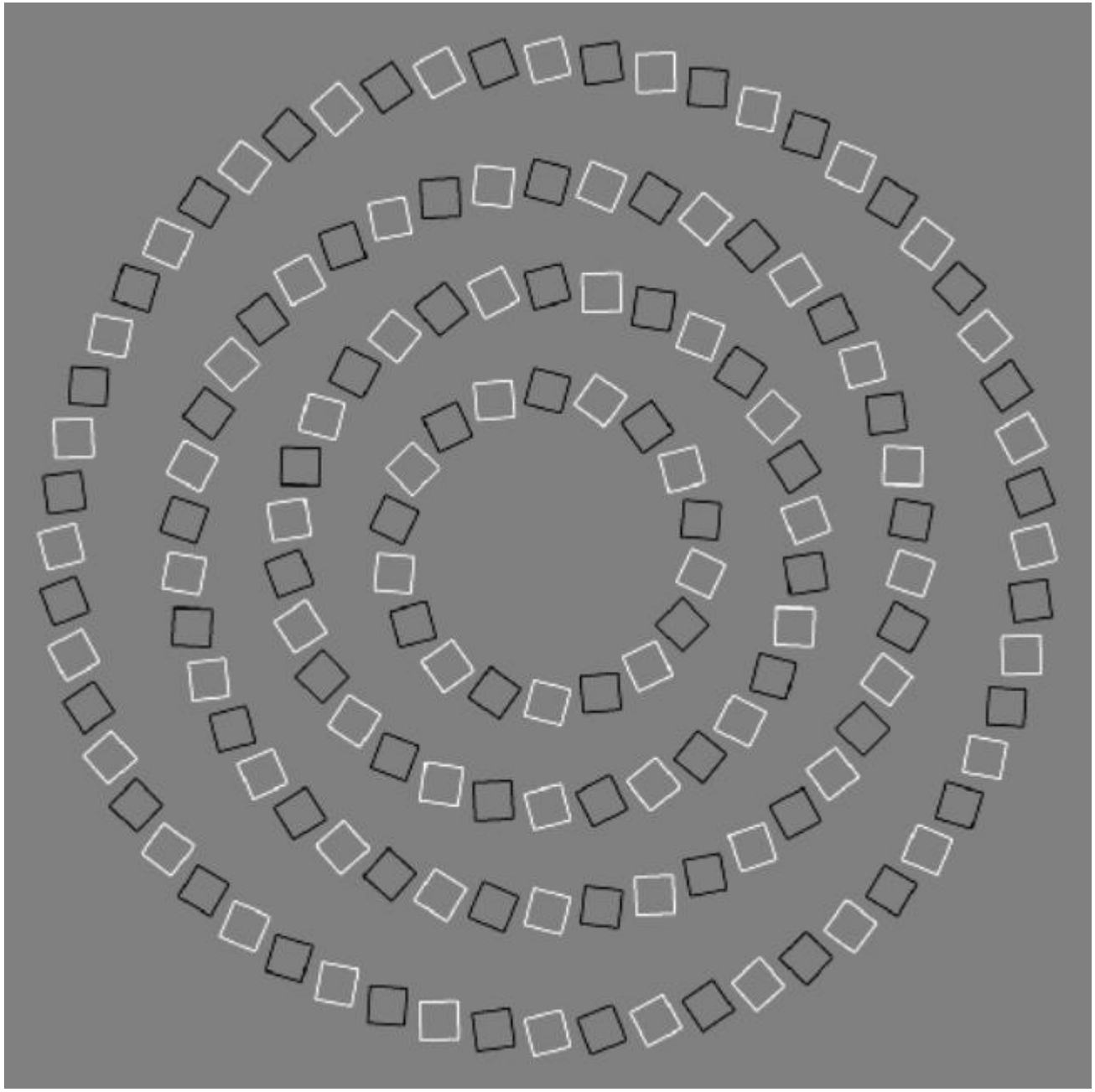
# Глубокое обучение
# Состязательные атаки и сети

**Дьяконов А.Г.**

**Московский государственный университет имени М.В. Ломоносова (Москва, Россия)**



# Состязательные атаки (Adversarial attacks)

- White Box (знаем, что атакуем)
- Black Box (не знаем, что атакуем)

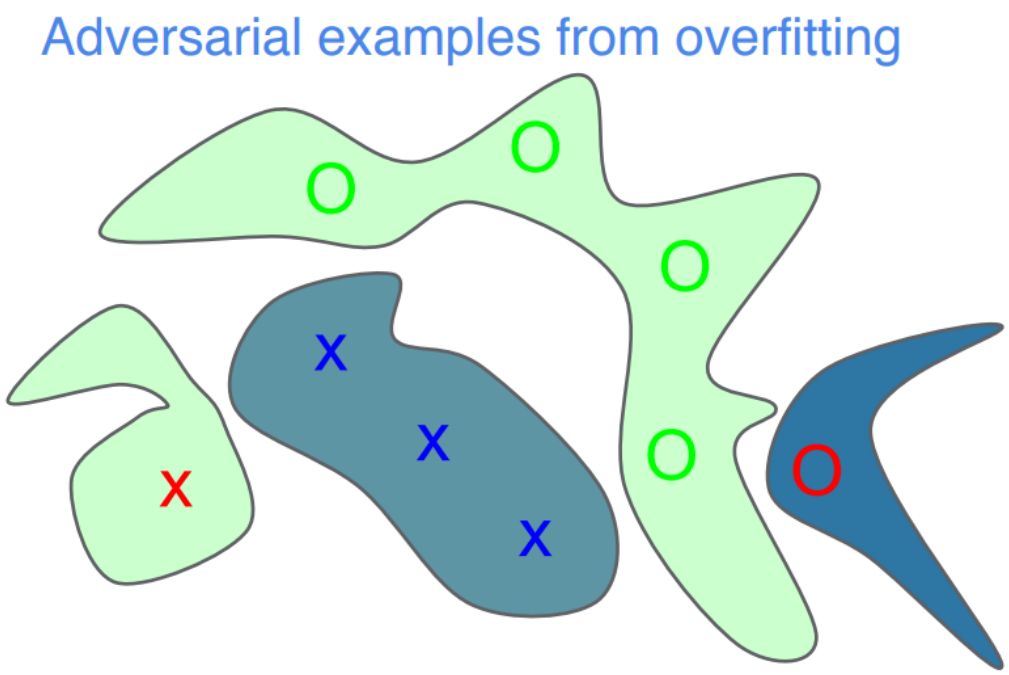
но можно попытаться промоделировать

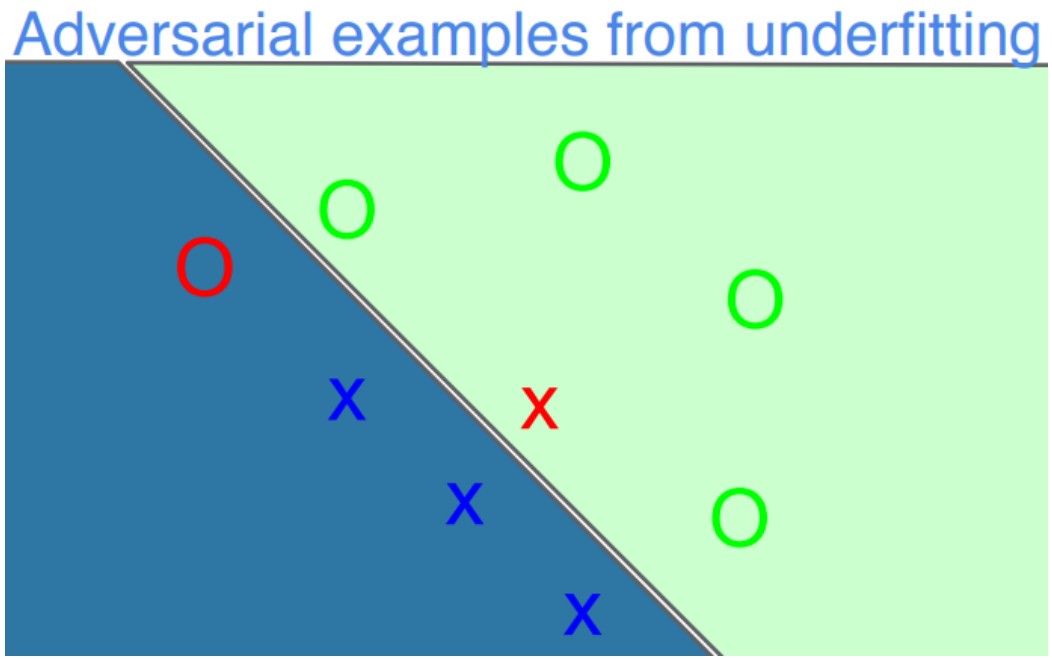
- non-targeted (цель – обмануть)
- targeted (цель – обмануть определённым образом)

Небольшая сложность

как формализовать «немного изменить объект»

# Состязательные примеры – Adversarial examples

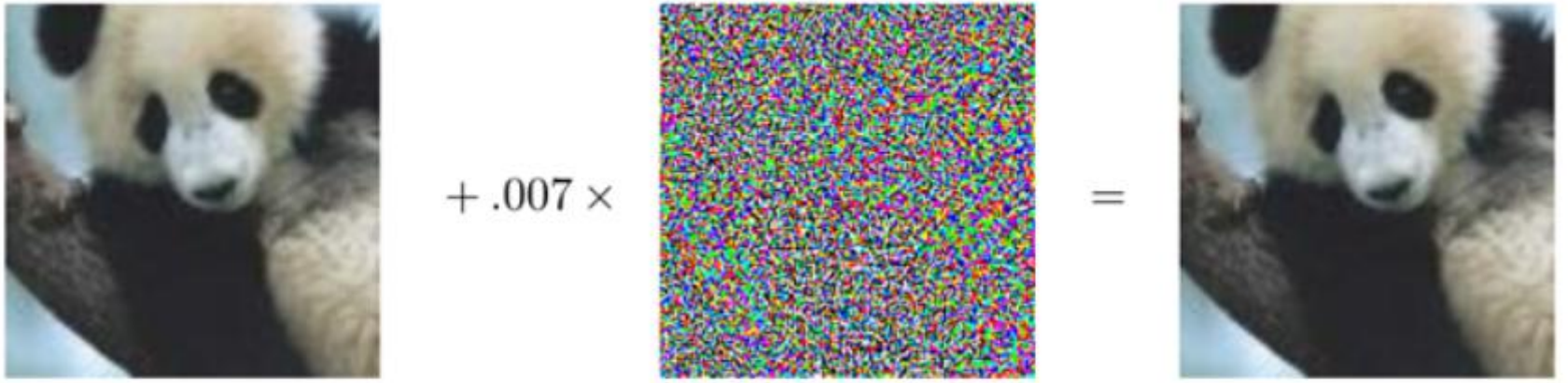



http://www.iro.umontreal.ca/~memisevr/dlss2015/goodfellow_adv.pdf

# Изменение по градиенту

## Fast Gradient Sign Method, FGSM




## Защита:

**обучение на атакованных данных**

**усложнение модели (ансамбли)**

**дистилляция (упрощение) сетей**

# Изменение по градиенту

## Незначительные изменения меняют ответ!

«Intriguing properties of neural networks» [Szegedy C. и др. https://arxiv.org/pdf/1312.6199.pdf]

# Состязательные примеры – в тексте

**Task:** Spam filtering. **Classifier:** LSTM. **Original label:** 100% Spam. **New label:** 89% Non-Spam.

**Text**: your ~~application~~ petition has been ~~accepted~~ recognized thank you for your ~~loan~~ borrower ~~request~~ petition , which we recieved yesterday , your ~~refinance~~ subprime ~~application~~ petition has been ~~accepted~~ recognized good credit or not , we are ready to give you a $ oov loan , after further review , our lenders have established the lowest monthly payments . approval process will take only 1 minute . please visit the confirmation link below and fill-out our short 30 second secure web-form . http : oov

---

**Task:** Sentiment analysis. **Classifier:** CNN. **Original label:** 81% Positive. **New label:** 100% Negative.

**Text**: i ~~went~~ moved to wing wednesday which is all-you-can-eat wings for $ oov even though they raise the prices it 's ~~still~~ ever really great deal . you can eat as many wings you want to get all the different ~~flavors~~ tastes and have a good time enjoying the atmosphere . the girls are smoking hot ! all the types of ~~sauces~~ dressings are awesome ! and i had at least 25 wings in one sitting . i would ~~definitely~~ certainly go again ~~just~~ simply not every ~~wednesday~~ friday maybe once a month .

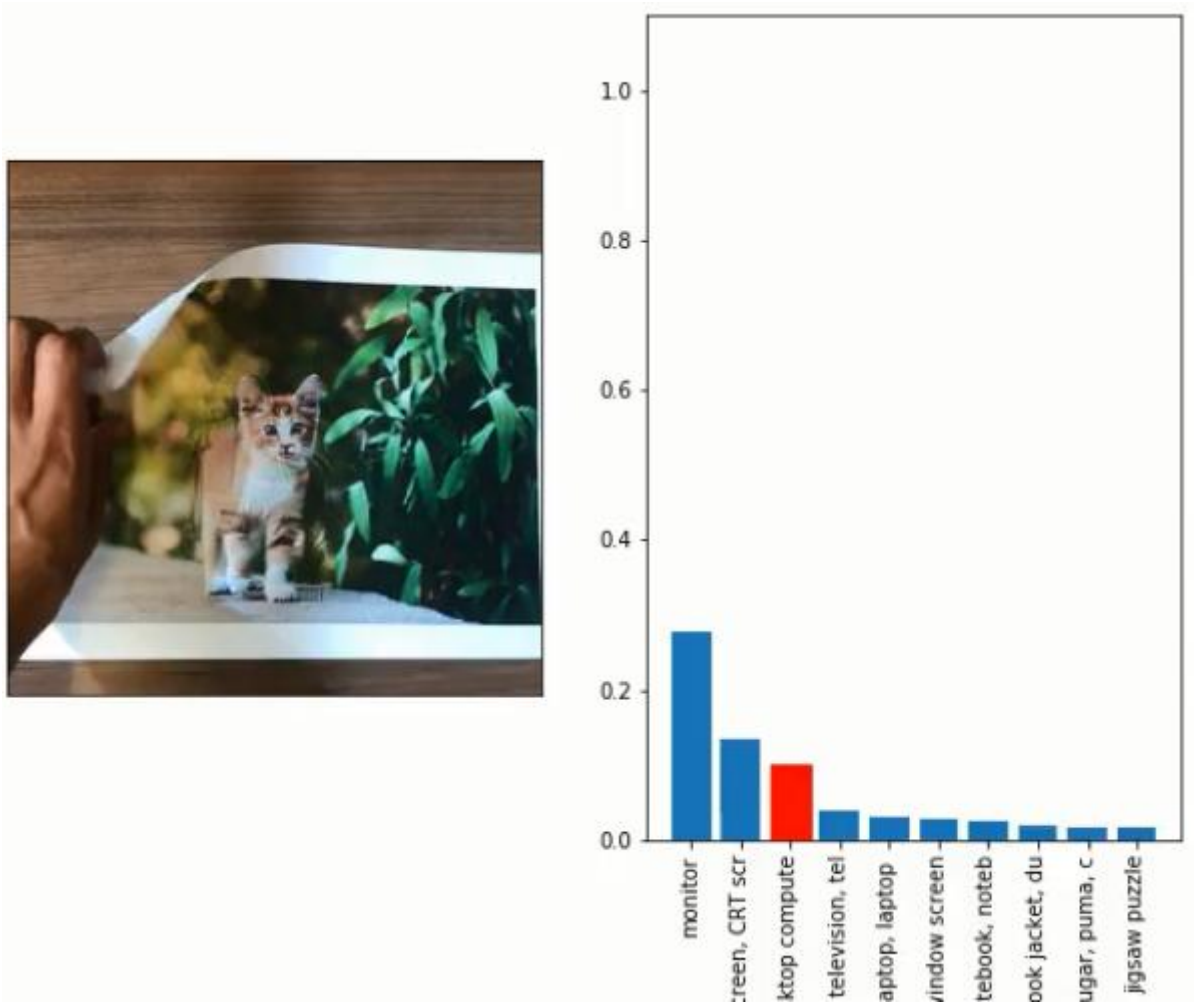
---

**Task:** Fake news detection. **Classifier:** Naive Bayes. **Original label:** 97% Fake. **New label:** 100% Real

**Text**: trump supporter whose ~~brutal~~ ferocious beating by black ~~mob~~ gangsta was caught on ~~video~~ tape ~~asks~~ demands : " what happened to america ? " [ video ] , " david oov , a 49 year ~~old~~ former ~~chicago~~ rochester man who was brutally beaten by a ~~mob~~ lowlife of black democrats ~~asks~~ demands , " what happened to america ? " here is his very ~~sad~~ disappointing story

