FORNAX

Generative Adversarial Networks Overview and applications

- Rafał Cycoń, FORNAX

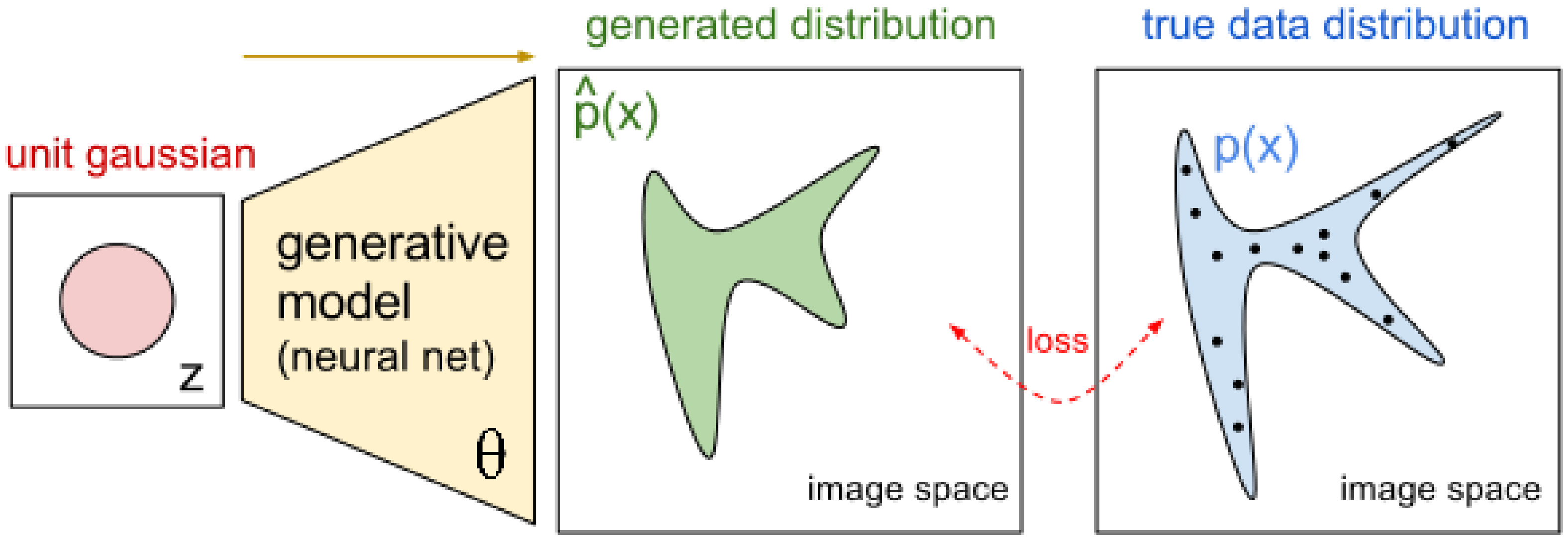
WWW.FORNAX.CO

How to approximate the true data distribution?

What for?

- More data – improve supervised and RL
- Fill in missing parts
- Better understanding of the data
- …?

How to approximate the true data distribution?

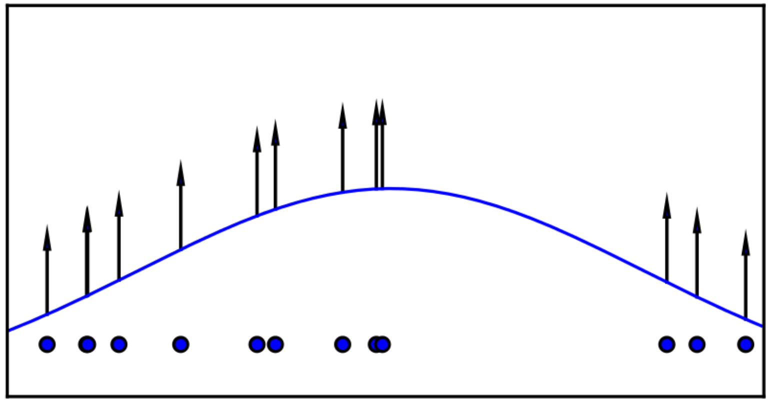


How to approximate the true data distribution?

Maximize log likelihood

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right).$$



Minimize KL divergence

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} D_{\text{KL}}\left(p_{\text{data}}(\boldsymbol{x}) \| p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})\right)$$

Money counterfeiting

Generative Adversarial Network

Notation:

x – sample from the data distribution p_{data}

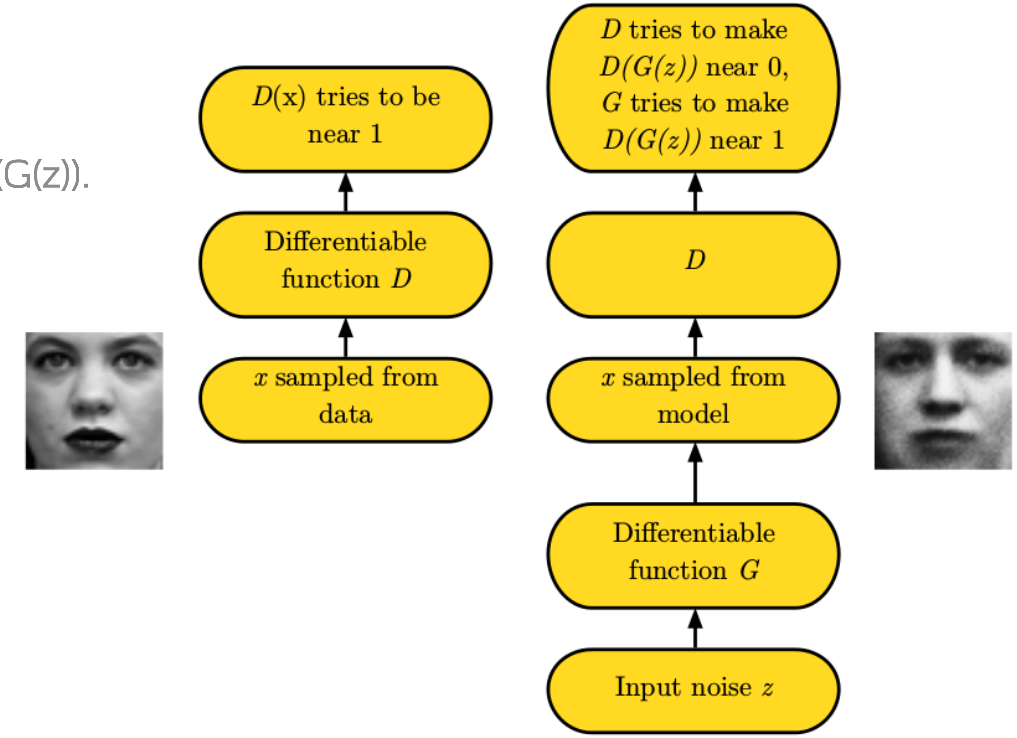
z – sample from an arbitrary (noise) distribution p_z

D – discriminator ("the police"), takes in x and z, outputs prob real vs. fake (targets: 0 – fake/generated, 1 – real)

G – generator ("the counterfeiter"), takes in z, converts it to data space

Pseudo-algorithm:

1. Generate same number of real and noise samples, get D(x) and D(G(z)).
2. Update the discriminator.
3. Generate more samples from the noise distribution, get D(G(z)).
4. Update the generator while keeping discriminator's params fixed.



Adversarial loss

$$J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right).$$

Zero-sum game:

$$J^{(G)} = -J^{(D)}.$$

$$V\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right) = -J^{(D)}\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right).$$

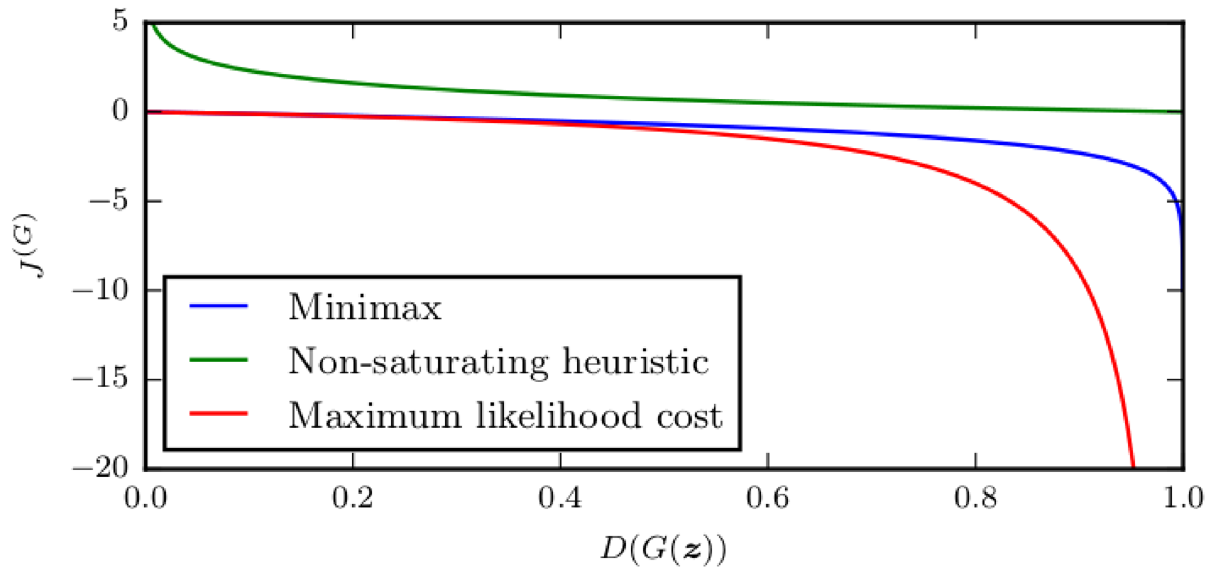
$$\boldsymbol{\theta}^{(G)*} = \underset{\boldsymbol{\theta}^{(G)}}{\arg\min}\, \max_{\boldsymbol{\theta}^{(D)}} V\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right).$$

