

Generative Adversarial Networks

Victor Kitov

v.v.kitov@yandex.ru

Table of Contents

1 Simple GAN

- Model
- Practical recommendations

2 Deep convolutional GAN

3 Progressive GAN

4 Wasserstein GAN

Simple GAN

Model

1 Simple GAN

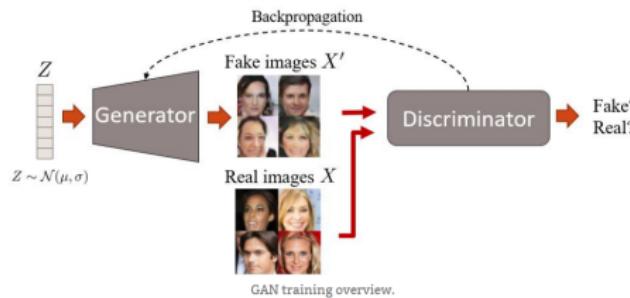
- Model
- Practical recommendations

Density estimation

- Consider i.i.d. sample x_1, \dots, x_N , $x_i \sim p(x)$.
- We want to sample $\hat{x}_{N+1}, \hat{x}_{N+2}, \dots$ from similar distribution.
- Statistical approach:
 - May take $q_\theta(x) \approx p(x)$ and sample from $q_\theta(x)$.
 - hard to estimate in high dimensional space
- Generative adversarial network (GAN):
 - replace hard problem (density estimation in high dim. space) with easy problem-binary classification
 - train discriminator, discriminating x from \hat{x}
 - fixed discriminator acts as trained loss function for improving generator $\hat{x} = g(z), z \sim p(z)$.

Intuition of adversarial learning

Generative adversarial learning for images:



Analogy for bank and a money counterfeiter (having a spy in the bank).

- they compete, until money counterfeiter learns to make perfect money replicas!

Seminal paper on GAN¹

- Two networks:
 - generator $G(z) : Z \rightarrow X$
 - outputs generated object x
 - discriminator $D(x) : X \rightarrow [0, 1]$
 - probability that x **is real** rather than generated by G .

¹[Link to paper.](#)

Seminal paper on GAN¹

- Two networks:
 - generator $G(z) : Z \rightarrow X$
 - outputs generated object x
 - discriminator $D(x) : X \rightarrow [0, 1]$
 - probability that x is **real** rather than generated by G .
- Define
 - $p(x)$ - true data distribution (from training set)
 - $q(x)$ - generated data distribution $x \sim q(x) = G(z)$, $z \sim p(z)$.
 - $p(z)$ - standard distribution (Gaussian or other-spherical recommended).

¹[Link to paper.](#)

Game

- Probability that x is correctly classified by discriminator:

$$\begin{cases} D(x), & x \text{ is real} \\ 1 - D(x), & x \text{ is fake} \end{cases}$$

Game

- Probability that x is correctly classified by discriminator:

$$\begin{cases} D(x), & x \text{ is real} \\ 1 - D(x), & x \text{ is fake} \end{cases}$$

- Log-probability of correct classification by discriminator:

- given that $p(\text{real}) = p(\text{fake}) = \frac{1}{2}$

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

Game

- Probability that x is correctly classified by discriminator:

$$\begin{cases} D(x), & x \text{ is real} \\ 1 - D(x), & x \text{ is fake} \end{cases}$$

- Log-probability of correct classification by discriminator:

