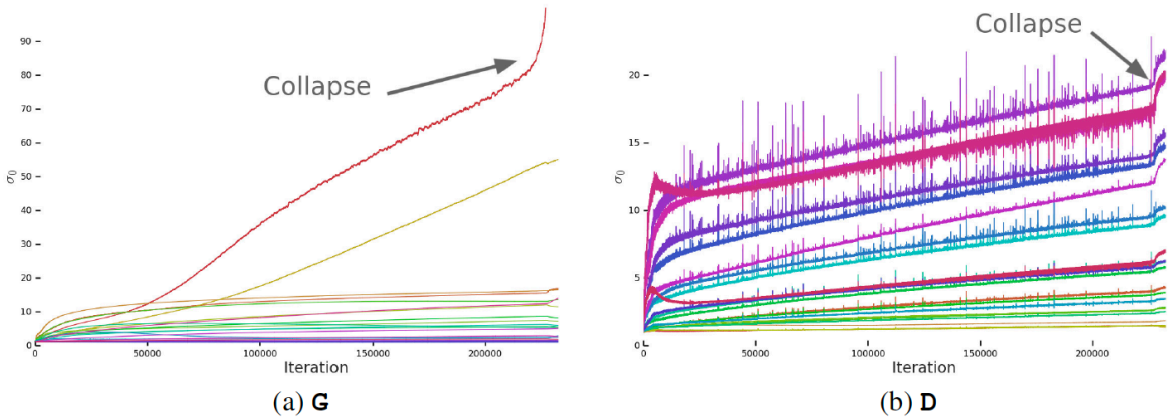


Figure 42: Generator and Discriminator singular values.

Architecture:

- Class Conditional Batch Normalization on Generator.
- Class Projection discriminator on Discriminator.
- Hierarchical Latent Space: Latent vector z is fed into multiple layers of the Generator.

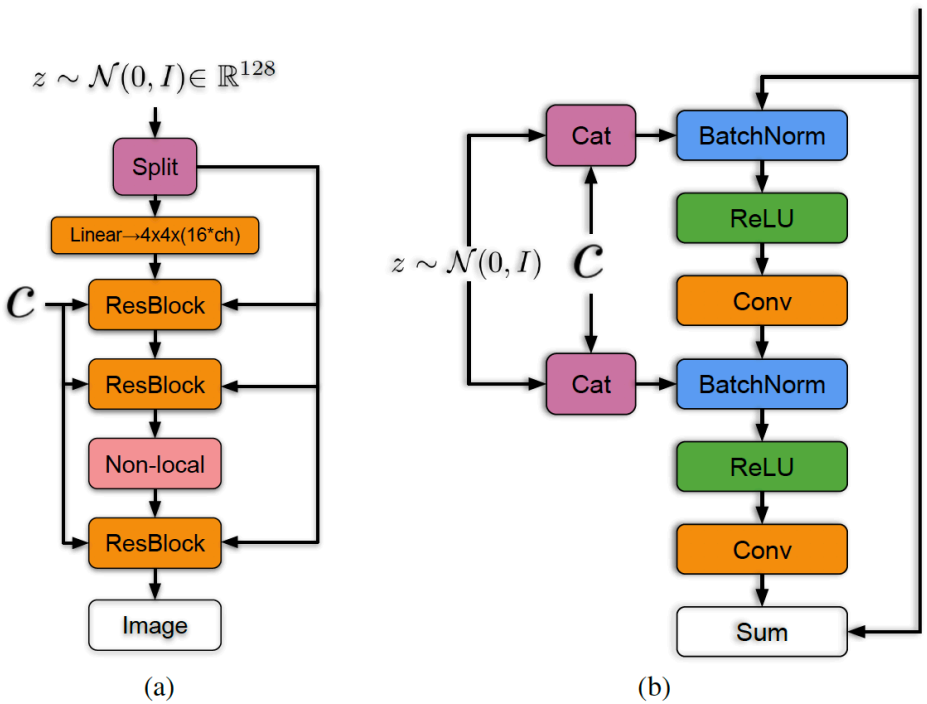


Figure 43: BigGAN Architecture sample.

Take aways:

- Increasing your batch size improves IS/FID.
- Increasing the number of channels improves IS/FID.
- Orthogonal regularization and Truncation trick.
- BigGAN collapses when Discriminator overfits the training data.