Heuristically motivated:

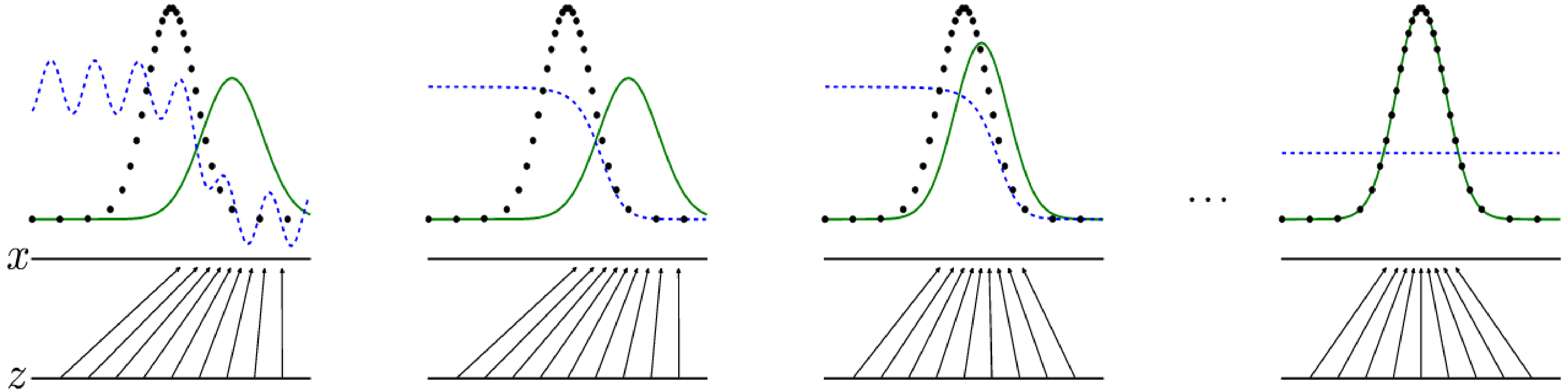
$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D(G(\boldsymbol{z}))$$

Maximum likelihood:

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{z} \exp\left(\sigma^{-1}\left(D(G(\boldsymbol{z}))\right)\right),$$



Learning process



(a) (b) (c) (d)

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

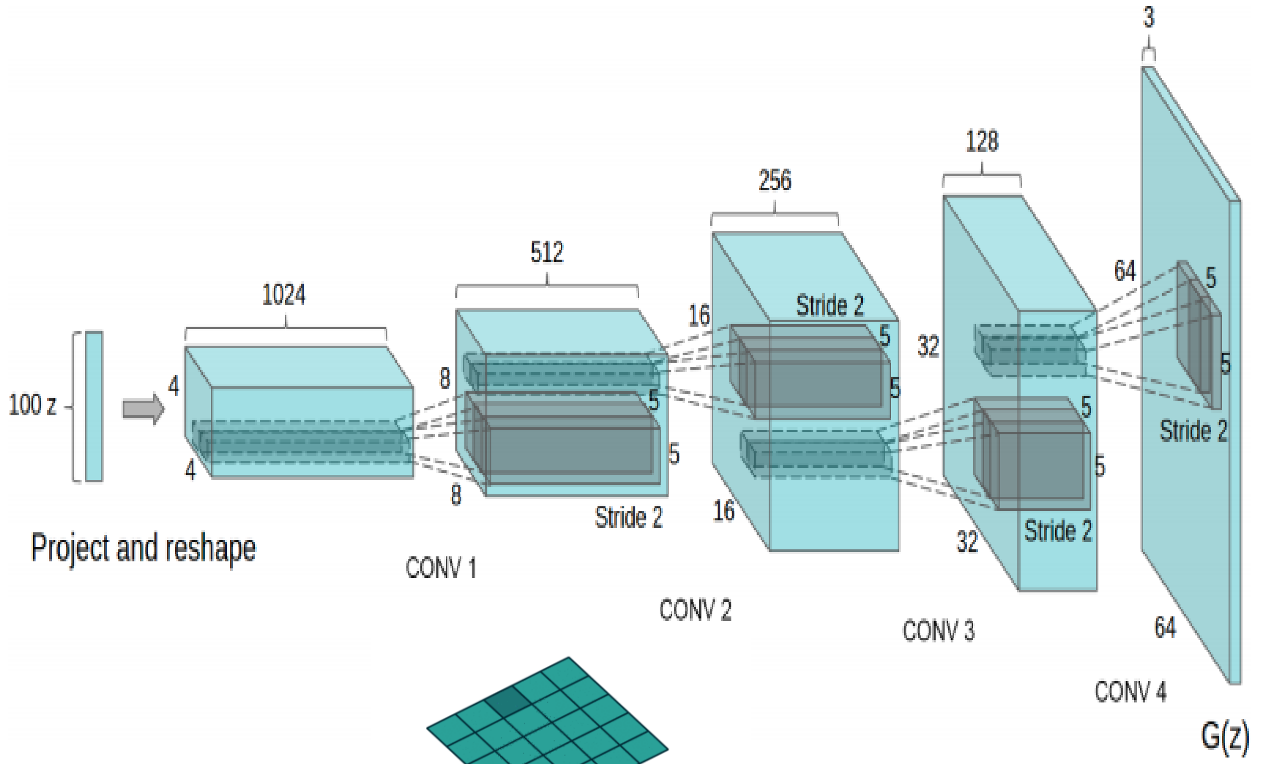
GAN (2014)

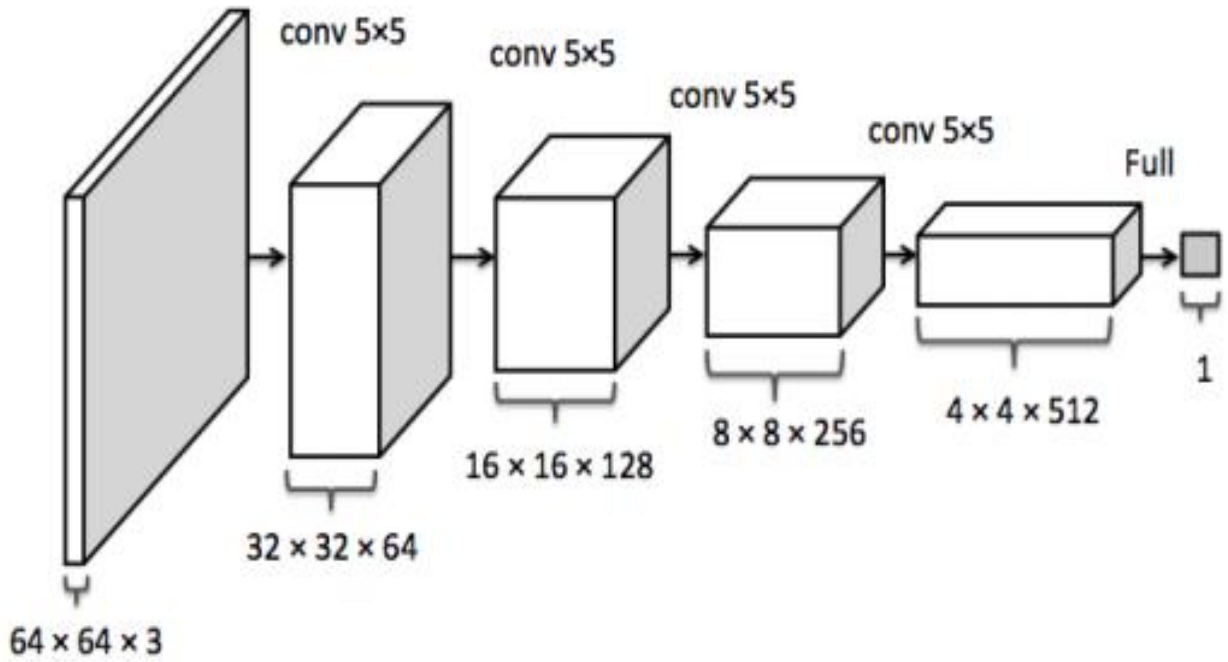
https://arxiv.org/abs/1406.2661

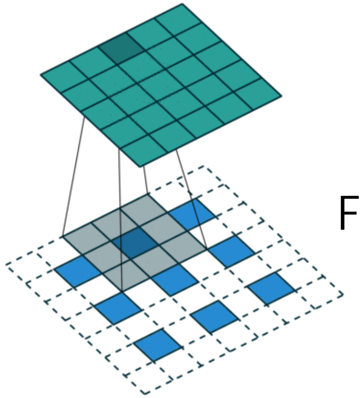
low resolution: 32x32
MNIST, TFD, CIFAR-10

