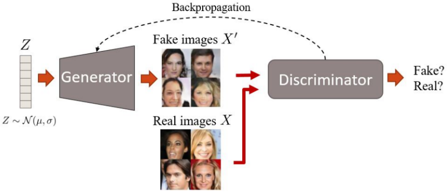
Generative adversarial networks

Victor Kitov

v.v.kitov@yandex.ru

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]

- 2 multilayer perceptrons:
  - generator $G(z) = G(z|\theta_g)$
    - outputs generated object x
  - discriminator $D(x) = D(x|\theta_d)$
    - probability that x is from training set and not generated by G.

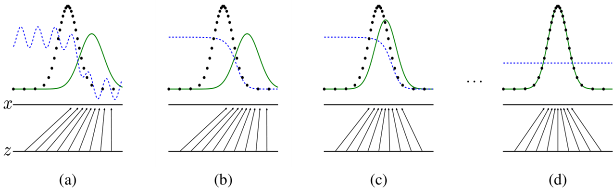
[1] https://arxiv.org/pdf/1406.2661.pdf

Game

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

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

Incremental learning: $\langle\rangle$



black dotted: $p_{data}(x)$; green: $p_{generated}(x)$; blue: $D(x) = p(x \text{ is true}|x)$

Losses

Score for discriminator (for fixed θ_g):

$$\mathbb{E}_{x\sim p_{data}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right] \to \max_{\theta_d}$$

Score for generator (probability of being detected):

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

- on early iterations generator is very unrealistic
- so $D(G(z)) \approx 0$, gradient of $\log(1-D(G(z))$ is small.
- better works another score:

$$\mathbb{E}_{z\sim p_z(z)}\left[\log(D(G(z)))\right] \to \max_{\theta_g}$$

Algorithm

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ 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.

Optimal value for discriminator

Theorem: For fixed G optimal discriminator is:

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

Proof:

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

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

Since $\arg\max_y \{a\log(y) + b\log(1-y)\} = \frac{a}{a+b}$ for any a, b =>

$$\arg\max_D V(G,D) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Optimal

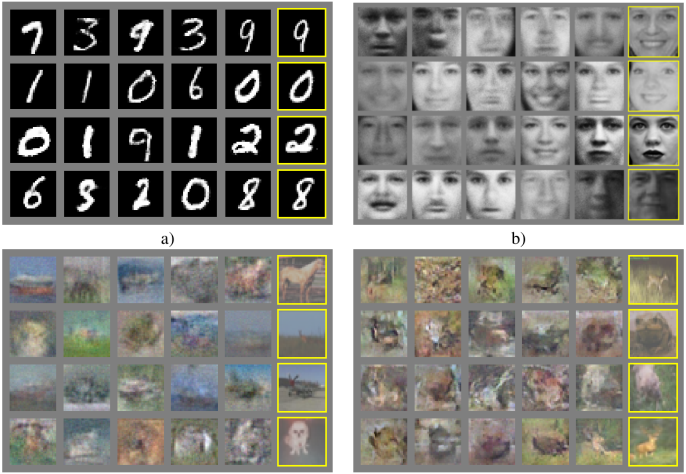
Generator cost function:

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

This is maximized for $p_g(x) = p_{data}(x)$:

$$C(G) = \mathbb{E} \log \frac{1}{2} + \mathbb{E} \log \frac{1}{2}$$

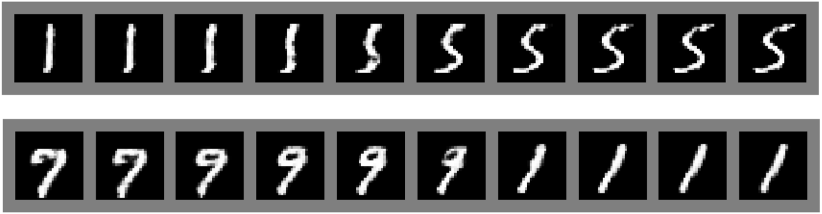
Generated images



a) b)

Latent space

Linear interpolation of objects in latent space:



Results

Parzen-window based log-likelihood:

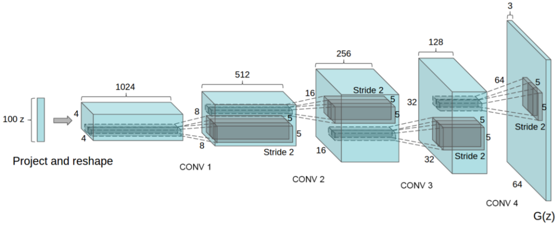
- MNIST - dataset of digit images
- TFD - Toronto faces dataset

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [6]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

Table of Contents

Deep convolutional GAN[2]

Architecture of DCGAN generator:



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

Architecture guidelines for stable DCGANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Generated bedroooms

Latent space arithmetics

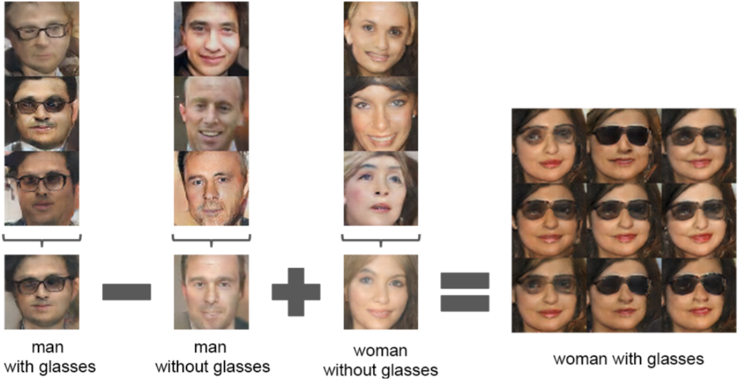


Table of Contents

Semi-supervised learning with GAN[3]

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

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

- Discriminator and classification have shared weights helping 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 $p_g(z)$.

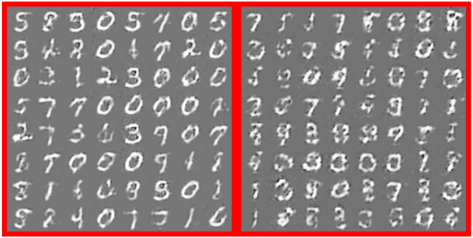
Perform gradient descent on the parameters of G w.r.t. the NLL of D/C's outputs on the minibatch of size m.

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

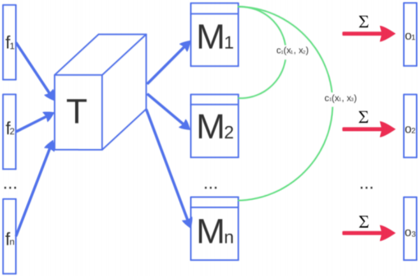
Problem of model collapse

- Problem - generator can converge to reproduce one most typical object
- Reason - discriminator deals with objects one by one.
- Solution - train discriminator on minibatch[4]

[4] https://arxiv.org/pdf/1606.03498.pdf

Algorithm

- Let $f(x_i) \in \mathbb{R}^A$ - vector of features of some intermediate layer of discriminator.
- Multiply it by trainable tensor $T \in \mathbb{R}^{AxBxC}$:
  $f(x_i) * T = M \in R^{BxC}$
- Calculate $c_b(x_i, x_j) = e^{-\left\|M_{i,b} - M_{j,b}\right\|_1} \in \mathbb{R}$, $b = 1, 2, ...B$, $i, j = 1, ...n$.
  - b-row number.



Algorithm

- $o(x_i)_b = \sum_{j=1}^{n} c_b(x_i, x_j) \in \mathbb{R}$
- $o(x_i) = [o(x_i)_1, o(x_i)_2, ...o(x_i)_B] \in \mathbb{R}^B$
- Feed to discriminator concatenation $[o(x_i), f(x_i)]$
- We compute minibatch features separately for
    - minibatches of training data
    - minibathes of generated data

How it works

- So discriminator classifies single object, but knows side information about its context.
    - in model collapse $f(x_i)$ would be much less diverse for generated minibatches, than for true ones
    - discriminator will account for that

Yet another criteria to train discriminator[5]

- Idea: discriminator on inner layers extracts discriminative features f.
- Fit generator so that statistics of $f(Gen(z))$ and $f(x)$ for real x are the same.
- New loss for generator:

$$\left\|\mathbb{E}_{x\sim p_{data}} f(x) - \mathbb{E}_{z\sim p_z(z)} f(Gen(z))\right\|_2^2$$

- This heuristic helps to improves convergence of original GAN algorithm
  - non-convergence is usually cyclic when modification of Gen makes modifications of Dis obsolete and vice versa in terms of common loss function.