https://openreview.net/pdf?id=r1QZ3zbAZ

# Атаки с помощью распечатки…




https://blog.openai.com/robust-adversarial-inputs/

# Атаки – резюме

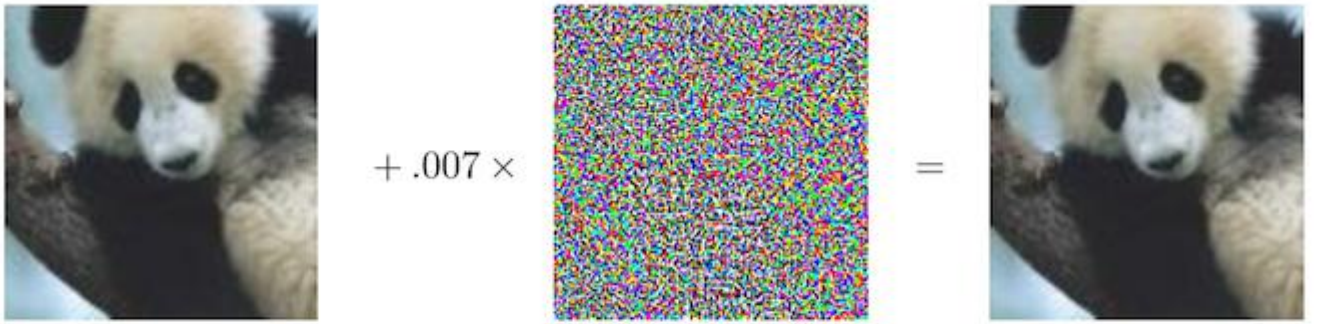
- атаковать легко
- защищаться сложно
- обучение на атакованных примерах (Adversarial Training) ~ регуляризация

# Состязательные примеры – Adversarial Examples

Goodfellow et. al (2014) https://arxiv.org/pdf/1412.6572.pdf

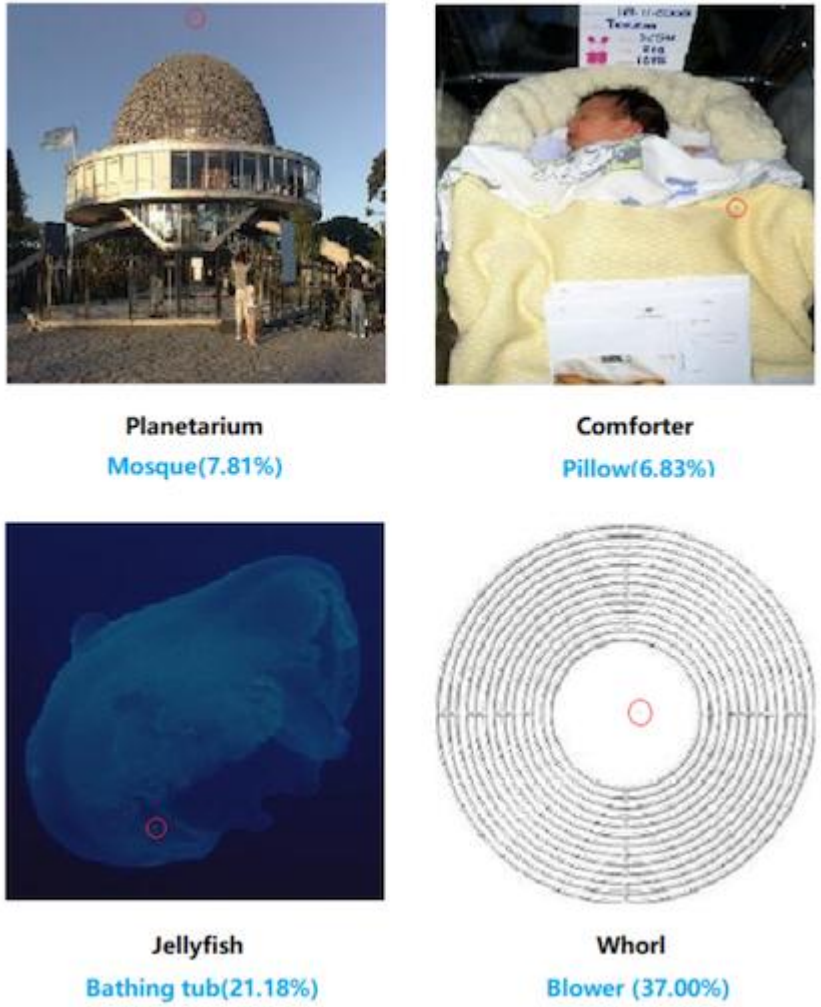
**Примеры небольших изменений объектов, которые существенно меняют ответ алгоритма**

**Показывают недостатки модели (такие примеры можно добавлять в обучение)**




$$x' = x + \varepsilon \operatorname{sgn}(\nabla_x L(x, y, \theta))$$

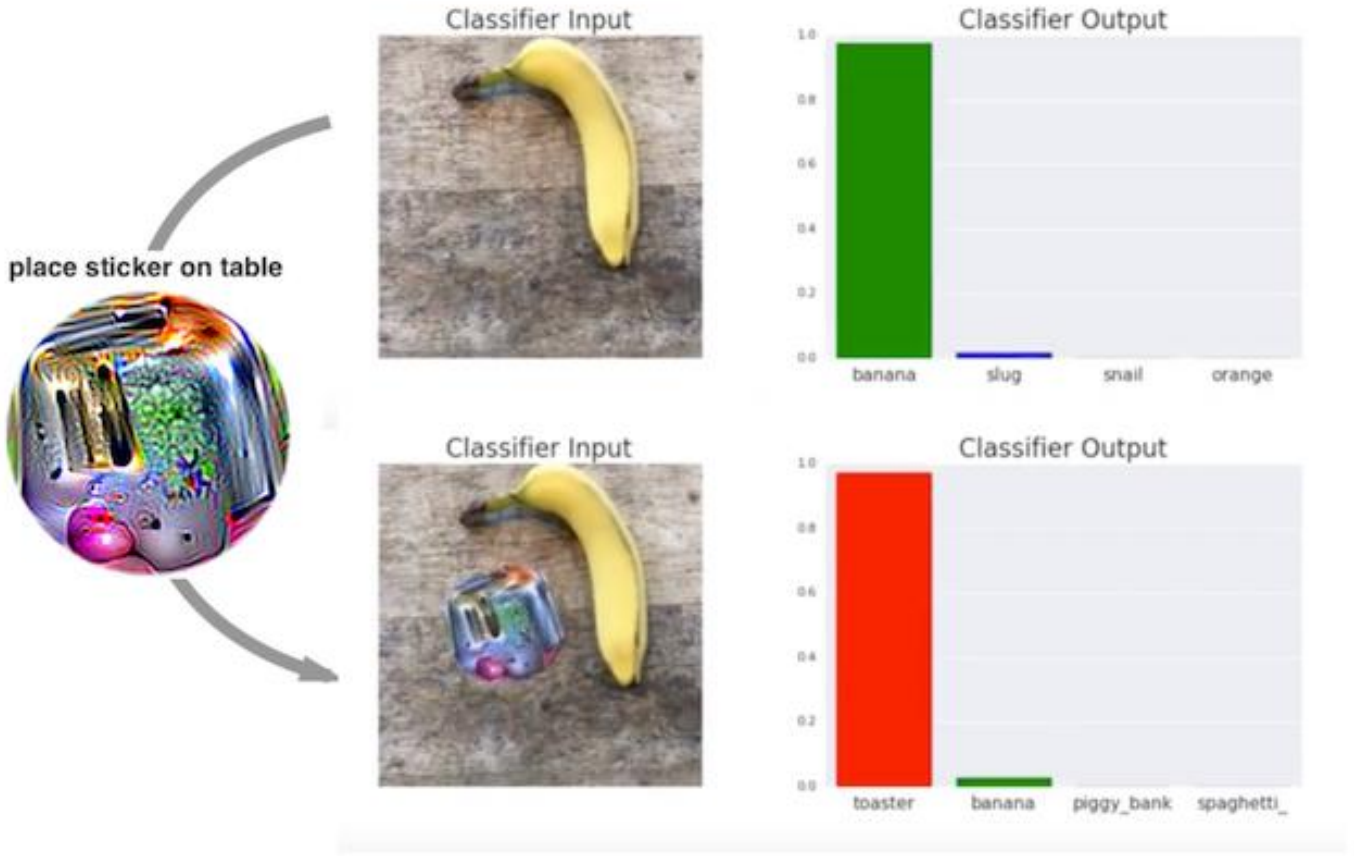
# Состязательные примеры – Adversarial Examples




## Атака изменением одного пикселя

Su, J., Vargas, D. V., & Kouichi, S. (2017). One pixel attack for fooling deep neural networks. Retrieved from http://arxiv.org/abs/1710.08864

# Состязательные участки – Adversarial patch

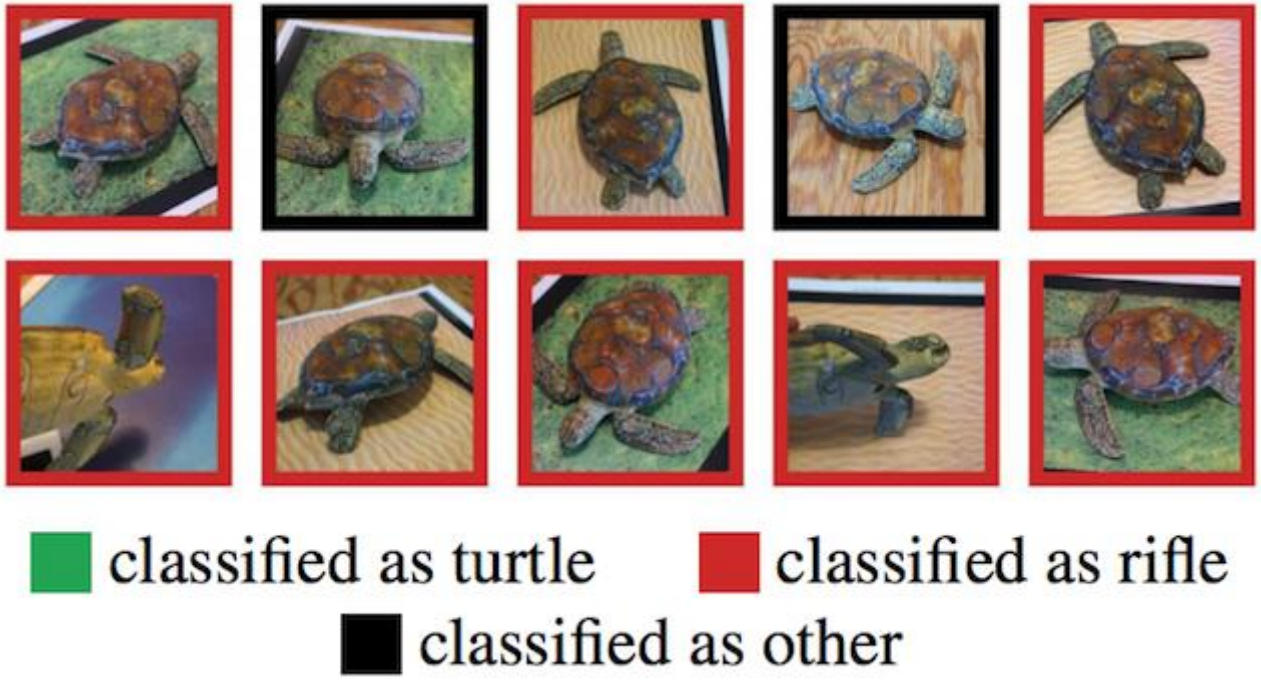



**Пример внедряемого участка,**
**который отвечает за нужную классификацию**

Brown, T. B., Mané, D., Roy, A., Abadi, M., & Gilmer, J. (2017). Adversarial Patch, (Nips).
http://arxiv.org/abs/1712.09665

# Робастные состязательные примеры

## устойчивы к некоторым преобразованиям




## Expectation Over Transformation (EOT) algorithm

## реальный 3D-объект (распечатан на принтере),

## который имеет неправильную определённую классификацию

Athalye, A., Engstrom, L., Ilyas, A., & Kwok, K. (2017). Synthesizing Robust Adversarial Examples. Retrieved from http://arxiv.org/abs/1707.07397

$$E_{t\sim T} \log P(y_{[s]} \mid t(x')) \to \max_{x'}$$

$t$ – **трансформация**

$T$ – **класс трансформаций (с распределением на нём)**

# Black box attack

**когда можно работать с моделью, но нельзя заглядывать внутрь (не знаем параметров, не можем вычислить градиент)**

- **дать ЧЯ несколько реальных примеров**
- **в цикле**
  - **(до)обучить суррогатную модель**
  - **сгенерировать новые примеры:**
    **суррогатная модель может с ними ошибаться**
  - **узнать результаты ЧЯ**
- **найти состязательные примеры для суррогатной модели**

Papernot, Nicolas, et al. “Practical black-box attacks against machine learning.” Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. ACM, 2017

# Обзор

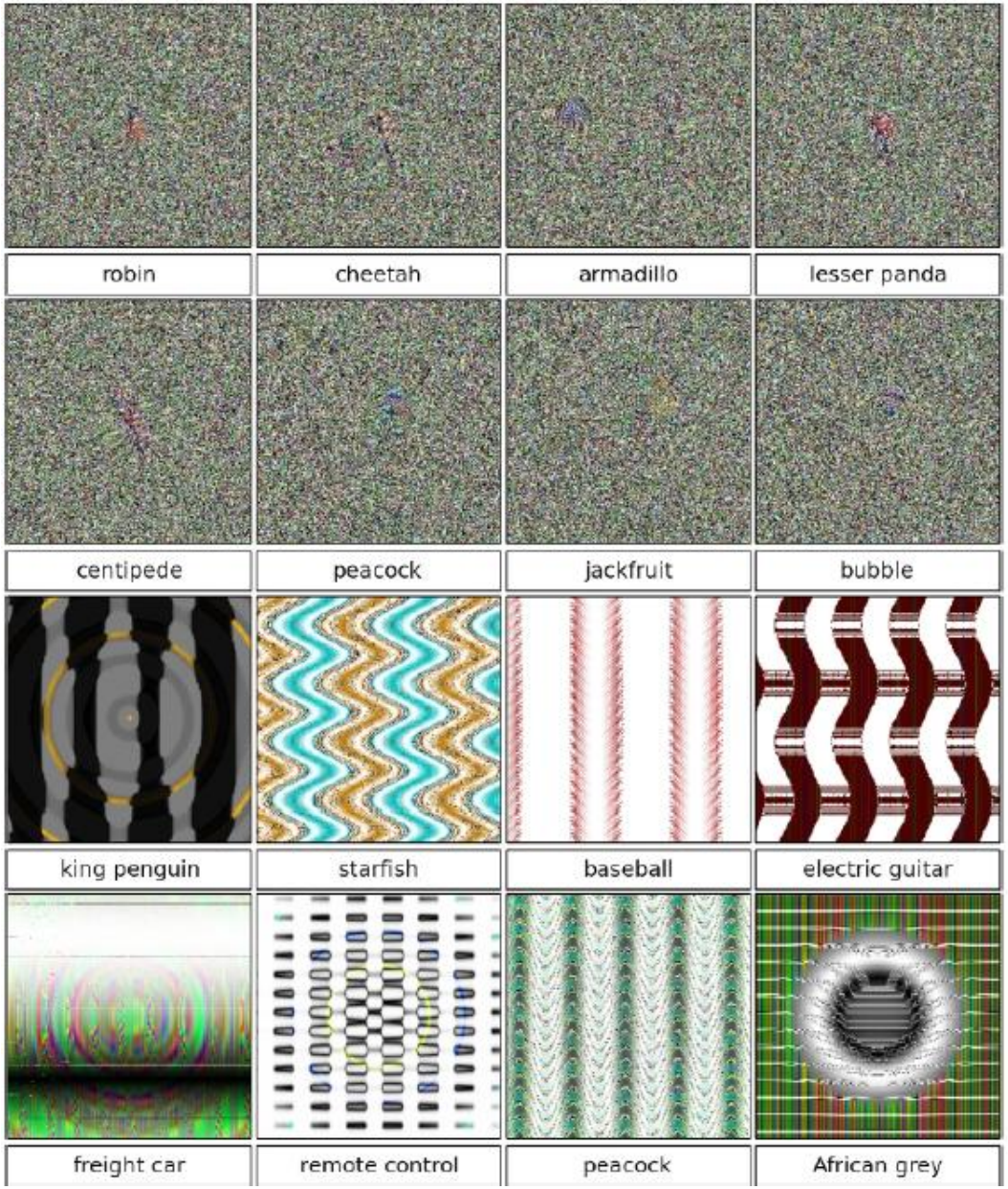
- двигаемся к поверхности, разделяющей классы (Bastani, 2016)
- ищем минимальные исправления в некоторой норме (Carlini, Wagner 2016)
- самые значимые пиксели – и их исправляем (Papernot, 2016)
- ...