DCGAN (2016)

http://arxiv.org/abs/1511.06434





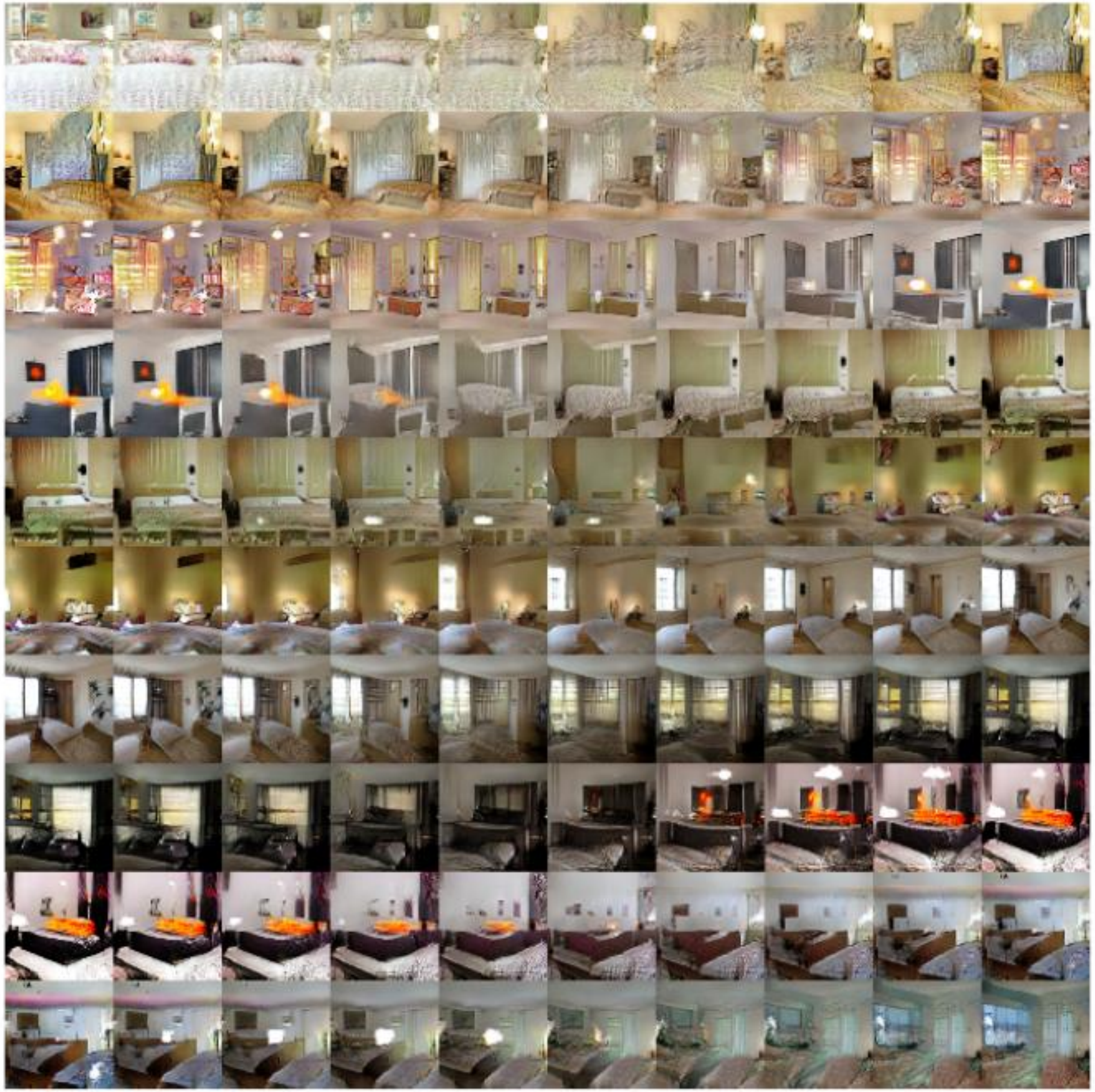


Fractionally-strided convolution

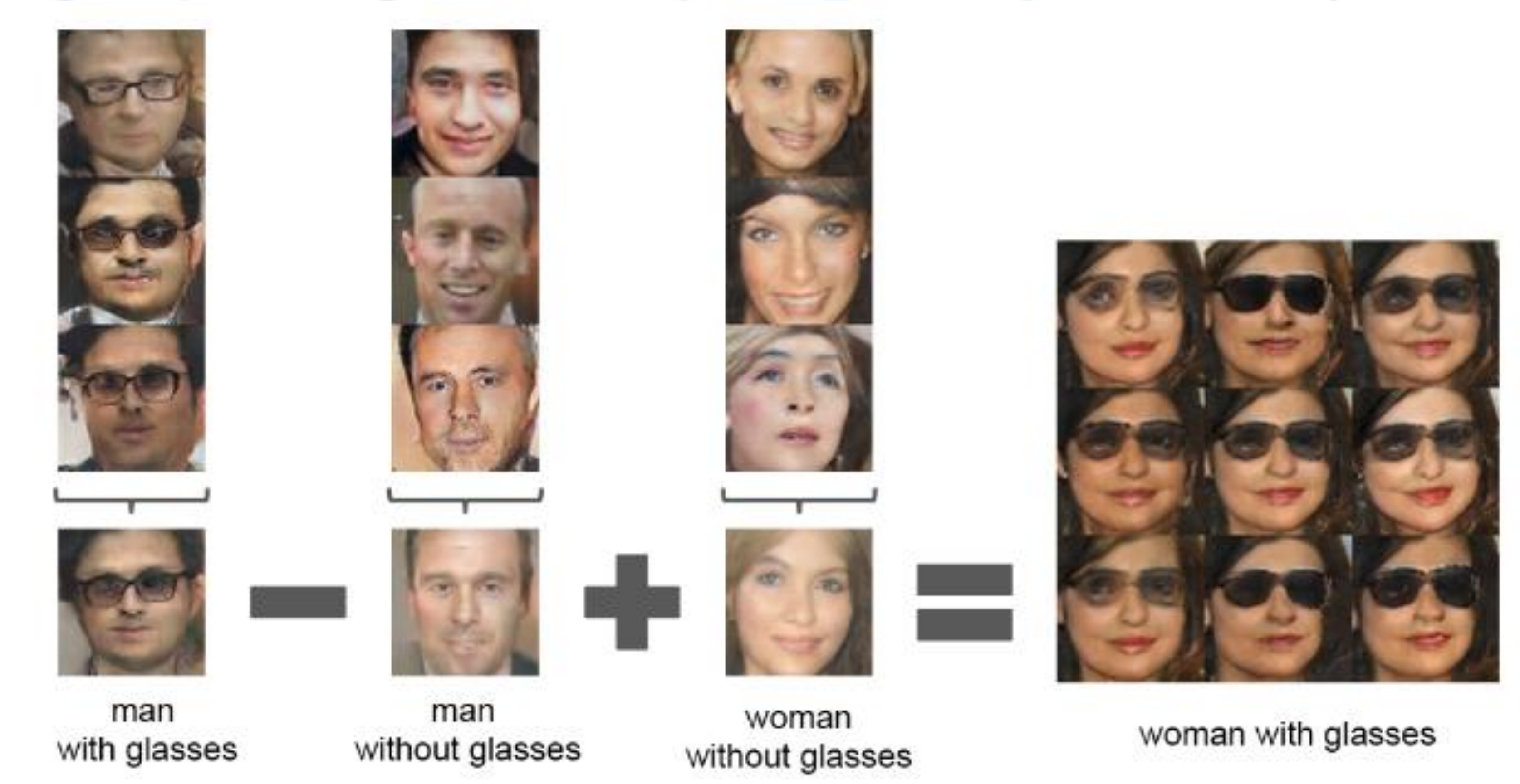
DCGAN (2016)

http://arxiv.org/abs/1511.06434

- No pooling (strides, the net can learn up/down-sampling by itself)
- No dense layers (full conv)
- G: ReLU, (Tanh on the last layer)
- D: LeakyReLU
- Batch Normalization (it was observed that BN helps the Generator to “get started”)
- Use Adam instead of SGD



Arithmetic operations



Tips & Tricks

https://arxiv.org/abs/1606.03498 (2016)

Train with labels

Train the discriminator to also classify the input image of real objects.

Why does it work? No one knows...

One-sided label smoothing

Discriminator predicts soft-probabilities of the real object class.

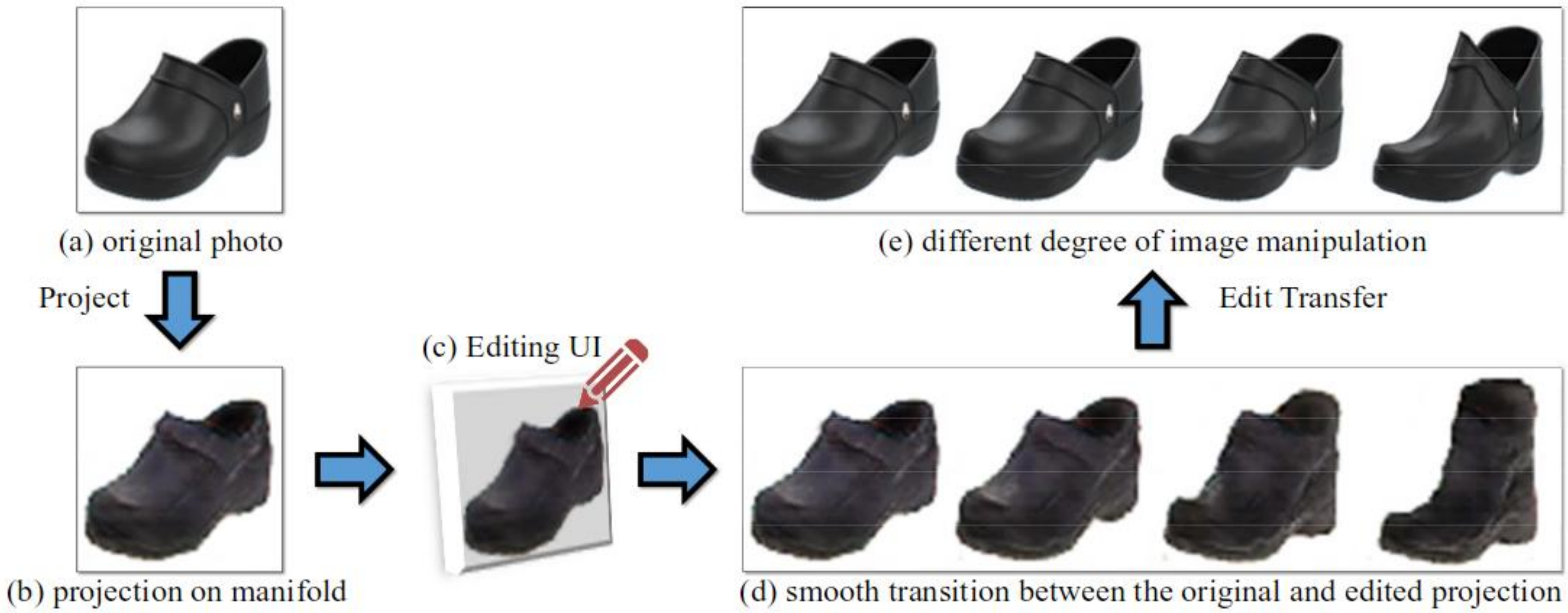
Virtual batch normalization

More tips&tricks:

https://github.com/soumith/ganhacks

Manipulation on the natural image manifold

https://arxiv.org/pdf/1609.03552v2.pdf



Manipulation on the natural image manifold

https://arxiv.org/pdf/1609.03552v2.pdf

Projection to the image manifold, i.e. "find z such that G(z) is most similar to x"

$$z^* = \arg\min_{z \in \tilde{\mathbb{Z}}} \mathcal{L}(G(z), x^R). \qquad \mathcal{L}(x_1, x_2) = ||\mathcal{C}(x_1) - \mathcal{C}(x_2)||^2$$

1. Optimization
   Establish C() as a combination of raw pixels and conv4 features from AlexNet. Optimize above eq. to look for z* (L-BFGS-B).
   Problems: cascade of C(G(z)) is highly non-convex, more than 100 initializations required to obtain a relatively stable solution.
2. Neural net
   Try to predict z given x with a feedforward neural network model.

$$\theta_P^* = \arg\min_{\theta_P} \sum_n \mathcal{L}(G(P(x_n^R; \theta_P)), x_n^R),$$

   Stable, but solution can be further improved.
3. Hybrid: initialize with neural net, then optimize for a few steps.

