Generative adversarial networks

Victor Kitov

v.v.kitov@yandex.ru

Yandex
School of Data Analysis.

Table of Contents

1. Simple GAN
   - Model
   - Practical recommendations
2. Deep convolutional GAN
3. Semi-supervised learning with GAN
4. Conditional GAN
5. CycleGAN
6. Progressive GAN

1 Simple GAN
- Model
- Practical recommendations

Density estimation

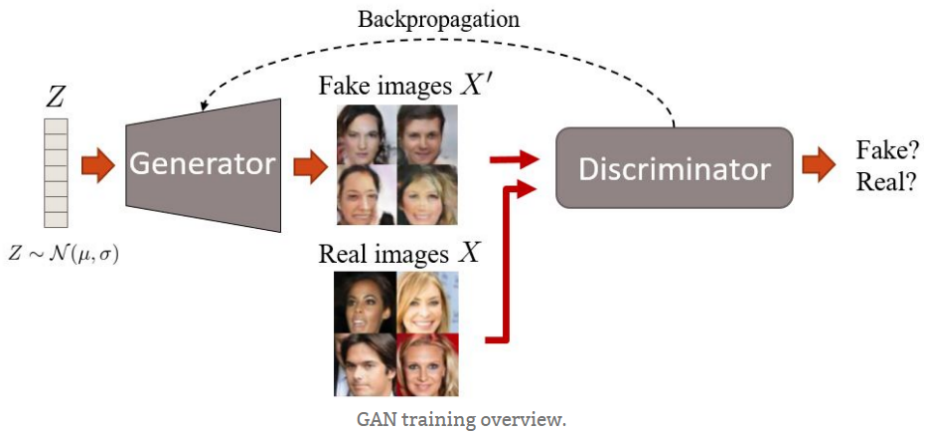
- Consider i.i.d. sample $x_1, ...x_N$, $x_i \sim p(x)$.
- Want to estimate $p(x)$. Approximate it with $q_\theta(x)$.
- **Explicit**: find θ from ML

$$\widehat{\theta} = \arg\max_\theta \frac{1}{N}\sum_{n=1}^{N} \ln q_\theta(x_n) = \arg\min_\theta KL\left(p||q_\theta\right)$$

- **Implicit density estimation** (used in GAN):
  - do not estimate function $q_\theta(x)$ explicity
  - instead make a sampler from $q_\theta(x)$
- We replace hard problem (estimation of multidimensional $p(x)$) with simple one (estimating a classifier).

Intuition of adversarial learning

Generative adversarial learning for images:



GAN training overview.

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 GAN[1]

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

[1] Link to paper.

Seminal paper on GAN[1]

- Two networks:
  - generator $G(z) : Z \to X$
    - outputs generated object x
  - discriminator $D(x) : X \to [0, 1]$
    - 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 $p(real) = p(fake) = \frac{1}{2}$

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

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 $p(real) = p(fake) = \frac{1}{2}$

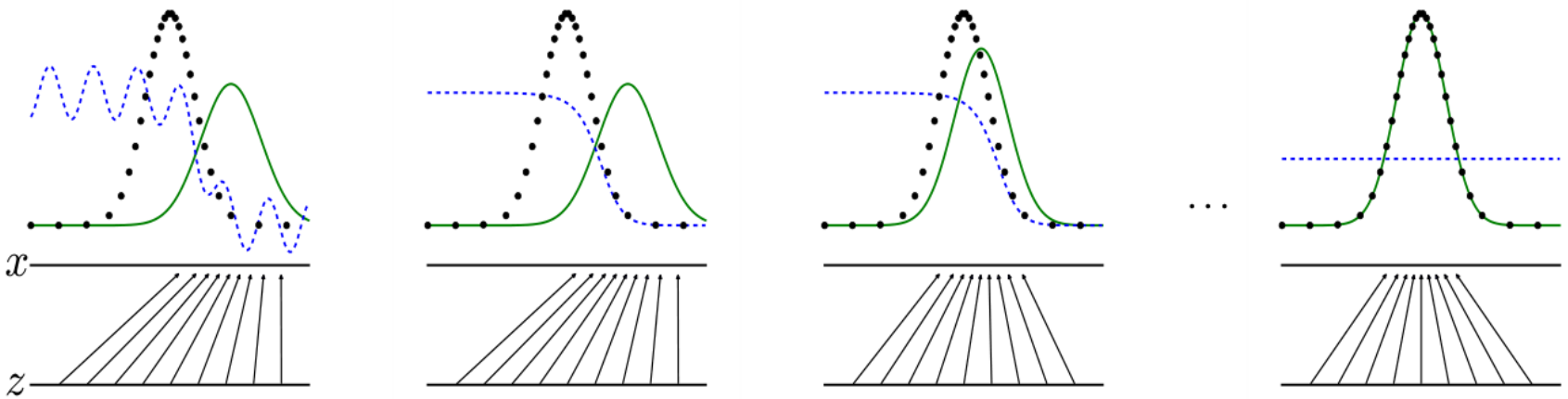
$$V(D, G) = \mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$

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

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

Illustration of incremetal 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):

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

G task (for fixed D):

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

Losses

D task (for fixed G):

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

G task (for fixed D):

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

- **Thus $G(z)$ learns to sample realistic samples from $p(x)$.**
- D and G should be trained syncronously.
  - 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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ 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:

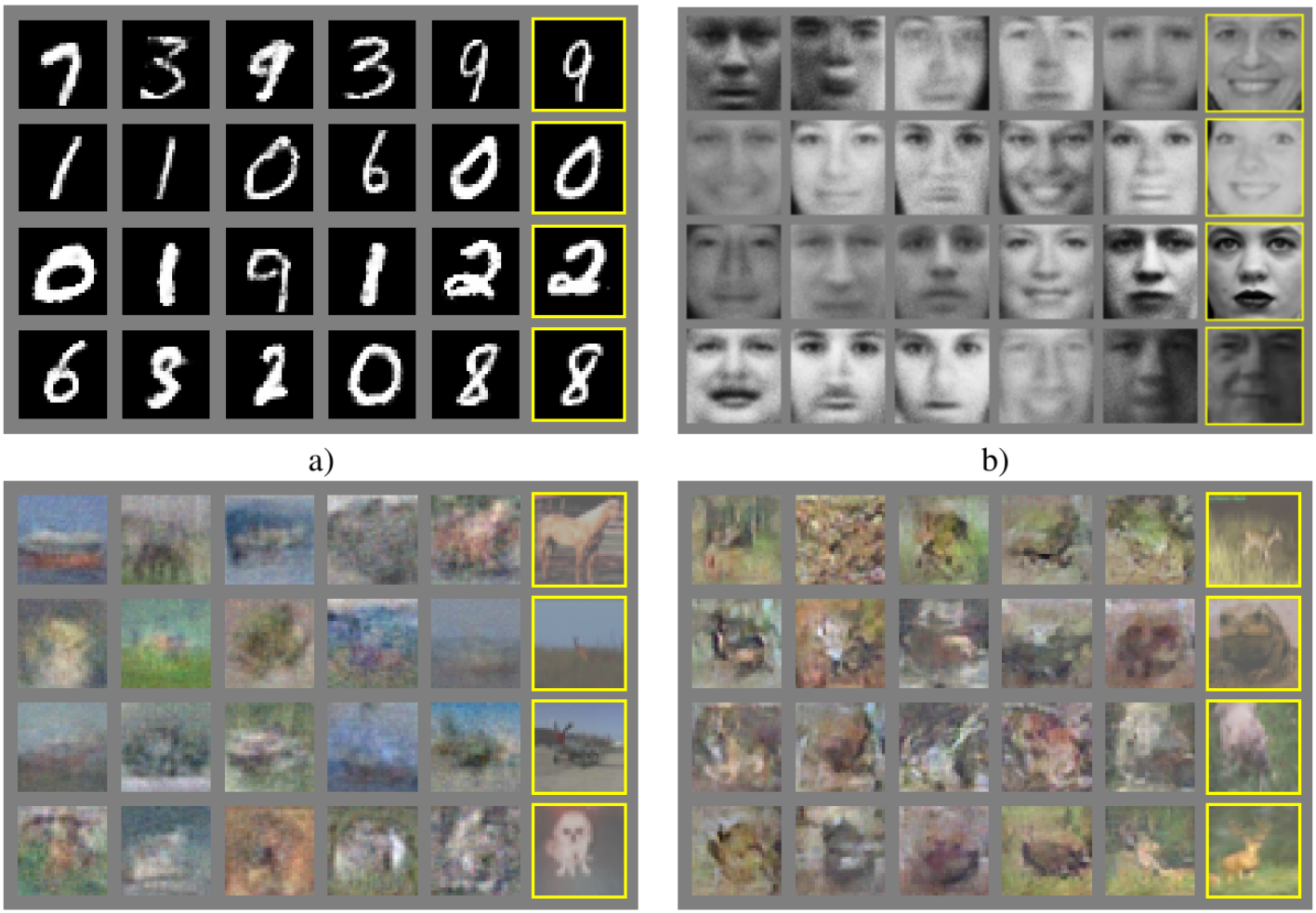
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  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.

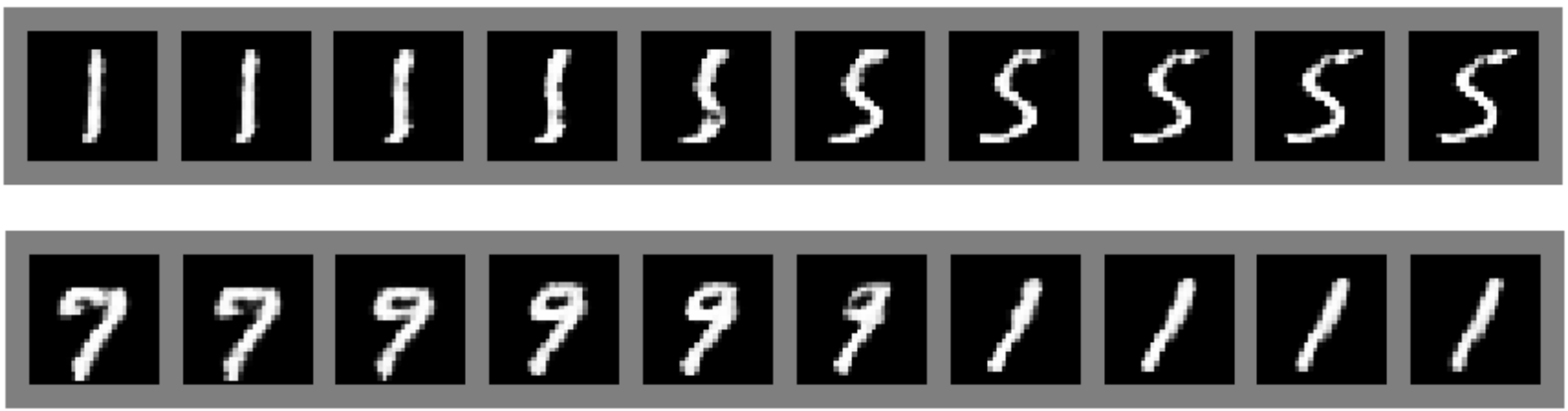
Generated images



a) b)

Interpolation in latent space

Linear interpolation of objects in latent space:



Optimal G^* given fixed D

- Optimal G^* given fixed D gives delta function $\delta\left(x - \arg\max_x D(x)\right)$.
  - called *mode collapse.*
- Even if didn't converge to mode, tends to collapse to regions with high $p(x)$.
  - no diversity of samples!