Biggio, B., & Roli, F. (2017). Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning, 32–37. http://arxiv.org/abs/1712.03141

# Обманные изображения (Fooling Images)

- Берём картинку
  - белый шум
  - произвольная картинка из базы
- Выбираем какой-то класс
- Модифицируем картинку, чтобы максимизировать вероятность этого класса
- Останавливаемся

# Обманные изображения (Fooling Images)



# Генеративная модель

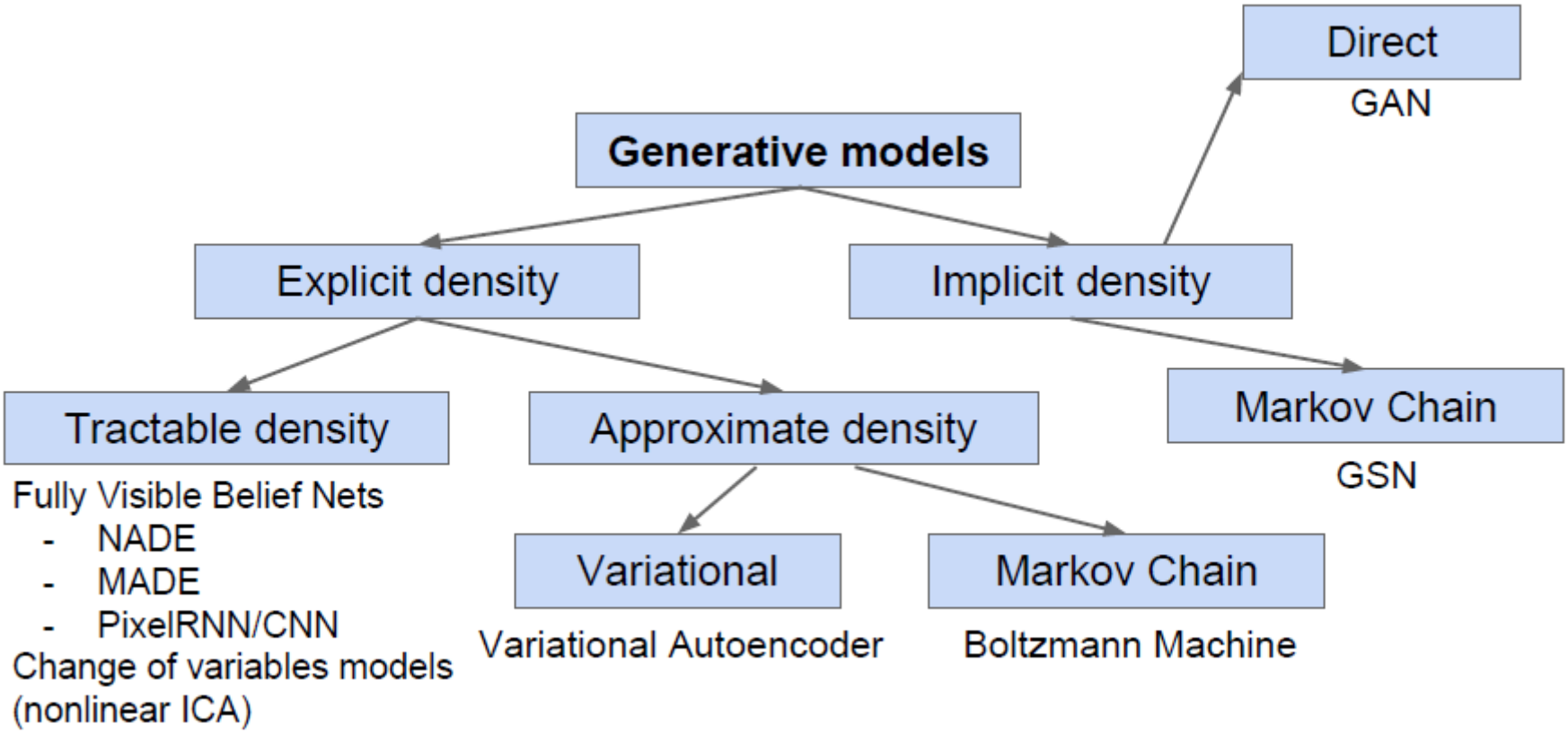
**Данные из некоторого распределения**

**Сгенерировать новые данные ~ это распределение**

**Цель: оценивание распределения данных высокой(!) размерности (изображение, аудио, видео, текст)**

- понять структуру данных
- найти зависимости между переменными
- генерировать новые данные с теми же свойствами
- генерация новых признаков без учителя

# Генеративные модели




**[Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017]**

# Генеративные модели

**подходы:**

**1) максимизация правдоподобия**

$$\prod_{x \in \text{train}} p_{\text{model}}(x;\theta) \to \max$$

**2) сделать распределение похожим на данные**

$$D_{\text{KL}}(p_{\text{data}} \,\|\, p_{\text{model}}) = \int p_{\text{data}}(x) \ln \frac{p_{\text{model}}(x)}{p_{\text{data}}(x)} \partial x \to \min$$

# VBN (fully visible belief networks)

$$p_{\text{model}}(x) = \prod_{i=1}^{m} p_{\text{model}}(x_i \mid x_1, \ldots, x_{i-1})$$

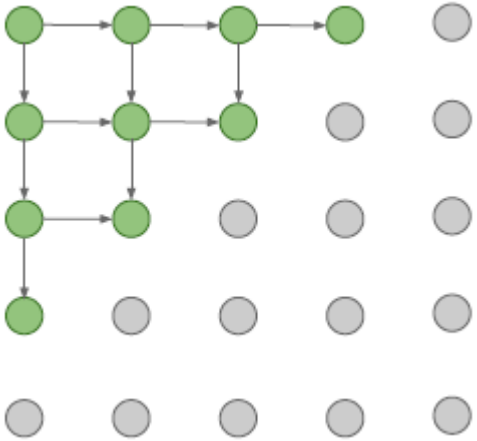
**пример – в генерации звука WaveNet [Oord et al., 2016]**

**---**

**в implicit-моделях – моделируем процесс сэмплирования, а не плотность.**

# PixelRNN

## Генерация изображения из угла (каждый пиксель зависит от предыдущих)

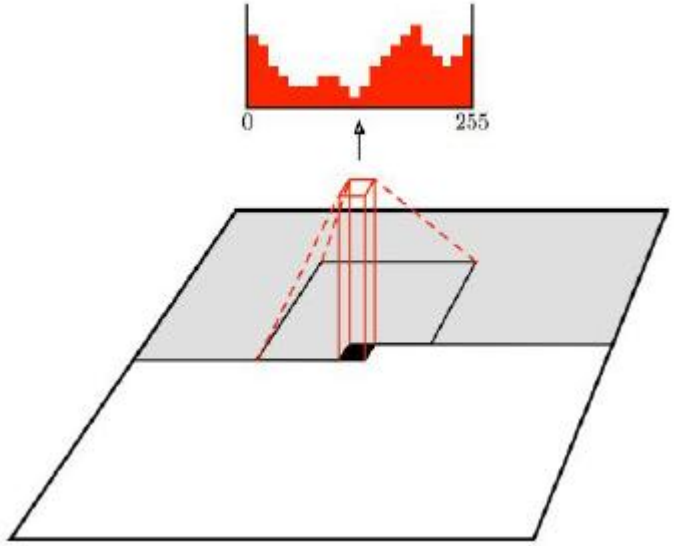


$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

**[van der Oord et al. 2016]**

# PixelCNN

## Генерация изображения из угла
## (каждый пиксель ~ CNN от предыдущего региона)




$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

**[van der Oord et al. 2016]**

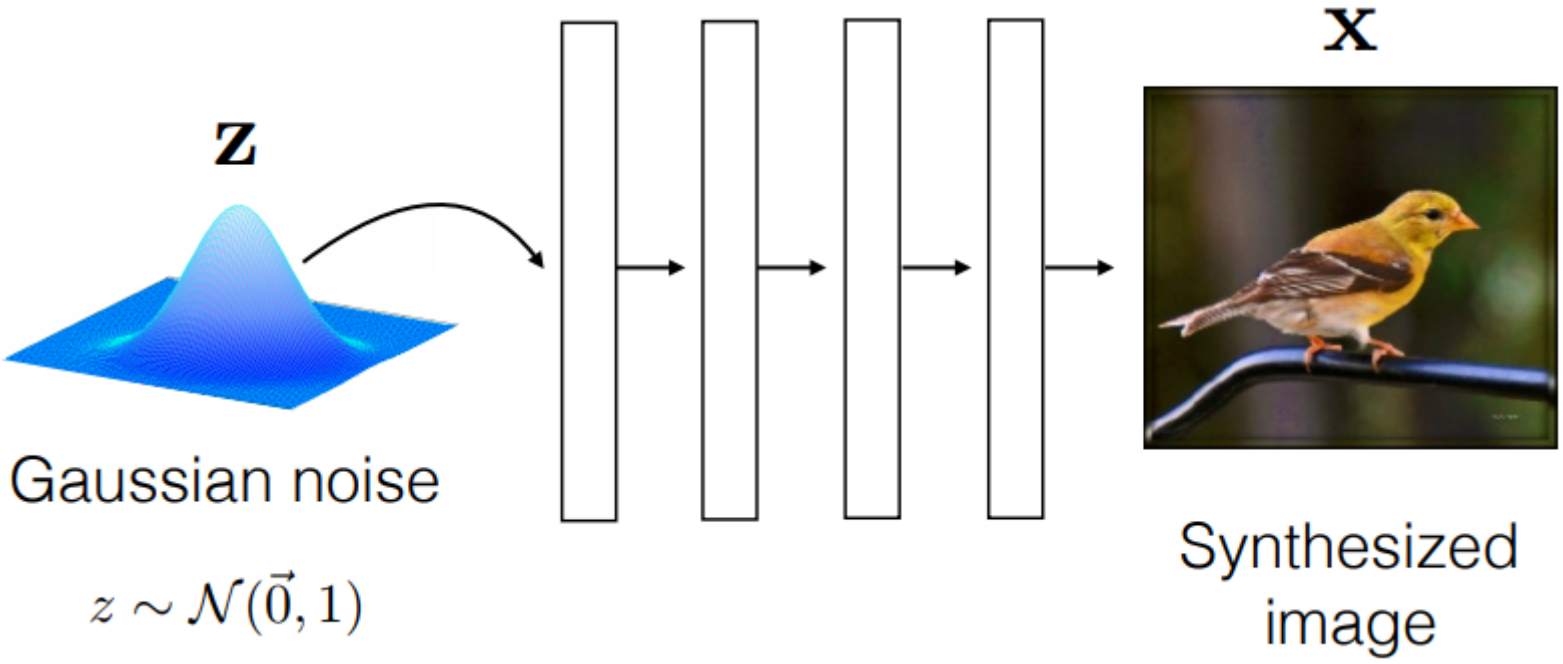
# PixelRNN и PixelCNN

**Низкая скорость генерации…**

**есть улучшения**

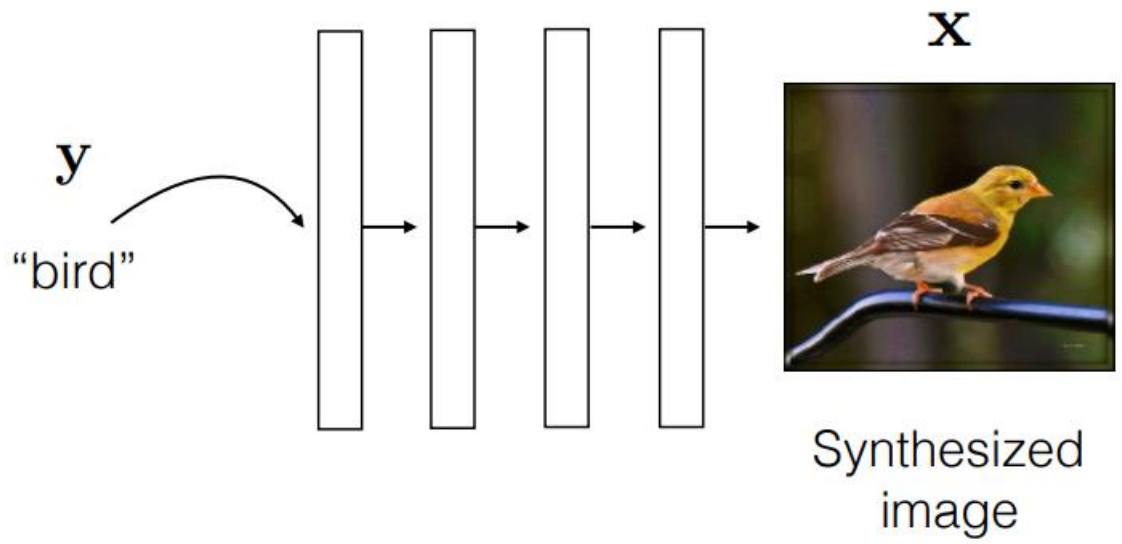
- Van der Oord et al. NIPS 2016
- Salimans et al. 2017 (PixelCNN++)