Manipulation on the natural image manifold

https://arxiv.org/pdf/1609.03552v2.pdf

Manipulating the latent space

$$z^* = \arg\min_{z \in \mathbb{Z}} \Big\{ \underbrace{\sum_g \|f_g(G(z)) - v_g\|^2}_{\text{data term}} + \underbrace{\lambda_s \cdot \|z - z_0\|^2}_{\text{manifold smoothness}} + E_D \Big\}.$$

v_g – constraint

manifold smoothness – make sure the image is not altered too much

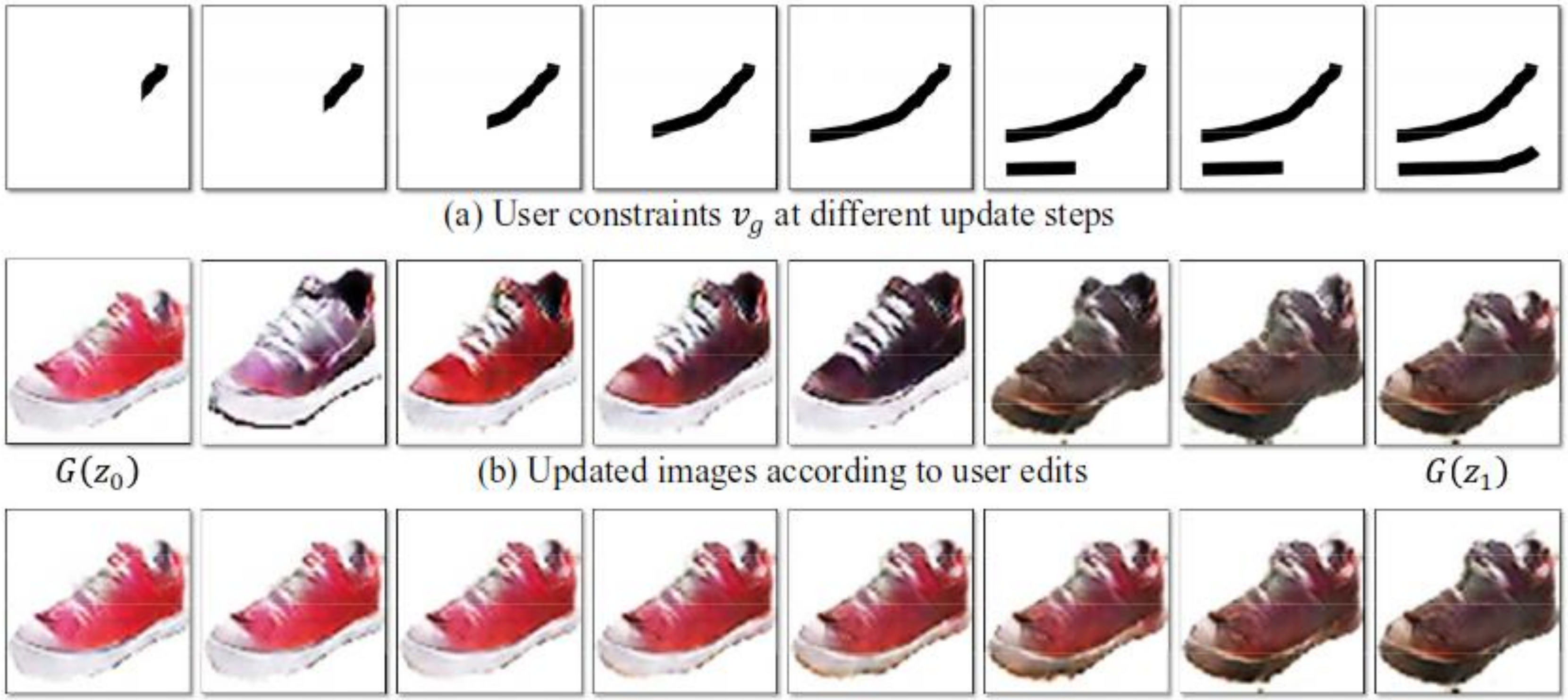
E_D – improve the realism of the image with the help of the discriminator: $E_D = \lambda_D \cdot \log(1 - D(G(z)))$

Constraints:

- coloring brush: constrain the color in of a pixel at a given location
- sketching brush: constrain the shape or add details. Uses a differentiable HOG descriptor at a given location to be close to the user's stroke.
- warping brush: impose above constrains on a local region so that it mimics the source region.

Manipulation on the natural image manifold

https://arxiv.org/pdf/1609.03552v2.pdf



(c) Linear interpolation between $G(z_0)$ and $G(z_1)$

Manipulation on the natural image manifold

https://arxiv.org/pdf/1609.03552v2.pdf

G(z) has a degraded quality w.r.t. image manifold (lower resolution)

Task: transfer the changes from the approximated image manifold to the natural image manifold ("edit transfer")

Solution: direct transfer of pixel changes

$$x_1^R = x_0^R + (G(z_1) - G(z_0))$$

but it introduces artifacts due to misalignments...

Better solution: generate a series of intermediate frames

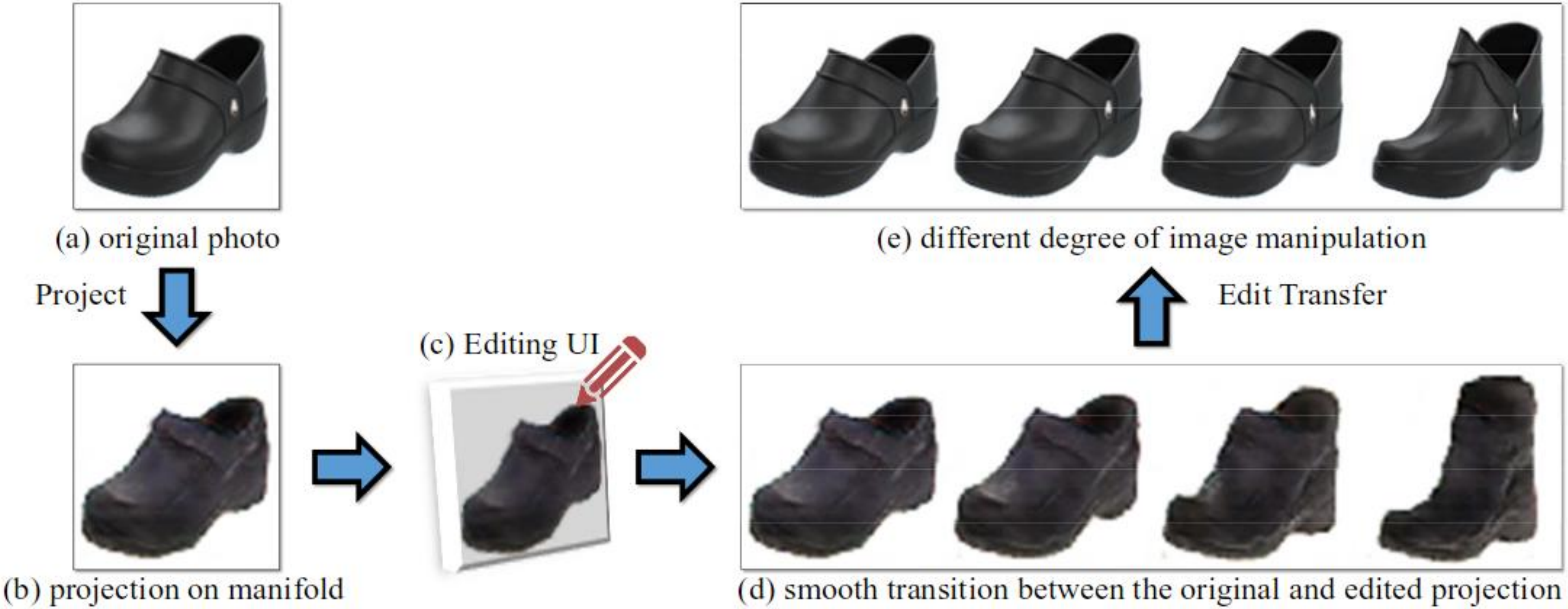
$$\left[G\left(\left(1-\frac{t}{N}\right)\cdot z_0 + \frac{t}{N}\cdot z_1\right)\right]_{t=0}^{N}$$

then calculate optical flow (motion+color) in approx. manifold, upsample it, and apply it to the natural image manifold

$$\iint \underbrace{\|I(x,y,t) - A\cdot I(x+u, y+v, t+1)\|^2}_{\text{data term}} + \underbrace{\sigma_s(\|\nabla u\|^2 + \|\nabla v\|^2)}_{\text{spatial reg}} + \underbrace{\sigma_c\|\nabla A\|^2}_{\text{color reg}} dx\,dy,$$