Optimal D^* given fixed G

Suppose D is flexible enough to take arbitrary values.

Theorem: For fixed G optimal discriminator is:

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

Optimal D^* given fixed G

Suppose D is flexible enough to take arbitrary values.
Theorem: For fixed G optimal discriminator is:

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

Proof:

$$V(G,D) = \int_x p(x)\log(D(x))dx + \int_z p(g(z))\log(1-D(g(z)))dz =$$
$$= \int_x p(x)\log(D(x))dx + q(x)\log(1-D(x))dx$$

Applying for $\forall x \in X$ $\arg\max_{d\in\mathbb{R}}\{p\log(d)+q\log(1-d)\} = \frac{p}{p+q}$ for any $p,q \in \mathbb{R}$ obtain

$$\arg\max_D V(G,D) = \frac{p(x)}{p(x)+q(x)}$$

Optimal G^* given D^*

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

Optimal G^* given D^*

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

$$V(G^*, D^*) = \int_x p(x) \log \frac{p(x)}{p(x) + q^*(x)} dx + \int_x q(x) \log \frac{q(x)}{p(x) + q^*(x)} dx$$
$$= \int_x p(x) \log \frac{1}{2} dx + \int_x q(x) \log \frac{1}{2} dx = 2 \log \frac{1}{2} = -2 \log 2$$

Optimal G^* given D^*

And $V(G^*, D^*) = \min_G V(G, D^*)$, because

$$V(G, D^*) - V(G^*, D^*)$$

$$= \int_x p(x) \log \frac{p(x)}{p(x)+q(x)} dx + \int_x q(x) \log \frac{q(x)}{p(x)+q(x)} dx$$

$$+ \int_x p(x) \log 2 dx + \int_x q(x) \log 2 dx$$

$$= \int_x p(x) \log \frac{p(x)}{\frac{p(x)+q(x)}{2}} dx + \int_x q(x) \log \frac{q(x)}{\frac{p(x)+q(x)}{2}} dx$$

$$= KL\left(p(x)||\frac{p(x)+q(x)}{2}\right) + KL\left(q(x)||\frac{p(x)+q(x)}{2}\right) \geq 0$$

1 Simple GAN
- Model
- Practical recommendations

Finding unstable saddle-point

D task (for fixed G):

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

G task (for fixed D):

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

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

- requires a lot of fine-tuning, see link to recommendations.

Amplification of gradients

Problem:

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

Amplification of gradients

Problem:

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

Solution:

- improve G from another criterion:

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

- $\log(\cdot)$ exponentially amplifies small changes in gradient near zero.
  - G gets non-degenerate feedback.
  - D, G game stops being game with zero-sum.