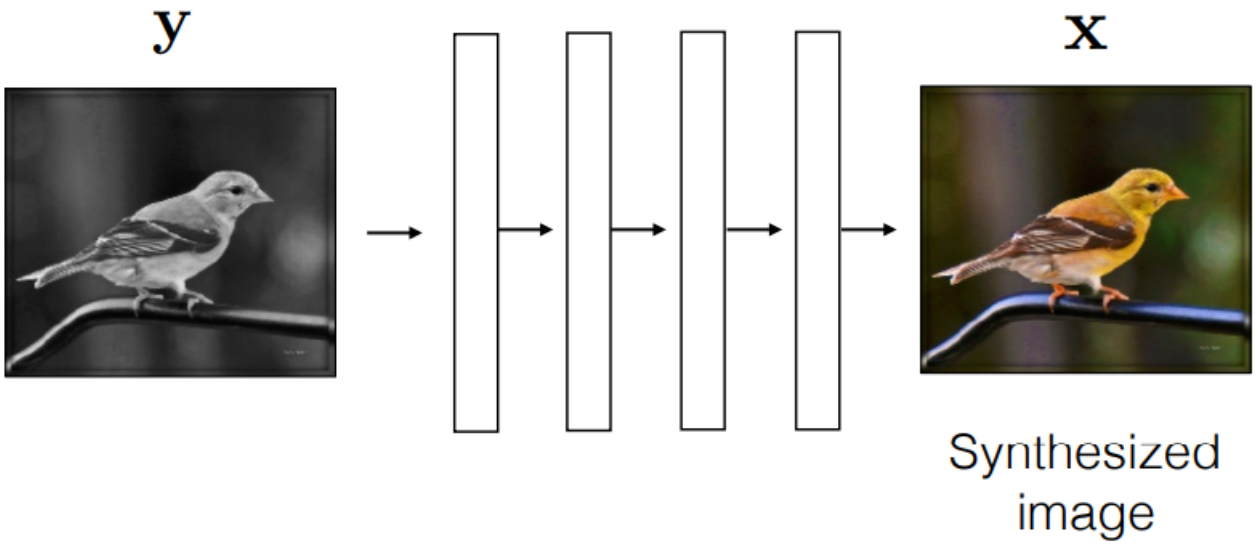
# Генеративная модель




http://www.mit.edu/~9.520/fall18/slides/Class12_GAN.pdf

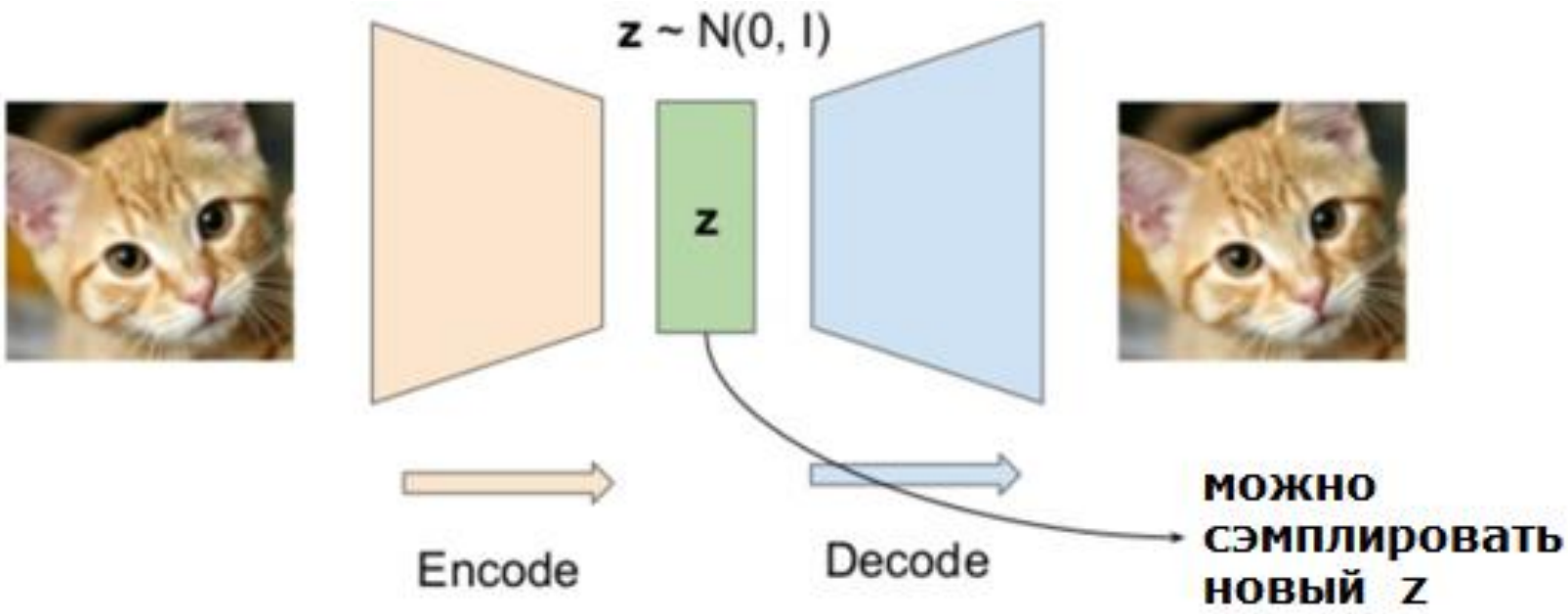
# Условная генеративная модель (Conditional Generative Model)



# VAE

**Обучать автокодировщик так, чтобы скрытые переменные ~ какое-то распределение**



# VAE

**Раньше:**

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i \mid x_1, \ldots, x_{i-1})$$

$$L \to \max_\theta$$

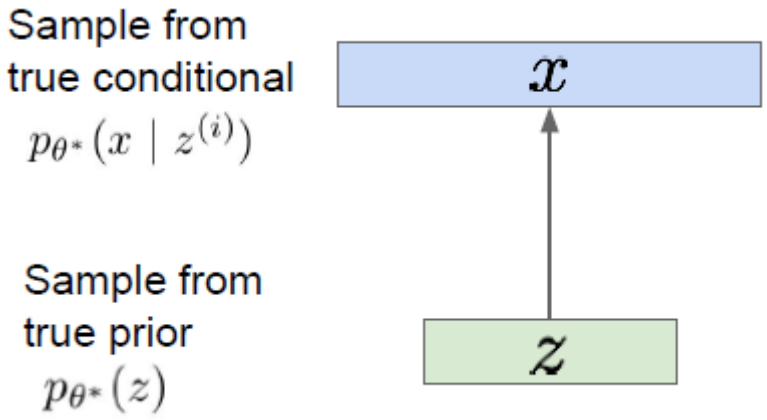
**Теперь:**

**через скрытую переменную**

$$p_\theta(x) = \int p_\theta(z)\, p_\theta(x \mid z) \partial z$$

**нельзя напрямую оптимизировать – оптимизируем оценку правдоподобия**

# Variational Autoencoders




**x – изображение, z – скрытая переменная**

$$p_\theta(x) = \int p_\theta(z)\, p_\theta(x \mid z) \partial z$$

$p_\theta(z)$ **~ пусть простое (нормальное)**

$p_\theta(x \mid z)$ **~ сложное, т.к. изображение по вектору**

**(КОДИРОВЩИК – пусть нейросеть)**

**Но невозможно вычислить** $p_\theta(x \mid z)$ **для всех z**

**(интегрирование невозможно)**

Kingma and Welling «Auto-Encoding Variational Bayes», ICLR 2014

# Variational Autoencoders

**Решение: кроме кодировщика сделать декодировщик $q_\varphi(z \mid x)$**

**это позволит получить оценку на правдоподобие**

**(заодно полезно для других задач как генератор признаков)**

**encoder (recognition/inference) network**

**decoder (generation) network**

# Variational Bayesian Inference

$$\log p_\theta(x) = \int q_\varphi(z \mid x) \log p_\theta(x) \partial z = \int q_\varphi(z \mid x) \log \frac{p_\theta(x,z)}{p_\theta(z \mid x)} \partial z =$$

$$= \int q_\varphi(z \mid x) \log \frac{p_\theta(x,z) q_\varphi(z \mid x)}{q_\varphi(z \mid x) p_\theta(z \mid x)} \partial z =$$

$$= \int q_\varphi(z \mid x) \log \left( p_\theta(x \mid z) \frac{p_\theta(z)}{q_\varphi(z \mid x)} \frac{q_\varphi(z \mid x)}{p_\theta(z \mid x)} \right) \partial z =$$

# Variational Bayesian Inference

$$= \int q_\varphi(z \mid x) \log p_\theta(x \mid z) \partial z - \quad \sim \mathbf{E} \log p_\theta(x \mid z)$$

$$- \int q_\varphi(z \mid x) \log \frac{q_\varphi(z \mid x)}{p_\theta(z)} \partial z + \quad \sim D_{KL}(q_\varphi(z \mid x) \| p_\theta(z))$$

$$+ \int q_\varphi(z \mid x) \log \frac{q_\varphi(z \mid x)}{p_\theta(z \mid x)} \partial z \quad \sim D_{KL}(q_\varphi(z \mid x) \| p_\theta(z \mid x))$$

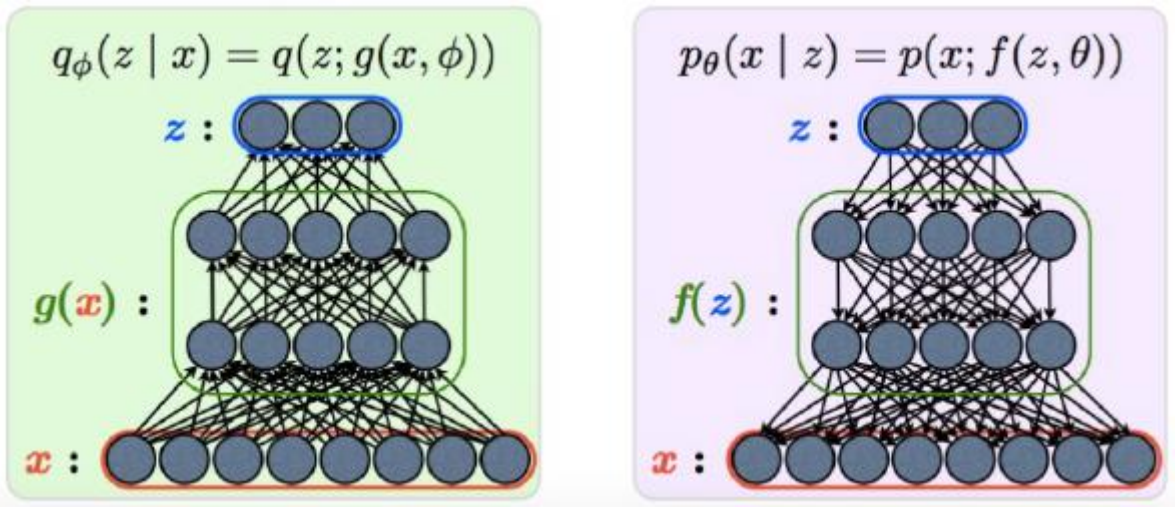
**1) с помощью сэмплирования и кодировщика – это реконструкция данных (Neural Decoder Reconstruction Loss)**

**2) KL между гауссианой для декодировщика и априорным (можно посчитать) – это близость априорного и апостериорного распределений (регуляризация латентного представления)**

**3) уже говорили о невозможности вычисления, но $D_{KL} \geq 0$**

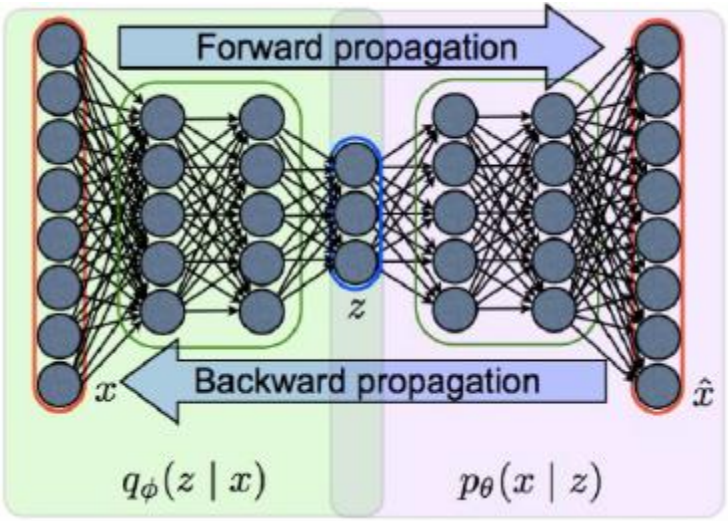
**Можно оценить всё выражение, отбросив последнее слагаемое**