Manipulation on the natural image manifold

https://arxiv.org/pdf/1609.03552v2.pdf



Manipulation on the natural image manifold

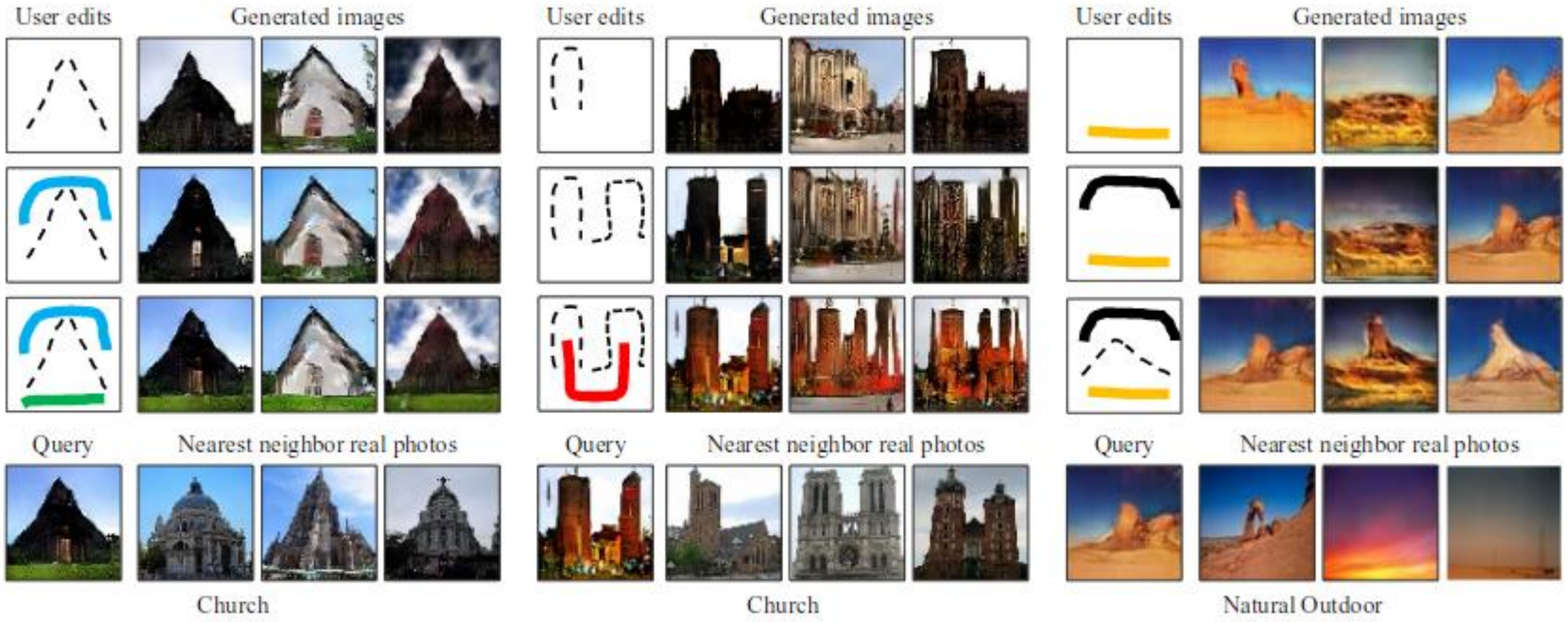
https://arxiv.org/pdf/1609.03552v2.pdf

Transform one natural image into the other

Manipulation on the natural image manifold

https://arxiv.org/pdf/1609.03552v2.pdf

Apply user edits without any images – algorithm produces images that best satisfy the constrains



Manipulation on the natural image manifold

https://arxiv.org/pdf/1609.03552v2.pdf

- Runs in real time
- Results tested on Amazon Mechanical Turk (is the image realistic or not?; 400 images):
  - real photos: 91.5%
  - DCGAN: 14.3%
  - shape+color: 25.9%
  - shape: 48.7%

most likely due to edit transfer

Image to image translation

https://arxiv.org/pdf/1611.07004v1.pdf

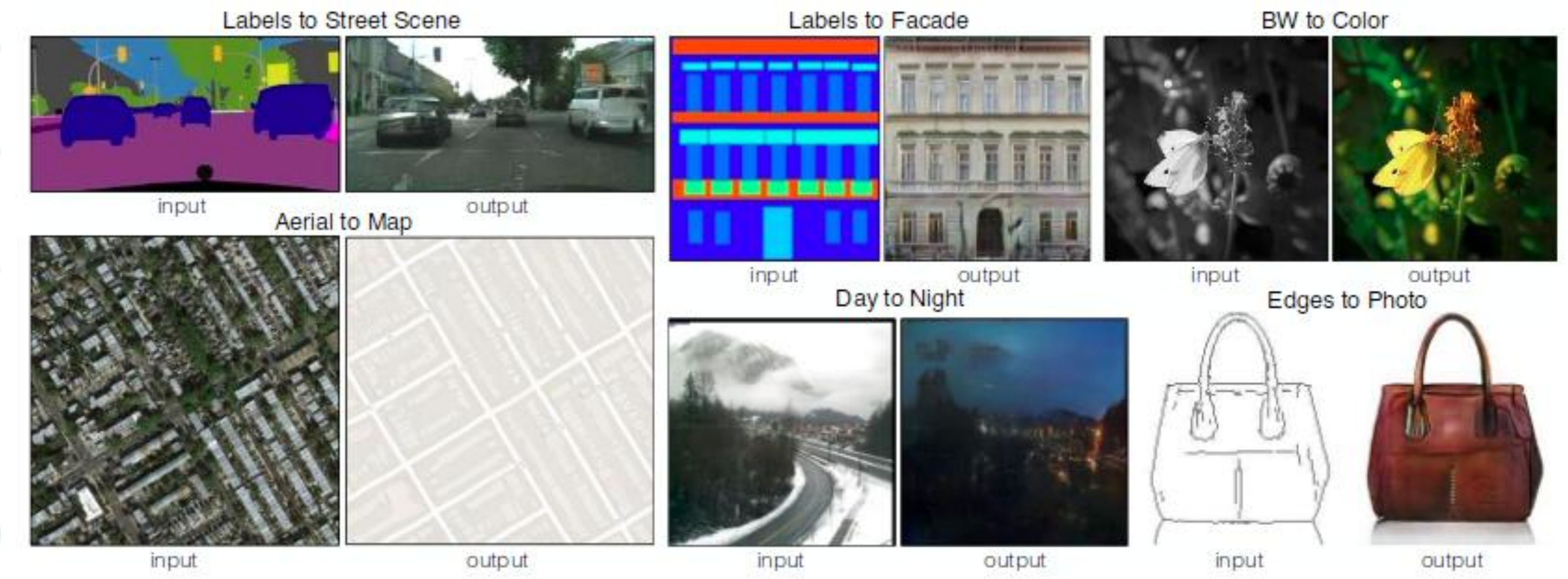


Image to image translation

https://arxiv.org/pdf/1611.07004v1.pdf

Condition GANs on an input image

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

Additionally help the generator by forcing it to be near the ground truth in L1 sense:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y\sim p_{data}(x,y),z\sim p_z(z)}[\|y - G(x,z)\|_1]$$

Final objective is then:

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

z is added to produce stochastic output, but generator ignores it. Instead, dropout is applied during both training and testing times.