- given that $p(\text{real}) = p(\text{fake}) = \frac{1}{2}$

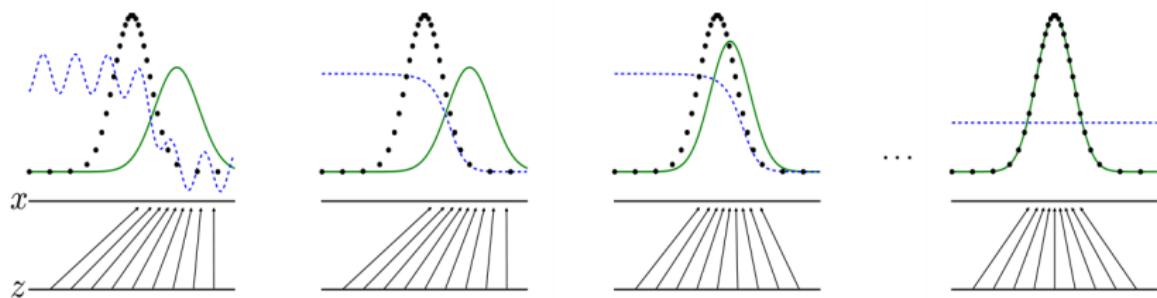
$$V(D, G) = \mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$

- D and G play two-player game with minimax function $V(G, D)$:

$$\min_G \max_D V(D, G)$$

Illustration of incremental learning

Incremental learning of D and G :



black dotted: $p(x)$ - density of true samples

green: $q(x)$ - density of fake samples

blue dashed: $D(x) = p(x \text{ is true} | x)$

Losses

D task (for fixed G):

$$\mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \max_D$$

G task (for fixed D):

$$\mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \min_G$$

Losses

D task (for fixed G):

$$\mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \max_D$$

G task (for fixed D):

$$\mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \min_G$$

- Thus $G(z)$ learns to sample realistic samples from $p(x)$.
- D and G should be trained synchronously.
 - if too much training of D , G will get zero-gradient, no feedback!
 - if too much training of G , it may converge to $\arg \max_x D(x)$.

Algorithm

for number of training iterations **do**

for k steps **do**

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

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

end for

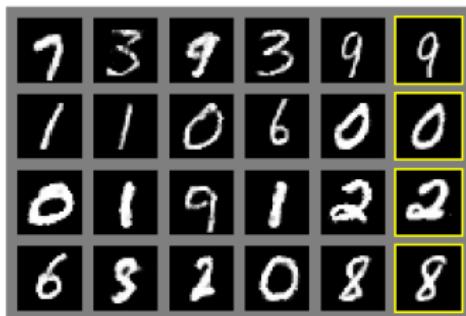
- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by descending its stochastic gradient:

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

end for

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

Generated images



a)



b)



Interpolation in latent space

Linear interpolation of objects in latent space:



Optimality theorem

Suppose D is flexible enough to take arbitrary values.

Theorem: For fixed G optimal discriminator is:

$$D^*(x|G) = \frac{p(x)}{p(x) + q(x)}$$

Theorem: given optimal D^* , optimal G^* should yield $q(x) = p(x)$.

1 Simple GAN

- Model
- Practical recommendations

Finding unstable saddle-point

D task (for fixed G):

$$\mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \max_D$$

G task (for fixed D):

$$\mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \min_G$$

Finding saddle-point $\min_G \max_D V(D, G)$ is very unstable.

- requires a lot of fine-tuning, see [link to recommendations](#).

Problem of smart discriminator

Problem:

- On early iterations G generates poor fakes.
- It becomes very easy for D to discriminate.
- So $D(G(z)) \approx 0 \Rightarrow \nabla_{\theta_G} \log(1 - D(G(z))) \approx 0$
 \Rightarrow no training in G .

Problem of smart discriminator: solutions

Improve G from another criterion:

$$\mathbb{E}_{z \sim p(z)} [\log(D(G(z)))] \rightarrow \max_G$$

- G is still tuned to fool the discriminator: $\uparrow D(G(z))$
- $\log(\cdot)$ exponentially amplifies small changes in $D(G(z))$ near zero.
- G gets non-degenerate feedback.
- D, G game stops being game with zero-sum.

