GANs: semi-supervised, pix2pix, cycleGAN

Victor Kitov

v.v.kitov@yandex.ru

Table of Contents

Semi-supervised learning with GAN[1]

- 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.

[1] Link to paper.

Algorithm of SGAN

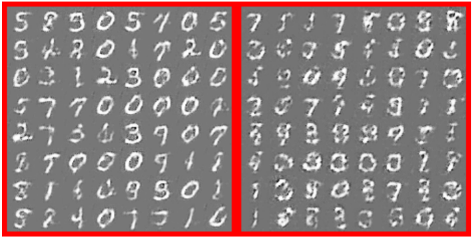
Algorithm 1 SGAN Training Algorithm

Input: I: number of total iterations
for $i = 1$ **to** I **do**
  Draw m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
  Draw m examples $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ 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 $\{z^{(1)}, \dots, z^{(m)}\}$ 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:

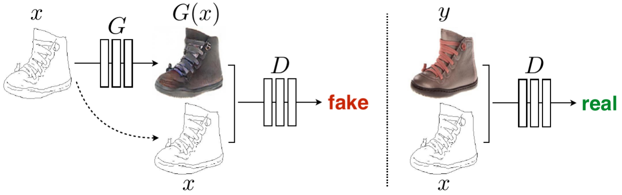
EXAMPLES	CNN	SGAN
1000	0.965	0.964
100	0.895	0.928
50	0.859	0.883
25	0.750	0.802

Table of Contents

Conditional GAN[2]

- 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:



[2] 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:

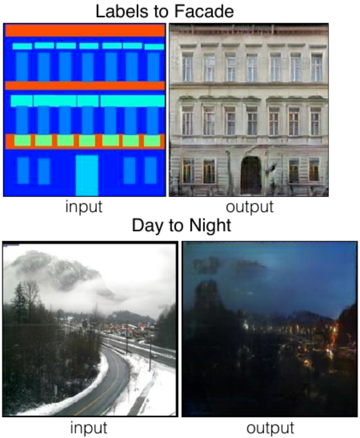
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}\left\|y - G(x, z)\right\|_1$$

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

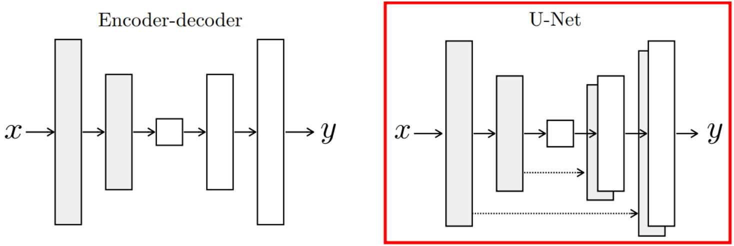
Results for different X,Y



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

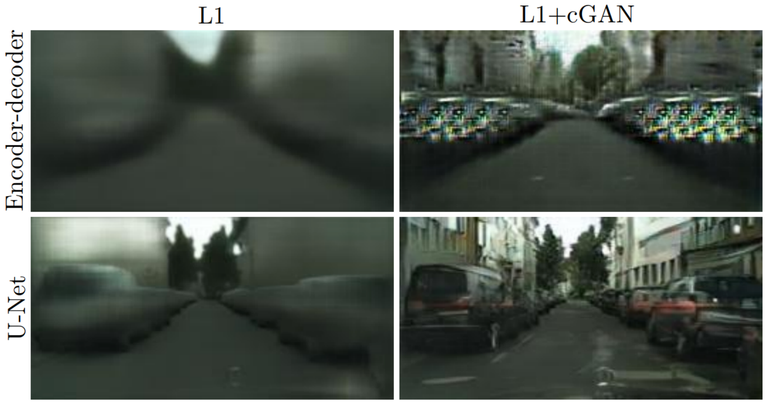
Evaluation

Evaluation

- human perceptual study
  - assessors had to discriminate between real and generated images.
- segmantation match
  - applies only to photographic Y
  - true segmentation map s for y is known
  - predict segmentation map $\widehat{s} = segm(G(x, z))$
    - FCN-8s was used as $segm(\cdot)$
  - quality of generator: $\|\widehat{s} - s\|$

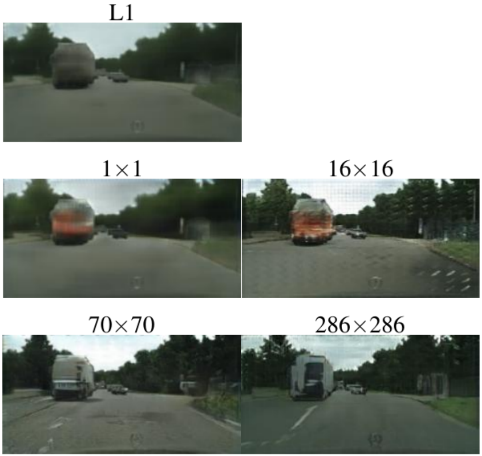
Results comparison

L1+cGAN better than L1 only, Unet better than encoder-decoder architecture.