Table of Contents

1 Simple GAN

2 Deep convolutional GAN

3 Semi-supervised learning with GAN

4 Conditional GAN

5 CycleGAN

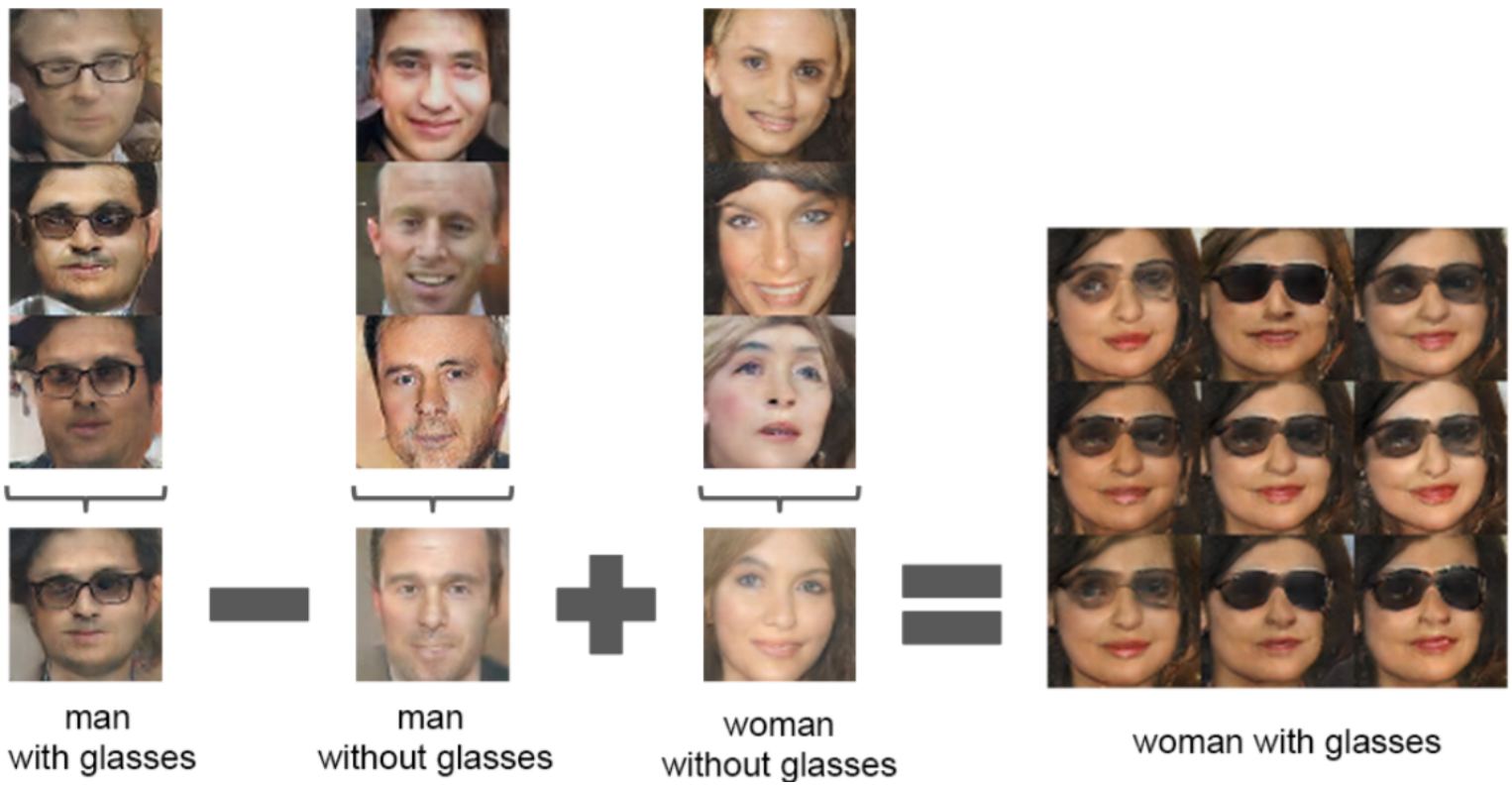
6 Progressive GAN

Deep convolutional GAN (DCGAN)[2]

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

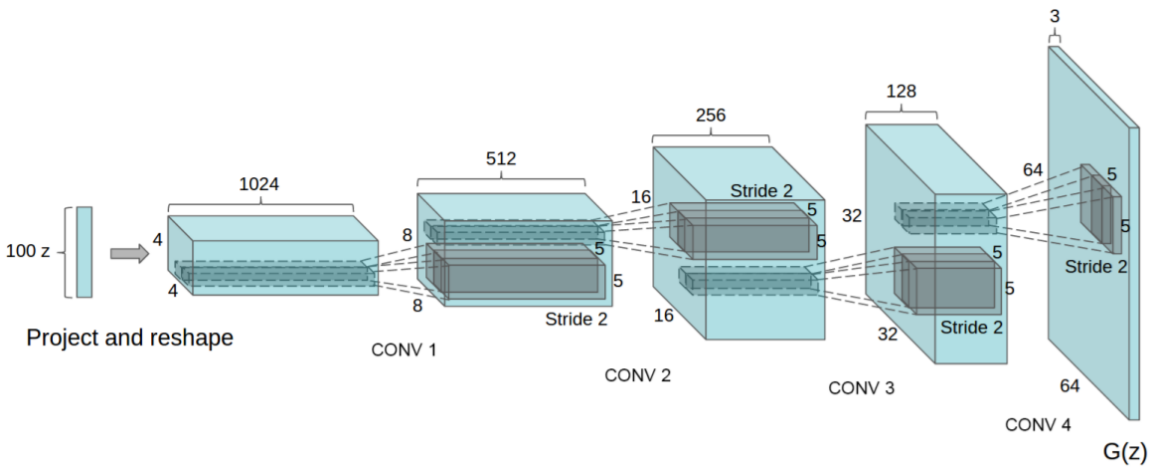
[2] https://arxiv.org/pdf/1511.06434.pdf

Latent space arithmetics



Generator

Uses fully-convolutional generator:



Architecture guidelines for stable DCGANs

1. Replace any pooling layers with fractional-strided convolutions in G and strided convolutions in D.
2. Use LeakyReLU activation in the discriminator for all layers.
3. Use batchnorm in both the generator and the discriminator.
4. Remove fully connected hidden layers for deeper architectures.
5. 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 Semi-supervised learning with GAN

4 Conditional GAN

5 CycleGAN

6 Progressive GAN

Semi-supervised learning with GAN[3]

- Semisupervised GAN (SGAN):
  - classifier and discriminator are united
  - classifier outputs $C+1$ probabilities:

$$[p(y=1|x), ...p(y=C|x), p(x \text{ was generated}|x)]$$

- **Weight sharing** between discriminator and classifier help each other.

[3] Link to paper.

Algorithm of SGAN

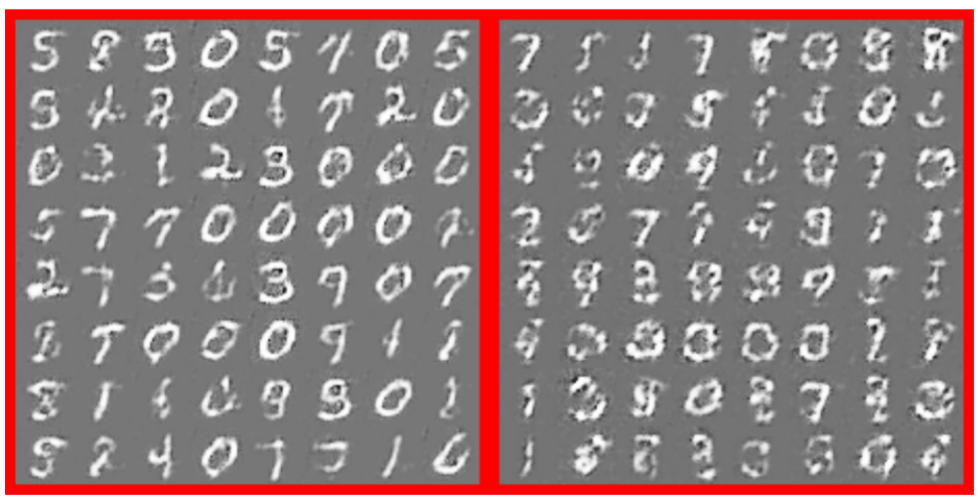
Algorithm 1 SGAN Training Algorithm

Input: I: number of total iterations
for $i = 1$ **to** I **do**
  Draw m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  Draw m examples $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ from data generating distribution $p_d(x)$.
  Perform gradient descent on the parameters of D w.r.t. the NLL of D/C's outputs on the combined minibatch of size $2m$.
  Draw m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  Perform gradient descent on the parameters of G w.r.t. the NLL of D/C's outputs on the minibatch of size m.
end for