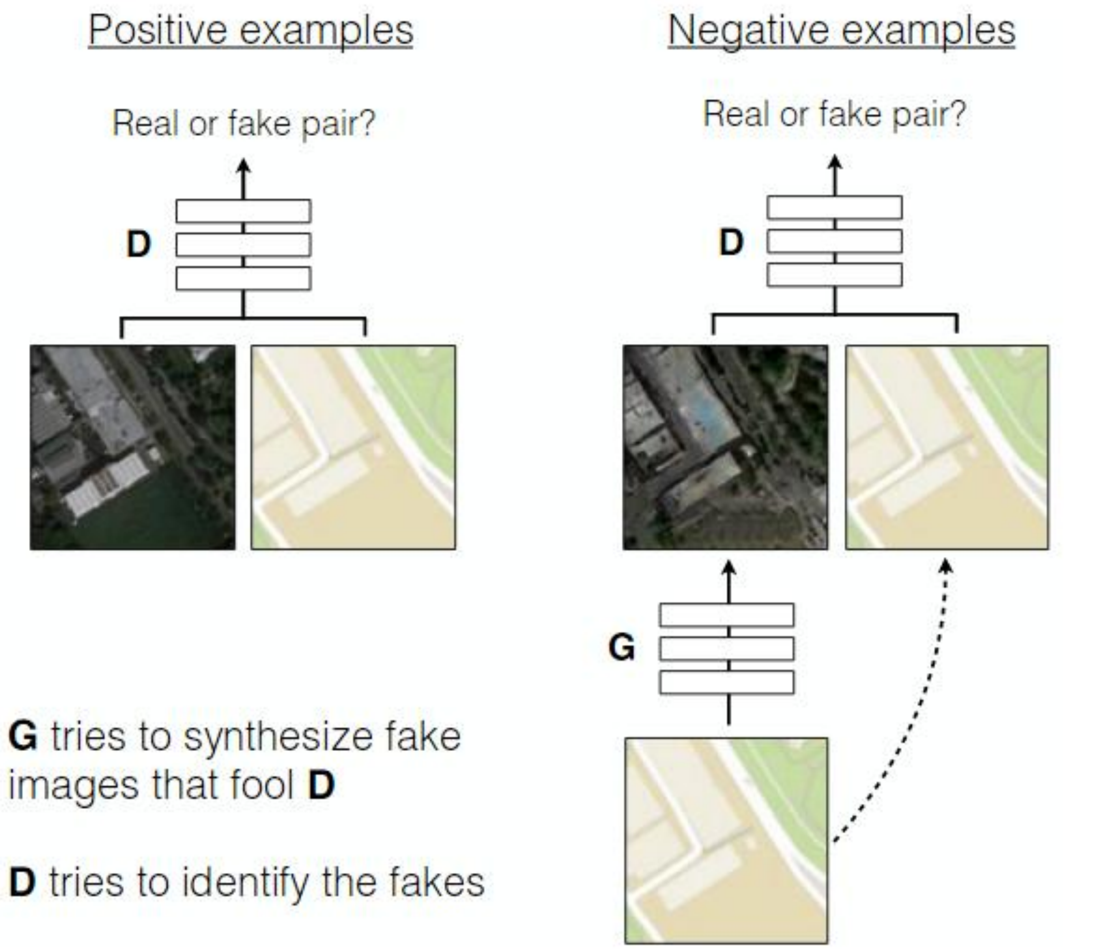


Image to image translation

https://arxiv.org/pdf/1611.07004v1.pdf

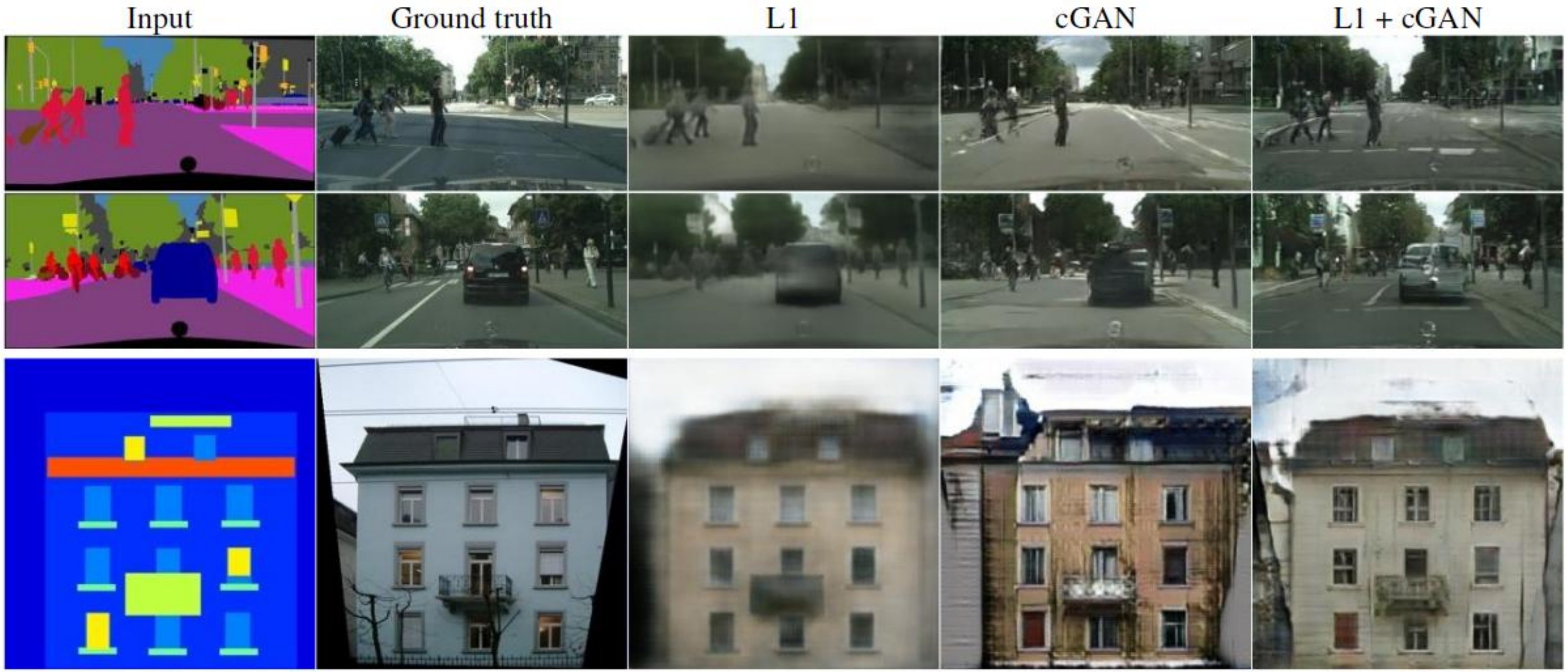


Image to image translation

https://arxiv.org/pdf/1611.07004v1.pdf

Adding skip connections, because:

- Mapping from a high-resolution to a high-resolution image.
- Structure in the output highly similar to structure in the input.

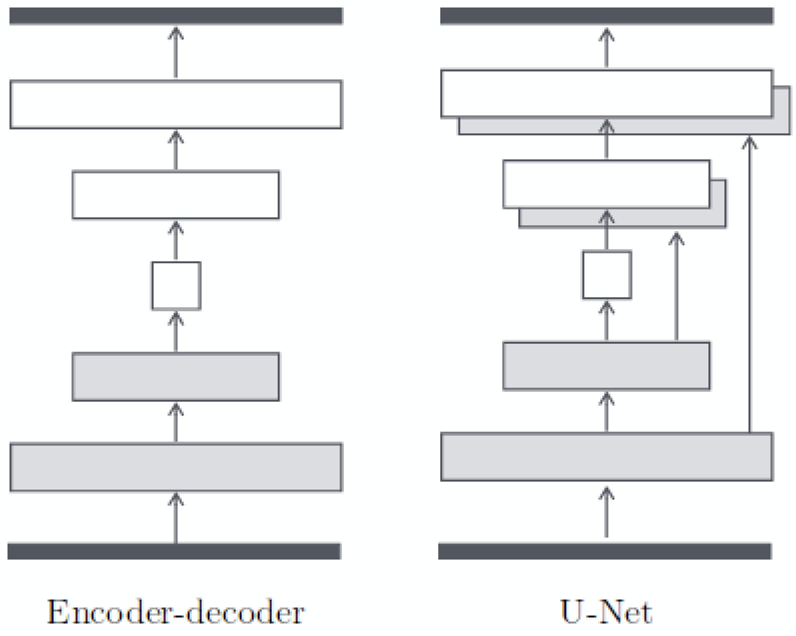


Image to image translation

https://arxiv.org/pdf/1611.07004v1.pdf

PatchGAN:

L1 loss forces low-frequency correctness.

Ensure high-frequency correctnes by focusing on local patches of the image.

Discriminator at its final layer convolves a NxN window over the image, averages all responses to produce final output.

1x1 – greater color diversity, no structural sharpness

16x16 – tiling artifacts

70x70 – sharp results, colorfull

256x256 (full res.) – visually similar to 70x70, lower in quality

Image to image translation

https://arxiv.org/pdf/1611.07004v1.pdf

Results tested on Amazon Mechanical Turk (is the image realistic or not?):

aerial map->photo: 18.9%

aerial photo->map: 6.1%

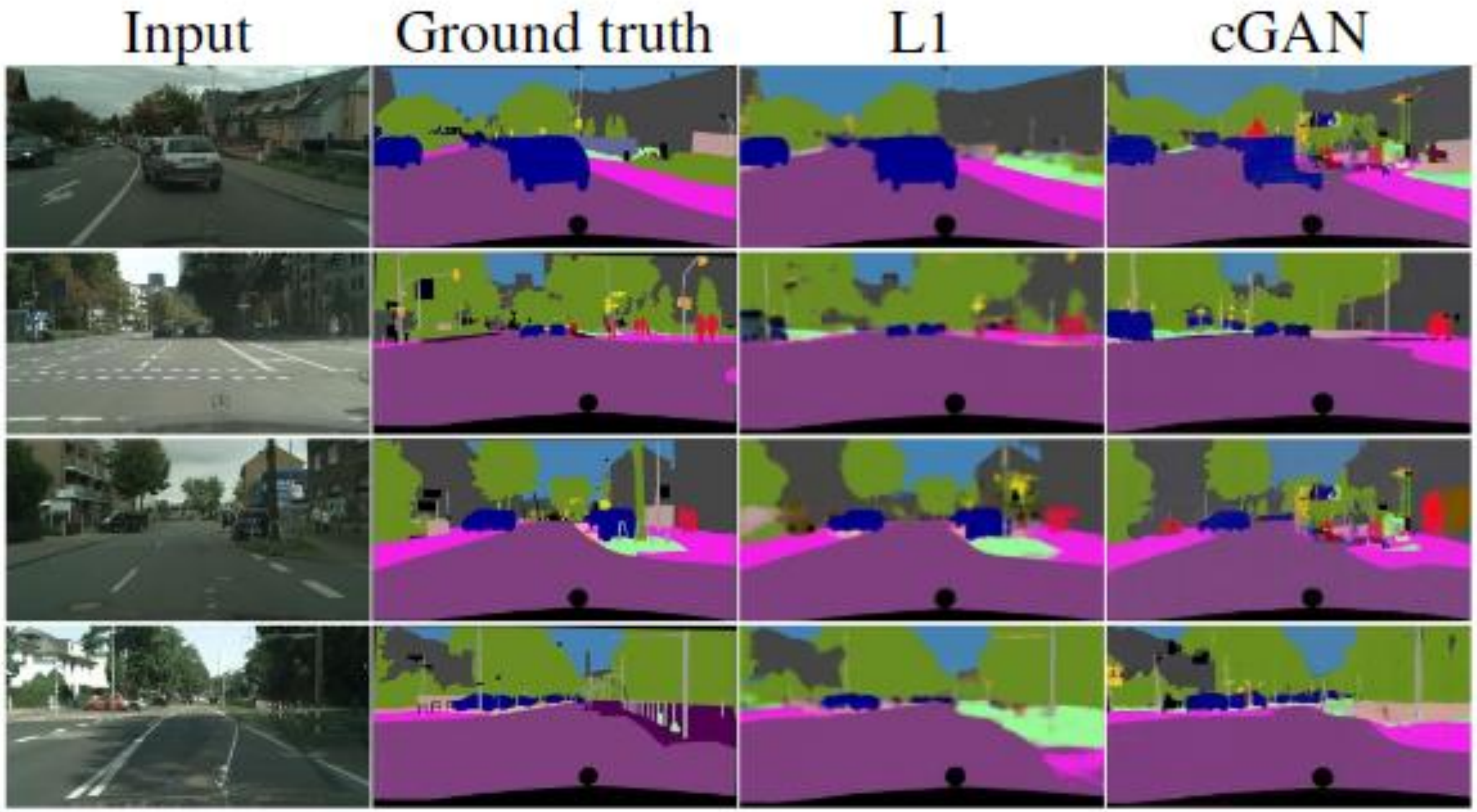
colorization: 22.5%

Image to image translation

https://arxiv.org/pdf/1611.07004v1.pdf

Image segmentation?
Output less complex than the input.