Problem of smart discriminator: solutions

- occasionally flip labels when training D
- or use soft labels ($1 \rightarrow \text{uniform}[0.7, 1.2]$, $0 \rightarrow \text{uniform}[0, 0.3]$)
- or add noise to input of $D(\cdot)$

Intuition:

- discriminator solves classification in more noisy setting and its predictions become smoother, so ≈ 0

More recommendations

- Track failures early:
 - D loss goes to 0: failure mode
 - check norms of gradients: they should not be high
- If you have object labels, use them by training combined discriminator-classifier
 - weight sharing helps!

Mode collapse

- We proved that $\arg \max_G V(G, D^*(G)) = G^*$, yielding $q^*(x) = p(x)$
- Finding $D^*(G)$ for every change in G is impractical
- Practical task: improve G for fixed D : $\arg \max_G V(G, D)$
 - Can give mode collapse: $G(z) \equiv \text{const} \equiv \arg \max_x D(x)$
- Solutions to mode collapse
 - add stochasticity to G :
 - add Gaussian noise to inner layers (Zhao et. al. EBGAN).
 - add dropout in both train and test phase
 - add penalty for too close $G(z_1), G(z_2), \dots, G(z_K)$ where z_1, \dots, z_K are mini-batch initializations.

Possible initializations

- May initialize G with right half of VAE.
- May initialize first layers of D with classification net (e.g. VGG).

Table of Contents

- 1 Simple GAN
- 2 Deep convolutional GAN
- 3 Progressive GAN
- 4 Wasserstein GAN

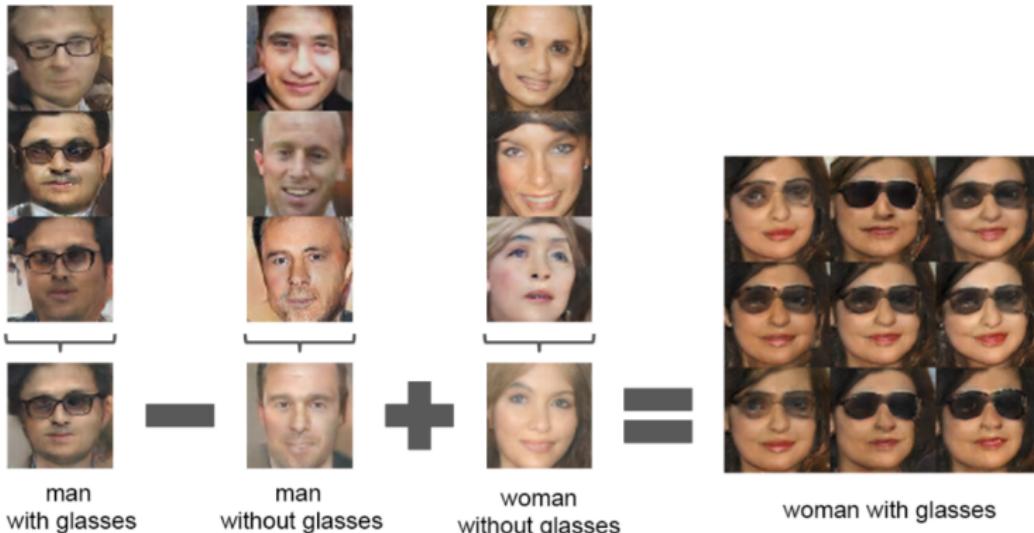
Deep convolutional GAN (DCGAN)²

Generates photorealistic 64x64 images (bedrooms in this case).