# Variational Bayesian Inference



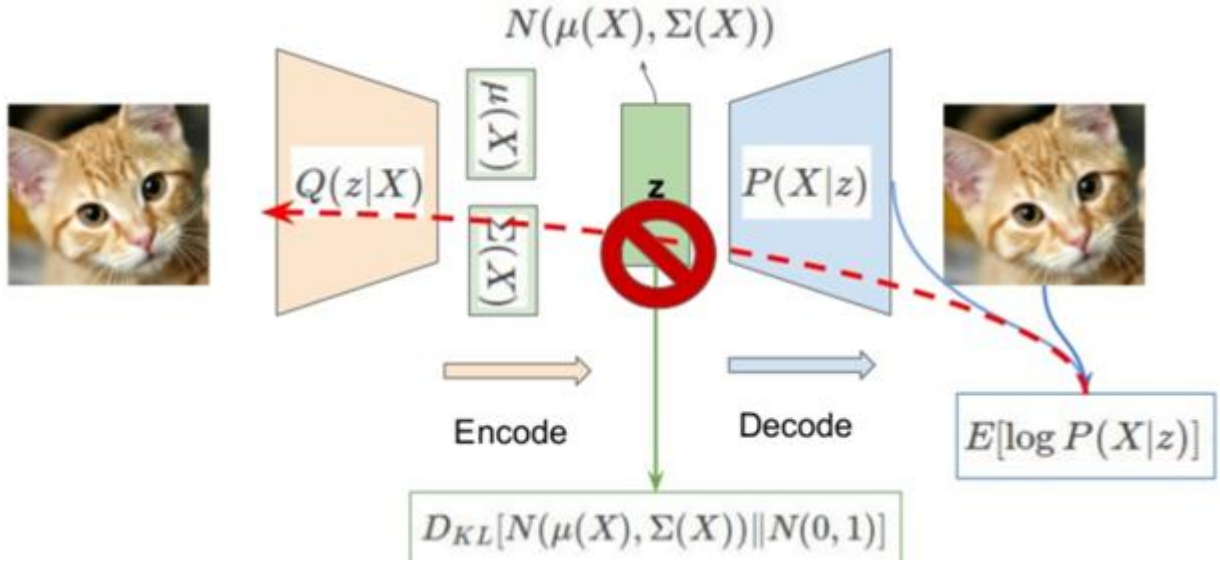

**Возьмём в качестве $q_\varphi(z \mid x)$ нормальное распределение**

$\text{norm}(z \mid \mu_z(x), \sigma_z(x))$ **тут reparametrization trick**

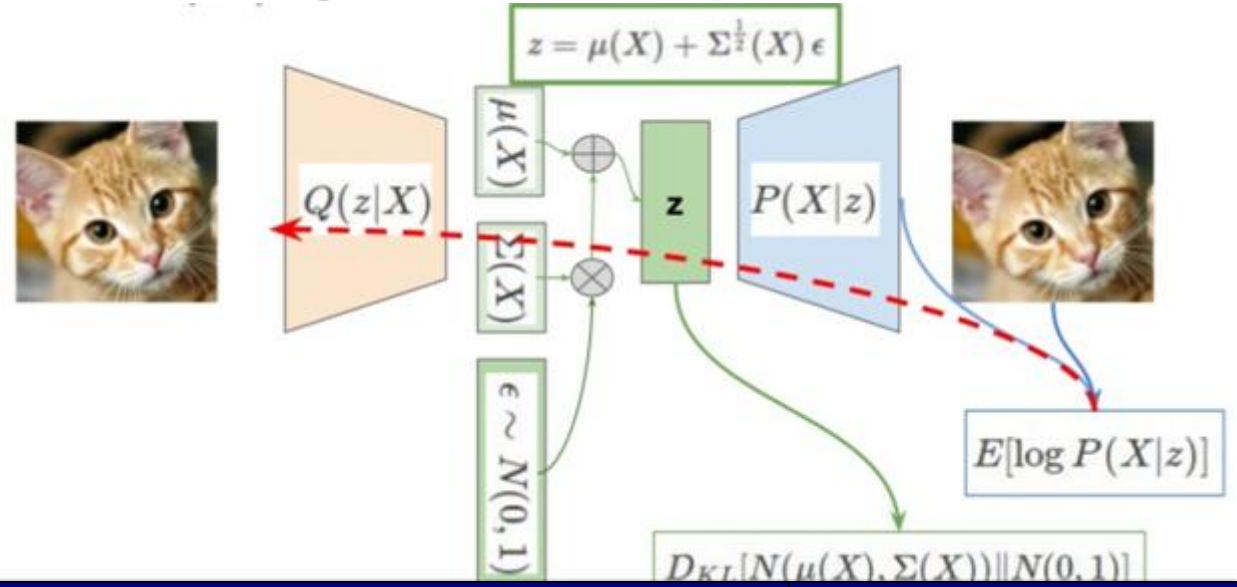


# Reparametrizaton trick

## как делать BP через распределение?



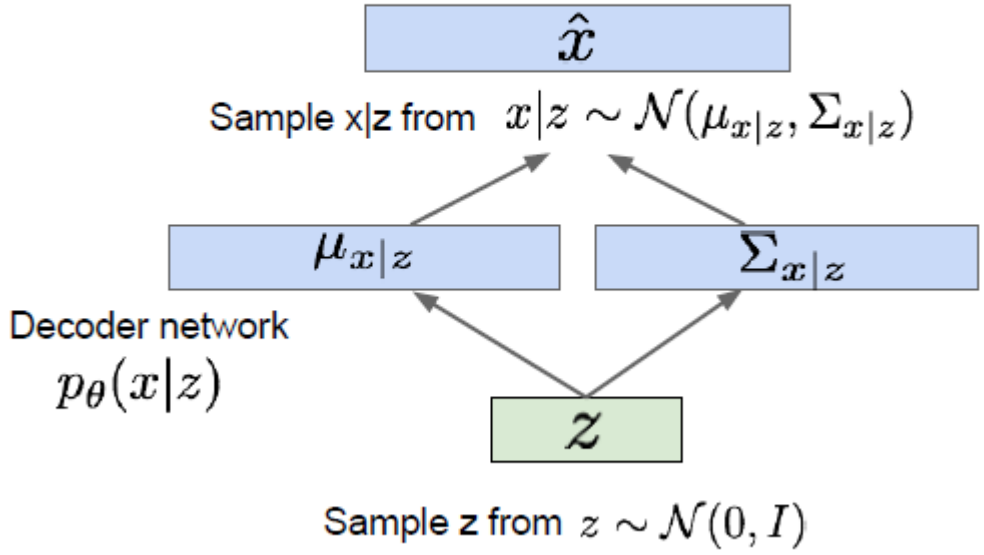

## Будем сэмплировать ε!



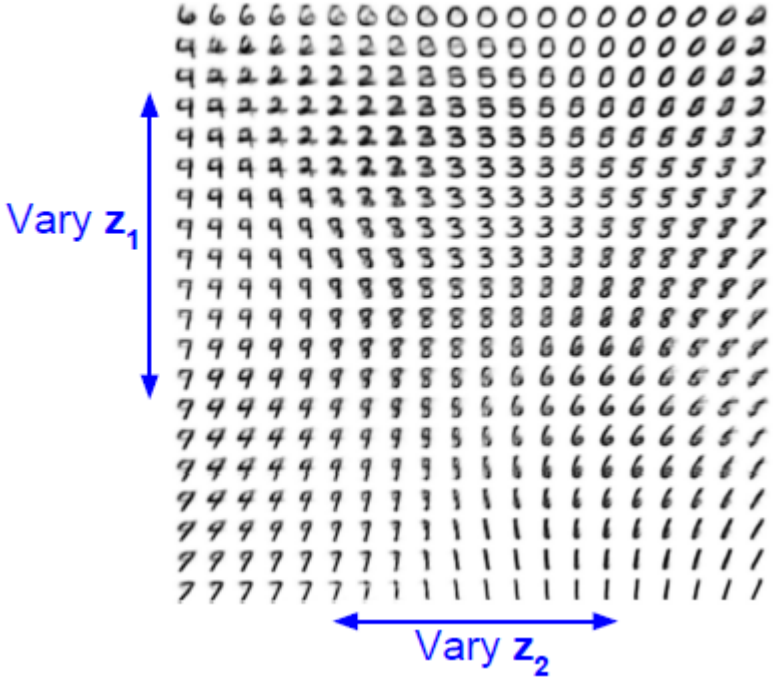
# Variational Bayesian Inference

## По поводу гауссаны – там берутся особые распределения

Use decoder network. Now sample z from prior!




Data manifold for 2-d **z**



# Generative Adversarial Networks

## [Goodfellow et al., 2014]

**Генератор – сеть, которая порождает объект (изображение) из шума**

**Дискриминатор – сеть, различающая настоящие и сгенерированные объекты**

**Самое главное:**

**По сути, дискриминатор – дифференцируемая функция ошибки!**

**Эту идею можно использовать там, где нет подходящих функций ошибок…**

**В отличие от PixelCNN, VAE нет явной функции плотности!**

**Игровой подход!**

## Что могут GAN

- научились работать со сложными объектами в пространствах высокой размерности
- генерация реалистичных объектов
- заполнение пропусков (и другие задачи USL)
- использование генеративных моделей-ассистентов (при редактировании)

## Задачи

- улучшение изображений (Image Inpainting)
- улучшение звука (speech enhancement)
- генерация изображений (Image Generation)
- супер-разрешение (Super-resolution)

# GAN

$$\min_{\theta}\max_{\varphi}\left[\mathbf{E}_{x\sim p_{\text{data}}}\log D_{\varphi}(x)+\mathbf{E}_{z\sim p(z)}\log(1-D_{\varphi}(G_{\theta}(x))\right]$$

**Дискриминатор выводит правдоподобие из [0, 1]**

$D_{\varphi}(x)$ **– для настоящих данных**

$D_{\varphi}(G_{\theta}(x))$ **- для сгенерированных данных**

**Дискриминатор хочет** $D_{\varphi}(x)\to 1,\ D_{\varphi}(G_{\theta}(x))\to 0$

**Генератор хочет** $D_{\varphi}(G_{\theta}(x))\to 1$

**Для оптимизации лучше:**

$$\max_{\varphi}\left[\mathbf{E}_{x\sim p_{\text{data}}}\log D_{\varphi}(x)+\mathbf{E}_{z\sim p(z)}\log(1-D_{\varphi}(G_{\theta}(x))\right]$$

$$\max_{\theta}\mathbf{E}_{z\sim p(z)}\log(D_{\varphi}(G_{\theta}(x)))$$

# Советы по настройке GAN

1. Нормализация входа (изображения → [–1, +1], выход - tanh)
2. Вместо min(log(1-D(G))) лучше max(log(D(G)))
приём: меняем метки местами real ↔ fake
3. z сэмплируется не из равномерного, а гауссовского распределения, см. также https://arxiv.org/abs/1609.04468
4. Минибатчи лучше делать чистыми (все real или все fake)
5. Не использовать Sparse Gradients (ReLU, MapPool)
лучше LeakyReLU,
Downsampling: Average Pooling, Conv2d + stride
Upsampling: PixelShuffle, ConvTranspose2d + stride
6. Лучше размывать метки: 0 → [0, 0.3], 1 → [0.7, 1.2]
7. Используйте DCGAN (или гибридные: KL + GAN или VAE + GAN)
8. Используйте трюки из RL (например, Experience Replay)
9. Adam для генератора, SGD для дискриминатора
10. Мониторьте ошибки (ex: loss(D)~0 failure mode – проверяйте градиенты)

# Советы по настройке GAN

**11. Добавляйте шум ко входу / к слоям генератора**

**12. Дискретные переменные в условных GANax: используйте Embedding layer, добавляйте как новый канал в изображениях, поддерживайте низкой embedding dimensionality**

**13. Используйте Dropouts в G**

https://github.com/soumith/ganhacks

# GAN в зависимости от функции ошибки в дискриминаторе

| | |
|---|---|
| **Binary cross-entropy** | **Vanilla GAN** |
| **Least squares** | **LSGAN** |
| **Wasserstein GAN + Gradient penalty** | **WGAN-GP** |

# DCGAN

## – Deep Convolutional Generative Adversarial Networks

## [Radford et al., 2016]

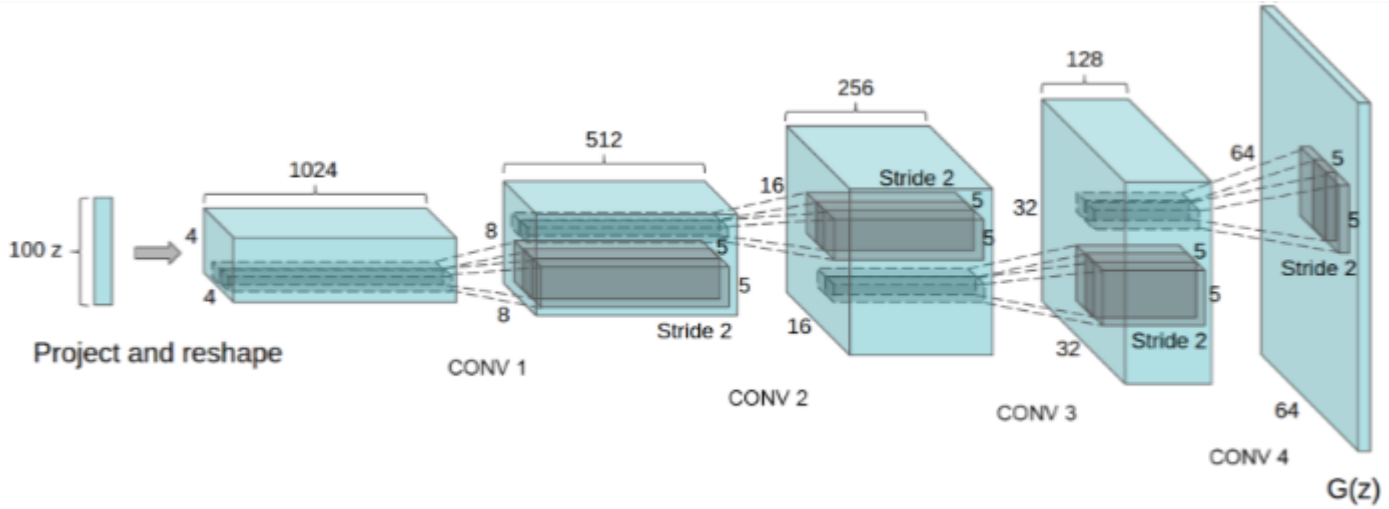


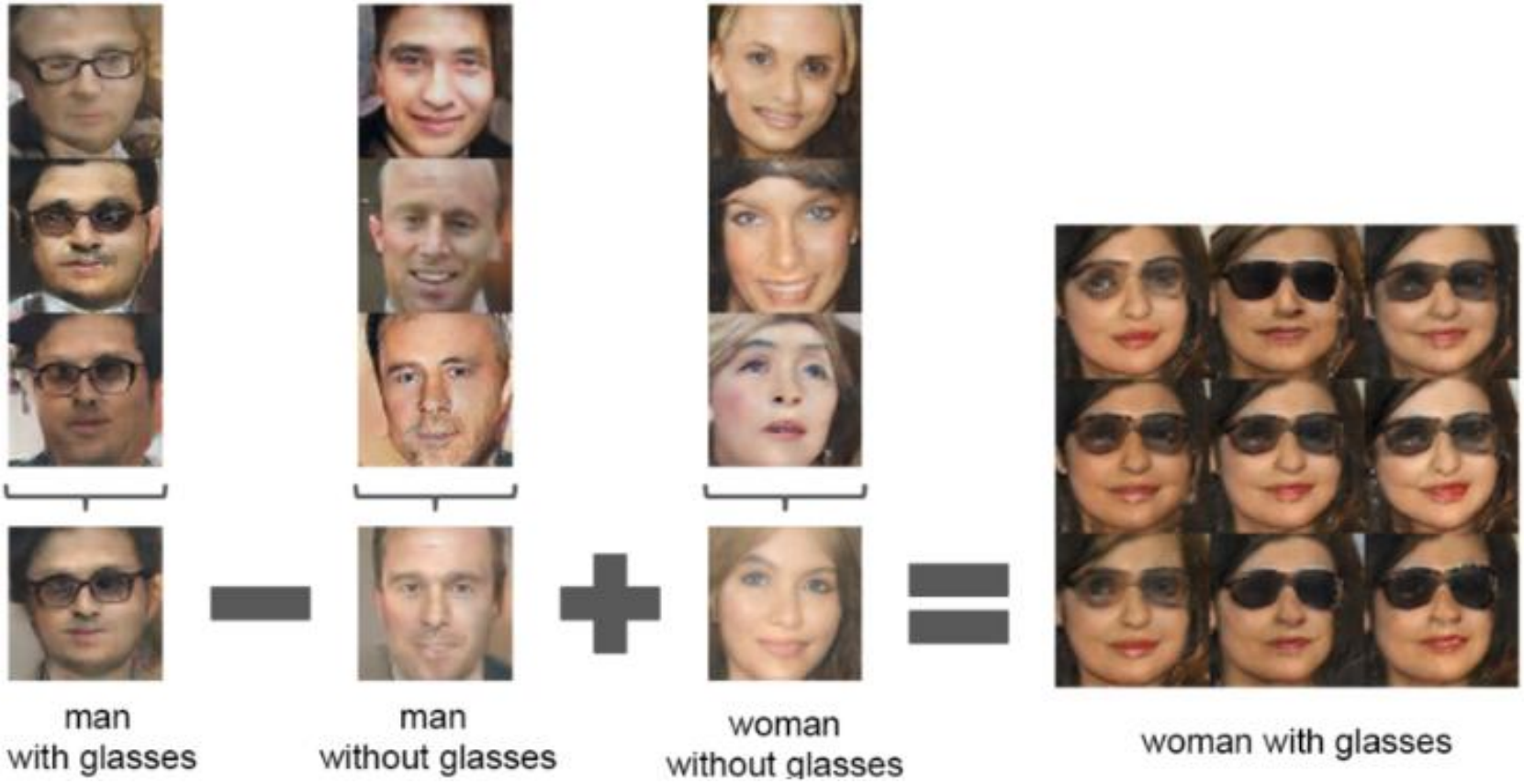

Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution $Z$ is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a $64 \times 64$ pixel image. Notably, no fully connected or pooling layers are used.

Architecture guidelines for stable Deep Convolutional GANs

- 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.