[5] https://arxiv.org/pdf/1606.03498.pdf

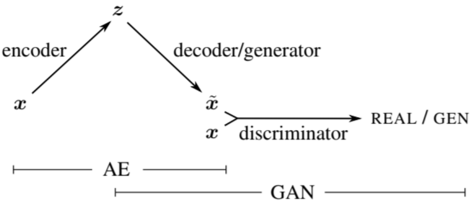
Table of Contents

Problem

- VAE is trained to generate versatile objects
  - because it is tuned to reproduce most of training set
  - but gives smoothed output
- GAN is subject to model collapse
  - but gives realistic output
- Idea: combine them[6]

[6] https://arxiv.org/pdf/1512.09300.pdf

Proposed arcitecture



- Generator of VAE = generator of GAN

VAE reminder

- $z \sim Enc(x) = q(z|x)$
- $\tilde{x} \sim Dec(z) = p(x|z)$
- $\mathcal{L}_{VAE} = \mathcal{L}_{log-lik} + \mathcal{L}_{prior}$
- $\mathcal{L}_{log-lik} = -\mathbb{E}_{q(z|x)}\left[\log p(x|z)\right]$: quality of reconstruction
  - shows, how close are x and $\tilde{x}$
- $\mathcal{L}_{prior} = D_{KL}(q(z|x)||p(z))$: prior for latent variables
  - usually $p(z) = \mathcal{N}(0, I)$

Proposed modification

- Oversmoothed output of VAE - because of element-wise loss in VAE
- Let's extract high level features from intemediate layer of discriminator!
- Replace $\mathcal{L}_{log-lik}$ from output space to high level features space
- x-real, from VAE: $x \to z \to \tilde{x}$ (z and $\tilde{x}$ are sampled). Are $\tilde{x}$ close to x?
- Let $Dis_l(x)$ - hidden layer l representation of x by discriminator.
- Assume
$$p(Dis_l(x)|z) = \mathcal{N}(Dis_l(x)|Dis_l(\tilde{x}), I)$$
- Replace original $\mathcal{L}_{log-lik}$ with
$$\mathcal{L}_{log-lik} = -\mathbb{E}_{q(z|x)}\left[\log p\left(Dis_l(x)|z\right)\right]$$
- Model is trained on $\mathcal{L}_{prior} + \mathcal{L}_{log-lik} + \mathcal{L}_{GAN}$

Training VAE/GAN model

$\boldsymbol{\theta}_{\text{Enc}}, \boldsymbol{\theta}_{\text{Dec}}, \boldsymbol{\theta}_{\text{Dis}} \leftarrow$ initialize network parameters
repeat

$\quad \boldsymbol{X} \leftarrow$ random mini-batch from dataset

$\quad \boldsymbol{Z} \leftarrow \text{Enc}(\boldsymbol{X})$

$\quad \mathcal{L}_{\text{prior}} \leftarrow D_{\text{KL}}(q(\boldsymbol{Z}|\boldsymbol{X}) \| p(\boldsymbol{Z}))$

$\quad \tilde{\boldsymbol{X}} \leftarrow \text{Dec}(\boldsymbol{Z})$

$\quad \mathcal{L}_{\text{llike}}^{\text{Dis}_l} \leftarrow -\mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{X})}\left[p(\text{Dis}_l(\boldsymbol{X})|\boldsymbol{Z})\right]$

$\quad \boldsymbol{Z}_p \leftarrow$ samples from prior $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

$\quad \boldsymbol{X}_p \leftarrow \text{Dec}(\boldsymbol{Z}_p)$

$$\mathcal{L}_{\text{GAN}} \leftarrow \log(\text{Dis}(\boldsymbol{X})) + \log(1 - \text{Dis}(\tilde{\boldsymbol{X}})) + \log(1 - \text{Dis}(\boldsymbol{X}_p))$$