Produces many small, hallucinated objects



Inpainting

https://arxiv.org/abs/1607.07539

Inpainting

https://arxiv.org/abs/1607.07539

M – binary mask that marks the missing region pixels to 0, marks 1 elsewhere.

$\odot$ – Hadamard product, elementwise matrix multiplication.

$$x_{\text{reconstructed}} = M \odot y + (1 - M) \odot G(\hat{z})$$

the photo is close to the original content in the unmasked parts

$$\mathcal{L}_{\text{contextual}}(z) = ||M \odot G(z) - M \odot y||_1,$$

the photo overall looks realistic, as according to the discriminator

$$\mathcal{L}_{\text{perceptual}}(z) = \log(1 - D(G(z)))$$

$$\mathcal{L}(z) \equiv \mathcal{L}_{\text{contextual}}(z) + \lambda \mathcal{L}_{\text{perceptual}}(z)$$

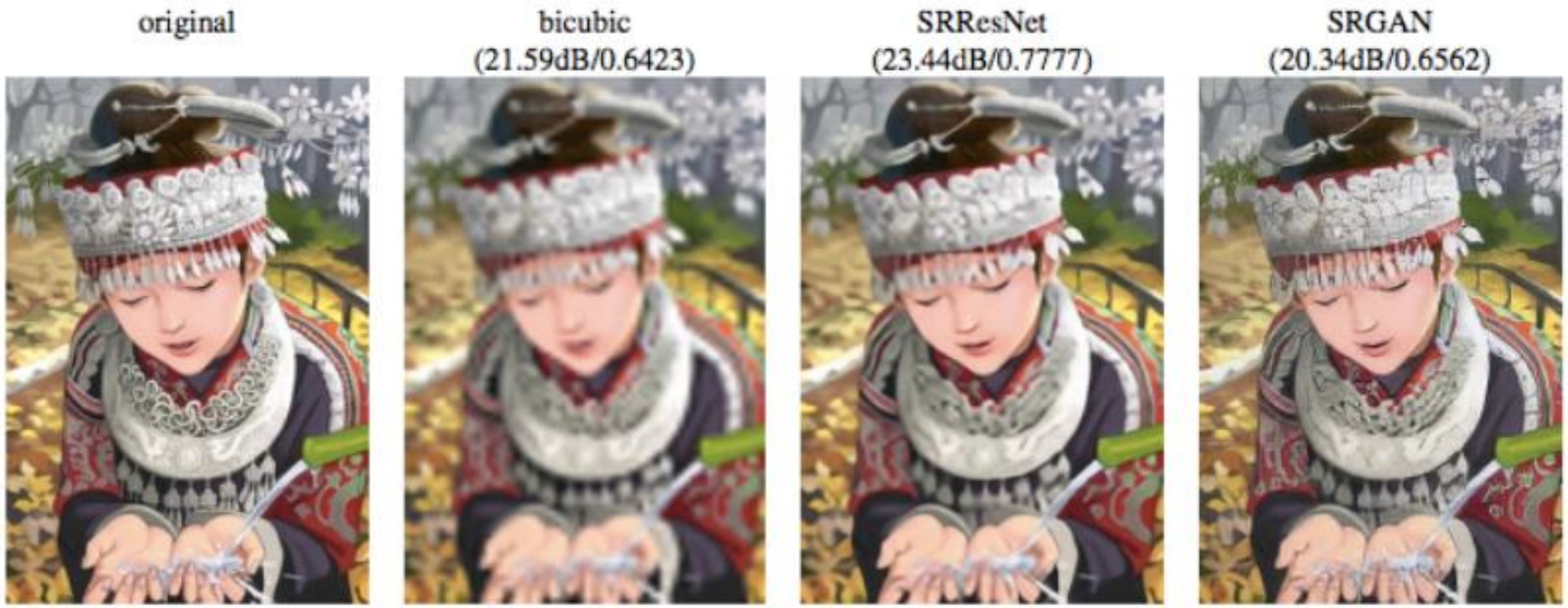
$$\hat{z} \equiv \arg\min_{z} \mathcal{L}(z)$$