# DCGAN

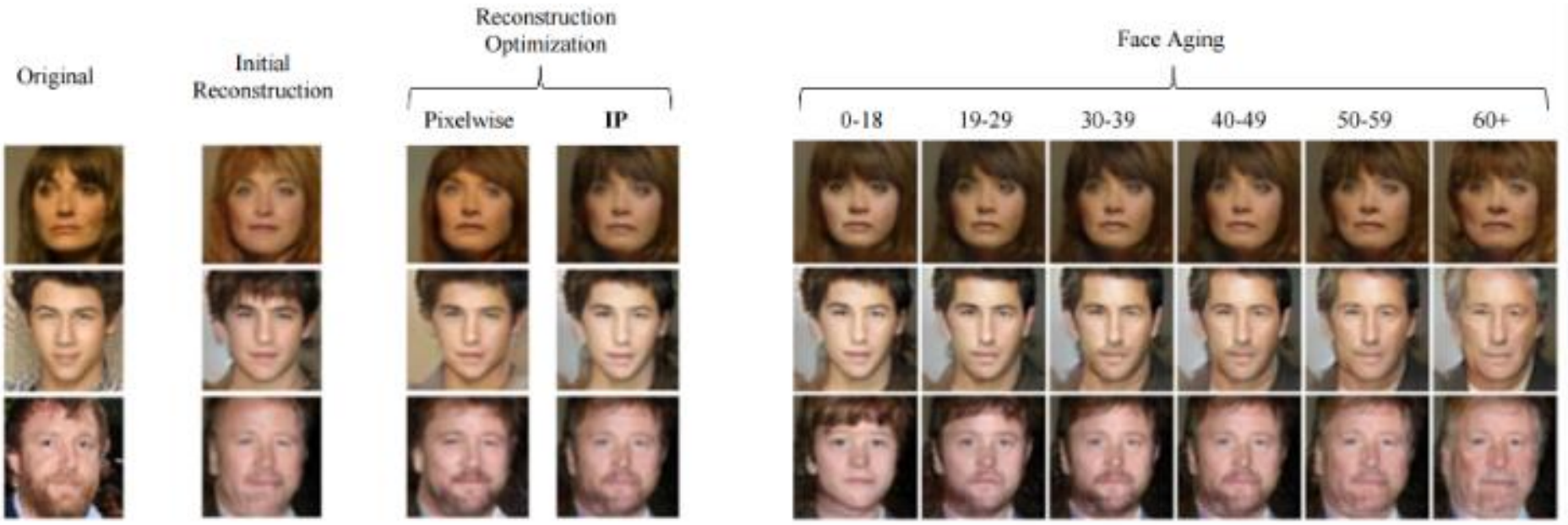
# DCGAN




[Radford et al, ICLR 2016]

# DCGAN

## [Antipov et al., 2017]




**состаривание условным GANом**

# DCGAN


[Ledig et al., 2017] https://arxiv.org/abs/1609.04802

# DCGAN

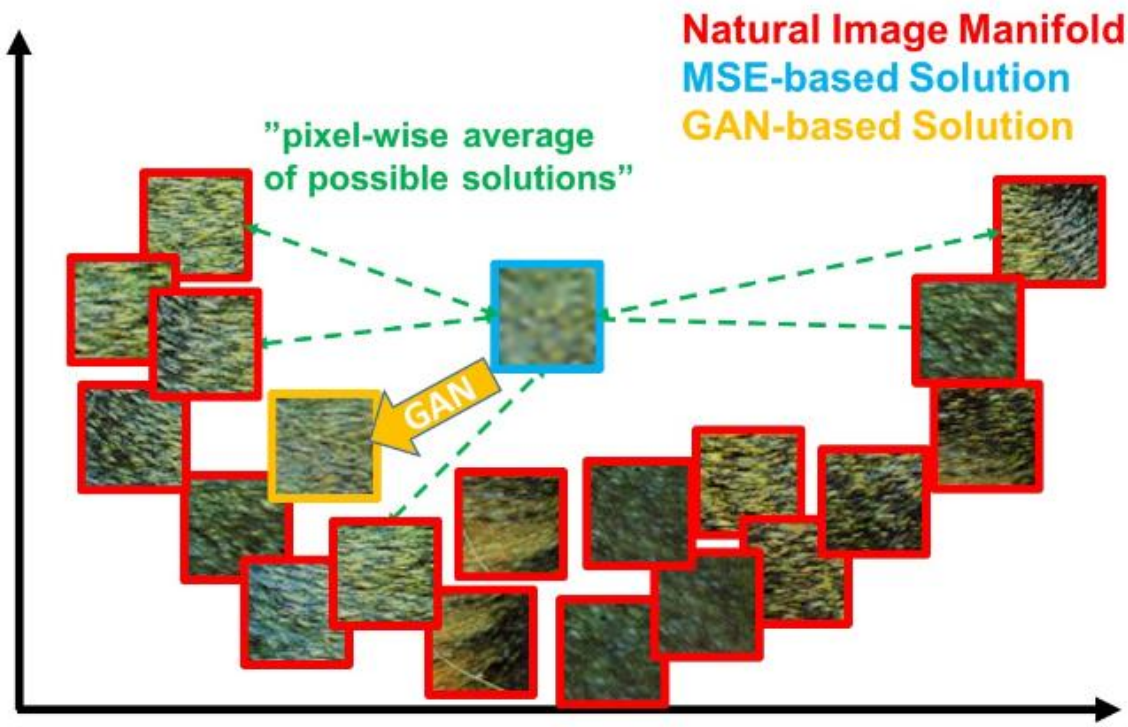
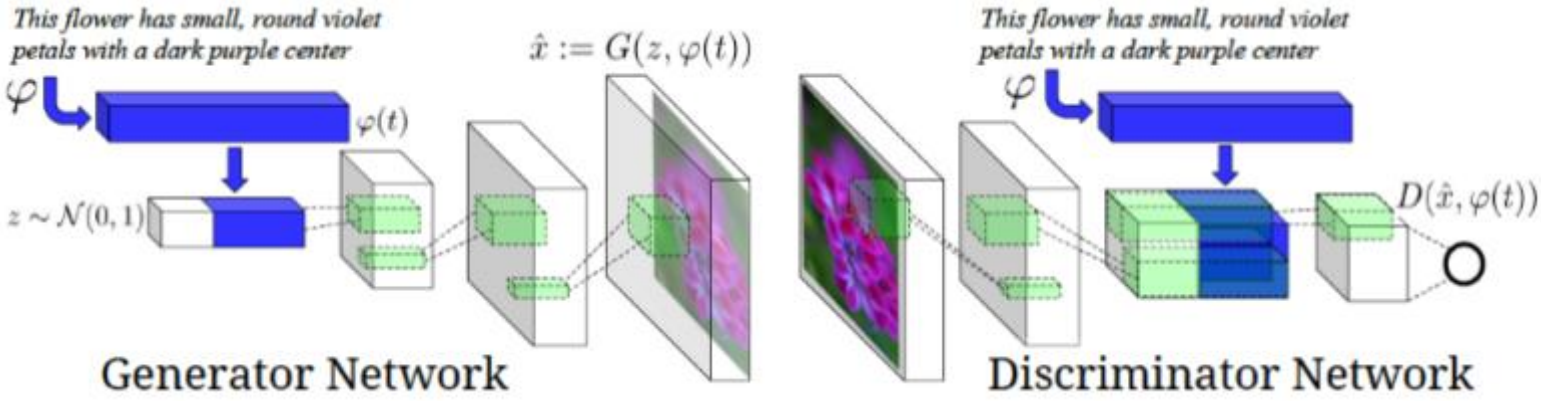



Figure 3: Illustration of patches from the natural image manifold (red) and super-resolved patches obtained with MSE (blue) and GAN (orange). The MSE-based solution appears overly smooth due to the pixel-wise average of possible solutions in the pixel space, while GAN drives the reconstruction towards the natural image manifold producing perceptually more convincing solutions.

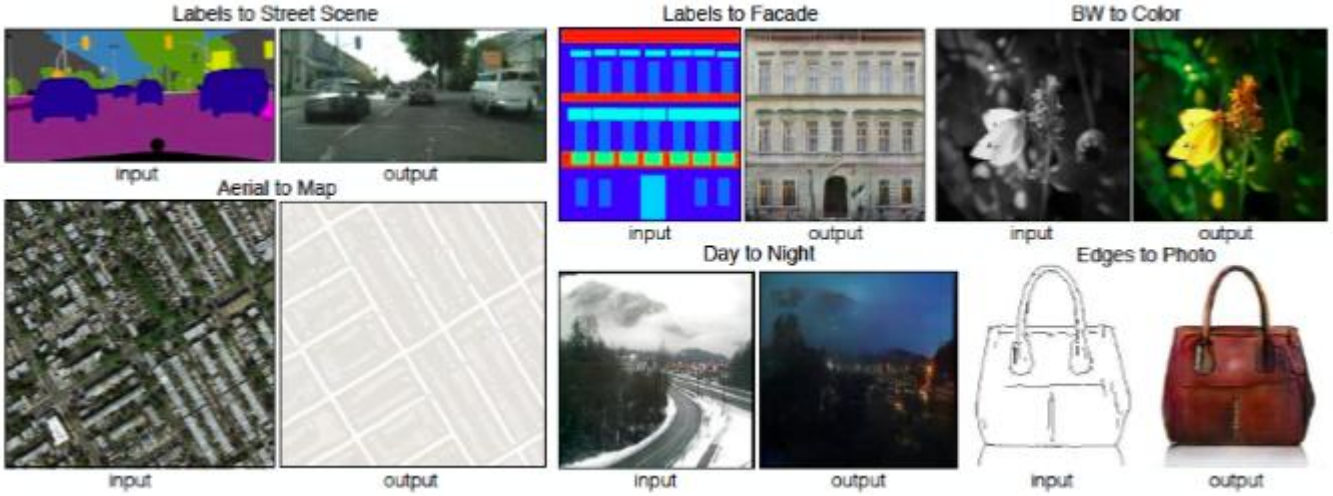
# SRGAN

## Super resolution GAN

## Text to image




## Image to image



# ~U-net

# LSGAN

(a) Church outdoor.

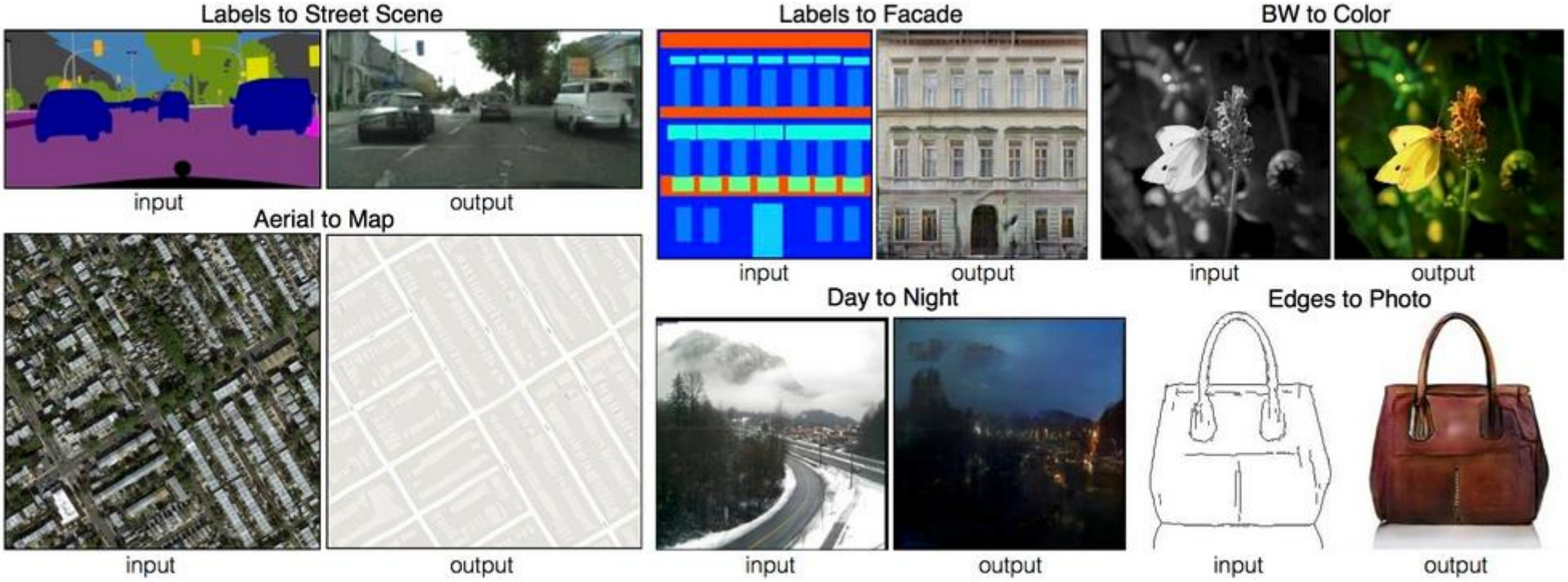
(b) Dining room.

(c) Kitchen.

(d) Conference room.

# [Karras et al. 2017]

# Pix2pix с условными состязательными сетями



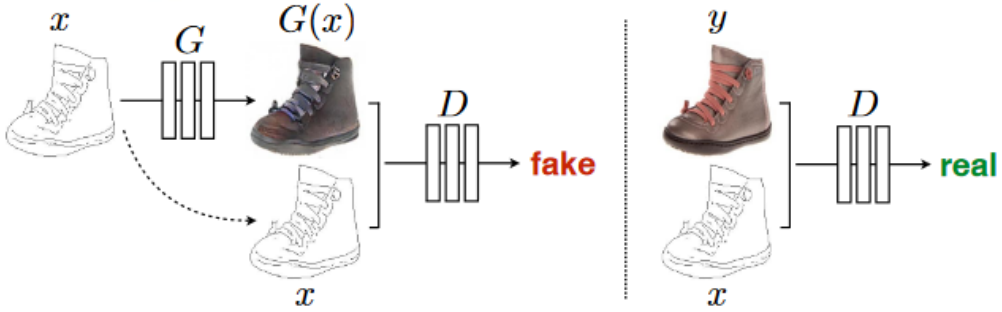
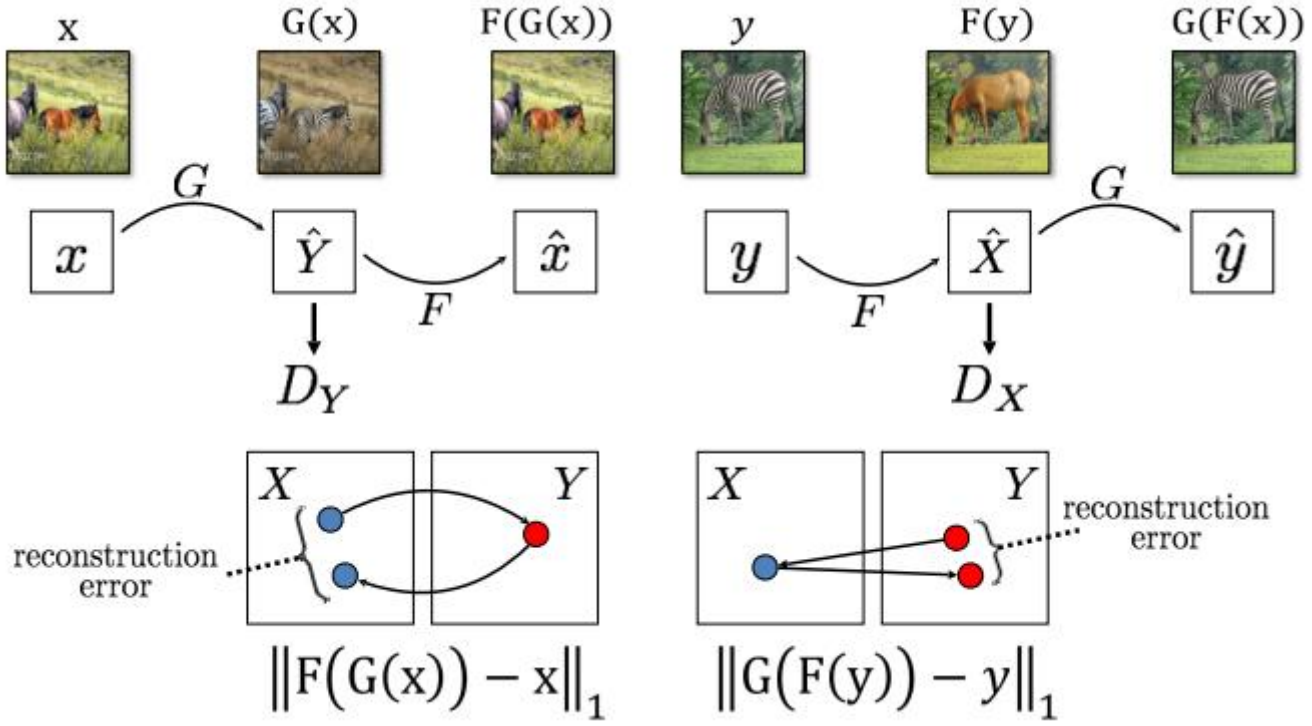




Figure 2: Training a conditional GAN to map edges→photo. The discriminator, $D$, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, $G$, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

**Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros «Image-to-Image Translation with Conditional Adversarial Nets» // CVPR 2017, https://phillipi.github.io/pix2pix/**