Batch	Ch.	Param (M)	Shared	Hier.	Ortho.	Itr $\times 10^3$	FID	IS
256	64	81.5	SA-GAN Baseline			1000	18.65	52.52
512	64	81.5	✗	✗	✗	1000	15.30	58.77(±1.18)
1024	64	81.5	✗	✗	✗	1000	14.88	63.03(±1.42)
2048	64	81.5	✗	✗	✗	732	12.39	76.85(±3.83)
2048	96	173.5	✗	✗	✗	295(±18)	9.54(±0.62)	92.98(±4.27)
2048	96	160.6	✓	✗	✗	185(±11)	9.18(±0.13)	94.94(±1.32)
2048	96	158.3	✓	✓	✗	152(±7)	8.73(±0.45)	98.76(±2.84)
2048	96	158.3	✓	✓	✓	165(±13)	8.51(±0.32)	99.31(±2.10)
2048	64	71.3	✓	✓	✓	371(±7)	10.48(±0.10)	86.90(±0.61)

Figure 44: BigGAN Score with different configurations.

Not Included But Still Important

- LSGAN
- InfoGAN
- StackedGAN
- CycleGAN
- EBGAN / BEGAN
- DRAGAN
- RSGAN / RASGAN
- StyleGAN

DRAFT - BACKUP

Relativistic Standard GANs: Alexia Jolicouer-Martineau. (2018) [18]

- In the optimal point of Standard GAN, Discriminator randomly assigns real or fake differentiation $D^*(x) = 0.5$
- This doesn't align well with Discriminator $D(x)$ labeling 0 or 1 for fake and real images. $\rightarrow$ Discriminator saturation.
- Instead, Discriminator measures the probability that the real images is more realistic than the generated data.

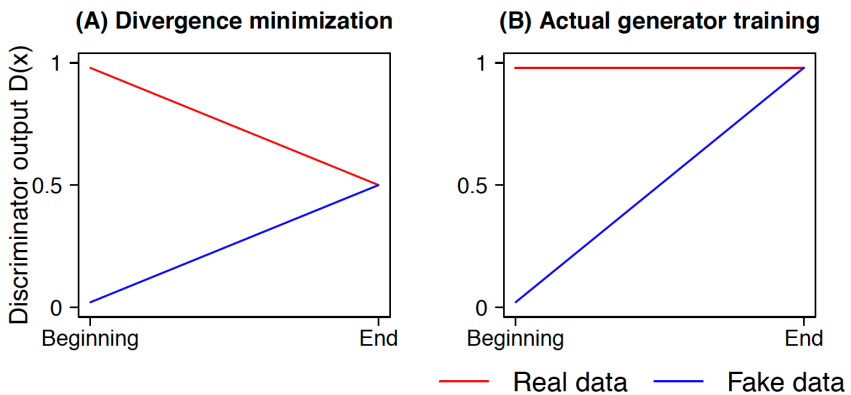


Figure 45: Divergence minimization.

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio.
Generative adversarial nets.
In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[2] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen.
Improved techniques for training gans.
In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234, 2016.

[3] Martín Arjovsky and Léon Bottou.
Towards principled methods for training generative adversarial networks.
CoRR, abs/1701.04862, 2017.

[4] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein.
Unrolled generative adversarial networks.
CoRR, abs/1611.02163, 2016.

[5] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter.
Gans trained by a two time-scale update rule converge to a local nash equilibrium.
In Guyon et al. [19], pages 6629–6640.

[6] Alec Radford, Luke Metz, and Soumith Chintala.
Unsupervised representation learning with deep convolutional generative adversarial networks.
CoRR, abs/1511.06434, 2015.

[7] Martín Arjovsky, Soumith Chintala, and Léon Bottou.
Wasserstein GAN.
CoRR, abs/1701.07875, 2017.

[8] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville.
Improved training of wasserstein gans.
In Guyon et al. [19], pages 5769–5779.

[9] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen.
Progressive growing of gans for improved quality, stability, and variation.
CoRR, abs/1710.10196, 2017.

[10] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida.
Spectral normalization for generative adversarial networks.
CoRR, abs/1802.05957, 2018.

[11] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur.
A learned representation for artistic style.
CoRR, abs/1610.07629, 2016.

[12] Mehdi Mirza and Simon Osindero.
Conditional generative adversarial nets.
CoRR, abs/1411.1784, 2014.

[13] Augustus Odena, Christopher Olah, and Jonathon Shlens.
Conditional image synthesis with auxiliary classifier gans.
In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2642–2651. PMLR, 2017.

[14] Takeru Miyato and Masanori Koyama.
cgans with projection discriminator.
CoRR, abs/1802.05637, 2018.

[15] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena.
Self-attention generative adversarial networks.
CoRR, abs/1805.08318, 2018.

[16] Andrew Brock, Jeff Donahue, and Karen Simonyan.
Large scale GAN training for high fidelity natural image synthesis.
CoRR, abs/1809.11096, 2018.

[17] Andrew M. Saxe, James L. McClelland, and Surya Ganguli.
Exact solutions to the nonlinear dynamics of learning in deep linear neural networks.
CoRR, abs/1312.6120, 2013.

[18] Alexia Jolicoeur-Martineau.
The relativistic discriminator: a key element missing from standard GAN.
CoRR, abs/1807.00734, 2018.

[19] Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors.
Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, 2017.