SGAN experiments

SGAN convegres faster:

Generated MNIST images by SGAN (left) and GAN (right) after 2 MNIST epochs



SGAN experiments

Semi-supervised learning improves accuracy for small training sets.

Accuracy comparisons of supervised and semisupervised classifier on MNIST:

EXAMPLES	CNN	SGAN
1000	0.965	0.964
100	0.895	0.928
50	0.859	0.883
25	0.750	0.802

Table of Contents

1 Simple GAN

2 Deep convolutional GAN

3 Semi-supervised learning with GAN

4 Conditional GAN

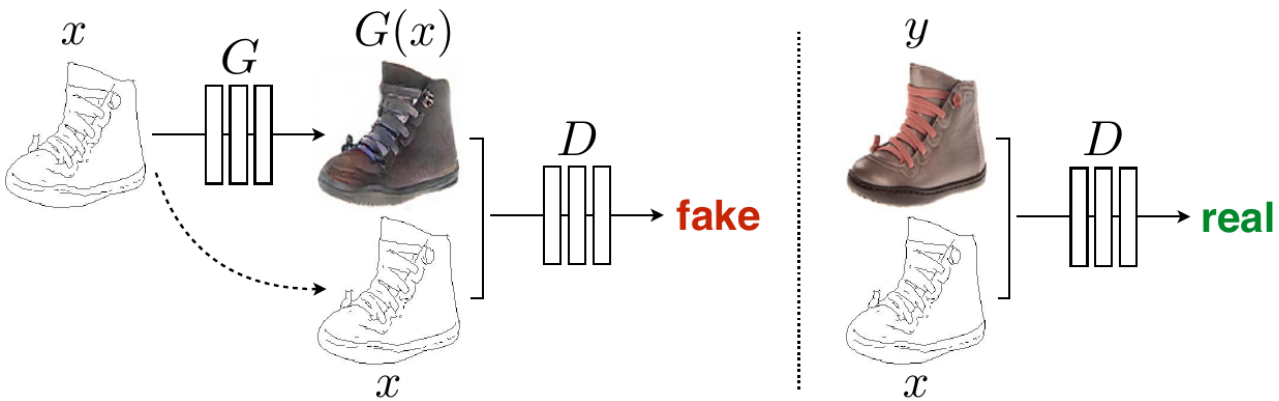
5 CycleGAN

6 Progressive GAN

Conditional GAN[4]

- z - random state from standard distribution $p(z)$
- x - source version of object.
- y - target version of object
- $\widehat{y} = G(x, z)$ - reconstruction of target

Conditional GAN:



[4] Isola et al. 2017.

Loss functions

ConditionalGAN (cGAN) objective:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}\left[\log D(x, y)\right] + \mathbb{E}_{x,z}\left[\log\left(1 - D(x, G(x, z)\right]\right.$$

- score for D, loss for G
- promotes realizm of generated $\widehat{y}$
- power of GAN: realism loss is determined automatically, not hard-coded.

Reconstruction loss for D:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}\left\|y - G(x, z)\right\|_1$$

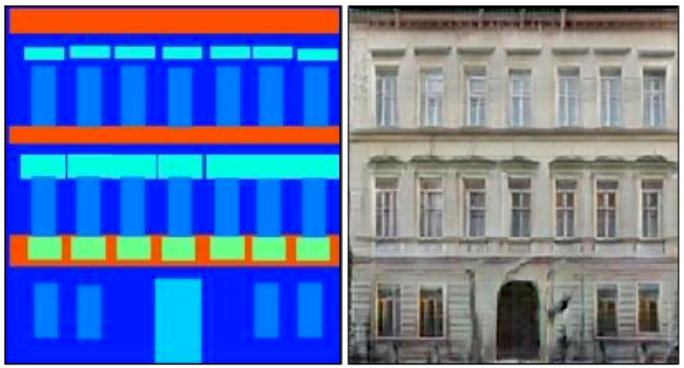
Generator solves:

$$G^* = \arg\min_G \max_D \left\{\mathcal{L}_{cGAN}(G, D) + \lambda\mathcal{L}_{L1}(G)\right\}$$