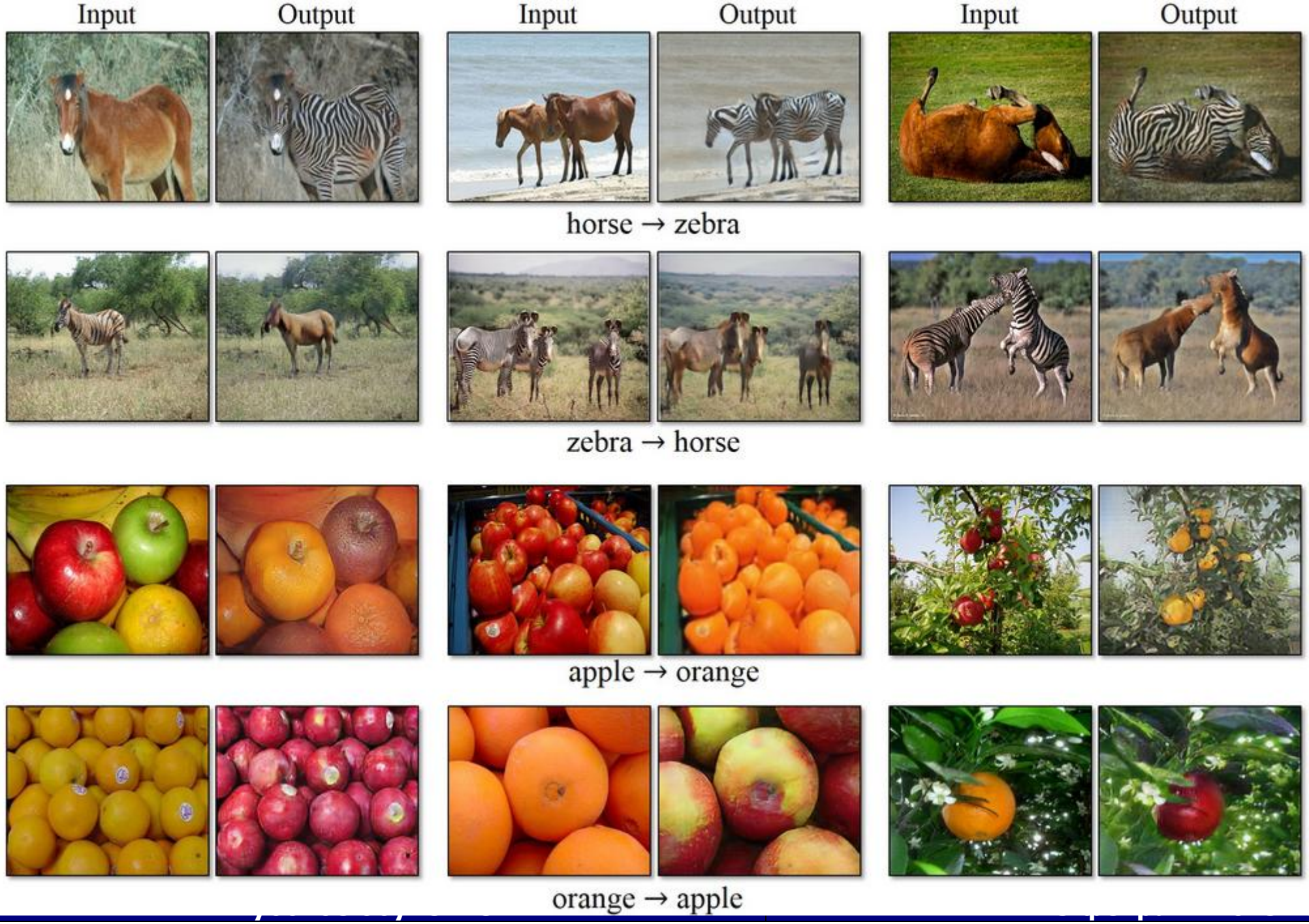
# CycleGAN

## Cycle Consistency Loss

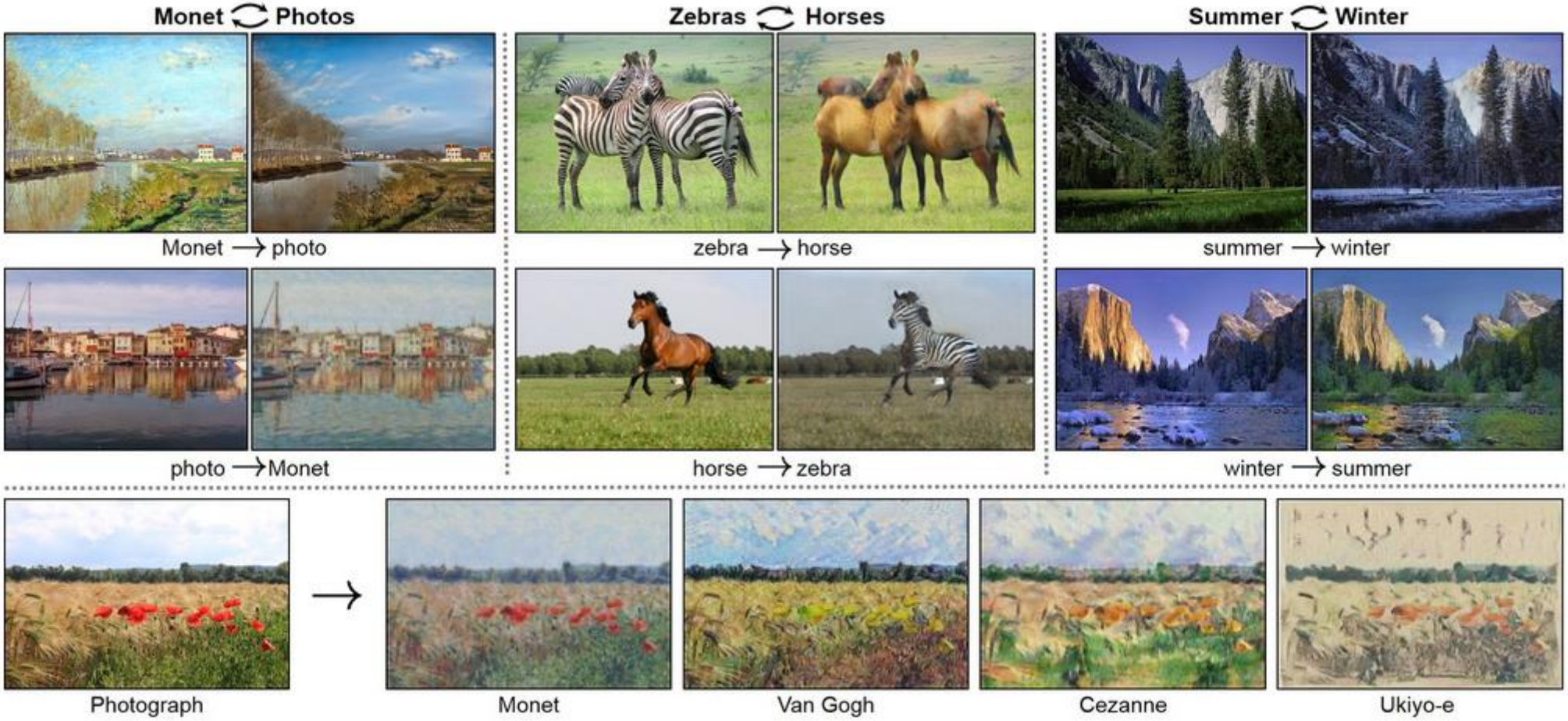



[Zhu et al., 2017]

https://junyanz.github.io/CycleGAN/

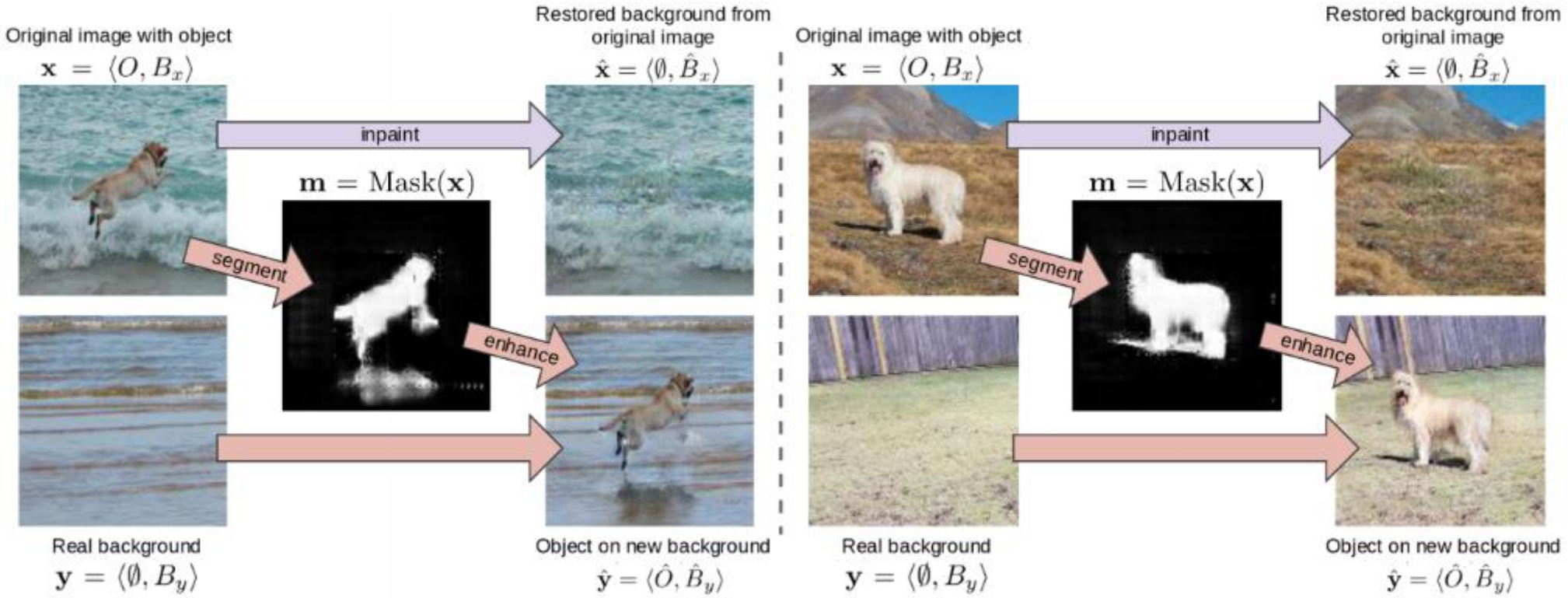

Input
Output
Input
Output
Input
Output
horse → zebra
zebra → horse
apple → orange
orange → apple

# CycleGAN


Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

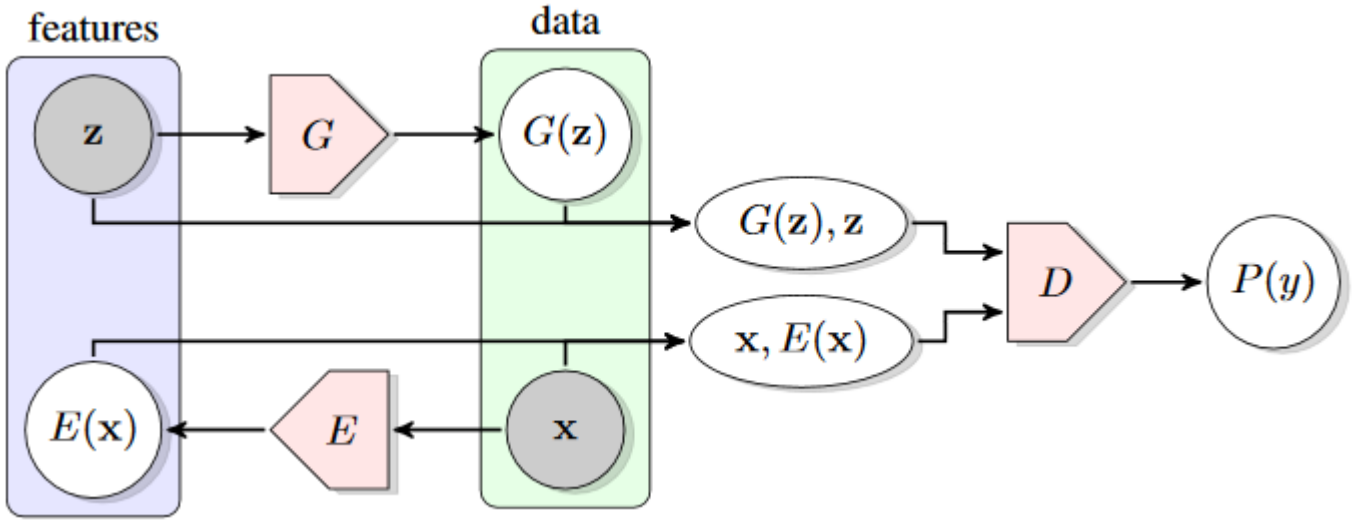
# SEIGAN

## [Ostyakov et al., 2018]



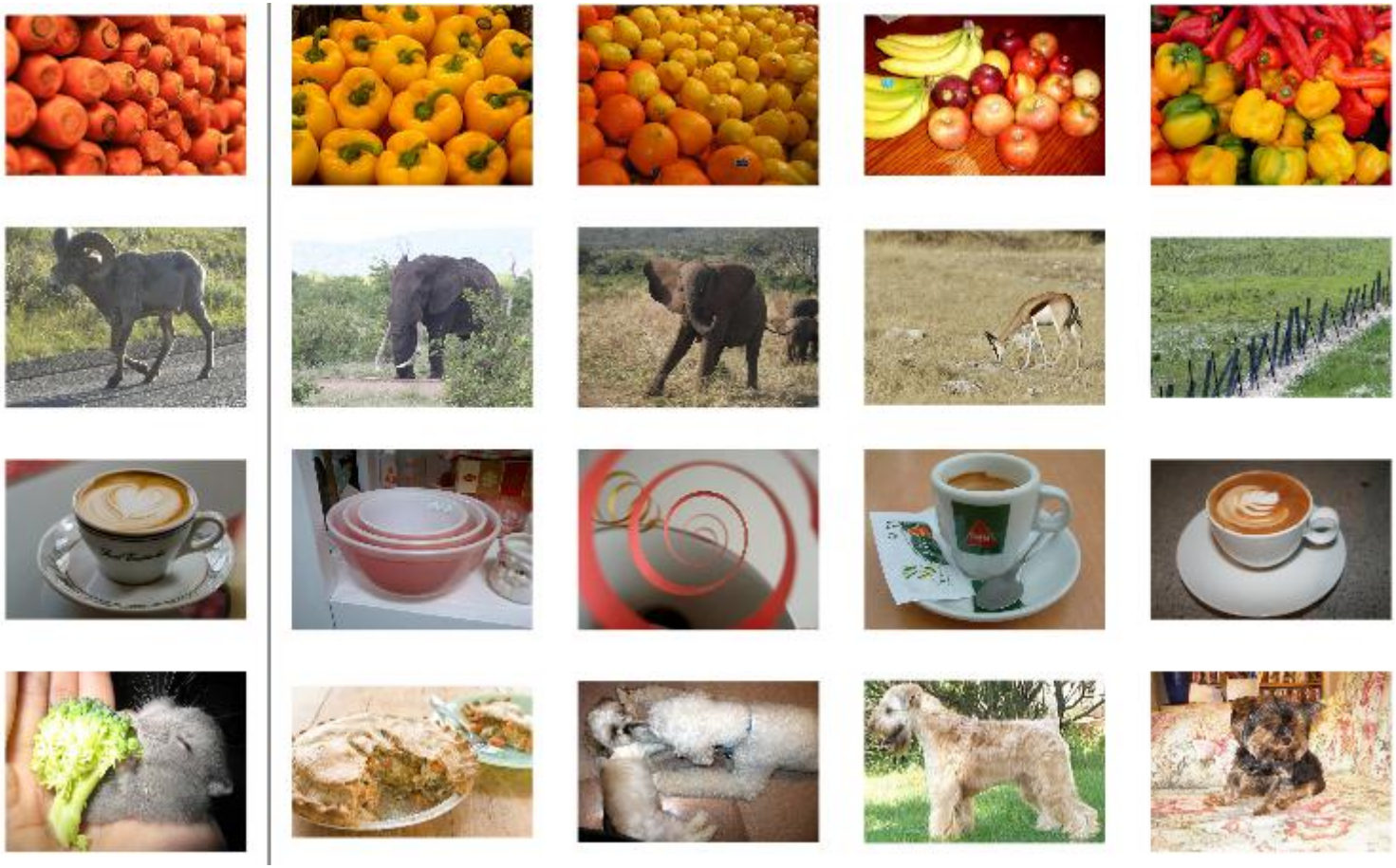
# BiGAN (Bidirectional)