$\quad$ // Update parameters according to gradients

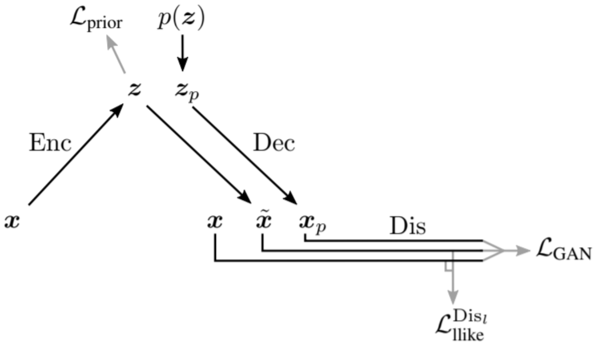
$\quad \boldsymbol{\theta}_{\text{Enc}} \stackrel{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Enc}}}(\mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{llike}}^{\text{Dis}_l})$

$\quad \boldsymbol{\theta}_{\text{Dec}} \stackrel{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Dec}}}(\gamma \mathcal{L}_{\text{llike}}^{\text{Dis}_l} - \mathcal{L}_{\text{GAN}})$

$\quad \boldsymbol{\theta}_{\text{Dis}} \stackrel{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Dis}}} \mathcal{L}_{\text{GAN}}$

until deadline

Data flow during training



Comparison of results

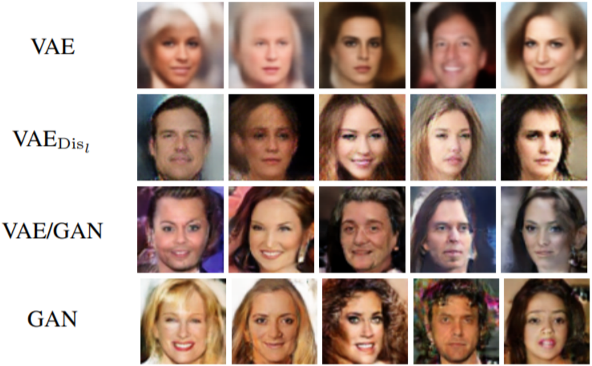


Table of Contents

Experiments[7]

- From transformed image->reconstruct original image
  - denoising, super-resolution, deblurring.
- Quality metric: peak signal-to-noise ratio (PSNR)
- Datasets:
  - Human faces - Large-scale CelebFaces Attributes Dataset
  - Natural scenes - MIT Places Database

[7] From http://stanford.edu/class/ee367/Winter2017/yan_wang_ee367_win17_report.pdf

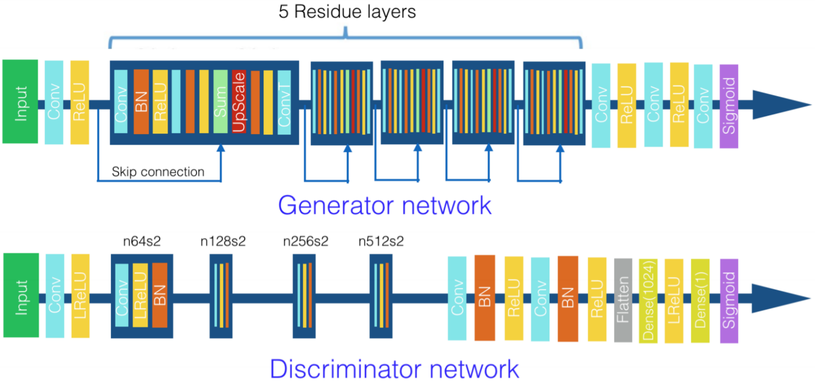
Architecture

- 2 networks: generator, discriminator.
- Discriminator tries to discriminate whether:
    - image came from the training set
    - image came from the generator
- Generator takes **corrupted image as input** and tries to reconstruct original image.

Losses

- **Generator loss:** $0.9\mathcal{L}_{content} + 0.1\mathcal{L}_{G,advers}$
    - $\mathcal{L}_{content} = \left\| I - \widehat{I} \right\|_1$, where I-original and $\widehat{I}$-reconstructed image.
    - $\mathcal{L}_{G,advers}$-standard generator adversarial loss.
- **Discriminator loss:** $\mathcal{L}_{D,advers}$
    - $\mathcal{L}_{D,advers}$-standard discriminator adversarial loss.