Results for different X,Y

Results for different X,Y

Labels to Facade



input output

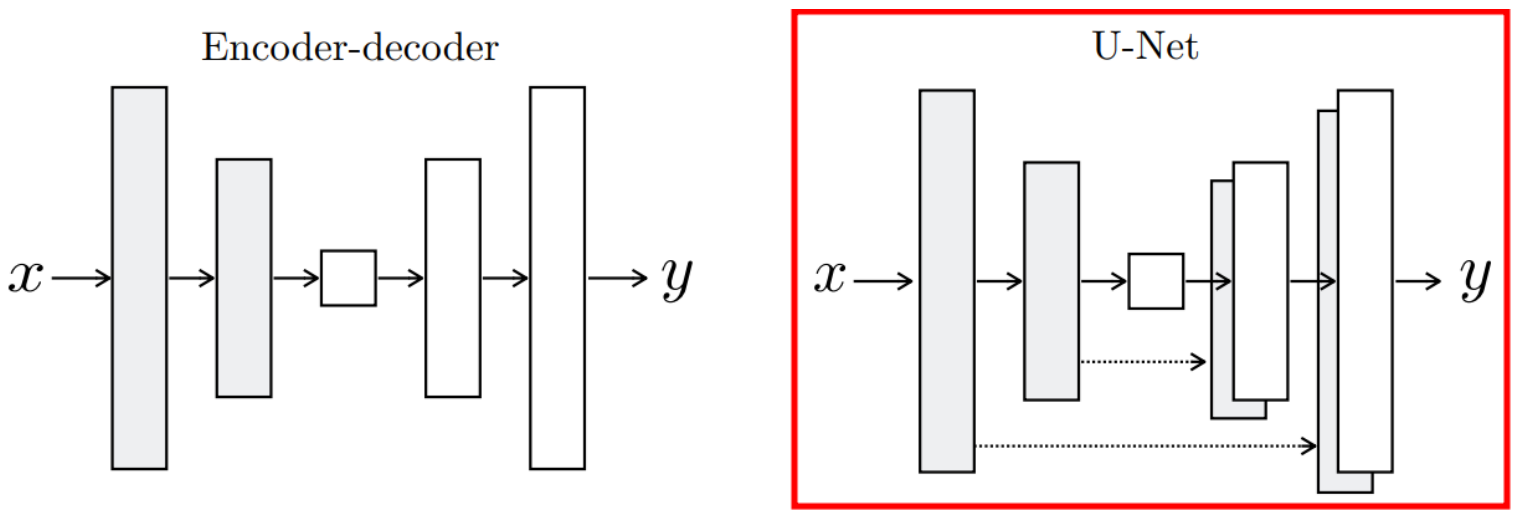
Day to Night

input output

Results for different X,Y

Comments

- Works for any X, Y.
  - in particular, may switch input and output spaces.
- For labeled X system generalizes to human input: online demo.
- Generator uses U-net architecture:



- Discriminator is applied to overlapping patches of size NxN
  - allows application to arbitrary sized images.
  - $N = 70$ worked best

Table of Contents

1 Simple GAN

2 Deep convolutional GAN

3 Semi-supervised learning with GAN

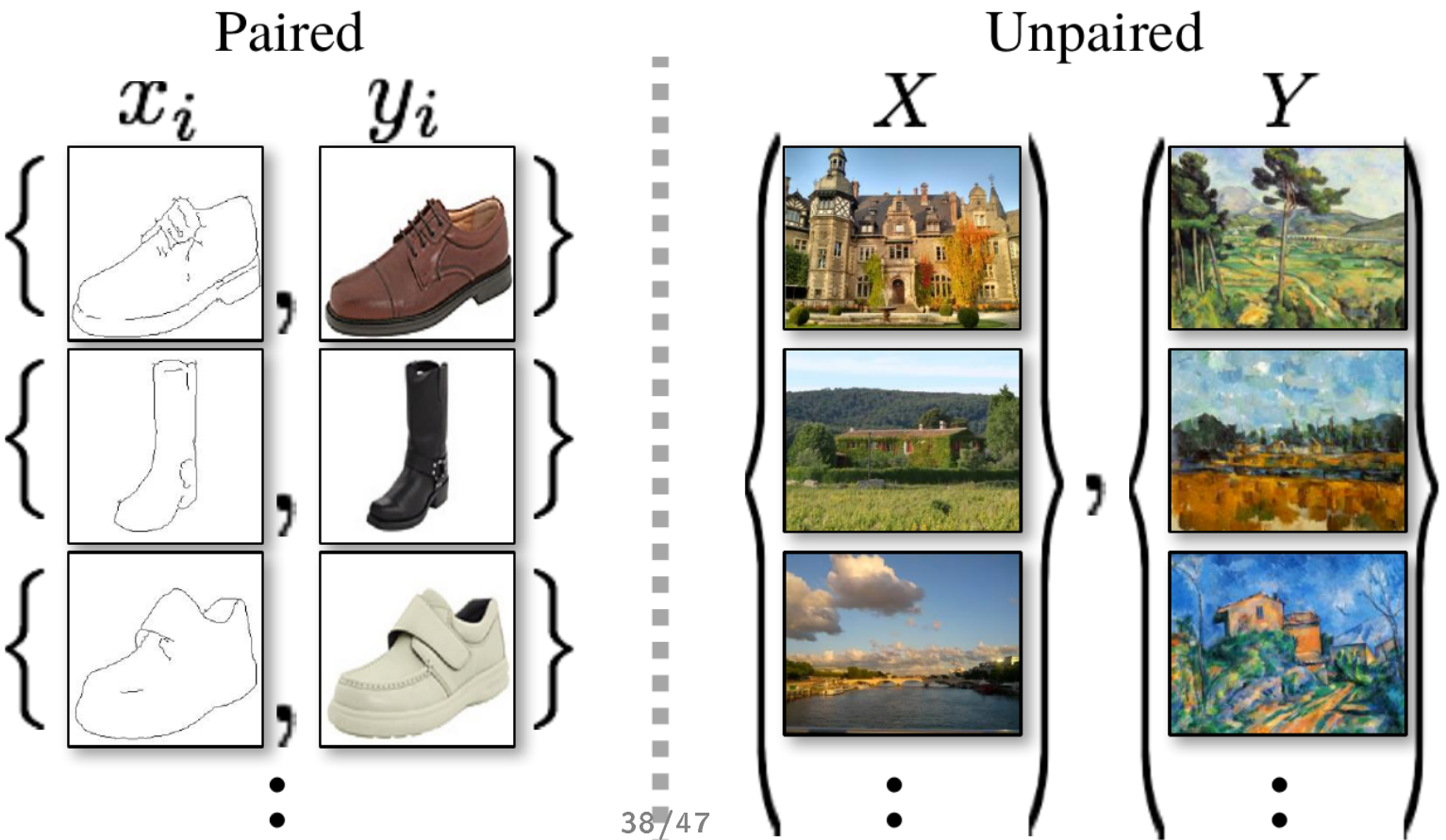
4 Conditional GAN

5 CycleGAN

6 Progressive GAN

CycleGAN[5]