Results comparison

L1 loss vs L1+cGAN with different patch size.



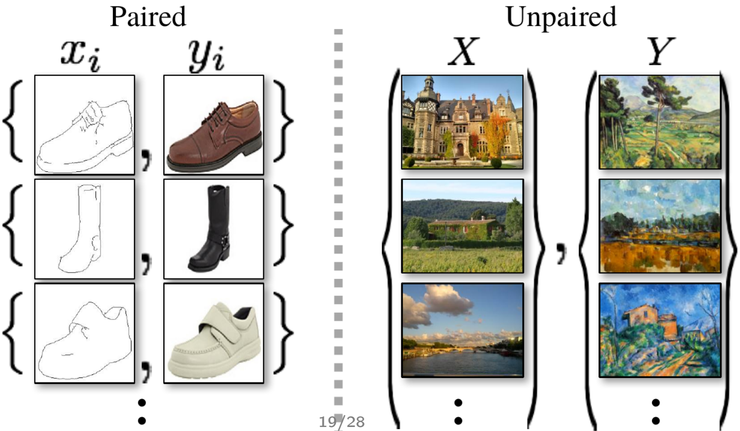
Results comparison

Generator fooled assessors on photorelistic output more than on simple label output.

Table of Contents

CycleGAN[3]

- 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

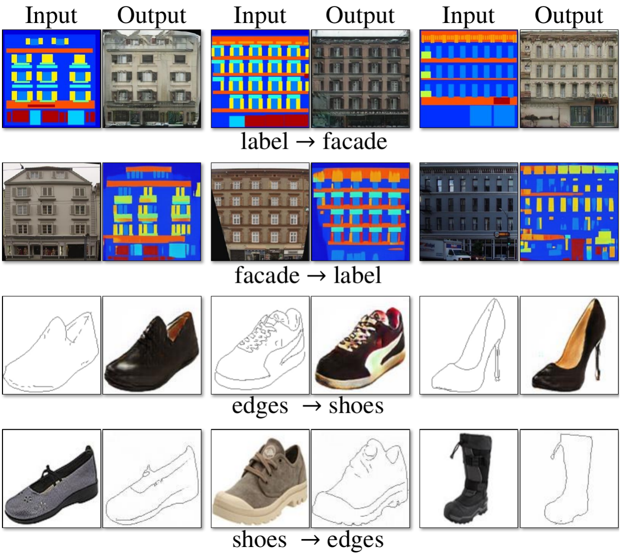
$$\begin{aligned}
\mathcal{L}_{GAN}(G_{XY}, D_Y) =& \mathbb{E}_{y\sim p(y)}\left[\log\left(D_Y(y)\right)\right] \\
&+ \mathbb{E}_{x\sim p(x)}\left[\log\left(1 - D_Y(G_{XY}(x))\right)\right] \\
\mathcal{L}_{GAN}(G_{YX}, D_X) =& \mathbb{E}_{x\sim p(x)}\left[\log\left(D_X(x)\right)\right] \\
&+ \mathbb{E}_{y\sim p(y)}\left[\log\left(1 - D_X(G_{YX}(y))\right)\right] \\
\mathcal{L}_{cyc}\left(G_{XY}, G_{YX}\right) =& \mathbb{E}_{x\sim p(x)}\left\|G_{YX}(G_{XY}(x)) - x\right\|_1 \\
&+ \mathbb{E}_{y\sim p(y)}\left\|G_{XY}(G_{YX}(y)) - y\right\|_1
\end{aligned}$$

CycleGAN

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

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

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.

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

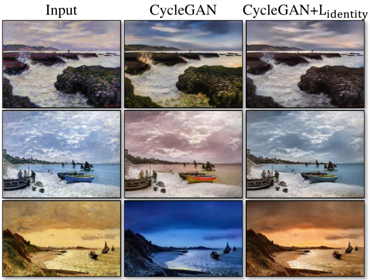


Table of Contents

Progressive GAN[4]

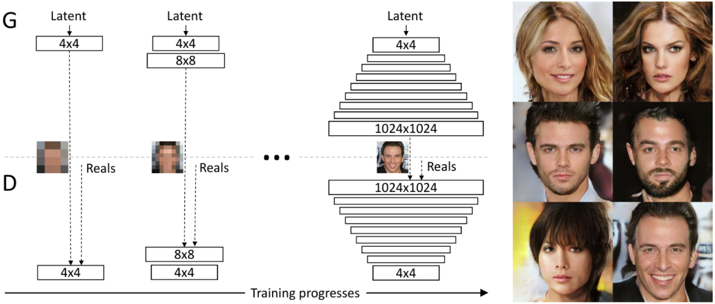
- Techique to generate high dimensional output.

Generated photographs, using celebreties dataset

[4] 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).

Conclusion

- GANs - method to generate new realistic samples based on $x_1, ...x_N$.
    - requires tuning to syncronize D and G training
    - suffers from mode collapse
- DCGAN - GAN designed for images generation.
- Wasserstein GAN works better when domains $p(x)$, $q_\theta(x)$ have low intersection.
- 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.