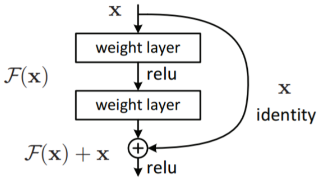
Generator, discriminator structure



Generator details

- Residual networks are used in generator.[8]
- Key idea of residual network:
    - use much more layers
    - layers grouped into groups with similar structure
    - each group learns **small correction** to identity function (to prevent overfitting)

Building block of residual network:



[8] https://arxiv.org/pdf/1512.03385.pdf

Definitions

- I: original image
- K: reconstructed image
- m, n: image dimensions
- Mean squared error (MSE):
    - for grayscale images:

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j) - K(i,j)]^2$$

    - for (r,g,b) images (let c be color channel):

$$MSE = \frac{1}{3mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{c=1}^{3}[I(i,j,c) - K(i,j,c)]^2$$

- MAX: maximum possible pixel value
    - for B-bit image $MAX = 2^B - 1$

Peak signal-to-noise ratio (PSNR)[9]

PSNR measures **quality** of image reconstruction:

$$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE}\right)$$

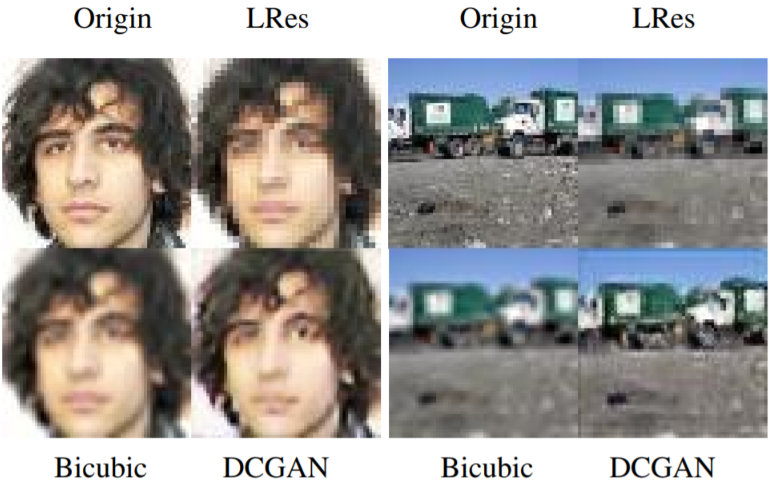
[9] https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

5. Application use-case
    - Peak signal-to-noise ratio (PSNR)
    - Experiments

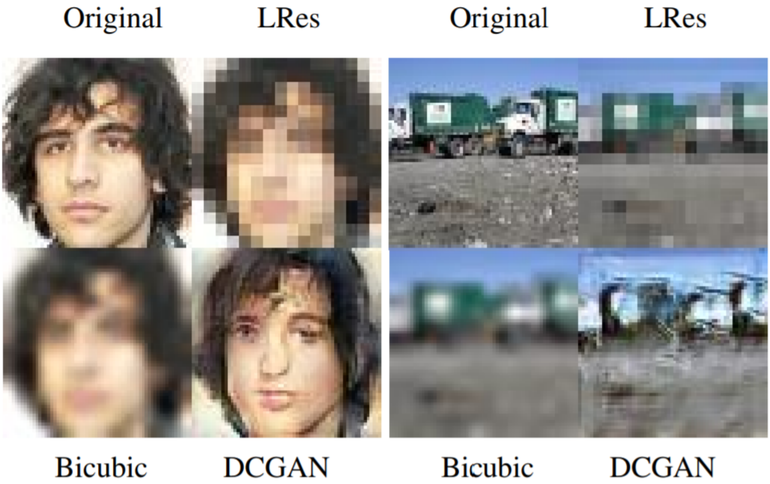
Super-resolution

- **Super-resolution:** recover higher resolution image from its low resolution variant.
  - e.g. from limited device zoom capacity (camera, microscope)
- **Baseline algorithms:**
  - naive scaling (LRes)
  - bicubic interpolation (Bicubic)
- **Results:**
  - PSNR of bicubic is best, but GAN-reconstructed images are more sharp
    and more good-looking for humans (retain high level features).
  - GAN super-resolution for faces works better than for places (which are less typical)

Super-resolution outputs (subsampling=2)



Super-resolution outputs (subsampling=4)



Baselines

- **Denoising:** noisy image->clean image
  - e.g. from measurement imperfection.
- **Baseline algorithms:**
  - median filter
  - non-local means
- **Results:**
  - PSNR are comparable, but GAN-reconstructed images are more sharp
    and more good-looking for humans (retain high level features).