- 2 domains: X, Y
- **Training set is unpaired**: $x_1, x_2, ...x_N \in X$, $y_1, y_2, ...y_M \in Y$.
  - in contrast, conditional GAN was trained on pairs $\{(x_n, y_n)\}_{n=1}^{N}$



Losses

$$\begin{aligned}\mathcal{L}_{GAN}(G_{XY}, D_Y) =& \mathbb{E}_{y\sim p(y)}\left[\log\left(D_Y(y)\right)\right] \\ &+ \mathbb{E}_{x\sim p(x)}\left[\log\left(1 - D_Y(G_{XY}(x))\right)\right]\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{GAN}(G_{YX}, D_X) =& \mathbb{E}_{x\sim p(x)}\left[\log\left(D_X(x)\right)\right] \\ &+ \mathbb{E}_{y\sim p(y)}\left[\log\left(1 - D_X(G_{YX}(y))\right)\right]\end{aligned}$$

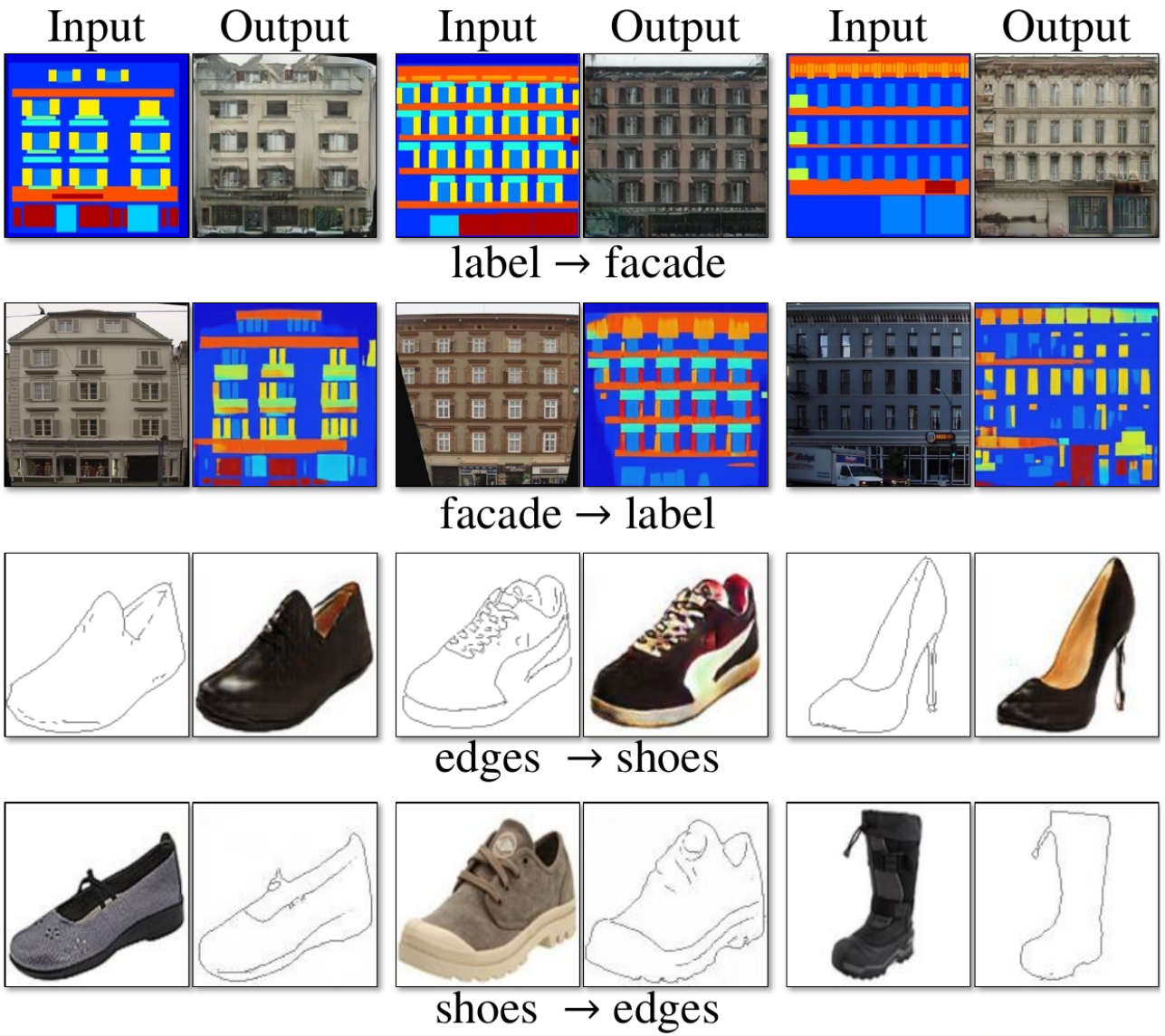
$$\begin{aligned}\mathcal{L}_{cyc}\left(G_{XY}, G_{YX}\right) =& \mathbb{E}_{x\sim p(x)}\left\|G_{YX}(G_{XY}(x)) - x\right\|_1 \\ &+ \mathbb{E}_{y\sim p(y)}\left\|G_{XY}(G_{YX}(y)) - y\right\|_1\end{aligned}$$