Inpainting

https://arxiv.org/abs/1607.07539

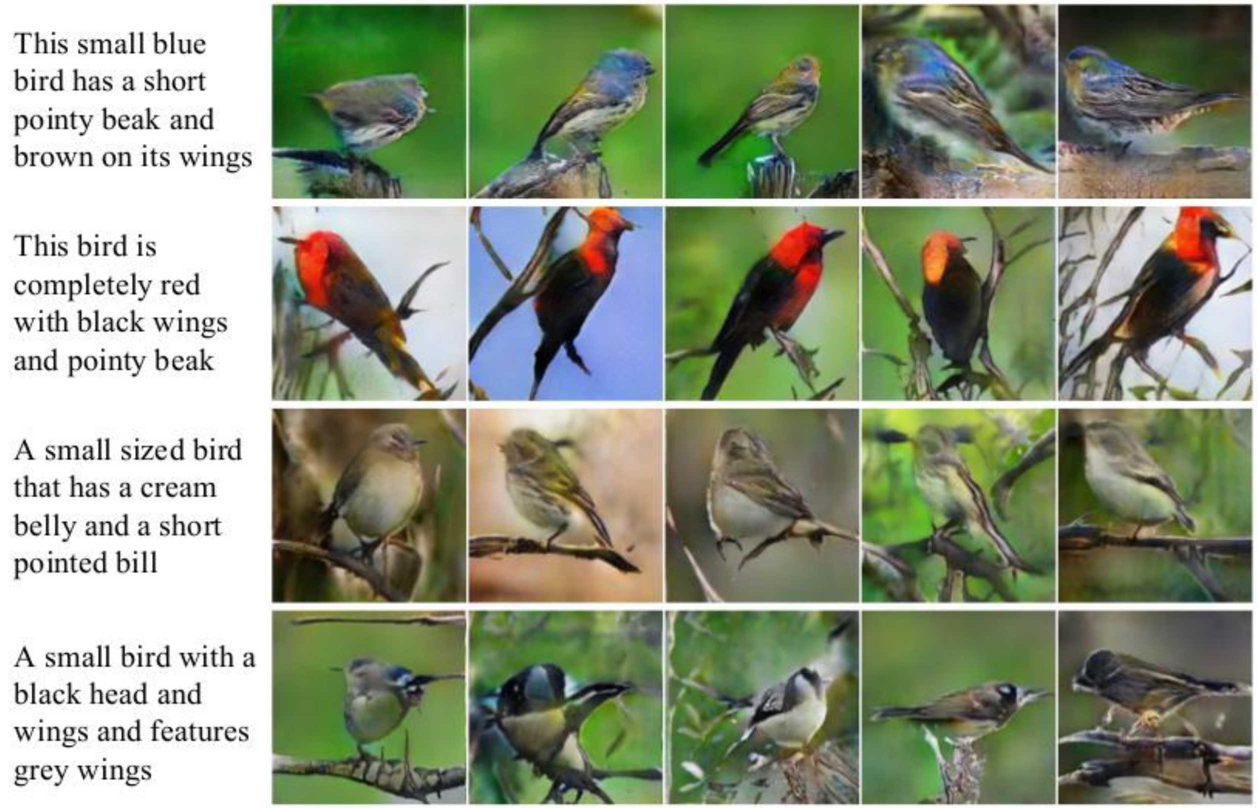
Super-resolution

https://arxiv.org/abs/1609.04802



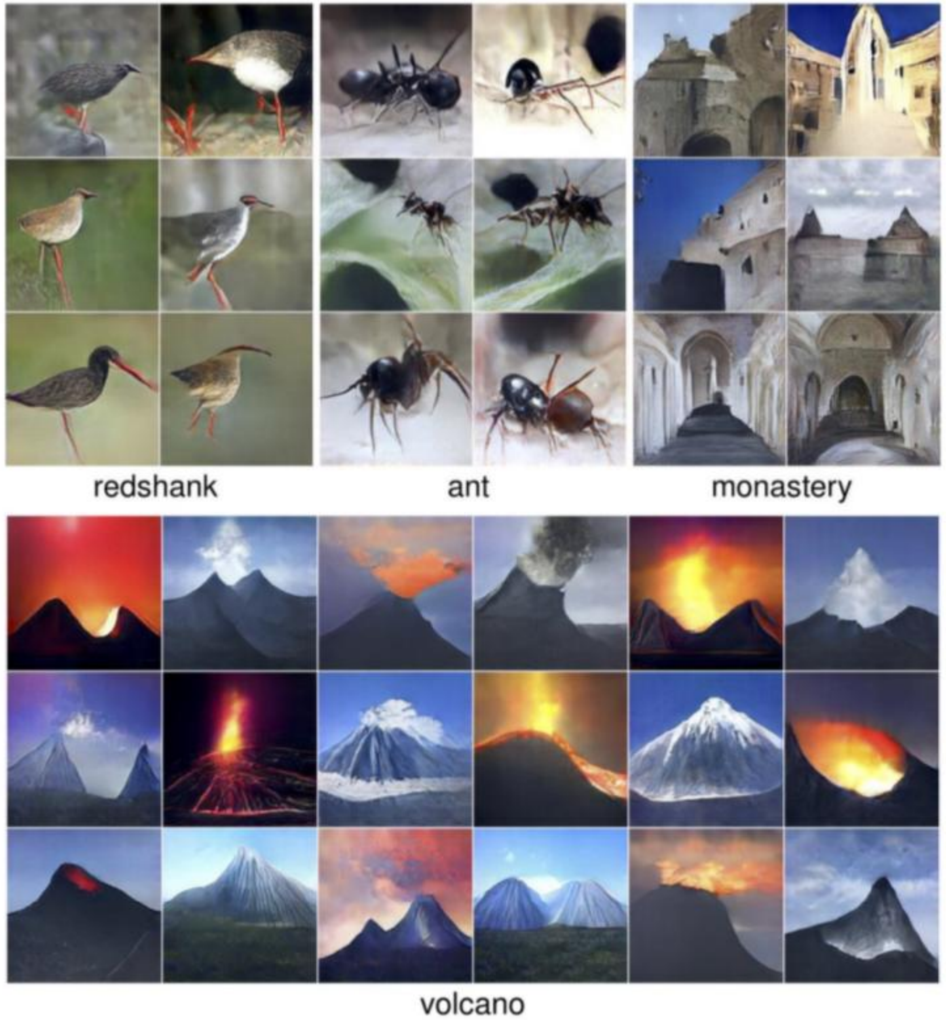
Text-to-image with GANs

e.g. StackGANs: https://arxiv.org/abs/1612.03242



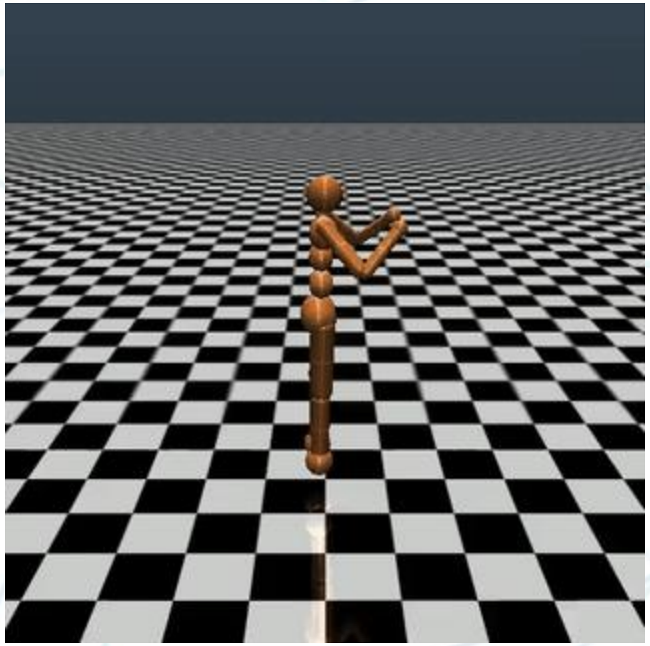
Plug and play GANs

https://arxiv.org/abs/1612.00005



Generative Adversarial Imitation Learning

http://arxiv.org/abs/1606.03476



Open problems

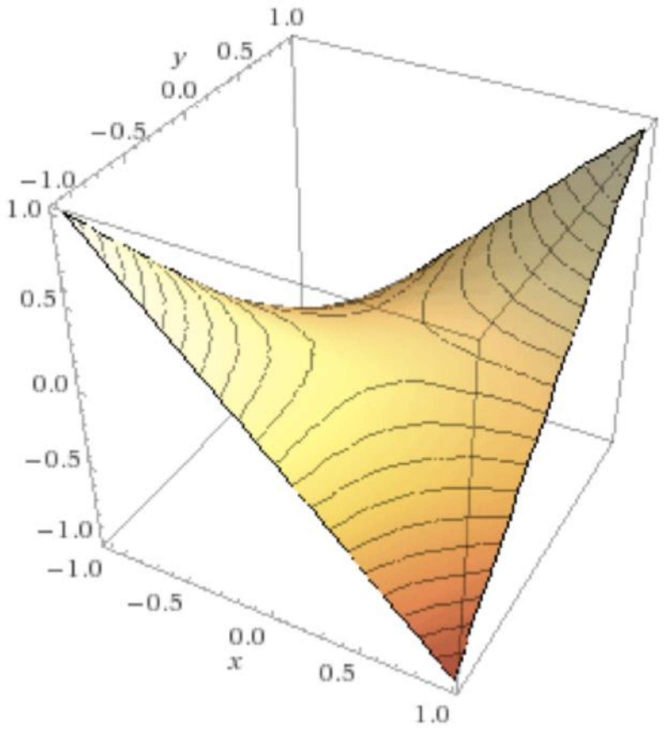
Non convergence

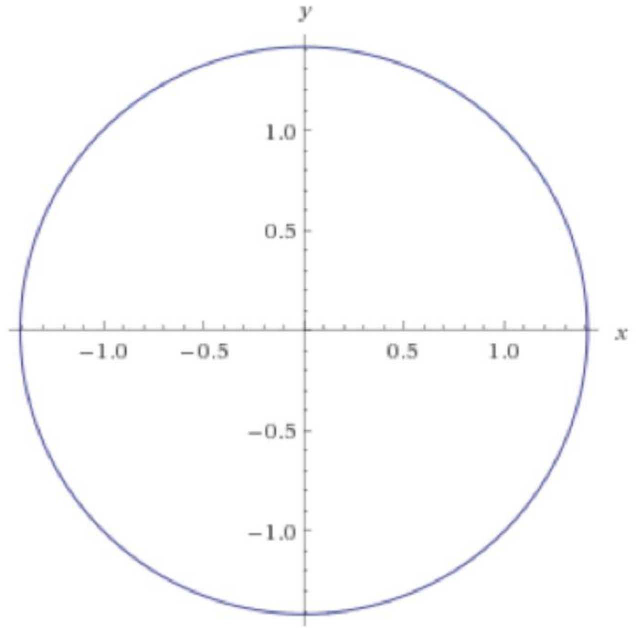
$$V(x, y) = xy$$

$$\frac{\partial x}{\partial t} = -y(t)$$
$$\frac{\partial y}{\partial t} = x(t).$$

$$\frac{\partial^2 y}{\partial t^2} = \frac{\partial x}{\partial t} = -y(t).$$

$$x(t) = x(0)\cos(t) - y(0)\sin(t)$$
$$y(t) = x(0)\sin(t) + y(0)\cos(t).$$





Open problems

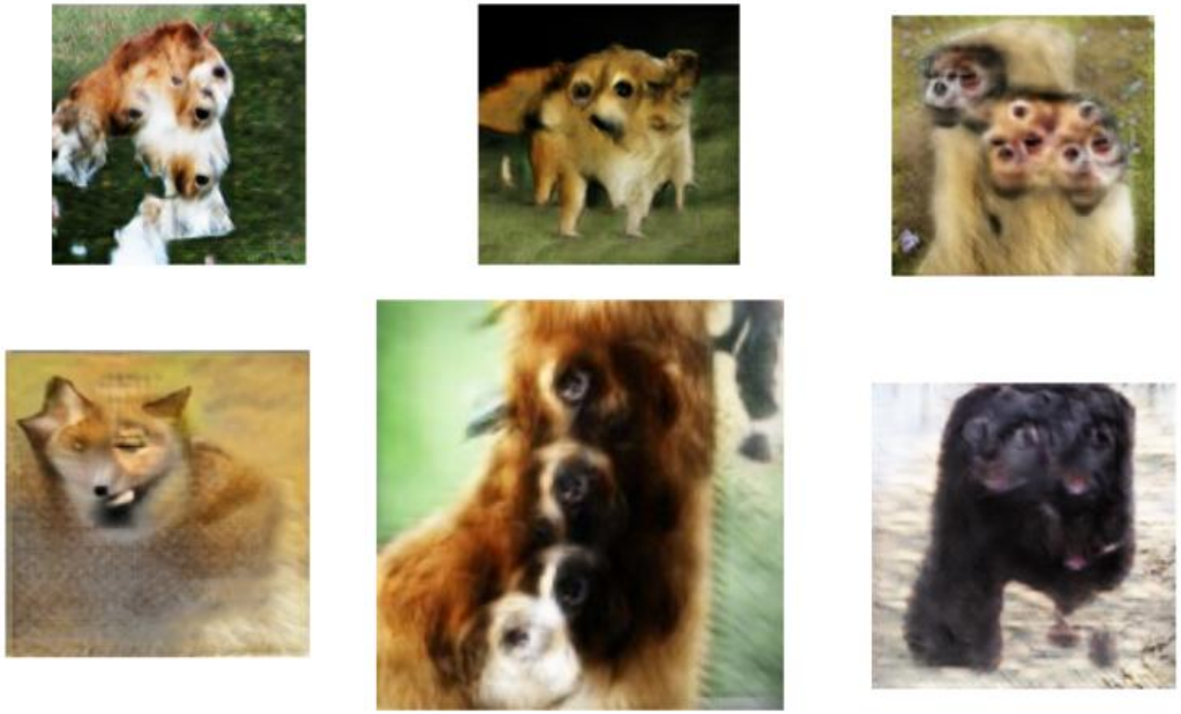
Mode collapse

Generator learns to produce the same output image, or very similar looking images.

Open problems

Low quality of high res. images

- Current optimization techniques are ineffective for GANs with too many parameters?
- *Solved by Plug and Play GANs ???*



References

"Generative Adversarial Nets", Goodfellow I. J. et al., https://arxiv.org/pdf/1406.2661v1.pdf
"NIPS 2016 Tutorial: Generative Adversarial Networks", Goodfellow I. J., https://arxiv.org/abs/1701.00160
"Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", Radford A. et al., https://arxiv.org/pdf/1511.06434v2.pdf
"Improved Techniques for Training GANs ", Salimans T. et al., https://arxiv.org/pdf/1606.03498v1.pdf
"Generative Visual Manipulation on the Natural Image Manifold", Jun-Yan Zhu et al., https://arxiv.org/abs/1609.03552
"Semantic Image Inpainting with Perceptual and Contextual Losses", Yeh R. et al., https://arxiv.org/abs/1607.07539
"Image-to-Image Translation with Conditional Adversarial Networks", Isola P. et al., https://arxiv.org/abs/1611.07004
"Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", Ledig C. et al., https://arxiv.org/abs/1609.04802
"StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks", Zhang Hao et al., https://arxiv.org/abs/1612.03242
"Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space", Anh Nguyen et al., https://arxiv.org/abs/1612.00005
"Generative Adversarial Imitation Learning", Jonathan Ho, Stefano Ermon, https://arxiv.org/abs/1606.03476

https://github.com/soumith/ganhacks

FORNAX

Rafał Cycoń
Co-founder, CTO

+48 607 697 054
rafal.cycon@fornax.co

WWW.FORNAX.CO