**Учим одновременно и обратную функцию к генератору**
**Получается хорошая арифметика в латентном пространстве**



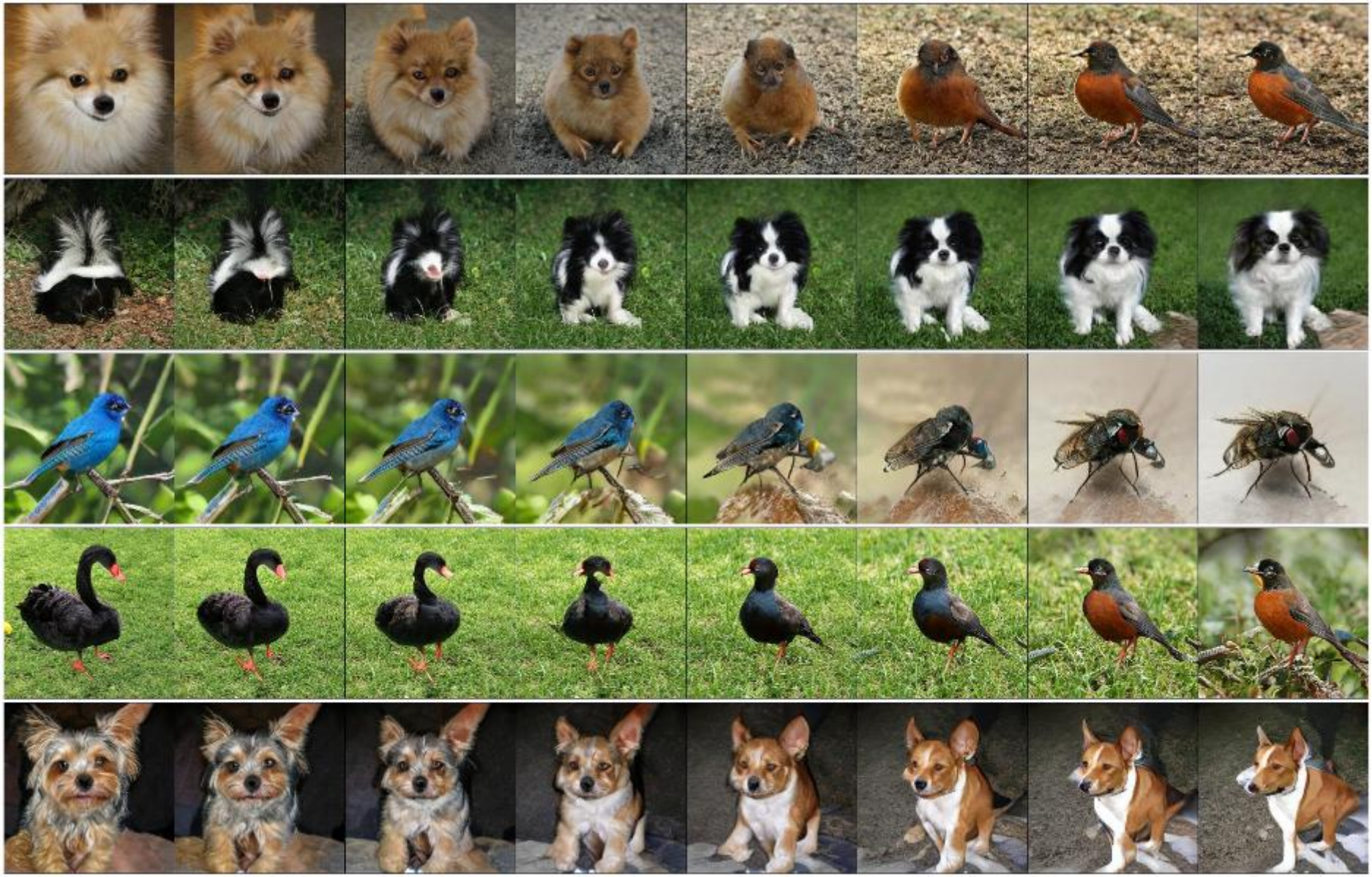

Jeff Donahue, Philipp Krähenbühl, Trevor Darrell «Adversarial Feature Learning»
https://arxiv.org/abs/1605.09782

# BiGAN (Bidirectional)



**с помощью BiGAN поиск ближайших соседей в признаковом пространстве**

# BigGAN: Генерация изображений / интерполяция (2018)



https://arxiv.org/pdf/1809.11096.pdf

# SAGAN (Self-Attention Generative Adversarial Networks)

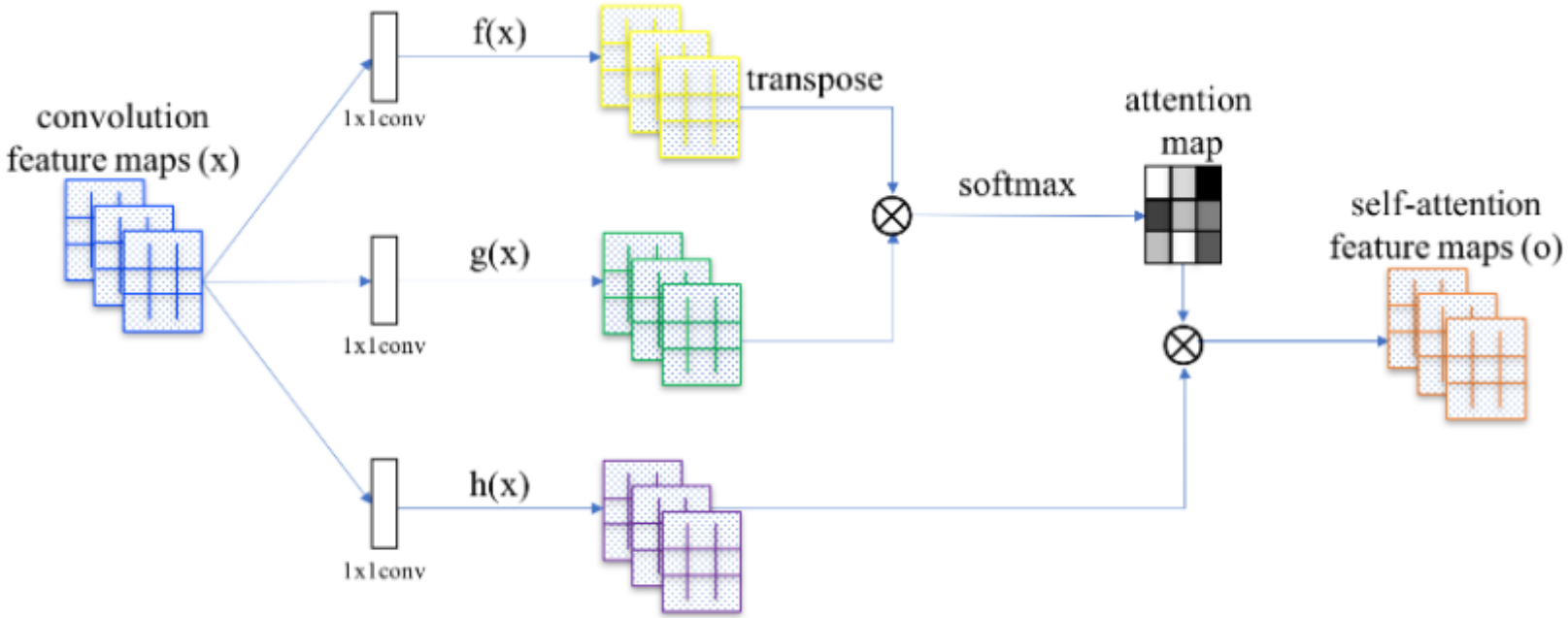



Figure 2: The proposed self-attention mechanism. The ⊗ denotes matrix multiplication. The softmax operation is performed on each row.

**внимание + GAN**

Han Zhang, Ian Goodfellow, Dimitris Metaxas, Augustus Odena «Self-Attention Generative Adversarial Networks», 2018 https://arxiv.org/abs/1805.08318

# SAGAN (Self-Attention Generative Adversarial Networks)

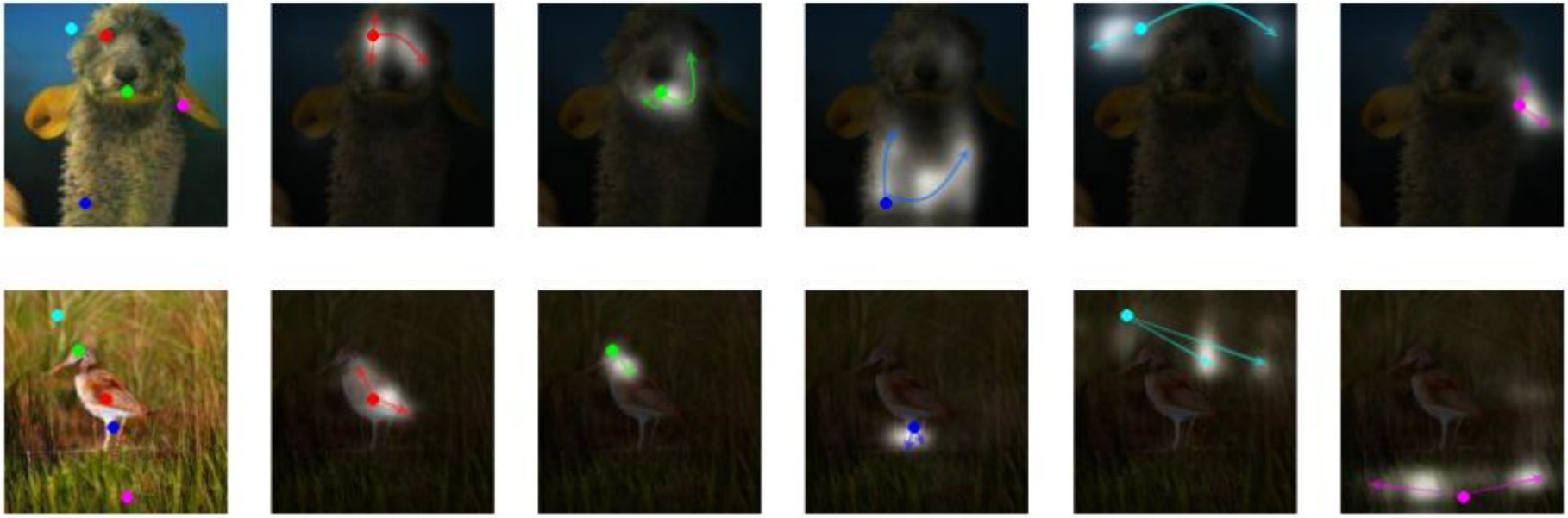
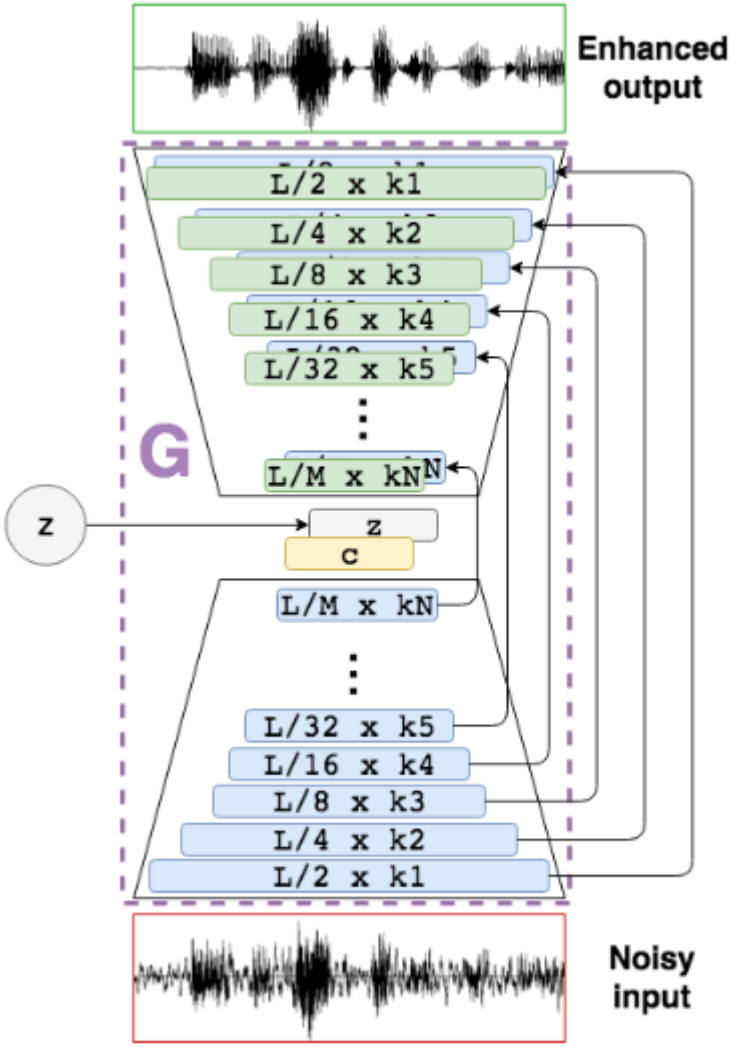


Figure 1: The proposed SAGAN generates images by leveraging complementary features in distant portions of the image rather than local regions of fixed shape to generate consistent objects/scenarios. In each row, the first image shows five representative query locations with color coded dots. The other five images are attention maps for those query locations, with corresponding color coded arrows summarizing the most-attended regions.

# SEGAN: Speech Enhancement Generative Adversarial Network



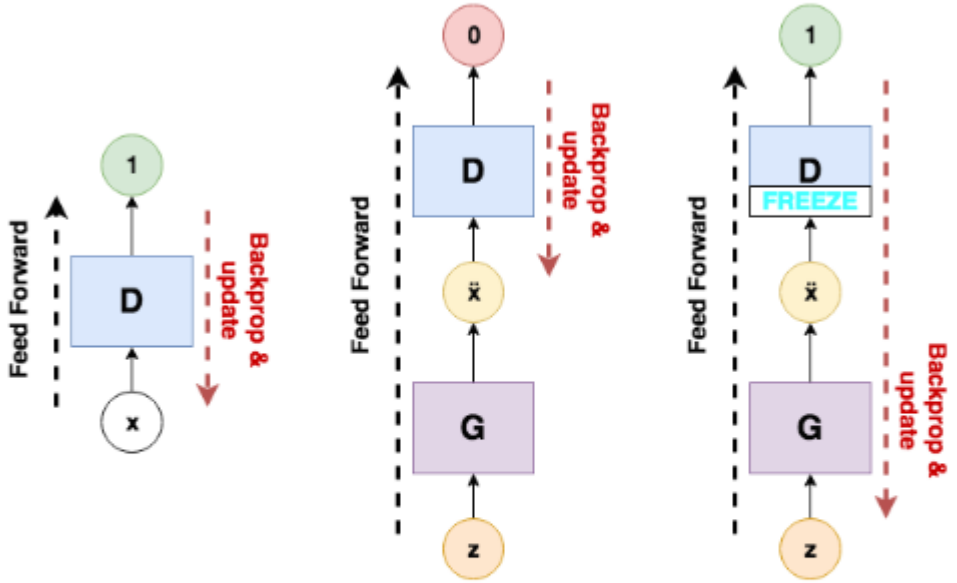




Figure 1: *GAN training process. First, D back-props a batch of real examples. Then, D back-props a batch of fake examples that come from G, and classifies them as fake. Finally, D's parameters are frozen and G back-props to make D misclassify.*

Santiago Pascual, Antonio Bonafonte, Joan Serrà SEGAN: Speech Enhancement Generative Adversarial Network, 2017 // https://arxiv.org/abs/1703.09452

# Различные виды GAN

**перечень видов со ссылками на первоисточники**

https://github.com/wiseodd/generative-models

## PixelRNN и PixelCNN итоги

- явное задание плотности
- оптимизация правдоподобия
- неэффективная последовательная генерация

## VAE итоги

- оптимизация вариационной нижней оценки правдоподобия
- полезное представление через скрытые переменные
- сейчас – не очень хороши

## GAN итоги

- игровой подход
- есть трюки в обучении
- сейчас – лучшие