CycleGAN

$$\mathcal{L}(G_{XY}, G_{YX}, D_X, D_Y) = \mathcal{L}_{GAN}(G_{XY}, D_Y) + \mathcal{L}_{GAN}(G_{YX}, D_X) + \mathcal{L}_{cyc}(G_{XY}, G_{YX})$$

$$G^*_{XY}, G^*_{YX} = \arg\min_{G_{XY}, G_{YX}} \max_{D_X, D_Y} \mathcal{L}(G_{XY}, G_{YX}, D_X, D_Y)$$

Demo results



Identity regularizer

- Consider paintings->photo reconstruction.
- To remove unnecessary colour change in output identity loss was added:
  - intuition: do nothing, when input is from target set.

$$\mathcal{L}_{identity} = \mathbb{E}_{x\sim p(x)} \|G_{YX}(x) - x\|_1 + \mathbb{E}_{y\sim p(y)} \|G_{XY}(y) - y\|_1$$

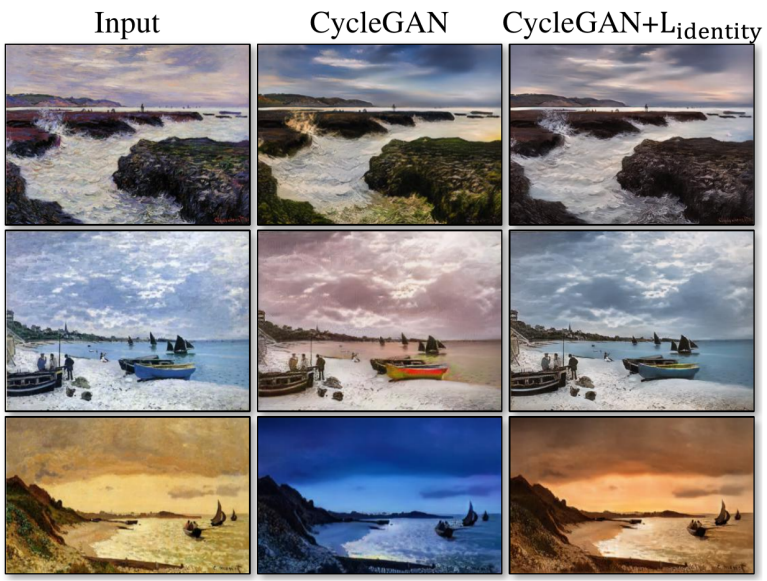


Table of Contents

1 Simple GAN

2 Deep convolutional GAN

3 Semi-supervised learning with GAN

4 Conditional GAN

5 CycleGAN

6 Progressive GAN

Progressive GAN[6]

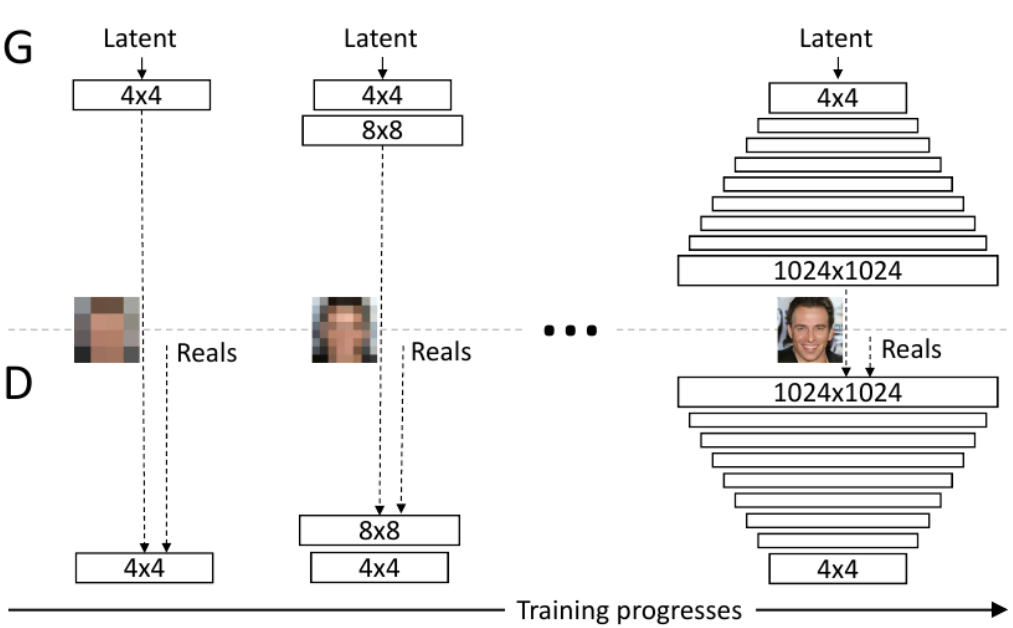
- Techique to generate high dimensional output.

Generated photographs, using celebreties dataset

[6] Karras 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).
This way we gradually introduce new layer to the model.

Conclusion

- GANs - method to generate new realistic samples based on $x_1, \ldots x_N$.
  - requires tuning to syncronize D and G training
  - suffers from mode collapse
- DCGAN - GAN designed for images generation.
- Conditional GAN generates $\widehat{y}$ conditional on input
  - impainting, colorization, deblurring, noise removal, etc.
- CycleGAN learns reconstructions on unpaired datasets.
- ProgressiveGAN generates high resolution images.