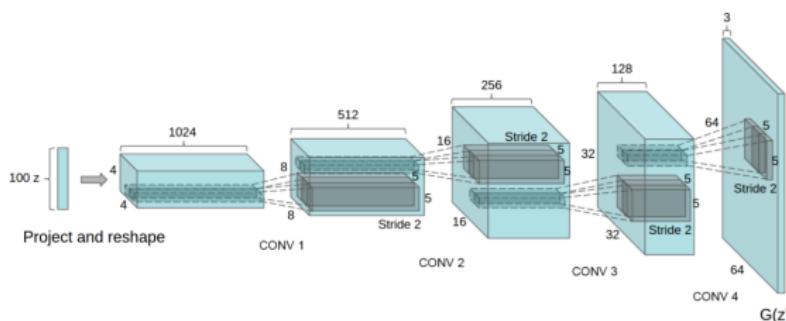
²<https://arxiv.org/pdf/1511.06434.pdf>

Latent space arithmetics



Generator

Uses fully-convolutional generator:



Architecture guidelines for stable DCGANs

- ➊ Replace any pooling layers with fractional-strided convolutions in G and strided convolutions in D .
- ➋ Use LeakyReLU activation in the discriminator for all layers.
- ➌ Use batchnorm in both the generator and the discriminator.
- ➍ Remove fully connected hidden layers for deeper architectures.
- ➎ Use ReLU activation in generator for all layers except for the output, which uses Tanh.

Intuition: 1 and 2 allow better propagation of gradients.

Table of Contents

- 1 Simple GAN
- 2 Deep convolutional GAN
- 3 Progressive GAN
- 4 Wasserstein GAN

Progressive GAN³

- Technique to generate high dimensional output.

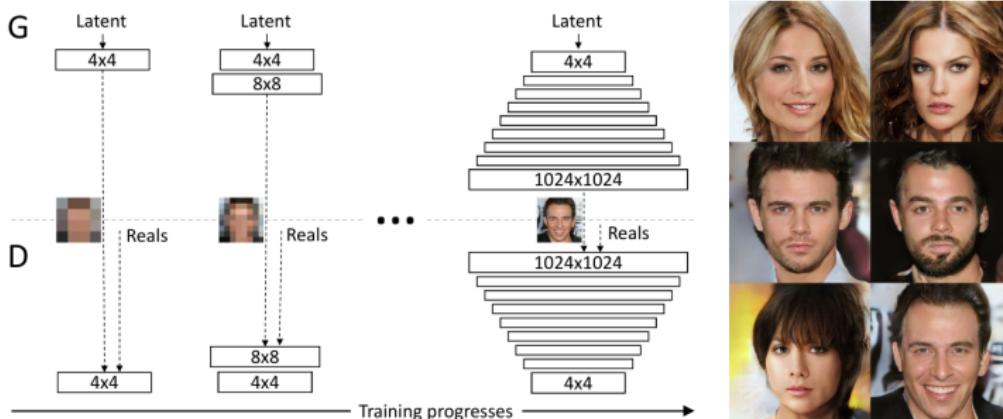
Generated photographs, using celebrities dataset



³Karras et al. 2017.

Training

Training is done sequentially. G, D trained on low-resolution layers, then higher resolution layers added:



Training

When new layer added, 2 branches appear (like in ResNet):

- rescale+identity with weight $1 - \alpha$
- rescale+conv+nonlinearity with weight α

During training α gradually changes from 0 to 1 (drop identity branch).

Table of Contents

- 1 Simple GAN
- 2 Deep convolutional GAN
- 3 Progressive GAN
- 4 Wasserstein GAN

Proposed approach⁴

Proposed approach for generator:

- ① $z \sim p(z)$ (e.g. $\sim \mathcal{N}(0, I)$)
- ② $x = g_\theta(z)$, so $x \sim q_\theta(x)$ [distribution of generator]

We want $p(x) \approx q_\theta(x)$, so we require $\rho(p(x), q_\theta(x)) \rightarrow \min_\theta$.

- $\rho(\cdot, \cdot)$ is distance between probability distributions

By using Earth mover distance we get non-trivial distances even when domains of $p(x)$ and $q(x)$ don't intersect!

⁴Arjovsky et al. Wasserstein GAN

Popular distances

Let $x \in \mathcal{X}$ and define Σ -all measurable subsets of \mathcal{X} .