Non-local means baseline[10]

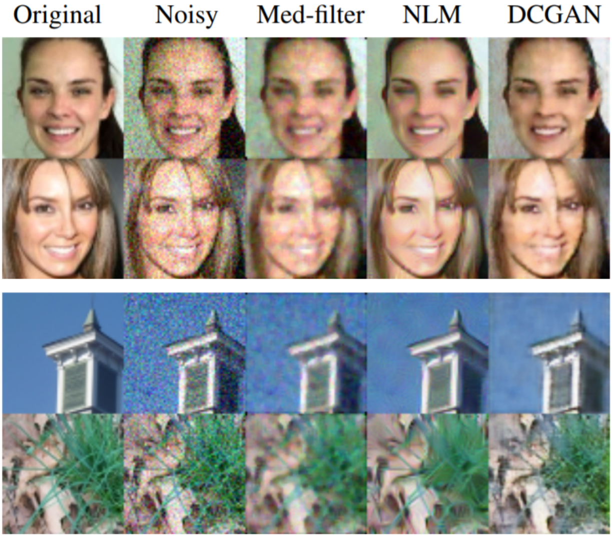
$$u(p)=\frac{1}{C(p)}\sum_{q\in\Omega}v(q)f(p,q)$$

where we used definitions:

- $v(\cdot)$: original image with noise
- $u(\cdot)$: denoised image
- p,q: image locations
- $f(p,q)$: similarity of pixels p,q by their neighborhoods $R(\cdot)$
- $C(p)=\sum_{q\in\Omega}f(p,q)$
- $f(p,q)=e^{-\frac{1}{h^2}|B(q)-B(p)|^2}$
- $B(p)=\frac{1}{|R(p)|}\sum_{i\in R(p)}v(i)$

[10] https://en.wikipedia.org/wiki/Non-local_means

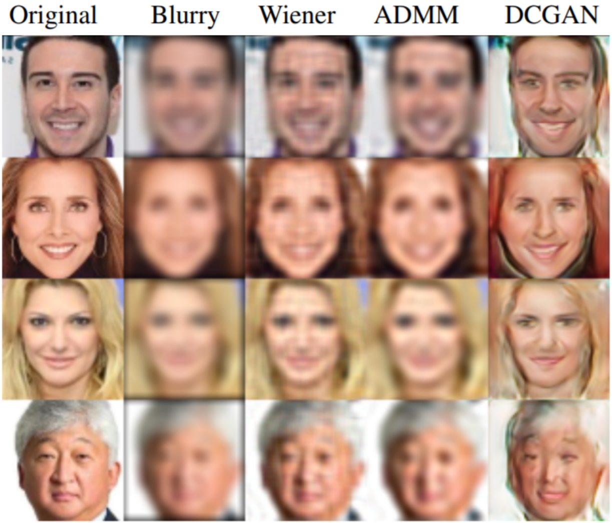
Denoising outputs



Deblurring

- **Deblurring:** images blurred and small Gaussian noise added.
    - e.g. from camera motion.
- **Baseline algorithms:**
    - Wiener filter
    - alternating direction method of multipliers (ADMM)
- **Results:**
    - PSNR of GAN is lower, but GAN-reconstructed images are more sharp
      and more good-looking for humans (retain high level features).
    - GAN super-resolution for faces works better than for places (which are less typical)

Deblurring faces outputs



Deblurring places outputs (not accurate)

Analysis of experiments

- Unequal conditions:
  - Baseline methods use only test image.
  - GAN uses information from the whole training set.
- GANs give smaller PSNR
  - may be attributed to small training set
- GANs give more sharp output
  - to fool "blurry-based" discriminator
  - do not fallback to averaging as standard methods

Another possible GAN application: impainting

Analysis of experiments

- GANs reproduce small details on images
  - details learned from other images of the training set.
- GAN performance can be improved by training on specific subsets of objects
  - e.g. train separate face models for different sex, age, nationality, etc.
  - especially important for diverse objects such as places.