- Total Variation (TV):

$$\delta(p, q) = \sup_{A \in \Sigma} |p(A) - q(A)|$$

- Kullback-Leibler (KL) divergence:

$$KL(p||q) = \int p(x) \ln \left(\frac{p(x)}{q(x)} \right) dx$$

- asymmetric
- infinite if exist regions where $q(x) = 0$ and $p(x) \neq 0$.

Popular distances

- Jensen-Shannon (JS) divergence:

$$JS(p, q) = KL(p||a) + KL(q||a)$$

where $a(x) := (p(x) + q(x)) / 2$.

- Earth-Mover (EM) distance or Wasserstein-1:

$$W(p, q) = \inf_{\gamma \in \Pi(p, q)} \mathbb{E}_{(x, y) \sim \gamma} [\|x - y\|]$$

where $\Pi(p, q)$ is a set of all densities $\in \mathbb{R}^{2D}$, having

- p - marginal distribution along $1 : D$ dimensions
- q - marginal distribution along $D + 1 : 2D$ dimensions.

Example

Setup-consider 2D r.v. belonging to 1D manifold:

- $z \sim U[0, 1]$
- p :=distribution of $(0, z) \in \mathbb{R}^2$.
- q_θ :=distribution of $(\theta, z) \in \mathbb{R}^2$ for some $\theta \in \mathbb{R}$.

Distance values $\rho(p, q_\theta)$:

$$W(p, q_\theta) = |\theta|$$

$$\delta(p, q_\theta) = \begin{cases} 1, & \theta \neq 0 \\ 0, & \theta = 0 \end{cases}$$

$$KL(p||q_\theta) = KL(q_\theta||p) = \begin{cases} +\infty, & \theta \neq 0 \\ 0, & \theta = 0 \end{cases}$$

$$JS(p, q_\theta) = \begin{cases} \ln 2, & \theta \neq 0 \\ 0, & \theta = 0 \end{cases}$$

If sequence $\theta_t \rightarrow 0$ then $\rho(p_r, p_{\theta_t}) \rightarrow 0$ only according to

Kantorovich-Rubinstein duality

- $f(x)$ is K-Lipshitz if $\forall x, x' \in \text{dom}(f) :$

$$f(x) - f(x') \leq K \|x - x'\|$$

- Intuitively K-Lipshitz means at any point x variation of $f(x + \Delta x)$ stays within a conus.

Wasserstein distance can be estimated with
Kantorovich-Rubinstein duality:

$$W(p, q_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p} [f(x)] - \mathbb{E}_{x \sim q_\theta} [f(x)]$$

where supremum is taken w.r.t all 1-Lipshitz functions.

- if $f(x)$ is taken among K-Lipshitz functions, duality will estimate $K \cdot W(p, q_\theta)$.
- since we want $W(p, q_\theta) \rightarrow \min$, multiplier K is not important.

Practical estimation of Wasserstein distance

- We may estimate Wasserstein distance with parametrized family of K-Lipschitz functions $\{f_w\}_{w \in W}$.
- f_w is taken as multi-layer perceptron with typical activations and constrained weights $W \in [-0.01, 0.01]^{\#\text{weights}}$ (to ensure K-Lipschitz property)
- For such function family:
 - ➊ $\max_{w \in W} \mathbb{E}_{x \sim p} [f_w(x)] - \mathbb{E}_{z \sim p(z)} [f_w(g_\theta(z))]$ - maximum is achieved.
 - ➋ $\nabla_\theta W(p, q_\theta) \propto -\mathbb{E}_{z \sim p(z)} [\nabla f(g_\theta(z))]$

Algorithm

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

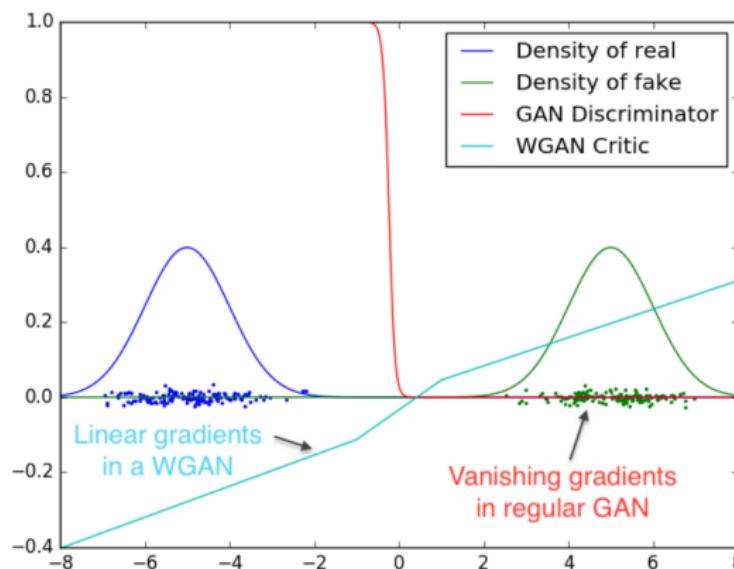
Require: : α , the learning rate. c , the clipping parameter. m , the batch size.
 n_{critic} , the number of iterations of the critic per generator iteration.

Require: : w_0 , initial critic parameters. θ_0 , initial generator's parameters.

- 1: **while** θ has not converged **do**
- 2: **for** $t = 0, \dots, n_{\text{critic}}$ **do**
- 3: Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
- 4: Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
- 5: $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))]$
- 6: $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
- 7: $w \leftarrow \text{clip}(w, -c, c)$
- 8: **end for**
- 9: Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
- 10: $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
- 11: $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
- 12: **end while**

Wasserstein GAN

For distinct $p(x)$ and $q(x)$ and tuned D usual GAN can't train ($\nabla D(G(z)) \approx 0$), but WGAN can.



Benefits

Benefits of WGAN:

- Objective convergence criterion $W(p, q_\theta)$
 - in usual GAN generator and discriminator optimize different functions
- For distinct $p(x)$ and $q(x)$ and tuned D usual GAN can't train ($\nabla D(G(z)) \approx 0$), but WGAN can.
- No mode collapse because
$$W(p, \delta(x - \arg \max_u p(u))) > W(p, p)$$
- More stable convergence&results for different architectures of generator/discriminator.

Wasserstein GAN with gradient penalty⁵

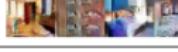
Another way to enforce Lipschitz property:

- remove weights clipping
- add regularizer $\lambda (\|\nabla_x f(x)\|_2 - 1)^2$ in optimization.
 - because Lipschitz property of order 1: $\|\nabla_x f(x)\|_2 \leq 1$

⁵Gulrajani et al. 2017

DCGAN vs. WGAN vs. WGAN-GP

Gives better quality for variable model design:

DCGAN	WGAN (clipping)	WGAN-GP	
Baseline (G : DCGAN, D : DCGAN)			
G : No BN and a constant number of filters, D : DCGAN			
G : 4-layer 512-dim ReLU MLP, D : DCGAN			
No normalization in either G or D			
Gated multiplicative nonlinearities everywhere in G and D			
tanh nonlinearities everywhere in G and D			
101-layer ResNet G and D			

Conclusion

- GAN - method to generate new realistic samples based on x_1, \dots, x_N .
- Original approach suffers from:
 - smart discriminator $\Rightarrow \approx 0$ gradients for G
 - flip labels, use soft labels, WassersteinGAN
 - mode collapse in G
 - use Gaussian noise, dropout in G
 - use WassersteinGAN (brings closer whole distributions)
- DCGAN - GAN designed for images generation.
 - fully convolutional G, D designed to better propagate gradients.
- ProgressiveGAN generates high resolution images.
 - gradual training of new layers